Porting Guide (Open Source Components)

Issue 25

Date 2023-06-07

HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions

and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Meteorology....1
1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)....2
1.1.1 Introduction....2
1.1.2 Environment Requirements....3
1.1.3 Paths for Software Porting....4
1.1.4 Configuring the Compilation Environment....4
1.1.4.1 Configuration Description....5
1.1.4.2 Setting Up the Basic Environment....5
1.1.4.3 Installing HDF5....5
1.1.4.4 Installing PnetCDF....6
1.1.4.5 Installing NetCDF-C....6
1.1.4.6 Installing NetCDF-Fortran....7
1.1.4.7 Installing OPTIMIZED-ROUTINES....7
1.1.4.8 Installing ClusterShell....8
1.1.5 Obtaining Source Code....8
1.1.6 Compilation and Installation....8
1.1.7 Running and Verification....10
1.1.8 Change History....13
1.2 CAMx 6.50 Porting Guide (CentOS 7.6)....13
1.2.1 Introduction....14
1.2.2 Environment Requirements....14
1.2.3 Paths for Software Porting....15
1.2.4 Configuring the Compilation Environment....15
1.2.5 Obtaining the Source Code....16
1.2.6 Compilation and Installation....16
1.2.7 Running and Verification....17
1.2.8 More Information....18
1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)....18
1.3.1 Introduction....18
1.3.2 Environment Requirements....19
1.3.3 Paths for Software Porting....20
1.3.4 Configuring the Compilation Environment....20
1.3.4.1 Configuration Description....20

 Copyright © Huawei Technologies Co., Ltd.

1.3.4.2 Installing OpenBLAS....21
1.3.4.3 Installing HDF5....22
1.3.4.4 Installing PnetCDF....22
1.3.4.5 Installing NetCDF....23
1.3.4.6 Installing NetCDF-Fortran....23
1.3.5 Obtaining the Source Code....24
1.3.6 Compiling and Installing CESM....24
1.3.7 Running and Verifying CESM....25
1.3.8 More Information....26
1.4 NEMO V3.6 Porting Guide (CentOS 7.6)....26
1.4.1 Introduction....26
1.4.2 Environment Requirements....27
1.4.3 Paths for Software Porting....28
1.4.4 Configuring the Compilation Environment....28
1.4.4.1 Configuration Description....29
1.4.4.2 Installing PnetCDF....29
1.4.4.3 Installing HDF5....30
1.4.4.4 Installing NetCDF-C....30
1.4.4.5 Installing NetCDF-Fortran....31
1.4.4.6 Installing Blitz....31
1.4.4.7 Installing XIOS....32
1.4.5 Obtaining the Source Code....34
1.4.6 Compilation and Installation....35
1.4.7 Running and Verification....36
1.5 ROMS 3.6 Porting Guide (CentOS 7.6)....38
1.5.1 Introduction....38
1.5.2 Environment Requirements....38
1.5.3 Paths for Software Porting....39
1.5.4 Configuring the Compilation Environment....40
1.5.4.1 Configuration Description....40
1.5.4.2 Creating an Environment Variable File....40
1.5.4.3 Installing HDF5....41
1.5.4.4 Installing PnetCDF....41
1.5.4.5 Installing NetCDF-C....42
1.5.4.6 Installing NetCDF-Fortran....42
1.5.5 Obtaining the Source Code....43
1.5.6 Compilation and Installation....43
1.5.7 Running and Verification....45
1.5.8 Troubleshooting....46
1.5.9 More Information....47
1.6 WAVEWATCH III 6.07.1 Porting Guide (CentOS 7.6)....47
1.6.1 Introduction....47

 Copyright © Huawei Technologies Co., Ltd.

1.6.2 Environment Requirements....47
1.6.3 Paths for Software Porting....48
1.6.4 Configuring the Compilation Environment....49
1.6.4.1 Configuration Description....49
1.6.4.2 Installing HDF5....50
1.6.4.3 Installing NetCDF....50
1.6.4.4 Installing NetCDF-Fortran....51
1.6.4.5 Installing ParMETIS....51
1.6.5 Obtaining the Source Code....52
1.6.6 Compilation and Installation....52
1.6.7 Running and Verification....54
1.6.8 Support and Other Resources....55
1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)....55
1.7.1 Introduction....55
1.7.2 Environment Requirements....56
1.7.3 Path for Software Porting....56
1.7.4 Configuring the Compilation Environment....57
1.7.5 Obtaining the Source Code....57
1.7.6 Compilation and Installation....58
1.7.7 Running and Verification....59
1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)....60
1.8.1 Introduction....60
1.8.2 Environment Requirements....61
1.8.3 Paths for Software Porting....62
1.8.4 Configuring the Compilation Environment....62
1.8.4.1 Configuration Description....62
1.8.4.2 Installing HDF5....63
1.8.4.3 Installing NetCDF-C....63
1.8.4.4 Installing NetCDF-Fortran....64
1.8.5 Obtaining the Source Code....64
1.8.6 Compilation and Installation....65
1.8.7 Running and Verifying FVCOM....68
1.8.7.1 Running and Verifying the Estuary Case....68
1.8.7.2 Running and Verifying the Inlet Case....68
1.8.8 More Information....70
1.9 MITgcm c67o Porting Guide (CentOS 7.6)....70
1.9.1 Introduction....70
1.9.2 Environment Requirements....71
1.9.3 Paths for Software Porting....71
1.9.4 Configuring the Compilation Environment....72
1.9.4.1 Configuration Description....72
1.9.4.2 Installing HDF5....73

 Copyright © Huawei Technologies Co., Ltd.

1.9.4.3 Installing NetCDF-C....73
1.9.4.4 Installing NetCDF-Fortran....74
1.9.5 Obtaining the Source Code....74
1.9.6 Compilation and Installation....74
1.9.7 Running and Verification....76
1.9.8 More Resources....77
1.10 POM POM2K Porting Guide (CentOS 7.6)....77
1.10.1 Introduction....77
1.10.2 Environment Requirements....78
1.10.3 Paths for Software Porting....79
1.10.4 Configuring the Compilation Environment....79
1.10.4.1 Configuration Description....79
1.10.4.2 Installing NetCDF-C....80
1.10.5 Obtaining the Source Code....80
1.10.6 POM2K Model Running....81
1.10.7 More Information....83
1.11 MOM 6 Porting Guide (CentOS 7.6)....83
1.11.1 Introduction....83
1.11.2 Environment Requirements....83
1.11.3 Paths for Software Porting....84
1.11.4 Configuring the Compilation Environment....85
1.11.4.1 Configuration Description....85
1.11.4.2 Installing PnetCDF....86
1.11.4.3 Installing HDF5....86
1.11.4.4 Installing NetCDF-C....87
1.11.4.5 Installing NetCDF-Fortran....87
1.11.5 Obtaining the Source Code....88
1.11.6 Compilation and Installation....88
1.11.7 Running and Verification....89
1.11.8 Troubleshooting....89
1.11.9 More Information....90
1.12 SMOKE 4.7 Porting Guide (CentOS 7.6)....90
1.12.1 Introduction....90
1.12.2 Environment Requirements....90
1.12.3 Paths for Software Porting....92
1.12.4 Configuring the Compilation Environment....92
1.12.4.1 Configuration Description....92
1.12.4.2 Installing HDF5....93
1.12.4.3 Installing PnetCDF....94
1.12.4.4 Installing NetCDF....94
1.12.4.5 Installing NetCDF-Fortran....95
1.12.5 Obtaining the Source Code....95

 Copyright © Huawei Technologies Co., Ltd.

1.12.6 Compiling and Installing SMOKE 95
1.12.6.1 Installing the I/O API 96
1.12.6.2 Installing SMOKE 97
1.12.7 Running and Verifying SMOKE 98
1.12.8 Troubleshooting 99
1.12.9 More Information 99
1.13 SPECFEM3D GLOBE 7.0.0 Porting Guide (CentOS 7.6) 99
1.13.1 Introduction 100
1.13.2 Environment Requirements 100
1.13.3 Paths for Software Porting 101
1.13.4 Configuring the Compilation Environment 101
1.13.5 Obtaining the Source Code 102
1.13.6 Compiling and Installing SPECFEM3D GLOBE 102
1.13.7 Running and Verifying SPECFEM3D GLOBE 103
1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6) 103
1.14.1 Introduction 103
1.14.2 Environment Requirements 104
1.14.3 Paths for Software Porting 105
1.14.4 Configuring the Compilation Environment 106
1.14.4.1 Configuration Description 106
1.14.4.2 Installing HDF5 106
1.14.4.3 Installing NetCDF-C 107
1.14.4.4 Installing NetCDF-Fortran 107
1.14.4.5 Installing OPTIMIZED-ROUTINES 108
1.14.5 Obtaining the Source Code 108
1.14.6 Compilation and Installation 109
1.14.7 Running and Verification 111
1.14.8 Troubleshooting 112
1.15 SWAN 41.31 Porting Guide (CentOS 7.6) 112
1.15.1 Introduction 112
1.15.2 Environment Requirements 113
1.15.3 Paths for Software Porting 114
1.15.4 Configuring the Compilation Environment 115
1.15.4.1 Configuration Description 115
1.15.4.2 Installing HDF5 115
1.15.4.3 Installing PnetCDF 116
1.15.4.4 Installing NetCDF-C 117
1.15.4.5 Installing NetCDF-Fortran 117
1.15.5 Obtaining the Source Code 118
1.15.6 Compiling and Installing SWAN 118
1.15.7 Running and Verifying SWAN 119
1.15.8 More Information 120

 Copyright © Huawei Technologies Co., Ltd.

1.16 CDO 1.9.8 Porting Guide (CentOS 7.6) 120
1.16.1 Introduction 120
1.16.2 Environment Requirements 121
1.16.3 Paths for Software Porting 122
1.16.4 Configuring the Compilation Environment 122
1.16.4.1 Configuration Description 122
1.16.4.2 Setting Up the Basic Environment 123
1.16.4.3 Configuring the Huawei Yum Source 123
1.16.4.4 Installing the Dependencies 123
1.16.5 Obtaining Source Code 124
1.16.6 Compilation and Installation 124
1.16.7 Running and Verification 124
1.16.8 More Resources 125
1.17 ecCodes 2.18.0 Porting Guide (CentOS 7.6) 125
1.17.1 Introduction 125
1.17.2 Environment Requirements 126
1.17.3 Paths for Software Porting 126
1.17.4 Configuring the Compilation Environment 127
1.17.4.1 Configuration Description 127
1.17.4.2 Installing CMake 127
1.17.5 Obtaining Source Code 128
1.17.6 Compilation and Installation 128
1.17.7 Running and Verification 129
1.17.8 More Resources 129
1.18 ecFlow 5.5.2 Porting Guide (CentOS 7.6) 129
1.18.1 Introduction 129
1.18.2 Environment Requirements 130
1.18.3 Paths for Software Porting 131
1.18.4 Configuring the Compilation Environment 131
1.18.4.1 Configuration Description 131
1.18.4.2 Installing CMake 132
1.18.5 Obtaining Source Code 132
1.18.6 Compilation and Installation 133
1.18.7 Running and Verifying ecFlow 133
1.18.8 More Resources 134
1.19 NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6) 134
1.19.1 Introduction 134
1.19.2 Environment Requirements 135
1.19.3 Paths for Software Porting 136
1.19.4 Configuring the Compilation Environment 136
1.19.4.1 Configuration Description 136
1.19.4.2 Installing HDF5 137

 Copyright © Huawei Technologies Co., Ltd.

1.19.4.3 Installing PnetCDF 138
1.19.4.4 Installing NetCDF 138
1.19.4.5 Installing NetCDF-Fortran 139
1.19.4.6 Installing CMake 139
1.19.5 Obtaining Source Code 139
1.19.6 Compilation and Installation 140
1.19.7 Running and Verification 140
1.19.8 Troubleshooting 141
1.19.9 More Resources 142
1.20 PyFerret 7.6.0 Porting Guide (CentOS 7.6) 142
1.20.1 Introduction 142
1.20.2 Environment Requirements 142
1.20.3 Paths for Software Porting 143
1.20.4 Configuring the Compilation Environment 144
1.20.4.1 Configuration Description 144
1.20.4.2 Setting Up the Basic Environment 145
1.20.4.3 Configuring the Yum Source and Installing the Dependencies 145
1.20.4.4 Installing HDF5 145
1.20.4.5 Installing NetCDF 146
1.20.4.6 Installing NetCDF-Fortran 146
1.20.5 Obtaining Source Code 147
1.20.6 Compilation and Installation 147
1.20.7 Running and Verification 149
1.20.8 More Resources 150
1.21 NCL 6.3.0 Porting Guide (CentOS 7.6) 150
1.21.1 Introduction 150
1.21.2 Environment Requirements 151
1.21.3 Paths for Software Porting 151
1.21.4 Configuring the Compilation Environment 152
1.21.5 Compilation and Installation 152
1.21.6 Running and Verification 155
1.22 ARPS 5.3.4 Porting Guide (CentOS 7.6) 155
1.22.1 Introduction 155
1.22.2 Environment Requirements 156
1.22.3 Paths for Software Porting 156
1.22.4 Configuring the Compilation Environment 157
1.22.4.1 Configuration Description 157
1.22.4.2 Installing HDF4 158
1.22.5 Obtaining the Source Code 158
1.22.6 Compilation and Installation 158
1.22.7 Running and Verification 160
1.23 CAMx 7.10 Porting Guide (openEuler 21.03) 160

 Copyright © Huawei Technologies Co., Ltd.

1.23.1 Introduction 160
1.23.2 Environment Requirements 161
1.23.3 Paths for Software Porting 162
1.23.4 Configuring the Compilation Environment 163
1.23.4.1 Configuration Description 163
1.23.4.2 Installing HDF5 164
1.23.4.3 Installing NetCDF-C 165
1.23.4.4 Installing NetCDF-Fortran 165
1.23.5 Obtaining the Source Code 166
1.23.6 Compilation and Installation 166
1.23.7 Running and Verification 167
1.24 CDO 1.9.8 Porting Guide (openEuler 20.03) 168
1.24.1 Introduction 168
1.24.2 Environment Requirements 168
1.24.3 Paths for Software Porting 170
1.24.4 Configuring the Compilation Environment 170
1.24.4.1 Configuration Description 170
1.24.4.2 Configuring the Yum Source 171
1.24.4.3 Installing the Dependencies 172
1.24.5 Obtaining the Source Code 172
1.24.6 Compilation and Installation 172
1.24.7 Running and Verification 173
1.24.8 Troubleshooting 174
1.24.8.1 Failed to Install an RPM Package Online Due to Network Problems 174
1.24.8.2 Error Reported When Installing Dependency Packages with Yum 175
1.25 ecCodes 2.18.0 Porting Guide (openEuler 20.03) 175
1.25.1 Introduction 176
1.25.2 Environment Requirements 176
1.25.3 Paths for Software Porting 177
1.25.4 Configuring the Compilation Environment 178
1.25.4.1 Configuration Description 178
1.25.4.2 Configuring the Yum Source 178
1.25.4.3 Installing CMake 179
1.25.5 Obtaining the Source Code 181
1.25.6 Compilation and Installation 181
1.25.7 Running and Verification 184
1.26 ecFlow 5.5.2 Porting Guide (openEuler 20.03) 185
1.26.1 Introduction 185
1.26.2 Environment Requirements 186
1.26.3 Paths for Software Porting 187
1.26.4 Configuring the Compilation Environment 187
1.26.4.1 Configuration Description 187

 Copyright © Huawei Technologies Co., Ltd.

1.26.4.2 Configuring the Yum Source ... 188
1.26.4.3 Installing CMake ... 189
1.26.5 Obtaining the Source Code ... 190
1.26.6 Compilation and Installation ... 191
1.26.7 Running and Verification ... 194
1.27 MITgcm c67o Porting Guide (openEuler 21.03) ... 196
1.27.1 Introduction ... 196
1.27.2 Environment Requirements ... 196
1.27.3 Paths for Software Porting ... 197
1.27.4 Configuring the Compilation Environment ... 198
1.27.4.1 Configuration Description ... 198
1.27.4.2 Installing HDF5 ... 199
1.27.4.3 Installing NetCDF-C ... 200
1.27.4.4 Installing NetCDF-Fortran ... 200
1.27.5 Obtaining the Source Code ... 201
1.27.6 Compilation and Installation ... 201
1.27.7 Running and Verification ... 202
1.27.8 Troubleshooting ... 203
1.27.8.1 After a Dependency Package Is Configured, an Error Message Is Displayed Indicating that the Dynamic Link Library Cannot Be Found ... 203
1.28 NCEPLIBS 1.2.0 Porting Guide (openEuler 20.03) ... 204
1.28.1 Introduction ... 204
1.28.2 Environment Requirements ... 204
1.28.3 Paths for Software Porting ... 205
1.28.4 Configuring the Compilation Environment ... 206
1.28.4.1 Configuration Description ... 206
1.28.4.2 Configuring the Yum Source ... 207
1.28.4.3 Installing HDF5 ... 208
1.28.4.4 Installing PnetCDF ... 210
1.28.4.5 Installing NetCDF-C ... 212
1.28.4.6 Installing NetCDF-Fortran ... 214
1.28.4.7 Installing CMake ... 216
1.28.5 Obtaining the Source Code ... 218
1.28.6 Compilation and Installation ... 218
1.28.7 Running and Verification ... 220
1.28.8 Troubleshooting ... 221
1.28.8.1 Error Reported When Compiling the sigio Module ... 221
1.28.8.2 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the w3emc Module ... 222
1.28.8.3 Error "F90-F-0004-Unable to open MODULE file sigio_module.mod" Reported When Compiling the w3emc Module ... 223
1.28.8.4 Error Reported When Compiling the gfsio Module ... 224
1.28.8.5 Error Reported When Compiling the sfcio Module ... 225

 Copyright © Huawei Technologies Co., Ltd.

1.28.8.6 Error Reported When Compiling the g2 Module 226
1.28.8.7 Error Reported When Compiling the nemsiogfs Module 227
1.28.8.8 Error Reported When Compiling the landsfcutil Module 228
1.28.8.9 Error "Failed to clone repository: 'git://github.com/noaa-emc/emc_post'" Reported When Compiling the nceppost Module 229
1.28.8.10 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the nceppost Module 230
1.28.8.11 Errors "F90-S-0021-Label field of continuation line is not blank" and "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module 231
1.28.8.12 Error "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module 232
1.28.8.13 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wrf_io Module 233
1.28.8.14 Error "F90-S-0017-Unable to open include file: netcdf.inc" Reported When Compiling the wrf_io Module 234
1.28.8.15 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wgrib2 Module 235
1.28.8.16 Errors "Could NOT find JPEG" and "Could NOT find Jasper" Reported When Compiling the wgrib2 Module 235
1.29 SWAN 41.31 Porting Guide (openEuler 20.03) 236
1.29.1 Introduction 237
1.29.2 Environment Requirements 237
1.29.3 Paths for Software Porting 238
1.29.4 Configuring the Compilation Environment 239
1.29.4.1 Configuration Description 239
1.29.4.2 Configuring the Yum Source 240
1.29.4.3 Installing HDF5 240
1.29.4.4 Installing PnetCDF 243
1.29.4.5 Installing NetCDF-C 245
1.29.4.6 Installing NetCDF-Fortran 247
1.29.5 Obtaining the Source Code 249
1.29.6 Compilation and Installation 250
1.29.7 Running and Verification 251
1.30 WAVEWATCH III 6.07.1 Porting Guide (openEuler 21.03) 253
1.30.1 Introduction 253
1.30.2 Environment Requirements 254
1.30.3 Paths for Software Porting 255
1.30.4 Configuring the Compilation Environment 255
1.30.4.1 Configuration Description 256
1.30.4.2 Installing HDF5 257
1.30.4.3 Installing NetCDF-C 257
1.30.4.4 Installing NetCDF-Fortran 258
1.30.4.5 Installing ParMETIS 259
1.30.5 Obtaining the Source Code 259
1.30.6 Compilation and Installation 260

 Copyright © Huawei Technologies Co., Ltd.

1.30.7 Running and Verification 261
1.31 WRF 4.2 Porting Guide (openEuler 21.03) 264
1.31.1 Introduction 264
1.31.2 Environment Requirements 264
1.31.3 Paths for Software Porting 265
1.31.4 Configuring the Compilation Environment 266
1.31.4.1 Configuration Description 266
1.31.4.2 Installing zlib 267
1.31.4.3 Installing PnetCDF 267
1.31.4.4 Installing HDF5 268
1.31.4.5 Installing NetCDF-C 268
1.31.4.6 Installing NetCDF-Fortran 269
1.31.5 Obtaining the Source Code 270
1.31.6 Compilation and Installation 270
1.31.7 Running and Verification 272
1.32 CESM 2.1.1 Porting Guide (openEuler 21.03) 273
1.32.1 Introduction 273
1.32.2 Environment Requirements 273
1.32.3 Paths for Software Porting 275
1.32.4 Configuring the Compilation Environment 275
1.32.4.1 Configuration Description 276
1.32.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables 276
1.32.4.3 Installing zlib 277
1.32.4.4 Installing szip 277
1.32.4.5 Installing curl 278
1.32.4.6 Installing HDF5 278
1.32.4.7 Installing PnetCDF 279
1.32.4.8 Installing NetCDF-C 279
1.32.4.9 Installing NetCDF-Fortran 280
1.32.4.10 Installing OpenBLAS 280
1.32.5 Obtaining the Source Code 281
1.32.6 Compilation and Installation 281
1.32.7 Running and Verification 282
1.33 CESM 2.1.3 Porting Guide (openEuler 21.03) 297
1.33.1 Introduction 297
1.33.2 Environment Requirements 298
1.33.3 Paths for Software Porting 299
1.33.4 Configuring the Compilation Environment 300
1.33.4.1 Configuration Description 300
1.33.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables 300
1.33.4.3 Installing OpenBLAS 301
1.33.4.4 Installing HDF5 301

 Copyright © Huawei Technologies Co., Ltd.

1.33.4.5 Installing NetCDF-C 302
1.33.4.6 Installing NetCDF-Fortran 304
1.33.5 Obtaining the Source Code 307
1.33.6 Compilation and Installation 307
1.33.7 Running and Verification 308
1.33.8 More Resources 325
1.34 NEMO 3.6 Porting Guide (openEuler 21.03) 325
1.34.1 Introduction 325
1.34.2 Environment Requirements 325
1.34.3 Paths for Software Porting 326
1.34.4 Configuring the Compilation Environment 327
1.34.4.1 Configuration Description 327
1.34.4.2 Installing PnetCDF 328
1.34.4.3 Installing HDF5 329
1.34.4.4 Installing NetCDF 329
1.34.4.5 Installing NetCDF-Fortran 330
1.34.4.6 Installing Blitz 330
1.34.4.7 Installing XIOS 331
1.34.5 Obtaining the Source Code 333
1.34.6 Compilation and Installation 333
1.34.7 Running and Verification 335
1.35 ROMS 4.0 Porting Guide (openEuler 21.03) 336
1.35.1 Introduction 336
1.35.2 Environment Requirements 336
1.35.3 Paths for Software Porting 338
1.35.4 Configuring the Compilation Environment 338
1.35.4.1 Configuration Description 338
1.35.4.2 Installing HDF5 339
1.35.4.3 Installing NetCDF-C 340
1.35.4.4 Installing NetCDF-Fortran 340
1.35.4.5 Installing PnetCDF 341
1.35.5 Obtaining the Source Code 341
1.35.6 Compilation and Installation 341
1.35.7 Running and Verification 347
1.36 ncl 6.3.0 Porting Guide (openEuler 20.03) 347
1.36.1 Introduction 347
1.36.2 Environment Requirements 348
1.36.3 Paths for Software Porting 349
1.36.4 Configuring the Compilation Environment 350
1.36.4.1 Configuration Description 350
1.36.4.2 Installing the BiSheng Compiler 350
1.36.4.3 Installing HDF5 351

 Copyright © Huawei Technologies Co., Ltd.

1.36.4.4 Installing NetCDF-C 353
1.36.4.5 Installing NetCDF-Fortran 355
1.36.4.6 Installing HDF 357
1.36.4.7 Installing GDAL 359
1.36.4.8 Installing Jasper 361
1.36.4.9 Installing G2Clib 362
1.36.4.10 Installing Udunits2 363
1.36.5 Obtaining the Source Code 365
1.36.6 Compilation and Installation 365
1.36.7 Running and Verification 370
1.36.8 Troubleshooting 370
1.36.8.1 Error Reported When Packaging **rpmbuild -ba ncl.spec** 371
1.37 PyFerret 7.6.0 Porting Guide (openEuler 21.03) 371
1.37.1 Introduction 371
1.37.2 Environment Requirements 372
1.37.3 Paths for Software Porting 373
1.37.4 Configuring the Compilation Environment 373
1.37.4.1 Configuration Description 373
1.37.4.2 Configuring the Local Yum Source 374
1.37.4.3 Installing HDF5 375
1.37.4.4 Installing NetCDF-C 376
1.37.4.5 Installing NetCDF-Fortran 377
1.37.5 Obtaining the Source Code 379
1.37.6 Compilation and Installation 380
1.37.7 Running and Verification 381
1.37.8 Troubleshooting 383
1.37.8.1 Error "fatal error: 'hb.h' file not found" Reported During the Compilation 383
1.37.8.2 Error "cannot connect to X server" Reported During the Verification 384
1.38 SMOKE 4.7 Porting Guide (openEuler 21.03) 385
1.38.1 Introduction 385
1.38.2 Environment Requirements 386
1.38.3 Paths for Software Porting 387
1.38.4 Configuring the Compilation Environment 388
1.38.4.1 Configuration Description 388
1.38.4.2 Installing HDF5 389
1.38.4.3 Installing NetCDF-C 389
1.38.4.4 Installing NetCDF-Fortran 390
1.38.4.5 Installing PnetCDF 391
1.38.5 Obtaining the Source Code 391
1.38.6 Compilation and Installation 391
1.38.7 Running and Verification 394
1.39 SPECFEM3D GLOBE 7.0.0 Porting Guide (openEuler 21.03) 394

 Copyright © Huawei Technologies Co., Ltd.

1.39.1 Introduction 395
1.39.2 Environment Requirements 395
1.39.3 Paths for Software Porting 396
1.39.4 Configuring the Compilation Environment 397
1.39.5 Obtaining the Source Code 397
1.39.6 Compilation and Installation 398
1.39.7 Running and Verification 398
1.40 CMAQ 5.3.1 Porting Guide (openEuler 21.03) 399
1.40.1 Introduction 399
1.40.2 Environment Requirements 400
1.40.3 Paths for Software Porting 401
1.40.4 Configuring the Compilation Environment 402
1.40.4.1 Configuration Description 402
1.40.4.2 Installing HDF5 403
1.40.4.3 Installing NetCDF-C 403
1.40.4.4 Installing NetCDF-Fortran 404
1.40.4.5 Installing OPTIMIZED-ROUTINES 404
1.40.5 Obtaining the Source Code 405
1.40.6 Compilation and Installation 405
1.40.7 Running and Verification 409
1.40.8 Troubleshooting 410
1.40.8.1 Error Reported When Executing the make Command During the OPTIMIZED-ROUTINES Installation 410
1.40.8.2 Error Reported When Running and Verifying a CMAQ Test 411
1.41 POM POM2K Porting Guide (openEuler 21.03) 411
1.41.1 Introduction 411
1.41.2 Environment Requirements 412
1.41.3 Paths for Software Porting 412
1.41.4 Configuring the Compilation Environment 413
1.41.4.1 Configuration Description 413
1.41.4.2 Installing NetCDF-C 414
1.41.5 Obtaining the Source Code 414
1.41.6 Generating an IC Data File 415
1.41.7 Compilation and Installation 416
1.42 FVCOM 4.1 Porting Guide (openEuler 21.03) 417
1.42.1 Introduction 417
1.42.2 Environment Requirements 418
1.42.3 Paths for software porting 419
1.42.4 Configuring the Compilation Environment 420
1.42.4.1 Configuration Description 420
1.42.4.2 Installing HDF5 421
1.42.4.3 Installing NetCDF-C 421
1.42.4.4 Installing NetCDF-Fortran 422

 Copyright © Huawei Technologies Co., Ltd.

1.42.5 Obtaining the Source Code 423
1.42.6 Compilation and Installation 423
1.42.7 Running and Verification 426
1.42.8 Troubleshooting 427
1.42.8.1 Programs Not Supported by the BiSheng Compiler 427
1.42.8.2 isnan Function Reports an Error 427
1.42.8.3 Version Conflict 427
1.43 gadget2 2.3.5 Porting Guide (Kylin V10) 428
1.43.1 Introduction 428
1.43.2 Environment Requirements 428
1.43.3 Paths for Software Porting 429
1.43.4 Configuring the Compilation Environment 430
1.43.5 Obtaining the Source Code 430
1.43.6 Compilation and Installation 431
1.43.7 Running and Verification 431
1.44 GrADS 2.2.1 Porting Guide (CentOS 8.2) 431
1.44.1 Introduction 431
1.44.2 Environment Requirements 432
1.44.3 Paths for Software Porting 433
1.44.4 Configuring the Compilation Environment 433
1.44.4.1 Configuration Description 433
1.44.4.2 Installing Dependencies 434
1.44.4.3 Installing libgd 434
1.44.4.4 Installing libsx 435
1.44.5 Obtaining the Source Code 435
1.44.6 Compilation and Installation 435
1.44.7 Running and Verification 436
1.45 wgrib 1.8.2 Porting Guide (Kylin V10) 438
1.45.1 Introduction 438
1.45.2 Environment Requirements 438
1.45.3 Paths for Software Porting 439
1.45.4 Configuring the Compilation Environment 439
1.45.5 Obtaining the Source Code 440
1.45.6 Compilation and Installation 440
1.45.7 Running and Verification 440
1.45.8 Change History 441
1.46 wgrib2 3.1.1 Porting Guide (Kylin V10) 441
1.46.1 Introduction 441
1.46.2 Environment Requirements 442
1.46.3 Paths for Software Porting 442
1.46.4 Configuring the Compilation Environment 443
1.46.5 Obtaining the Source Code 443

 Copyright © Huawei Technologies Co., Ltd.

1.46.6 Compilation and Installation 443
1.46.7 Running and Verification 444
1.46.8 Change History 444
1.47 WPS 3.9 Porting Guide (CentOS 8.2) 444
1.47.1 Introduction 445
1.47.2 Environment Requirements 445
1.47.3 Paths for Software Porting 446
1.47.4 Configuring the Compilation Environment 446
1.47.4.1 Configuration Description 447
1.47.4.2 Installing HDF5 447
1.47.4.3 Installing PnetCDF 448
1.47.4.4 Installing NetCDF-C 449
1.47.4.5 Installing NetCDF-Fortran 449
1.47.5 Obtaining the Source Code 450
1.47.6 Compilation and Installation 450
1.47.6.1 Compiling and Installing WRF 450
1.47.6.2 Compiling and Installing WPS 452
1.47.7 Running and Verification 453
1.47.8 Change History 454

2 Manufacturing 455

2.1 OpenFOAM 1906 Porting Guide (CentOS 7.6) 455
2.1.1 Introduction 455
2.1.2 Environment Requirements 456
2.1.3 Paths for Software Porting 457
2.1.4 Configuring the Compilation Environment 457
2.1.4.1 Configuration Description 457
2.1.4.2 Installing ClusterShell 458
2.1.5 Obtaining the Source Code 458
2.1.6 Compiling and Installing OpenFOAM 458
2.1.7 Running and Verifying OpenFOAM 459
2.1.8 Troubleshooting 461
2.2 SU2 Porting Guide (CentOS 7.6) 464
2.2.1 Introduction 464
2.2.2 Environment Requirements 464
2.2.3 Paths for Software Porting 465
2.2.4 Configuring the Compilation Environment 465
2.2.4.1 Configuration Description 466
2.2.4.2 Installing HDF5 466
2.2.5 SU2 6.0.1 Porting Guide (CentOS 7.6) 466
2.2.5.1 Obtaining the Source Code 467
2.2.5.2 Compiling and Installing SU2 467
2.2.6 SU2 7.0.4 Porting Guide (CentOS 7.6) 467

 Copyright © Huawei Technologies Co., Ltd.

2.2.6.1 Obtaining the Source Code 468
2.2.6.2 Compiling and Installing SU2 468
2.2.7 Running and Verifying SU2 468
2.2.8 Troubleshooting 470
2.2.9 More Information 470
2.3 PALABOS 2.1r0 Porting Guide (CentOS 7.6) 470
2.3.1 Introduction 471
2.3.2 Environment Requirements 471
2.3.3 Paths for Software Porting 472
2.3.4 Configuring the Compilation Environment 472
2.3.4.1 Configuration Description 472
2.3.4.2 Installing Python 473
2.3.4.3 Installing ImageMagick 473
2.3.5 Obtaining the Source Code 474
2.3.6 Compiling and Installing Palabos 474
2.3.7 Running and Verifying Palabos 474
2.3.7.1 Running and Verifying the cavity2d Case 475
2.3.7.2 Running and Verifying the boussinesqThermal3d Case 475
2.3.8 Support and Other Resources 476
2.4 deal.II 9.1.1 Porting Guide (CentOS 7.6) 476
2.4.1 Introduction 476
2.4.2 Environment Requirements 477
2.4.3 Paths for Software Porting 478
2.4.4 Configuring the Compilation Environment 478
2.4.5 Obtaining the Source Code 479
2.4.6 Compiling and Installing deal.II 479
2.4.7 Running and Verifying deal.II 480
2.4.8 More Information 481
2.5 AMG 1.2 Porting Guide (CentOS 7.6) 481
2.5.1 Introduction 481
2.5.2 Environment Requirements 481
2.5.3 Paths for Software Porting 482
2.5.4 Configuring the Compilation Environment 483
2.5.5 Obtaining the Source Code 483
2.5.6 Compiling and Installing AMG 483
2.5.7 Running and Verifying AMG 484
2.6 Code_Saturne 6.0.2 Porting Guide (CentOS 7.6) 485
2.6.1 Introduction 485
2.6.2 Environment Requirements 485
2.6.3 Paths for Software Porting 486
2.6.4 Configuring the Compilation Environment 487
2.6.4.1 Configuration Description 487

 Copyright © Huawei Technologies Co., Ltd.

2.6.4.2 Installing the BLAS Dependency 488
2.6.4.3 Installing PyQt4 488
2.6.4.4 Installing HDF5 488
2.6.4.5 Installing MED 489
2.6.4.6 Installing SCOTCH 489
2.6.4.7 Installing METIS 490
2.6.5 Obtaining the Source Code 490
2.6.6 Compiling and Installing Code_Saturne 490
2.6.7 Running and Verifying Code_Saturne 491
2.6.7.1 Running and Verifying Code_Saturne on a Single Node 491
2.6.7.2 Running and Verifying Code_Saturne on Multiple Nodes 492
2.6.8 More Resources 494
2.7 FDS 6.7.4 Porting Guide (CentOS 7.6) 494
2.7.1 Introduction 494
2.7.2 Environment Requirements 494
2.7.3 Paths for Software Porting 495
2.7.4 Configuring the Compilation Environment 496
2.7.5 Obtaining the Source Code 496
2.7.6 Compiling and Installing FDS 496
2.7.7 Running and Verifying FDS 497
2.7.8 Troubleshooting 498
2.7.9 More Information 498
2.8 Elmer 8.4 Porting Guide (CentOS 7.6) 499
2.8.1 Introduction 499
2.8.2 Environment Requirements 499
2.8.3 Paths for Software Porting 500
2.8.4 Configuring the Compilation Environment 500
2.8.4.1 Configuration Description 501
2.8.4.2 Installing CMake 501
2.8.4.3 Installing OpenBLAS 502
2.8.5 Obtaining the Source Code 502
2.8.6 Compiling and Installing Elmer 502
2.8.7 Running and Verifying Elmer 503
2.8.8 Troubleshooting 504
2.8.9 More Information 504
2.9 Code_Aster 14.6.0 Porting Guide (CentOS 7.6) 504
2.9.1 Introduction 504
2.9.2 Environment Requirements 505
2.9.3 Paths for Software Porting 506
2.9.4 Configuring the Compilation Environment 506
2.9.4.1 Configuration Description 506
2.9.4.2 Configuring the Yum Source 507

 Copyright © Huawei Technologies Co., Ltd.

2.9.5 Obtaining the Source Code 508
2.9.6 Compilation and Installation 508
2.9.7 Running and Verification 512
2.9.8 More Resources 513
2.10 OpenFOAM 1906 Porting Guide (openEuler 21.03) 513
2.10.1 Introduction 513
2.10.2 Environment Requirements 514
2.10.3 Paths for Software Porting 515
2.10.4 Configuring the Compilation Environment 515
2.10.4.1 Configuration Description 515
2.10.4.2 Configuring the Installation Path 516
2.10.4.3 Installing the BiSheng Compiler 516
2.10.4.4 Installing Hyper MPI 517
2.10.4.5 Installing the Dependencies 518
2.10.5 Obtaining the Source Code 518
2.10.6 Compilation and Installation 518
2.10.7 Running and Verification 520
2.11 Code_Aster 14.6.0 Porting Guide (openEuler 21.03) 522
2.11.1 Introduction 522
2.11.2 Environment Requirements 522
2.11.3 Paths for Software Porting 523
2.11.4 Configuring the Compilation Environment 524
2.11.5 Obtaining the Source Code 524
2.11.6 Compilation and Installation 525
2.11.7 Running and Verification 528
2.12 OpenLB 1.4-0 Porting Guide (GCC for openEuler) 530
2.12.1 Introduction 530
2.12.2 Environment Requirements 530
2.12.3 Paths for Software Porting 531
2.12.4 Configuring the Compilation Environment 532
2.12.4.1 Configuration Description 532
2.12.4.2 Configuring the GCC for openEuler Compilation Environment 533
2.12.4.3 Setting Environment Variables 533
2.12.5 Installing OpenLB 533
2.12.5.1 Installing OpenLB 534
2.12.5.2 Modifying the Configuration File 534
2.12.5.3 Running Test Cases 535
2.12.5.3.1 Overview 535
2.12.5.3.2 Non-parallel Computing 535
2.12.5.3.3 Parallel Computing 535
2.12.5.4 (Optional) Visualization Tool 535
2.13 OpenLB 1.4-0 Porting Guide (BiSheng Compiler) 536

 Copyright © Huawei Technologies Co., Ltd.

2.13.1 Introduction 536
2.13.2 Environment Requirements 536
2.13.3 Paths for Software Porting 537
2.13.4 Configuring the Compilation Environment 538
2.13.4.1 Configuration Description 538
2.13.4.2 Configuring the BiSheng Compiler Environment 539
2.13.4.3 Installing Hyper MPI 539
2.13.4.4 Setting Environment Variables 541
2.13.5 Installing OpenLB 541
2.13.5.1 Installing OpenLB 542
2.13.5.2 Modifying the Configuration File 542
2.13.5.3 Running Test Cases 543
2.13.5.3.1 Non-parallel Computing 543
2.13.5.3.2 Parallel Computing 543
2.13.5.4 (Optional) Visualization Tool 543
2.14 Kokkos 3.5.00 Porting Guide (Kylin V10) 544
2.14.1 Introduction 544
2.14.2 Environment Requirements 544
2.14.3 Paths for Software Porting 545
2.14.4 Configuring the Compilation Environment 546
2.14.4.1 Configuration Description 546
2.14.4.2 Installing CMake 546
2.14.5 Obtaining the Source Code 547
2.14.6 Compilation and Installation 547
2.14.7 Running and Verification 548
2.15 Pencil Code 2021.02.20 Porting Guide (Kylin V10) 548
2.15.1 Introduction 548
2.15.2 Environment Requirements 548
2.15.3 Paths for Software Porting 549
2.15.4 Configuring the Compilation Environment 550
2.15.5 Obtaining the Source Code 550
2.15.6 Compilation and Installation 550
2.15.7 Running and Verification 551
2.16 SPARTA 7Jan2022 Porting Guide (Kylin V10) 552
2.16.1 Introduction 552
2.16.2 Environment Requirements 552
2.16.3 Paths for Software Porting 553
2.16.4 Configuring the Compilation Environment 554
2.16.5 Obtaining the Source Code 554
2.16.6 Compilation and Installation 554
2.16.7 Running and Verification 555
3 Education and Scientific Research 556

 Copyright © Huawei Technologies Co., Ltd.

3.1 UDUNITS 2.2.28 Porting Guide (CentOS 7.6) 558
3.1.1 Introduction 558
3.1.2 Environment Requirements 558
3.1.3 Paths for Software Porting 559
3.1.4 Configuring the Compilation Environment 560
3.1.5 Obtaining the Source Code 560
3.1.6 Compiling and Installing UDUNITS 560
3.1.7 Running and Verifying UDUNITS 561
3.2 GROMACS 2019.3 Porting Guide (CentOS 7.6) 561
3.2.1 Introduction 561
3.2.2 Environment Requirements 562
3.2.3 Paths for Software Porting 563
3.2.4 Configuring the Compilation Environment 564
3.2.4.1 Configuration Description 564
3.2.4.2 Installing OpenBLAS 564
3.2.4.3 Installing FFTW 565
3.2.4.4 Installing CMake 565
3.2.5 Obtaining the Source Code 566
3.2.6 Compiling and Installing GROMACS 566
3.2.7 Running and Verifying GROMACS 567
3.2.8 More Resources 568
3.3 LAMMPS 5 Jun 2019 Porting Guide (CentOS 7.6) 568
3.3.1 Introduction 568
3.3.2 Environment Requirements 569
3.3.3 Paths for Software Porting 569
3.3.4 Configuring the Compilation Environment 570
3.3.4.1 Configuration Description 570
3.3.4.2 Installing FFTW 570
3.3.5 Obtaining the Source Code 571
3.3.6 Compiling and Installing LAMMPS 571
3.3.7 Running and Verifying LAMMPS 572
3.4 Quantum ESPRESSO 6.4.1 Porting Guide (CentOS 7.6) 572
3.4.1 Introduction 572
3.4.2 Environment Requirements 573
3.4.3 Paths for Software Porting 574
3.4.4 Configuring the Compilation Environment 575
3.4.4.1 Configuration Description 575
3.4.4.2 Installing OpenBLAS 575
3.4.4.3 Installing ScaLAPACK 576
3.4.5 Obtaining the Source Code 577
3.4.6 Compiling and Installing Quantum ESPRESSO 577
3.4.7 Running and Verifying Quantum ESPRESSO 577

 Copyright © Huawei Technologies Co., Ltd.

3.5 NAMD 2.13 Porting Guide (CentOS 7.6) 579
3.5.1 Introduction 579
3.5.2 Environment Requirements 579
3.5.3 Paths for Software Porting 580
3.5.4 Configuring the Compilation Environment 581
3.5.4.1 Configuration Description 581
3.5.4.2 Installing FFTW 582
3.5.5 Obtaining the Source Code 582
3.5.6 Compiling and Installing NAMD 583
3.5.7 Running and Verifying NAMD 584
3.5.8 More Information 585
3.6 VASP 5.4.4 Porting Guide (CentOS 7.6) 585
3.6.1 Introduction 585
3.6.2 Environment Requirements 586
3.6.3 Paths for Software Porting 587
3.6.4 Configuring the Compilation Environment 587
3.6.4.1 Configuration Description 587
3.6.4.2 Installing OpenBLAS 588
3.6.4.3 Installing ScaLAPACK 588
3.6.4.4 Installing FFTW 589
3.6.5 Obtaining the Source Code 589
3.6.6 Compiling and Installing VASP 590
3.6.6.1 Basic Installation 590
3.6.6.2 (Optional) Extended Installation 591
3.6.6.2.1 Installing the VTST Transition State Tools 591
3.6.6.2.2 Installing the Wannier90 Interfaces 592
3.6.7 Running and Verifying VASP 593
3.7 ABINIT 8.10.3 Porting Guide (CentOS 7.6) 594
3.7.1 Introduction 594
3.7.2 Environment Requirements 595
3.7.3 Paths for Software Porting 596
3.7.4 Configuring the Compilation Environment 597
3.7.4.1 Configuration Description 597
3.7.4.2 Installing OpenBLAS 597
3.7.4.3 Installing PnetCDF 598
3.7.4.4 Installing HDF5 598
3.7.4.5 Installing NetCDF-C 599
3.7.4.6 Installing NetCDF-Fortran 599
3.7.5 Obtaining the Source Code 600
3.7.6 Compiling and Installing ABINIT 600
3.7.7 Running and Verifying ABINIT 601
3.7.8 Troubleshooting 602

 Copyright © Huawei Technologies Co., Ltd.

3.7.9 More Information....604
3.8 AmberTools 19 Porting Guide (CentOS 7.6)....605
3.8.1 Introduction....605
3.8.2 Environment Requirements....605
3.8.3 Paths for Software Porting....606
3.8.4 Configuring the Compilation Environment....606
3.8.5 Obtaining the Source Code....607
3.8.6 Compiling and Installing AmberTools....607
3.8.7 Running and Verifying AmberTools....608
3.8.8 Troubleshooting....609
3.9 CP2K 7.1 Porting Guide (CentOS 7.6)....610
3.9.1 Introduction....610
3.9.2 Environment Requirements....610
3.9.3 Paths for Software Porting....612
3.9.4 Configuring the Compilation Environment....613
3.9.4.1 Configuration Description....613
3.9.4.2 Installing CMake....614
3.9.4.3 Installing GMP....614
3.9.4.4 Installing Boost....615
3.9.4.5 Installing Libint....615
3.9.4.6 Installing FFTW....616
3.9.4.7 Installing LAPACK....616
3.9.4.8 Installing ScaLAPACK....617
3.9.4.9 Installing ELPA....617
3.9.4.10 Installing Spglib....618
3.9.4.11 Installing Libxc....618
3.9.4.12 Installing GSL....619
3.9.4.13 Installing PLUMED....619
3.9.5 Obtaining the Source Code....619
3.9.6 Compiling and Installing CP2K....620
3.9.7 Running and Verifying CP2K....621
3.9.8 More Information....621
3.10 NWChem 6.8.1 Porting Guide (CentOS 7.6)....621
3.10.1 Introduction....622
3.10.2 Environment Requirements....622
3.10.3 Paths for Software Porting....623
3.10.4 Configuring the Compilation Environment....623
3.10.5 Obtaining the Source Code....624
3.10.6 Compiling and Installing NWChem....624
3.10.7 Running and Verifying NWChem....625
3.10.8 Troubleshooting....626
3.11 ROOT 6.20 Porting Guide (CentOS 7.6)....626

 Copyright © Huawei Technologies Co., Ltd.

3.11.1 Introduction 626
3.11.2 Environment Requirements 627
3.11.3 Paths for Software Porting 628
3.11.4 Configuring the Compilation Environment 628
3.11.4.1 Configuration Description 628
3.11.4.2 Installing CMake 629
3.11.4.3 Installing zstd 629
3.11.5 Obtaining the Source Code 629
3.11.6 Compiling and Installing ROOT 630
3.11.7 Running and Verifying ROOT 630
3.11.8 Troubleshooting 632
3.11.9 More Resources 632
3.12 Geant4 10.6 Porting Guide (CentOS 7.6) 632
3.12.1 Introduction 632
3.12.2 Environment Requirements 633
3.12.3 Paths for Software Porting 634
3.12.4 Configuring the Compilation Environment 634
3.12.4.1 Configuration Description 634
3.12.4.2 Installing CMake 635
3.12.5 Obtaining the Source Code 635
3.12.6 Compiling and Installing Geant4 635
3.12.7 Running and Verifying Geant4 637
3.12.8 Troubleshooting 638
3.12.9 More Information 638
3.13 DL_POLY 1.10 Porting Guide (CentOS 7.6) 638
3.13.1 Introduction 638
3.13.2 Environment Requirements 638
3.13.3 Paths for Software Porting 639
3.13.4 Configuring the Compilation Environment 640
3.13.5 Obtaining the Source Code 640
3.13.6 Compiling and Installing DL_POLY 640
3.13.7 Running and Verifying DL_POLY 641
3.13.7.1 Running and Verifying DL_POLY on a Single Node 641
3.13.7.2 Running and Verifying DL_POLY in Multi-node Mode 641
3.13.8 More Information 642
3.14 GAMESS Porting Guide (CentOS 7.6) 642
3.14.1 Introduction 643
3.14.2 Environment Requirements 643
3.14.3 Paths for Software Porting 644
3.14.4 Configuring the Compilation Environment 645
3.14.4.1 Configuration Description 645
3.14.4.2 Installing HDF5 645

 Copyright © Huawei Technologies Co., Ltd.

3.14.4.3 Installing NetCDF-C 646
3.14.4.4 Installing NetCDF-Fortran 646
3.14.5 Obtaining the Source Code 647
3.14.6 Compiling and Installing GAMESS 647
3.14.7 Running and Verifying GAMESS 648
3.14.8 More Resources 649
3.15 ESPResSo 4.1.4 Porting Guide (CentOS 7.6) 649
3.15.1 Introduction 649
3.15.2 Environment Requirements 650
3.15.3 Paths for Software Porting 651
3.15.4 Configuring the Compilation Environment 651
3.15.4.1 Configuration Description 651
3.15.4.2 Installing Dependencies with Yum 652
3.15.4.3 Installing FFTW 653
3.15.4.4 Installing CMake 653
3.15.4.5 Installing Boost 654
3.15.5 Obtaining the Source Code 654
3.15.6 Compilation and Installation 655
3.15.7 Running and Verification 656
3.16 MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6) 656
3.16.1 Introduction 656
3.16.2 Environment Requirements 657
3.16.3 Paths for Software Porting 657
3.16.4 Configuring the Compilation Environment 658
3.16.4.1 Configuration Description 658
3.16.4.2 Installing CMake 659
3.16.4.3 Installing Python 659
3.16.5 Obtaining the Source Code 660
3.16.6 Compilation and Installation 660
3.16.7 Running and Verification 660
3.17 MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6) 661
3.17.1 Introduction 661
3.17.2 Environment Requirements 661
3.17.3 Paths for Software Porting 662
3.17.4 Configuring the Compilation Environment 663
3.17.4.1 Configuration Description 663
3.17.4.2 Installing HDF5 664
3.17.4.3 Installing BLAS 665
3.17.4.4 Installing LAPACK 665
3.17.4.5 Installing HARMINV 665
3.17.4.6 Installing FFTW 666
3.17.4.7 Installing libctl 666

 Copyright © Huawei Technologies Co., Ltd.

3.17.4.8 Installing MPB 667
3.17.4.9 Installing libGDSII 668
3.17.5 Obtaining the Source Code 668
3.17.6 Compilation and Installation 668
3.17.7 Running and Verification 669
3.17.8 More Resources 669
3.18 Octopus 10.3 Porting Guide (CentOS 7.6) 669
3.18.1 Introduction 669
3.18.2 Environment Requirements 670
3.18.3 Paths for Software Porting 671
3.18.4 Configuring the Compilation Environment 671
3.18.4.1 Configuration Description 671
3.18.4.2 Installing Dependencies 672
3.18.4.3 Installing FFTW 672
3.18.4.4 Installing GSL 673
3.18.4.5 Installing Libxc 673
3.18.5 Obtaining the Source Code 674
3.18.6 Compilation and Installation 674
3.18.7 Running and Verification 674
3.19 Psi4 1.3.2 Porting Guide (CentOS 7.6) 675
3.19.1 Introduction 675
3.19.2 Environment Requirements 676
3.19.3 Paths for Software Porting 677
3.19.4 Configuring the Compilation Environment 677
3.19.4.1 Configuration Description 678
3.19.4.2 Installing System Dependencies 679
3.19.4.3 Installing CMake 679
3.19.4.4 Installing OpenBLAS 679
3.19.4.5 Installing gau2grid 680
3.19.4.6 Installing Libint 680
3.19.4.7 Installing Libxc 681
3.19.4.8 Installing pybind11 681
3.19.5 Obtaining the Source Code 682
3.19.6 Compilation and Installation 682
3.19.7 Running and Verification 682
3.20 SIESTA 4.0.2 Porting Guide (CentOS 7.6) 683
3.20.1 Introduction 683
3.20.2 Environment Requirements 683
3.20.3 Paths for Software Porting 684
3.20.4 Configuring the Compilation Environment 684
3.20.4.1 Configuration Description 684
3.20.4.2 Installing Dependencies 685

 Copyright © Huawei Technologies Co., Ltd.

3.20.4.3 Installing ScaLAPACK 685
3.20.5 Obtaining the Source Code 686
3.20.6 Compilation and Installation 686
3.20.7 Running and Verification 687
3.20.8 Troubleshooting 687
3.21 Wannier90 3.1.0 Porting Guide (CentOS 7.6) 688
3.21.1 Introduction 688
3.21.2 Environment Requirements 688
3.21.3 Paths for Software Porting 689
3.21.4 Configuring the Compilation Environment 689
3.21.4.1 Configuration Description 690
3.21.4.2 Installing the Dependency Libraries and Tools Using Yum 690
3.21.5 Obtaining the Source Code 691
3.21.6 Compilation and Installation 691
3.21.7 Running and Verification 692
3.21.8 More Resources 693
3.22 WanT 2.6.1 Porting Guide (CentOS 7.6) 693
3.22.1 Introduction 693
3.22.2 Environment Requirements 694
3.22.3 Paths for Software Porting 694
3.22.4 Configuring the Compilation Environment 695
3.22.4.1 Configuration Description 695
3.22.4.2 Installing QE 696
3.22.5 Obtaining the Source Code 697
3.22.6 Compilation and Installation 697
3.22.7 Running and Verification 698
3.22.8 More Resources 699
3.23 GROMACS 2019.3 Porting Guide (CentOS 8.2) 700
3.23.1 Introduction 700
3.23.2 Environmental Requirements 700
3.23.3 Paths for Software Porting 701
3.23.4 Configuring the Compilation Environment 702
3.23.4.1 Configuration Description 702
3.23.4.2 Disabling the Nouveau Driver 703
3.23.4.3 Installing NVIDIA CUDA 704
3.23.4.4 Installing the BiSheng Compiler 704
3.23.4.5 Installing Hyper MPI 705
3.23.4.6 Installing OpenBLAS 706
3.23.4.7 Installing FFTW 706
3.23.4.8 Installing CMake 707
3.23.5 Obtaining the Source Code 707
3.23.6 Compilation and Installation 708

 Copyright © Huawei Technologies Co., Ltd.

3.23.7 Running and Verification ... 709
3.24 NAMD 2.14 Porting Guide (CentOS 8.2) ... 710
3.24.1 Introduction ... 710
3.24.2 Environment Requirements ... 710
3.24.3 Paths for Software Porting ... 711
3.24.4 Configuring the Compilation Environment ... 712
3.24.4.1 Configuration Description ... 712
3.24.4.2 Installing FFTW and TCL ... 712
3.24.4.3 Disabling the Nouveau Driver ... 713
3.24.4.4 Installing NVIDIA CUDA ... 713
3.24.4.5 Installing the BiSheng Compiler ... 714
3.24.5 Obtaining the Source Code ... 714
3.24.6 Compilation and Installation ... 714
3.24.7 Running and Verification ... 716
3.25 CP2K 7.1 Porting Guide (openEuler 21.03) ... 716
3.25.1 Introduction ... 716
3.25.2 Environment Requirements ... 717
3.25.3 Paths for Software Porting ... 718
3.25.4 Configuring the Compilation Environment ... 720
3.25.4.1 Configuration Description ... 720
3.25.4.2 Installing OpenSSL ... 721
3.25.4.3 Installing CMake ... 721
3.25.4.4 Installing GMP ... 722
3.25.4.5 Installing Boost ... 722
3.25.4.6 Installing Libint ... 723
3.25.4.7 Installing FFTW ... 724
3.25.4.8 Installing LAPACK ... 724
3.25.4.9 Installing ScaLAPACK ... 725
3.25.4.10 Installing ELPA ... 726
3.25.4.11 Installing Spglib ... 727
3.25.4.12 Installing Libxc ... 727
3.25.4.13 Installing GSL ... 727
3.25.4.14 Installing PLUMED ... 728
3.25.5 Obtaining the Source Code ... 728
3.25.6 Compilation and Installation ... 729
3.25.7 Running and Verification ... 730
3.26 DL_POLY 1.10 Porting Guide (openEuler 21.03) ... 731
3.26.1 Introduction ... 731
3.26.2 Environment Requirements ... 731
3.26.3 Paths for Software Porting ... 732
3.26.4 Configuring the Compilation Environment ... 733
3.26.5 Obtaining the Source Code ... 734

 Copyright © Huawei Technologies Co., Ltd.

3.26.6 Compilation and Installation 734
3.26.7 Running and Verification 735
3.27 GROMACS 2019.3 Porting Guide (openEuler 21.03) 737
3.27.1 Introduction 737
3.27.2 Environment Requirements 738
3.27.3 Paths for Software Porting 739
3.27.4 Configuring the Compilation Environment 740
3.27.4.1 Configuration Description 740
3.27.4.2 Installing OpenBLAS 741
3.27.4.3 Installing FFTW 741
3.27.4.4 Installing CMake 742
3.27.5 Obtaining the Source Code 742
3.27.6 Compilation and Installation 742
3.27.7 Running and Verification 743
3.28 LAMMPS 5 Jun 2019 Porting Guide (openEuler 21.03) 745
3.28.1 Introduction 745
3.28.2 Environment Requirements 746
3.28.3 Paths for Software Porting 746
3.28.4 Configuring the Compilation Environment 747
3.28.4.1 Configuration Description 747
3.28.4.2 Installing FFTW 748
3.28.5 Obtaining the Source Code 749
3.28.6 Compilation and Installation 749
3.28.7 Running and Verification 750
3.29 NWChem 6.8.1 Porting Guide (openEuler 21.03) 751
3.29.1 Introduction 751
3.29.2 Environment Requirements 752
3.29.3 Paths for Software Porting 753
3.29.4 Configuring the Compilation Environment 753
3.29.5 Obtaining the Source Code 754
3.29.6 Compilation and Installation 754
3.29.7 Running and Verification 755
3.30 VASP 5.4.4 Porting Guide (openEuler 21.03) 756
3.30.1 Introduction 757
3.30.2 Environment Requirements 757
3.30.3 Paths for Software Porting 758
3.30.4 Configuring the Compilation Environment 759
3.30.4.1 Configuration Description 759
3.30.4.2 Installing OpenBLAS 760
3.30.4.3 Installing ScaLAPACK 760
3.30.4.4 Installing FFTW 761
3.30.5 Obtaining the Source Code 762

 Copyright © Huawei Technologies Co., Ltd.

3.30.6 Compilation and Installation 762
3.30.7 Running and Verification 766
3.31 Wannier90 3.1.0 Porting Guide (openEuler 21.03) 768
3.31.1 Introduction 768
3.31.2 Environment Requirements 768
3.31.3 Paths for Software Porting 769
3.31.4 Configuring the Compilation Environment 770
3.31.4.1 Configuration Description 770
3.31.4.2 Installing OpenBLAS Dependencies 771
3.31.4.3 Installing Gnuplot 771
3.31.5 Obtaining the Source Code 772
3.31.6 Compilation and Installation 772
3.31.7 Running and Verification 773
3.32 QE 6.4.1 Porting Guide (openEuler 21.03) 775
3.32.1 Introduction 775
3.32.2 Environment Requirements 776
3.32.3 Paths for Software Porting 777
3.32.4 Configuring the Compilation Environment 777
3.32.4.1 Configuration Description 777
3.32.4.2 Installing OpenBLAS 778
3.32.4.3 Installing ScaLAPACK 778
3.32.5 Obtaining the Source Code 779
3.32.6 Compilation and Installation 779
3.32.7 Running and Verification 780
3.33 MOOSE 1.0.0 Porting Guide (openEuler 21.03) 781
3.33.1 Introduction 781
3.33.2 Environment Requirements 782
3.33.3 Paths for Software Porting 783
3.33.4 Configuring the Compilation Environment 783
3.33.4.1 Configuration Description 784
3.33.4.2 Installing CMake 784
3.33.4.3 Installing Python 785
3.33.5 Obtaining the Source Code 785
3.33.6 Compilation and Installation 785
3.33.7 Running and Verification 786
3.34 SIESTA 4.0.2 Porting Guide (openEuler 21.03) 787
3.34.1 Introduction 787
3.34.2 Environment Requirements 788
3.34.3 Paths for Software Porting 789
3.34.4 Configuring the Compilation Environment 789
3.34.4.1 Configuration Description 789
3.34.4.2 Installing ScaLAPACK 790

 Copyright © Huawei Technologies Co., Ltd.

3.34.4.3 Installing OpenBLAS....790
3.34.5 Obtaining the Source Code....791
3.34.6 Compilation and Installation....791
3.34.7 Running and Verification....792
3.35 UDUNITS 2.2.28 Porting Guide (openEuler 21.03)....793
3.35.1 Introduction....793
3.35.2 Environment Requirements....794
3.35.3 Paths for Software Porting....795
3.35.4 Configuring the Compilation Environment....795
3.35.5 Obtaining the Source Code....796
3.35.6 Compilation and Installation....796
3.35.7 Running and Verification....797
3.36 ROOT 6.20 Porting Guide (openEuler 21.03)....797
3.36.1 Introduction....797
3.36.2 Environment Requirements....797
3.36.3 Paths for Software Porting....798
3.36.4 Configuring the Compilation Environment....799
3.36.4.1 Configuration Description....799
3.36.4.2 Installing CMake....800
3.36.4.3 Installing zstd....801
3.36.5 Obtaining the Source Code....801
3.36.6 Compilation and Installation....801
3.36.7 Running and Verification....802
3.37 LAMMPS 10 Feb 2021 Porting Guide (CentOS 8.2)....802
3.37.1 Introduction....802
3.37.2 Environment Requirements....803
3.37.3 Paths for Software Porting....804
3.37.4 Configuring the Compilation Environment....804
3.37.4.1 Configuration Description....804
3.37.4.2 Installing FFTW....805
3.37.4.3 Disabling the Nouveau Driver....806
3.37.4.4 Installing NVIDIA CUDA....806
3.37.4.5 Installing the BiSheng Compiler....807
3.37.4.6 Installing Hyper MPI....807
3.37.5 Obtaining the Source Code....808
3.37.6 Compilation and Installation....808
3.37.7 Running and Verification....810
3.38 QE 6.8 Porting Guide (CentOS 8.2)....810
3.38.1 Introduction....811
3.38.2 Environment Requirements....811
3.38.3 Paths for Software Porting....812
3.38.4 Configuring the Compilation Environment....813

 Copyright © Huawei Technologies Co., Ltd.

3.38.4.1 Configuration Description 813
3.38.4.2 Disabling the Nouveau Driver 814
3.38.4.3 Installing NVIDIA CUDA 814
3.38.4.4 Installing Hyper MPI 815
3.38.4.5 Installing the NVIDIA HPC SDK 816
3.38.4.6 Installing OpenBLAS 817
3.38.5 Obtaining the Source Code 817
3.38.6 Compilation and Installation 817
3.38.7 Running and Verification 819
3.39 ABINIT 8.10.3 Porting Guide (openEuler 21.03) 819
3.39.1 Introduction 819
3.39.2 Environment Requirements 820
3.39.3 Paths for Software Porting 821
3.39.4 Configuring the Compilation Environment 822
3.39.4.1 Configuration Description 822
3.39.4.2 Installing HDF5 823
3.39.4.3 Installing NetCDF-C 824
3.39.4.4 Installing NetCDF-Fortran 825
3.39.4.5 Installing PnetCDF 825
3.39.4.6 Installing OpenBLAS 826
3.39.5 Obtaining the Source Code 826
3.39.6 Compilation and Installation 826
3.39.7 Running and Verification 829
3.39.8 Troubleshooting 830
3.39.8.1 After a Dependency Package Is Configured, an Error Message Is Displayed Indicating that the Dynamic Link Library Cannot Be Found 830
3.40 AmberTools 21 Porting Guide (openEuler 21.03) 831
3.40.1 Introduction 831
3.40.2 Environment Requirements 831
3.40.3 Paths for Software Porting 832
3.40.4 Configuring the Compilation Environment 833
3.40.5 Obtaining the Source Code 833
3.40.6 Compilation and Installation 834
3.40.7 Running and Verification 835
3.40.8 Troubleshooting 836
3.40.8.1 Error "mclock, has not been explicitly declared" Reported When Compiling AmberTools 836
3.41 NAMD 2.13 Porting Guide (openEuler 21.03) 837
3.41.1 Introduction 837
3.41.2 Environment Requirements 837
3.41.3 Paths for Software Porting 838
3.41.4 Configuring the Compilation Environment 839
3.41.4.1 Configuration Description 839
3.41.4.2 Installing FFTW 840

 Copyright © Huawei Technologies Co., Ltd.

3.41.5 Obtaining the Source Code 840
3.41.6 Compilation and Installation 841
3.41.7 Running and Verification 842
3.42 miniGhost 1.0 Porting Guide (openEuler 21.03) 844
3.42.1 Introduction 844
3.42.2 Environment Requirements 844
3.42.3 Paths for Software Porting 845
3.42.4 Configuring the Compilation Environment 846
3.42.5 Obtaining the Source Code 847
3.42.6 Compilation and Installation 847
3.42.7 Running and Verification 848
3.42.8 Troubleshooting 848
3.42.8.1 Error Reported When Running make 848
3.43 MiniFE 2.2.0 Porting Guide (CentOS 7.6) 849
3.43.1 Introduction 849
3.43.2 Environment Requirements 849
3.43.3 Paths for Software Porting 850
3.43.4 Configuring the Compilation Environment 850
3.43.5 Obtaining the Source Code 851
3.43.6 Compiling and Installing MiniFE 851
3.43.7 Running and Verifying MiniFE 851
3.44 miniGhost v1.0 Porting Guide (CentOS 7.6) 852
3.44.1 Introduction 852
3.44.2 Environment Requirements 852
3.44.3 Paths for Software Porting 853
3.44.4 Configuring the Compilation Environment 854
3.44.5 Obtaining the Source Code 854
3.44.6 Compiling and Installing miniGhost 854
3.44.7 Running and Verifying miniGhost 856
3.45 ELK 8.4.30 Porting Guide (Kylin V10) 856
3.45.1 Introduction 856
3.45.2 Environment Requirements 856
3.45.3 Paths for Software Porting 857
3.45.4 Configuring the Compilation Environment 858
3.45.4.1 Configuration Description 858
3.45.4.2 Installing the Dependencies 858
3.45.5 Obtaining the Source Code 859
3.45.6 Compilation and Installation 859
3.45.7 Running and Verification 860
3.46 MOPAC 22.0.4 Porting Guide (Kylin V10) 862
3.46.1 Introduction 862
3.46.2 Environment Requirements 862

 Copyright © Huawei Technologies Co., Ltd.

3.46.3 Paths for Software Porting 863
3.46.4 Configuring the Compilation Environment 864
3.46.5 Obtaining the Source Code 864
3.46.6 Compilation and Installation 864
3.46.7 Running and Verification 865
3.47 QMCPACK 3.14.0 Porting Guide (Kylin V10) 865
3.47.1 Introduction 865
3.47.2 Environment Requirements 866
3.47.3 Paths for Software Porting 866
3.47.4 Configuring the Compilation Environment 867
3.47.4.1 Configuration Description 867
3.47.4.2 Installing the Dependencies 868
3.47.5 Obtaining the Source Code 868
3.47.6 Compilation and Installation 868
3.47.7 Running and Verification 868
3.48 OpenCV 4.5.5 Porting Guide (openEuler 20.03) 869
3.48.1 Introduction 869
3.48.2 Environment Requirements 869
3.48.3 Paths for Software Porting 870
3.48.4 Configuring the Compilation Environment 871
3.48.5 Obtaining the Source Code 871
3.48.6 Compilation and Installation 871
3.48.7 Running and Verification 872
3.49 Phono3py 2.3.2 Porting Guide (Kylin V10) 872
3.49.1 Introduction 872
3.49.2 Environment Requirements 873
3.49.3 Paths for Software Porting 873
3.49.4 Configuring the Compilation Environment 874
3.49.4.1 Configuration Description 874
3.49.4.2 Installing OpenBLAS 874
3.49.5 Obtaining the Source Code 875
3.49.6 Compilation and Installation 875
3.49.7 Running and Verification 876
3.50 QDK 2.3.12 Porting Guide (Kylin V10) 876
3.50.1 Introduction 876
3.50.2 Environment Requirements 877
3.50.3 Paths for Software Porting 877
3.50.4 Configuring the Compilation Environment 878
3.50.5 Obtaining the Source Code 878
3.50.6 Compilation and Installation 879
3.50.7 Running and Verification 879
3.51 Towhee 8.2.3 Porting Guide (Kylin V10) 880

 Copyright © Huawei Technologies Co., Ltd.

3.51.1 Introduction 880
3.51.2 Environment Requirements 880
3.51.3 Paths for Software Porting 881
3.51.4 Configuring the Compilation Environment 882
3.51.5 Obtaining the Source Code 882
3.51.6 Compilation and Installation 882
3.51.7 Running and Verification 883
3.52 CosmoMC June2021 Porting Guide (Kylin V10) 884
3.52.1 Introduction 884
3.52.2 Environment Requirements 885
3.52.3 Paths for Software Porting 885
3.52.4 Configuring the Compilation Environment 886
3.52.5 Obtaining the Source Code 886
3.52.6 Compilation and Installation 886
3.52.7 Running and Verification 887
3.53 ParaFEM 5.0.3 Porting Guide (Kylin V10) 887
3.53.1 Introduction 887
3.53.2 Environment Requirements 888
3.53.3 Paths for Software Porting 888
3.53.4 Configuring the Compilation Environment 889
3.53.5 Obtaining the Source Code 889
3.53.6 Compilation and Installation 890
3.53.7 Running and Verification 890
3.54 VASPKIT 0.52 Porting Guide (Kylin V10) 891
3.54.1 Introduction 891
3.54.2 Environment Requirements 891
3.54.3 Paths for Software Porting 892
3.54.4 Configuring the Compilation Environment 893
3.54.5 Obtaining the Source Code 893
3.54.6 Compilation and Installation 893
3.54.7 Running and Verification 894
3.55 MNE-Python Porting Guide (openEuler 20.03) 894
3.55.1 Introduction 894
3.55.2 Environment Requirements 895
3.55.3 Paths for Software Porting 896
3.55.4 Configuring the Compilation Environment 896
3.55.5 Compilation and Installation 896
3.55.6 Running and Verification 897
3.56 MiniDFT 1.1.1 Porting Guide (CentOS 7.6) 897
3.56.1 Introduction 897
3.56.2 Environment Requirements 898
3.56.3 Paths for Software Porting 899

 Copyright © Huawei Technologies Co., Ltd.

3.56.4 Configuring the Compilation Environment....899
3.56.4.1 Configuration Description....899
3.56.4.2 Installing OpenBLAS....900
3.56.4.3 Installing FFTW....900
3.56.4.4 Installing ScaLAPACK....901
3.56.5 Obtaining the Source Code....901
3.56.6 Compiling and Installing MiniDFT....902
3.56.7 Running and Verifying MiniDFT....902
3.56.8 Troubleshooting....903
3.56.9 More Information....904
3.57 dugksFoam 1.0 Porting Guide (Kylin V10)....904
3.57.1 Introduction....904
3.57.2 Environment Requirements....904
3.57.3 Paths for Software Porting....905
3.57.4 Configuring the Compilation Environment....905
3.57.4.1 Configuration Description....905
3.57.4.2 Installing OpenFOAM v2206....906
3.57.5 Obtaining the Source Code....907
3.57.6 Compilation and Installation....907
3.57.7 Running and Verification....908
3.57.8 Change History....908
3.58 DUNS 4.7.1 Porting Guide (Kylin V10)....908
3.58.1 Introduction....908
3.58.2 Environment Requirements....909
3.58.3 Paths for Software Porting....909
3.58.4 Configuring the Compilation Environment....910
3.58.5 Obtaining the Source Code....910
3.58.6 Compilation and Installation....910
3.58.7 Running and Verification....911
3.58.8 Change History....912
3.59 CalculiX 2.19.0 Porting Guide (Kylin V10)....912
3.59.1 Introduction....912
3.59.2 Environment Requirements....913
3.59.3 Paths for Software Porting....914
3.59.4 Configuring the Compilation Environment....914
3.59.4.1 Configuration Description....914
3.59.4.2 Installing SPOOLES....915
3.59.4.3 Installing ARPACK....915
3.59.4.4 Installing yaml-cpp....916
3.59.4.5 Installing Eigen3....916
3.59.4.6 Installing preCICE....916
3.59.5 Obtaining the Source Code....917

3.59.6 Compilation and Installation 917
3.59.7 Running and Verification 918
3.59.8 Change History 918
3.60 BerkeleyGW 3.0.1 Porting Guide (Kylin V10) 918
3.60.1 Introduction 918
3.60.2 Environment Requirements 919
3.60.3 Paths for Software Porting 920
3.60.4 Configuring the Compilation Environment 920
3.60.4.1 Configuration Description 920
3.60.4.2 Installing LAPACK 921
3.60.4.3 Installing FFTW 921
3.60.4.4 Installing ScaLAPACK 921
3.60.5 Obtaining the Source Code 921
3.60.6 Compilation and Installation 922
3.60.7 Running and Verification 923
3.60.8 Change History 923
3.61 Amber 20 Porting Guide (CentOS 8.2) 923
3.61.1 Introduction 923
3.61.2 Environment Requirements 924
3.61.3 Paths for Software Porting 925
3.61.4 Configuring the Compilation Environment 926
3.61.4.1 Configuration Description 926
3.61.4.2 Disabling the Nouveau Driver 926
3.61.4.3 Installing NVIDIA CUDA 927
3.61.4.4 Installing the BiSheng Compiler 927
3.61.4.5 Installing HMPI 928
3.61.4.6 Installing Anaconda3 929
3.61.4.7 Installing CMake 929
3.61.5 Obtaining the Source Code 929
3.61.6 Compilation and Installation 930
3.61.7 Running and Verification 930
3.61.8 Change History 931
3.62 Galaxy 22.01 Porting Guide (Kylin V10) 931
3.62.1 Introduction 931
3.62.2 Environment Requirements 932
3.62.3 Paths for Software Porting 932
3.62.4 Configuring the Compilation Environment 933
3.62.4.1 Configuration Description 933
3.62.4.2 Installing Python 3.7.6 933
3.62.4.3 Installing virtualenv 934
3.62.5 Obtaining the Source Code 934
3.62.6 Compilation and Installation 934

3.62.7 Running and Verification 935
3.63 NWChem 6.8.1 Porting Guide (Kylin V10) 935
3.63.1 Introduction 935
3.63.2 Environment Requirements 936
3.63.3 Paths for Software Porting 937
3.63.4 Configuring the Compilation Environment 937
3.63.4.1 Configuration Description 937
3.63.4.2 Disabling the Nouveau Driver 938
3.63.4.3 Installing NVIDIA CUDA (Including the GPU Driver) 938
3.63.5 Obtaining the Source Code 939
3.63.6 Compilation and Installation 939
3.63.7 Running and Verification 940
3.63.8 Comparison Test 941
3.63.8.1 Performance Test 941
3.63.9 Change History 942
3.64 T-Flows 0.9 Porting Guide (Kylin V10) 942
3.64.1 Introduction 942
3.64.2 Environment Requirements 942
3.64.3 Paths for Software Porting 943
3.64.4 Configuring the Compilation Environment 944
3.64.4.1 Configuration Description 944
3.64.4.2 Installing metis 944
3.64.5 Obtaining the Source Code 944
3.64.6 Compilation and Installation 945
3.64.7 Running and Verification 946
3.64.8 Change History 948
3.65 yambo 5.1.1 Porting Guide (openEuler 20.03) 948
3.65.1 Introduction 948
3.65.2 Environment Requirements 949
3.65.3 Paths for Software Porting 949
3.65.4 Configuring the Compilation Environment 950
3.65.4.1 Configuration Description 950
3.65.4.2 Installing Dependencies 950
3.65.5 Obtaining the Source Code 950
3.65.6 Compilation and Installation 951
3.65.7 Running and Verification 951
3.65.8 Change History 951
4 Life Science 952
4.1 BLAST 2.9.0 Porting Guide (CentOS 7.6) 958
4.1.1 Introduction 958
4.1.2 Environment Requirements 959
4.1.3 Paths for Software Porting 960

 Copyright © Huawei Technologies Co., Ltd.

4.1.4 Configuring the Compilation Environment 960
4.1.5 Obtaining the Source Code 961
4.1.6 Compiling and Installing BLAST 961
4.1.7 Running and Verifying BLAST 962
4.1.8 More Information 962
4.2 BWA 0.7.17 Porting Guide (CentOS 7.6) 963
4.2.1 Introduction 963
4.2.2 Environment Requirements 963
4.2.3 Paths for Software Porting 964
4.2.4 Configuring the Compilation Environment 965
4.2.5 Obtaining the Source Code 965
4.2.6 Compiling and Installing BWA 965
4.2.7 Running and Verifying BWA 966
4.3 CNVnator 0.4.1 Porting Guide (CentOS 7.6) 967
4.3.1 Introduction 967
4.3.2 Environment Requirements 968
4.3.3 Paths for Software Porting 968
4.3.4 Configuring the Compilation Environment 969
4.3.4.1 Configuration Description 969
4.3.4.2 Installing the Pre-installation Packages and Dependencies 970
4.3.4.3 Installing CMake 970
4.3.4.4 Installing Samtools 971
4.3.4.5 Installing the ROOT Tool Package 971
4.3.5 Obtaining the Source Code 972
4.3.6 Compiling and Installing CNVnator 972
4.3.7 Running and Verifying CNVnator 973
4.3.8 Troubleshooting 974
4.3.9 More Information 975
4.4 Cufflinks 2.2.1 Porting Guide (CentOS 7.6) 975
4.4.1 Introduction 976
4.4.2 Environment Requirements 976
4.4.3 Paths for Software Porting 977
4.4.4 Configuring the Compilation Environment 978
4.4.4.1 Configuration Description 978
4.4.4.2 Installing Boost 978
4.4.4.3 Installing Samtools 979
4.4.4.4 Installing HTSlib 980
4.4.4.5 Installing Eigen 981
4.4.5 Obtaining the Source Code 981
4.4.6 Compiling and Installing Cufflinks 981
4.4.7 Running and Verifying Cufflinks 982
4.4.8 Troubleshooting 982

 Copyright © Huawei Technologies Co., Ltd.

4.4.9 More Information....984
4.5 GATK 4.0.0.0 Porting Guide (CentOS 7.6)....985
4.5.1 Introduction....985
4.5.2 Environment Requirements....985
4.5.3 Paths for Software Porting....987
4.5.4 Configuring the Compilation Environment....987
4.5.4.1 Configuration Description....987
4.5.4.2 Installing Dependencies....988
4.5.4.3 Installing BLAT....988
4.5.4.4 Installing BWA....989
4.5.4.5 Installing Samtools....990
4.5.5 Obtaining the Source Code....990
4.5.6 Compiling and Installing GATK....991
4.5.7 Running and Verifying GATK....991
4.5.7.1 Running GATK4 in Spark Mode....991
4.5.7.2 Running GATK4 in Non-Spark Mode....997
4.5.8 Troubleshooting....998
4.5.9 More Resources....998
4.6 HISAT2 2.1.0 Porting Guide (CentOS 7.6)....999
4.6.1 Introduction....999
4.6.2 Environment Requirements....999
4.6.3 Paths for Software Porting....1000
4.6.4 Configuring the Compilation Environment....1001
4.6.5 Obtaining the Source Code....1001
4.6.6 Compiling and Installing HISAT2....1002
4.6.7 Running and Verifying HISAT2....1004
4.6.8 More Information....1005
4.7 STAR 2.7.1a Porting Guide (CentOS 7.6)....1005
4.7.1 Introduction....1005
4.7.2 Environment Requirements....1006
4.7.3 Paths for Software Porting....1007
4.7.4 Configuring the Compilation Environment....1007
4.7.5 Obtaining the Source Code....1008
4.7.6 Compiling and Installing STAR....1008
4.7.7 Running and Verifying STAR....1009
4.7.8 More Resources....1010
4.8 TopHat2 2.1.1 Porting Guide (CentOS 7.6)....1010
4.8.1 Introduction....1010
4.8.2 Environment Requirements....1010
4.8.3 Paths for Software Porting....1011
4.8.4 Configuring the Compilation Environment....1012
4.8.4.1 Configuration Description....1012

4.8.4.2 Installing Bowtie2 1013
4.8.4.3 Installing Boost 1013
4.8.5 Obtaining the Source Code 1014
4.8.6 Compiling and Installing TopHat2 1014
4.8.7 Running and Verifying TopHat2 1015
4.8.8 More Information 1016
4.9 Bowtie2 2.4.1 Porting Guide (CentOS 7.6) 1016
4.9.1 Introduction 1016
4.9.2 Environment Requirements 1017
4.9.3 Paths for Software Porting 1018
4.9.4 Configuring the Compilation Environment 1018
4.9.4.1 Configuration Description 1018
4.9.4.2 Installing Python 3 1019
4.9.5 Obtaining the Source Code 1019
4.9.6 Compiling and Installing bowtie 2 1020
4.9.6.1 Installing Bowtie2 1020
4.9.6.2 Creating a Bowtie2 Index 1021
4.9.7 Running and Verifying Bowtie2 1021
4.9.8 More Information 1021
4.10 FastQC 0.11.9 Porting Guide (CentOS 7.6) 1021
4.10.1 Introduction 1021
4.10.2 Environment Requirements 1022
4.10.3 Paths for Software Porting 1023
4.10.4 Configuring the Compilation Environment 1023
4.10.4.1 Configuration Description 1023
4.10.4.2 Installing OpenJDK 1024
4.10.4.3 Installing the GUI 1024
4.10.5 Obtaining the Source Code 1024
4.10.6 Compiling and Installing FastQC 1025
4.10.7 Running and Verifying FastQC 1026
4.10.8 More Resources 1029
4.11 Beagle 5.1 Porting Guide (CentOS 7.6) 1029
4.11.1 Introduction 1029
4.11.2 Environment Requirements 1029
4.11.3 Paths for Software Porting 1030
4.11.4 Configuring the Compilation Environment 1031
4.11.4.1 Configuration Description 1031
4.11.4.2 Setting Up the Java Environment 1032
4.11.5 Obtaining the Source Code 1032
4.11.6 Running and Verifying Beagle 1032
4.12 Bowtie 1.2.3 Porting Guide (CentOS 7.6) 1034
4.12.1 Introduction 1034

 Copyright © Huawei Technologies Co., Ltd.

4.12.2 Environment Requirements 1034
4.12.3 Paths for Software Porting 1035
4.12.4 Configuring the Compilation Environment 1036
4.12.5 Obtaining the Source Code 1036
4.12.6 Compiling and Installing Bowtie 1037
4.12.6.1 Installing Bowtie 1037
4.12.6.2 Creating a Bowtie Index 1039
4.12.7 Running and Verifying Bowtie 1039
4.12.8 Troubleshooting 1039
4.12.9 More Information 1040
4.13 pblat 2.1 Porting Guide (CentOS 7.6) 1040
4.13.1 Introduction 1040
4.13.2 Environment Requirements 1041
4.13.3 Paths for Software Porting 1041
4.13.4 Configuring the Compilation Environment 1042
4.13.5 Obtaining the Source Code 1042
4.13.6 Compiling and Installing pblat 1043
4.13.7 Running and Verifying pblat 1043
4.13.8 Troubleshooting 1044
4.13.9 More Resources 1044
4.14 SnpEff V4_3 Porting Guide (CentOS 7.6) 1044
4.14.1 Introduction 1045
4.14.2 Environment Requirements 1045
4.14.3 Paths for Software Porting 1046
4.14.4 Configuring the Compilation Environment 1046
4.14.5 Obtaining the Source Code 1047
4.14.6 Compiling and Installing SnpEff 1047
4.14.7 Running and Verifying SnpEff 1047
4.14.8 More Resources 1048
4.15 Velvet 1.2.10 Porting Guide (CentOS 7.6) 1048
4.15.1 Introduction 1048
4.15.2 Environment Requirements 1049
4.15.3 Paths for Software Porting 1049
4.15.4 Configuring the Compilation Environment 1050
4.15.5 Obtaining the Source Code 1050
4.15.6 Compiling and Installing Velvet 1051
4.15.7 Running and Verifying Velvet 1051
4.15.8 More Resources 1052
4.16 XHMM Porting Guide (CentOS 7.6) 1052
4.16.1 Introduction 1053
4.16.2 Environment Requirements 1053
4.16.3 Paths for Software Porting 1054

 Copyright © Huawei Technologies Co., Ltd.

4.16.4 Configuring the Compilation Environment 1054
4.16.4.1 Configuration Description 1054
4.16.4.2 Installing OpenBLAS 1055
4.16.4.3 Installing LAPACK 1055
4.16.5 Obtaining the Source Code 1056
4.16.6 Compiling and Installing XHMM 1056
4.16.7 Running and Verifying XHMM 1057
4.17 pysam 0.15.4 Porting Guide (CentOS 7.6) 1058
4.17.1 Introduction 1058
4.17.2 Environment Requirements 1058
4.17.3 Paths for Software Porting 1059
4.17.4 Configuring the Compilation Environment 1060
4.17.4.1 Configuration Description 1060
4.17.4.2 Installing the Python Interpreter 1060
4.17.4.3 Installing the Cython Package 1061
4.17.5 Obtaining the Source Code 1061
4.17.6 Compiling and Installing Pysam 1061
4.17.7 Running and Verifying Pysam 1062
4.17.8 Troubleshooting 1062
4.17.9 More Information 1063
4.18 Abyss 2.2.4 Porting Guide (CentOS 7.6) 1063
4.18.1 Introduction 1063
4.18.2 Environment Requirements 1063
4.18.3 Paths for Software Porting 1064
4.18.4 Configuring the Compilation Environment 1065
4.18.5 Obtaining the Source Code 1065
4.18.6 Compiling and Installing Abyss 1066
4.18.7 Running and Verifying Abyss 1066
4.19 Chaste 2019.1 Porting Guide (CentOS 7.6) 1068
4.19.1 Introduction 1068
4.19.2 Environment Requirements 1068
4.19.3 Paths for Software Porting 1070
4.19.4 Configuring the Compilation Environment 1071
4.19.4.1 Configuration Description 1071
4.19.4.2 Installing Xerces and python-dateutil 1072
4.19.4.3 Installing PETSc 1072
4.19.4.4 Installing Boost 1073
4.19.4.5 Installing Python 1074
4.19.4.6 Installing Amara 1074
4.19.4.7 Installing XSD 1075
4.19.4.8 Installing RDFLib 1075
4.19.4.9 Installing VTK 1075

 Copyright © Huawei Technologies Co., Ltd.

4.19.4.10 Installing lxml 1076
4.19.4.11 Installing OpenBLAS 1076
4.19.5 Obtaining the Source Code 1077
4.19.6 Compiling and Installing Chaste 1077
4.19.7 Running and Verifying Chaste 1078
4.19.8 Troubleshooting 1079
4.19.9 More Resources 1081
4.20 COPASI 4.27.217 Porting Guide (CentOS 7.6) 1081
4.20.1 Introduction 1081
4.20.2 Environment Requirements 1081
4.20.3 Paths for Software Porting 1082
4.20.4 Configuring the Compilation Environment 1083
4.20.5 Obtaining the Source Code 1083
4.20.6 Compiling and Installing COPASI 1083
4.20.7 Running and Verifying COPASI 1084
4.20.8 Troubleshooting 1085
4.21 FASTA 36.3.8 Porting Guide (CentOS 7.6) 1085
4.21.1 Introduction 1085
4.21.2 Environment Requirements 1086
4.21.3 Paths for Software Porting 1087
4.21.4 Configuring the Compilation Environment 1087
4.21.5 Obtaining the Source Code 1087
4.21.6 Compiling and Installing FASTA 1088
4.21.7 Running and Verifying FASTA 1089
4.22 GMAP 2015-9-21 Porting Guide (CentOS 7.6) 1089
4.22.1 Introduction 1089
4.22.2 Environment Requirements 1089
4.22.3 Paths for Software Porting 1090
4.22.4 Configuring the Compilation Environment 1091
4.22.5 Obtaining the Source Code 1091
4.22.6 Compiling and Installing GMAP 1092
4.22.7 Running and Verifying GMAP 1092
4.23 VarScan 2.4.2 Porting Guide (CentOS 7.6) 1093
4.23.1 Introduction 1093
4.23.2 Environment Requirements 1093
4.23.3 Paths for Software Porting 1094
4.23.4 Configuring the Compilation Environment 1095
4.23.5 Obtaining the Source Code 1095
4.23.6 Compiling and Installing VarScan 1096
4.23.7 Running and Verifying VarScan 1097
4.24 Minimap2 Porting Guide (CentOS 7.6) 1097
4.24.1 Introduction 1097

 Copyright © Huawei Technologies Co., Ltd.

4.24.2 Environment Requirements 1098
4.24.3 Paths for Software Porting 1098
4.24.4 Configuring the Compilation Environment 1099
4.24.5 Obtaining the Source Code 1099
4.24.6 Compiling and Installing Minimap2 1100
4.24.7 Running and Verifying Minimap2 1100
4.25 bedtools 2.29.2 Porting Guide (CentOS 7.6) 1100
4.25.1 Introduction 1101
4.25.2 Environment Requirements 1101
4.25.3 Paths for Software Porting 1102
4.25.4 Configuring the Compilation Environment 1102
4.25.5 Obtaining the Source Code 1103
4.25.6 Compiling and Installing bedtools 1103
4.25.7 Running and Verifying bedtools 1103
4.26 BEAST2 2.6.3 Porting Guide (CentOS 7.6) 1104
4.26.1 Introduction 1104
4.26.2 Environment Requirements 1104
4.26.3 Paths for Software Porting 1105
4.26.4 Configuring the Compilation Environment 1106
4.26.5 Obtaining the Source Code 1106
4.26.6 Compiling and Installing BEAST 2 1106
4.26.7 Running and Verifying BEAST 2 1107
4.26.8 More Resources 1112
4.27 bcftools 1.10.2 Porting Guide (CentOS 7.6) 1112
4.27.1 Introduction 1112
4.27.2 Environment Requirements 1113
4.27.3 Paths for Software Porting 1113
4.27.4 Configuring the Compilation Environment 1114
4.27.5 Obtaining the Source Code 1114
4.27.6 Compiling and Installing BCFtools 1115
4.27.7 Running and Verifying BCFtools 1115
4.28 Picard 2.23.3 Porting Guide (CentOS 7.6) 1116
4.28.1 Introduction 1116
4.28.2 Environment Requirements 1116
4.28.3 Paths for Software Porting 1117
4.28.4 Configuring the Compilation Environment 1118
4.28.5 Obtaining the Source Code 1119
4.28.6 Running and Installing Picard 1119
4.28.7 Running and Verifying Picard 1120
4.28.8 Troubleshooting 1121
4.29 SOAPdenovo r241 Porting Guide (CentOS 7.6) 1121
4.29.1 Introduction 1121

 Copyright © Huawei Technologies Co., Ltd.

4.29.2 Environment Requirements 1122
4.29.3 Paths for Software Porting 1123
4.29.4 Configuring the Compilation Environment 1123
4.29.5 Obtaining the Source Code 1124
4.29.6 Compiling and Installing SOAPdenovo 1124
4.29.7 Running and Verifying SOAPdenovo 1125
4.30 Blastz 2004-12-27 Porting Guide (CentOS 7.6) 1126
4.30.1 Introduction 1126
4.30.2 Environment Requirements 1127
4.30.3 Paths for Software Porting 1127
4.30.4 Configuring the Compilation Environment 1128
4.30.5 Obtaining Source Code 1128
4.30.6 Compilation and Installation 1129
4.30.7 Running and Verifying Blastz 1130
4.31 CAFE 5.0b2 Porting Guide (CentOS 7.6) 1131
4.31.1 Introduction 1131
4.31.2 Environment Requirements 1131
4.31.3 Paths for Software Porting 1132
4.31.4 Configuring the Compilation Environment 1133
4.31.5 Obtaining Source Code 1133
4.31.6 Compilation and Installation 1133
4.31.7 Running and Verifying CAFE 1134
4.31.8 More Resources 1135
4.32 fastp Porting Guide (CentOS 7.6) 1135
4.32.1 Introduction 1135
4.32.2 Environment Requirements 1135
4.32.3 Paths for Software Porting 1136
4.32.4 Configuring the Compilation Environment 1137
4.32.5 Obtaining Source Code 1137
4.32.6 Compiling and Installing fastp 1137
4.32.7 Running and Verifying fastp 1138
4.32.8 More Resources 1138
4.33 GCE 1.0.2 Porting Guide (CentOS 7.6) 1138
4.33.1 Introduction 1138
4.33.2 Environment Requirements 1139
4.33.3 Paths for Software Porting 1139
4.33.4 Configuring the Compilation Environment 1140
4.33.4.1 Configuration Description 1140
4.33.4.2 Installing CMake 1140
4.33.5 Obtaining Source Code 1141
4.33.6 Compilation and Installation 1141
4.33.7 Running and Verifying GCE 1142

 Copyright © Huawei Technologies Co., Ltd.

4.33.8 More Resources....1142
4.34 GEMMA 0.98.1 Porting Guide (CentOS 7.6)....1142
4.34.1 Introduction....1142
4.34.2 Environment Requirements....1143
4.34.3 Paths for Software Porting....1144
4.34.4 Configuring the Compilation Environment....1144
4.34.4.1 Installing CMake....1145
4.34.4.2 Installing OpenBLAS....1145
4.34.4.3 Installing Eigen....1146
4.34.4.4 Installing GSL....1146
4.34.5 Obtaining Source Code....1147
4.34.6 Compilation and Installation....1147
4.34.7 Running and Verifying GEMMA....1148
4.34.8 More Resources....1148
4.35 kmergenie 1.7051 Porting Guide (CentOS 7.6)....1148
4.35.1 Introduction....1148
4.35.2 Environment Requirements....1149
4.35.3 Paths for Software Porting....1150
4.35.4 Configuring the Compilation Environment....1150
4.35.5 Obtaining the Source Code....1151
4.35.6 Compiling and Installing KmerGenie....1151
4.35.6.1 Installing R....1151
4.35.6.2 Installing Distribute....1152
4.35.6.3 Installing KmerGenie....1152
4.35.7 Running and Verifying KmerGenie....1153
4.36 LASTZ 1.04.03 Porting Guide (CentOS 7.6)....1154
4.36.1 Introduction....1154
4.36.2 Environment Requirements....1154
4.36.3 Paths for Software Porting....1155
4.36.4 Configuring the Compilation Environment....1156
4.36.5 Obtaining the Source Code....1156
4.36.6 Compiling and Installing LASTZ....1156
4.36.7 Running and Verifying LASTZ....1157
4.36.8 Troubleshooting....1158
4.36.9 More Resources....1159
4.37 MAFFT Porting Guide (CentOS 7.6)....1159
4.37.1 Introduction....1159
4.37.2 Environment Requirements....1159
4.37.3 Paths for Software Porting....1160
4.37.4 Obtaining the Source Code....1161
4.37.5 Compiling and Installing MAFFT....1161
4.37.5.1 Basic Installation....1161

4.37.5.2 (Optional) Installing MPI....1162
4.37.5.3 (Optional) Installing MAFFT Extensions....1162
4.37.6 Running and Verifying MAFFT....1162
4.37.7 More Resources....1163
4.38 megahit 1.2.9 Porting Guide (CentOS 7.6)....1163
4.38.1 Introduction....1163
4.38.2 Environment Requirements....1164
4.38.3 Paths for Software Porting....1165
4.38.4 Configuring the Compilation Environment....1165
4.38.4.1 Configuration Description....1165
4.38.4.2 Installing CMake....1166
4.38.4.3 Installing sse2neon....1166
4.38.5 Obtaining the Source Code....1167
4.38.6 Compiling and Installing MEGAHIT....1167
4.38.7 Running and Verifying MEGAHIT....1169
4.38.8 More Resources....1169
4.39 PASA 2.4.1 Porting Guide (CentOS 7.6)....1169
4.39.1 Introduction....1169
4.39.2 Environment Requirements....1170
4.39.3 Paths for Software Porting....1171
4.39.4 Configuring the Compilation Environment....1172
4.39.4.1 Configuration Description....1172
4.39.4.2 Installing Perl....1172
4.39.4.3 Installing URI....1173
4.39.4.4 Installing Blat....1173
4.39.4.5 Installing MariaDB....1174
4.39.4.6 Installing Samtools....1175
4.39.5 Obtaining the Source Code....1175
4.39.6 Compiling and Installing PASA....1176
4.39.7 Running and Verifying PASA....1176
4.39.8 Troubleshooting....1177
4.39.9 More Resources....1178
4.40 Augustus 3.3.3 Porting Guide (CentOS 7.6)....1178
4.40.1 Introduction....1178
4.40.2 Environment Requirements....1179
4.40.3 Planning the Paths for Software Porting....1180
4.40.4 Configuring the Compilation Environment....1180
4.40.4.1 Installing CMake....1181
4.40.4.2 Installing HTSlib....1182
4.40.4.3 Installing BCFtools....1182
4.40.4.4 Installing SAMtools....1183
4.40.4.5 Installing Bamtools....1184

 Copyright © Huawei Technologies Co., Ltd.

4.40.5 Obtaining the Source Code....1184
4.40.6 Compiling and Installing AUGUSTUS....1184
4.40.7 Running and Verifying AUGUSTUS....1185
4.41 Delly 0.8.5 Porting Guide (CentOS 7.6)....1186
4.41.1 Introduction....1186
4.41.2 Environment Requirements....1187
4.41.3 Paths for Software Porting....1188
4.41.4 Configuring the Compilation Environment....1188
4.41.4.1 Configuration Description....1188
4.41.4.2 Installing Boost....1189
4.41.5 Obtaining Source Code....1189
4.41.6 Compilation and Installation....1190
4.41.7 Running and Verification....1190
4.42 FastTree 2.1.11 Porting Guide (CentOS 7.6)....1192
4.42.1 Introduction....1193
4.42.2 Environment Requirements....1193
4.42.3 Paths for Software Porting....1194
4.42.4 Configuring the Compilation Environment....1194
4.42.4.1 Installing GCC....1195
4.42.5 Obtaining Source Code....1195
4.42.6 Compilation and Installation....1195
4.42.7 Running and Verifying FastTree....1196
4.42.8 More Resources....1196
4.43 Jellyfish 2.2.10 Porting Guide (CentOS 7.6)....1196
4.43.1 Introduction....1196
4.43.2 Environment Requirements....1197
4.43.3 Paths for Software Porting....1198
4.43.4 Configuring the Compilation Environment....1198
4.43.5 Obtaining the Source Code....1198
4.43.6 Compiling and Installing Jellyfish....1199
4.43.7 Running and Verifying Jellyfish....1199
4.44 PLINK 1.9 Porting Guide (CentOS 7.6)....1200
4.44.1 Introduction....1200
4.44.2 Environment Requirements....1200
4.44.3 Planning Paths for Software Porting....1201
4.44.4 Obtaining the Source Code....1201
4.44.5 Configuring the Compilation Environment....1202
4.44.6 Compiling and Installing PLINK....1202
4.44.7 More Resources....1203
4.45 SNAP 2013-11-29 Porting Guide (CentOS 7.6)....1203
4.45.1 Introduction....1203
4.45.2 Environment Requirements....1203

 Copyright © Huawei Technologies Co., Ltd.

4.45.3 Paths for Software Porting....1204
4.45.4 Configuring the Compilation Environment....1205
4.45.4.1 Configuration Description....1205
4.45.4.2 Installing Maker....1205
4.45.5 Obtaining the Source Code....1206
4.45.6 Compiling and Installing SNAP....1206
4.45.7 Running and Verifying SNAP....1207
4.45.8 More Resources....1208
4.46 kmersGWAS 0.2 Porting Guide (CentOS 7.6)....1208
4.46.1 Introduction....1209
4.46.2 Environment Requirements....1209
4.46.3 Paths for Software Porting....1210
4.46.4 Configuring the Compilation Environment....1211
4.46.4.1 Configuration Description....1211
4.46.4.2 Installing R....1212
4.46.4.3 Installing CMake....1212
4.46.4.4 Installing OpenBLAS....1213
4.46.4.5 Installing Eigen....1213
4.46.4.6 Installing GSL....1214
4.46.4.7 Installing Gemma....1214
4.46.5 Obtaining the Source Code....1215
4.46.6 Compiling and Installing kmersGWAS....1215
4.46.7 Running and Verifying kmersGWAS....1216
4.47 PAML 4.9j Porting Guide (CentOS 7.6)....1217
4.47.1 Introduction....1217
4.47.2 Environment Requirements....1217
4.47.3 Planning Paths for Software Porting....1218
4.47.4 Obtaining the Source Code....1219
4.47.5 Compiling and Installing PAML....1219
4.47.6 Running and Verifying PAML....1220
4.47.7 More Resources....1220
4.48 Diamond 2.0.4 Porting Guide (CentOS 7.6)....1221
4.48.1 Introduction....1221
4.48.2 Environment Requirements....1221
4.48.3 Paths for Software Porting....1222
4.48.4 Configuring the Compilation Environment....1223
4.48.4.1 Installing CMake....1223
4.48.5 Obtaining Source Code....1224
4.48.6 Compilation and Installation....1224
4.48.7 Running and Verifying Diamond....1225
4.48.8 More Resources....1225
4.49 HISAT 0.1.6 Porting Guide (CentOS 7.6)....1225

 Copyright © Huawei Technologies Co., Ltd.

4.49.1 Introduction....1225
4.49.2 Environment Requirements....1226
4.49.3 Paths for Software Porting....1227
4.49.4 Configuring the Compilation Environment....1227
4.49.5 Obtaining Source Code....1228
4.49.6 Compilation and Installation....1228
4.49.7 Running and Verification....1230
4.50 MACS2 2.1.4 Porting Guide (CentOS 7.6)....1231
4.50.1 Introduction....1231
4.50.2 Environment Requirements....1232
4.50.3 Paths for Software Porting....1233
4.50.4 Configuring the Compilation Environment....1233
4.50.4.1 Configuration Description....1233
4.50.4.2 Installing python-devel....1234
4.50.4.3 Installing Cython....1234
4.50.4.4 Installing NumPy....1234
4.50.5 Obtaining the Source Code....1235
4.50.6 Compiling and Installing MACS2....1235
4.50.7 Running and Verifying MACS2....1235
4.50.8 More Resources....1237
4.51 Trinity 2.11.0 Porting Guide (CentOS 7.6)....1237
4.51.1 Introduction....1237
4.51.2 Environment Requirements....1238
4.51.3 Paths for Software Porting....1240
4.51.4 Configuring the Compilation Environment....1240
4.51.4.1 Configuration Description....1241
4.51.4.2 Installing All Dependencies Using Yum....1241
4.51.4.3 Installing Boost....1242
4.51.4.4 Installing CMake....1242
4.51.4.5 Installing Salmon....1242
4.51.4.6 Installing HTSlib....1243
4.51.4.7 Installing Samtools....1243
4.51.4.8 Installing Jellyfish....1244
4.51.4.9 Installing Bowtie 2....1244
4.51.4.10 Installing Python....1245
4.51.4.11 Installing Cython....1245
4.51.4.12 Installing NumPy....1246
4.51.5 Obtaining the Source Code....1246
4.51.6 Compiling and Installing Trinity....1246
4.51.7 Running and Verifying Trinity....1247
4.52 LDBlockshow 1.36 Porting Guide (CentOS 7.6)....1248
4.52.1 Introduction....1248

 Copyright © Huawei Technologies Co., Ltd.

4.52.2 Environment Requirements 1249
4.52.3 Paths for Software Porting 1250
4.52.4 Obtaining the Source Code 1250
4.52.5 Configuring the Compilation Environment 1251
4.52.6 Compiling and Installing LDBlockShow 1251
4.52.7 Running and Verifying LDBlockShow 1252
4.53 bioBakery Porting Guide (CentOS 8.2) 1252
4.53.1 Introduction 1252
4.53.2 Environment Requirements 1253
4.53.3 Paths for Software Porting 1253
4.53.4 Configuring the Compilation Environment 1254
4.53.4.1 Installing Anaconda 1254
4.53.5 Compilation and Installation 1256
4.53.6 Running and Verification 1259
4.53.7 More Resources 1259
4.54 Bioconda Porting Guide (CentOS 8.2) 1259
4.54.1 Introduction 1259
4.54.2 Environment Requirements 1260
4.54.3 Paths for Software Porting 1261
4.54.4 Configuring the Compilation Environment 1261
4.54.5 Compilation and Installation 1261
4.54.6 Running and Verification 1263
4.54.7 More Resources 1263
4.55 Boost 1.72 Porting Guide (openEuler 20.03) 1264
4.55.1 Introduction 1264
4.55.2 Environment Requirements 1264
4.55.3 Paths for Software Porting 1265
4.55.4 Configuring the Compilation Environment 1265
4.55.5 Obtaining the Source Code 1266
4.55.6 Compilation and Installation 1266
4.55.7 Running and Verification 1266
4.56 Bustools 0.41.0 Porting Guide (CentOS 8.2) 1267
4.56.1 Introduction 1267
4.56.2 Environment Requirements 1267
4.56.3 Paths for Software Porting 1268
4.56.4 Configuring the Compilation Environment 1269
4.56.4.1 Installing GCC, CMake, and Git 1269
4.56.5 Obtaining the Source Code 1269
4.56.6 Compilation and Installation 1270
4.57 Circos 0.69.9 Porting Guide (CentOS 8.2) 1270
4.57.1 Introduction 1270
4.57.2 Environment Requirements 1271

4.57.3 Paths for Software Porting....1271
4.57.4 Configuring the Compilation Environment....1272
4.57.4.1 Configuration Description....1272
4.57.4.2 Installing Perl....1273
4.57.4.3 Installing libgd....1273
4.57.4.4 Installing GD....1274
4.57.5 Obtaining the Source Code....1274
4.57.6 Compilation and Installation....1274
4.57.7 Running and Verification....1275
4.58 cisTEM 1.0 Porting Guide (CentOS 7.6)....1276
4.58.1 Introduction....1276
4.58.2 Environment Requirements....1276
4.58.3 Paths for Software Porting....1277
4.58.4 Configuring the Compilation Environment....1278
4.58.4.1 Configuration Description....1278
4.58.4.2 Installing FFTW....1278
4.58.4.3 Installing wxWidgets....1279
4.58.5 Obtaining the Source Code....1279
4.58.6 Compilation and Installation....1279
4.58.7 Running and Verification....1281
4.59 EMAN2 2.91 Porting Guide (CentOS 8.2)....1281
4.59.1 Introduction....1281
4.59.2 Environment Requirements....1282
4.59.3 Paths for Software Porting....1282
4.59.4 Configuring the Compilation Environment....1283
4.59.4.1 Installing Anaconda....1283
4.59.4.2 Porting bsddb3....1285
4.59.4.3 Porting TensorFlow 2.4.1....1287
4.59.5 Compilation and Installation....1288
4.59.6 Running and Verification....1290
4.59.7 Troubleshooting....1291
4.59.7.1 Many Resources Fail to Be Automatically Obtained for Bazel Build When Porting TensorFlow 2.4.1....1291
4.59.8 More Resources....1292
4.60 EMBOSS 6.5.7 Porting Guide (CentOS 7.6)....1292
4.60.1 Introduction....1293
4.60.2 Environment Requirements....1293
4.60.3 Paths for Software Porting....1294
4.60.4 Configuring the Compilation Environment....1294
4.60.4.1 Installing the Dependencies....1295
4.60.5 Obtaining the Source Code....1295
4.60.6 Compilation and Installation....1295
4.60.7 More Resources....1296

 Copyright © Huawei Technologies Co., Ltd.

4.61 HumanN 3.0.1 Porting Guide (CentOS 8.2) 1296
4.61.1 Introduction 1296
4.61.2 Environment Requirements 1296
4.61.3 Paths for Software Porting 1297
4.61.4 Configuring the Compilation Environment 1298
4.61.5 Compilation and Installation 1298
4.61.5.1 Installing HUMAnN 1298
4.61.5.2 Installing Bowtie 2 1299
4.61.6 Running and Verification 1300
4.62 kallisto 0.46.2 Porting Guide (CentOS 8.2) 1301
4.62.1 Introduction 1301
4.62.2 Environment Requirements 1301
4.62.3 Paths for Software Porting 1302
4.62.4 Configuring the Compilation Environment 1303
4.62.4.1 Installing GCC, CMake, and Git 1303
4.62.4.2 Installing HTSlib 1304
4.62.5 Obtaining the Source Code 1304
4.62.6 Installing kallisto 1304
4.63 kb-python 0.26.4 Porting Guide (CentOS 8.2) 1305
4.63.1 Introduction 1305
4.63.2 Environment Requirements 1305
4.63.3 Paths for Software Porting 1306
4.63.4 Configuring the Compilation Environment 1307
4.63.4.1 Installing GCC, CMake, and Git 1307
4.63.4.2 Installing Anaconda3 1308
4.63.5 Obtaining the Source Code 1308
4.63.6 Compilation and Installation 1308
4.63.7 More Resources 1310
4.64 KneadData 0.7.7 Porting Guide (CentOS 8.2) 1310
4.64.1 Introduction 1310
4.64.2 Environment Requirements 1310
4.64.3 Paths for Software Porting 1311
4.64.4 Configuring the Compilation Environment 1311
4.64.5 Obtaining the Source Code 1312
4.64.6 Compilation and Installation 1312
4.64.7 Running and Verification 1313
4.64.7.1 Verifying the Version 1313
4.64.7.2 Verifying the Single-Node Running 1313
4.65 Kraken2 Porting Guide (CentOS 8.2) 1314
4.65.1 Introduction 1314
4.65.2 Environment Requirements 1314
4.65.3 Paths for Software Porting 1315

 Copyright © Huawei Technologies Co., Ltd.

4.65.4 Configuring the Compilation Environment 1315
4.65.5 Obtaining the Source Code 1316
4.65.6 Compilation and Installation 1316
4.65.7 Running and Verification 1317
4.66 MetaPhlAn 3.0.13 Porting Guide (CentOS 8.2) 1317
4.66.1 Introduction 1317
4.66.2 Environment Requirements 1318
4.66.3 Paths for Software Porting 1318
4.66.4 Configuring the Compilation Environment 1319
4.66.5 Obtaining the Source Code 1319
4.66.6 Compilation and Installation 1320
4.66.6.1 Installing Python 1320
4.66.6.2 Installing Samtools 1321
4.66.6.3 Installing MetaPhlAn 1321
4.66.7 Running and Verification 1321
4.66.7.1 Downloading the Input File 1322
4.66.7.2 Installing Bowtie2 1322
4.66.7.3 Viewing the Running Result 1323
4.66.8 Troubleshooting 1324
4.66.8.1 Python Installation Failure 1324
4.66.8.2 SSL Exception Occurs During MetaPhlAn Installation 1325
4.66.8.3 **os/rand.c** File Not Found When Installing MetaPhlAn 1325
4.66.8.4 MetaPhlAn Version Verification Error 1326
4.66.8.5 An Error Occurs When Running a Single Sample 1326
4.67 muparser v2.3.2 Porting Guide (openEuler 20.03) 1327
4.67.1 Introduction 1327
4.67.2 Environment Requirements 1327
4.67.3 Paths for Software Porting 1328
4.67.4 Configuring the Compilation Environment 1329
4.67.5 Obtaining the Source Code 1329
4.67.6 Compilation and Installation 1330
4.67.7 Running and Verification 1330
4.67.8 More Resources 1332
4.68 Open CASCADE Technology 7.5.0 Porting Guide (openEuler 20.03) 1332
4.68.1 Introduction 1332
4.68.2 Environment Requirements 1332
4.68.3 Paths for Software Porting 1333
4.68.4 Configuring the Compilation Environment 1334
4.68.4.1 Installing FreeType 1334
4.68.4.2 Installing FreeImage 1335
4.68.5 Obtaining the Source Code 1335
4.68.6 Compilation and Installation 1335

 Copyright © Huawei Technologies Co., Ltd.

4.68.7 Running and Verification....1336
4.68.8 More Resources....1337
4.69 PYCXX 6.2.8 Porting Guide (openEuler 20.03)....1337
4.69.1 Introduction....1337
4.69.2 Environment Requirements....1337
4.69.3 Paths for Software Porting....1338
4.69.4 Configuring the Compilation Environment....1338
4.69.5 Obtaining the Source Code....1339
4.69.6 Compilation and Installation....1339
4.69.7 Running and Verification....1340
4.69.8 More Resources....1340
4.70 Python 3.8.2 Porting Guide (openEuler 20.03)....1340
4.70.1 Introduction....1340
4.70.2 Environment Requirements....1341
4.70.3 Paths for Software Porting....1341
4.70.4 Configuring the Compilation Environment....1342
4.70.5 Obtaining the Source Code....1342
4.70.6 Compilation and Installation....1343
4.70.7 Running and Verification....1343
4.70.8 Troubleshooting....1344
4.70.8.1 Error Reported When Starting Python3....1344
4.70.8.2 Error Reported for Running the Yum Command After Python3 Is Installed....1344
4.70.8.3 System Suspended When Running the "Run tests sequebtially" Command....1345
4.71 Qt 5.13.0 Porting Guide (openEuler 20.03)....1346
4.71.1 Introduction....1347
4.71.2 Environment Requirements....1347
4.71.3 Paths for Software Porting....1348
4.71.4 Configuring the Compilation Environment....1348
4.71.5 Obtaining the Source Code....1349
4.71.6 Compilation and Installation....1349
4.71.7 Running and Verification....1350
4.71.7.1 Compiling and Generating helloworld....1351
4.71.7.2 Compiling and Generating standarddialogs....1352
4.71.8 Troubleshooting....1352
4.71.8.1 Error Reported When Running the **./helloworld** Command....1352
4.71.9 More Resources....1353
4.72 RSeQC 4.0.0 Porting Guide (CentOS 8.2)....1353
4.72.1 Introduction....1353
4.72.2 Environment Requirements....1354
4.72.3 Paths for Software Porting....1355
4.72.4 Configuring the Compilation Environment....1355
4.72.4.1 Installing Python....1356

4.72.5 Compilation and Installation....1356
4.72.6 Running and Verification....1357
4.73 Sniffles 1.0.12 Porting Guide (CentOS 7.6)....1357
4.73.1 Introduction....1357
4.73.2 Environment Requirements....1357
4.73.3 Paths for Software Porting....1358
4.73.4 Configuring the Compilation Environment....1359
4.73.4.1 Configuration Description....1359
4.73.4.2 Configuring the Yum Source....1360
4.73.4.3 Installing the Dependencies....1360
4.73.5 Obtaining the Source Code....1360
4.73.6 Compilation and Installation....1361
4.73.7 Running and Verification....1361
4.73.8 More Resources....1362
4.74 SPIDER 24.08 Porting Guide (CentOS 7.6)....1362
4.74.1 Introduction....1362
4.74.2 Environment Requirements....1362
4.74.3 Paths for Software Porting....1363
4.74.4 Configuring the Compilation Environment....1364
4.74.4.1 Configuration Description....1364
4.74.4.2 Installing FFTW....1364
4.74.5 Obtaining the Source Code....1365
4.74.6 Compilation and Installation....1365
4.74.7 Running and Verification....1367
4.75 VEP 104.3 Porting Guide (CentOS 7.6)....1368
4.75.1 Introduction....1368
4.75.2 Environment Requirements....1369
4.75.3 Paths for Software Porting....1370
4.75.4 Configuring the Compilation Environment....1370
4.75.4.1 Installing HTSlib....1371
4.75.5 Obtaining the Source Code....1371
4.75.6 Compilation and Installation....1371
4.75.7 Running and Verification....1372
4.76 VTK 9.0.1 Porting Guide (openEuler 20.03)....1374
4.76.1 Introduction....1374
4.76.2 Environment Requirements....1374
4.76.3 Paths for Software Porting....1375
4.76.4 Configuring the Compilation Environment....1375
4.76.5 Obtaining the Source Code....1376
4.76.6 Compilation and Installation....1376
4.76.7 Running and Verification....1377
4.76.8 Troubleshooting....1377

 Copyright © Huawei Technologies Co., Ltd.

4.76.8.1 Error Reported During Test Case Compilation....1377
4.76.9 More Resources....1379
4.77 BWA 0.7.17 Porting Guide (openEuler 21.03)....1380
4.77.1 Introduction....1380
4.77.2 Environment Requirements....1380
4.77.3 Paths for Software Porting....1381
4.77.4 Configuring the Compilation Environment....1382
4.77.5 Obtaining the Source Code....1383
4.77.6 Compilation and Installation....1383
4.77.7 Running and Verification....1384
4.78 ABSOLUTE 1.0.6 Porting Guide (CentOS 7.6)....1385
4.78.1 Introduction....1385
4.78.2 Environment Requirements....1385
4.78.3 Paths for Software Porting....1386
4.78.4 Configuring the Compilation Environment....1387
4.78.5 Obtaining the Source Code....1387
4.78.6 Compilation and Installation....1387
4.78.7 Running and Verification....1388
4.79 GDS3D 1.9 Porting Guide (CentOS 7.6)....1388
4.79.1 Introduction....1388
4.79.2 Environment Requirements....1389
4.79.3 Paths for Software Porting....1389
4.79.4 Configuring the Compilation Environment....1390
4.79.5 Obtaining the Source Code....1390
4.79.6 Compilation and Installation....1391
4.79.7 Running and Verification....1391
4.80 HOMER 4.11 Porting Guide (CentOS 8.2)....1392
4.80.1 Introduction....1392
4.80.2 Environment Requirements....1393
4.80.3 Paths for Software Porting....1393
4.80.4 Configuring the Compilation Environment....1394
4.80.5 Obtaining the Source Code....1394
4.80.6 Compilation and Installation....1395
4.80.7 Running and Verification....1395
4.81 mCaller 1.0.3 Porting Guide (CentOS 8.2)....1396
4.81.1 Introduction....1396
4.81.2 Environment Requirements....1396
4.81.3 Paths for Software Porting....1397
4.81.4 Configuring the Compilation Environment....1397
4.81.4.1 Installing the Dependencies....1398
4.81.4.2 Installing Python....1398
4.81.5 Obtaining the Source Code....1399

4.81.6 Compilation and Installation....1399
4.81.7 Running and Verification....1399
4.82 Muscle 5.0.1428 Porting Guide (CentOS 7.9)....1400
4.82.1 Introduction....1400
4.82.2 Environment Requirements....1400
4.82.3 Paths for Software Porting....1401
4.82.4 Configuring the Compilation Environment....1402
4.82.5 Obtaining the Source Code....1402
4.82.6 Compilation and Installation....1402
4.82.7 Running and Verification....1427
4.83 Nanopolish 0.13.2 Porting Guide (CentOS 8.2)....1428
4.83.1 Introduction....1428
4.83.2 Environment Requirements....1429
4.83.3 Paths for Software Porting....1429
4.83.4 Configuring the Compilation Environment....1430
4.83.5 Obtaining the Source Code....1430
4.83.6 Compilation and Installation....1431
4.83.7 Running and Verification....1432
4.84 Octave 6.4.0 Porting Guide (CentOS 8.2)....1433
4.84.1 Introduction....1433
4.84.2 Environment Requirements....1433
4.84.3 Paths for Software Porting....1434
4.84.4 Configuring the Compilation Environment....1435
4.84.4.1 Installing the Dependencies....1435
4.84.5 Obtaining the Source Code....1436
4.84.6 Installing Octave....1436
4.85 PanPhlAn 3.1 Porting Guide (openEuler 20.03)....1437
4.85.1 Introduction....1437
4.85.2 Environment Requirements....1437
4.85.3 Paths for Software Porting....1438
4.85.4 Configuring the Compilation Environment....1439
4.85.5 Obtaining the Source Code....1439
4.85.6 Compilation and Installation....1440
4.85.7 Running and Verification....1440
4.86 PhyloPhlAn 3.0.60 Porting Guide (CentOS 8.2)....1441
4.86.1 Introduction....1441
4.86.2 Environment Requirements....1441
4.86.3 Paths for Software Porting....1442
4.86.4 Configuring the Compilation Environment....1443
4.86.4.1 Installing Python....1443
4.86.5 Compilation and Installation....1444
4.86.6 Running and Verification....1444

 Copyright © Huawei Technologies Co., Ltd.

4.87 PyClone 0.13.1 Porting Guide (CentOS 8.2) 1444
4.87.1 Introduction 1444
4.87.2 Environment Requirements 1445
4.87.3 Paths for Software Porting 1446
4.87.4 Configuring the Compilation Environment 1447
4.87.4.1 Installing Anaconda 1447
4.87.4.2 Installing PyDP 1448
4.87.4.3 Installing Numba 1449
4.87.4.4 Installing LLVM 1449
4.87.4.5 Installing llvmlite 1450
4.87.5 Obtaining the Source Code 1451
4.87.6 Compilation and Installation 1451
4.87.7 Running and Verification 1452
4.88 pyem v0.5 Porting Guide (CentOS 8.2) 1452
4.88.1 Introduction 1453
4.88.2 Environment Requirements 1453
4.88.3 Paths for Software Porting 1454
4.88.4 Configuring the Compilation Environment 1454
4.88.4.1 Installing Anaconda 1455
4.88.5 Obtaining the Source Code 1455
4.88.6 Compilation and Installation 1455
4.88.7 Running and Verification 1456
4.89 PyTorch 1.9 Porting Guide (CentOS 8.2) 1457
4.89.1 Introduction 1457
4.89.2 Environment Requirements 1457
4.89.3 Paths for Software Porting 1458
4.89.4 Configuring the Compilation Environment 1459
4.89.5 Obtaining the Source Code 1459
4.89.6 Compilation and Installation 1459
4.89.7 Running and Verification 1461
4.90 SURVIVOR 1.0.6 Porting Guide (CentOS 8.2) 1462
4.90.1 Introduction 1462
4.90.2 Environment Requirements 1462
4.90.3 Paths for Software Porting 1463
4.90.4 Configuring the Compilation Environment 1464
4.90.4.1 Installing GCC and CMake 1464
4.90.5 Obtaining the Source Code 1464
4.90.6 Compilation and Installation 1464
4.90.7 Running and Verification 1465
4.91 RELION 3.1.2 Porting Guide (CentOS 8.2) 1465
4.91.1 Introduction 1465
4.91.2 Environment Requirements 1466

 Copyright © Huawei Technologies Co., Ltd.

4.91.3 Paths for Software Porting....1467
4.91.4 Configuring the Compilation Environment....1468
4.91.4.1 Configuration Description....1468
4.91.4.2 Disabling the Nouveau Driver....1469
4.91.4.3 Installing NVIDIA CUDA....1469
4.91.4.4 Installing FFTW....1470
4.91.4.5 Installing CMake....1470
4.91.4.6 Installing wxWidgets....1471
4.91.4.7 Installing CTFFIND....1471
4.91.5 Obtaining the Source Code....1473
4.91.6 Compilation and Installation....1473
4.91.7 Running and Verification....1474
4.92 VMD 1.9.4 Porting Guide (CentOS 8.2)....1474
4.92.1 Introduction....1475
4.92.2 Environment Requirements....1475
4.92.3 Paths for Software Porting....1476
4.92.4 Configuring the Compilation Environment....1477
4.92.4.1 Disabling the Nouveau Driver....1477
4.92.4.2 Installing NVIDIA CUDA....1478
4.92.5 Obtaining the Source Code....1478
4.92.6 Compilation and Installation....1478
4.92.7 Running and Verification....1483
4.93 DBG2OLC Porting Guide (openEuler 20.03)....1483
4.93.1 Introduction....1483
4.93.2 Environmental Requirements....1484
4.93.3 Paths for Software Porting....1484
4.93.4 Configuring the Compilation Environment....1485
4.93.5 Obtaining the Source Code....1485
4.93.6 Compilation and Installation....1486
4.93.7 Running and Verification....1487
4.94 MAKER 2.31.11 Porting Guide (openEuler 20.03)....1490
4.94.1 Introduction....1490
4.94.2 Environmental Requirements....1491
4.94.3 Paths for Software Porting....1492
4.94.4 Configuring the Compilation Environment....1492
4.94.4.1 Configuration Description....1492
4.94.4.2 Installing GLib....1493
4.94.4.3 Installing Exonerate....1493
4.94.4.4 Installing MPICH....1494
4.94.4.5 Installing BLAST....1494
4.94.4.6 Installing RMBlast....1494
4.94.4.7 Installing SNAP....1495

 Copyright © Huawei Technologies Co., Ltd.

4.94.4.8 Installing TRF....1496
4.94.4.9 Installing RepeatMasker....1497
4.94.5 Obtaining the Source Code....1497
4.94.6 Compilation and Installation....1497
4.94.7 Running and Verification....1498
4.94.8 Troubleshooting....1500
4.94.8.1 An Error Similar to "cannot locate Text/Soundex.pm in *****" Is Reported When Running a Test Case....1500
4.95 Platanus 1.2.4 Porting Guide (openEuler 20.03)....1500
4.95.1 Introduction....1500
4.95.2 Environmental Requirements....1500
4.95.3 Paths for Software Porting....1501
4.95.4 Configuring the Compilation Environment....1502
4.95.5 Obtaining the Source Code....1502
4.95.6 Compilation and Installation....1503
4.95.7 Running and Verification....1505
4.96 TASSEL 5.0 Porting Guide (openEuler 20.03)....1505
4.96.1 Introduction....1505
4.96.2 Environmental Requirements....1506
4.96.3 Paths for Software Porting....1506
4.96.4 Obtaining the Source Code....1507
4.96.5 Compilation and Installation....1507
4.96.6 Running and Verification....1507
4.97 MWASTools 1.20.0 Porting Guide (Kylin V10)....1508
4.97.1 Introduction....1508
4.97.2 Environmental Requirements....1509
4.97.3 Paths for Software Porting....1509
4.97.4 Configuring the Compilation Environment....1510
4.97.4.1 Configuration Description....1510
4.97.4.2 Installing the R Language....1510
4.97.5 Obtaining the Source Code....1511
4.97.6 Compilation and Installation....1511
4.97.7 Running and Verification....1513
4.97.8 Troubleshooting....1513
4.97.8.1 Failed to Install the broom Dependency....1513
4.97.8.2 Failed to Install the Rgraphviz Dependency....1514
4.98 sRAP 1.26.0 Porting Guide (Kylin V10)....1515
4.98.1 Introduction....1515
4.98.2 Environmental Requirements....1516
4.98.3 Paths for Software Porting....1516
4.98.4 Configuring the Compilation Environment....1517
4.98.4.1 Configuration Description....1517
4.98.4.2 Installing the R Language....1518

 Copyright © Huawei Technologies Co., Ltd.

4.98.4.3 Installing BiocManager....1519
4.98.4.4 Installing WriteXLS....1519
4.98.4.5 Installing gplots....1520
4.98.4.6 Installing pls....1520
4.98.4.7 Installing ROCR....1521
4.98.5 Obtaining the Source Code....1522
4.98.6 Compilation and Installation....1522
4.98.7 Running and Verification....1523
4.99 BRAKER 2.1.6 Porting Guide (Kylin V10)....1524
4.99.1 Introduction....1524
4.99.2 Environmental Requirements....1525
4.99.3 Paths for Software Porting....1525
4.99.4 Configuring the Compilation Environment....1526
4.99.4.1 Configuration Description....1526
4.99.4.2 Installing Dependencies....1527
4.99.4.3 Installing GeneMark....1527
4.99.4.4 Installing CDBtools....1529
4.99.4.5 Installing ProtHint....1530
4.99.5 Obtaining the Source Code....1531
4.99.6 Compilation and Installation....1532
4.99.7 Running and Verification....1532
4.100 Cactus 2.1.0 Porting Guide (Kylin V10)....1533
4.100.1 Introduction....1533
4.100.2 Environmental Requirements....1533
4.100.3 Paths for Software Porting....1534
4.100.4 Configuring the Compilation Environment....1535
4.100.5 Obtaining the Source Code....1535
4.100.6 Compilation and Installation....1536
4.100.7 Running and Verification....1536
4.101 CAFE 4.2.1 Porting Guide (Kylin V10)....1537
4.101.1 Introduction....1537
4.101.2 Environmental Requirements....1537
4.101.3 Paths for Software Porting....1538
4.101.4 Configuring the Compilation Environment....1539
4.101.5 Obtaining the Source Code....1539
4.101.6 Compilation and Installation....1539
4.101.7 Running and Verification....1545
4.102 Clustal X 2.1 Porting Guide (Kylin V10)....1546
4.102.1 Introduction....1546
4.102.2 Environmental Requirements....1546
4.102.3 Paths for Software Porting....1547
4.102.4 Configuring the Compilation Environment....1548

4.102.5 Obtaining the Source Code ... 1548
4.102.6 Compilation and Installation ... 1548
4.103 CNVfilteR 1.10.0 Porting Guide (Kylin V10) ... 1549
4.103.1 Introduction ... 1549
4.103.2 Environmental Requirements ... 1549
4.103.3 Paths for Software Porting ... 1550
4.103.4 Configuring the Compilation Environment ... 1551
4.103.4.1 Configuration Description ... 1551
4.103.4.2 Installing the R Language ... 1551
4.103.5 Obtaining the Source Code ... 1552
4.103.6 Compilation and Installation ... 1552
4.103.7 Running and Verification ... 1553
4.104 DIAMOND 2.0.15 Porting Guide (Kylin V10) ... 1554
4.104.1 Introduction ... 1554
4.104.2 Environmental Requirements ... 1554
4.104.3 Paths for Software Porting ... 1555
4.104.4 Configuring the Compilation Environment ... 1556
4.104.5 Obtaining the Source Code ... 1556
4.104.6 Compilation and Installation ... 1556
4.104.7 Running and Verification ... 1557
4.105 Exonerate 2.4.0 Porting Guide (Kylin V10) ... 1557
4.105.1 Introduction ... 1557
4.105.2 Environmental Requirements ... 1558
4.105.3 Paths for Software Porting ... 1559
4.105.4 Configuring the Compilation Environment ... 1559
4.105.5 Obtaining the Source Code ... 1559
4.105.6 Compilation and Installation ... 1560
4.105.7 Running and Verification ... 1560
4.106 FreeBayes 1.3.6 Porting Guide (Kylin V10) ... 1561
4.106.1 Introduction ... 1561
4.106.2 Environmental Requirements ... 1561
4.106.3 Paths for Software Porting ... 1562
4.106.4 Configuring the Compilation Environment ... 1563
4.106.4.1 Configuration Description ... 1563
4.106.4.2 Installing Parallel ... 1564
4.106.5 Obtaining the Source Code ... 1564
4.106.6 Compilation and Installation ... 1564
4.106.7 Running and Verification ... 1565
4.107 GMAP 2021-12-17 Porting Guide (Kylin V10) ... 1565
4.107.1 Introduction ... 1565
4.107.2 Environmental Requirements ... 1566
4.107.3 Paths for Software Porting ... 1567

 Copyright © Huawei Technologies Co., Ltd.

4.107.4 Configuring the Compilation Environment....1567
4.107.5 Obtaining the Source Code....1568
4.107.6 Compilation and Installation....1568
4.107.7 Running and Verification....1568
4.108 GUIDEseq 1.26.0 Porting Guide (Kylin V10)....1569
4.108.1 Introduction....1569
4.108.2 Environmental Requirements....1569
4.108.3 Paths for Software Porting....1570
4.108.4 Configuring the Compilation Environment....1571
4.108.4.1 Configuration Description....1571
4.108.4.2 Installing the R Language....1571
4.108.5 Obtaining the Source Code....1572
4.108.6 Compilation and Installation....1572
4.108.7 Running and Verification....1574
4.109 HiCUP v0.8.3 Porting Guide (Kylin V10)....1574
4.109.1 Introduction....1574
4.109.2 Environmental Requirements....1575
4.109.3 Paths for Software Porting....1576
4.109.4 Configuring the Compilation Environment....1576
4.109.5 Obtaining the Source Code....1577
4.109.6 Compilation and Installation....1577
4.109.7 Running and Verification....1577
4.110 Jellyfish 2.3.0 Porting Guide (Kylin V10)....1578
4.110.1 Introduction....1578
4.110.2 Environmental Requirements....1578
4.110.3 Paths for Software Porting....1579
4.110.4 Configuring the Compilation Environment....1580
4.110.5 Obtaining the Source Code....1580
4.110.6 Compilation and Installation....1581
4.110.7 Running and Verification....1581
4.111 KOBAS 3.0.3 Porting Guide (Kylin V10)....1581
4.111.1 Introduction....1581
4.111.2 Environmental Requirements....1582
4.111.3 Paths for Software Porting....1582
4.111.4 Configuring the Compilation Environment....1583
4.111.5 Obtaining the Source Code....1583
4.111.6 Compilation and Installation....1584
4.111.7 Running and Verification....1584
4.112 LTR_Finder Porting Guide (Kylin V10)....1585
4.112.1 Introduction....1585
4.112.2 Environmental Requirements....1585
4.112.3 Paths for Software Porting....1586

 Copyright © Huawei Technologies Co., Ltd.

4.112.4 Configuring the Compilation Environment....1587
4.112.5 Obtaining the Source Code....1587
4.112.6 Compilation and Installation....1588
4.113 LTR_retriever 2.9.0 Porting Guide (Kylin V10)....1588
4.113.1 Introduction....1588
4.113.2 Environmental Requirements....1588
4.113.3 Paths for Software Porting....1589
4.113.4 Configuring the Compilation Environment....1590
4.113.4.1 Configuration Description....1590
4.113.4.2 Installing Perl....1591
4.113.4.3 Installing GenomeTools....1591
4.113.4.4 Installing CD-HIT....1591
4.113.5 Obtaining the Source Code....1591
4.113.6 Compilation and Installation....1592
4.113.7 Running and Verification....1592
4.114 MAFFT 7.490 Porting Guide (Kylin V10)....1593
4.114.1 Introduction....1593
4.114.2 Environmental Requirements....1593
4.114.3 Paths for Software Porting....1594
4.114.4 Configuring the Compilation Environment....1595
4.114.5 Obtaining the Source Code....1595
4.114.6 Compilation and Installation....1596
4.115 Meryl 1.3 Porting Guide (Kylin V10)....1596
4.115.1 Introduction....1596
4.115.2 Environmental Requirements....1596
4.115.3 Paths for Software Porting....1597
4.115.4 Configuring the Compilation Environment....1598
4.115.5 Obtaining the Source Code....1598
4.115.6 Compilation and Installation....1598
4.116 Minigraph 0.19 Porting Guide (Kylin V10)....1599
4.116.1 Introduction....1599
4.116.2 Environmental Requirements....1599
4.116.3 Paths for Software Porting....1600
4.116.4 Configuring the Compilation Environment....1601
4.116.5 Obtaining the Source Code....1601
4.116.6 Compilation and Installation....1601
4.116.7 Running and Verification....1601
4.117 MISA 25082020 Porting Guide (Kylin V10)....1602
4.117.1 Introduction....1602
4.117.2 Environmental Requirements....1602
4.117.3 Paths for Software Porting....1603
4.117.4 Configuring the Compilation Environment....1604

4.117.5 Obtaining the Source Code ... 1604
4.117.6 Compilation and Installation ... 1604
4.117.7 Running and Verification ... 1605
4.118 MODELTEST 0.1.7 Porting Guide (Kylin V10) ... 1605
4.118.1 Introduction ... 1605
4.118.2 Environmental Requirements ... 1605
4.118.3 Paths for Software Porting ... 1606
4.118.4 Configuring the Compilation Environment ... 1607
4.118.5 Obtaining the Source Code ... 1607
4.118.6 Compilation and Installation ... 1607
4.118.7 Running and Verification ... 1608
4.119 NINJA 1.2.2 Porting Guide (Kylin V10) ... 1608
4.119.1 Introduction ... 1608
4.119.2 Environmental Requirements ... 1609
4.119.3 Paths for Software Porting ... 1610
4.119.4 Configuring the Compilation Environment ... 1610
4.119.5 Obtaining the Source Code ... 1610
4.119.6 Compilation and Installation ... 1611
4.119.7 Running and Verification ... 1611
4.120 NOVOPlasty 4.3.1 Porting Guide (Kylin V10) ... 1612
4.120.1 Introduction ... 1612
4.120.2 Environmental Requirements ... 1612
4.120.3 Paths for Software Porting ... 1613
4.120.4 Configuring the Compilation Environment ... 1613
4.120.5 Obtaining the Source Code ... 1614
4.120.6 Compilation and Installation ... 1614
4.120.7 Running and Verification ... 1614
4.121 OrthoMCL 2.0.9 Porting Guide (Kylin V10) ... 1615
4.121.1 Introduction ... 1615
4.121.2 Environmental Requirements ... 1615
4.121.3 Paths for Software Porting ... 1616
4.121.4 Configuring the Compilation Environment ... 1616
4.121.5 Obtaining the Source Code ... 1617
4.121.6 Running and Verification ... 1617
4.122 PAL2NAL 14 Porting Guide (Kylin V10) ... 1618
4.122.1 Introduction ... 1618
4.122.2 Environmental Requirements ... 1618
4.122.3 Paths for Software Porting ... 1619
4.122.4 Configuring the Compilation Environment ... 1619
4.122.5 Obtaining the Source Code ... 1620
4.122.6 Compilation and Installation ... 1620
4.122.7 Running and Verification ... 1620

 Copyright © Huawei Technologies Co., Ltd.

4.123 PASA r20130907 Porting Guide (Kylin V10) 1621
4.123.1 Introduction 1621
4.123.2 Environmental Requirements 1621
4.123.3 Paths for Software Porting 1622
4.123.4 Configuring the Compilation Environment 1622
4.123.5 Obtaining the Source Code 1623
4.123.6 Compilation and Installation 1623
4.123.7 Running and Verification 1624
4.124 Picard 2.27.3 Porting Guide (Kylin V10) 1624
4.124.1 Introduction 1624
4.124.2 Environmental Requirements 1624
4.124.3 Paths for Software Porting 1625
4.124.4 Configuring the Compilation Environment 1626
4.124.5 Obtaining the Source Code 1626
4.124.6 Compilation and Installation 1626
4.124.7 Running and Verification 1627
4.124.8 Troubleshooting 1627
4.124.8.1 Failed to Download gradle-5.6-bin.zip 1627
4.125 ProtTest 3.4.2 Porting Guide (Kylin V10) 1628
4.125.1 Introduction 1628
4.125.2 Environmental Requirements 1628
4.125.3 Paths for Software Porting 1629
4.125.4 Configuring the Compilation Environment 1630
4.125.4.1 Configuration Description 1630
4.125.4.2 Installing PhyML 1630
4.125.5 Obtaining the Source Code 1631
4.125.6 Compilation and Installation 1631
4.125.7 Running and Verification 1631
4.126 Purge_Dups 1.2.6 Porting Guide (Kylin V10) 1632
4.126.1 Introduction 1632
4.126.2 Environmental Requirements 1632
4.126.3 Paths for Software Porting 1633
4.126.4 Configuring the Compilation Environment 1634
4.126.5 Obtaining the Source Code 1634
4.126.6 Compilation and Installation 1634
4.126.7 Running and Verification 1635
4.127 Ragout 2.3 Porting Guide (Kylin V10) 1635
4.127.1 Introduction 1635
4.127.2 Environmental Requirements 1636
4.127.3 Paths for Software Porting 1636
4.127.4 Configuring the Compilation Environment 1637
4.127.5 Obtaining the Source Code 1637

 Copyright © Huawei Technologies Co., Ltd.

4.127.6 Compilation and Installation 1638
4.127.7 Running and Verification 1638
4.128 RepeatMasker 4.1.2 Porting Guide (Kylin V10) 1639
4.128.1 Introduction 1639
4.128.2 Environmental Requirements 1639
4.128.3 Paths for Software Porting 1640
4.128.4 Configuring the Compilation Environment 1641
4.128.4.1 Configuration Description 1641
4.128.4.2 Installing Perl 1641
4.128.4.3 Installing TRF 1642
4.128.5 Obtaining the Source Code 1642
4.128.6 Compilation and Installation 1642
4.128.7 Running and Verification 1644
4.129 RepeatModeler 2.0.1 Porting Guide (Kylin V10) 1644
4.129.1 Introduction 1644
4.129.2 Environmental Requirements 1645
4.129.3 Paths for Software Porting 1645
4.129.4 Configuring the Compilation Environment 1646
4.129.4.1 Configuration Description 1646
4.129.4.2 Installing Perl 1647
4.129.4.3 Installing RECON 1647
4.129.5 Obtaining the Source Code 1647
4.129.6 Compilation and Installation 1647
4.129.7 Running and Verification 1650
4.130 RepeatScout 1.0.6 Porting Guide (Kylin V10) 1651
4.130.1 Introduction 1651
4.130.2 Environmental Requirements 1652
4.130.3 Paths for Software Porting 1652
4.130.4 Configuring the Compilation Environment 1653
4.130.5 Obtaining the Source Code 1653
4.130.6 Compilation and Installation 1654
4.130.7 Running and Verification 1654
4.131 RMBlast 2.10.0 Porting Guide (Kylin V10) 1655
4.131.1 Introduction 1655
4.131.2 Environmental Requirements 1655
4.131.3 Paths for Software Porting 1656
4.131.4 Configuring the Compilation Environment 1657
4.131.5 Obtaining the Source Code 1657
4.131.6 Compilation and Installation 1657
4.131.7 Running and Verification 1658
4.132 RSEM 1.3.3 Porting Guide (Kylin V10) 1659
4.132.1 Introduction 1659

 Copyright © Huawei Technologies Co., Ltd.

4.132.2 Environmental Requirements 1660
4.132.3 Paths for Software Porting 1660
4.132.4 Configuring the Compilation Environment 1661
4.132.5 Obtaining the Source Code 1661
4.132.6 Compilation and Installation 1662
4.132.7 Running and Verification 1662
4.133 CANU 1.8 Porting Guide (CentOS 7.6) 1662
4.133.1 Introduction 1663
4.133.2 Environment Requirements 1663
4.133.3 Paths for Software Porting 1663
4.133.4 Configuring the Compilation Environment 1664
4.133.5 Obtaining the Source Code 1664
4.133.6 Compiling and Installing CANU 1664
4.133.7 Running and Verifying CANU 1665
4.134 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6) 1666
4.134.1 Introduction 1666
4.134.2 Environment Requirements 1667
4.134.3 Planning the Paths for Software Porting 1667
4.134.4 Configuring the Compilation Environment 1668
4.134.4.1 Configuration Description 1668
4.134.4.2 Installing Dependencies 1669
4.134.4.3 Installing Python 1669
4.134.4.4 Installing PIP 1670
4.134.4.5 Installing OpenBLAS 1670
4.134.4.6 Installing HDF5 1671
4.134.5 Obtaining the Source Code 1671
4.134.6 Compiling and Installing Ont-Tombo 1671
4.135 CPAT 3.0.4 Porting Guide (Kylin V10) 1672
4.135.1 Introduction 1672
4.135.2 Environment Requirements 1672
4.135.3 Paths for Software Porting 1673
4.135.4 Configuring the Compilation Environment 1673
4.135.5 Compilation and Installation 1674
4.135.6 Running and Verification 1674
4.136 MrBayes 3.2.7a Porting Guide (Kylin V10) 1674
4.136.1 Introduction 1674
4.136.2 Environment Requirements 1675
4.136.3 Paths for Software Porting 1676
4.136.4 Configuring the Compilation Environment 1676
4.136.5 Obtaining the Source Code 1677
4.136.6 Compilation and Installation 1677
4.136.7 Running and Verification 1677

 Copyright © Huawei Technologies Co., Ltd.

4.137 MSMC 1.1.0 Porting Guide (Kylin V10) 1678
4.137.1 Introduction 1678
4.137.2 Environment Requirements 1678
4.137.3 Paths for Software Porting 1679
4.137.4 Configuring the Compilation Environment 1680
4.137.4.1 Configuration Description 1680
4.137.4.2 Installing GSL 1680
4.137.4.3 Installing the D Compiler 1681
4.137.5 Obtaining the Source Code 1681
4.137.6 Compilation and Installation 1681
4.137.7 Running and Verification 1681
4.138 Prodigal 2.6.3 Porting Guide (Kylin V10) 1682
4.138.1 Introduction 1682
4.138.2 Environment Requirements 1683
4.138.3 Paths for Software Porting 1684
4.138.4 Configuring the Compilation Environment 1684
4.138.5 Obtaining the Source Code 1685
4.138.6 Compilation and Installation 1685
4.138.7 Running and Verification 1685
4.139 Vaa3D 2.2.0.0 Porting Guide (openEuler 20.03) 1686
4.139.1 Introduction 1686
4.139.2 Environment Requirements 1687
4.139.3 Paths for Software Porting 1687
4.139.4 Configuring the Compilation Environment 1688
4.139.4.1 Configuration Description 1688
4.139.4.2 Installing szip 1688
4.139.4.3 Installing teem 1689
4.139.5 Compilation and Installation 1689
4.139.6 Running and Verification 1690
4.140 WGCNA 1.70 Porting Guide (Kylin V10) 1691
4.140.1 Introduction 1691
4.140.2 Environment Requirements 1691
4.140.3 Paths for Software Porting 1692
4.140.4 Configuring the Compilation Environment 1692
4.140.4.1 Configuration Description 1692
4.140.4.2 Installing R-4.2.0 1693
4.140.5 Obtaining the Source Code 1693
4.140.6 Compilation and Installation 1694
4.140.7 Running and Verification 1695
4.141 AutoDock 4.2.6 Porting Guide (Kylin V10) 1695
4.141.1 Introduction 1695
4.141.2 Environment Requirements 1696

 Copyright © Huawei Technologies Co., Ltd.

4.141.3 Paths for Software Porting 1697
4.141.4 Configuring the Compilation Environment 1697
4.141.5 Obtaining the Source Code 1698
4.141.6 Compilation and Installation 1698
4.141.7 Running and Verification 1698
4.142 bioconductor-manta 1.32.0 Porting Guide (Kylin V10) 1699
4.142.1 Introduction 1699
4.142.2 Environment Requirements 1699
4.142.3 Paths for Software Porting 1700
4.142.4 Configuring the Compilation Environment 1701
4.142.4.1 Configuration Description 1701
4.142.4.2 Installing R-4.2.0 1701
4.142.5 Obtaining the Source Code 1702
4.142.6 Compilation and Installation 1702
4.142.7 Running and Verification 1703
4.143 CellSys 0.1.0 Porting Guide (Kylin V10) 1704
4.143.1 Introduction 1704
4.143.2 Environment Requirements 1704
4.143.3 Paths for Software Porting 1705
4.143.4 Configuring the Compilation Environment 1705
4.143.4.1 Configuration Description 1706
4.143.4.2 Installing Dependencies 1706
4.143.5 Obtaining the Source Code 1706
4.143.6 Compilation and Installation 1706
4.143.7 Running and Verification 1707
4.144 cutadapt 4.1 Porting Guide (Kylin V10) 1707
4.144.1 Introduction 1708
4.144.2 Environment Requirements 1708
4.144.3 Paths for Software Porting 1709
4.144.4 Configuring the Compilation Environment 1709
4.144.4.1 Configuration Description 1709
4.144.4.2 Installing Dependencies 1710
4.144.5 Obtaining the Source Code 1710
4.144.6 Compilation and Installation 1710
4.144.7 Running and Verification 1711
4.145 deepTools 3.5.1 Porting Guide (Kylin V10) 1711
4.145.1 Introduction 1711
4.145.2 Environment Requirements 1712
4.145.3 Paths for Software Porting 1712
4.145.4 Configuring the Compilation Environment 1713
4.145.4.1 Configuration Description 1713
4.145.4.2 Installing Python 3 1713

 Copyright © Huawei Technologies Co., Ltd.

4.145.5 Obtaining the Source Code 1714
4.145.6 Compilation and Installation 1714
4.145.7 Running and Verification 1714
4.146 DESeq2 1.36.0 Porting Guide (Kylin V10) 1715
4.146.1 Introduction 1715
4.146.2 Environment Requirements 1715
4.146.3 Paths for Software Porting 1716
4.146.4 Configuring the Compilation Environment 1717
4.146.4.1 Configuration Description 1717
4.146.4.2 Installing R-4.2.0 1717
4.146.5 Obtaining the Source Code 1718
4.146.6 Compilation and Installation 1718
4.146.7 Running and Verification 1719
4.147 featurecounts 2.0.3 Porting Guide (Kylin V10) 1719
4.147.1 Introduction 1719
4.147.2 Environment Requirements 1720
4.147.3 Paths for Software Porting 1720
4.147.4 Configuring the Compilation Environment 1721
4.147.5 Obtaining the Source Code 1721
4.147.6 Compilation and Installation 1722
4.147.7 Running and Verification 1722
4.148 ggplot2 3.3.6 Porting Guide (Kylin V10) 1723
4.148.1 Introduction 1723
4.148.2 Environment Requirements 1723
4.148.3 Paths for Software Porting 1724
4.148.4 Configuring the Compilation Environment 1724
4.148.4.1 Configuration Description 1725
4.148.4.2 Installing R-4.2.0 1725
4.148.5 Obtaining the Source Code 1726
4.148.6 Compilation and Installation 1726
4.148.7 Running and Verification 1726
4.149 MultiQC 1.12 Porting Guide (Kylin V10) 1727
4.149.1 Introduction 1727
4.149.2 Environment Requirements 1727
4.149.3 Paths for Software Porting 1728
4.149.4 Configuring the Compilation Environment 1729
4.149.4.1 Configuration Description 1729
4.149.4.2 Installing Python 3 1729
4.149.5 Obtaining the Source Code 1730
4.149.6 Compilation and Installation 1730
4.149.7 Running and Verification 1730
4.150 Open Babel 3.1.1 Porting Guide (Kylin V10) 1731

 Copyright © Huawei Technologies Co., Ltd.

4.150.1 Introduction 1731
4.150.2 Environment Requirements 1731
4.150.3 Paths for Software Porting 1732
4.150.4 Configuring the Compilation Environment 1733
4.150.5 Obtaining the Source Code 1733
4.150.6 Compilation and Installation 1733
4.150.7 Running and Verification 1734
4.151 PBSuite 15.8.24 Porting Guide (Kylin V10) 1734
4.151.1 Introduction 1734
4.151.2 Environment Requirements 1735
4.151.3 Paths for Software Porting 1736
4.151.4 Configuring the Compilation Environment 1736
4.151.4.1 Configuration Description 1736
4.151.4.2 Installing Python 2 1737
4.151.4.3 Installing Dependencies 1737
4.151.5 Obtaining the Source Code 1738
4.151.6 Compilation and Installation 1738
4.151.7 Running and Verification 1738
4.152 psipred 4.0 Porting Guide (Kylin V10) 1739
4.152.1 Introduction 1739
4.152.2 Environment Requirements 1740
4.152.3 Paths for Software Porting 1740
4.152.4 Configuring the Compilation Environment 1741
4.152.4.1 Configuration Description 1741
4.152.4.2 Installing blast+ 1742
4.152.5 Obtaining the Source Code 1742
4.152.6 Compilation and Installation 1742
4.152.7 Running and Verification 1743
4.153 SNPhylo 20180901 Porting Guide (Kylin V10) 1744
4.153.1 Introduction 1744
4.153.2 Environment Requirements 1744
4.153.3 Paths for Software Porting 1745
4.153.4 Configuring the Compilation Environment 1745
4.153.4.1 Configuration Description 1745
4.153.4.2 Installing R-4.2.0 1746
4.153.4.3 Installing Dependencies 1747
4.153.4.4 Installing Python 2 1747
4.153.4.5 Installing phylip 1748
4.153.5 Obtaining the Source Code 1748
4.153.6 Compilation and Installation 1748
4.153.7 Running and Verification 1749
4.154 TreeBeST 1.9.2 Porting Guide (Kylin V10) 1749

 Copyright © Huawei Technologies Co., Ltd.

4.154.1 Introduction 1750
4.154.2 Environment Requirements 1750
4.154.3 Paths for Software Porting 1751
4.154.4 Configuring the Compilation Environment 1751
4.154.5 Obtaining the Source Code 1752
4.154.6 Compilation and Installation 1752
4.154.7 Running and Verification 1753
4.155 BLASR Porting Guide (Kylin V10) 1753
4.155.1 Introduction 1753
4.155.2 Environment Requirements 1754
4.155.3 Paths for Software Porting 1755
4.155.4 Configuring the Compilation Environment 1755
4.155.5 Obtaining the Source Code 1756
4.155.6 Compilation and Installation 1756
4.155.7 Running and Verification 1756
4.156 BreakDancer Porting Guide (Kylin V10) 1757
4.156.1 Introduction 1757
4.156.2 Environment Requirements 1757
4.156.3 Paths for Software Porting 1758
4.156.4 Configuring the Compilation Environment 1759
4.156.5 Obtaining the Source Code 1759
4.156.6 Compilation and Installation 1759
4.156.7 Running and Verification 1760
4.157 Mugsy 1.2.3 Porting Guide (Kylin V10) 1760
4.157.1 Introduction 1760
4.157.2 Environment Requirements 1760
4.157.3 Paths for Software Porting 1761
4.157.4 Configuring the Compilation Environment 1762
4.157.4.1 Configuration Description 1762
4.157.4.2 Installing Anaconda 1762
4.157.5 Obtaining the Source Code 1763
4.157.6 Compilation and Installation 1763
4.157.7 Running and Verification 1764
4.158 AlphaFold Porting Guide (CentOS 8.2) 1764
4.158.1 Introduction 1765
4.158.2 Environment Requirements 1765
4.158.3 Paths for Software Porting 1766
4.158.4 Configuring the Compilation Environment 1767
4.158.4.1 Configuration Description 1767
4.158.4.2 Installing Conda 1768
4.158.4.3 Installing Bazel 1768
4.158.4.4 Installing TensorFlow 1769

 Copyright © Huawei Technologies Co., Ltd.

4.158.4.5 Installing jaxlib 1770
4.158.4.6 Install AlphaFold Dependencies 1771
4.158.4.7 Installing NVIDIA CUDA 1771
4.158.5 Obtaining the Source Code 1772
4.158.6 Compilation and Installation 1772
4.158.7 Running and Verification 1773
4.159 Glimmer v3.02 Porting Guide (Kylin V10) 1773
4.159.1 Introduction 1773
4.159.2 Environment Requirements 1773
4.159.3 Paths for Software Porting 1774
4.159.4 Configuring the Compilation Environment 1775
4.159.5 Obtaining the Source Code 1775
4.159.6 Compilation and Installation 1776
4.159.7 Running and Verification 1776
4.160 HaploView 4.1 Porting Guide (Kylin V10) 1777
4.160.1 Introduction 1777
4.160.2 Environment Requirements 1778
4.160.3 Paths for Software Porting 1778
4.160.4 Configuring the Compilation Environment 1779
4.160.5 Obtaining the Source Code 1779
4.160.6 Compilation and Installation 1780
4.160.7 Running and Verification 1780
4.161 HH-suite v3.3.0 Porting Guide (Kylin V10) 1781
4.161.1 Introduction 1781
4.161.2 Environment Requirements 1781
4.161.3 Paths for Software Porting 1782
4.161.4 Configuring the Compilation Environment 1782
4.161.4.1 Configuration Description 1782
4.161.4.2 Installing CMake 1783
4.161.5 Obtaining the Source Code 1783
4.161.6 Compilation and Installation 1784
4.161.7 Running and Verification 1784
4.161.8 Troubleshooting 1785
4.161.8.1 An Error Is Reported When Running MPI as the Root User 1785
4.162 art-nextgen-simulation-tools 20160605 Porting Guide (Kylin V10) 1785
4.162.1 Introduction 1785
4.162.2 Environment Requirements 1786
4.162.3 Paths for Software Porting 1787
4.162.4 Configuring the Compilation Environment 1787
4.162.4.1 Configuration Description 1787
4.162.4.2 Installing Dependencies 1788
4.162.5 Obtaining the Source Code 1788

 Copyright © Huawei Technologies Co., Ltd.

4.162.6 Compilation and Installation 1788
4.162.7 Running and Verification 1789
4.162.8 Change History 1789
4.163 bioconductor-appreci8r 1.14.0 Porting Guide (Kylin V10) 1789
4.163.1 Introduction 1789
4.163.2 Environment Requirements 1790
4.163.3 Paths for Software Porting 1791
4.163.4 Configuring the Compilation Environment 1791
4.163.4.1 Configuration Description 1791
4.163.4.2 Installing R 4.2.0 1792
4.163.5 Obtaining the Source Code 1792
4.163.6 Compilation and Installation 1792
4.163.7 Running and Verification 1794
4.163.8 Troubleshooting 1794
4.163.8.1 Failed to Download the Installation Package of the Dependency Software 1795
4.163.9 Change History 1795
4.164 ANGSD 0.938 Porting Guide (Kylin V10) 1795
4.164.1 Introduction 1795
4.164.2 Environment Requirements 1796
4.164.3 Paths for Software Porting 1797
4.164.4 Configuring the Compilation Environment 1797
4.164.4.1 Configuration Description 1797
4.164.4.2 Installing HTSlib 1.16 1797
4.164.5 Obtaining the Source Code 1798
4.164.6 Compilation and Installation 1798
4.164.7 Running and Verification 1798
4.164.8 Change History 1799
4.165 bioconductor-decontam 1.16.0 Porting Guide (Kylin V10) 1799
4.165.1 Introduction 1799
4.165.2 Environment Requirements 1800
4.165.3 Paths for Software Porting 1801
4.165.4 Configuring the Compilation Environment 1801
4.165.4.1 Configuration Description 1801
4.165.4.2 Installing R 4.2.0 1801
4.165.5 Obtaining the Source Code 1802
4.165.6 Compilation and Installation 1802
4.165.7 Running and Verification 1804
4.165.8 Change History 1804
4.166 bioconductor-clippda 1.46.0 Porting Guide (Kylin V10) 1805
4.166.1 Introduction 1805
4.166.2 Environment Requirements 1805
4.166.3 Paths for Software Porting 1806

 Copyright © Huawei Technologies Co., Ltd.

4.166.4 Configuring the Compilation Environment....1806
4.166.4.1 Configuration Description....1806
4.166.4.2 Installing R 4.2.0....1807
4.166.5 Compilation and Installation....1807
4.166.6 Running and Verification....1809
4.166.7 Change History....1809
4.167 bioconductor-bubbletree 2.26.0 Porting Guide (Kylin V10)....1809
4.167.1 Introduction....1809
4.167.2 Environment Requirements....1810
4.167.3 Paths for Software Porting....1811
4.167.4 Configuring the Compilation Environment....1811
4.167.4.1 Configuration Description....1811
4.167.4.2 Installing the R Language....1811
4.167.5 Obtaining the Source Code....1812
4.167.6 Compilation and Installation....1812
4.167.7 Running and Verification....1813
4.167.8 Change History....1814
4.168 AMAS 1.2 Porting Guide (Kylin V10)....1814
4.168.1 Introduction....1814
4.168.2 Environment Requirements....1815
4.168.3 Paths for Software Porting....1815
4.168.4 Obtaining the Source Code....1816
4.168.5 Compilation and Installation....1816
4.168.6 Running and Verification....1816
4.168.7 Change History....1817
4.169 bioconductor-ebcoexpress 1.42.0 Porting Guide (Kylin V10)....1817
4.169.1 Introduction....1817
4.169.2 Environment Requirements....1817
4.169.3 Paths for Software Porting....1818
4.169.4 Configuring the Compilation Environment....1818
4.169.4.1 Configuration Description....1819
4.169.4.2 Installing the R Language....1819
4.169.5 Obtaining the Source Code....1820
4.169.6 Compilation and Installation....1820
4.169.7 Running and Verification....1821
4.169.8 Change History....1821
4.170 athena_meta 1.3 Porting Guide (Kylin V10)....1821
4.170.1 Introduction....1821
4.170.2 Environment Requirements....1822
4.170.3 Paths for Software Porting....1822
4.170.4 Configuring the Compilation Environment....1823
4.170.4.1 Configuration Description....1823

 Copyright © Huawei Technologies Co., Ltd.

4.170.4.2 Installing Python 2.7 1824
4.170.4.3 Installing IDBA 1.1.3a1 1824
4.170.4.4 Installing Samtools 1.16.1 1824
4.170.4.5 Installing HTSlib 1.16 1825
4.170.4.6 Installing minimap2 1825
4.170.4.7 Installing BWA 0.7.17 1825
4.170.4.8 Installing Flye 2.3.1 1826
4.170.5 Obtaining the Source Code 1826
4.170.6 Compilation and Installation 1826
4.170.7 Running and Verification 1827
4.170.8 Change History 1828
4.171 bioconductor-GARS 1.16.0 Porting Guide (Kylin V10) 1828
4.171.1 Introduction 1828
4.171.2 Environment Requirements 1828
4.171.3 Paths for Software Porting 1829
4.171.4 Configuring the Compilation Environment 1829
4.171.4.1 Configuration Description 1829
4.171.4.2 Installing R 4.2.0 1830
4.171.5 Obtaining the Source Code 1830
4.171.6 Compilation and Installation 1830
4.171.7 Running and Verification 1832
4.171.8 Troubleshooting 1833
4.171.8.1 Failed to Download the Installation Package of the Dependency Software 1833
4.171.9 Change History 1833
4.172 CD-HIT 4.8.1 Porting Guide (Kylin V10) 1833
4.172.1 Introduction 1833
4.172.2 Environment Requirements 1834
4.172.3 Paths for Software Porting 1835
4.172.4 Configuring the Compilation Environment 1835
4.172.5 Obtaining the Source Code 1836
4.172.6 Compilation and Installation 1836
4.172.7 Running and Verification 1836
4.173 bioconductor-grohmm 1.30.1 Porting Guide (Kylin V10) 1837
4.173.1 Introduction 1837
4.173.2 Environment Requirements 1837
4.173.3 Paths for Software Porting 1838
4.173.4 Configuring the Compilation Environment 1838
4.173.4.1 Configuration Description 1839
4.173.4.2 Installing R 4.2.0 1839
4.173.5 Obtaining the Source Code 1839
4.173.6 Compilation and Installation 1840
4.173.7 Running and Verification 1841

 Copyright © Huawei Technologies Co., Ltd.

4.174 bioconductor-MAGeCKFlute 2.0.0 Porting Guide (Kylin V10) 1842
4.174.1 Introduction 1842
4.174.2 Environment Requirements 1842
4.174.3 Paths for Software Porting 1843
4.174.4 Configuring the Compilation Environment 1844
4.174.4.1 Configuration Description 1844
4.174.4.2 Installing R 4.2.0 1844
4.174.5 Obtaining the Source Code 1845
4.174.6 Compilation and Installation 1845
4.174.7 Running and Verification 1846
4.174.8 Troubleshooting 1846
4.174.8.1 Failed to Download the Installation Package of the Dependency Software 1846
4.175 bioconductor-intansv 1.36.0 Porting Guide (Kylin V10) 1847
4.175.1 Introduction 1847
4.175.2 Environment Requirements 1847
4.175.3 Paths for Software Porting 1848
4.175.4 Configuring the Compilation Environment 1849
4.175.4.1 Configuration Description 1849
4.175.4.2 Installing the R Language 1849
4.175.5 Obtaining the Source Code 1850
4.175.6 Compilation and Installation 1850
4.175.7 Running and Verification 1851
4.175.8 Change History 1851
4.176 bioconductor-multihiccompare 1.14.0 Porting Guide (Kylin V10) 1851
4.176.1 Introduction 1851
4.176.2 Environment Requirements 1852
4.176.3 Paths for Software Porting 1853
4.176.4 Configuring the Compilation Environment 1853
4.176.4.1 Configuration Description 1853
4.176.4.2 Installing R 4.2.0 1854
4.176.5 Obtaining the Source Code 1854
4.176.6 Compilation and Installation 1854
4.176.7 Running and Verification 1856
4.177 ClustalW 2.1 Porting Guide (Kylin V10) 1856
4.177.1 Introduction 1857
4.177.2 Environment Requirements 1857
4.177.3 Paths for Software Porting 1858
4.177.4 Configuring the Compilation Environment 1858
4.177.5 Obtaining the Source Code 1859
4.177.6 Compilation and Installation 1859
4.177.7 Running and Verification 1860
4.178 HMMER 3.3.2 Porting Guide (Kylin V10) 1860

 Copyright © Huawei Technologies Co., Ltd.

4.178.1 Introduction 1860
4.178.2 Environment Requirements 1861
4.178.3 Paths for Software Porting 1862
4.178.4 Configuring the Compilation Environment 1862
4.178.4.1 Configuration Description 1862
4.178.4.2 Installing Dependencies 1862
4.178.5 Obtaining the Source Code 1863
4.178.6 Compilation and Installation 1863
4.178.7 Running and Verification 1864
4.178.8 Change History 1865
4.179 IDBA-UD 1.0 Porting Guide (Kylin V10) 1865
4.179.1 Introduction 1865
4.179.2 Environment Requirements 1866
4.179.3 Paths for Software Porting 1867
4.179.4 Configuring the Compilation Environment 1867
4.179.4.1 Configuration Description 1867
4.179.4.2 Installing the BiSheng Compiler 1868
4.179.4.3 Installing CMake 1868
4.179.5 Obtaining the Source Code 1869
4.179.6 Compilation and Installation 1869
4.179.7 Running and Verification 1869
4.179.8 Change History 1870
4.180 InterProScan 5.53-87.0 Porting Guide (Kylin V10) 1870
4.180.1 Introduction 1870
4.180.2 Environment Requirements 1871
4.180.3 Paths for Software Porting 1871
4.180.4 Configuring the Compilation Environment 1872
4.180.4.1 Configuration Description 1872
4.180.4.2 Installing Dependencies 1872
4.180.5 Obtaining the Source Code 1873
4.180.6 Compilation and Installation 1873
4.180.7 Change History 1875
4.181 MEME 5.4.0 Porting Guide (Kylin V10) 1875
4.181.1 Introduction 1875
4.181.2 Environment Requirements 1875
4.181.3 Paths for Software Porting 1876
4.181.4 Configuring the Compilation Environment 1877
4.181.5 Obtaining the Source Code 1877
4.181.6 Compilation and Installation 1877
4.181.7 Running and Verification 1878
4.181.8 Change History 1878
4.182 MetaVelvet 1.2.01 Porting Guide (Kylin V10) 1879

 Copyright © Huawei Technologies Co., Ltd.

4.182.1 Introduction 1879
4.182.2 Environment Requirements 1879
4.182.3 Paths for Software Porting 1880
4.182.4 Configuring the Compilation Environment 1880
4.182.4.1 Configuration Description 1881
4.182.4.2 Installing the BiSheng Compiler 1881
4.182.5 Obtaining the Source Code 1881
4.182.6 Compilation and Installation 1882
4.182.7 Running and Verification 1882
4.182.8 Change History 1883
4.183 mothur 1.48 Porting Guide (Kylin V10) 1883
4.183.1 Introduction 1883
4.183.2 Environment Requirements 1884
4.183.3 Paths for Software Porting 1884
4.183.4 Configuring the Compilation Environment 1885
4.183.4.1 Configuration Description 1885
4.183.5 Obtaining the Source Code 1885
4.183.6 Compilation and Installation 1886
4.183.7 Running and Verification 1886
4.183.8 Change History 1886
4.184 PHYLIP 3.697 Porting Guide (Kylin V10) 1887
4.184.1 Introduction 1887
4.184.2 Environment Requirements 1887
4.184.3 Paths for Software Porting 1888
4.184.4 Configuring the Compilation Environment 1888
4.184.5 Obtaining the Source Code 1889
4.184.6 Compilation and Installation 1889
4.184.7 Running and Verification 1889
4.184.8 Change History 1890
4.185 Primer3 2.4.0 Porting Guide (Kylin V10) 1890
4.185.1 Introduction 1890
4.185.2 Environment Requirements 1891
4.185.3 Paths for Software Porting 1891
4.185.4 Configuring the Compilation Environment 1892
4.185.5 Obtaining the Source Code 1892
4.185.6 Compilation and Installation 1892
4.185.7 Running and Verification 1893
4.185.8 Change History 1894
4.186 Sambamba 0.8.2 Porting Guide (openEuler 20.03) 1894
4.186.1 Introduction 1894
4.186.2 Environment Requirements 1894
4.186.3 Paths for Software Porting 1895

 Copyright © Huawei Technologies Co., Ltd.

4.186.4 Configuring the Compilation Environment....1896
4.186.4.1 Configuration Description....1896
4.186.4.2 Installing LDC....1896
4.186.5 Obtaining the Source Code....1897
4.186.6 Compilation and Installation....1897
4.186.7 Running and Verification....1897
4.186.8 Change History....1898
4.187 SOAPnuke 2.1.7 Porting Guide (Kylin V10)....1898
4.187.1 Introduction....1898
4.187.2 Environment Requirements....1898
4.187.3 Paths for Software Porting....1899
4.187.4 Configuring the Compilation Environment....1900
4.187.4.1 Configuration Description....1900
4.187.4.2 Installing the BiSheng Compiler....1900
4.187.5 Obtaining the Source Code....1901
4.187.6 Compilation and Installation....1901
4.187.7 Running and Verification....1901
4.187.8 Change History....1902
4.188 SOAPsnp Master Porting Guide (Kylin V10)....1902
4.188.1 Introduction....1902
4.188.2 Environment Requirements....1903
4.188.3 Paths for Software Porting....1903
4.188.4 Configuring the Compilation Environment....1904
4.188.4.1 Configuration Description....1904
4.188.4.2 Installing the BiSheng Compiler....1904
4.188.5 Obtaining the Source Code....1905
4.188.6 Compilation and Installation....1905
4.188.7 Running and Verification....1905
4.188.8 Change History....1906
4.189 trimmomatic 0.33.0 Porting Guide (Kylin V10)....1906
4.189.1 Introduction....1906
4.189.2 Environment Requirements....1906
4.189.3 Paths for Software Porting....1907
4.189.4 Configuring the Compilation Environment....1908
4.189.4.1 Configuration Description....1908
4.189.4.2 Installing Java....1908
4.189.5 Obtaining the Source Code....1909
4.189.6 Compilation and Installation....1909
4.189.7 Running and Verification....1909
4.189.8 Change History....1909
4.190 tRNAscan-SE 2.0.9 Porting Guide (Kylin V10)....1909
4.190.1 Introduction....1910

 Copyright © Huawei Technologies Co., Ltd.

4.190.2 Environment Requirements 1910
4.190.3 Paths for Software Porting 1911
4.190.4 Configuring the Compilation Environment 1911
4.190.4.1 Configuration Description 1911
4.190.4.2 Installing the BiSheng Compiler 1912
4.190.5 Obtaining the Source Code 1912
4.190.6 Compilation and Installation 1913
4.190.7 Running and Verification 1913
4.190.8 Change History 1914
4.191 vcfSampleCompare 2.013 Porting Guide (Kylin V10) 1914
4.191.1 Introduction 1914
4.191.2 Environment Requirements 1915
4.191.3 Paths for Software Porting 1916
4.191.4 Configuring the Compilation Environment 1916
4.191.4.1 Configuration Description 1916
4.191.4.2 Installing Dependencies 1916
4.191.5 Obtaining the Source Code 1917
4.191.6 Compilation and Installation 1917
4.191.7 Running and Verification 1917
4.191.8 Change History 1917
4.192 VelvetOptimiser 2.2.6 Porting Guide (Kylin V10) 1918
4.192.1 Introduction 1918
4.192.2 Environment Requirements 1918
4.192.3 Paths for Software Porting 1919
4.192.4 Configuring the Compilation Environment 1919
4.192.4.1 Configuration Description 1919
4.192.4.2 Installing Perl 1920
4.192.4.3 Installing Velvet 1921
4.192.5 Obtaining the Source Code 1921
4.192.6 Compilation and Installation 1922
4.192.7 Running and Verification 1922
4.192.8 Change History 1922
4.193 wtdbg 1.1.006 Porting Guide (Kylin V10) 1923
4.193.1 Introduction 1923
4.193.2 Environment Requirements 1923
4.193.3 Paths for Software Porting 1924
4.193.4 Configuring the Compilation Environment 1924
4.193.4.1 Configuration Description 1924
4.193.4.2 Installing GCC 1925
4.193.5 Obtaining the Source Code 1925
4.193.6 Compilation and Installation 1925
4.193.7 Running and Verification 1927

 Copyright © Huawei Technologies Co., Ltd.

4.193.8 Change History....1928
4.194 wtdbg2 2.5 Porting Guide (Kylin V10)....1928
4.194.1 Introduction....1928
4.194.2 Environment Requirements....1929
4.194.3 Paths for Software Porting....1929
4.194.4 Configuring the Compilation Environment....1930
4.194.4.1 Configuration Description....1930
4.194.4.2 Installing GCC....1930
4.194.5 Obtaining the Source Code....1930
4.194.6 Compilation and Installation....1931
4.194.7 Running and Verification....1934
4.194.8 Change History....1934
4.195 xpressplot 0.2.5 Porting Guide (Kylin V10)....1934
4.195.1 Introduction....1934
4.195.2 Environment Requirements....1935
4.195.3 Paths for Software Porting....1936
4.195.4 Configuring the Compilation Environment....1936
4.195.5 Obtaining the Source Code....1936
4.195.6 Compilation and Installation....1937
4.195.7 Running and Verification....1937
4.195.8 Change History....1937
5 Other....1938
5.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)....1939
5.1.1 Introduction....1939
5.1.2 Environment Requirements....1939
5.1.3 Paths for Software Porting....1940
5.1.4 Configuring the Compilation Environment....1941
5.1.4.1 Configuration Description....1941
5.1.4.2 Installing the Java Environment....1941
5.1.5 Obtaining the Source Code....1942
5.1.6 Compiling and Installing Hadoop and Spark....1942
5.1.6.1 Configuring Hadoop....1942
5.1.6.2 Configuring Spark....1944
5.1.7 Running and Verifying Hadoop and Spark....1947
5.1.8 Troubleshooting....1948
5.1.9 More Information....1949
5.2 SGE 8.1.9 Porting Guide (CentOS 7.6)....1949
5.2.1 Introduction....1949
5.2.2 Environment Requirements....1950
5.2.3 Planning the Paths for Software Porting....1951
5.2.4 Configuring the Compilation Environment....1951
5.2.5 Obtaining the Source Code....1952

 Copyright © Huawei Technologies Co., Ltd.

5.2.6 Compiling and Installing SGE....1952
5.2.6.1 Compiling SGE....1952
5.2.6.2 Installing SGE on the Master Host....1953
5.2.6.3 Installing SGE on the Execution Host....1962
5.2.7 Running and Verifying SGE....1964
5.2.8 SGE Common Commands....1966
5.3 Conda 4.3.16 Porting Guide (CentOS 7.6)....1967
5.3.1 Introduction....1967
5.3.2 Environment Requirements....1968
5.3.3 Paths for Software Porting....1968
5.3.4 Configuring the Compilation Environment....1969
5.3.4.1 Configuration Description....1969
5.3.4.2 Installing Dependencies....1969
5.3.5 Obtaining the Source Code....1970
5.3.6 Compiling and Installing Conda....1970
5.4 NPB 3.4 Porting Guide (CentOS 7.6)....1970
5.4.1 Introduction....1970
5.4.2 Environment Requirements....1971
5.4.3 Paths for Software Porting....1972
5.4.4 Configuring the Compilation Environment....1972
5.4.5 Obtaining the Source Code....1973
5.4.6 Compiling and Installing NPB....1973
5.4.7 Running and Verifying NPB....1974
5.4.8 More Resources....1974
5.5 Double-conversion 3.2.0 Porting Guide (Kylin V10)....1976
5.5.1 Introduction....1976
5.5.2 Environment Requirements....1977
5.5.3 Paths for Software Porting....1978
5.5.4 Configuring the Compilation Environment....1978
5.5.5 Obtaining the Source Code....1979
5.5.6 Compilation and Installation....1979
5.5.7 Running and Verification....1979
5.6 LIBPNG 1.6.37 Porting Guide (Kylin V10)....1980
5.6.1 Introduction....1980
5.6.2 Environmental Requirements....1980
5.6.3 Paths for Software Porting....1981
5.6.4 Configuring the Compilation Environment....1981
5.6.5 Obtaining the Source Code....1982
5.6.6 Compilation and Installation....1982
5.6.7 Running and Verification....1983
5.7 Linenoise-NG 1.0.1 Porting Guide (Kylin V10)....1983
5.7.1 Introduction....1983

 Copyright © Huawei Technologies Co., Ltd.

5.7.2 Environment Requirements....1984
5.7.3 Paths for Software Porting....1984
5.7.4 Configuring the Compilation Environment....1985
5.7.5 Obtaining the Source Code....1985
5.7.6 Compilation and Installation....1986
5.7.7 Running and Verification....1986
5.8 NCO 5.1.0 Porting Guide (Kylin V10)....1987
5.8.1 Introduction....1987
5.8.2 Environmental Requirements....1987
5.8.3 Paths for Software Porting....1988
5.8.4 Configuring the Compilation Environment....1989
5.8.4.1 Configuration Description....1989
5.8.4.2 Installing the Dependencies....1990
5.8.4.3 Installing PnetCDF....1990
5.8.4.4 Installing HDF5....1991
5.8.4.5 Installing NetCDF....1991
5.8.4.6 Installing GSL....1992
5.8.4.7 Installing UDUNITS....1993
5.8.5 Obtaining the Source Code....1993
5.8.6 Compilation and Installation....1993
5.8.7 Running and Verification....1994
5.9 PETSc 3.16 Porting Guide (Kylin V10)....1994
5.9.1 Introduction....1994
5.9.2 Environmental Requirements....1994
5.9.3 Paths for Software Porting....1995
5.9.4 Configuring the Compilation Environment....1996
5.9.4.1 Configuration Description....1996
5.9.4.2 Installing the Dependencies....1997
5.9.5 Obtaining the Source Code....1997
5.9.6 Compilation and Installation....1997
5.9.7 Running and Verification....1997
5.10 Chainer 7.8.1 Porting Guide (Kylin V10)....1998
5.10.1 Introduction....1998
5.10.2 Environment Requirements....1998
5.10.3 Paths for Software Porting....1999
5.10.4 Configuring the Compilation Environment....1999
5.10.4.1 Configuration Description....1999
5.10.4.2 Installing Dependencies....2000
5.10.5 Obtaining the Source Code....2000
5.10.6 Compilation and Installation....2000
5.10.7 Running and Verification....2001
5.11 Horovod 0.25.0 Porting Guide (Kylin V10)....2001

 Copyright © Huawei Technologies Co., Ltd.

5.11.1 Introduction....2001
5.11.2 Environment Requirements....2002
5.11.3 Paths for Software Porting....2003
5.11.4 Configuring the Compilation Environment....2003
5.11.5 Compilation and Installation....2004
5.11.6 Running and Verification....2004
5.12 Keras 2.9.0 Porting Guide (Kylin V10)....2005
5.12.1 Introduction....2005
5.12.2 Environment Requirements....2005
5.12.3 Paths for Software Porting....2006
5.12.4 Configuring the Compilation Environment....2006
5.12.4.1 Configuration Description....2006
5.12.4.2 Installing Dependencies....2007
5.12.5 Obtaining the Source Code....2008
5.12.6 Compilation and Installation....2008
5.12.7 Running and Verification....2008
5.13 Lasagne 0.1 Porting Guide (Kylin V10)....2009
5.13.1 Introduction....2009
5.13.2 Environment Requirements....2009
5.13.3 Paths for Software Porting....2010
5.13.4 Configuring the Compilation Environment....2011
5.13.4.1 Configuration Description....2011
5.13.4.2 Installing Python....2011
5.13.5 Obtaining the Source Code....2012
5.13.6 Compilation and Installation....2012
5.13.7 Running and Verification....2012
5.14 mpi4py 3.1.3 Porting Guide (Kylin V10)....2013
5.14.1 Introduction....2013
5.14.2 Environment Requirements....2013
5.14.3 Paths for Software Porting....2014
5.14.4 Configuring the Compilation Environment....2015
5.14.4.1 Configuration Description....2015
5.14.4.2 Installing Dependencies....2015
5.14.5 Obtaining the Source Code....2015
5.14.6 Compilation and Installation....2016
5.14.7 Running and Verification....2016
5.15 MXNet 2.0.0 Porting Guide (Kylin V10)....2017
5.15.1 Introduction....2017
5.15.2 Environment Requirements....2017
5.15.3 Paths for Software Porting....2018
5.15.4 Configuring the Compilation Environment....2018
5.15.4.1 Configuration Description....2018

 Copyright © Huawei Technologies Co., Ltd.

5.15.4.2 Installing Dependencies....2019
5.15.4.3 Installing OpenCV....2019
5.15.5 Obtaining the Source Code....2020
5.15.6 Compilation and Installation....2020
5.15.7 Running and Verification....2021
5.16 sparsehash 2.0.4 Porting Guide (Kylin V10)....2021
5.16.1 Introduction....2021
5.16.2 Environment Requirements....2021
5.16.3 Paths for Software Porting....2022
5.16.4 Configuring the Compilation Environment....2023
5.16.5 Obtaining the Source Code....2023
5.16.6 Compilation and Installation....2024
5.16.7 Running and Verification....2024
5.17 SUNDIALS 6.3.0 Porting Guide (Kylin V10)....2024
5.17.1 Introduction....2025
5.17.2 Environment Requirements....2025
5.17.3 Paths for Software Porting....2026
5.17.4 Configuring the Compilation Environment....2026
5.17.5 Obtaining the Source Code....2027
5.17.6 Compilation and Installation....2027
5.17.7 Running and Verification....2027
5.18 Theano 1.0.5 Porting Guide (Kylin V10)....2028
5.18.1 Introduction....2029
5.18.2 Environment Requirements....2029
5.18.3 Paths for Software Porting....2030
5.18.4 Configuring the Compilation Environment....2030
5.18.4.1 Configuration Description....2030
5.18.4.2 Installing Dependencies....2031
5.18.5 Obtaining the Source Code....2031
5.18.6 Compilation and Installation....2031
5.18.7 Running and Verification....2032
5.19 Thrust 1.12 Porting Guide (Kylin V10)....2033
5.19.1 Introduction....2033
5.19.2 Environment Requirements....2033
5.19.3 Paths for Software Porting....2034
5.19.4 Configuring the Compilation Environment....2034
5.19.4.1 Configuration Description....2034
5.19.4.2 Installing CUDA....2035
5.19.4.3 Installing CMake....2036
5.19.5 Obtaining the Source Code....2036
5.19.6 Compilation and Installation....2036
5.19.7 Running and Verification....2037

 Copyright © Huawei Technologies Co., Ltd.

5.20 Tinker 8.10.3 Porting Guide (Kylin V10) 2038
5.20.1 Introduction 2038
5.20.2 Environment Requirements 2038
5.20.3 Paths for Software Porting 2039
5.20.4 Configuring the Compilation Environment 2039
5.20.5 Obtaining the Source Code 2040
5.20.6 Compilation and Installation 2040
5.20.7 Running and Verification 2041
5.21 west Porting Guide (Kylin V10) 2042
5.21.1 Introduction 2042
5.21.2 Environment Requirements 2042
5.21.3 Paths for Software Porting 2043
5.21.4 Configuring the Compilation Environment 2043
5.21.5 Obtaining the Source Code 2043
5.21.6 Compilation and Installation 2044
5.21.7 Running and Verification 2044
5.22 Crystal 1.7.2 Porting Guide (Kylin V10) 2044
5.22.1 Introduction 2044
5.22.2 Environment Requirements 2045
5.22.3 Paths for Software Porting 2046
5.22.4 Configuring the Compilation Environment 2046
5.22.4.1 Configuration Description 2046
5.22.4.2 Installing Docker 2047
5.22.5 Obtaining the Source Code 2048
5.22.6 Compilation and Installation 2048
5.22.7 Running and Verification 2048
5.22.8 Change History 2050
5.23 GRIB API 1.21.0 Porting Guide (Kylin V10) 2050
5.23.1 Introduction 2050
5.23.2 Environment Requirements 2050
5.23.3 Paths for Software Porting 2051
5.23.4 Configuring the Compilation Environment 2052
5.23.4.1 Configuration Description 2052
5.23.4.2 Installing Dependencies 2052
5.23.4.3 Installing HDF5 2053
5.23.4.4 Installing PnetCDF 2053
5.23.4.5 Installing NetCDF 2054
5.23.5 Obtaining the Source Code 2054
5.23.6 Compilation and Installation 2054
5.23.7 Running and Verification 2055
5.24 IPM 2.0.6 Porting Guide (Kylin V10) 2055
5.24.1 Introduction 2055

 Copyright © Huawei Technologies Co., Ltd.

5.24.2 Environment Requirements..... 2056
5.24.3 Paths for Software Porting..... 2057
5.24.4 Configuring the Compilation Environment..... 2057
5.24.4.1 Configuration Description..... 2057
5.24.4.2 Installing Perl..... 2058
5.24.5 Obtaining the Source Code..... 2058
5.24.6 Compilation and Installation..... 2058
5.24.7 Running and Verification..... 2059
5.24.8 Change History..... 2059
5.25 ncview 2.1.5 Porting Guide (Kylin V10)..... 2059
5.25.1 Introduction..... 2059
5.25.2 Environment Requirements..... 2060
5.25.3 Paths for Software Porting..... 2061
5.25.4 Configuring the Compilation Environment..... 2061
5.25.4.1 Configuration Description..... 2061
5.25.4.2 Installing Dependencies..... 2062
5.25.4.3 Installing HDF5..... 2062
5.25.4.4 Installing PnetCDF..... 2062
5.25.4.5 Installing NetCDF..... 2063
5.25.4.6 Installing libexpat..... 2063
5.25.4.7 Installing UDUNITS..... 2064
5.25.5 Obtaining the Source Code..... 2064
5.25.6 Compilation and Installation..... 2064
5.25.7 Running and Verification..... 2065
5.25.8 Change History..... 2065
5.26 paddle 2.3.1 Porting Guide (Kylin V10)..... 2066
5.26.1 Introduction..... 2066
5.26.2 Environment Requirements..... 2066
5.26.3 Paths for Software Porting..... 2067
5.26.4 Configuring the Compilation Environment..... 2067
5.26.4.1 Configuration Description..... 2068
5.26.4.2 Installing Dependencies..... 2068
5.26.5 Obtaining the Source Code..... 2068
5.26.6 Compilation and Installation..... 2068
5.26.7 Running and Verification..... 2069
5.26.8 Change History..... 2070
5.27 ploticus 242 Porting Guide (Kylin V10)..... 2070
5.27.1 Introduction..... 2070
5.27.2 Environment Requirements..... 2070
5.27.3 Paths for Software Porting..... 2071
5.27.4 Configuring the Compilation Environment..... 2072
5.27.5 Obtaining the Source Code..... 2072

 Copyright © Huawei Technologies Co., Ltd.

5.27.6 Compilation and Installation....2072
5.27.7 Running and Verification....2073
5.27.8 Change History....2073
5.28 Rosetta Porting Guide (Kylin V10)....2073
5.28.1 Introduction....2073
5.28.2 Environment Requirements....2074
5.28.3 Paths for Software Porting....2075
5.28.4 Configuring the Compilation Environment....2076
5.28.4.1 Configuration Description....2076
5.28.4.2 Installing Python 3.7.6....2076
5.28.4.3 Installing JDK 8....2077
5.28.4.4 Installing Bazel 0.25.0....2077
5.28.4.5 Installing TensorFlow 1.14.0....2078
5.28.5 Obtaining the Source Code....2078
5.28.6 Compilation and Installation....2078
5.28.7 Change History....2079
5.29 SimpleITK Master Porting Guide (openEuler 20.03)....2079
5.29.1 Introduction....2079
5.29.2 Environment Requirements....2079
5.29.3 Paths for Software Porting....2080
5.29.4 Configuring the Compilation Environment....2081
5.29.4.1 Configuration Description....2081
5.29.4.2 Installing Dependencies....2081
5.29.4.3 Installing the R Language....2081
5.29.4.4 Installing Miniconda....2082
5.29.5 Obtaining the Source Code....2082
5.29.6 Compilation and Installation....2082
5.29.7 Running and Verification....2083
5.29.8 Change History....2083
5.30 Singularity 3.9.9 Porting Guide (Kylin V10)....2083
5.30.1 Introduction....2083
5.30.2 Environment Requirements....2084
5.30.3 Paths for Software Porting....2085
5.30.4 Configuring the Compilation Environment....2085
5.30.4.1 Configuration Description....2085
5.30.4.2 Installing Go....2086
5.30.5 Obtaining the Source Code....2086
5.30.6 Compilation and Installation....2086
5.30.7 Running and Verification....2086
5.30.8 Change History....2087
5.31 szip 2.1.1 Porting Guide (Kylin V10)....2087
5.31.1 Introduction....2087

5.31.2 Environment Requirements..... 2087
5.31.3 Paths for Software Porting..... 2088
5.31.4 Configuring the Compilation Environment..... 2089
5.31.4.1 Configuration Description..... 2089
5.31.5 Obtaining the Source Code..... 2089
5.31.6 Compilation and Installation..... 2090
5.31.7 Running and Verification..... 2090
5.31.8 Change History..... 2090
5.32 VTK 7.1.1 Porting Guide (openEuler 20.03)..... 2090
5.32.1 Introduction..... 2090
5.32.2 Environment Requirements..... 2091
5.32.3 Paths for Software Porting..... 2092
5.32.4 Configuring the Compilation Environment..... 2092
5.32.4.1 Configuration Description..... 2092
5.32.4.2 Configuring the Huawei Yum Source..... 2092
5.32.4.3 Disabling the Firewall..... 2093
5.32.4.4 Installing GCC for openEuler..... 2093
5.32.5 Obtaining the Source Code..... 2094
5.32.6 Compilation and Installation..... 2094
5.32.7 Running and Verification..... 2095
5.32.8 Troubleshooting..... 2095
5.32.8.1 Test Case Compilation Error: vtkVersion.h: No such file or directory..... 2095
5.32.9 Change History..... 2097
6 Change History..... 2098

1 Meteorology

1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)
1.2 CAMx 6.50 Porting Guide (CentOS 7.6)
1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)
1.4 NEMO V3.6 Porting Guide (CentOS 7.6)
1.5 ROMS 3.6 Porting Guide (CentOS 7.6)
1.6 WAVEWATCH III 6.07.1 Porting Guide (CentOS 7.6)
1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)
1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)
1.9 MITgcm c67o Porting Guide (CentOS 7.6)
1.10 POM POM2K Porting Guide (CentOS 7.6)
1.11 MOM 6 Porting Guide (CentOS 7.6)
1.12 SMOKE 4.7 Porting Guide (CentOS 7.6)
1.13 SPECFEM3D GLOBE 7.0.0 Porting Guide (CentOS 7.6)
1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6)
1.15 SWAN 41.31 Porting Guide (CentOS 7.6)
1.16 CDO 1.9.8 Porting Guide (CentOS 7.6)
1.17 ecCodes 2.18.0 Porting Guide (CentOS 7.6)
1.18 ecFlow 5.5.2 Porting Guide (CentOS 7.6)
1.19 NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6)
1.20 PyFerret 7.6.0 Porting Guide (CentOS 7.6)
1.21 NCL 6.3.0 Porting Guide (CentOS 7.6)
1.22 ARPS 5.3.4 Porting Guide (CentOS 7.6)
1.23 CAMx 7.10 Porting Guide (openEuler 21.03)

 Copyright © Huawei Technologies Co., Ltd.

1.24 CDO 1.9.8 Porting Guide (openEuler 20.03)
1.25 ecCodes 2.18.0 Porting Guide (openEuler 20.03)
1.26 ecFlow 5.5.2 Porting Guide (openEuler 20.03)
1.27 MITgcm c67o Porting Guide (openEuler 21.03)
1.28 NCEPLIBS 1.2.0 Porting Guide (openEuler 20.03)
1.29 SWAN 41.31 Porting Guide (openEuler 20.03)
1.30 WAVEWATCH III 6.07.1 Porting Guide (openEuler 21.03)
1.31 WRF 4.2 Porting Guide (openEuler 21.03)
1.32 CESM 2.1.1 Porting Guide (openEuler 21.03)
1.33 CESM 2.1.3 Porting Guide (openEuler 21.03)
1.34 NEMO 3.6 Porting Guide (openEuler 21.03)
1.35 ROMS 4.0 Porting Guide (openEuler 21.03)
1.36 ncl 6.3.0 Porting Guide (openEuler 20.03)
1.37 PyFerret 7.6.0 Porting Guide (openEuler 21.03)
1.38 SMOKE 4.7 Porting Guide (openEuler 21.03)
1.39 SPECFEM3D GLOBE 7.0.0 Porting Guide (openEuler 21.03)
1.40 CMAQ 5.3.1 Porting Guide (openEuler 21.03)
1.41 POM POM2K Porting Guide (openEuler 21.03)
1.42 FVCOM 4.1 Porting Guide (openEuler 21.03)
1.43 gadget2 2.3.5 Porting Guide (Kylin V10)
1.44 GrADS 2.2.1 Porting Guide (CentOS 8.2)
1.45 wgrib 1.8.2 Porting Guide (Kylin V10)
1.46 wgrib2 3.1.1 Porting Guide (Kylin V10)
1.47 WPS 3.9 Porting Guide (CentOS 8.2)

1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)

1.1.1 Introduction

The Weather Research and Forecasting (WRF) Model can be used for fine-scale weather simulation and forecasting, which is one of the important application scenarios of high-performance computing (HPC).

For more information about WRF, visit the **WRF official website**.

Programming languages: C/Fortran

Brief description: mesoscale weather forecasting model.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: public domain

NOTE

For WRF v3.8.1 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port WRF to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

WRF v3.8.1

1.1.2 Environment Requirements

Hardware Requirements

Table 1-1 lists the hardware requirements.

Table 1-1 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-2 lists the software requirements.

Table 1-2 Software requirements

Item	Version	Download Address
HDF5	1.10.1	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/**
PNETCDF	1.9.0	**https://parallel-netcdf.github.io/wiki/Download.html**
NETCDF	4.4.1.1	**https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1**
NETCDF-F	4.4.1	**https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1**
OPTIMIZED-ROUTINES	V201910	**https://github.com/ARM-software/optimized-routines/releases/tag/v201910**

OS Requirements

Table 1-3 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-3 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.1.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WRF software porting.

Table 1-4 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/PNETCDF	Installation path of PnetCDF.	
4	/path/to/NETCDF	Installation path of NetCDF-C.	
5	/path/to/NETCDF	Installation path of NetCDF-Fortran.	
6	/path/to/MATH	Installation path of OPTIMIZED-ROUTINES.	
7	/path/to/WRF	Installation path of WRF.	
8	/path/to/WRFTEST	Test path of WRF.	

1.1.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

1.1.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-5 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.1.4.3 Installing HDF5**.
3	Installing PnetCDF	See **1.1.4.4 Installing PnetCDF**.
4	Installing NetCDF-C	See **1.1.4.5 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.1.4.6 Installing NetCDF-Fortran**.
6	Installing OPTIMIZED-ROUTINES	See **1.1.4.7 Installing OPTIMIZED-ROUTINES**.
7	Installing ClusterShell	See **1.1.4.8 Installing ClusterShell**.

1.1.4.2 Setting Up the Basic Environment

Install GNU and Open MPI. For details, see "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.1.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install HDF5.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
tar -xvf hdf5-1.10.1.tar.gz
cd hdf5-1.10.1
```

 Copyright © Huawei Technologies Co., Ltd.

```
mkdir -p /path/to/HDF5
./configure --prefix=/path/to/HDF5  --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
make -j 16
make install
```

Step 3 Run the following commands to check whether HDF5 is successfully installed:

```
cd /path/to/HDF5
ls lib
```

If the following information is displayed, the installation is successful.

----End

1.1.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install PnetCDF.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
tar -xvf parallel-netcdf-1.9.0.tar.bz2
cd parallel-netcdf-1.9.0
mkdir -p /path/to/PNETCDF
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
make -j 16
make install
```

NOTE

If the environment variables of the compiler and MPI have been declared, skip the first two commands for declaring environment variables in this step.

Step 3 Run the following commands to check whether PnetCDF is successfully installed.

```
cd /path/to/PNETCDF
ls lib
```

If the following information is displayed, the installation is successful.

```
[root@2280node24 PNETCDF]# cd /path/to/PNETCDF/
[root@2280node24 PNETCDF]# ls lib/
libpnetcdf.a  libpnetcdf.la  pkgconfig
```

----End

1.1.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install NetCDF-C.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
cd netcdf-c-4.4.1.1
mkdir -p /path/to/NETCDF
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/PNETCDF/lib" CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
make -j 16
make install
```

NOTE

If the environment variables of the compiler and MPI have been declared, skip the first two commands for declaring environment variables in this step.

----End

1.1.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install NetCDF-Fortran.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
tar -xvf netcdf-fortran-4.4.1.tar.gz
cd netcdf-fortran-4.4.1
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib" CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
make -j 16
make install
```

NOTE

If the environment variables of the compiler and MPI have been declared, skip the first two commands for declaring environment variables in this step.

Step 3 Check whether NetCDF-Fortran is successfully installed.

```
cd /path/to/NETCDF
ls lib
```

If the following information is displayed, the installation is successful.

----End

1.1.4.7 Installing OPTIMIZED-ROUTINES

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install OPTIMIZED-ROUTINES.

```
yum install glibc* glibm* -y
tar -xvf optimized-routines-201910.tar.gz
cd optimized-routines-201910
mkdir -p /path/to/MATH
cp config.mk.dist config.mk
make -j 16
cp -r ./build/* /path/to/MATH
```

Step 3 Check whether the installation is successful.

```
cd /path/to/MATH
ls lib
```

If the following information is displayed, the installation is successful.

```
[root@2280node24 MATH]# cd /path/to/MATH/
[root@2280node24 MATH]# ls lib/
libmathlib.a  libmathlib.so  libstringlib.a  libstringlib.so
```

----End

1.1.4.8 Installing ClusterShell

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to download the ClusterShell RPM installation package **clustershell-1.8.2-1.el7.noarch.rpm** and its dependency **python2-clustershell-1.8.2-1.el7.noarch.rpm**:

```
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/c/
clustershell-1.8.2-1.el7.noarch.rpm
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/p/python2-
clustershell-1.8.2-1.el7.noarch.rpm
```

Step 3 Run the following commands to install ClusterShell:

```
yum install python-setuptools –y
rpm -ivh python2-clustershell-1.8.2-1.el7.noarch.rpm clustershell-1.8.2-1.el7.noarch.rpm
```

NOTE

ClusterShell is installed to clear the cache of each node when multiple nodes are running. If only one node is running, you do not need to install ClusterShell.

----End

1.1.5 Obtaining Source Code

Step 1 Download the WRF source code package **WRF-3.8.1.tar.gz**.

https://github.com/wrf-model/WRF/releases/tag/V3.8.1

Step 2 Use an SFTP tool to copy the WRF source code package to the */path/to/WRF* directory on the server.

----End

1.1.6 Compilation and Installation

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the WRF installation package.

```
cd /path/to/WRF
tar -xvf WRF-3.8.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to WRF source code directory:

```
cd WRF-3.8.1
```

Step 4 Run the following command to modify the **arch/configure_new.defaults** file.

1. Open the **arch/configure_new.defaults** file.

   ```
   vim arch/configure_new.defaults
   ```

2. Press **i** to enter the insert mode and add the following content above 1856:

   ```
   ################################################### ##########
   #ARCH   Linux   aarch64,gnu OpenMPI #serial smpar dmpar dm+sm
   DESCRIPTION     =       GNU ($SFC/$SCC)
   DMPARALLEL      =       1
   OMPCPP          =       -D_OPENMP
   OMP             =       -fopenmp
   OMPCC           =       -fopenmp
   SFC             =       gfortran
   SCC             =       gcc
   CCOMP           =       gcc
   DM_FC           =       mpif90 -f90=$(SFC)
   DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
   FC              =       CONFIGURE_FC
   CC              =       CONFIGURE_CC
   LD              =       $(FC)
   RWORDSIZE       =       CONFIGURE_RWORDSIZE
   PROMOTION       =       #-fdefault-real-8
   ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR  -DWRF_USE_CLM
   CFLAGS_LOCAL    =       -w -O3 -c -march=armv8.2-a -L/path/to/MATH/lib -lmathlib
   LDFLAGS_LOCAL   =
   CPLUSPLUSLIB    =
   ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
   FCOPTIM         =       -O3 -ftree-vectorize -funroll-loops -march=armv8.2-a -L/path/to/MATH/lib -
   lmathlib
   FCREDUCEDOPT    =       $(FCOPTIM)
   FCNOOPT         =       -O0
   FCDEBUG         =       # -g $(FCNOOPT)  # -fbacktrace -ggdb-fcheck=bounds,do,mem,pointer -ffpe-
   trap=invalid,zero,overflow
   FORMAT_FIXED    =       -ffixed-form
   FORMAT_FREE     =       -ffree-form -ffree-line-length-none
   FCSUFFIX        =
   BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
   FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
   FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
   MODULE_SRCH_FLAG =
   TRADFLAG        =       -traditional
   CPP             =       /lib/cpp -P
   AR              =       ar
   ARFLAGS         =       ru
   M4              =       m4 -G
   RANLIB          =       ranlib
   RLFLAGS         =
   CC_TOOLS        =       $(SCC)
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following commands to configure the pre-compilation environment:

```
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
export NETCDF=/path/to/NETCDF
export HDF5=/path/to/HDF5
export PNETCDF=/path/to/PNETCDF
export CPPFLAGS="-I$HDF5/include -I$PNETCDF/include -I$NETCDF/include"
export LDFLAGS="-L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib -lnetcdf -lnetcdff -lpnetcdf -lhdf5_hl -
lhdf5 -lz"
```

Step 6 Run the following command to generate a configuration file:

```
echo 4 | ./configure
```

Step 7 Run the following command to modify the **./phys/module_cu_g3.F** file:

 Copyright © Huawei Technologies Co., Ltd.

1. Open the **./phys/module_cu_g3.F** file.
   ```
   vim ./phys/module_cu_g3.F
   ```
2. Press **i** to enter the insert mode and modify the content in line 3125.
   ```
   integer,  dimension (33) :: seed
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

NOTE

You can skip this step during the installation of WRF 4.1.2.

Step 8 Run the following commands to load the compiler and set the Open MPI environment variables.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

NOTE

If the environment variables of the compiler and Open MPI have been declared, skip this step.

Step 9 Run the following command to perform compilation and installation:

```
./compile -j 16 em_real 2>&1 | tee -a compile.log
```

Step 10 Run the following command to verify the installation:

```
ls main
```

If the following information is displayed (the **wrf.exe** file is generated), the installation is successful. The installation of the main program takes approximately 10 minutes.

----End

1.1.7 Running and Verification

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the case file from the following link:

https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/

Three target files: **wrfrst_d01_2005-06-04_06_00_00**, **namelist.input**, and **wrfbdy_d01**

Step 3 Run the following commands on PuTTY to create the */path/to/WRFTEST* directory and upload the case to the directory:

```
mkdir -p /path/to/WRFTEST
cd /path/to/WRFTEST
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/RST/ -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html*"
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/wrfbdy_d01.gz -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html*"
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/namelist.input -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html*"
```

Step 4 Run the following command to place **namelist.input** in the test directory and delete unnecessary directories:

```
mv 1-RST/namelist.input ./
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to generate a case and delete unnecessary directories:

```
cat 1-RST/RST/rst_6hr* | gunzip -c > wrfrst_d01_2005-06-04_06_00_00
rm -rf 1-RST
```

Step 6 Run the following command to decompress the **wrfbdy_d01** file:

```
gunzip wrfbdy_d01.gz
```

Step 7 Run the following command to modify the **namelist.input** file:

1. Open the **namelist.input** file.

   ```
   vim namelist.input
   ```

2. Press **i** to enter the insert mode and modify the following information:

   ```
   &time_control
   run_days                            = 0,
   run_hours                           = 3,
   run_minutes                         = 0,
   run_seconds                         = 0,
   start_year                          = 2005,
   start_month                         = 06,
   start_day                           = 04,
   start_hour                          = 06,
   start_minute                        = 00,
   start_second                        = 00,
   end_year                            = 2005,
   end_month                           = 06,
   end_day                             = 04,
   end_hour                            = 12,
   end_minute                          = 00,
   end_second                          = 00,
   interval_seconds                    = 10800
   input_from_file                     = .true.,
   history_interval                    = 180,
   frames_per_outfile                  = 1,
   nocolons                            = .true.,
   restart                             = .true.,
   restart_interval                    = 360,
   io_form_history                     = 11,
   io_form_restart                     = 11,
   io_form_input                       = 2,
   io_form_boundary                    = 2,
   io_form_auxhist2                    = 2,
   debug_level                         = 0
   /
   &domains
   time_step                           = 15,
   time_step_fract_num                 = 0,
   time_step_fract_den                 = 1,
   max_dom                             = 1,
   s_we                                = 1,
   e_we                                = 1501,
   s_sn                                = 1,
   e_sn                                = 1201,
   s_vert                              = 1,
   e_vert                              = 35,
   dx                                  = 2500,
   dy                                  = 2500,
   grid_id                             = 1,
   parent_id                           = 0,
   i_parent_start                      = 0,
   j_parent_start                      = 0,
   parent_grid_ratio                   = 1,
   parent_time_step_ratio              = 1,
   feedback                            = 1,
   smooth_option                       = 0
   lagrange_order                      = 2
   interp_type                         = 2
   extrap_type                         = 2
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
t_extrap_type                       = 2
use_surface                         = .true.
use_levels_below_ground             = .true.
num_metgrid_levels                  = 40,
eta_levels                          = 1.00000 , 0.99258 , 0.98275 , 0.96996 , 0.95372 ,
0.93357 , 0.90913 , 0.87957 , 0.84531 , 0.80683 ,
0.76467 , 0.71940 , 0.67163 , 0.62198 , 0.57108 ,
0.51956 , 0.46803 , 0.42030 , 0.37613 , 0.33532 ,
0.29764 , 0.26290 , 0.23092 , 0.20152 , 0.17452 ,
0.14978 , 0.12714 , 0.10646 , 0.08761 , 0.07045 ,
0.05466 , 0.03981 , 0.02580 , 0.01258 , 0.00000
/
&physics
mp_physics                          = 4,
mp_zero_out                         = 2,
mp_zero_out_thresh                  = 1.e-9,
ra_lw_physics                       = 1,
ra_sw_physics                       = 1,
radt                                = 10,
sf_sfclay_physics                   = 1,
sf_surface_physics                  = 2,
bl_pbl_physics                      = 1,
bldt                                = 0,
cu_physics                          = 0,
cudt                                = 5,
isfflx                              = 1,
ifsnow                              = 0,
icloud                              = 1,
surface_input_source                = 1,
num_soil_layers                     = 4,
maxiens                             = 1,
maxens                              = 1,
maxens2                             = 1,
maxens3                             = 1,
ensdim                              = 1,
/
&dynamics
w_damping                           = 1,
diff_opt                            = 1,
km_opt                              = 4,
khdif                               = 0,
kvdif                               = 0,
non_hydrostatic                     = .true.,
use_baseparam_fr_nml                = .t.,
/
&bdy_control
spec_bdy_width                      = 5,
spec_zone                           = 1,
relax_zone                          = 4,
specified                           = .true.,
nested                              = .false.,
/
&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to copy all files in the */path/to/WRF*/**WRF-3.8.1/run** directory to the */path/to/WRFTEST* directory:

```
cp /path/to/WRF/WRF-3.8.1/run/* /path/to/WRFTEST/
```

Step 9 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:/path/to/NETCDF/lib:/path/to/
PNETCDF/lib:/path/to/HDF5/lib:/path/to/MATH/lib:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Run the following command to create the **hostfile** file:

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   Node1
   Node2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Run the following command to start the WRF program.

```
clush --hostfile hostfile "sync;echo 3 > /proc/sys/vm/drop_caches"
```

- If an IB network is used, run the following command:
  ```
  time -p `which mpirun` --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=
  $LD_LIBRARY_PATH -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so -x OMP_NUM_THREADS=4 -
  map-by ppr:32:node:pe=4 --bind-to core --hostfile hostfile --mca pml ucx ./wrf.exe
  ```
- If an RoCE network is used, run the following command:
  ```
  time -p mpirun --allow-run-as-root -hostfile hostfile -x PATH -x LD_LIBRARY_PATH -x
  LD_PRELOAD=/path/to/MATH/lib/libmathlib.so -x OMP_NUM_THREADS=4 -map-by ppr:
  24:node:pe=4 -bind-to core -display-map -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
  UCX_TLS=self,sm,rc -x UCX_NET_DEVICES=mlx5_0:1 -x UCX_IB_GID_INDEX=5 ./wrf.exe
  ```

NOTE

- **--hostfile hostfile**: specifies the list of node names to be used.
- **-x OMP_NUM_THREADS=4**: specifies the number of threads to be used.
- **-map-by ppr:32:node:pe=4**: specifies how processes and threads are bound. Each node has 32 processes, and each process has four threads. (The 128-core CPU on a single node is used as an example.)
- **UCX_NET_DEVICES**: specifies the RoCE network port.
- **UCX_IB_GID_INDEX**: specifies the RoCE network type.

Step 12 Run the following command to check whether the program has stopped properly:

```
less rsl.out.0000
```

```
Timing for main: time 2005-06-04_08:58:30 on domain   1:    3.46594 elapsed seconds
Timing for main: time 2005-06-04_08:58:45 on domain   1:    3.44134 elapsed seconds
Timing for main: time 2005-06-04_08:59:00 on domain   1:    3.44925 elapsed seconds
Timing for main: time 2005-06-04_08:59:15 on domain   1:    3.41475 elapsed seconds
Timing for main: time 2005-06-04_08:59:30 on domain   1:    3.43619 elapsed seconds
Timing for main: time 2005-06-04_08:59:45 on domain   1:    3.51455 elapsed seconds
Timing for main: time 2005-06-04_09:00:00 on domain   1:    3.44250 elapsed seconds
Timing for Writing wrfout_d01_2005-06-04_09_00_00 for domain        1:   11.98498 elapsed seconds
wrf: SUCCESS COMPLETE WRF
```

----End

1.1.8 Change History

Date	Description
2019-12-30	This issue is the first official release.

1.2 CAMx 6.50 Porting Guide (CentOS 7.6)

1.2.1 Introduction

CAMx is a multi-scale photochemical modeling system for gas and particulate air pollution.

For more information about CAMx, visit the **official CAMx website**.

Programming language: Fortran

Brief description: multi-scale photochemical modeling system

Open source license: custom open source license

Recommended Software Version

CAMx 6.50

1.2.2 Environment Requirements

Hardware Requirements

Table 1-6 lists the hardware requirements.

Table 1-6 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-7 lists the software requirements.

Table 1-7 Software requirements

Item	Version	How to Obtain
CAMx	6.50	**http://www.camx.com/download/**
Compute instance file 1	CAMx6-x-test_run-inputs_met-140402.tgz	**https://www.camx.com/download/camx-test-case/**
Compute instance file 2	CAMx6-30-test_run-inputs_other-160408.tgz	**https://www.camx.com/download/camx-test-case/**
Compute instance file 3	v6-50-specific-inputs-180430.tgz	**https://www.camx.com/download/camx-test-case/**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 1-8 lists the OS requirements.

Table 1-8 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.2.3 Paths for Software Porting

This section describes the software installation paths involved in the CAMx software porting.

Table 1-9 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CAMX	Installation path of CAMx	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing the CAMx test cases	

1.2.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-10 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.2.5 Obtaining the Source Code

Procedure

Step 1 Download the **CAMx_v6-50-src-180430.tgz** installation package.

http://www.camx.com/download/

Step 2 Use an SFTP tool to upload the CAMx installation package to the */path/to/CAMX* directory on the server.

----End

1.2.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the installation file package:

```
mkdir -p /path/to/CAMX
cd /path/to/CAMX
mkdir -p CAMX_v6-50
tar xzvf CAMx_v6-50-src-180430.tgz -C CAMX_v6-50
```

Step 3 Run the following command to go to source code directory:

```
cd CAMX_v6-50
```

Step 4 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify line 64 in **Makefile**.
   ```
   MPI_INST = /path/to/OPENMPI
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to modify the **MPI/util/Makefile** file:

1. Open **MPI/util/Makefile**.
   ```
   vi MPI/util/Makefile
   ```
2. Press **i** to enter the insert mode and modify lines 3, 9, and 11 in **MPI/util/Makefile**.

 Copyright © Huawei Technologies Co., Ltd.

```
MPI_INST = /path/to/OPENMPI
# LIB = -L$(MPI_INST)/lib -lmpi
# CFLAGS = -mcmodel=small -O2 -I $(INC)
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

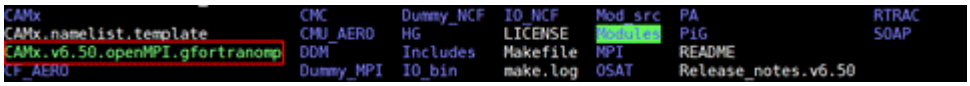
Step 6 Run the following commands to compile and install the software:

```
make COMPILER=gfortranomp MPI=openmpi -j20
```

After the compilation is successful, the **CAMx.v6.50.openMPI.gfortranomp** executable file is generated in the **src** directory. You can run the **ls** command to view the file.

After the compilation is successful, the **CAMx.v6.50.openMPI.gfortranomp** executable file is generated in the ***/path/to/CAMX/CAMX_v6-50*** directory. You can run the **ls** */path/to/CAMX/***CAMX_v6-50** command to view the file. **Figure 1-1** shows the command output.

Figure 1-1 Result example



----End

1.2.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the compute instance files:

```
cd /path/to/CASE
tar xzvf v6-50-specific-inputs-180430.tgz
tar xzvf CAMx6-x-test_run-inputs_met-140402.tgz
tar xzvf CAMx6-30-test_run-inputs_other-160408.tgz
```

Step 3 Run the following commands to copy the test script:

```
cd runfiles
cp CAMx_v6.50.midwest.36.12.20020506-07.MPICH2.job openmpi4.job
```

Step 4 Run the following command to modify the test script:

1. Open the test script.
   ```
   vi openmpi4.job
   ```
2. Press **i** to enter the insert mode and modify the test script. Pay attention to the information in bold.
   ```
   set EXEC      = "/path/to/CAMX/CAMX_v6-50/CAMx.v6.50.openMPI.gfortranomp"
   cat << ieof > nodes
   192.168.47.111
   set NUMPROCS = 96
   #mpdboot -n $RING -f nodes --verbose
   if( ! { mpirun --allow-run-as-root --mca btl_openib_allow_ib 1 -machinefile nodes -np
   $NUMPROCS $EXEC } ) then
   #mpdallexit
   exit
   endif
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the compute instances.

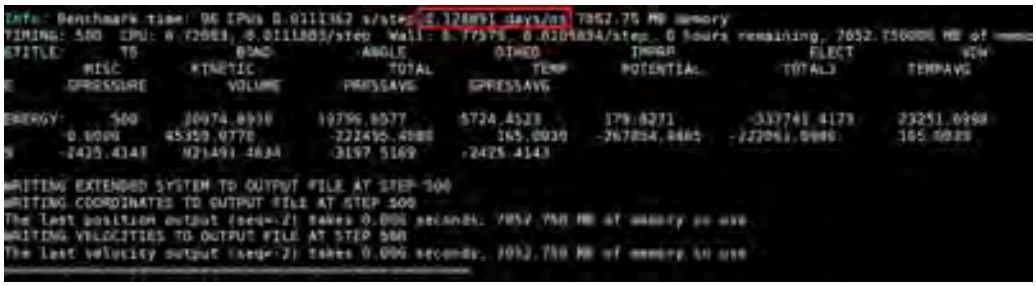
 Copyright © Huawei Technologies Co., Ltd.

```
./openmpi4.job
```

Check the value of **days/ns** in the last **Info:Benchmark time** in the log. The unit is **days/ns**. A smaller value indicates better performance. You can also convert the value to the value of **ns/days**. A higher value indicates better performance.

Figure 1-2 shows the command output.

Figure 1-2 Result example



----End

1.2.8 More Information

Official Arm website:

https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/resources/porting-and-tuning/building-namd-with-arm-compiler-deprecated

1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)

1.3.1 Introduction

The Community Earth System Model (CESM) is a fully coupled numerical simulation of the Earth system consisting of atmospheric, ocean, ice, land surface, carbon cycle, and other components. CESM includes a climate model providing state-of-art simulations of the Earth's past, present, and future. It is the successor of the Community Climate System Model (CCSM). CESM1 was released in 2010 with primary development by the Climate and Global Dynamics Division (CGD) of the National Center for Atmospheric Research (NCAR), and significant funding by the National Science Foundation (NSF) and the Department of Energy (DoE).

For more information about Bifrost, visit the **CESM official website**.

Programming language: Fortran

Brief description: a simulation of the Earth system

Open source license: custom open source license

NOTE

For CESM 2.1.1 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port CESM to the Kunpeng platform. For details, see **Dedicated Software Porting**.

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

CESM 2.1.1

1.3.2 Environment Requirements

Hardware Requirements

Table 1-11 lists the hardware requirements.

Table 1-11 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-12 lists the software requirements.

Table 1-12 Software Requirements

Item	Version	How to Obtain
CESM	2.1.1	https://github.com/ESCOMP/cesm/releases/tag/release-cesm2.1.1
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/releases
HDF5	1.10.1	http://hdfgroup.org/HDF5/
PNETCDF	1.9.0	https://parallel-netcdf.github.io/wiki/Download.html
NETCDF	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NETCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1

OS Requirements

Table 1-13 lists the OS requirements.

Table 1-13 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

1.3.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CESM software porting.

Table 1-14 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/HDF5	Installation path of HDF5	
4	/path/to/PNETCDF	Installation path of PnetCDF	
5	/path/to/NETCDF	Installation path of NetCDF	
		Installation path of NetCDF-Fortran	
6	/path/to/CESM	Installation path of CESM	

1.3.4 Configuring the Compilation Environment

1.3.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-15 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **1.3.4.2 Installing OpenBLAS**.
3	Installing HDF5	See **1.3.4.3 Installing HDF5**.
4	Installing PnetCDF	See **1.3.4.4 Installing PnetCDF**.
5	Installing NetCDF	See **1.3.4.5 Installing NetCDF**.
6	Installing NetCDF-Fortran	See **1.3.4.6 Installing NetCDF-Fortran**.

1.3.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the OpenBLAS installation package:

```
tar xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.6
```

Step 4 Run the following commands to perform configuration:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.3.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.1
```

Step 4 Run the following commands to configure the environment before compilation:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/HDF5  --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

----End

1.3.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar -xvf parallel-netcdf-1.9.0.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd parallel-netcdf-1.9.0
```

Step 4 Run the following commands to configure the environment before compilation:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC"  CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.3.4.5 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF installation package:

```
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-4.4.1.1
```

Step 4 Run the following commands to configure the environment before compilation:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/PNETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile  CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

----End

1.3.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.1
```

Step 4 Run the following commands to configure the environment before compilation:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following command to clear compilation parameters:

```
unset CC CXX FC F77 CPPFLAGS LDFLAGS CFLAGS CXXFLAGS FCFLAGS
```

----End

1.3.5 Obtaining the Source Code

Procedure

Step 1 Download the CESM installation package **cesm-release-cesm2.1.1.tar.gz**.

https://github.com/ESCOMP/CESM/tags?after=cesm2.1-nemo.v01

Step 2 Use an SFTP tool to upload the CESM installation package to the */path/to/CESM* directory on the server.

----End

1.3.6 Compiling and Installing CESM

Prerequisites

Configure an external network so that servers can access the Internet network.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the CESM installation package and go to the directory generated after the package is decompressed:

```
tar -xvf cesm-release-cesm2.1.1.tar.gz
cd cesm-release-cesm2.1.1
```

Step 3 Run the following command to install environment-modules, git, and svn:

```
yum install git svn environment-modules -y
```

Step 4 Run the following command to add verification information:

```
svn ls https://svn-ccsm-models.cgd.ucar.edu/ww3/release_tags
```

Step 5 Run the following command to install other CESM components:

```
./manage_externals/checkout_externals
```

Step 6 Run the following command to check the component installation: (If the following information is displayed, the installation is successful.)

```
./manage_externals/checkout_externals -S
Processing externals description file : Externals.cfg
Processing externals description file : Externals_CLM.cfg
Processing externals description file : Externals_POP.cfg
Processing externals description file : Externals_CISM.cfg
```

 Copyright © Huawei Technologies Co., Ltd.

```
Checking status of externals: clm, fates, ptclm, mosart, ww3, cime, cice, pop, cvmix, marbl, cism,
source_cism, rtm, cam,
./cime
./components/cam
./components/cice
./components/cism
./components/cism/source_cism
./components/clm
./components/clm/src/fates
./components/clm/tools/PTCLM
./components/mosart
./components/pop
./components/pop/externals/CVMix
./components/pop/externals/MARBL
./components/rtm
./components/ww3
```

Step 7 Run the following command to modify the **config_machines.xml** file:

1. Open the **config_machines.xml** file.
   ```
   vi cime/config/cesm/machines/config_machines.xml
   ```
2. Press **i** to enter the insert mode and modify lines 153, 170, 171, and 176 in the **config_machines.xml** file. Pay attention to the information in bold.
   ```
   153  <NODENAME_REGEX>node1</NODENAME_REGEX>
   170  <MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
   171  <MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
   176  <arg name="ntasks"> --allow-run-as-root --mca btl ^openib  -np {{ total_tasks }} </arg>
   ```

   NOTE

   In the preceding command, **node1** indicates the host name of the current node, and **96** indicates the number of running processes. Replace them with actual values.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

1.3.7 Running and Verifying CESM

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the working directory:

```
cd /path/to/CESM/cesm-release-cesm2.1.1/cime/scripts
```

Step 3 Run the following command to create a case:

```
./create_newcase --case mycase --compset X --res f19_g16
```

Step 4 Run the following commands to download the import data:

```
cd mycase
./check_case
./check_input_data --download
```

Step 5 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:/path/to/OPENBLAS/lib:/path/to/
HDF5/lib:/path/to/PNETCDF/lib:/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF_PATH=/path/to/NETCDF
```

Step 6 Run the following commands to start the CESM program:

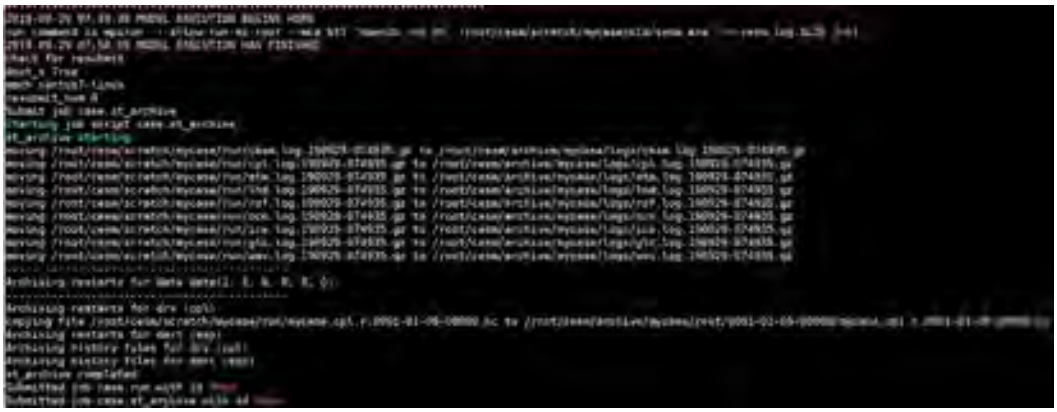
```
./case.setup
./case.build
./case.submit
```

 Copyright © Huawei Technologies Co., Ltd.

If the information shown in **Figure 1-3** is displayed at the end of the **OUTCAR** log generated by CESM, the CESM program ends normally.

View the logs displayed on the screen, as shown in the red line in **Figure 1-3**. You need to manually calculate the time difference. A smaller value indicates better performance.

Figure 1-3 Test example



----End

1.3.8 More Information

GitHub developer forum: **https://esmci.github.io/cime/index.html**

1.4 NEMO V3.6 Porting Guide (CentOS 7.6)

1.4.1 Introduction

NEMO (Nucleus for European Modeling of the Ocean) is a state-of-the-art modeling framework for research activities and forecasting services in ocean and climate sciences.

For more information about NEMO, visit the **NEMO official website**.

Programming language: C++

Brief description: Modelling framework for research activities and forecasting services in ocean and climate sciences.

Open source license: Artistic License 2.0

NOTE

For NEMO V3.6 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port NEMO to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

NEMO V3.6

 Copyright © Huawei Technologies Co., Ltd.

1.4.2 Environment Requirements

Hardware Requirements

Table 1-16 lists the hardware requirements.

Table 1-16 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-17 lists the software requirements.

Table 1-17 Software requirements

Item	Version	How to Obtain
XIOS	1.0	https://forge.ipsl.jussieu.fr/ioserver/svn/XIOS1/branches/xios-1.0/
Blitz	1.0.1	https://github.com/blitzpp/blitz.git
NEMO	V3.6	http://forge.ipsl.jussieu.fr/nemo/browser/NEMO/releases/release-3.6
PNETCDF	1.9.0	http://cucis.ece.northwestern.edu/projects/PnetCDF
NETCDF-C	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NETCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
AMM12 test case file	V3.6	https://zenodo.org/record/2640767/files/AMM12_v3.6.tar?download=1

OS Requirements

Table 1-18 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-18 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NEMO software porting.

Table 1-19 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PNETCDF	Installation path of PnetCDF	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/HDF5	Installation path of HDF5	
4	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
6	/path/to/BLITZ	Installation path of Blitz	
7	/path/to/XIOS	Installation path of XIOS	
8	/path/to/NEMO	Installation path of NEMO	

1.4.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

1.4.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-20 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing PnetCDF	See **1.4.4.2 Installing PnetCDF**.
3	Installing HDF5	See **1.4.4.3 Installing HDF5**.
4	Installing NetCDF-C	See **1.4.4.4 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.4.4.5 Installing NetCDF-Fortran**.
6	Installing Blitz	See **1.4.4.6 Installing Blitz**.
7	Installing XIOS	See **1.4.4.7 Installing XIOS**.

1.4.4.2 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install PnetCDF:

```
tar -xvf parallel-netcdf-1.9.0.tar.bz2
cd parallel-netcdf-1.9.0
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
make -j 16
make install
```

Step 3 Run the following commands to load environment variables:

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
./configure --prefix=/path/to/HDF5  --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF-C installation package:

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.4.1.1
```

Step 4 Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/install/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.1
```

Step 4 Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib "
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/ib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.6 Installing Blitz

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the Blitz installation package:

```
tar -xvf blitz-1.0.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd blitz-1.0.1/
```

Step 4 Run the following command to perform configuration:

```
autoreconf -fiv
./configure --prefix=/path/to/BLITZ --enable-fortran --enable-64bit
```

Step 5 Run the following commands to perform compilation and installation:

```
make lib
make install
cp -rf ./src /path/to/BLITZ
```

Step 6 Run the following commands to load environment variables:

```
export BLITZ_INCLUDEDIR=/path/to/BLITZ/include
export LD_LIBRARY_PATH=/path/to/BLITZ/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.7 Installing XIOS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create the XIOS installation directory:

```
mkdir -p /path/to/XIOS
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/XIOS
```

Step 4 Run the following command to download the XIOS source code:

> NOTE
>
> Before running this command, ensure that the server can access the Internet.

Step 5 Run the following commands to install the system dependency libraries:

```
yum install -y perl-lib* boost-*
```

Step 6 Run the following command to switch to the directory generated after the package is decompressed:

```
cd xios-1.0
```

Step 7 Run the following commands to modify the **bld.cfg** file:

1. Open **bld.cfg**.
   ```
   vi bld.cfg
   ```
2. Press **I** to enter the insert mode and modify the **bld.cfg** file. Pay attention to the information in bold.
   ```
   search_src            true
   src::zzz .
   #src::date $PWD/extern/boost/src/date_time
   src::blitz $PWD/extern/blitz/src
   src::netcdf $PWD/extern/netcdf4
   bld::lib xios
   bld::target libxios.a
   #bld::target generate_fortran_interface.exe
   #bld::target xios_server.exe test_client.exe parse_xml.exe test_complete.exe
   ```

```
test_xios_interface.exe
bld::exe_dep
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to create and edit the **arch-AARCH64_GNU_LINUX.env** configuration file:

1. Create an **arch-AARCH64_GNU_LINUX.env** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.env
   ```
2. Press **i** to enter the insert mode and create and edit the **arch-AARCH64_GNU_LINUX.env** file.

   Edit the file based on the actual environment variables and dependency library installation path.

   ```
   export HDF5_INC_DIR="/path/to/HDF5/include"
   export HDF5_LIB_DIR="/path/to/HDF5/lib"

   export NETCDF_INC_DIR="/path/to/NETCDF/include"
   export NETCDF_LIB_DIR="/path/to/NETCDF/lib"

   export BOOST_INC_DIR="/usr/include/boost/"
   export BOOST_LIB_DIR="/usr/lib64/"

   export BLITZ_INC_DIR="/path/to/BLITZ/include"
   export BLITZ_LIB_DIR="/path/to/BLITZ/lib"
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following command to create and edit the **arch-AARCH64_GNU_LINUX.fcm** configuration file:

1. Create an **arch-AARCH64_GNU_LINUX.fcm** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.fcm
   ```
2. Press **i** to enter the insert mode and create and edit the **arch-AARCH64_GNU_LINUX.fcm** file.

   Edit the file based on the actual environment variables and dependency library installation path.

   ```
   ############################################################################
   ###########
   ####################       Projet xios - xmlioserver       ####################
   ############################################################################
   ###########

   %CCOMPILER      mpicc
   %FCOMPILER      mpif90
   %LINKER         mpif90

   %BASE_CFLAGS    -ansi -w -D_GLIBCXX_USE_CXX11_ABI=0
   %PROD_CFLAGS    -O3 -DBOOST_DISABLE_ASSERTS
   %DEV_CFLAGS     -g -O2
   %DEBUG_CFLAGS   -g

   %BASE_FFLAGS    -D__NONE__
   %PROD_FFLAGS    -O3
   %DEV_FFLAGS     -g -O2
   %DEBUG_FFLAGS   -g

   %BASE_INC       -D__NONE__
   %BASE_LD        -lstdc++

   %CPP            cpp
   %FPP            cpp -P
   %MAKE           gmake
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Run the following command to create and edit the **arch-AARCH64_GNU_LINUX.path** configuration file:

1. Create an **arch-AARCH64_GNU_LINUX.path** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.path
   ```
2. Press **i** to enter the insert mode and create and edit the **arch-AARCH64_GNU_LINUX.path** file.

   Edit the file based on the actual environment variables and dependency library installation path.

   ```
   NETCDF_INCDIR="-I /path/to/NETCDF/include"
   NETCDF_LIBDIR="-L /path/to/NETCDF/lib"
   NETCDF_LIB="-lnetcdff -lnetcdf"

   MPI_INCDIR="-I /path/to/OPENMPI/include"
   MPI_LIBDIR="-L /path/to/OPENMPI/lib"
   MPI_LIB="-lmpi"

   HDF5_INCDIR="-I /path/to/HDF5/include"
   HDF5_LIBDIR="-L /path/to/HDF5/lib"
   HDF5_LIB="-lhdf5_hl -lhdf5  -lz"

   BOOST_INCDIR="-I /usr/include/boost/"
   BOOST_LIBDIR="-L /usr/lib64/"
   BOOST_LIB=""

   BLITZ_INCDIR="-I /path/to/BLITZ/include"
   BLITZ_LIBDIR="-L /path/to/BLITZ/lib"
   BLITZ_LIB="-lblitz"
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Run the following commands to perform the first compilation and installation:

```
cd /path/to/XIOS/xios-1.0
chmod +x ./make_xios
cd extern
ln -s /path/to/BLITZ ./blitz
ln -s /path/to/NETCDF ./netcdf4
ln -s /usr/include/boost ./boost
cd ..
./make_xios --dev --job 32 --full --arch AARCH64_GNU_LINUX
```

Step 12 Run the following commands to perform the second compilation and installation:

If an error is reported after the first compilation, run the following command again:

```
cd extern
rm -rf ./blitz
rm -rf ./netcdf4
rm -rf ./boost
ln -s /path/to/BLITZ ./blitz
ln -s /path/to/NETCDF ./netcdf4
ln -s /usr/include/boost ./boost
cd ..
./make_xios --dev --job 32 --full --arch AARCH64_GNU_LINUX
```

----End

1.4.5 Obtaining the Source Code

Procedure

Step 1 Download the NEMO installation package **NEMO_releases_release-3.6-10083.zip**.

 Copyright © Huawei Technologies Co., Ltd.

http://www.nemo-ocean.eu/

Step 2 Use an SFTP tool to upload the NEMO installation package to the */path/to/NEMO* directory on the server.

----End

1.4.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create and go to the NEMO installation directory:

```
mkdir -p /path/to/NEMO
cd /path/to/NEMO
```

Step 3 Run the following commands to configure the NEMO environment variables:

```
echo "export NETCDF_DIR=/path/to/NETCDF" >> nemo-env.sh
echo "export HDF_DIR=/path/to/HDF5" >> nemo-env.sh
echo "export XIOS_DIR=/path/to/XIOS/xios-1.0" >> nemo-env.sh
source nemo-env.sh
```

Step 4 Run the following command to decompress the installation package:

```
unzip NEMO_releases_release-3.6-10083.zip
```

Step 5 Run the following command to go to the NEMO source code directory:

```
cd NEMO/releases/release-3.6/NEMOGCM
```

Step 6 Run the following command to create an **arch-aarch64_gnu.fcm** file:

1. Create **arch-aarch64_gnu.fcm**.
   ```
   vi ARCH/arch-aarch64_gnu.fcm
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   #generic gfortran compiler options for linux
   #NCDF_INC    netcdf include file
   #NCDF_LIB    netcdf library
   #FC          Fortran compiler command
   #FCFLAGS     Fortran compiler flags
   #FFLAGS      Fortran 77 compiler flags
   #LD          linker
   #LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
   #FPPFLAGS    pre-processing flags
   #AR          assembler
   #ARFLAGS     assembler flags
   #MK          make
   #USER_INC    additional include files for the compiler,  e.g. -I<include dir>
   #USER_LIB    additional libraries to pass to the linker, e.g. -l<library>
   #CC          C compiler used to compile conv for AGRIF
   #CFLAGS      compiler flags used with CC
   #
   #Note that:
   # - unix variables "$..." are accpeted and will be evaluated before calling fcm.
   # - fcm variables are starting with a % (and not a $)
   %NCDF_HOME           $NETCDF_DIR
   %HDF5_HOME           $HDF_DIR
   %XIOS_HOME           $XIOS_DIR

   %NCDF_INC            -I%NCDF_HOME/include -I%HDF5_HOME/include
   %NCDF_LIB            -L%HDF5_HOME/lib -L/%NCDF_HOME/lib -lnetcdf -lnetcdff
   %XIOS_INC            -I%XIOS_HOME/inc
   %XIOS_LIB            -L%XIOS_HOME/lib -lxios

   %CPP                 cpp -Dkey_nosignedzero
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
%FC                mpif90 -c -cpp
%FCFLAGS             -mcpu=native -fdefault-real-8 -fdefault-double-8 -O3 -funroll-all-loops -fcray-
pointer -ffree-line-length-none -g
%FFLAGS              %FCFLAGS
%LD                mpif90
%LDFLAGS             -lstdc++
%FPPFLAGS            -P -C -traditional
%AR                ar
%ARFLAGS             rs
%MK                gmake
%USER_INC            %XIOS_INC %NCDF_INC
%USER_LIB            %XIOS_LIB %NCDF_LIB

%CC                cc
%CFLAGS              -O0
```

NOTE

Ensure that no space exists at the end of a line or empty line.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following commands to compile and install the software:

```
chmod 777 ./ -R
cd ./CONFIG/
./makenemo -m aarch64_gnu -j 32 -r AMM12 -n 'MY_AMM12' add_key "key_nosignedzero"
```

----End

1.4.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the work directory:

```
cd /path/to/NEMO/NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00
```

Step 3 Upload **AMM12_v3.6.tar** to */path/to/NEMO/***NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00**.

Step 4 Run the following commands to decompress the package:

```
tar xf AMM12_v3.6.tar
gunzip *.gz */*.gz
rm domain_def.xml field_def.xml namelist_ref -f
ln -s ../../SHARED/domain_def.xml ./domain_def.xml
ln -s ../../SHARED/field_def.xml ./field_def.xml
ln -s ../../SHARED/namelist_ref ./namelist_ref
```

Step 5 Run the following command to create a **host** file:

1. Create **host**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   node1
   node2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to start the NEMO program.

{ time mpirun --allow-run-as-root -np 192 -N 96 -hostfile hostfile --mca btl ^openib ./opa ; } 2>&1 | tee -a NEMO.log

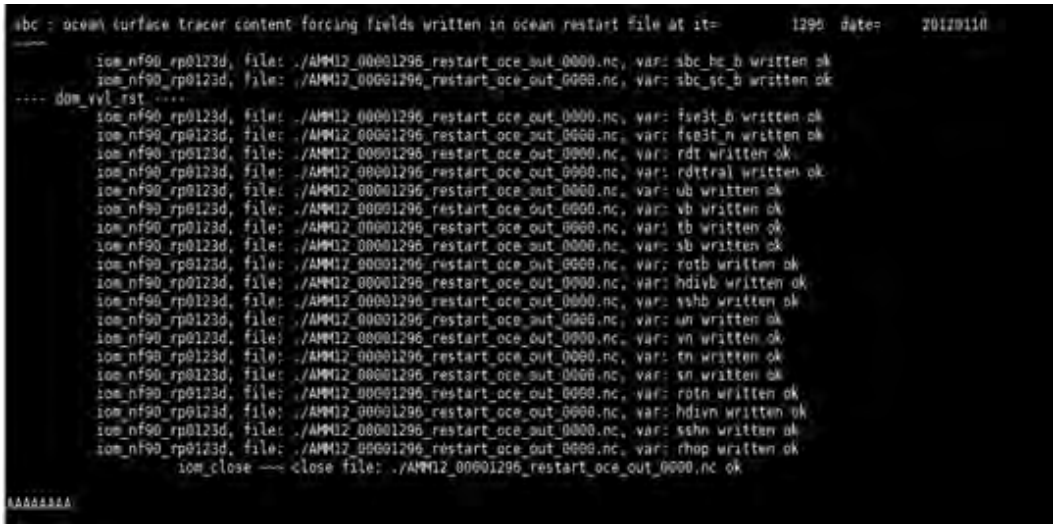
 Copyright © Huawei Technologies Co., Ltd.

Table 1-21 Parameters

Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
--hostfile	Name of the node to be used.

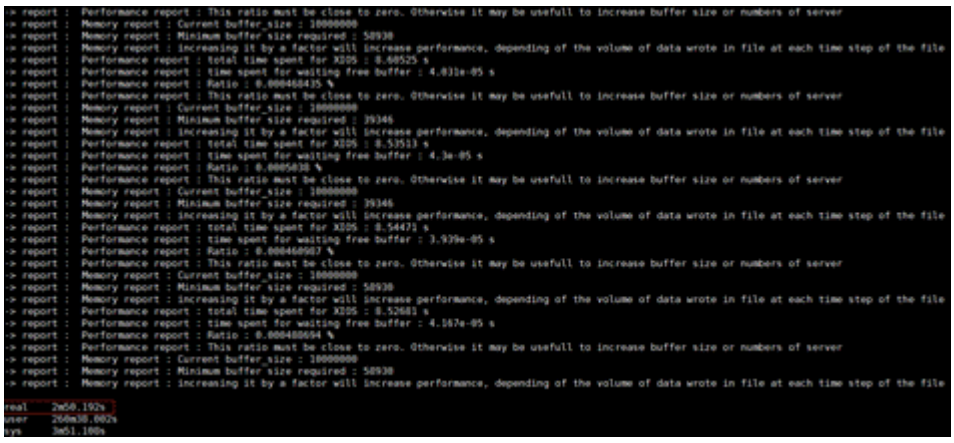
If the information shown in **Figure 1-4** is displayed at the end of the **ocean.output** log generated by NEMO, the NEMO program ends normally.

Figure 1-4 Test example



Check the value of **real** in the **NEMO.log** file. The unit is second. A higher the value indicates a lower performance. **Figure 1-5** shows the output result.

Figure 1-5 Test example



----End

 Copyright © Huawei Technologies Co., Ltd.

1.5 ROMS 3.6 Porting Guide (CentOS 7.6)

1.5.1 Introduction

The Regional Ocean Modeling System (ROMS) is a 3D regional ocean modeling system developed by the Rutgers University, University of California Los Angeles. It is widely used to simulate the hydrodynamic and water environment in the ocean and estuary.

For more information about the ROMS, visit the **official ROMS website**.

Programming language: Fortran

Brief description: 3D regional ocean modeling system

Open source license: MIT/X License

Recommended Software Version

ROMS 3.6

1.5.2 Environment Requirements

Hardware Requirements

Table 1-22 lists the hardware requirements.

Table 1-22 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-23 lists the software requirements.

Table 1-23 Software requirements

Item	Version	How to Obtain
ROMS	3.6	**https://www.myroms.org/index.php?page=login**
PNETCFD	1.11.2	**http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz**
NETCFD-C	4.7.0	**https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
NETCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5
HDF5	1.10.5	https://portal.hdfgroup.org/display/support/HDF5+1.10.5#files
Test case	ocean_upwelling	Calculation case provided by the software

OS Requirements

Table 1-24 lists the OS requirements.

Table 1-24 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.5.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ROMS software porting.

Table 1-25 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/PNETCDF	Installation path of PnetCDF	
4	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	/path/to/ROMS	Installation path of ROMS	

1.5.4 Configuring the Compilation Environment

1.5.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-26 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Creating an environment variable file	See **1.5.4.2 Creating an Environment Variable File**.
3	Installing HDF5	See **1.5.4.3 Installing HDF5**.
4	Installing PnetCDF	See **1.5.4.4 Installing PnetCDF**.
5	Installing NetCDF-C	See **1.5.4.5 Installing NetCDF-C**.
6	Installing NetCDF-Fortran	See **1.5.4.6 Installing NetCDF-Fortran**.

1.5.4.2 Creating an Environment Variable File

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to create the **env.sh** environment variable file:

1. Create **env.sh**.
   ```
   vi /root/env.sh
   ```
2. Press **i** to enter the insert mode and edit the **env.sh** file.
   ```
   #!/bin/bash
   export GNU_HOME=/path/to/GNU
   export MPI_HOME=/path/to/OPENMPI
   export HDF5=/path/to/HDF5
   export PNETCDF=/path/to/PNETCDF
   export NETCDF=/path/to/NETCDF
   export PHDF5=$HDF5
   export CPPFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/
   include "
   export CFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/include
   "
   export CXXFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/
   include "
   export FCFLAGS=" -I$HDF5/include -I$PNETCDF/include"
   export FFLAGS=" -I$HDF5/include -I$PNETCDF/include"
   export LDFLAGS=" -L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib"
   export LD_LIBRARY_PATH=$HDF5/lib:$PNETCDF/lib:$NETCDF/lib:$MPI_HOME/lib: $GNU_HOME/
   lib64:$LD_LIBRARY_PATH
   export PATH=$NETCDF/bin:$PNETCDF/bin:$MPI_HOME/bin:$GNU_HOME/bin:$PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Run the following command to make the configuration file take effect:

```
source /root/env.sh
```

----End

1.5.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar zxvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install HDF5:

```
make -j 16
make install
```

----End

1.5.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar zxvf pnetcdf-1.11.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd pnetcdf-1.11.2
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install PnetCDF:

```
make -j 16
make install
```

----End

1.5.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF-C installation package:

```
tar zxvf netcdf-c-4.7.0.tar.gz
yum install curl.aarch64 libcurl.aarch64 libcurl-devel.aarch64 -y
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf  --enable-large-file-tests --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install NetCDF-C:

```
make -j 16
make install
```

----End

1.5.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform configuration:

 Copyright © Huawei Technologies Co., Ltd.

```
./configure --prefix=$NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install NetCDF-Fortran:

```
make -j 16
make install
```

----End

1.5.5 Obtaining the Source Code

Procedure

Step 1 Download the ROMS installation package **ROMS_3.6**.

https://www.myroms.org/

NOTE

To download the ROMS installation package, you need to register an account on the ROMS website and log in to the website using the registered account.

Step 2 Use an SFTP tool to upload the ROMS installation package to the */path/to/ROMS* directory on the server.

----End

1.5.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use SVN to check out and obtain the ROMS source code.

Create a folder and check out the ROMS source code to the folder. To obtain the source code, you need to register an account.

```
cd /path/to/ROMS
mkdir ROMS_3.6
svn checkout --username Account --password Password https://www.myroms.org/svn/src/trunk ROMS_3.6
```

Example:

Run the following command to check out the source code using **HPC/Huawei12#$**:

```
svn checkout --username HPC --password Huawei12#$ https://www.myroms.org/svn/src/trunk ROMS_3.6
```

The source code file structure is as follows:

 Copyright © Huawei Technologies Co., Ltd.

```
trunk/                          Main trunk directory
     /Atmosphere/               Atmosphere models root directory
               /COAMPS          COAMPS root directory (empty)
               /WRF             WRF root directory (empty)
     /Compilers/                make configuration files
     /Data/                     Input data root directory
          /ROMS/                ROMS data root directory
               /CDL             ROMS NetCDF metadata design
               /Forcing         Input test cases forcing NetCDF files
               /Grid            Input test cases grid NetCDF files
               /Initial         Input test cases initial conditions NetCDF files
     /Lib/                      External libraries
         /ARPACK                Arpack eigenvalue problems library
         /MCT                   Modeling Coupling Tool library
     /Master                    Main standalone and coupling programs
     /ROMS/                     ROMS root directory
          /Adjoint              Adjoint model
                 /Biology       Adjoint biology/ecosystem models
          /Bin                  Executable scripts
          /Drivers              Computational drivers
          /External             Standard input scripts
          /Functionals          Analytical expression header files
          /Include              Test cases configuration header files
          /Modules              Declaration modules
          /Nonlinear            Nonlinear model
                   /Biology     Nonlinear biology/ecosystem models
                   /Sediment    Nonlinear sediment transport model
          /Obsolete             Discontinued files
          /Programs             Support programs
          /SeaIce               Sea-ice model (empty)
          /Representer          Representer model
                     /Biology   Representer biology/ecosystem models
          /Tangent              Tangent linear model
                  /Biology      Tangent linear biology/ecosystem models
          /Utility              Generic utility files
          License_ROMS.text     Open source license
          Version               SVN Version information
     /User/                     ROMS User interface root directory
          /External             User standard input scripts
          /Functionals          User analytical expressions templates
          /Include              User application header files
     /WAVES/                    Waves models root directory
           /SWAN                SWAN root directory
                /External       SWAN input data and standard input files
                /Src            SWAN model
```

Step 3 Customize the build script.

1. To keep the source code unchanged, run the following command to create a **ROMSProjects** folder for compiling and running the ROMS. **upwelling** is the default example in the source code package.

   ```
   cd /path/to/ROMS
   mkdir ROMSProjects
   cd ROMSProjects
   mkdir upwelling
   cd upwelling
   ```

2. Run the following command to copy the **.bash** file to the **upwelling** directory in the current environment. The **build.sh** and **build.bash** files exist in the **ROMS/Bin** directory of the source code.

   ```
   cp /path/to/ROMS/ROMS_3.6/ROMS/Bin/build.bash .
   ```

3. Run the following command to edit the **build.bash** file:

   a. Open **build.bash**.

   ```
   vi build.bash
   ```

   b. Press **i** to enter the insert mode and modify the file as follows:

   ```
   export ROMS_APPLICATION=UPWELLING
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
export MY_ROOT_DIR=/path/to/ROMS
export MY_PROJECT_DIR=${MY_ROOT_DIR}/ROMSProjects/upwelling
export MY_ROMS_SRC=/path/to/ROMS/ROMS_3.6
export USE_MPI=on
export USE_MPIF90=on
export which_MPI=openmpi
export FORT=gfortran
export USE_LARGE=on
export USE_NETCDF4=on
export USE_PARALLEL_IO=on
export PATH=/path/to/OPENMPI/bin:$PATH
export NF_CONFIG=/path/to/NETCDF/bin/nf-config
export NETCDF_INCDIR=/path/to/NETCDF/include
export NETCDF_LIBDIR=/path/to/NETCDF/lib
```

c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following commands to copy the input and configuration files:

```
cp /path/to/ROMS/ROMS_3.6/ROMS/External/ocean_upwelling.in .
cp /path/to/ROMS/ROMS_3.6/ROMS/External/varinfo.dat
cp /path/to/ROMS/ROMS_3.6/ROMS/Include/upwelling.h .
```

Step 5 Run the following command to edit the **ocean_upwelling.in** file:

1. Open **ocean_upwelling.in**.
   ```
   vi ocean_upwelling.in
   ```
2. Press **i** to enter the insert mode and modify the **ocean_upwelling.in** file.

   Change the path whose **VARNAME** is **varinfo.dat**. Pay attention to the information in bold.
   ```
   VARNAME = /path/to/ROMS/ROMSProjects/upwelling/varinfo.dat
   ```

   Set **NtileI** and **NtileJ**. The product of **NtileI** and **NtileJ** must be equal to the value of **np**. For example, if **np** is set to **96**, you can set **NtileI** and **NtileJ** as follows (pay attention to the information in bold):
   ```
   NtileI == 6
   NtileJ == 16
   ```

   NOTE

   For details about how to set **NtileI** and **NtileJ**, see 1.5.8 Troubleshooting.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Compile ROMS.

```
./build.bash -j 8
```

When MPI is used, the executable file **OceanM** is generated after successful compilation. You can run the **ls** command to view the file.

```
ll /path/to/ROMS/ROMSProjects/upwelling
-rwxr-xr-x 1 root root 1671056 Aug 15 18:41 oceanM
```

----End

1.5.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/ROMS/ROMSProjects/upwelling
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to start ROMS:

```
mpirun --allow-run-as-root --mca btl ^openib -np 96 ./oceanM ocean_upwelling.in > log 2>&1
```

The following files are generated: **ocean_avg.nc**, **ocean_dia.nc**, **ocean_his.nc**, and **ocean_rst.nc**.

Check the value of **Node# CPU: 12.334** in the log file. The unit is second. A smaller value indicates better performance.

Figure 1-6 shows the command output.

Figure 1-6 Result example

```
Tile partition information for Grid 01:  0041x0080x0016  tiling: 006x016

    tile     Istr     Iend     Jstr     Jend     Npts

       0        1        7        1        5      560
       1        8       14        1        5      560
       2       15       21        1        5      560
       3       22       28        1        5      560
       4       29       35        1        5      560
       5       36       41        1        5      480
       6        1        7        6       10      560
       7        8       14        6       10      560
       8       15       21        6       10      560
       9       22       28        6       10      560
      10       29       35        6       10      560
      11       36       41        6       10      480
      12        1        7       11       15      560
      13        8       14       11       15      560
      14       15       21       11       15      560
```

Figure 1-7 shows an example of the CPU running time.

Figure 1-7 Result example

```
Tile partition information for Grid 01:  0041x0080x0016  tiling: 006x016

    tile     Istr     Iend     Jstr     Jend     Npts

       0        1        7        1        5      560
       1        8       14        1        5      560
       2       15       21        1        5      560
       3       22       28        1        5      560
       4       29       35        1        5      560
       5       36       41        1        5      480
       6        1        7        6       10      560
       7        8       14        6       10      560
       8       15       21        6       10      560
       9       22       28        6       10      560
      10       29       35        6       10      560
      11       36       41        6       10      480
      12        1        7       11       15      560
      13        8       14       11       15      560
      14       15       21       11       15      560
```

----End

1.5.8 Troubleshooting

Problem 1: The format of the source code checked out in Windows is inconsistent with that in Linux.

Symptom

The format of the source code checked out in Windows is different from that in Linux.

Procedure

 Copyright © Huawei Technologies Co., Ltd.

Install and use dos2unix to convert the entire directory checked out by SVN in Windows to the Linux format.

```
yum -y install dos2unix*
cd /home/hesj/tool/ROMS_916
find . -type f -exec dos2unix {} \;
```

1.5.9 More Information

Wikipedia:

https://www.myroms.org/wiki/Getting_Started

1.6 WAVEWATCH III 6.07.1 Porting Guide (CentOS 7.6)

1.6.1 Introduction

WAVEWATCH III is a community wave modeling framework that includes the latest scientific advances in wave modeling and dynamics.

WAVEWATCH III resolves the random-phase spectrum action balance equation of the wavenumber direction spectrum. The model includes options for shallow water (surfing area) applications and wetting and drying of grid points.

For details about WAVEWATCH III, visit the **official WAVEWATCH III website**.

Programming language: Fortran

Brief description: Third-generation wave mode.

Open source license: custom open source license

Recommended Software Version

WAVEWATCH III 6.07.1

1.6.2 Environment Requirements

Hardware Requirements

Table 1-27 lists the hardware requirements.

Table 1-27 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-28 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-28 Software requirements

Item	Version	Download Address
WAVEWATCH III	6.07.1	**https://github.com/NOAA-EMC/WW3/releases/tag/6.07.1**
HDF5	1.10.1	**http://hdfgroup.org/HDF5/**
NETCDF	4.4.1.1	**https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1**
NETCDF-Fortran	4.4.1	**https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1**
ParMETIS	4.0.3	**http://pkgs.freefem.org/ParMetis-4.0.3.tar.gz**
Test case	mww3_data_01	-

OS Requirements

Table 1-29 lists the OS requirements.

Table 1-29 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.6.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WAVEWATCH III software porting.

Table 1-30 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only examples. Shared paths

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/NETCDF	Installation path of NetCDF	are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
		Installation path of NetCDF-Fortran	
4	/path/to/METIS	Installation path of ParMETIS	
5	/path/to/WAVEWATCH	Installation path of WAVEWATCH III	
6	/path/to/CASE	Test path of WAVEWATCH III	

1.6.4 Configuring the Compilation Environment

1.6.4.1 Configuration Description

Prerequisites

Use SFTP to upload the installation packages to the planned directories on the server.

Configuration Process

Table 1-31 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Installing HDF5	See **1.6.4.2 Installing HDF5**.
3	Installing NetCDF	See **1.6.4.3 Installing NetCDF**.
4	Installing NetCDF-Fortran	See **1.6.4.4 Installing NetCDF-Fortran**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
5	Installing ParMETIS	See **1.6.4.5 Installing ParMETIS**.

1.6.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd hdf5-1.10.1
```

Step 4 Run the following command perform configuration:

```
./configure --prefix=/path/to/HDF5  --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 20
make install
```

Step 6 Run the following commands to load environment variables:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
export F77=$MPIFC
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.6.4.3 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF installation package:

```
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd netcdf-4.4.1.1
```

Step 4 Run the following command perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-largefile CPPFLAGS="-I/
```

 Copyright © Huawei Technologies Co., Ltd.

```
path/to/HDF5/include " LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-L/path/to/HDF5/lib -I/path/to/HDF5/include "
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 20
make install
```

----End

1.6.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd netcdf-fortran-4.4.1
```

Step 4 Run the following command perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"  LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib "  CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include "  CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include "  FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include "
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 20
make install
```

Step 6 Run the following command to delete the environment variables:

```
unset CC CXX FC F77 MPICC MPICXX MPIFC
```

----End

1.6.4.5 Installing ParMETIS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the ParMETIS installation package:

```
tar -xvf parmetis-4.0.3.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd parmetis-4.0.3
```

Step 4 Run the following command to modify the **metis/include/metis.h** file:

1. Open **metis/include/metis.h**.
   ```
   vi metis/include/metis.h
   ```

2. Press **i** to enter the insert mode and modify the content in line 33.
   ```
   #define IDXTYPEWIDTH 64
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to install CMake.

```
yum install cmake -y
```

Step 6 Run the following commands to go to the **metis** directory and compile and install METIS:

```
cd metis/
make config shared=1 prefix=/path/to/METIS
make install
```

Step 7 Run the following commands to go to the upper-layer directory and perform compilation and installation:

```
cd ../
make config shared=1 prefix=/path/to/METIS
make install
```

----End

1.6.5 Obtaining the Source Code

Procedure

Step 1 Download the WAVEWATCH III installation package **WW3-6.07.1.tar.gz**.

Download address: **https://polar.ncep.noaa.gov/waves/wavewatch/**.

Step 2 Use SFTP to upload the WAVEWATCH III installation package to the */path/to/WAVEWATCH* directory on the server.

----End

1.6.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the directory where WAVEWATCH III is installed:

```
cd /path/to/WAVEWATCH
```

Step 3 Run the following command to decompress the WAVEWATCH III source code package:

```
tar -xvf WW3-6.07.1.tar.gz
```

Step 4 Run the following command to switch to the directory containing the decompressed files:

```
cd  WW3-6.07.1
```

Step 5 Run the following command to create the environment variable file:

1. Create an environment variable file.
   ```
   vi env_ww3.sh
   ```
2. Press **i** to enter the insert mode and add the following content:

 Copyright © Huawei Technologies Co., Ltd.

```
############## gnu + openmpi #################
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OpenMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OpenMPI/lib:$LD_LIBRARY_PATH

############## netcdf #################
export NETCDF_DIR=/path/to/NETCDF
export NETCDF_LIBDIR=$NETCDF_DIR/lib
export NETCDF_INCDIR=$NETCDF_DIR/include
export PATH=$NETCDF_DIR/bin:$PATH
export LD_LIBRARY_PATH=$NETCDF_LIBDIR:$LD_LIBRARY_PATH
export WWATCH3_NETCDF=NC4
export NETCDF_CONFIG=$NETCDF_DIR/bin/nc-config

############## metis #################
export METIS_PATH=/path/to/METIS
export PATH=$METIS_PATH/bin:$PATH
export LD_LIBRARY_PATH=$METIS_PATH/lib:$LD_LIBRARY_PATH
```

NOTE

/path/to/GNU indicates the GNU 9.1 installation path, */path/to/OPENMPI* indicates the Open MPI installation path, */path/to/NETCDF* indicates the NetCDF and NetCDF-Fortran installation path, and */path/to/METIS* indicates the ParMETIS installation path.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to load environment variables:

```
source env_ww3.sh
```

Step 7 Run the following command to configure the WAVEWATCH III parameters (pay attention to the information in bold):

```
./model/bin/w3_setup  model/
*****************************
***   WAVEWATCH III setup     ***
*****************************


[INFO] local env file wwatch3.env created in /home/WW3-6.07.1/model/bin/wwatch3.env
Setup file /home/WW3-6.07.1/model/bin/wwatch3.env not found

Default set up :
Printer (listings)       : printer
Auxiliary FORTRAN comp.  : gfortran
Auxiliary C compiler     : gcc
Scratch directory        : /path/to/WAVEWATCH/WW3-6.07.1/model/tmp
Save sources             : yes
Save listings            : yes
Update settings ? [y/n] y

Creating new set-up :

Printer for listings [printer] :
Auxiliary FORTRAN compiler [gfortran] :
Auxiliary C compiler [gcc] :
Scratch space [/home/wdy/WW3-6.07.1/model/tmp] :
Save source code files (*.f)  [yes] :
Save listing files  [yes] :

Modified set up :
Printer (listings)       : printer
Auxiliary FORTRAN comp.  : gfortran
Auxiliary C compiler     : gcc
Scratch directory        : /path/to/WAVEWATCH III/WW3-6.07.1/model/tmp
Save sources             : yes
Save listings            : yes
New settings OK ? [y/n] y
```

 Copyright © Huawei Technologies Co., Ltd.

```
Setup makefile for auxiliary programs
......
Create required model subdirectories
Finished setting up WAVEWATCH III
```

----End

1.6.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create and go to the working directory:

```
mkdir /path/to/WW3_TEST
cd /path/to/WW3_TEST
```

Step 3 Download the test case and copy it to the */path/to/WW3_TEST* directory.

Step 4 Run the following commands to decompress the package:

```
tar -xvf ww3_from_ftp.v6.07.tar.gz
cd cases
```

Step 5 Run the following commands to set environment variables:

```
source /path/to/WAVEWATCH/WW3-6.07.1/env_ww3.sh
export ww3_dir=/path/to/WAVEWATCH/WW3-6.07.1/model
```

Step 6 Run the following commands to modify the **mww3_case_01** file:

1. Open **mww3_case_01**.
   ```
   vi  mww3_case_01
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Line 61:
   ```
   #cd /scratch1/portfolios/NCEPDEV/ocean/noscrub/Henrique.Alves/WW3_CASES/cases
   ```
   Line 65:
   ```
   #module load intel mpt grads
   ```
   Line 72:
   ```
   compstr="Gnu"
   ```
   Line 143:
   ```
   #ww3_dir='echo $case_dir | sed 's/\/cases\/*//g''
   ```
   Line 138:
   ```
   proc=96
   ```
   Line 420:
   ```
   mpirun --allow-run-as-root --mca btl ^openib -np $proc --hostfile /path/to/WW3_TEST/hostfile
   $path_e/ww3_multi
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to create a **host** file:

1. Create **host**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   node1
   node2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Open another login window and use PuTTY to log in to node 2 as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 9 On node 2, run the following command to configure environment variables:

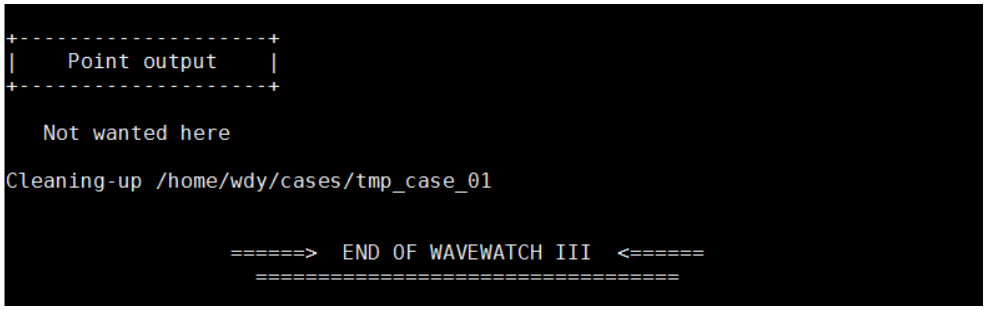
1. Open **.bashrc**.
   ```
   vi /root/.bashrc
   ```
2. Press **i** to enter the insert mode, and add the following content to the end of the **.bashrc** file:
   ```
   export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 On node 1, run the following command to start the WAVEWATCH III program:

```
./mww3_case_01
```

If the information shown in **Figure 1-8** is displayed when the WAVEWATCH III program is complete, the WAVEWATCH III program is running properly.

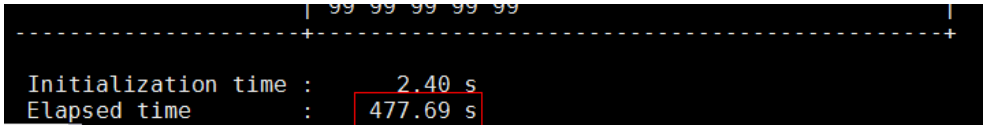
Figure 1-8 Result example



Check the value of **Elapsed time** in the **work_case_01/log.mww3** log file. The unit is second. A smaller value indicates better performance.

Figure 1-9 shows the command output.

Figure 1-9 Result example



----End

1.6.8 Support and Other Resources

WAVEWATCH III quick installation guide:

https://github.com/NOAA-EMC/WW3/wiki/Quick-Start

1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)

1.7.1 Introduction

Bifrost is a numerical model software for star convection and light bulbs.

For more information about Bifrost, visit the **official Bifrost website**.

 Copyright © Huawei Technologies Co., Ltd.

Programming language: Fortran

Brief description: numerical model software for star convection and light bulbs

Open source license: custom open source license

Recommended Software Version

Bifrost V1.1.1

1.7.2 Environment Requirements

Hardware Requirements

Table 1-32 lists the hardware requirements.

Table 1-32 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-33 lists the software requirements.

Table 1-33 Software requirements

Item	Version	How to Obtain
Bifrost	V1.1.1	https://www.sigma2.no/procurement-project-hpc-b1

OS Requirements

Table 1-34 lists the OS requirements.

Table 1-34 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

1.7.3 Path for Software Porting

This chapter lists the software installation paths involved in the Bifrost software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-35 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BIFROST	Installation path of Bifrost	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

1.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-36 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.7.5 Obtaining the Source Code

Procedure

Step 1 Download the Bifrost installation package **Bifrost_bench_v1.1.1-sigma2.tgz**.

https://www.sigma2.no/procurement-project-hpc-b1

Step 2 Use an SFTP tool to upload the Bifrost installation package to the */path/to/BIFROST* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.7.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Bifrost installation package:

```
cd /path/to/BIFROST
tar -xvf Bifrost_bench_v1.1.1-sigma2.tgz
cd Bifrost_bench_v1.1.1
```

Step 3 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **I** to enter the insert mode and modify the **Makefile** file. Pay attention to the information in bold.
   ```
   # set operating system dependent parameters
   F90              ?= mpifort
   F77               = $(F90)
   CC                = mpicc
   LD                = $(F90)
   F90FLAGS  ?= -O3 -ffree-line-length-0 -fno-range-check
   F77FLAGS   = $(F90FLAGS)
   LIBS       =
   INCL       =
   DEBUG      =
   LDFLAGS   ?= $(F90FLAGsS) $(DEBUG) $(INCL)
   OUTP      += compiler
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to modify the **mpi.f90** file:

1. Open **mpi.f90**.
   ```
   vi mpi.f90
   ```
2. Press **i** to enter the insert mode and modify lines 776 and 778 in the **mpi.f90** file. Pay attention to the information in bold.
   ```
   # if(gridsize < 1000) then
     fmti="(' ',I3)"
   else
     fmti="(' ',I4)"
   endif
   do k=dims(3)-1,0,-1
     do j=dims(2)-1,0,-1
       fmt="(A1)"
       do kk=0,(j-1)*2+5
         write (*,fmt,ADVANCE='NO') " "
       end do
       fmt=fmti
       do i=0,dims(1)-1
         do n=0,gridsize-1
           if (i.eq.(griddat(1,n)).and.(j.eq.griddat(2,n)).and.(k.eq.griddat(3,n))) write
   (*,fmt,ADVANCE='NO') n
         end do
       end do
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to perform compilation and installation:

```
make
```

Step 6 Run the following command to view whether the **photo_tr.x** file is generated:

```
ll RUNS/photo_tr.x
```

 Copyright © Huawei Technologies Co., Ltd.

```
-rwxr-xr-x 1 root root 2521960 Jan 11 14:39 RUNS/photo_tr.x
```

----End

1.7.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the **RUNS** directory:

```
cd /path/to/BIFROST/Bifrost_bench_v1.1.1/RUNS
```

Step 3 Run the following command to create a **hostfile** file:

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   node1
   node2
   ```

   NOTE

   In the preceding command, **node1** and **node2** are the host names of the test servers. Change them based on the site requirements.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Open another login window and use PuTTY to log in to node 2 as the **root** user.

Step 5 On node 2, run the following command to configure environment variables:

1. Open **.bashrc**.
   ```
   vi /root/.bashrc
   ```
2. Press **i** to enter the insert mode, and add the following content to the end of the **.bashrc** file:
   ```
   export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 On node 1, run the following command to start the Bifrost program:

```
mpirun --allow-run-as-root -np 192 -N 96 -hostfile hostfile photo_tr.x 2>&1 | tee -a Bifrost.out
```

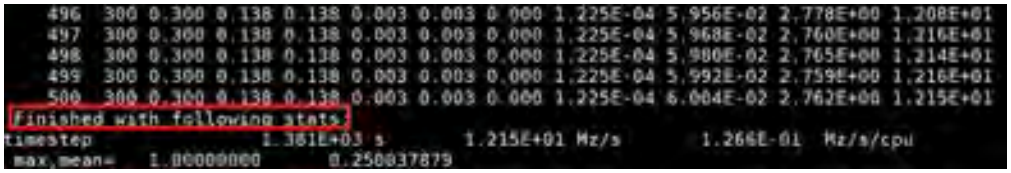
Table 1-37 Parameters

Parameter	Description
-np	Total number of processes in use
-N	Number of processes running on each server
-hostfile	Host name of the node in use

If the information shown in **Figure 1-10** is displayed at the end of the generated log file **Bifrost.out** after 500 iterations, the Bifrost program is running properly.

 Copyright © Huawei Technologies Co., Ltd.

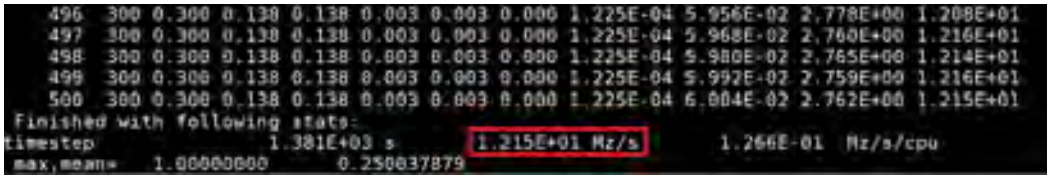
Figure 1-10 Result example



Check the value of **Mz/s** (unit: Mz/s) in the **Bifrost.out** log. A greater value indicates better performance.

Figure 1-11 shows the command output.

Figure 1-11 Result example



----End

1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)

1.8.1 Introduction

FVCOM is an unstructured-grid, finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model. The model consists of momentum, continuity, temperature, salinity, and density equations and is closed physically and mathematically using turbulence closure submodels. The horizontal grid consists of unstructured triangular cells and the irregular bottom is presented using generalized terrain-following coordinates. The General Ocean Turbulent Model (GOTM) developed by Burchard's research group in Germany (Burchard, 2002) has been added to FVCOM to provide optional vertical turbulent closure schemes. FVCOM is solved numerically by a second-order accurate discrete flux calculation in the integral form of the governing equations over an unstructured triangular grid. This approach combines the best features of finite-element methods (grid flexibility) and finite-difference methods (numerical efficiency and code simplicity), and provides a much better numerical representation of both local and global momentum, mass, salt, heat, and tracer conservation. The ability of FVCOM to accurately solve scalar conservation equations in addition to the topological flexibility provided by unstructured meshes and the simplicity of the coding structure make FVCOM ideally suited for many coastal and interdisciplinary scientific applications.

For more information about FVCOM, visit the **official FVCOM website**.

Programming language: C/Fortran 90

Brief description: FVCOM is a finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model comprised of unstructured triangular cells.

Open source license: GPL 3.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

FVCOM 4.1

1.8.2 Environment Requirements

Hardware Requirements

Table 1-38 lists the hardware requirements.

Table 1-38 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software requirements

Table 1-39 lists the software requirements.

Table 1-39 Software Requirements

Item	Version	How to Obtain
FVCOM	4.1	http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz
METIS	4.0	Provided by the software.
NetCDF-C	4.7.0	https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0
NetCDF-Fortran	4.4.5	https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5
HDF5	1.10.5	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz
Test case	tst_run.nml	Provided by the software.

OS Requirements

Table 1-40 lists the OS requirements.

Table 1-40 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

1.8.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FVCOM software porting.

Table 1-41 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
4	/path/to/HDF5	Installation path of HDF5	The planned installation paths are only examples. You are advised to deploy the software in shared paths. Replace the paths used in commands in this document with the actual paths planned during the installation process.
5	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
6	/path/to/METIS	Path for storing the downloaded METIS patch	
7	/path/to/FVCOM	Installation path of FVCOM	

1.8.4 Configuring the Compilation Environment

1.8.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-42 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing HDF5	See **1.8.4.2 Installing HDF5**.
3	Installing NetCDF-C	See **1.8.4.3 Installing NetCDF-C**.
4	Installing NetCDF-Fortran	See **1.8.4.4 Installing NetCDF-Fortran**.

1.8.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.5
```

Step 4 Run the following command to perform the configuration:

./configure --prefix=*/path/to/HDF5* **--enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx**

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the HDF5 environment variables:

export LD_LIBRARY_PATH=*/path/to/HDF5***/lib:$LD_LIBRARY_PATH**
export HDF5=*/path/to/HDF5*

----End

1.8.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L$HDF5/lib" CPPFLAGS="-I$HDF5/include" CC=mpicc --disable-dap
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.8.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$HDF5/include -I$NETCDF/include" LDFLAGS="-L$HDF5/lib -L$NETCDF/lib" CC=mpicc FC=mpif90 F77=mpif90
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End

1.8.5 Obtaining the Source Code

Procedure

Step 1 Download the FVCOM installation package **FVCOM-4.1.tar.gz** from the following address:

http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz

Step 2 Use an SFTP tool to upload the FVCOM installation package to the */path/to/FVCOM* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.8.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/FVCOM
```

Step 3 Run the following command to copy the installation package to the main program installation directory:

```
cp fvcom-4.1.tar.gz /path/to/FVCOM
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/FVCOM
```

Step 5 Run the following command to decompress the installation package:

```
tar -xvf fvcom-4.1.tar.gz
```

Step 6 Perform the following operations to configure the files:

1. Copy the **make.inc_example** file as **make.inc**.
   ```
   cp Examples/Estuary/make.inc_example FVCOM_source/make.inc
   ```
2. Create a soft link to the **make.inc** file.
   ```
   ln -sf FVCOM_source/make.inc ./
   ```
3. Edit the **make.inc** file.
   a. Open **make.inc**.
      ```
      vi make.inc
      ```
   b. Press **i** to enter the insert mode and modify the file as follows:

      Line 51:
      ```
      TOPDIR = /path/to/FVCOM/FVCOM4.1/FVCOM_source
      ```

      Lines 79 and 80:
      ```
      LIBDIR = -L$(INSTALLDIR)/lib -L../METIS_source/metis -L./libs/julian
      INCDIR = -I$(INSTALLDIR)/include -I../METIS_source/metis -I./libs/julian
      ```

      Line 97:
      ```
      IOLIBS = -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -lnetcdff -lnetcdf -lhdf5_hl -lhdf5 -lz -lcurl -lm
      ```

      Line 98:
      ```
      IOINCS = -I/path/to/NETCDF/include -I/path/to/HDF5/include
      ```

      Comment out and add the following content as follows:
      ```
      #  Intel/MPI Compiler Definitions (SMAST)
      #------------------------------------------------------------------------
      #         CPP      = /usr/bin/cpp
      #         COMPILER = -DIFORT
      #         CC       = mpicc
      #         CXX      = mpicxx
      #         CFLAGS   = -O3
      #         FC       = mpif90
      #         DEBFLGS  = -check all traceback
      # Use 'OPT = -O0 -g'  for fast compile to test the make
      # Use 'OPT = -xP' for fast run on em64t (Hydra and Guppy)
      # Use 'OPT = -xN' for fast run on ia32 (Salmon and Minke)
      #         OPT      = -O0 –g
      #         OPT      = -axN –xN
      #         OPT      = -O3

      #  gfortran defs
      #------------------------------------------------------------------------
              CPP      = /usr/bin/cpp
      ```

 Copyright © Huawei Technologies Co., Ltd.

```
COMPILER = -DGFORTRAN
CC       = mpicc
CXX      = mpicxx
FC       = mpif90
DEBFLGS  =
OPT      = -O3 -ffixed-line-length-none -ffree-form -ffree-line-length-none
CLIB     =
```

c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to create a compilation environment variable file:

1. Create an environment variable file.
   a. Create an environment variable file.
      vi env-fvcom.sh
   b. Press **i** to enter the insert mode and add the following content:
      ```
      export GCC_HOME=/path/to/GNU
      export PATH=$GCC_HOME/bin:$PATH
      export LD_LIBRARY_PATH=$GCC_HOME/lib:$GCC_HOME/lib64:$LD_LIBRARY_PATH
      export MPI_HOME=/path/to/OPENMPI
      export PATH=$MPI_HOME/bin:$PATH
      export LD_LIBRARY_PATH=$MPI_HOME/lib:$LD_LIBRARY_PATH
      export HDF5=/path/to/HDF5
      export PATH=$HDF5/bin:$PATH
      export LD_LIBRARY_PATH=$HDF5/lib:$LD_LIBRARY_PATH
      export NETCDF=/path/to/NETCDF
      export PATH=$NETCDF/bin:$PATH
      export LD_LIBRARY_PATH=$NETCDF/lib:$LD_LIBRARY_PATH
      ```
   c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
2. Load the environment variables.
   source env-fvcom.sh

Step 8 Run the following command to compile the METIS library:

1. Go to the METIS installation directory.
   cd ./METIS_source
2. Decompress the METIS installation package.
   tar -zxvf metis.tgz
3. Copy the **metis-4.0.patch** file to the current directory:
   cp */path/to/METIS*/**metis-4.0.patch ./**
4. Go to the **metis** directory and execute the patch file.
   ```
   cd metis
   patch -p2 < ../metis-4.0.patch
   ```
5. Create the corresponding directories.
   ```
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/lib
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/include
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/bin
   ```
6. Compile and install the METIS library.
   ```
   make -j
   make install
   ```

Step 9 Run the following commands to compile the julian library:

1. Go to the julian installation directory.
   cd ../../FVCOM_source/libs
2. Decompress the **julian.tgz** installation package.
   tar -zxvf julian.tgz
3. Go to the directory generated after the decompression.
   cd julian
4. Perform the compilation and installation.
   ```
   make -j
   make install
   ```

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Run the following command to compile and install the FVCOM main program:

1. Go to the FVCOM installation directory.
   cd */path/to/FVCOM*/**FVCOM4.1/FVCOM_source**
2. Edit the **mod_newinp.F** file.
   a. Open **mod_newinp.F**.
      vi mod_newinp.F
   b. Press **i** to enter the insert mode and modify the file as follows:
      Add the following content to the first **contains** statement:
      ```
      !---------------------------------------
      ! functions
      !---------------------------------------

      contains
        Character( Len = 256 ) Function N_Fmt( c , n )
          Character( Len = * ) , Intent( IN ) :: c
          Integer , Intent( IN ) :: n
          integer :: i , j
          character( len = 16 ) :: cn
          i = index( c , '<' )
          j = index( c , '>' )
          write( cn , '(g0)' ) n
          N_Fmt = c(:i-1) // Trim(adjustL(cn)) // c(j+1:)
        End Function N_Fmt
      ```
      Line 352:
      Before the modification:
      ```
      write(*,'(A20,<size>F10.4)')trim(argname)//': ',fval(1:SIZE)
      ```
      After the modification:
      ```
      write(*,N_Fmt('(A20,<size>F10.4)',SIZE))trim(argname)//': ',fval(1:SIZE)
      ```
      Line 421:
      Before the modification:
      ```
      write(*,'(A20,<size>I10)')trim(argname)//': ',ival(1:SIZE)
      ```
      After the modification:
      ```
      write(*,N_Fmt('(A20,<size>I10)',SIZE))trim(argname)//': ',ival(1:SIZE)
      ```
      Line 494:
      Before the modification:
      ```
      write(*,'(A20,<size>L10)')trim(argname)//': ',cval(1:SIZE)
      ```
      After the modification:
      ```
      write(*,N_Fmt('(A20,<size>L10)',SIZE))trim(argname)//': ',cval(1:SIZE)
      ```
      Line 567:
      Before the modification:
      ```
      write(*,'(A20,<size>A10)')trim(argname)//': ',sval(1:SIZE)
      ```
      After the modification:
      ```
      write(*,N_Fmt('(A20,<size>A10)',SIZE))trim(argname)//': ',sval(1:SIZE)
      ```
   c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
3. Modify the **mod_scal.F**, **internal_step.F**, **adv_t.F**, and **adv_s.F** files. Modify the statements starting with **IF(BACKWARD_ADVECTION** in the files. Change **==** to **.eqv** and **/=** to **.neqv**.
   ```
   sed -i 's/\/=\.TRUE\./\.neqv\.\.TRUE\./g' mod_scal.F
   sed -i 's/==\.TRUE/\.eqv\.\.TRUE/g' internal_step.F
   sed -i 's/==\.FALSE\./\.eqv\.\.FALSE\./g' adv_t.F
   sed -i 's/==\.FALSE\./\.eqv\.\.FALSE\./g' adv_s.F
   ```
4. Compile and install the main program.
   make

 Copyright © Huawei Technologies Co., Ltd.

After the installation is successful, you can run the **ll** command to view the generated executable file **fvcom** in the current directory.

----End

1.8.7 Running and Verifying FVCOM

1.8.7.1 Running and Verifying the Estuary Case

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create a test directory:

```
mkdir -p /path/to/FVCOM/test
```

Step 3 Switch to the test directory.

```
cd/path/to/FVCOM/test
```

Step 4 Run the following command to copy the test case to the current directory:

```
cp -r /path/to/FVCOM/FVCOM4.1/Examples/Estuary/ ./
```

Step 5 Run the following command to switch to the **run** directory:

```
cd ./Estuary/run
```

Step 6 Run the following command to edit the case input file:

1. Open the test case input file.
   ```
   vi tst_run.nml
   ```
2. Press **i** to enter the insert mode and add single quotation marks to **tst_grd.dat**.
   ```
   GRID_FILE = 'tst_grd.dat'
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to run the test:

```
time mpirun -n 24 --allow-run-as-root --bind-to core /path/to/FVCOM/FVCOM4.1/FVCOM_source/
fvcom --casename=tst
```

Step 8 Run the following command to view the current directory. After the command is executed successfully, the .nc result file is generated.

Figure 1-12 Result example

```
-rw-r--r-- 1 root root 48517284 Feb  5 12:07 tst_0001.nc
-rw------- 1 root root    10578 Feb  5 09:55 tst_run.nml
```

----End

1.8.7.2 Running and Verifying the Inlet Case

Prerequisites

FVCOM requires compiling an executable file based on the case. If the inlet case is used, replace the **make.inc** file with the **make.inc** file of the Intel case (see **1.8.6 Compilation and Installation**). Modify the **mod_cstms_vars.F** and

 Copyright © Huawei Technologies Co., Ltd.

mod_sed_cstms.F files in the */path/to/FVCOM/FVCOM_source* directory. Other steps are the same as those in **1.8.6 Compilation and Installation**.

Step 1 Run the following command to modify the **mod_cstms_vars.F** file.

1. Open **mod_cstms_vars.F**.
   ```
   vi mod_cstms_vars.F
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:
   ```
   494       (/       "bo_sd50"                                &
   529       (/       "mean grain diameter"                    &
   563       character(len=80), dimension(34) :: bot_units = (" m","kg&
   568       character(len=80), dimension(5) :: bed_snames =
   (/"bed_thick","bed_age","bed_por","bed_diff","bed_btcr"/)
   569       character(len=80), dimension(5) :: bed_lnames = (/"bed layer thickness","bed layer age","bed
   layer porosity","bed layer bio-diffusivity","bed critical stress"/)
   570       character(len=80), dimension(5) :: bed_units = (/"m","days","-","-","N/m2"/)
   ```

   After the modification:
   ```
   494       (/character(80):: "bo_sd50"                                &
   529       (/ character(80)::        "mean grain diameter"                    &
   563       character(len=80), dimension(34) :: bot_units = (/character(80)::" m","kg&
   568       character(len=80), dimension(5) :: bed_snames = (/character(80)::"|
   (/"bed_thick","bed_age","bed_por","bed_diff","bed_btcr"/)
   569       character(len=80), dimension(5) :: bed_lnames = (/character(80)::"bed layer thickness","bed
   layer age","bed layer porosity","bed layer bio-diffusivity","bed critical stress"/)
   570       character(len=80), dimension(5) :: bed_units = (/character(80)::"m","days","-","-","N/m2"/)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Modify the **mod_sed_cstms.F** file.

1. Open **mod_sed_cstms.F**.
   ```
   vi mod_sed_cstms.F
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:
   ```
   5797    IF(BACKWARD_ADVECTION==.TRUE.)THEN
   ```

   After the modification:
   ```
   5797    IF(BACKWARD_ADVECTION .EQV. .TRUE.)THEN
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a test directory and go to the test directory:
```
mkdir -p /path/to/FVCOM/test
cd /path/to/FVCOM/test
```

Step 3 Run the following command to copy the test case to the current directory:
```
cp -r /path/to/FVCOM/FVCOM4.1/Examples/cohesive_bed/tidal_inlet/ ./
```

Step 4 Run the following command to switch to the **run** directory:
```
cd ./tidal_inlet/run
```

Step 5 Run the following command to edit the case input file:

1. Open the test case input file.
   ```
   vi inlet_run.nml
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and add single quotation marks to **none**, **inlet_nesting.nc**, and **inlet_nesting.nc**.

```
NCNEST_NODE_FILES_WAVE  = 'none'
NESTING_FILE_NAME       = 'inlet_nesting.nc'
NESTING_FILE_NAME_WAVE  = 'inlet_nesting_wave.nc'
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to run the test:

time mpirun -np 96 --allow-run-as-root --bind-to core */path/to/FVCOM*/**FVCOM4.1/FVCOM_source/fvcom --casename=inlet**

Example:

----End

1.8.8 More Information

Official FVCOM website:

http://fvcom.smast.umassd.edu/

1.9 MITgcm c67o Porting Guide (CentOS 7.6)

1.9.1 Introduction

MIT General Circulation Model (MITgcm) is a numerical model designed for study of the atmosphere, ocean, and climate. MITgcm's flexible non-hydrostatic formulation enables it to efficiently simulate fluid phenomena over a wide range of scales. Its adjoint capabilities enable it to be applied to sensitivity questions and to parameter and state estimation problems. By employing fluid equation isomorphisms, a single dynamical kernel can be used to simulate flow of both the atmosphere and ocean.

For more information about MITgcm, visit the **official MITgcm website**.

Programming language: C/Fortran 90

Brief description: MITgcm is a modeling software for simulating large-scale ocean circulation.

Open source license: GPL 3.0

Recommended Software Version

MITgcm_c67o

 Copyright © Huawei Technologies Co., Ltd.

1.9.2 Environment Requirements

Hardware Requirements

Table 1-43 lists the hardware requirements.

Table 1-43 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-44 lists the software requirements.

Table 1-44 Software requirements

Item	Version	How to Obtain
MITgcm	c67o	http://mitgcm.org/download/
NetCDF-C	4.7.0	https://www.unidata.ucar.edu/downloads/netcdf/
NetCDF-Fortran	4.4.5	https://www.unidata.ucar.edu/downloads/netcdf/
HDF5	1.10.1	http://hdfgroup.org/HDF5/
Test case	exp4	Provided by the software.

OS Requirements

Table 1-45 lists the OS requirements.

Table 1-45 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.9.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MITgcm software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-46 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
4	/path/to/MITGCM	Installation path of MITgcm	

1.9.4 Configuring the Compilation Environment

1.9.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-47 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.9.4.2 Installing HDF5**.
3	Installing NetCDF-C	See **1.9.4.3 Installing NetCDF-C**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Description
4	Installing NetCDF-Fortran	See **1.9.4.4 Installing NetCDF-Fortran**.

1.9.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.9.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile CPPFLAGS="-I/path/to/HDF5/include " LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-L/path/to/HDF5/lib -I/path/to/HDF5/include"
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.9.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

----End

1.9.5 Obtaining the Source Code

Procedure

Step 1 Download the MITgcm installation package **MITgcm_c67o.tar.gz**.

http://mitgcm.org/download/

Step 2 Use an SFTP tool to upload the MITgcm installation package to the **/path/to/MITGCM** directory on the server.

----End

1.9.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a main program installation directory.

```
mkdir -p /path/to/MITGCM
```

Step 3 Copy the installation package to the main program installation directory.

```
cp MITgcm_c67o.tar.gz /path/to/MITGCM
```

Step 4 Go to the main program installation directory.

```
cd /path/to/MITGCM
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Decompress the installation package.

```
tar -xvf MITgcm_c67o.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd MITgcm
```

Step 7 Configure the files.

1. Go to the **build_options** directory.

   ```
   cd tools/build_options
   ```

2. Create and edit the **Linux_arm64_gfortran** file.

   a. Create **Linux_arm64_gfortran**.

   ```
   vi linux_arm64_gfortran
   ```

   b. Press **i** to enter the insert mode and add the following content:

```
#!/bin/bash

MPI='true'
CC=mpicc
FC=mpif77
F90C=mpif90

DEFINES='-DWORDLENGTH=4 -DNML_TERMINATOR -DALLOW_USE_MPI -DALWAYS_USE_MPI'
EXTENDED_SRC_FLAG='-ffixed-line-length-132'
F90FIXEDFORMAT='-ffixed-form'
GET_FC_VERSION="--version"
OMPFLAG='-fopenmp'

NOOPTFLAGS='-O0 -g'
NOOPTFILES=''

CFLAGS='-O3 -march=armv8.2-a'
FFLAGS="$FFLAGS -fconvert=big-endian -fimplicit-none"
CFLAGS="$CFLAGS -fPIC"
FFLAGS="$FFLAGS -fPIC"
#-  with FC 19, need to use this without -fPIC (which cancels -mcmodel option):
# CFLAGS="$CFLAGS -mcmodel=medium"
# FFLAGS="$FFLAGS -mcmodel=medium"
#- might want to use '-fdefault-real-8' for fizhi pkg:
#FFLAGS="$FFLAGS -fdefault-real-8 -fdefault-double-8"

if test "x$IEEE" = x ; then     #- with optimisation:
   #- full optimisation
    FOPTIM='-O3 -funroll-loops -march=armv8.2-a'
    NOOPTFILES="$NOOPTFILES ini_masks_etc.F"
   #- can use -O2 (safe optimisation) to avoid Pb with some gcc version of -O3:
   #FOPTIM='-O2 -funroll-loops'
else
   # these may also be useful, but require specific gfortran versions:
   # -Wnonstd-intrinsics        for gfortran <= 4.3
   # -Wintrinsics-std           for gfortran >= 4.4
   # -Wno-tabs                  for gfortran >= 4.3
   # -Wno-unused-dummy-argument for gfortran >= 4.6
   #FFLAGS="$FFLAGS -Waliasing -Wampersand -Wsurprising -Wline-truncation"
   #- or simply:
    FFLAGS="$FFLAGS -Wall"
   #- to get plenty of warnings: -Wall -Wextra (older form: -Wall -W) or:
   #FFLAGS="$FFLAGS -Wconversion -Wimplicit-interface -Wunused-labels"
  if test "x$DEVEL" = x ; then  #- no optimisation + IEEE :
    FOPTIM='-O0'
  else                          #- development/check options:
    FOPTIM='-O0 -g -fbounds-check'
    FOPTIM="$FOPTIM -ffpe-trap=invalid,zero,overflow -finit-real=inf"
  fi
fi

F90FLAGS=$FFLAGS
```

 Copyright © Huawei Technologies Co., Ltd.

```
F90OPTIM=$FOPTIM

# --------------------------
# NETCDF DIRECTORIES
# --------------------------
HDF5_HOME=/path/to/HDF5
NETCDF_HOME=/path/to/NETCDF

INCLUDEDIRS=''
INCLUDES="-I$NETCDF_HOME/include -I$HDF5_HOME/include"
LIBS="-L$NETCDF_HOME/lib -lnetcdff -lnetcdf -L$HDF5_HOME/lib -lhdf5_hl -lhdf5"

# ----------------------
# INCLUDE MPI DIRECTORIES
# ----------------------
MPI_HOME=/path/to/OPENMPI-4.0.1
MPI_INC_DIR=$MPI_HOME/include
INCLUDES="$INCLUDES -I$MPI_HOME/include"
INCLUDEDIRS="$INCLUDEDIRS $MPI_HOME/include"
MPIINCLUDEDIR="$MPI_INC_DIR"
```

c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation and installation, using exp4 as an example:

1. Go to the corresponding directory.
   ```
   cd /path/to/MITGCM/MITgcm/verification/exp4/code
   ```
2. Rename **SIZE.h_mpi** to **SIZE.h** to replace the original **SIZE.h** file.
   ```
   mv SIZE.h_mpi SIZE.h
   ```
3. Go to the corresponding directory.
   ```
   cd /path/to/MITGCM/MITgcm/verification/exp4/build
   ```
4. Generate a **makefile** file.
   ```
   ../../../tools/genmake2 -mods=../code -mpi -of=../../../tools/build_options/linux_arm64_gfortran
   ```
5. Perform the compilation and installation.
   ```
   make depend
   make -j
   ```

----End

1.9.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/MITGCM/MITgcm/verification/exp4/run
```

Step 3 Copy the test case to the current directory.

```
cp ../input/* ./
cp ../build/mitgcmuv ./
```

Step 4 Start a test.

```
time mpirun -n 4 --allow-run-as-root ./mitgcmuv
```

Step 5 View the **STDOUT.0000** output file.

```
vi STDOUT.0000
```

If the information shown in the figure is displayed, the test is successful:

 Copyright © Huawei Technologies Co., Ltd.

Figure 1-13 An example result

```
(PID.TID 0000.0001) // =======================================================
(PID.TID 0000.0001) // Tile <-> Tile communication statistics
(PID.TID 0000.0001) // =======================================================
(PID.TID 0000.0001) // o Tile number: 000001
(PID.TID 0000.0001) //         No. X exchanges =              0
(PID.TID 0000.0001) //            Max. X spins =              0
(PID.TID 0000.0001) //            Min. X spins =     1000000000
(PID.TID 0000.0001) //          Total. X spins =              0
(PID.TID 0000.0001) //            Avg. X spins =       0.00E+00
(PID.TID 0000.0001) //         No. Y exchanges =              0
(PID.TID 0000.0001) //            Max. Y spins =              0
(PID.TID 0000.0001) //            Min. Y spins =     1000000000
(PID.TID 0000.0001) //          Total. Y spins =              0
(PID.TID 0000.0001) //            Avg. Y spins =       0.00E+00
(PID.TID 0000.0001) // o Thread number: 000001
(PID.TID 0000.0001) //            No. barriers =          35584
(PID.TID 0000.0001) //      Max. barrier spins =              1
(PID.TID 0000.0001) //      Min. barrier spins =              1
(PID.TID 0000.0001) //     Total barrier spins =          35584
(PID.TID 0000.0001) //      Avg. barrier spins =       1.00E+00
PROGRAM MAIN: Execution ended Normally
```

----End

1.9.8 More Resources

Official MITgcm website:

http://mitgcm.org/

MITgcm Installation Guide:

https://mitgcm.readthedocs.io/en/latest/overview/overview.html

1.10 POM POM2K Porting Guide (CentOS 7.6)

1.10.1 Introduction

Princeton Ocean Model (POM) is a three-dimensional shallow sea model of the inclined continental shelf vertically using the sigma coordinate. Based on the original equation set, the POM contains a second-order turbulence sub-model, which can provide the turbulent eddy coefficient that varies with the flow rate. Developed by Blumberg and Mellor in the 1970s, it has been improved with the joint efforts of many scholars. It can be used in estuaries, coastal areas and open oceans.

The main feature of POM is that it uses the sigma coordinate in the vertical direction, so it can be easily introduced into the continental shelf. In determining the vertical turbulence exchange coefficient, the second-order turbulence kinetic energy (TKE) closure scheme is used. Therefore, the vertical difference is divided into implicit ones, and the limitation of time on the vertical coordinate is eliminated, so that the resolution of the upper and lower boundary layers of the ocean is improved and the calculation is stable. To save the calculation time and increase the calculation stability of the model, the POM separates the positive pressure model and the oblique pressure model of the current by using different time steps. The outer model is two-dimensional with short time step. The inner model is three-dimensional with long time step. In the horizontal direction, the

 Copyright © Huawei Technologies Co., Ltd.

POM uses the Arakawa C interlaced grid system. In the vertical direction, the turbulence kinetic energy equation, the vertical diffusion exchange coefficient and the vertical velocity ω are staggered with the temperature, salinity and the current field. The horizontal coordinate system may be optional, and may be a curve orthogonal coordinate or a longitude and latitude coordinate.

For details about AmberTools, visit the **official POM website**.

Programming language: Fortran

Brief description: POM is the Princeton Ocean Model.

Open source license: GPL 3.0

Recommended Software Version

POM POM2K

1.10.2 Environment Requirements

Hardware Requirements

Table 1-48 lists the hardware requirements.

Table 1-48 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-49 lists the software requirements.

Table 1-49 Software requirements

Item	Version	Download Address
POM	POM2K	**http://www.ccpo.odu.edu/POMWEB/userinfo.htm**
Test case	GRID-DATA	**http://www.ccpo.odu.edu/POMWEB/userinfo.htm**

OS Requirements

Table 1-50 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-50 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.10.3 Paths for Software Porting

This chapter lists the software installation paths involved in the POM software porting.

Table 1-51 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/POM	Installation path of POM	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/NETCDF	Installation path of NetCDF-C	

1.10.4 Configuring the Compilation Environment

1.10.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-52 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing NetCDF-C	See **1.10.4.2 Installing NetCDF-C**.

1.10.4.2 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.4.1.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.10.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to download POM:

```
wget -r -p -k -np -nc -e robots=off http://www.ccpo.odu.edu/POMWEB/POM2K/zip -m POM2K.zip -j www.ccpo.odu.edu/POMWEB/POM2K/*
```

Step 3 Run the following command to download the test case **GRID-DATA**:

```
wget -r -p -k -np -nc -e robots=off http://www.ccpo.odu.edu/POMWEB/GRID-DATA/zip -m GRID-DATA.zip -j www.ccpo.odu.edu/POMWEB/GRID-DATA/*
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 After the download is complete, use an SFTP tool to upload **POM2K.zip** to the */**path/to/POM/POM2K*** directory on the server and upload **GRID-DATA.zip** to the */path/to/POM/GRID-DATA* directory on the server.

----End

1.10.6 POM2K Model Running

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the POM installation directory:

```
cd /path/to/POM/POM2K
```

Step 3 Run the following command to decompress the POM2K installation package:

```
unzip POM2K.zip
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd POM2K
```

Step 5 Run the following commands to modify the **pom2k.f** file:

1. Open **pom2k.f**.
   ```
   vi pom2k.f
   ```
2. Press **i** to enter the insert mode and change mode **1** to mode **3**.

   Before the modification:
   ```
   iproblem=1
   ```
   After the modification:
   ```
   iproblem=3
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to copy the header file **gridcom** of the installation package:

```
cp /path/to/POM/GRID-DATA/IC/include/gridcom ./
```

Step 7 Run the following commands to modify the **runpom2k** file:

1. Open **runpom2k**.
   ```
   vi runpom2k
   ```
2. Press **i** to enter the insert mode and change mode **1** to mode **3**.

   Before the modification:
   ```
   #
   # Runscript for pom2k with example of reading IC from file.
   #  IC file generated by /GRID-DATA/GRID.f     (T.E. Dec04)
   #
   # Note: if "grid" not created here you can put it in pom2k.c
   #       "params" replaces default values set in pom2k.f
   #
   #
   # ------------------ PARAMETER FILES FOR EACH CASE -----------
   #
   # -- SEAMOUNT TEST CASE (initial cond. calc. in pom2k.f)
   #
    echo '      parameter(im=65, jm=49, kb=21)' >  grid
    echo '      iproblem=  1                  ' >  params
    echo '      days=      0.025              ' >> params
    echo '      prtd1=     0.0125              ' >> params
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
 echo '      dte=       6.                ' >> params
#
# -- COAST TEST CASE (initial cond. from file)
#
#echo '      parameter(im=41, jm=61, kb=16)' >  grid
#echo '      iproblem=  3                 ' >  params
#echo '      days=      0.50              ' >> params
#echo '      prtd1=     0.25              ' >> params
#echo '      dte=      12.                ' >> params
#cp IC.dat fort.40            # Initial Condition file
#
# ------------------ COMPILE & RUN -----------------------------
#
# g77 -O3 pom2k.f /usr/local/lib/libnetcdf.a
# ---- now netCDF output not used (see "CDF" in pom2k.f)
  gfortran -O3 pom2k.f
# f90 -O3 pom2k.f  # or any other compiler you wish to use
#
# ------------------ OUTPUT FILES -----------------------------
#
a.out > pom2k.out        # printout file
#
#       pom2k.nc         # netCDF file for MATLAB plotting
```

After the modification:

```
#
# Runscript for pom2k with example of reading IC from file.
#  IC file generated by /GRID-DATA/GRID.f     (T.E. Dec04)
#
# Note: if "grid" not created here you can put it in pom2k.c
#       "params" replaces default values set in pom2k.f
#
#
# ------------------ PARAMETER FILES FOR EACH CASE -----------
#
# -- SEAMOUNT TEST CASE (initial cond. calc. in pom2k.f)
#
# echo '      parameter(im=65, jm=49, kb=21)' >  grid
# echo '      iproblem=  1                 ' >  params
# echo '      days=      0.025             ' >> params
# echo '      prtd1=     0.0125            ' >> params
# echo '      dte=       6.                ' >> params
#
# -- COAST TEST CASE (initial cond. from file)
#
echo '      parameter(im=41, jm=61, kb=16)' >  grid
echo '      iproblem=  3                 ' >  params
echo '      days=      0.50              ' >> params
echo '      prtd1=     0.25              ' >> params
echo '      dte=      12.                ' >> params
cp IC.dat fort.40            # Initial Condition file
#
# ------------------ COMPILE & RUN -----------------------------
#
# g77 -O3 pom2k.f /usr/local/lib/libnetcdf.a
# ---- now netCDF output not used (see "CDF" in pom2k.f)
  gfortran -O3 pom2k.f /path/to/NETCDF/netcdf-4.4.1.1/lib/libnetcdf.a -I/path/to/NETCDF/
netcdf-4.4.1.1/include
# f90 -O3 pom2k.f  # or any other compiler you wish to use
#
# ------------------ OUTPUT FILES -----------------------------
#
./a.out > pom2k.out        # printout file
#
#       pom2k.nc         # netCDF file for MATLAB plotting
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following commands to grant the permissions to **runpom2k** and execute it:

 Copyright © Huawei Technologies Co., Ltd.

```
chmod 755 runpom2k
./runpom2k
```

The following is an example of the command output.

Figure 1-14 Result example

NOTE

After the command is executed, the **fort.71** file is generated in the current directory. This file is the required NC file.

----End

1.10.7 More Information

Official POM website:

http://www.ccpo.odu.edu/POMWEB/userinfo.htm

1.11 MOM 6 Porting Guide (CentOS 7.6)

1.11.1 Introduction

The Modular Ocean Model (MOM) is a three-dimensional ocean circulation model used to study the ocean climate system.

For more information about MOM, visit the **official MOM website**.

Programming language: C/Fortran

Brief description: modular ocean model

Open source license: GPL 3.0

Recommended Software Version

MOM 6

1.11.2 Environment Requirements

Hardware Requirements

Table 1-53 lists the hardware requirements.

Table 1-53 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-54 lists the software requirements.

Table 1-54 Software requirements

Item	Version	How to Obtain
MOM	6	**https://github.com/NOAA-GFDL/MOM6-examples.git**
PNETCDF	1.9.0	**https://parallel-netcdf.github.io/wiki/Download.html**
NETCDF-C	4.7.0	**https://www.unidata.ucar.edu/downloads/netcdf/index.jsp**
NETCDF-Fortran	4.4.5	**https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5**
HDF5	1.10.1	**http://hdfgroup.org/HDF5/**
Test case	double_gyre	Provided by the software.

OS Requirements

Table 1-55 lists the OS requirements.

Table 1-55 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.11.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MOM software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-56 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PNETCDF	Installation path of PnetCDF	The installation paths listed in this table are examples only. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the porting process.
3	/path/to/HDF5	Installation path of HDF5	
4	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
5	/path/to/MOM	Installation path of MOM	

1.11.4 Configuring the Compilation Environment

1.11.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-57 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing PnetCDF	See **1.11.4.2 Installing PnetCDF**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing HDF5	See **1.11.4.3 Installing HDF5**.
4	Installing NetCDF-C	See **1.11.4.4 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.11.4.5 Installing NetCDF-Fortran**.

1.11.4.2 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd parallel-netcdf-1.9.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test
```

Step 5 Run the following commands to compile and install PnetCDF:

```
make -j
make install
```

Step 6 Run the following commands to set the PnetCDF environment variables:

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.11.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Switch to the folder generated after decompression.

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to compile and install HDF5:

```
make -j
make install
```

Step 6 Configure HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.11.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to compile and install NetCDF-C:

```
make -j
make install
```

Step 6 Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.11.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to compile and install NetCDF-Fortran:

```
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.11.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the MOM installation directory:

```
cd /path/to/MOM
```

Step 3 After verifying that the server can connect to the Internet, run the following commands to obtain the source code:

```
git clone https://github.com/NOAA-GFDL/MOM6-examples.git MOM6-examples
cd MOM6-examples
git submodule init
git submodule update --recursive
```

Step 4 Run the following commands to update the modules:

```
git submodule update src/FMS
git submodule update --init --recursive src/MOM6
git submodule update src/SIS2
git submodule update tools/matlab/gtools
git submodule update tools/python/MIDAS
cd src/MOM6
git submodule init
git submodule update
```

----End

1.11.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to modify the **linux-gnu.mk** file:

```
cd /path/to/MOM/MOM6-examples/src/mkmf/templates/
sed -ri 's/mpich2/ompi/g' linux-gnu.mk
```

Step 3 Compile **libfms.a**.

```
mkdir -p /path/to/MOM/MOM6-examples/build/gnu/shared/repro/
cd /path/to/MOM/MOM6-examples/build/gnu/shared/repro/
../../../../src/mkmf/bin/list_paths -l ../../../../src/FMS
../../../../src/mkmf/bin/mkmf -t ../../../../src/mkmf/templates/linux-gnu.mk -p libfms.a -c "-Duse_libMPI
-Duse_netCDF -DSPMD" path_names
make NETCDF=3 REPRO=1 libfms.a -j
```

Step 4 Compile **MOM 6**.

```
cd /path/to/MOM/MOM6-examples
mkdir -p build/gnu/ocean_only/repro/
cd build/gnu/ocean_only/repro/
../../../../src/mkmf/bin/list_paths -l ./ ../../../../src/MOM6/{config_src/infra/FMS1,config_src/memory/
dynamic_symmetric,config_src/drivers/solo_driver,config_src/external,src/{*,*/*}}/
../../../../src/mkmf/bin/mkmf -t ../../../../src/mkmf/templates/linux-gnu.mk -o '-I../../shared/repro' -p
'MOM6 -L../../shared/repro -lfms' -c "-Duse_libMPI -Duse_netCDF -DSPMD" path_names
make NETCDF=3 REPRO=1 MOM6 -j
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.11.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to start a benchmark test:

```
cd /path/to/MOM/MOM6-examples/ocean_only/benchmark/
```

Step 3 Run the following commands to modify the case scale:

```
cat << EOF >> MOM_override
#override NIGLOBAL = 720
#override NJGLOBAL = 360
EOF
```

Step 4 Run the following commands to modify the case time:

```
cat << EOF >> input.nml
 &ocean_solo_nml
       months = 0
       days = 20 /
EOF
```

Step 5 Run the following command to create a **RESTART** directory:

```
mkdir RESTART
```

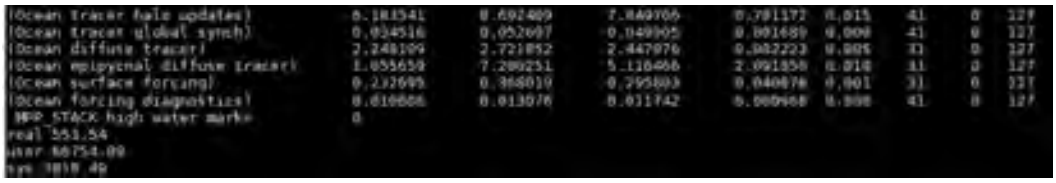
Step 6 Run the following command to start a benchmark test:

```
{ time -p mpirun --allow-run-as-root -np 128 -mca btl ^openib ../../build/gnu/ocean_only/repro/MOM6 > mom.log ;} 2>>mom.log
```

View the value of **real** in the **mom.log** log. A smaller value indicates a higher performance.

Figure 1 shows an example output.

Figure 1-15 An example result



----End

1.11.8 Troubleshooting

An Error Is Reported During libfms.a Compilation

Symptom

When **libfms.a** is compiled, an error message is reported stating "Package mpich2 was not found in the pkg-config search path."

Possible Causes

The **mpich2** parameter in **linux-gnu.mk** is not modified.

Procedure

Check and modify the **linux-gnu.mk** file. For details, see **Step 2**.

An Error Is Reported During libfms.a Compilation

Symptom

When **libfms.a** is compiled, an error message is reported stating "make: nc-config: Command not found."

Possible Causes

The NetCDF environment has errors.

Procedure

Check the NetCDF environment configuration or perform **1.11.4.4 Installing NetCDF-C** again.

1.11.9 More Information

MOM installation guide:

https://github.com/NOAA-GFDL/MOM6-examples/wiki/Getting-started#downloading-the-source-code-and-tools

1.12 SMOKE 4.7 Porting Guide (CentOS 7.6)

1.12.1 Introduction

The Sparse Matrix Operator Kernel Emissions (SMOKE) modeling system is primarily an emissions processing system designed to create gridded, speciated, hourly emissions for input into a variety of air quality models, such as CMAQ, REMSAD, CAMX, and UAM. SMOKE supports area, biogenic, mobile (both onroad and nonroad), and point source emissions processing for criteria, particulate, and toxic pollutants. For biogenic emissions modeling, SMOKE uses the Biogenic Emission Inventory System, version 2.5 (BEIS2) and versions 3.09 and 3.14 (BEIS3). SMOKE is also integrated with the onroad emissions model MOBILE6 and MOVES.

For more information, visit the **official SMOKE website**.

Programming language: Fortran 90

Brief description: source emission tool for air modeling using sparse matrix

Open source license: LGPL 3.0

Recommended Software Version

SMOKE 4.7

1.12.2 Environment Requirements

Hardware Requirements

Table 1-58 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-58 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-59 lists the software requirements.

Table 1-59 Software requirements

Item	Version	How to Obtain
SMOKE	smoke_v47.Linux2_x86_64ifort.tar.gz smoke_v47.nctox.data.tar.gz smoke_install_v47.csh	https://github.com/CEMPD/SMOKE/releases
I/O API	3.2	https://github.com/cjcoats/ioapi-3.2
HDF5	1.10.1	http://hdfgroup.org/HDF5/
PNETCDF	1.8.0	http://cucis.ece.northwestern.edu/projects/PnetCDF
NETCDF	4.4.1.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
NETCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
Test computing instance	$SMK_HOME/data/ge_dat	Provided by the software.

OS Requirements

Table 1-60 lists the OS requirements.

Table 1-60 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

1.12.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SMOKE software porting.

Table 1-61 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation path provided here is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/PNETCDF	Installation path of PnetCDF	
4	/path/to/NETCDF	Installation path of NetCDF	
5	/path/to/NETCDF-FORTRAN	Installation path of NetCDF-Fortran	
6	/path/to/SMOKE	Installation path of SMOKE	

1.12.4 Configuring the Compilation Environment

1.12.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-62 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.12.4.2 Installing HDF5**.
3	Installing PnetCDF	See **1.12.4.3 Installing PnetCDF**.
4	Installing NetCDF	See **1.12.4.4 Installing NetCDF**.
5	Installing NetCDF-Fortran	See **1.12.4.5 Installing NetCDF-Fortran**.

1.12.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar zxvf hdf5-1.10.1.tar.gz
```

Step 3 Switch to the directory generated after the HDF5 installation package is decompressed.

```
cd hdf5-1.10.1
```

Step 4 Set environment variables before compilation.

```
export MPICC=/path/to/OPENMPI/bin/mpicc
export MPICXX=/path/to/OPENMPI/bin/mpicxx
export MPIFC=/path/to/OPENMPI/bin/mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
export F77=$MPIFC
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared --enable-fortran20
```

Step 6 Run the following commands to compile and install HDF5:

```
make -j 16
make install
```

Step 7 Load environment variables.

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.12.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar -xvf parallel-netcdf-1.8.0.tar.bz2
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd parallel-netcdf-1.8.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC"
```

Step 5 Run the following commands to compile and install PnetCDF.

```
make -j 16
make install
```

Step 6 Run the following commands to load the environment variables:

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.12.4.4 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF installation package:

```
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-4.4.1.1
```

Step 4 Set environment variables before compilation.

```
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib - I/path/to/HDF5/include"
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile
```

Step 6 Run the following commands to compile and install NetCDF:

```
make -j 16
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.12.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-fortran-4.4.1
```

Step 4 Run the following commands to set environment variables before compilation:

```
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static
```

Step 6 Run the following commands to compile and install NetCDF-Fortran:

```
make -j 16
make install
```

Step 7 Run the following command to release the environment variables:

```
unset CC CXX FC F77 CPPFLAGS LDFLAGS CFLAGS CXXFLAGS FCFLAGS
```

----End

1.12.5 Obtaining the Source Code

Procedure

Step 1 Download the SMOKE installation package.

https://github.com/CEMPD/SMOKE/releases

Step 2 Use an SFTP tool to upload the SMOKE installation package to the */path/to/SMOKE* directory on the server.

----End

1.12.6 Compiling and Installing SMOKE

 Copyright © Huawei Technologies Co., Ltd.

1.12.6.1 Installing the I/O API

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to run SMOKE in **csh**:

```
chsh -s /bin/csh
```

Step 3 Run the following command to reboot the system for the installation to take effect:

```
reboot
```

Step 4 Run the following commands to set the environment variables:

```
echo "setenv SMK_HOME /path/to/SMOKE" >> /root/.tcsh
echo "setenv BIN Linux2_aarch64gfort" >> /root/.tcsh
echo "setenv LD_LIBRARY_PATH /path/to/GNU/gcc9.1/lib64: /path/to/OPENMPI/openmpi-4.0.1-gcc9.1/
lib" >> /root/.tcsh
echo setenv PATH /path/to/OPENMPI/openmpi-4.0.1- gcc9.1/bin:/path/to/GNU/gcc9.1/bin:$PATH>> /
root/.tcsh
echo setenv LD_LIBRARY_PATH /path/to/NETCDF/lib:$LD_LIBRARY_PATH >> /root/.tcsh
```

Step 5 Run the following command to make the environment variables take effect:

```
source /root/.tcsh
```

Step 6 Run the following commands to copy the installation packages to the installation paths:

```
cp smoke_v47.Linux2_x86_64ifort.tar.gz /path/to/SMOKE
cp smoke_v47.nctox.data.tar.gz /path/to/SMOKE
cp smoke_install.csh /path/to/SMOKE
cp ioapi-3.2.tar.gz /path/to/SMOKE
cd /path/to/SMOKE
source smoke_install.csh
```

Step 7 Run the following command to create the directory:

```
mkdir -p $SMK_HOME/subsys/ioapi
mkdir -p $SMK_HOME/subsys/ioapi/$BIN
```

Step 8 Run the following command to decompress the **ioapi-3.2.tar.gz** installation package:

```
cp ioapi-3.2.tar.gz ./subsys/ioapi/
cd ./subsys/ioapi
tar -xvf ioapi-3.2.tar.gz
```

Step 9 Run the following command to create a **Makefile** file.

```
cd $SMK_HOME/subsys/ioapi/ioapi
cp Makefile.nocpl Makefile
```

Step 10 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   BASEDIR = ${SMK_HOME}/subsys/ioapi
   INSTDIR = ${BASEDIR}/${BIN}
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Run the following command to modify the **Makeinclude.Linux2_x86_64gfort** file:

1. Open **Makeinclude.Linux2_x86_64gfort**.
   ```
   vi Makeinclude.Linux2_x86_64gfort
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   MFLAGS = -ffast-math -funroll-loops -march=armv8-a
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 12 Run the following command to perform the compilation:

```
make
```

Step 13 Run the following commands to set soft links:

```
cd ../$BIN
ln -sf /path/to/NETCDF/lib/libnetcdf.so ./
ln -sf /path/to/NETCDF/lib/libnetcdff.so ./
```

Step 14 Run the following command to switch to the **m3tools** directory:

```
cd ../m3tools
```

Step 15 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   cp Makefile.nocpl Makefile
   ```
   Run the **vi Makefile** command.
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   BASEDIR = ${SMK_HOME}/subsys/ioapi
   INSTDIR = ${BASEDIR}/${BIN}
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 16 Run the following command to perform the compilation:

```
make
```

----End

1.12.6.2 Installing SMOKE

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the directory:

```
cd $SMK_HOME/subsys/smoke/assigns/
```

Step 3 Run the following command to modify the **ASSIGNS.nctox.cmaq.cb05_soa.us12-nc** file:

1. Open the **ASSIGNS.nctox.cmaq.cb05_soa.us12-nc** file.
   ```
   vi ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
   ```
2. Press **i** to enter the insert mode and modify line 25 as follows:
   ```
   setenv BIN    Linux2_aarch64gfort
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to modify the **Makeinclude** file:

1. Open **Makeinclude**.
   ```
   cd $SMK_HOME/subsys/smoke/src
   vi Makeinclude
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   INSTDIR = ${OBJDIR}/${BIN}
   #EFLAG = -extend-source 132 -zero
   EFLAG = -ffixed-line-length-132  -fno-backslash
   IOLIB = -L$(IOBIN) -lioapi -lnetcdf -lnetcdff
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to modify the **biog/normbeis361.f** file:

1. Open **biog/normbeis361.f**.
   ```
   vi biog/normbeis361.f
   ```
2. Press **i** to enter the insert mode and change the types of the IS_AG and IS_TAG functions in line 161 from **INTEGER** to **LOGICAL**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to modify the **smkinven/rdinvsrcs.f** file:

1. Open **smkinven/rdinvsrcs.f**.
   ```
   vi smkinven/ rdinvsrcs.f
   ```
2. Press **i** to enter the insert mode and change the type of the GETPID function in line 86 from **EXTERNAL** to **INTRINSIC**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to modify the **emutil/gentpro.f** file:

1. Open **emutil/gentpro.f**.
   ```
   vi emutil/gentpro.f
   ```
2. Press **i** to enter the insert mode and change the code in lines 1663 and 1695 as follows:
   ```
   1663  (( PROF_MON( S,NP )), NP = 1,12 )
   1695  ( ( PROF_DAY( S,NP ) ), NP = 1,31 )
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to create a directory:

```
mkdir ${SMK_HOME}/subsys/smoke/${BIN}
```

Step 9 Run the following commands to perform the compilation:

```
source /path/to/SMOKE/subsys/smoke/assigns/ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
make
```

----End

1.12.7 Running and Verifying SMOKE

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to perform parallel computing:

```
cd $SMK_HOME/subsys/smoke/script/run/
```

Step 3 Run the following command to test the execution time of each script:

```
time./Script name
```

Example: **time./ smk_point_nctox.csh**

The following figure shows an example of the test result.

Figure 1-16 Example test result

script	time/s
smk_point_nctox.csh	35.489
smk_area_nctox.csh	10.859
smk_rateperhour_nctox.csh	16.874
smk_rateperdistance_nctox.csh	34.97

----End

1.12.8 Troubleshooting

An Error Is Reported When SMOKE Is Run

Symptom

Possible causes

The code space of fitrxref.f:192 is too long.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory in which the **Makefileinclude** file is stored:

```
cd $SMK_HOME/subsys/smoke/src
```

Step 3 Modify the **Makefileinclude** file.

1. Open **Makefileinclude**.
   ```
   vi Makefileinclude
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   EFLAG = -ffixed-line-length-132 -fno-backslash
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

1.12.9 More Information

Official installation guide:

https://www.cmascenter.org

1.13 SPECFEM3D GLOBE 7.0.0 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

1.13.1 Introduction

SPECFEM3D GLOBE simulates global and regional (continental scale) seismic wave propagation. Additional new model routines are provided for the Comprehensive Earth Model (CEM) project, generic point-profile models (PPM) and Gauss-Lobatto-Legendre based models (GLL), with complementary tools for postprocessing adjoint sensitivity kernels and gradient-based model updates. The structure of the software has been simplified to facilitate easier implementation of new 3D models. The code accommodates general moment tensor files, and provides complete information in the SAC headers, as explained in detail in the updated user manual. Loop-vectorization helps reduce the total number of memory accesses performed in each spectral element and improve code vectorization, enhancing numerical performance of the version.

For more information about SPECFEM3D GLOBE, visit the **official SPECFEM3D GLOBE website**.

Programming language: C/Fortran

Brief description: SPECFEM3D GLOBE simulates global and regional (continental scale) seismic wave propagation.

Open source license: GNU Public License

Recommended Software Version

SPECFEM3D GLOBE 7.0.0

1.13.2 Environment Requirements

Hardware Requirements

Table 1-63 lists the hardware requirements.

Table 1-63 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-64 lists the software requirements.

Table 1-64 Software requirements

Item	Version	How to Obtain
SPECFEM3D GLOBE	7.0.0	**https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 1-65 lists the OS requirements.

Table 1-65 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.13.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SPECFEM3D GLOBE software porting.

Table 1-66 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/SPECFEM3D_GLOBE	Installation path of SPECFEM3D GLOBE	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

1.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-67 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.13.5 Obtaining the Source Code

Procedure

Step 1 Download the SPECFEM3D GLOBE installation package **v7.0.0.tar.gz**.

https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz

Step 2 Use an SFTP tool to upload the SPECFEM3D GLOBE installation package to the */path/to/SPECFEM3D_GLOBE* directory on the server.

----End

1.13.6 Compiling and Installing SPECFEM3D GLOBE

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/SPECFEM3D_GLOBE
```

Step 3 Run the following commands to decompress the source code package and go to the source code directory:

```
cd /path/to/SPECFEM3D_GLOBE
tar -xvf v7.0.0.tar.gz
cd specfem3d_globe-7.0.0
```

Step 4 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib: $LD_LIBRARY_PATH
```

Step 5 Run the following command to generate a configuration file:

```
./configure --enable-openmp FC=mpif90 MPIFC=mpif90 CC=mpicc MPICC=mpicc
```

Step 6 Run the following command to modify the **Par_file** file:

1. Open **Par_file**.
   ```
   vi DATA/Par_file
   ```
2. Press **i** to enter the insert mode and modify lines 23 and 24 of the file.

   Before the modification:

 Copyright © Huawei Technologies Co., Ltd.

```
NPROC_XI                = 2
NPROC_ETA               = 2
```

After the modification:

```
NPROC_XI                = 8
NPROC_ETA               = 8
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to perform compilation.

```
make -j
```

----End

1.13.7 Running and Verifying SPECFEM3D GLOBE

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Switch to the test directory.

```
cd /path/to/SPECFEM3D_GLOBE/specfem3d_globe-7.0.0
```

Step 3 Run the following commands to perform the test:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib: $LD_LIBRARY_PATH
mpirun  --allow-run-as-root -np 64 --mca btl ^openib ./bin/xmeshfem3D
mpirun  --allow-run-as-root -np 64 --mca btl ^openib ./bin/xspecfem3D
less OUTPUT_FILES/output_solver.txt
```

If the information shown in the following figure is displayed, the test is successful.

Figure 1-17 Result example

```
Time-Loop Complete. Timing info:
Total elapsed time in seconds =    40.010946439999998
Total elapsed time in hh:mm:ss =     0 h 00 m 40 s
```

----End

1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6)

1.14.1 Introduction

The Community Multiscale Air Quality (CMAQ) modeling system is at the heart of the third-generation air quality modeling system (Models-3) developed by the U.S. Environmental Protections Agency (USEPA). It is a three-dimensional Euler grid-based atmospheric chemistry and transmission simulation system to provide sound estimates of ozone, particulates, toxics, and acid deposition. The CMAQ model comprehensively represents air quality problems at different spatial scales and is widely used to address environmental issues and to conduct atmospheric research.

For more information, visit the **official CMAQ website**.

Programming language: Fortran

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a three-dimensional Euler grid-based atmospheric chemistry and transmission simulation system

Recommended Software Version

CMAQ 5.3.1

1.14.2 Environment Requirements

Hardware Requirements

Table 1-68 lists the hardware requirements.

Table 1-68 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-69 lists the software requirements.

Table 1-69 Software requirements

Item	Version	How to Obtain
CMAQ	5.3.1	https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019
GNU	9.3.0	https://codeload.github.com/gcc-mirror/gcc/tar.gz/releases/gcc-9.3.0
HPC-X	2.6	https://www.mellanox.com/page/hpcx_eula?mtag=hpc-x&mrequest=downloads&mtype=hpc&mver=hpc-x&mname=v2.6/hpcx-v2.6.0-gcc-OFED-4.17-redhat7.6-aarch64.tbz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
PNETCDF	1.9.0	https://parallel-netcdf.github.io/wiki/Download.html
NETCDF-C	4.7.0	https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0
NETCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5
OPTIMIZED-ROUTINES	V20.02	https://codeload.github.com/ARM-software/optimized-routines/tar.gz/v20.02

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
ioapi	3.2	https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111
Test case	2016_12SE1	https://dataverse.unc.edu/dataset.xhtml?persistentId=doi:10.15139/S3/IQVABD

OS Requirements

Table 1-70 lists the OS requirements.

Table 1-70 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.14.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CMAQ software porting.

Table 1-71 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
4	/path/to/MATH	Installation path of OPTIMIZED-ROUTINES	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	/path/to/CMAQ	Installation path of CMAQ	

1.14.4 Configuring the Compilation Environment

1.14.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-72 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.14.4.2 Installing HDF5**.
3	Installing NetCDF-C	See **1.14.4.3 Installing NetCDF-C**.
4	Installing NetCDF-Fortran	See **1.14.4.4 Installing NetCDF-Fortran**.
5.	Installing OPTIMIZED-ROUTINES	See **1.14.4.5 Installing OPTIMIZED-ROUTINES**.

1.14.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following command to set the HDF5 environment variable:

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.14.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.14.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-fortran-4.4.5
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End

1.14.4.5 Installing OPTIMIZED-ROUTINES

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the OPTIMIZED-ROUTINES installation package:

```
tar -xvf optimized-routines-20.02.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd optimized-routines-20.02
```

Step 4 Run the following command to create the MATH directory:

```
mkdir -p /path/to/MATH
```

Step 5 Run the following command to copy the configuration file:

```
cp config.mk.dist config.mk
```

Step 6 Run the following command to perform compilation and installation:

```
make -j16
```

Step 7 Run the following command to copy the related library file:

```
cp -r ./build/* /path/to/MATH
```

Step 8 Run the following commands to add MATH to environment variables:

```
export PATH=/path/to/MATH/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MATH/lib:$LD_LIBRARY_PATH
```

----End

1.14.5 Obtaining the Source Code

Procedure

Step 1 Download the CMAQ dependency **ioapi-3.2-2020111.tar.gz**

https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111

Step 2 Download the CMAQ installation package **CMAQ-CMAQv5.3.1_19Dec2019.tar.gz**

https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019

Step 3 Use an SFTP tool to upload the CMAQ installation and dependency packages to the */path/to/CMAQ* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.14.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the **CMAQ** directory:

```
cd /path/to/CMAQ
```

Step 3 Run the following commands to decompress the package and rename ioapi:

```
tar -xvf ioapi-3.2-2020111.tar.gz
mv ioapi-3.2-2020111 ioapi-3.2
```

Step 4 Run the following commands to copy the configuration file:

```
cd ioapi-3.2
cp ioapi/Makeinclude.Linux2_ia64gfort ioapi/Makeinclude.Linux4_aarch64
```

Step 5 Edit the configuration file.

1. Run the following command to modify the **Makeinclude.Linux4_aarch64** configuration file:
   ```
   vim ioapi/Makeinclude.Linux4_aarch64
   ```
2. Press **i** to enter the insert mode.

   Modify the compiler options:
   ```
   CC   = mpicc
   CXX  = mpicxx
   FC   = mpifort
   ```
   Comment out the line corresponding to the FSFLAGS keyword, for example:
   ```
   #FSFLAGS = -save
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to copy **Makefile** and configure **HOME**:

```
cp ioapi/Makefile.nocpl ioapi/Makefile
export HOME=/path/to/CMAQ
```

Step 7 Run the following command to copy the configuration file:

```
cp m3tools/Makefile.nocpl m3tools/Makefile
```

Step 8 Edit the configuration file.

1. Run the following command to modify the **Makefile** configuration file:
   ```
   vim m3tools/Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   LIBS = -L${OBJDIR} -lioapi -L/path/to/NETCDF/lib -lnetcdff –lnetcdf -L/path/to /HDF5/lib -lhdf5_hl -lhdf5 -lz $(OMPLIBS) $(ARCHLIB) $(ARCHLIBS)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following command to copy the configuration file:

```
cp Makefile.template Makefile
```

Step 10 Edit the configuration file.

1. Open the configuration file.
   ```
   vim Makefile
   ```
2. Press **i** to enter the insert mode.

   Modify the following content and delete the comment tag:
   ```
   BIN       = Linux4_aarch64
   BASEDIR    = ${PWD}
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
INSTALL    = ${HOME}
LIBINST    = $(INSTALL)/$(BIN)
BININST    = $(INSTALL)/$(BIN)
CPLMODE    = nocpl
IOAPIDEFS  = "-DIOAPI_NCF4"
```

Modify the NCFLIBS item:

```
NCFLIBS    = -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Run the following command to compile ioapi:

```
make BIN=Linux4_aarch64
```

Step 12 Modify the **STATE3.EXT** file.

1. Open **STATE3.EXT**.

   ```
   vim ioapi/STATE3.EXT
   ```

2. Press **i** to enter the insert mode. Delete **&** at the end of some lines in the **STATE3.EXT** file, as shown in the red box in the following figure:

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 13 Run the following commands to decompress the package and go to the directory:

```
tar -xvf CMAQ-CMAQv5.3.1_19Dec2019.tar.gz
cd CMAQ-CMAQv5.3.1_19Dec2019
```

Step 14 Edit the configuration file.

1. Open the configuration file.

   ```
   vim bldit_project.csh
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   ```
   set CMAQ_HOME = /path/to/CMAQ/CMAQ_Project
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 15 Run the following command to create the files required for initialization:

```
./bldit_project.csh
```

Step 16 Run the following command to switch to the working directory:

```
cd ../CMAQ_Project/
```

Step 17 Edit the configuration file.

1. Open the configuration file.

   ```
   vim config_cmaq.csh
   ```

2. Press **i** to enter the insert mode.

In the **case gcc** area, modify the paths to dependencies, as shown in the following figure:

```
setenv IOAPI_MOD_DIR   /path/to/CMAQ/ioapi-3.2/Linux4_aarch64/
setenv IOAPI_INCL_DIR  /path/to/CMAQ/ioapi-3.2/ioapi/
setenv IOAPI_LIB_DIR   /path/to/CMAQ/ioapi-3.2/Linux4_aarch64/
setenv NETCDF_LIB_DIR  /path/to/NETCDF/lib/
setenv NETCDF_INCL_DIR /path/to/NETCDF/include/
setenv MPI_LIB_DIR     /path/to/OPENMPI/
```

Modify the compiler parameters. For example:

```
setenv myCC mpicc
setenv myLINK_FLAG  "-fopenmp"
setenv mpi_lib "-lmpi"
```

Add the **openmp** attribute to the **netcdf_lib** variable. For example:

```
setenv netcdf_lib "-lnetcdf -lnetcdff -lgomp"  #> -lnetcdff -lnetcdf for netCDF v4.2.0 and later
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 18 Run the following command to connect to various dependent libraries:

```
./config_cmaq.csh gcc 9.3.0
```

Step 19 Run the following commands in sequence to go to the compilation directory and compile the main program:

```
cd CCTM/scripts/
./bldit_cctm.csh gcc 9.3.0
```

After the compilation is complete, the executable program **CCTM_v531.exe** is generated in the **BLD_CCTM_v531_gcc9.3.0** directory.

----End

1.14.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the case files:

```
tar -xvf CMAQv5.3.1_Benchmark_2Day_Input.tar.gz -C /path/to/CMAQ/CMAQ_Project/data
```

Step 3 Run the following command to switch to the test directory:

```
cd /path/to/CMAQ/CMAQ_Project/CCTM/scripts
```

Step 4 Configure the running script.

1. Open the running script.

```
vim run_cctm_Bench_2016_12SE1.csh
```

2. Press **i** to enter the insert mode.

Change the value of **compiler** to **gcc** and the value of **compilerVrsn** to **9.3.0**:

```
setenv compiler gcc
setenv compilerVrsn 9.3.0
```

Change the values of **NPCOL** and **NPROW**. The product of the two values is equal to the number of parallel cores. For example:

```
@ NPCOL  =  12; @ NPROW =  8
```

Modify the INPDIR path, for example:

```
setenv INPDIR /path/to/CMAQ/CMAQ_Project/data/CMAQv5.3.1_Benchmark_2Day_Input/2016_12SE1
```

Modify the Open MPI running parameters. For example:

```
/usr/bin/time -p mpirun -np $NPROCS -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so --allow-run-as-root --mca btl ^openib $BLD/$EXEC
```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

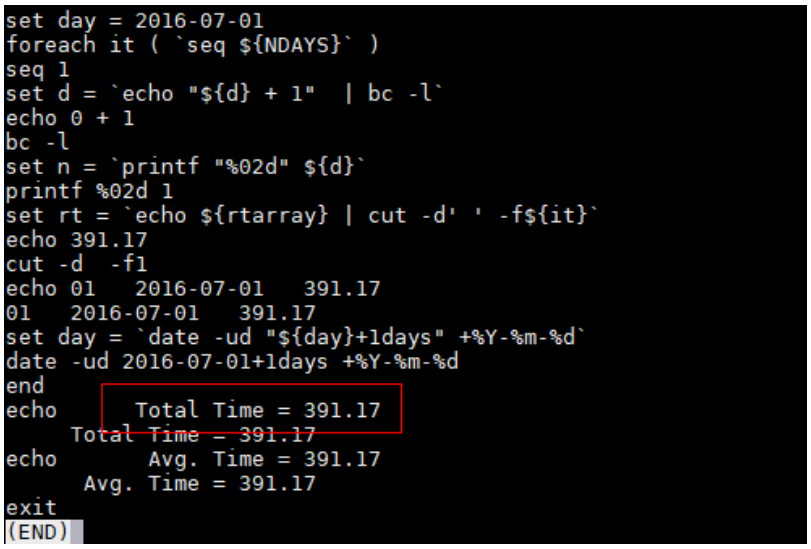
Step 5 Run the following command to perform the test:

```
csh -x /path/to/CMAQ/CMAQ_Project/CCTM/scripts/run_cctm_Bench_2016_12SE1.csh 2>&1 | tee -a /path/to/CMAQ/CMAQ_Project/CCTM/scripts/csh.log
```

Check the value of **Total Time** in the **csh.log** file. The unit is second. A smaller value indicates better performance.

Figure 1-18 shows an example of the running result.

Figure 1-18 Example result



----End

1.14.8 Troubleshooting

Problem 1: An error is reported during CMAQ compilation.

Symptom

An error "Error:Syntax in COMMON statement at (1) STATE3.EXT:174:27:" is reported during CMAQ compilation.

Possible Causes

The data format is incorrect.

Procedure

Modify the **STATE3.EXT** file by referring to **Step 12**.

1.15 SWAN 41.31 Porting Guide (CentOS 7.6)

1.15.1 Introduction

SWAN is a third-generation wave model for obtaining realistic estimates of wave parameters in coastal areas, lakes and estuaries from given wind, bottom and

 Copyright © Huawei Technologies Co., Ltd.

current conditions. However, SWAN can be used on any scale relevant for wind-generated surface gravity waves. The model is based on the wave action balance equation with sources and sinks.

For more information about SWAN, visit the **official SWAN website**.

Programming language: Fortran

Brief description: a wave model

Recommended Software Version

SWAN 41.31

1.15.2 Environment Requirements

Hardware Requirements

Table 1-73 lists the hardware requirements.

Table 1-73 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-74 lists the software requirements.

Table 1-74 Software requirements

Item	Version	How to Obtain
GNU	9.1.0	**https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/**
GMP	6.1.0	**http://gcc.gnu.org/pub/gcc/infrastructure/**
MPC	3.1.4	**http://gcc.gnu.org/pub/gcc/infrastructure/**
MPFR	1.0.3	**http://gcc.gnu.org/pub/gcc/infrastructure/**
OpenMPI	4.0.1	**https://www.open-mpi.org/software/ompi/v4.0/**
HDF5	1.10.5	**https://portal.hdfgroup.org/display/support/HDF5+1.10.5#files**
PNETCDF	1.11.2	**http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz**
NETCDF-C	4.7.0	**https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
NETCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5
SWAN	41.31	http://swanmodel.sourceforge.net/download/zip/swan4131.tar.gz
Test case (**refrac.tar.gz**)	-	http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz

OS Requirements

Table 1-75 lists the OS requirements.

Table 1-75 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

1.15.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SWAN software porting.

Table 1-76 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/PNETCDF	Installation path of PnetCDF	
4	/path/to/NETCDF	Installation path of NetCDF-C	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
		Installation path of NetCDF-Fortran	
5	/path/to/SWAN	Installation path of SWAN	

1.15.4 Configuring the Compilation Environment

1.15.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-77 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.15.4.2 Installing HDF5**.
3	Installing PnetCDF	See **1.15.4.3 Installing PnetCDF**.
4	Installing NetCDF-C	See **1.15.4.4 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.15.4.5 Installing NetCDF-Fortran**.

1.15.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following commands to load the compiler and Open MPI:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export  LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to decompress the HDF5 installation package:

```
tar zxvf hdf5-1.10.5.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

Step 5 Run the following commands to perform configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
./configure --prefix=$HDF5 --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=
$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following commands to load environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.15.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export  LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to decompress the PnetCDF installation package:

```
tar zxvf pnetcdf-1.11.2.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd pnetcdf-1.11.2
```

Step 5 Run the following command to perform the configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
./configure --prefix=$PNETCDF CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC"
FFLAGS="-fPIC" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following commands to load environment variables:

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.15.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependency:

```
yum install curl libcurl libcurl-devel -y
```

Step 3 Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export  LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following commands to decompress the NetCDF-C installation package:

```
tar zxvf netcdf-c-4.7.0.tar.gz
yum install curl.aarch64 libcurl.aarch64 libcurl-devel.aarch64 -y
```

Step 5 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-c-4.7.0
```

Step 6 Run the following command to perform configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
./configure --prefix=$NETCDF --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CPPFLAGS=" -I${HDF5}/include -I${PNETCDF}/include" LDFLAGS=" -L${HDF5}/lib -L${PNETCDF}/lib -Wl,-rpath=${HDF5}/lib -Wl,-rpath=${PNETCDF}/lib " CFLAGS=" -L${HDF5}/lib -L${PNETCDF}/lib -I${HDF5}/include -I${PNETCDF}/include " CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 7 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

Step 8 Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.15.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export  LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-fortran-4.4.5
```

Step 5 Run the following commands to perform configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
./configure --prefix=$NETCDF --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS=" -I${HDF5}/include -I${NETCDF}/include" LDFLAGS=" -L${HDF5}/lib -L${NETCDF}/lib -Wl,-rpath=${HDF5}/lib -Wl,-rpath=${NETCDF}/lib" CFLAGS="-L${HDF5}/lib -L${NETCDF}/lib -I${HDF5}/include -I${NETCDF}/include" CXXFLAGS=" -L${HDF5}/lib -L${NETCDF}/lib -I${HDF5}/include -I${NETCDF}/include" FCFLAGS=" -L${HDF5}/lib -L${NETCDF}/lib -I${HDF5}/include -I${NETCDF}/include" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.15.5 Obtaining the Source Code

Procedure

Step 1 Download the SWAN installation package **swan4131.tar.gz**.

http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz

Step 2 Use an SFTP tool to upload the SWAN installation package to the */path/to/SWAN* directory on the server.

----End

1.15.6 Compiling and Installing SWAN

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to go to the installation directory and decompress the software package:

```
cd /path/to/SWAN
tar -zxvf swan4131.tar.gz
cd swan4131
```

Step 3 Run the following commands to modify the **platform.pl** file:

1. Open **platform.pl**.
   ```
   vim platform.pl
   ```
2. Press **i** to go to the insert mode.

   Modify line 406:
   ```
   print OUTFILE "F90_MPI = mpifort\n";
   ```

Modify line 414:

```
print OUTFILE "NETCDFROOT =/path/to/NETCDF\n";
```

Modify line 788 to **$compiler = "gfortran";**.

```
782 sub getcmpl {
783
784    my $compiler = $ENV{'FC'};
785
786    unless ( $compiler ) {
787       foreach ('ifort','gfortran','f90','ifc','efc','pgf90','xlf90', 'lf95','g95') {
788          $compiler = "gfortran";
789          my $path  = `which $compiler`;
790          last if $path;
791       }
792    }
793
794    return $compiler;
795 }
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to generate the configuration file:

```
make config
```

Step 5 Run the following command to generate the **swan.exe** file:

```
make mpi
```

----End

1.15.7 Running and Verifying SWAN

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a test case directory and upload the test case package to the directory:

```
mkdir -p /path/to/SWAN/swan-testcse
cd /path/to/SWAN/swan-testcse
tar -zxvf refrac.tar.gz
```

Step 3 Run the following commands to go to the directory and create a soft link:

```
cd refract
ln -s /path/to/SWAN/swan4131/swan.exe ./
```

Step 4 Run the following commands to run the case:

```
cp a11refr.swn INPUT
time mpirun --allow-run-as-root -np 40 ./swan.exe
```

Figure 1-19 shows an example of the CPU running time.

Figure 1-19 Result example

```
 SWAN is preparing computation

 iteration    1; sweep 1
+iteration    1; sweep 2
+iteration    1; sweep 3
+iteration    1; sweep 4
 not possible to compute, first iteration

 iteration    2; sweep 1
+iteration    2; sweep 2
+iteration    2; sweep 3
+iteration    2; sweep 4
 accuracy OK in  98.99 % of wet grid points ( 99.50 % required)

 iteration    3; sweep 1
+iteration    3; sweep 2
+iteration    3; sweep 3
+iteration    3; sweep 4
 accuracy OK in   1.02 % of wet grid points ( 99.50 % required)

 iteration    4; sweep 1
+iteration    4; sweep 2
+iteration    4; sweep 3
+iteration    4; sweep 4
 accuracy OK in 100.00 % of wet grid points ( 99.50 % required)

+SWAN is processing output request    1
+SWAN is processing output request    2
+SWAN is processing output request    3
+SWAN is processing output request    4

real    0m2.371s
user    0m47.113s
sys     0m23.792s
```

----End

1.15.8 More Information

Wikipedia:

https://www.myroms.org/wiki/Getting_Started

1.16 CDO 1.9.8 Porting Guide (CentOS 7.6)

1.16.1 Introduction

The Climate Data Operators (CDO) software is a collection of many operators for standard processing of climate and numerical weather prediction (NWP) model output. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. CDO was developed to have the same set of processing functions for GRIB and NetCDF datasets in one package.

Programming language: C++

Brief description: meteorological data processing

Open source license: GPL

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

CDO 1.9.8

1.16.2 Environment Requirements

Hardware Requirements

Table 1-78 lists the hardware requirements.

Table 1-78 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 1-79 Software requirements

Item	Version	Download Link
CDO	1.9.8	https://code.mpimet.mpg.de/attachments/download/20826/cdo-1.9.8.tar.gz
GNU	9.3.0	https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/
Open MPI	4.0.3	https://www.open-mpi.org/software/ompi/v4.0/
HDF5	1.8.12-11	Provided by the Huawei Yum source
GRAB_API	1.12.3-5	Provided by the Huawei Yum source
NetCDF	4.3.3.1-5	Provided by the Huawei Yum source

OS Requirements

Table 1-80 lists the OS requirements.

Table 1-80 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

1.16.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CDO software porting.

Table 1-81 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CDO	Installation path of CDO	This installation path is only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

1.16.4 Configuring the Compilation Environment

1.16.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-82 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Configuring the Huawei Yum source	See **1.16.4.3 Configuring the Huawei Yum Source**.
3	Installing the dependencies	See **1.16.4.4 Installing the Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

1.16.4.2 Setting Up the Basic Environment

Install GNU and Open MPI. For details, see "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.16.4.3 Configuring the Huawei Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Back up the Base source file.

```
cp -a /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak
```

Step 3 Download the new **CentOS-Base.repo** file to the **/etc/yum.repos.d/** directory.

```
wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-AltArch-7.repo
```

Step 4 Clear the original Yum cache.

```
yum clean all
```

Step 5 Refresh the Yum source cache.

```
yum makecache
yum repolist all
```

Step 6 Back up the EPEL source file.

```
cp -a /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel.repo.backup
```

Step 7 Modify the **epel.repo** file.

```
sed -i "s/#baseurl/baseurl/g" /etc/yum.repos.d/epel.repo
sed -i "s/metalink/#metalink/g" /etc/yum.repos.d/epel.repo
sed -i "s@https\?://download.fedoraproject.org/pub@https://mirrors.huaweicloud.com@g" /etc/yum.repos.d/epel.repo
```

Step 8 Update the Yum source.

```
yum update
```

----End

1.16.4.4 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install all the dependencies.

```
yum install nco netcdf netcdf-devel hdf hdf-devel hdf5 hdf5-devel grib_api grib_api-devel -y
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.16.5 Obtaining Source Code

Procedure

Step 1 Download the CDO installation package **cdo-1.9.8.tar.gz** from the following link:

https://code.mpimet.mpg.de/projects/cdo/

Step 2 Use an SFTP tool to upload the CDO installation package to the */path/to/CDO* directory on the server.

----End

1.16.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/CDO
```

Step 3 Decompress the installation package.

```
tar -zxvf cdo-1.9.8.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cdo-1.9.8/
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/CDO --with-netcdf=/usr --with-hdf5=/usr --with-grib_api=/usr
```

Step 6 Perform the compilation.

```
make -j 16
make install
```

----End

1.16.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the outputted file **wrfout_d01_2005-06-04_09_00_00** to the test directory.

NOTE

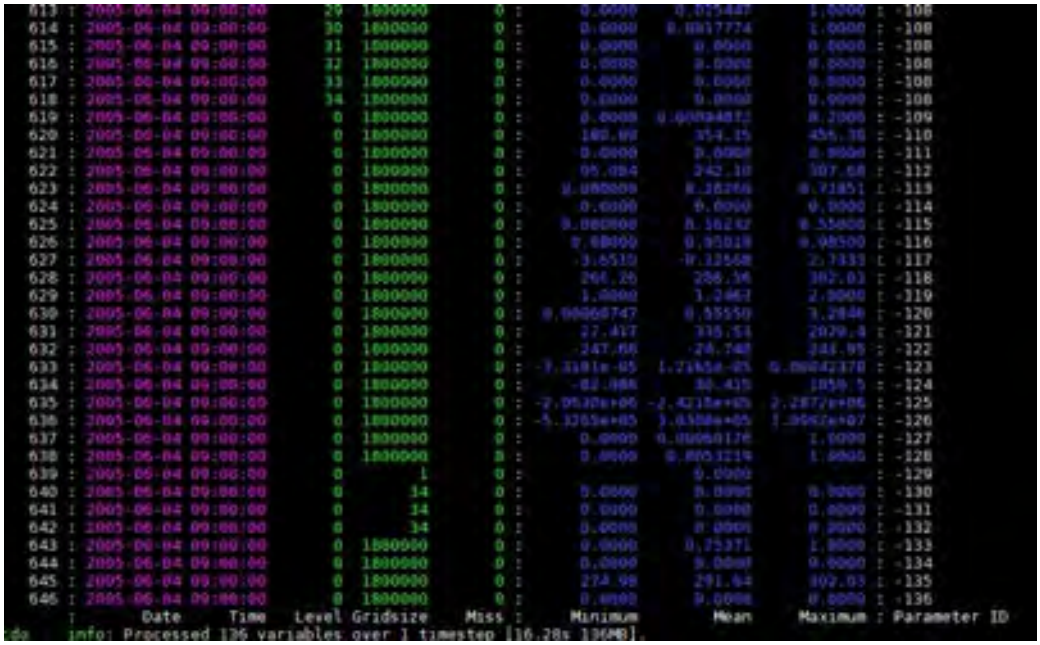
The **wrfout_d01_2005-06-04_09_00_00** file is the result file of the **count2.5** test case of WRF. For details, see **WRF 3.8.1 Porting Guide (CentOS 7.6)**.

Step 3 Perform the verification.

```
export PATH=/path/to/CDO/bin:$PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Figure 1-20 An example result



----End

1.16.8 More Resources

To obtain more resources, visit the CDO official website:

https://code.mpimet.mpg.de/projects/cdo

1.17 ecCodes 2.18.0 Porting Guide (CentOS 7.6)

1.17.1 Introduction

ecCodes is a package developed by ECMWF which provides an application programming interface and a set of tools for decoding and encoding messages in the following formats:

- WMO FM-92 GRIB edition 1 and edition 2
- WMO FM-94 BUFR edition 3 and edition 4
- WMO GTS abbreviated header (only decoding)

A useful set of command line tools provide quick access to the messages. C, Fortran 90 and Python interfaces provide access to the main ecCodes functionality.

ecCodes is an evolution of GRIB-API. It is designed to provide the user with a simple set of functions to access data from several formats with a key/value approach.

Programming language: C/Fortran

Brief description: Provides an application programming interface and a set of tools for decoding and encoding messages in WMO GRIB and BUFR formats.

Open source license: Apache License 2.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

ecCodes 2.18.0

1.17.2 Environment Requirements

Hardware Requirements

Table 1-83 lists the hardware requirements.

Table 1-83 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 1-84 Software requirements

Item	Version	Download Link
ecCodes	2.18.0	**https://confluence.ecmwf.int/download/attachments/45757960/eccodes-2.18.0-Source.tar.gz**
CMake	3.16.4	**https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz**

OS Requirements

Table 1-85 lists the OS requirements.

Table 1-85 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.17.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ecCodes software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-86 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/ECCODES	Installation path of ecCodes	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CMAKE	Installation path of CMake	

1.17.4 Configuring the Compilation Environment

1.17.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-87 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **1.17.4.2 Installing CMake**.

1.17.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the package:

```
tar -xvf cmake-3.16.4.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to go to the directory generated after decompression:

```
cd cmake-3.16.4
```

Step 4 Run the following commands to perform the compilation and installation:

```
./bootstrap --prefix=/path/to/CMAKE
make -j 16
make install
```

Step 5 Run the following command to set environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

1.17.5 Obtaining Source Code

Procedure

Step 1 Download the ecCodes installation package **eccodes-2.18.0-Source.tar.gz** from the following link:

https://confluence.ecmwf.int/download/attachments/45757960/eccodes-2.18.0-Source.tar.gz

Step 2 Use an SFTP tool to upload the ecCodes installation package to the */path/to/ECCODES* directory on the server.

----End

1.17.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install openjpeg-devel jasper-devel python-devel numpy -y
```

Step 3 Run the following command to go to the main program installation directory:

```
cd /path/to/ECCODES
```

Step 4 Run the following command to decompress the package:

```
tar -xvf eccodes-2.18.0-Source.tar.gz
```

Step 5 Run the following command to go to the directory generated after decompression:

```
cd eccodes-2.18.0-Source
```

Step 6 Run the following commands to create the build directory.

```
mkdir build
cd build
```

Step 7 Run the following command:

```
cmake ../../eccodes-2.18.0-Source -DCMAKE_INSTALL_PREFIX=/path/to/ECCODES
```

Step 8 Run the following commands to perform compilation, testing, and installation:

```
make -j 16
ctest
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 Run the following command to set environment variables:

```
export PATH=/path/to/ECCODES/bin:$PATH
```

----End

1.17.7 Running and Verification

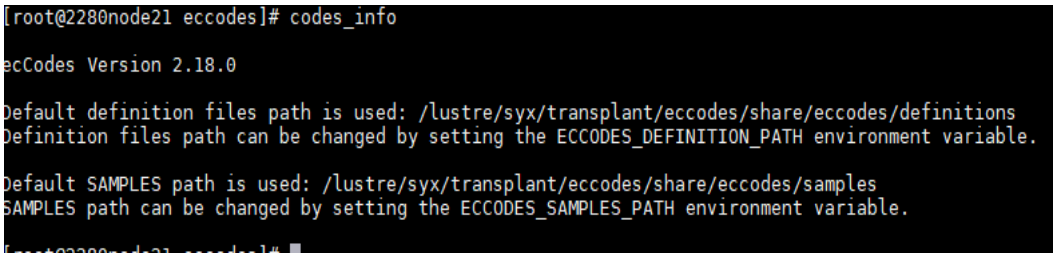
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to perform running and verification:

```
code_info
```

If the following information is displayed, the installation is successful:



----End

1.17.8 More Resources

ecCodes official website:

https://confluence.ecmwf.int/display/ECC/ecCodes+Home

1.18 ecFlow 5.5.2 Porting Guide (CentOS 7.6)

1.18.1 Introduction

ecFlow is a client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment. It provides reasonable tolerance for hardware and software failures, combined with restart capabilities.

ecFlow runs as a server receiving requests from clients. The command-line interface (CLI), the graphical interface, scripts and the Python API (application interface) are the clients. The server is based on C++/boost ASIO and uses TCP/IP for communication. Multiple servers can be run on the same hardware. ecFlow submits tasks (jobs) and receives acknowledgements from tasks via specific commands embedded in the scripts. The relationship between tasks is stored in ecFlow, and it is able to submit tasks dependent on the status of other tasks and attributes like time.

The CLI for ecFlow allows the suite definition to be loaded and retrieved from the server. Also it provides a rich set of commands for communication with the server.

 Copyright © Huawei Technologies Co., Ltd.

The Python API allows the entire suite definition structure to be specified and loaded into the server. A suite is a collection of interrelated tasks. In ecFlow suites are described by a definition file. The Python API also provides functionality for client to server communication. In addition, it allows checking of the suite, testing the defined interrelations between tasks, and other references and limits.

Programming language: C++

Brief description: A client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment.

Open source license: Apache License 2.0

Recommended Software Version

ecFlow 5.5.2

1.18.2 Environment Requirements

Hardware Requirements

Table 1-88 lists the hardware requirements.

Table 1-88 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 1-89 Software requirements

Item	Version	Download URL
ecFlow	5.5.2	https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz
Boost	1.72.0	https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz
CMake	3.16.4	https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz

OS Requirements

Table 1-90 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-90 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.18.3 Paths for Software Porting

This section describes the software installation paths involved in the ecFlow software porting.

Table 1-91 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/ECFLOW_BUILD	Installation path of ecFlow and Boost.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/CMAKE	Installation path of CMake.	

1.18.4 Configuring the Compilation Environment

1.18.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-92 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **1.18.4.2 Installing CMake**.

1.18.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the package:

```
tar -xvf cmake-3.16.4.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd cmake-3.16.4
```

Step 4 Run the following command to perform compilation and installation:

```
./bootstrap --prefix=/path/to/CMAKE
make -j 16
make install
```

Step 5 Run the following command to set environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

1.18.5 Obtaining Source Code

Procedure

Step 1 Download the ecFlow installation package **ecFlow-5.5.2-Source.tar.gz** and the Boost installation package **boost_1_72_0.tar.gz**.

ecFlow is available at: **https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz**

Boost is available at: **https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz**

Step 2 Use an SFTP tool to upload the ecFlow and Boost installation packages to the */path/to/ECFLOW_BUILD* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.18.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/ECFLOW_BUILD
```

Step 3 Run the following commands to decompress the Boost installation package:

```
tar -xvf boost_1_72_0.tar.gz
```

Step 4 Run the following command to decompress the ecFlow installation package:

```
tar -xvf ecFlow-5.5.2-Source.tar.gz
```

Step 5 Run the following commands to create the two environment variables:

```
export WK=/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source
export BOOST_ROOT=/path/to/ECFLOW_BUILD/boost_1_72_0
```

Step 6 Run the following commands to go to the Boost installation directory and build the Boost library:

```
cd $BOOST_ROOT
./bootstrap.sh
./b2 install --prefix=/path/to/ECFLOW_BUILD/boost_1_72_0
```

Step 7 Run the following command to compile the Boost library using ecFlow:

```
$WK/build_scripts/boost_build.sh
```

Step 8 Run the following command to go to the directory generated after decompression:

```
cd $WK
```

Step 9 Run the following commands to create an independent directory for ecFlow:

```
mkdir build
cd build
```

Step 10 Run the following commands to perform compilation and installation:

```
cmake .. -DENABLE_UI=OFF -DENABLE_PYTHON=OFF
make -j 16
make install
```

----End

1.18.7 Running and Verifying ecFlow

Procedure

Step 1 Run the following command to go to the test directory:

```
cd /path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build
```

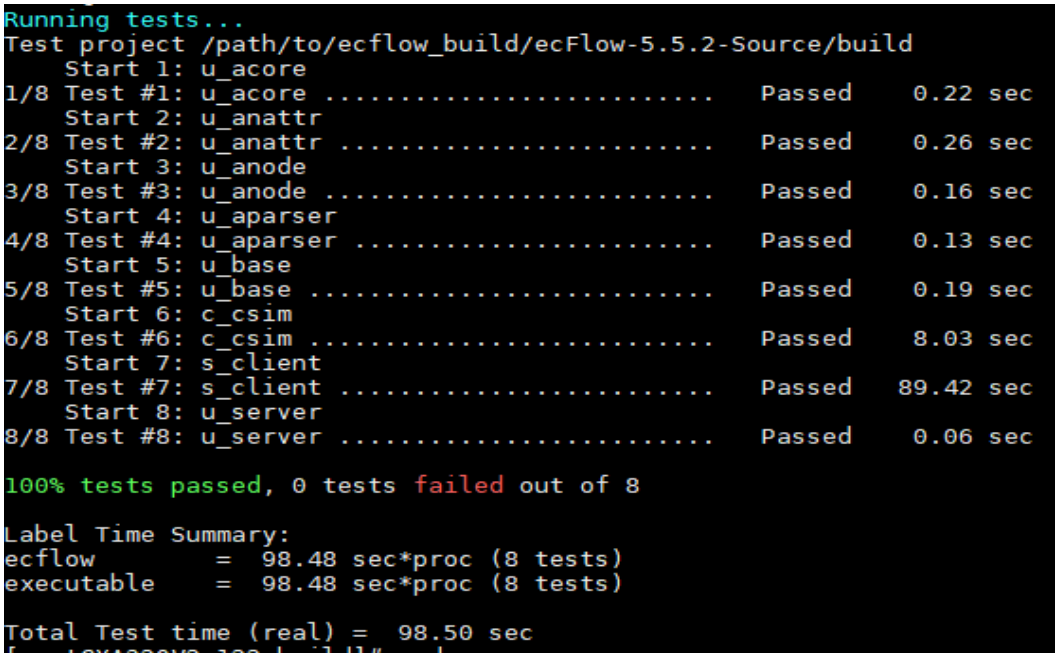
Step 2 Run the following command to perform the test:

```
make test
```

Figure 1-21 shows an example of the result.

 Copyright © Huawei Technologies Co., Ltd.

Figure 1-21 An example result



----End

1.18.8 More Resources

ecFlow official website:

https://confluence.ecmwf.int/display/ECFLOW/ecflow+home

1.19 NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6)

1.19.1 Introduction

NCEPLIBS provides the dependency libraries for the meteorological field. It is widely used to build dependency relationships for meteorological software such as weather models or weather applications.

For more information about NCEPLIBS, visit the **NCEPLIBS official website**.

Programming languages: C/Fortran

Brief description: common dependency libraries in the meteorological field

Open source license: Apache

Recommended Software Version

NCEPLIBS v1.2.0

 Copyright © Huawei Technologies Co., Ltd.

1.19.2 Environment Requirements

Hardware Requirements

Table 1-93 lists the hardware requirements.

Table 1-93 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-94 lists the software requirements.

Table 1-94 Software requirements

Item	Version	Download URL
NCEPLIBS	v1.3.0	https://github.com/NOAA-EMC/NCEPLIBS
GNU	9.1.0	https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/
Open MPI	4.0.1	https://www.open-mpi.org/software/ompi/v4.0/
CMake	3.17.3	https://github.com/Kitware/CMake/releases/download/v3.17.3/cmake-3.17.3.tar.gz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
PnetCDF	1.11.2	https://parallel-netcdf.github.io/wiki/Download.html
NetCDF	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NetCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1

OS Requirements

Table 1-95 lists the OS requirements.

Table 1-95 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

1.19.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NCEPLIBS software porting.

Table 1-96 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/NCEPLIBS	Installation path of NCEPLIBS.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/HDF5	Installation path of HDF5.	
4	/path/to/PNETCDF	Installation path of PnetCDF.	
5	/path/to/CMAKE	Installation path of CMake.	
6	/path/to/NETCDF	Installation path of NetCDF.	
		Installation path of NetCDF-Fortran.	

1.19.4 Configuring the Compilation Environment

1.19.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-97 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **1.19.4.2 Installing HDF5**.
3	Installing PnetCDF	See **1.19.4.3 Installing PnetCDF**.
4	Installing NetCDF	See **1.19.4.4 Installing NetCDF**.
5	Installing NetCDF-Fortran	See **1.19.4.5 Installing NetCDF-Fortran**.
6	Installing CMake	See **1.19.4.6 Installing CMake**.

1.19.4.2 Installing HDF5

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 5 Create a **/path/to/HDF5** directory.

```
mkdir -p /path/to/HDF5
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/HDF5  --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 7 Perform the compilation and installation.

```
make -j 16
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.19.4.3 Installing PnetCDF

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.11.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd parallel-netcdf-1.11.2
```

Step 5 Create a **/path/to/PNETCDF** directory.

```
mkdir -p /path/to/PNETCDF
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 7 Perform the compilation and installation.

```
make -j 16
make install
```

----End

1.19.4.4 Installing NetCDF

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Install the base dependencies.

```
yum install curl* -y
```

Step 4 Decompress the NetCDF installation package.

```
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd netcdf-4.4.1.1
```

Step 6 Create a **/path/to/NETCDF** directory.

```
mkdir -p /path/to/NETCDF
```

Step 7 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile  CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/PNETCDF/lib" CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
```

Step 8 Perform the compilation and installation.

```
make -j 16
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.19.4.5 Installing NetCDF-Fortran

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib" CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 6 Perform the compilation and installation.

```
make -j 16
make install
```

----End

1.19.4.6 Installing CMake

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar zxvf cmake-3.17.3.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.17.3
```

Step 4 Perform the compilation and installation.

```
./configure -prefix=/path/to/CMAKE
make -j 32
make install
```

Step 5 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

1.19.5 Obtaining Source Code

Procedure

Step 1 Obtain the NCEPLIBS package **NCEPLIBS-1.2.0.tar.gz** from the following link:

https://github.com/NOAA-EMC/NCEPLIBS

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the downloaded installation package to the */path/to/NCEPLIBS* directory on the server.

----End

1.19.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/NCEPLIBS
```

Step 3 Create directories **build** and **nceplibs**, and go to the **build** directory.

```
mkdir -p build && mkdir nceplibs && cd build
```

Step 4 Build the software.

```
cmake ../ -DCMAKE_INSTALL_PREFIX=../nceplibs
```

Step 5 Perform the compilation and installation.

```
make -j 16
```

NOTE

If an error is reported during the installation, see **1.19.8 Troubleshooting** for the solution.

----End

1.19.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the library file installation directory.

```
cd /path/to/NCEPLIBS/NCEPLIBS
```

Step 3 View the library file.

```
ll
```

If the following dependency library information is displayed, the installation is complete:

Figure 1-22 Dependency libraries

```
total 0
drwxr-xr-x 5 root root 51 Sep 27 09:58 bacio-2.4.1
drwxr-xr-x 7 root root 88 Sep 27 09:59 bufr-11.4.0
drwxr-xr-x 4 root root 32 Sep 27 09:59 crtm-2.3.0
drwxr-xr-x 5 root root 51 Sep 27 09:59 g2-3.4.1
drwxr-xr-x 5 root root 45 Sep 27 10:30 g2tmpl-1.9.1
drwxr-xr-x 4 root root 32 Sep 27 09:58 gfsio-1.4.1
drwxr-xr-x 6 root root 68 Sep 27 09:59 ip-3.3.3
drwxr-xr-x 5 root root 51 Sep 27 09:58 landsfcutil-2.4.1
drwxr-xr-x 5 root root 45 Sep 27 10:35 nceppost-dceca26
drwxr-xr-x 5 root root 43 Sep 27 10:33 nemsio-2.5.2
drwxr-xr-x 4 root root 32 Sep 27 10:35 nemsiogfs-2.5.3
drwxr-xr-x 4 root root 32 Sep 27 09:59 sfcio-1.4.1
drwxr-xr-x 4 root root 32 Sep 27 09:59 sigio-2.3.2
drwxr-xr-x 6 root root 68 Sep 27 09:59 sp-2.3.3
drwxr-xr-x 6 root root 68 Sep 27 10:33 w3emc-2.7.3
drwxr-xr-x 6 root root 68 Sep 27 09:59 w3nco-2.4.1
drwxr-xr-x 5 root root 43 Sep 27 10:00 wgrib2-2.0.8
drwxr-xr-x 4 root root 32 Sep 27 09:59 wrf_io-1.1.1
```

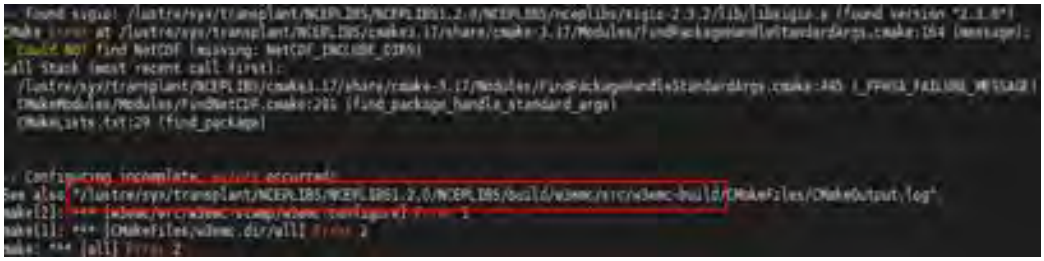
----End

1.19.8 Troubleshooting

Problem 1: An Error Is Reported During Software Build

Symptom:

If the following error is reported during the installation, add the NetCDF installation path to the specified file based on the error information.



Possible Cause:

The NetCDF installation path is not added to the specified file.

Procedure:

Step 1 Modify the **CmakCache.txt** file.

1. Open **CmakCache.txt**.
   vi */path/to/NCEPLIBS*/**NCEPLIBS1.3.0/build/w3emc/src/w3emc-build/CmakCache.txt**
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   //NetCDF C include directory
   NetCDF_C_INCLUDE_FILE:FILEPATH=NetCDF_C_INCLUDE_FILE-NOTFOUND
   ```

   After the modification:

   ```
   //NetCDF C include directory
   NetCDF_C_INCLUDE_FILE:FILEPATH=/path/to/NETCDF/include
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Build the software. If a similar error occurs, perform the previous steps to rectify the error.

```
make -j 16
```

----End

1.19.9 More Resources

For details about how to install NCEPLIBS, visit the following website:

https://github.com/NOAA-EMC/NCEPLIBS%0c

1.20 PyFerret 7.6.0 Porting Guide (CentOS 7.6)

1.20.1 Introduction

Ferret is an interactive computer visualization and analysis environment designed to meet the needs of oceanographers and meteorologists analyzing large and complex gridded data sets.

Programming languages: C/Fortran/Python

Brief description: ecosystem dynamics modeling software

Open source license: GPL

Recommended Software Version

PyFerret 7.6.0

1.20.2 Environment Requirements

Hardware Requirements

Table 1-98 lists the hardware requirements.

Table 1-98 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 1-99 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-99 Software requirements

Item	Version	Download URL
PyFerret	7.6.0	**https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0**
GNU	9.3.0	**https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/**
HDF5	1.10.6	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.6/src/**
NetCDF	4.7.3	**https://github.com/Unidata/netcdf-c/releases/tag/v4.7.3**
NetCDF-Fortran	4.5.2	**https://github.com/Unidata/netcdf-fortran/releases/tag/v4.5.2**

OS Requirements

Table 1-100 lists the OS requirements.

Table 1-100 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

1.20.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PyFerret software porting.

Table 1-101 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PyFerret	Installation path of PyFerret.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/HDF5	Installation path of HDF5.	document with the actual paths planned during the installation process.
6	/path/to/NETCDF	Installation path of NetCDF.	
		Installation path of NetCDF-Fortran.	

1.20.4 Configuring the Compilation Environment

1.20.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-102 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Configuring the Yum source and installing the dependencies	See **1.20.4.3 Configuring the Yum Source and Installing the Dependencies**.
3	Installing HDF5	See **1.20.4.4 Installing HDF5**.
4	Installing NetCDF	See **1.20.4.5 Installing NetCDF**.
5	Installing NetCDF-Fortran	See **1.20.4.6 Installing NetCDF-Fortran**.

 Copyright © Huawei Technologies Co., Ltd.

1.20.4.2 Setting Up the Basic Environment

Install GNU. For details, see "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

1.20.4.3 Configuring the Yum Source and Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Configure the basic Yum source.

```
cd /etc/yum.repos.d/
mv CentOS-Base.repo CentOS-Base.repo-bak
wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-AltArch-7.repo
```

Step 3 Configure the additional Yum source.

```
vim epel.repo
```

Add the following content to the file:

```
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.tools.huawei.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0
```

Step 4 Clear the Yum cache.

```
yum clean all
yum makecache
```

Step 5 Install the dependencies.

```
yum install python numpy libgfortran python-qt4 scipy pyshp -y
```

----End

1.20.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin: $PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64: $LD_LIBRARY_PATH
```

Step 3 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.6.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd hdf5-1.10.6
```

Step 5 Create a **/path/to/HDF5** directory.

```
mkdir -p /path/to/HDF5
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-shared CC=gcc CXX=g++ FC=gfortran F77=gfortran CFLAGS=-fPIC
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Perform the compilation and installation.

```
make -j 16
make install
```

----End

1.20.4.5 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/GNU/bin: $PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64: $LD_LIBRARY_PATH
```

Step 3 Install the base dependencies.

```
yum install curl* -y
```

Step 4 Decompress the NetCDF installation package.

```
tar -xvf netcdf-4.7.3.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.3
```

Step 6 Create a **/path/to/NETCDF** directory.

```
mkdir -p /path/to/NETCDF
```

Step 7 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-largefile CC=gcc CXX=g++ FC=gfortran F77=gfortran CPPFLAGS="-I/path/to/HDF5/include" LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-L/path/to/HDF5/lib -I/path/to/HDF5/include"
```

Step 8 Perform the compilation and installation.

```
make -j 16
make install
```

----End

1.20.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install NetCDF-Fortran.

```
export PATH=/path/to/GNU/bin: $PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64: $LD_LIBRARY_PATH
```

Step 3 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.5.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.5.2
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=gcc CXX=g++ FC=gfortran F77=gfortran
```

 Copyright © Huawei Technologies Co., Ltd.

```
CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib" CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 6 Perform the compilation and installation.

```
make -j 16
make install
```

----End

1.20.5 Obtaining Source Code

Procedure

Step 1 Obtain the PyFerret installation package **PyFerret-7.6.0.tar.gz** from the following link:

https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0

Step 2 Use an SFTP tool to upload **PyFerret-7.6.0.tar.gz** to the */path/to/PyFerret* directory on the server.

----End

1.20.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/PyFerret
```

Step 3 Install the base dependencies.

```
yum -y install cairo*
yum -y install pango-devel pangomm pango-tests pangomm-devel pangomm-doc
```

Step 4 Decompress the installation package.

```
tar -zxvf PyFerret-7.6.0.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd PyFerret-7.6.0
```

Step 6 Copy the configuration file.

```
cp site_specific.mk.in site_specific.mk
```

Step 7 Modify the **site_specific.mk** file:

1. Open **site_specific.mk**.
   ```
   vim site_specific.mk
   ```
2. Press **i** to go to the insert mode.

   Add the following content to line 10:

   ```
   HOME=/path/to/PyFerret
   ```

   Modify the following information:

   Before the modification:

```
DIR_PREFIX = $(HOME)/git/PyFerret
INSTALL_FER_DIR = $(HOME)/PyFerret
BUILDTYPE = x86_64-linux
PYTHON_EXE = python3.6
HDF5_LIBDIR = /usr/local/hdf5/lib
NETCDF_LIBDIR = /usr/local/netcdf/lib
```

After the modification:

```
DIR_PREFIX = $(HOME)/PyFerret-7.6.0
INSTALL_FER_DIR = $(HOME)/pyferret
BUILDTYPE = aarch64-linux
PYTHON_EXE = python2.7
HDF5_LIBDIR = /path/to/HDF5/lib
NETCDF_LIBDIR = /path/to/NETCDF/lib
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 8 Copy the configuration file.

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 9 Modify the **platform_specific.mk.aarch64-linux** file.

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 10 Go to the **ef_utility** directory.

```
cd external_functions/ef_utility/
```

Step 11 Copy the configuration file.

```
cp site_specific.mk.in site_specific.mk
```

Step 12 Modify the **site_specific.mk** file.

1. Open **site_specific.mk**.
   ```
   vim site_specific.mk
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   BUILDTYPE = x86_64-linux
   PYTHON_EXE = python3.6
   ```

   After the modification:

   ```
   BUILDTYPE = aarch64-linux
   PYTHON_EXE = python2.7
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 13 Copy the configuration file.

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 14 Modify the **platform_specific.mk.aarch64-linux** file.

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 15 Go back to the original directory.

```
cd -
```

Step 16 Perform the installation.

```
make
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.20.7 Running and Verification

Procedure

Step 1 Download the PyFerret installation package **FerretDatasets-7.6.tar.gz** from the following link:

https://github.com/NOAA-PMEL/FerretDatasets/archive/v7.6.tar.gz

Step 2 Use an SFTP tool to upload **FerretDatasets-7.6.tar.gz** to the */path/to/PyFerret* directory on the server.

Step 3 Use PuTTY to log in to the server as the root user.

Step 4 Decompress **FerretDatasets-7.6.tar.gz**.

```
tar -zxvf FerretDatasets-7.6.tar.gz
```

Step 5 Go to the PyFerret installation directory.

```
cd pyferret/bin
```

Step 6 Run the **Finstall** script to generate a file.

```
./Finstall
```

NOTE

During the process, you need to type the PyFerret installation path, FerretDatasets package path, and **ferret_paths.sh** script generation path. This process is an interactive process, and you need to type the needed information at the following prompts:

- Prompt: **Proceed? (y/n)**: Type **y** and press **Enter** to continue.
- **FER_DIR**: PyFerret installation path. Type */path/to/PyFerret/***pyferret**.
- **FER_DSETS**: FerretDatasets package path. Type */path/to/PyFerret/***FerretDatasets-7.6**.
- **desired ferret_paths location**: **ferret_paths.sh** script path. Type */path/to/PyFerret/*.
- ferret_paths link to create? (c/s/n): Type **n**.
- **python executable to use**: Type **python**.

Step 7 Load the environment variables.

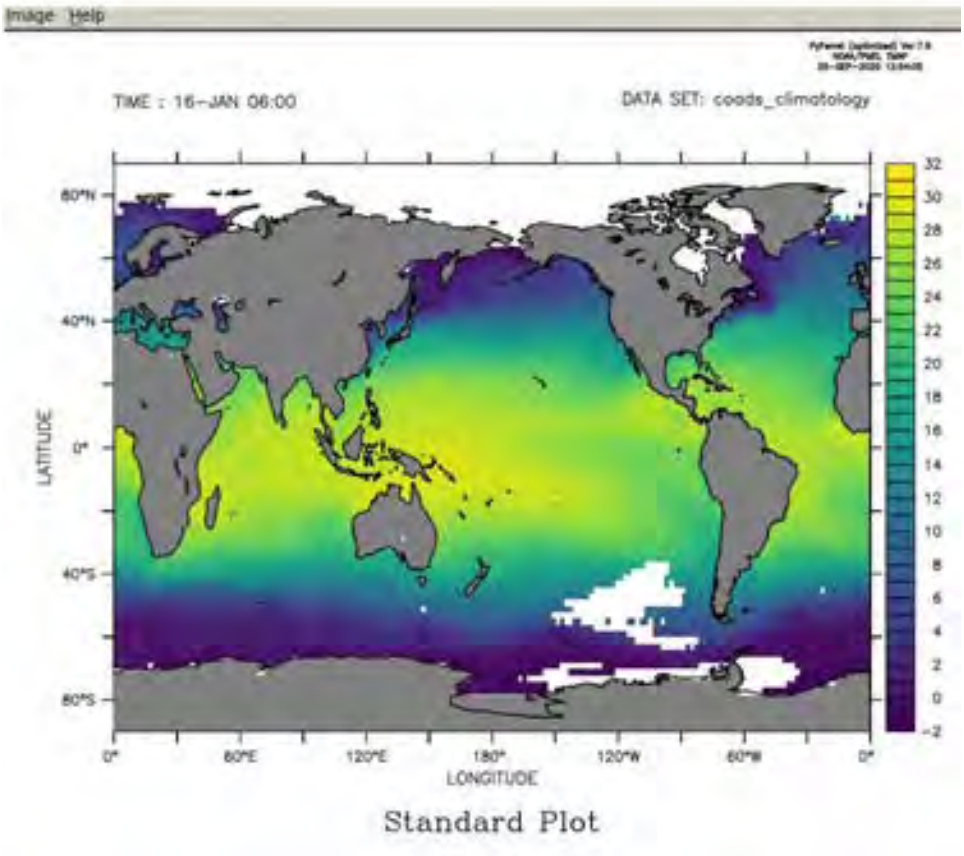
```
source ferret_paths.sh
```

Step 8 Start the main program.

```
./pyferret
yes? go /help mp_demo
yes? USE coads_climatology
yes? SHADE/X=0:360/Y=-90:90/L=1/TITLE="Standard Plot" sst
yes? GO fland
yes? GO land
yes? exit
```

 Copyright © Huawei Technologies Co., Ltd.

Figure 1-23 An example output



----End

1.20.8 More Resources

To obtain more resources, visit the PyFerret official website at **https://ferret.pmel.noaa.gov/Ferret/**.

1.21 NCL 6.3.0 Porting Guide (CentOS 7.6)

1.21.1 Introduction

NCL is designed specifically for scientific data processing and visualization, widely used in the area of atmospheric modeling.

Programming languages: C/Fortran

Brief description: data processing and visualization

Recommended Software Version

NCL 6.3.0

1.21.2 Environment Requirements

Hardware Requirements

Table 1-103 lists the hardware requirements.

Table 1-103 Hardware requirements

Item	Description
CPU	Kunpeng 920

OS Requirements

Table 1-104 lists the OS requirements.

Table 1-104 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

1.21.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NCL software porting.

Table 1-105 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	-	Installed in the system directory using Yum.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

1.21.4 Configuring the Compilation Environment

Procedure

Step 1 Configure the Yum network source. For details, see the Huawei mirror at **https://mirrors.huaweicloud.com/**.

Step 2 Configure the **CentOS-AltArch** and **EPEL** images.

Step 3 Check whether the configuration is successful.

```
yum repolist
```

If information similar to the following information is displayed, the configuration is successful.

```
repo id                repo name
base/aarch64           CentOS-7 - Base - mirrors.tools.huawei.com
epel/aarch64           Extra Packages for Enterprise Linux 7 - aarch64
extras/aarch64         CentOS-7 - Extras - mirrors.tools.huawei.com
updates/aarch64        CentOS-7 - Updates - mirrors.tools.huawei.com
repolist: 21,591
```

----End

1.21.5 Compilation and Installation

Procedure

CAUTION

Use the GCC embedded in the OS to perform the installation. Do not load the GCC 9.3 environment.

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install NCL and its dependencies.

```
yum install ncl.aarch64 ncl-common.noarch ncl-devel.aarch64 ncl-examples.noarch -y
```

Step 3 Set the environment variables.

```
export NCARG_FONTCAPS=/usr/lib64/ncarg/fontcaps
export NCARG_GRAPHCAPS=/usr/lib64/ncarg/graphcaps
export NCARG_ROOT=/usr
export NCARG_DATABASE=/usr/lib64/ncarg/database
export NCARG_LIB=/usr/lib64/ncarg
export NCARG_NCARG=/usr/share/ncarg
export EXTRA_CCOPTIONS=" -fsigned-char "$EXTRA_CCOPTIONS
```

Step 4 Perform the verification:

```
ng4ex xy01n -W png
```

If the NCL main program reports an error, as shown in the following figure, recompile the main program.

 Copyright © Huawei Technologies Co., Ltd.

```
NCAR Graphics NCL Example <xy01n>

  Copying xy01n.ncl
  Copying xy01.res

Running NCL...
 Copyright (C) 1995-2015 - All Rights Reserved
 University Corporation for Atmospheric Research
 NCAR Command Language Version 6.3.0
 The use of this software is governed by a License Agreement.
 See http://www.ncl.ucar.edu/ for more details.
Segmentation fault (core dumped)

ncl failed.
```

Step 5 Download the **ncl-6.3.0-6.el7.src.rpm** RPM package of NCL.

```
yum install -y yum-utils
yumdownloader --source ncl
```

Step 6 Recompile NCL and decompress the NCL installation package.

```
rpm -ivh ncl-6.3.0-6.el7.src.rpm
cd ~/rpmbuild/SOURCES
tar zxvf ncl_ncarg-6.3.0.tar.gz
```

Step 7 Modify the **/libncarg_c/yMakefile** file.

1. Open **/libncarg_c/yMakefile**.
   ```
   vim ncl_ncarg-6.3.0/common/src/libncarg_c/yMakefile
   ```
2. Press **i** to enter the insert mode and modify line 35.
   ```
   #elif (defined(ia64) || defined(x86_64) || defined(amd64) || defined(aarch64))
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Modify the **/nio/yMakefile** file.

1. Open **/nio/yMakefile**.
   ```
   vim ncl_ncarg-6.3.0/ni/src/lib/nio/yMakefile
   ```
2. Press **i** to enter the insert mode and modify line 85.
   ```
   #if defined(IRIX64) || defined(x86_64) || defined(__LP64__) || defined(aarch64)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Modify the **/ncl/yMakefile** file.

1. Open **/ncl/yMakefile**.
   ```
   vim ncl_ncarg-6.3.0/ni/src/ncl/yMakefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   - Modify line 46:
     ```
     EXTRA_CCOPTIONS = -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -fsigned-char
     ```
   - Modify line 113:
     ```
     #if defined(IRIX64) || defined(x86_64) || defined(ia64) || defined(__LP64__) || defined(aarch64)
     ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Delete the original TAR packages and pack the updated source code.

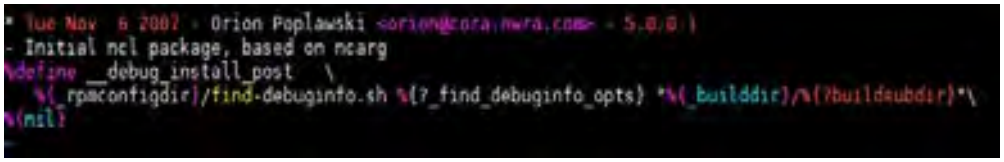
```
rm -f ncl_ncarg-6.3.0.tar.gz
tar zcvf ncl_ncarg-6.3.0.tar.gz ncl_ncarg-6.3.0
rm -rf ncl_ncarg-6.3.0
```

Step 11 Recompile NCL.

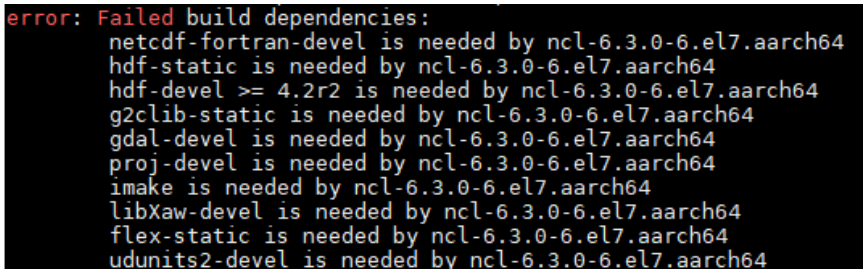
1. Go to the **SPECS** directory.
   ```
   cd ~/rpmbuild/SPECS
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Modify the **ncl.spec** file.
   ```
   vim ncl.spec
   ```
3. Press **i** to go to the insert mode, and append the following content to the file:
   ```
   %define __debug_install_post   \
   %{_rpmconfigdir}/find-debuginfo.sh %{?_find_debuginfo_opts} "%{_builddir}/%{?buildsubdir}"\
   %{nil}
   ```
   See the following figure:

4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
5. Perform the packing and compilation.
   ```
   rpmbuild -ba ncl.spec
   ```
   If an error is reported indicating that dependencies are missing, install the dependencies in sequence using Yum, as shown in the following figure:


   ```
   yum install netcdf-fortran-devel hdf-static hdf-devel g2clib-static gdal-devel proj-devel imake libXaw-devel flex-static udunits2-devel
   ```
   The **rpmbuild -ba ncl.spec** command can be executed when **exit 0** is displayed, as shown in the following figure:

Step 12 Install NCL (delete the original NCL first).

yum erase ncl.aarch64 ncl-common.noarch ncl-devel.aarch64 ncl-examples.noarch -y

```
cd ~/rpmbuild/RPMS
cp noarch/* aarch64/
cd aarch64
yum localinstall ncl-6.3.0-6.el7.aarch64.rpm ncl-common-6.3.0-6.el7.noarch.rpm ncl-debuginfo-6.3.0-6.el7.aarch64.rpm ncl-devel-6.3.0-6.el7.aarch64.rpm ncl-examples-6.3.0-6.el7.noarch.rpm -y
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.21.6 Running and Verification

Procedure

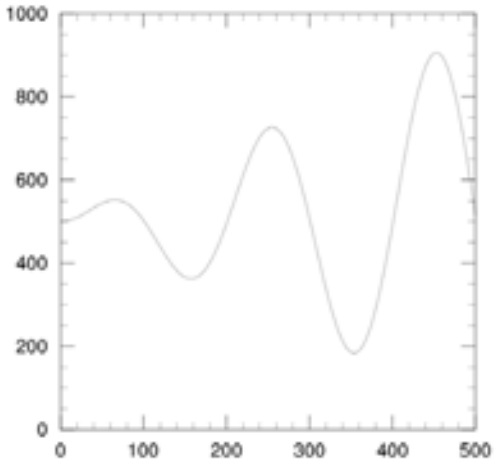
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
ng4ex xy01n -W png
```

If the **xy01n.png** image is generated in the current directory, the execution is successful, as shown in **Figure 1-24**:

Figure 1-24 Verification result



----End

1.22 ARPS 5.3.4 Porting Guide (CentOS 7.6)

1.22.1 Introduction

The Advanced Regional Prediction (ARPS) is a comprehensive regional to stormscale atmospheric modeling/prediction system.

For more information about ARPS, visit the **ARPS official website**.

Programming language: Fortran

Brief description: ARPS is a comprehensive regional to stormscale atmospheric modeling/prediction system.

Open source license: custom open source license

Recommended Software Version

ARPS 5.3.4

1.22.2 Environment Requirements

Hardware Requirements

Table 1-106 lists the hardware requirements.

Table 1-106 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-107 lists the software requirements.

Table 1-107 Software requirements

Item	Version	Download Address
ARPS	5.3.4	http://www.caps.ou.edu/ARPS/arpsdown.html
HDF4	4.2.13	Installing the EPEL source using Yum

OS Requirements

Table 1-108 lists the OS requirements.

Table 1-108 OS requirements

Item	Version	Download URL
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image

1.22.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ARPS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-109 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/HDF4	Installation path of HDF4.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
4	/path/to/ARPS	Installation path of ARPS.	

1.22.4 Configuring the Compilation Environment

1.22.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-110 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing HDF4	See **1.22.4.2 Installing HDF4**.

 Copyright © Huawei Technologies Co., Ltd.

1.22.4.2 Installing HDF4

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install HDF4 using Yum.

```
yum install hdf-devel.aarch64 -y
```

Step 3 Create an **HDF4** directory.

```
mkdir -p /path/to/HDF4/bin
mkdir -p /path/to/HDF4/lib
mkdir -p /path/to/HDF4/include
```

Step 4 Copy the related files to the planned directory.

```
cp /usr/bin/hdf* /path/to/HDF4/bin
cp /usr/lib64/hdf/* /path/to/HDF4/lib
cp /usr/include/hdf/* /path/to/HDF4/include
```

Step 5 Set the HDF4 environment.

```
export HDFPATH=/path/to/HDF4
```

----End

1.22.5 Obtaining the Source Code

Procedure

Step 1 Log in to the official website, register an account, and download the ARPS installation package.

http://www.caps.ou.edu/ARPS/arpsdown.html

Step 2 Use an SFTP tool to upload the ARPS installation package to the */path/to/ARPS* directory on the server.

----End

1.22.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ARPS installation package.

```
cd /path/to/ARPS
tar -xvf arps5.3.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd arps5.3.4
```

Step 4 Edit the **makearps** file.

```
sed -i '259s/ifort/gfortran/g' makearps
sed -i '3254s/-ffree-form/-ffree-form -Wno-argument-mismatch/g' makearps
```

Step 5 Perform the first compilation.

```
./makearps -io hdf  arps
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 After an error is reported, edit the **module_precision.f90** file.

1. Open the **module_precision.f90** file.
   ```
   vi src/arps/module_precision.f90
   ```
2. type **34dd** to delete the comments of the first 34 lines.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 7 Edit the **Makefile** file.

```
sed -i '501s/-$(RM)/\#-$(RM)/g' src/arps/Makefile
sed -i '502s/$(CPP)/\#$(CPP)/g' src/arps/Makefile
```

Step 8 Perform the second compilation.

```
./makearps -io hdf  arps
```

Step 9 After an error is reported, edit the **module_mp_wsm6.f90** file.

1. Open the **module_mp_wsm6.f90** file.
   ```
   vi src/arps/module_mp_wsm6.f90
   ```
2. type **35dd** to delete the comments of the first 35 lines.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 10 Edit the **Makefile** file.

```
sed -i '599s/-$(RM)/\#-$(RM)/g' src/arps/Makefile
sed -i '600s/$(CPP)/\#$(CPP)/g' src/arps/Makefile
```

Step 11 Perform the third compilation.

```
./makearps -io hdf  arps
```

Step 12 After an error is reported, edit the **lnxlib3d.f90** file.

1. Open the **lnxlib3d.f90** file.
   ```
   vi src/arps/lnxlib3d.f90
   ```
2. type **50dd** to delete the comments of the first 50 lines.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 13 Edit the **Makefile** file.

```
sed -i '464s/$(CPP)/\#$(CPP)/g' src/arps/Makefile
```

Step 14 Compile the ARPS serial main program.

```
./makearps -io hdf  arps
```

Step 15 Compile the ARPS parallel main program.

```
./makearps -io hdf  arps_mpi
```

Step 16 Check whether an executable file is generated.

```
ll /path/to/ARPS/arps5.3.4/bin
```

If the following information is displayed, the executable file is generated:

```
-rwxr-xr-x 1 root root 4609344 Jun 25 17:12 arps
-rwxr-xr-x 1 root root 4659672 Jun 25 17:21 arps_mpi
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.22.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Configure the environment variables of the ARPS executable file.

```
export PATH=/path/to/ARPS/arps5.3.4/bin:$PATH
```

Step 3 Go to the test directory.

```
cd /path/to/ARPS/arps5.3.4/sounding
```

Step 4 Run the serial ARPS.

```
{ time -p arps ../input/arps.input arps.output ; } 2>&1 | tee -a arps.log
```

Step 5 Modify the input file **arps.input**.

```
sed -i '211s/nproc_x = 1/nproc_x = 8/g' ../input/arps.input
sed -i '212s/nproc_y = 1/nproc_y = 16/g' ../input/arps.input
```

NOTE

nproc_x indicates the number of processes in the x direction, and **nproc_y** indicates the number of processes in the y direction. The product of **nproc_x** and **nproc_y** must be the same as the value of **np** in **Step 6**.

Step 6 Run the parallel ARPS.

```
{ time -p mpirun --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm
--bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 -np 128
arps_mpi ../input/arps.input arps.output ; } 2>&1 | tee -a arps_mpi_128.log
```

NOTE

The **-np** parameter indicates the total number of processes.

Step 7 View the last line **real** in the **arps.log** and **arps_mpi_128.log** files. A smaller value indicates a higher performance.

```
vi arps.log
```

Or

```
vi arps_mpi_128.log
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
real 11.96
user 1195.39
sys 145.14
```

----End

1.23 CAMx 7.10 Porting Guide (openEuler 21.03)

1.23.1 Introduction

CAMx is a multi-scale photochemical modeling system for gas and particulate air pollution.

For more information about CAMx, visit the **CAMx official website**.

Programming language: Fortran

Brief description: A multi-scale photochemical modeling system.

Open source license: custom open source license

Recommended Software Version

CAMx 7.10

1.23.2 Environment Requirements

Hardware Requirements

Table 1-111 lists the hardware requirements.

Table 1-111 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-112 lists the software requirements.

Table 1-112 Software requirements

Item	Version	Download URL
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
NetCDF-C	4.4.1.1	https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.4.1.1.tar.gz
NetCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.1.tar.gz
CAMx	7.10	https://camx-wp.azurewebsites.net/getmedia/CAMx_v7.10.src.210105.tgz
Test file 1 (version-specific input file)	7.10.	https://camx-wp.azurewebsites.net/getmedia/v7.10.specific.inputs.210105.tgz
Test file 2 (meteorological input file)	7.10	https://camx-wp.azurewebsites.net/getmedia/CAMx7-10-test_run-inputs_met-201215.tgz
Test file 3 (emission input file)	7.10	https://camx-wp.azurewebsites.net/getmedia/CAMx7-10-test_run-inputs_emiss-210105.tgz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Test file 4 (input file for IC/BC, optical decomposition, and others)	7.10	https://camx-wp.azurewebsites.net/getmedia/CAMx7-10-test_run-inputs_other-201215.tgz

OS Requirements

Table 1-113 lists the OS requirements.

Table 1-113 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	https://gitee.com/openeuler/kernel/tree/kernel-4.19/
BiSheng Compiler	1.3.1	https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng
Hyper MPI	1.0.2	https://www.hikunpeng.com/en/developer/hpc/hypermpi

1.23.3 Paths for Software Porting

This section describes the software installation paths involved in the CAMx software porting.

Table 1-114 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/install**	Installation path of Hyper MPI.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	**/opt/compiler**	Installation path of the BiSheng Compiler.	
3	**/path/to/HDF5**	Installation path of HDF5.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
4	**/path/to/NETCDF**	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	**/path/to/CAMX**	Installation path of CAMx.	
6	**/path/to/CASE**	Path for storing the CAMx test cases.	

1.23.4 Configuring the Compilation Environment

1.23.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-115 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **1.23.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.23.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.23.4.4 Installing NetCDF-Fortran**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/install/hmpi/include
export PATH=/path/to/install/hmpi/bin:/path/to/install/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/install/hmpi/
export LD_LIBRARY_PATH=/path/to/install/hmpi/lib:/path/to/install/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

1.23.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

NOTE

If the server has multiple MPIs, **CC**, **FC**, and **CXX** in the preceding command can be represented using absolute paths.

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify line 11835 in the file.
   ```
   wl="-Wl,"
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

 Copyright © Huawei Technologies Co., Ltd.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.23.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.4.1.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/include" CC=mpicc --disable-dap
```

Step 5 Perform the installation.

```
make -j
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.23.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -zxvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11838 in the file.
   ```
   wl='-Wl,'
   ```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the installation.

```
make -j
make install
```

----End

1.23.5 Obtaining the Source Code

Procedure

Step 1 Download the CAMx installation package.

https://camx-wp.azurewebsites.net/getmedia/CAMx_v7.10.src.210105.tgz

Step 2 Use an SFTP tool to upload the CAMx installation package to the */path/to/CAMX* directory on the server.

----End

1.23.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the CAMx installation package directory.

```
cd /path/to/CAMX
```

Step 3 Decompress the CAMx installation package.

```
tar -xvf CAMx_v7.10.src.210105.tgz
```

Step 4 Go to the source code directory.

```
cd src.v7.10
```

Step 5 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode and modify lines 60, 61, and 250 of **Makefile**.
   ```
   60 MPI_INST = /path/to/install/ompi
   61 NCF_INST = /path/to/NETCDF
   250 FLGS=$(INCLUDES) -O2 -Wno-unused-command-line-argument -fno-fortran-main -mcmodel=small -J${MOD_DIR} -fno-align-commons -fconvert=big-endian -frecord -marker=4 -ffixed-line-length-0 -lgfortran
   ```

   NOTE

   - After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
   - If an error similar to the line 25 error is reported when you run the **FLGS** command, set the **FLGS** parameter as follows:

     FLGS=$(INCLUDES) -O2 -mcmodel=small -J${MOD_DIR} -fno-align-commons -fconvert=big-endian -frecord-marker=4 -ffixed-line-length-132

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 6 Modify the **MPI/util/Makefile** file.

1. Open the **MPI/util/Makefile** file.
   vi MPI/util/Makefile
2. Press **i** to enter the edit mode and modify lines 4 and 18 of **MPI/util/Makefile**.
   4 MPI_INST = */path/to/install/ompi*
   18 CFLAGS = -mcmodel=small -O2 -I $(INC)

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 7 Compile and install the software.

make COMPILER=gfortran MPI=mpich3 NCF=NCF4_C -j 8

After the compilation is successful, the executable file **CAMx.v7.10.MPICH3.NCF4.gfortran** is generated in the current directory. You can run the **ls** or **ll** command to view the file. The following figure shows the command output.

----End

1.23.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the test case file.

cd */path/to/CASE*
tar -xvf v7.10.specific.inputs.210105.tgz
tar -xvf CAMx7-10-test_run-inputs_met-201215.tgz
tar -xvf CAMx7-10-test_run-inputs_emiss-210105.tgz
tar -xvf CAMx7-10-test_run-inputs_other-201215.tgz

Step 3 Go to the test directory.

cd runfiles

Step 4 Modify the test script.

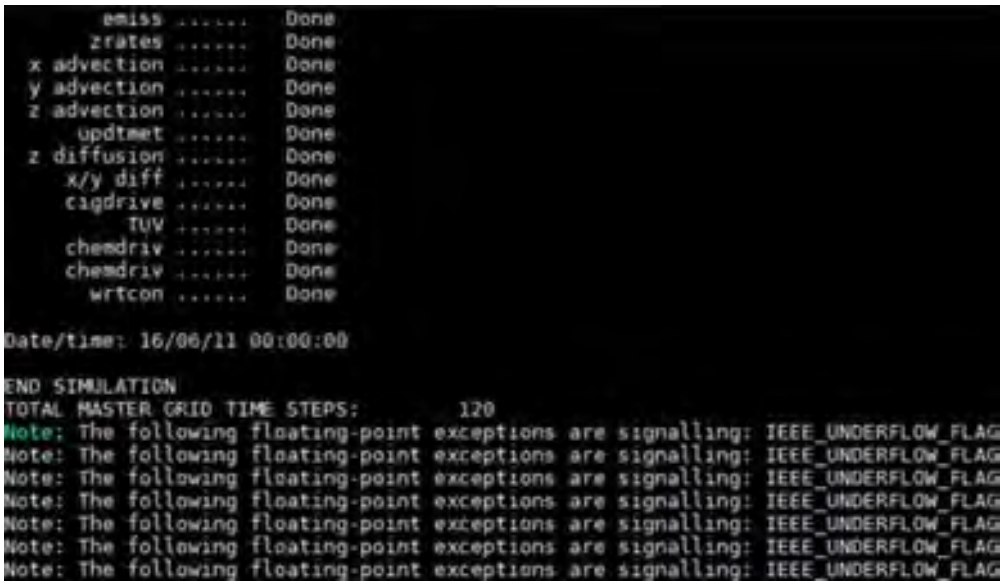
1. Open the test script.
   vi CAMx_v7.10.36.12.20160610-11.MPICH3.job
2. Press **i** to enter the edit mode and modify the file as follows:
   set EXEC = "/path/to/CAMX/src.v7.10/CAMx.v7.10.MPICH3.NCF4.gfortran"
   cat << ieof > nodes
   172.35.153.48
   set NUMPROCS = 8
   if(! { mpirun --allow-run-as-root -machinefile nodes -np $NUMPROCS $EXEC }) then
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 5 Perform the test.

 Copyright © Huawei Technologies Co., Ltd.

```
./CAMx_v7.10.36.12.20160610-11.MPICH3.job
```

The following figure shows an example of the command output, indicating that the program completes properly.



----End

1.24 CDO 1.9.8 Porting Guide (openEuler 20.03)

1.24.1 Introduction

The Climate Data Operators (CDO) software is a collection of many operators for standard processing of climate and numerical weather prediction (NWP) model output. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. CDO was developed to have the same set of processing functions for GRIB and NetCDF datasets in one package.

For more information about CDO, visit the **CDO official website**.

Programming language: C++

Brief description: meteorological data processing.

Open source license: GPL

Recommended Software Version

CDO 1.9.8

1.24.2 Environment Requirements

Hardware Requirements

Table 1-116 lists the hardware requirements.

Table 1-116 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-117 lists the software requirements.

Table 1-117 Software requirements

Item	Version	Download URL
CDO	1.9.8	https://code.mpimet.mpg.de/attachments/download/20826/cdo-1.9.8.tar.gz
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
HDF5	1.8.12-11	https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/h/hdf5-1.8.12-11.el7.aarch64.rpm
HDF	4.2.13-1	https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/h/hdf-devel-4.2.13-1.el7.aarch64.rpm
GRIB_API	1.12.3-5	https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/g/grib_api-1.12.3-5.el7.aarch64.rpm
NetCDF	4.3.3.1-5	https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/n/netcdf-devel-4.3.3.1-5.el7.aarch64.rpm

OS Requirements

Table 1-118 lists the OS requirements.

Table 1-118 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	https://repo.openeuler.org/openEuler-20.03-LTS-SP1/
Kernel	4.19	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

1.24.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CDO software porting.

Table 1-119 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	**/path/to/CDO**	Installation path of CDO.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	**/opt/compiler/BS**	Installation path of the BiSheng Compiler.	
4	**/path/to/hucx**	Installation path of Hyper MPI-HUCX.	
5	**/path/to/hmpi**	Installation path of Hyper MPI-Hyper MPI.	

1.24.4 Configuring the Compilation Environment

1.24.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-120 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Configuring the Yum source	See **1.24.4.2 Configuring the Yum Source**.
4	Installing the dependencies	See **1.24.4.3 Installing the Dependencies**.

1.24.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the following directory:

```
cd /etc/yum.repos.d/
```

Step 3 Obtain the RPM package of the EPEL repository.

```
wget  https://repo.huaweicloud.com/epel/epel-release-latest-7.noarch.rpm
```

Step 4 Perform the installation.

```
rpm  -ivh  --nodeps  epel-release-latest-7.noarch.rpm
```

Step 5 Replace **#baseurl** in the **epel.repo** file with **baseurl**.

```
sed -i "s/#baseurl/baseurl/g" /etc/yum.repos.d/epel.repo
```

Step 6 Refresh the Yum source cache.

```
yum clean all
yum makecache
```

Step 7 View the result.

```
yum repolist
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.24.4.3 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y nco netcdf netcdf-devel hdf hdf-devel grib_api grib_api-devel
```

NOTE

- If the RPM packages cannot be installed due to network issues, manually obtain the RPM packages. For details, see **1.24.8.1 Failed to Install an RPM Package Online Due to Network Problems**.
- If an error is reported when you install the dependencies using the **yum** command, see **1.24.8.2 Error Reported When Installing Dependency Packages with Yum**.

----End

1.24.5 Obtaining the Source Code

Procedure

Step 1 Create a program installation directory.

```
mkdir -p /path/to/CDO
```

Step 2 Obtain the CDO installation package.

https://code.mpimet.mpg.de/attachments/download/20826/cdo-1.9.8.tar.gz

Step 3 Use an SFTP tool to upload the CDO installation package to the */path/to/CDO* directory on the server.

----End

1.24.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation package directory.

```
cd /path/to/CDO
```

Step 3 Decompress the installation package.

```
tar -zxvf cdo-1.9.8.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cdo-1.9.8
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/CDO --with-netcdf=/usr --with-hdf5=/usr --with-grib_api=/usr CC=clang CXX=clang++ FC=flang
```

Step 6 Perform the compilation.

```
make -j 8
```

 Copyright © Huawei Technologies Co., Ltd.

```
Making all in contrib
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8/contrib'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8/contrib'
Making all in test/data
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8/test/data'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8/test/data'
Making all in test
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8/test'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8/test'
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8'
```

Step 7 Perform the installation.

```
make install
```

```
make[2]: Entering directory '/path/to/CDO/cdo-1.9.8/test/data'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/path/to/CDO/cdo-1.9.8/test/data'
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8/test/data'
Making install in test
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8/test'
make[2]: Entering directory '/path/to/CDO/cdo-1.9.8/test'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/path/to/CDO/cdo-1.9.8/test'
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8/test'
make[1]: Entering directory '/path/to/CDO/cdo-1.9.8'
make[2]: Entering directory '/path/to/CDO/cdo-1.9.8'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/path/to/CDO/cdo-1.9.8'
make[1]: Leaving directory '/path/to/CDO/cdo-1.9.8'
[root@host-10-208-179-187 cdo-1.9.8]# ls ../
```

Step 8 View the installation result.

```
ls /path/to/CDO/bin
```

```
[root@host-10-208-179-187 ~]# ls /path/to/CDO/bin
cdo
```

----End

1.24.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the outputted file **wrfout_d01_2019-11-27_00_00_00** to the test directory.

NOTE

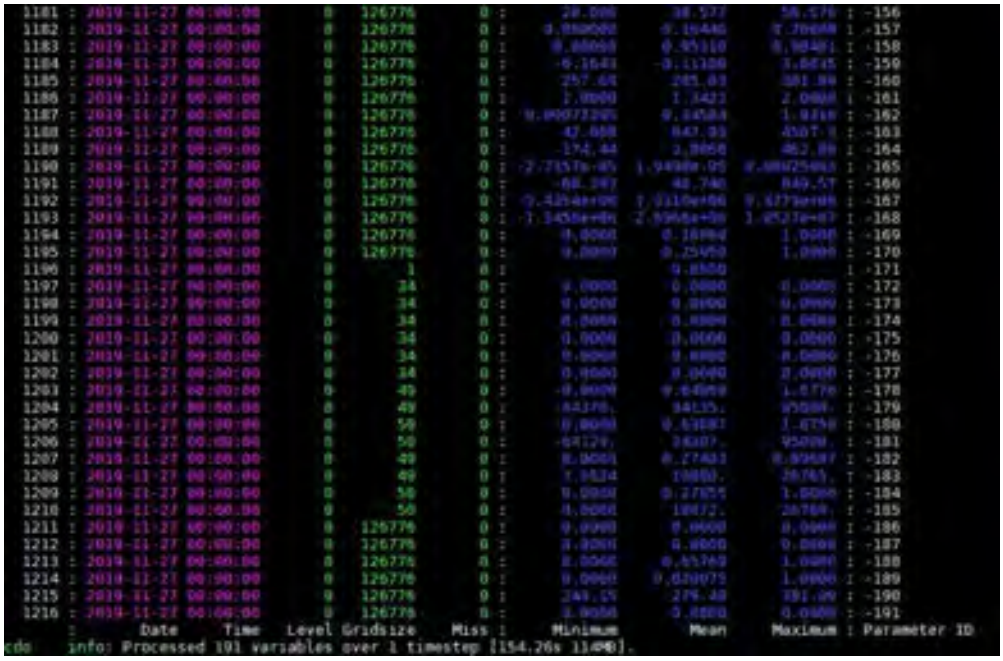
The **wrfout_d01_2019-11-27_00_00_00** file is the result file of the **conus12km** test case of WRF. For details, see **WRF 4.2 Porting Guide (openEuler 21.03)**.

Step 3 Verify the software enablement.

```
export PATH=/path/to/CDO/bin:$PATH
cdo info wrfout_d01_2019-11-27_00_00_00
```

 Copyright © Huawei Technologies Co., Ltd.

Information similar to the following is displayed:



----End

1.24.8 Troubleshooting

1.24.8.1 Failed to Install an RPM Package Online Due to Network Problems

Symptom

An RPM package fails to be installed online.

Key Process and Cause Analysis

A network exception occurs.

Conclusion and Solution

Manually obtain the RPM package. The following table lists the required RPM packages and their versions.

1.24.8.2 Error Reported When Installing Dependency Packages with Yum

Symptom

When using Yum to install dependency packages, the following error is reported:

Key Process and Cause Analysis

The required libraries are missing on openEuler. You can install the required dependency packages to solve this problem.

Conclusion and Solution

Based on the error message, run the following commands to obtain the RPM packages and install them:

- gsl
  wget http://mirror.centos.org/altarch/7/os/aarch64/Packages/gsl-1.15-13.el7.aarch64.rpm
  yum -y localinstall gsl-1.15-13.el7.aarch64.rpm
- libpng
  wget http://mirror.centos.org/centos/8-stream/AppStream/aarch64/os/Packages/libpng15-1.5.30- 7.el8.aarch64.rpm
  yum -y localinstall libpng15-1.5.30-7.el8.aarch64.rpm
- libfortran
  wget http://mirror.centos.org/centos/8-stream/AppStream/aarch64/os/Packages/compat-libgfortran-48-4.8.5-36.1.el8.aarch64.rpm
  yum -y localinstall compat-libgfortran-48-4.8.5-36.1.el8.aarch64.rpm
- jasper-libs
  wget http://mirror.centos.org/altarch/7/os/aarch64/Packages/jasper-libs-1.900.1-33.el7.aarch64.rpm
  yum -y localinstall jasper-libs-1.900.1-33.el7.aarch64.rpm
- jasper-devel
  wget http://mirror.centos.org/altarch/7/os/aarch64/Packages/jasper-devel-1.900.1-33.el7.aarch64.rpm
  yum -y localinstall jasper-devel-1.900.1-33.el7.aarch64.rpm

1.25 ecCodes 2.18.0 Porting Guide (openEuler 20.03)

 Copyright © Huawei Technologies Co., Ltd.

1.25.1 Introduction

ecCodes is a package developed by ECMWF which provides an application programming interface and a set of tools for decoding and encoding messages in the following formats:

- WMO FM-92 GRIB edition 1 and edition 2
- WMO FM-94 BUFR edition 3 and edition 4
- WMO GTS abbreviated header (only decoding)

A useful set of command line tools provide quick access to the messages. C, Fortran 90 and Python interfaces provide access to the main ecCodes functionality.

ecCodes is an evolution of GRIB-API. It is designed to provide the user with a simple set of functions to access data from several formats with a key/value approach.

For more information about ecCodes, visit the **ecCodes official website**.

Programming languages: C/Fortran

Brief description: A package that provides an application programming interface and a set of tools for decoding and encoding messages in the WMO GRIB and BUFR formats.

Open source license: Apache 2.0

Recommended Software Version

ecCodes 2.18.0

1.25.2 Environment Requirements

Hardware Requirements

Table 1-121 lists the hardware requirements.

Table 1-121 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-122 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-122 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
CMake	3.16.5	**https://cmake.org/files/v3.16/cmake-3.16.5.tar.gz**
ecCodes	2.18.0	**https://confluence.ecmwf.int/download/attachments/45757960/Eccodes-2.18.0-Source.tar.gz**

OS Requirements

Table 1-123 lists the OS requirements.

Table 1-123 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	**https://repo.openeuler.org/openEuler-20.03-LTS-SP1/**
Kernel	4.19	Included in the OS image

1.25.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ecCodes software porting.

Table 1-124 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler/BS**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/install/hucx** **/path/to/install/hmpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
3	**/path/to/CMAKE**	Test path of CMake.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	**/path/to/ECCODES**	Installation path of ecCodes.	

1.25.4 Configuring the Compilation Environment

1.25.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-125 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Configuring the Yum Source	See **1.25.4.2 Configuring the Yum Source**.
4	Installing CMake	See **1.25.4.3 Installing CMake**.

1.25.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether Yum is configured.

```
yum list
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-140-47 refrac]#
[root@host-10-208-140-47 refrac]# yum list
Last metadata expiration check: 1:33:20 ago on Mon 25 Oct 2021 03:20:02 PM CST.
Installed Packages
CUnit.aarch64                                                  2.1.3-22.oe1
GConf2.aarch64                                                 3.2.6-24.oe1
GeoIP.aarch64                                                  1.6.12-5.oe1
GeoIP-GeoLite-data.noarch                                      2018.06-3.oe1
ModemManager-glib.aarch64                                      1.14.8-2.oe1
NetworkManager.aarch64                                         1:1.26.2-4.oe1
NetworkManager-config-server.noarch                            1:1.26.2-4.oe1
NetworkManager-help.noarch                                     1:1.26.2-4.oe1
NetworkManager-libnm.aarch64                                   1:1.26.2-4.oe1
PackageKit.aarch64                                             1.1.12-9.oe1
abattis-cantarell-fonts.noarch                                 0.201-1.oe1
acl.aarch64                                                    2.2.53-8.oe1
adcli.aarch64                                                  0.9.0-1.oe1
adobe-mappings-cmap.noarch                                     20190730-2.oe1
adobe-mappings-cmap-lang.noarch                                20190730-2.oe1
adobe-mappings-pdf.noarch                                      20190401-1.oe1
adwaita-icon-theme.noarch                                      3.37.2-2.oe1
alsa-lib.aarch64                                               1.2.3-1.oe1
apr.aarch64                                                    1.7.0-2.oe1
apr-util.aarch64                                               1.6.1-12.oe1
asciidoc.noarch                                                8.6.10-3.oe1
```

NOTE

The Yum source is configured by default on openEuler 20.03 (LTS SP1) and does not need to be configured additionally.

Step 3 Install the base dependencies.

```
yum install -y gcc gcc-c++ m4 wget openssl-devel git
```

----End

1.25.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar -zxvf cmake-3.16.5.tar.gz
```

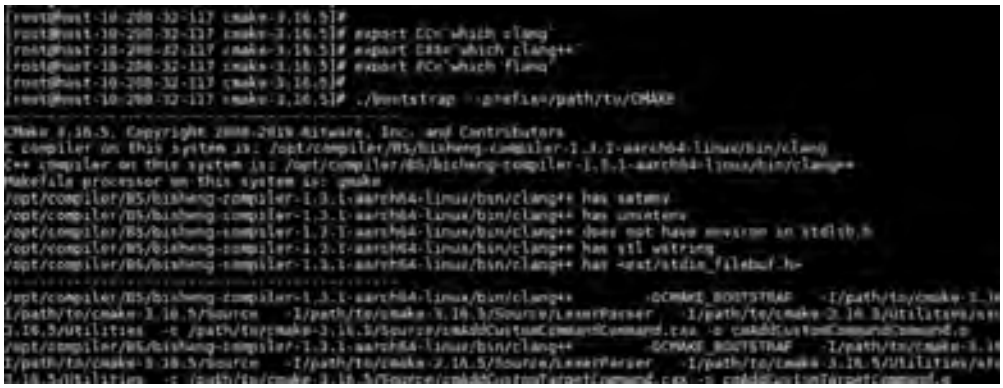
Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.16.5
```

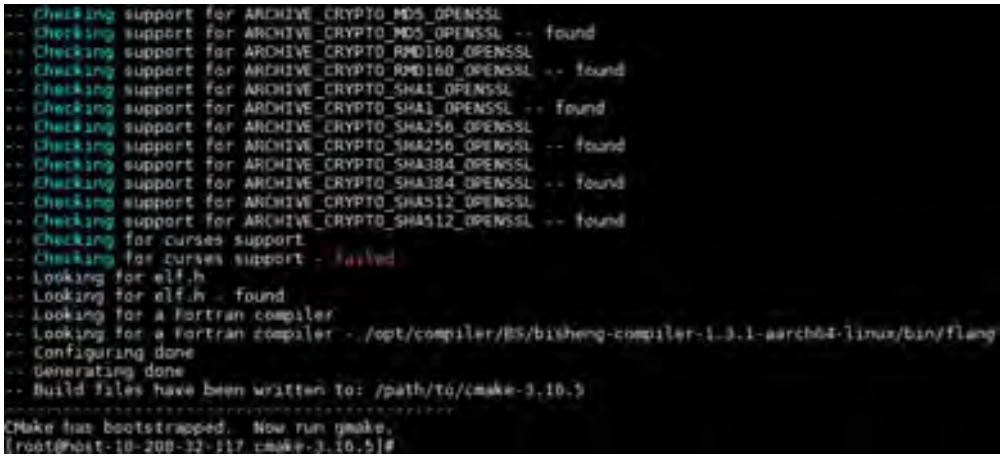
Step 4 Perform the configuration.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
./bootstrap --prefix=/path/to/CMAKE
```

Output for the start:



Output for the end:



Step 5 Perform the installation.

```
make
make install
```

```
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/ApplicationIcon.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Logo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SplashScreen.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Windows_TemporaryKey.pfx
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo44x44.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/StoreLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CPack.GenericLicense.txt
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CTestScript.cmake.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/AppleInfo.plist
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/TestDriver.cxx.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/emacs/cmake-mode.el
-- Installing: /path/to/CMAKE/share/aclocal/cmake.m4
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cmake
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cpack
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/ctest
[root@host-10-208-32-117 cmake-3.16.5]#
```

Step 6 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 7 Check the version after the installation.

```
cmake --version
```

```
[root@host-10-208-32-117 to]# cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
[root@host-10-208-32-117 to]# CMAKE/bin/cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.25.5 Obtaining the Source Code

Procedure

Step 1 Create the ecCodes installation directory */path/to/ECCODES*.

```
mkdir -p /path/to/ECCODES
```

Step 2 Go to the **ECCODES** directory.

```
cd /path/to/ECCODES
```

Step 3 Download the ecCodes installation package.

https://confluence.ecmwf.int/download/attachments/45757960/Eccodes-2.18.0-Source.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the ecCodes installation package.

wget https://confluence.ecmwf.int/download/attachments/45757960/Eccodes-2.18.0-Source.tar.gz --no-check-certificate

Step 4 Use an SFTP tool to upload the ecCodes installation package to the */path/to/ECCODES* directory on the server.

----End

1.25.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the ecCodes installation package directory.

```
cd /path/to/ECCODES
```

Step 3 Install the dependencies.

```
yum install -y openjpeg-devel python-devel numpy
```

Step 4 Decompress the ecCodes installation package.

```
tar -zxvf eccodes-2.18.0-Source.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd eccodes-2.18.0-Source
```

Step 6 Create a **build** directory.

```
mkdir build
cd build
```

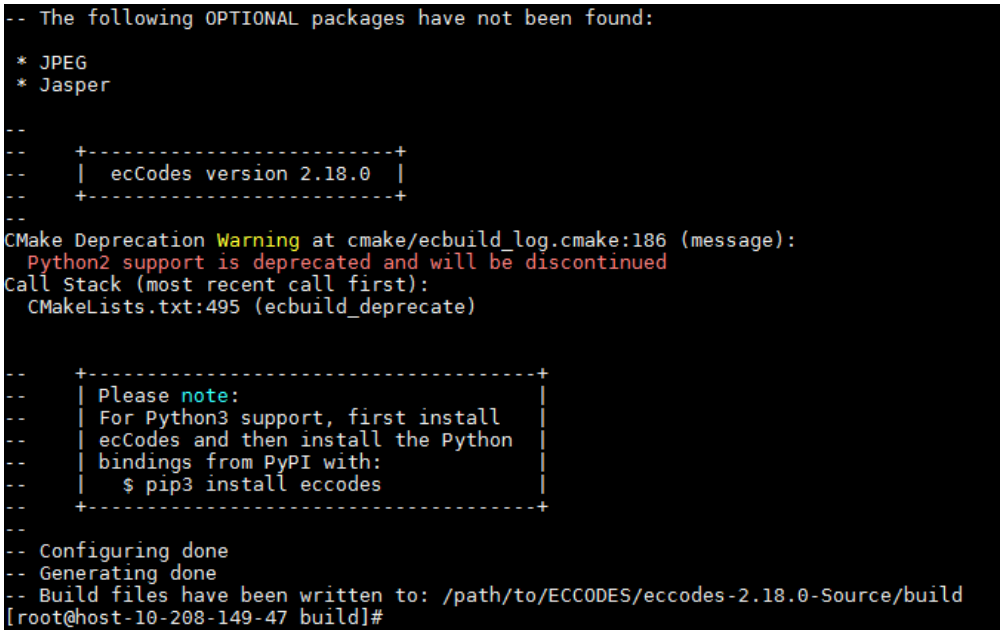
Step 7 Configure the pre-compilation environment.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
cmake ../../eccodes-2.18.0-Source -DCMAKE_INSTALL_PREFIX=/path/to/ECCODES
```

Output for the start:

 Copyright © Huawei Technologies Co., Ltd.

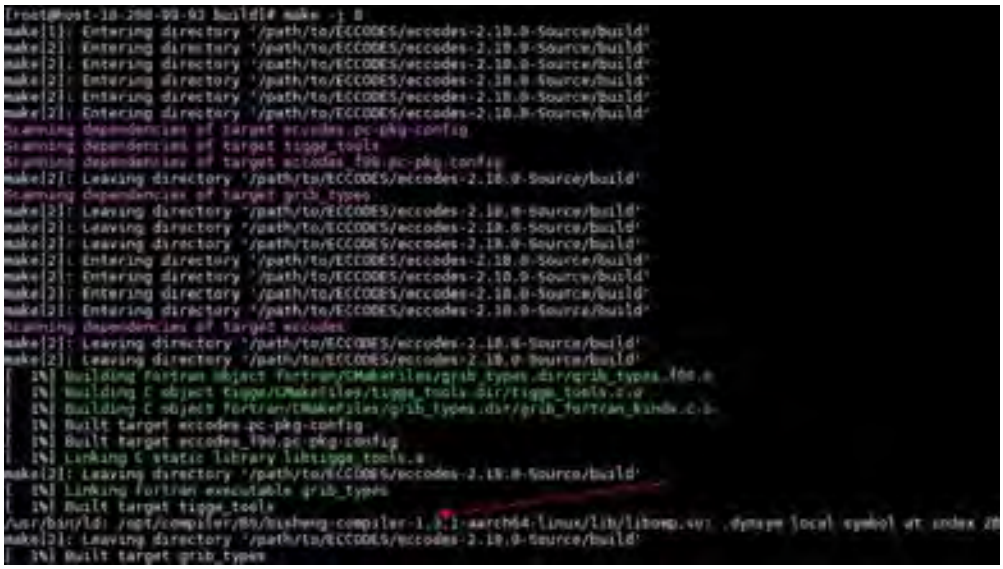
Output for the end:



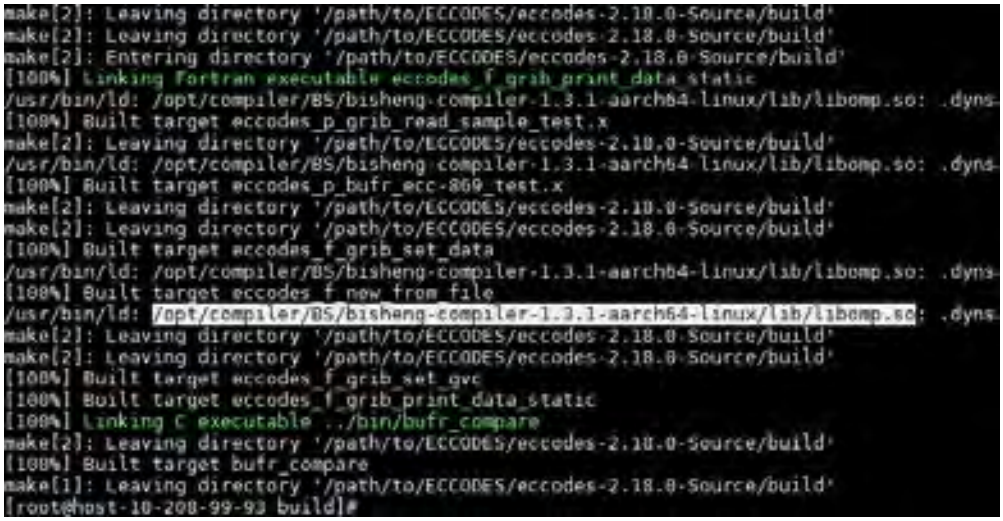
Step 8 Perform the compilation.

```
make -j 8
```

Output for the start:



Output for the end:



Step 9 Perform the test.

```
ctest
```

```
      Start 28: eccodes_c_new_sample
28/35 Test #28: eccodes_c_new_sample ...................   Passed    0.01 sec
      Start 29: eccodes_f_grib_set_pv
29/35 Test #29: eccodes_f_grib_set_pv ..................   Passed    0.01 sec
      Start 30: eccodes_f_grib_set_data
30/35 Test #30: eccodes_f_grib_set_data ................   Passed    0.01 sec
      Start 31: eccodes_f_grib_ecc-671
31/35 Test #31: eccodes_f_grib_ecc-671 .................   Passed    0.01 sec
      Start 32: eccodes_p_grib_set_pv_test
32/35 Test #32: eccodes_p_grib_set_pv_test .............   Passed    0.12 sec
      Start 33: eccodes_p_grib_read_sample_test
33/35 Test #33: eccodes_p_grib_read_sample_test ........   Passed    0.20 sec
      Start 34: eccodes_p_bufr_read_sample_test
34/35 Test #34: eccodes_p_bufr_read_sample_test ........   Passed    0.21 sec
      Start 35: eccodes_p_bufr_ecc-869_test
35/35 Test #35: eccodes_p_bufr_ecc-869_test ............   Passed    0.14 sec

100% tests passed, 0 tests failed out of 35

Label Time Summary:
eccodes       =  14.53 sec*proc (35 tests)
executable    =   0.01 sec*proc (1 test)
script        =  14.51 sec*proc (34 tests)

Total Test time (real) =  14.54 sec
[root@host-10-208-149-47 build]#
```

Step 10 Perform the installation.

```
make install
```

```
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/regular_ll_pl_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/regular_ll_pl_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/regular_ll_sfc_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/regular_ll_sfc_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_ml_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_ml_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_pl_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_pl_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_sfc_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_gg_sfc_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_ll_pl_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_ll_pl_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_ll_sfc_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/rotated_ll_sfc_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_ml_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_ml_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_pl_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_pl_grib2.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_sfc_grib1.tmpl
-- Installing: /path/to/ECCODES/share/eccodes/ifs_samples/grib1_mlgrib2_ieee64/sh_sfc_grib2.tmpl
[root@host-10-208-99-93 build]#
```

Step 11 Set the environment variables.

```
export PATH=/path/to/ECCODES/bin:$PATH
```

----End

1.25.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the running and verification.

```
codes_info
```

```
[root@host-10-208-99-93 build]# codes_info

ecCodes Version 2.18.0

Default definition files path is used: /path/to/ECCODES/share/eccodes/definitions
Definition files path can be changed by setting the ECCODES_DEFINITION_PATH environment variable.

Default SAMPLES path is used: /path/to/ECCODES/share/eccodes/samples
SAMPLES path can be changed by setting the ECCODES_SAMPLES_PATH environment variable.

[root@host-10-208-99-93 build]#
```

Step 3 Verify the running and obtain the running time.

```
time codes_info
```

NOTE

The **real** time (unit: s) in the return result indicates the time. A smaller value indicates higher performance.

----End

1.26 ecFlow 5.5.2 Porting Guide (openEuler 20.03)

1.26.1 Introduction

ecFlow is a client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment. It provides reasonable tolerance for hardware and software failures, combined with restart capabilities.

ecFlow runs as a server receiving requests from clients. The command line interface, the graphical interface, scripts and the Python API (application interface) are the clients. The server is based on C++/boost ASIO and uses TCP/IP for communication. Multiple servers can be run on the same hardware. ecFlow submits tasks (jobs) and receives acknowledgements from tasks via specific commands embedded in the scripts. The relationship between tasks is stored in ecFlow, and it is able to submit tasks dependent on the status of other tasks and attributes like time.

The command line interface for ecFlow allows the suite definition to be loaded and retrieved from the server. Also it provides a rich set of commands for communication with the server.

The Python API allows the entire suite definition structure to be specified and loaded into the server. A suite is a collection of interrelated tasks. In ecFlow suites are described by a definition file. The Python API also provides functionality for client to server communication. In addition, it allows checking of the suite, testing the defined interrelations between tasks, and other references and limits.

For more information about ecFlow, visit the **ecFlow official website**.

Programming language: C++

Brief description: A client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: Apache 2.0

Recommended Software Version

ecFlow 5.5.2

1.26.2 Environment Requirements

Hardware Requirements

Table 1-126 lists the hardware requirements.

Table 1-126 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-127 lists the software requirements.

Table 1-127 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
CMake	3.16.5	https://cmake.org/files/v3.16/cmake-3.16.5.tar.gz
Boost	1.72.0	https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz
ecFlow	5.5.2	https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz

OS Requirements

Table 1-128 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-128 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	**https://repo.openeuler.org/openEuler-20.03-LTS-SP1/**
Kernel	4.19	Included in the OS image

1.26.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ecFlow software porting.

Table 1-129 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler/BS**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/install/hucx** **/path/to/install/hmpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/CMAKE**	Test path of CMake.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	**/path/to/ECFLOW_BUILD**	Installation path of ecFlow and Boost.	

1.26.4 Configuring the Compilation Environment

1.26.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 1-130 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Configuring the Yum Source	See **1.26.4.2 Configuring the Yum Source**.
4	Installing CMake	See **1.26.4.3 Installing CMake**.

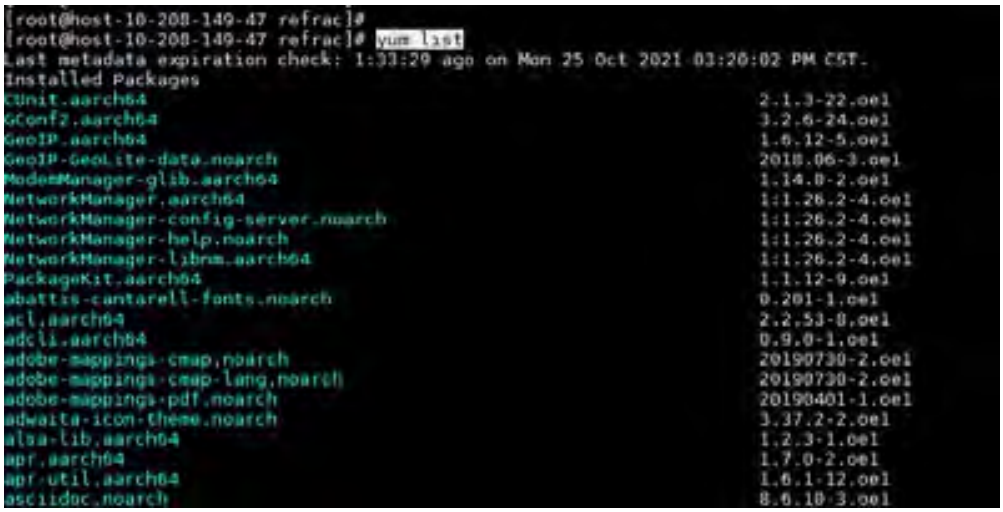
1.26.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether Yum is configured.

```
yum list
```



NOTE

The Yum source is configured by default on openEuler 20.03 (LTS SP1) and does not need to be configured additionally.

Step 3 Install the base dependencies.

```
yum install -y gcc gcc-c++ m4 wget openssl-devel git
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.26.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar -zxvf cmake-3.16.5.tar.gz
```

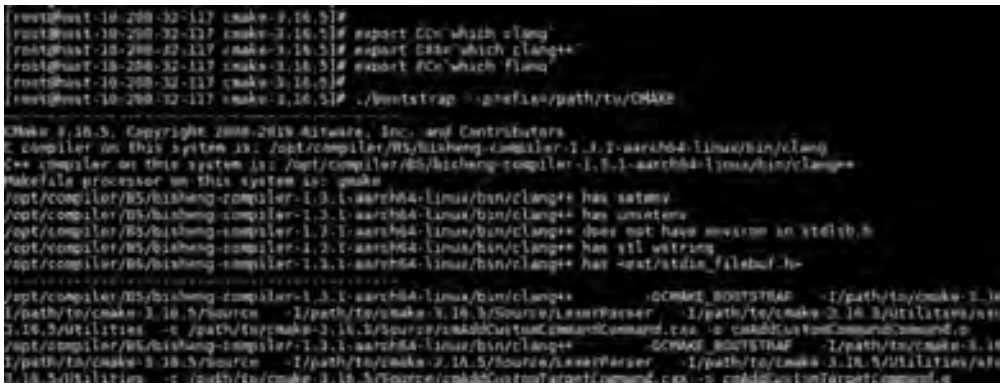
Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.16.5
```

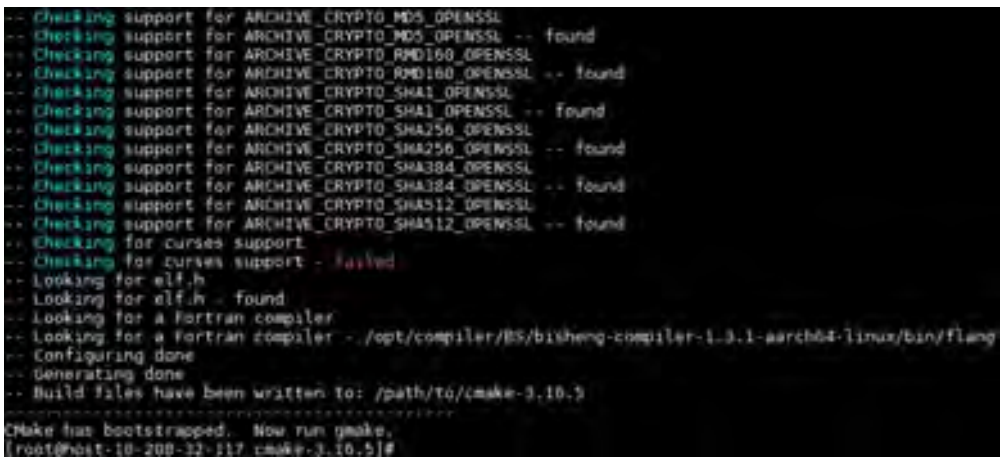
Step 4 Perform the configuration.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
./bootstrap --prefix=/path/to/CMAKE
```

Output for the start:



Output for the end:



Step 5 Perform the installation.

```
make
make install
```

```
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/ApplicationIcon.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Logo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SplashScreen.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Windows_TemporaryKey.pfx
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo44x44.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/StoreLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CPack.GenericLicense.txt
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CTestScript.cmake.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/AppleInfo.plist
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/TestDriver.cxx.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/emacs/cmake-mode.el
-- Installing: /path/to/CMAKE/share/aclocal/cmake.m4
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cmake
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cpack
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/ctest
[root@host-10-208-32-117 cmake-3.16.5]#
```

Step 6 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 7 Check the version after the installation.

```
cmake --version
```

```
[root@host-10-208-32-117 to]# cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
[root@host-10-208-32-117 to]# CMAKE/bin/cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
```

----End

1.26.5 Obtaining the Source Code

Procedure

Step 1 Create the ecFlow installation directory */path/to/ECFLOW_BUILD*.

```
mkdir -p /path/to/ECFLOW_BUILD
```

Step 2 Go to the **ECFLOW_BUILD** directory.

```
cd /path/to/ECFLOW_BUILD
```

Step 3 Download the ecFlow installation package.

https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the ecFlow installation package.

wget https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz --no-check-certificate

Step 4 Download the Boost installation package **boost_1_72_0.tar.gz**.

 Copyright © Huawei Technologies Co., Ltd.

https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the Boost installation package.

wget https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz --no-check-certificate

Step 5 Use an SFTP tool to upload the ecFlow and Boost installation packages to the */path/to/ECFLOW_BUILD* directory on the server.

----End

1.26.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ecFlow installation package.

```
tar -xvf ecFlow-5.5.2-Source.tar.gz
```

Step 3 Decompress the Boost installation package.

```
tar -xvf boost_1_72_0.tar.gz
```

Step 4 Create two environment variables.

```
export WK=/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source
export BOOST_ROOT=/path/to/ECFLOW_BUILD/boost_1_72_0
```

Step 5 Go to the Boost installation directory and build the Boost library.

```
cd $BOOST_ROOT
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
./bootstrap.sh
./b2 install --prefix=/path/to/ECFLOW_BUILD/boost_1_72_0
```

Step 6 Compile the Boost library using ecFlow.

```
$WK/build_scripts/boost_build.sh
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Go to the directory generated after the ecFlow decompression.

```
cd $WK
```

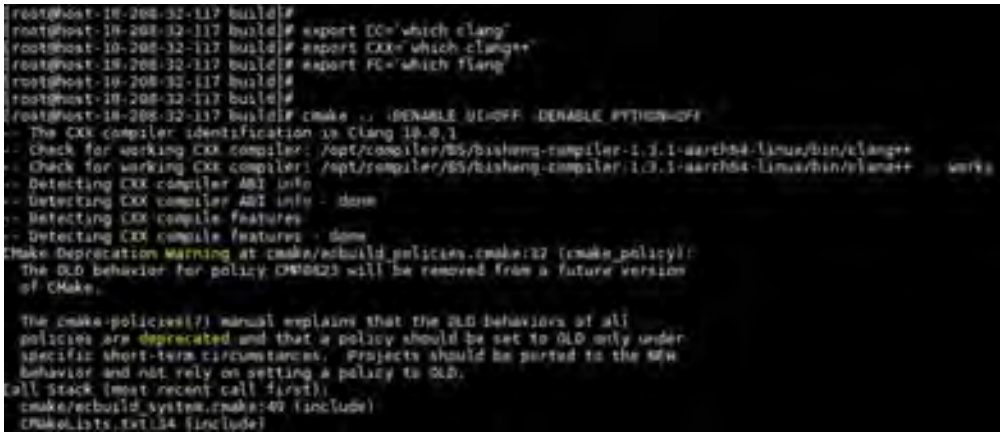
Step 8 Create an independent directory for ecFlow.

```
mkdir build
cd build
```

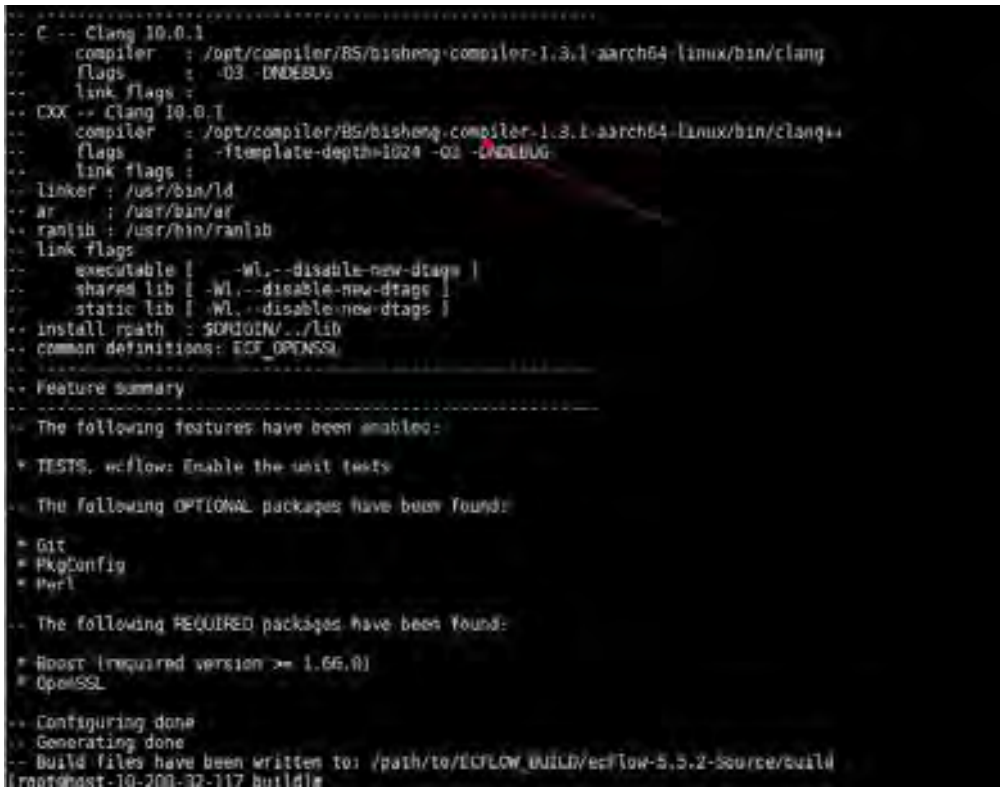
Step 9 Use the BiSheng Compiler and run the following **cmake** command:

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
cmake .. -DENABLE_UI=OFF -DENABLE_PYTHON=OFF
```

Output for the start:

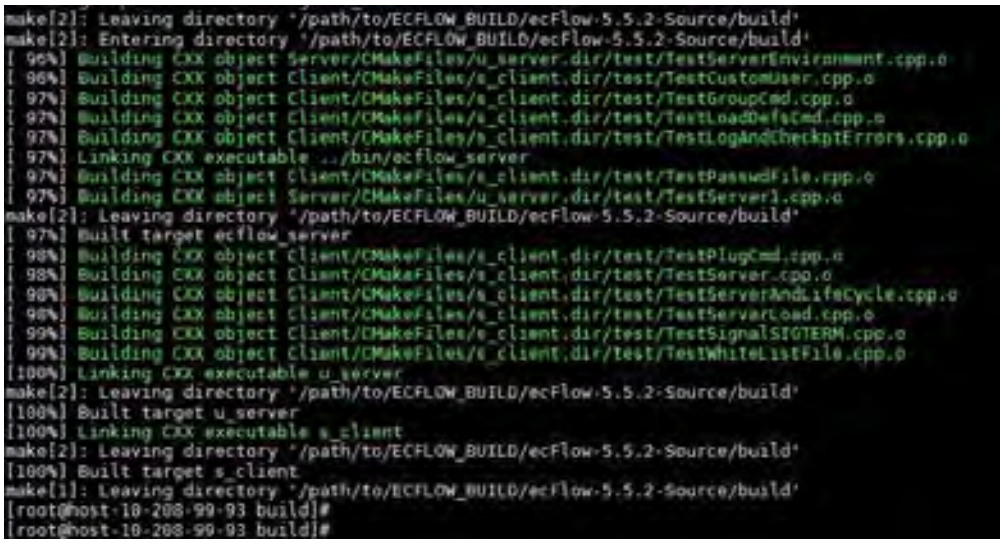


Output for the end:



Step 10 Perform the compilation.

```
make -j 8
```



Step 11 Perform the installation.

```
make install
```

```
[ 98%] Built target libserver
make[2]: Entering directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
make[2]: Leaving directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
[ 98%] Built target ecflow_server
make[2]: Entering directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
make[2]: Leaving directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
[ 99%] Built target u_server
make[2]: Entering directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
make[2]: Leaving directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
[100%] Built target libharness
make[1]: Leaving directory '/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build'
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/lib/cmake/ecflow/ecflow-config-version.cmake
-- Installing: /usr/local/lib/cmake/ecflow/ecflow-config.cmake
-- Installing: /usr/local/lib/cmake/ecflow/ecflow-targets.cmake
-- Installing: /usr/local/lib/cmake/ecflow/ecflow-targets-release.cmake
-- Installing: /usr/local/bin/ecflow_client
-- Up-to-date: /usr/local/bin/ecflow_client
-- Installing: /usr/local/bin/ecflow_standalone
-- Installing: /usr/local/bin/ecflow_fuse.py
-- Installing: /usr/local/bin/ecflow_logsvr.pl
-- Installing: /usr/local/bin/ecflow_start.sh
-- Installing: /usr/local/bin/ecflow_stop.sh
-- Installing: /usr/local/bin/ecflow_logserver.sh
-- Installing: /usr/local/bin/ecflow_server
-- Up-to-date: /usr/local/bin/ecflow_server
[root@host-10-208-99-93 build]#
```

----End

1.26.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build
```

Step 3 Perform the test.

```
make test
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-32-117 build]#
[root@host-10-208-32-117 build]# make test
Running tests...
Test project /path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build
    Start 1: u_acore
1/8 Test #1: u_acore ..........................   Passed    0.16 sec
    Start 2: u_anattr
2/8 Test #2: u_anattr .........................   Passed    0.30 sec
    Start 3: u_anode
3/8 Test #3: u_anode ..........................   Passed    0.21 sec
    Start 4: u_aparser
4/8 Test #4: u_aparser ........................   Passed    0.44 sec
    Start 5: u_base
5/8 Test #5: u_base ...........................   Passed    0.09 sec
    Start 6: c_csim
6/8 Test #6: c_csim ...........................   Passed    6.47 sec
    Start 7: s_client
7/8 Test #7: s_client .........................   Passed   97.35 sec
    Start 8: u_server
8/8 Test #8: u_server .........................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 8

Label Time Summary:
ecflow        = 105.04 sec*proc (8 tests)
executable    = 105.04 sec*proc (8 tests)

Total Test time (real) = 105.05 sec
[root@host-10-208-32-117 build]#
```

Step 4 Perform the test and obtain the test time.

```
time make test
```

```
[root@host-10-208-32-117 build]# time make test
Running tests...
Test project /path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build
    Start 1: u_acore
1/8 Test #1: u_acore ..........................   Passed    0.13 sec
    Start 2: u_anattr
2/8 Test #2: u_anattr .........................   Passed    0.24 sec
    Start 3: u_anode
3/8 Test #3: u_anode ..........................   Passed    0.12 sec
    Start 4: u_aparser
4/8 Test #4: u_aparser ........................   Passed    0.18 sec
    Start 5: u_base
5/8 Test #5: u_base ...........................   Passed    0.09 sec
    Start 6: c_csim
6/8 Test #6: c_csim ...........................   Passed    6.42 sec
    Start 7: s_client
7/8 Test #7: s_client .........................   Passed   97.34 sec
    Start 8: u_server
8/8 Test #8: u_server .........................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 8

Label Time Summary:
ecflow        = 104.49 sec*proc (8 tests)
executable    = 104.49 sec*proc (8 tests)

Total Test time (real) = 104.50 sec

real    1m44.510s
user    0m7.345s
sys     0m3.800s
[root@host-10-208-32-117 build]#
```

NOTE

The **real** time (unit: s) in the return result indicates the time. A smaller value indicates higher performance.

----End

1.27 MITgcm c67o Porting Guide (openEuler 21.03)

1.27.1 Introduction

MIT General Circulation Model (MITgcm) is a numerical model designed for study of the atmosphere, ocean, and climate. MITgcm's flexible non-hydrostatic formulation enables it to efficiently simulate fluid phenomena over a wide range of scales. Its adjoint capabilities enable it to be applied to sensitivity questions and to parameter and state estimation problems. By employing fluid equation isomorphisms, a single dynamical kernel can be used to simulate flow of both the atmosphere and ocean.

For more information about MITgcm, visit the **MITgcm official website**.

Programming languages: C/Fortran 90

Brief description: Modeling software for simulating large-scale ocean circulation.

Open source license: GPL 3.0

Recommended Software Version

MITgcm_c67o

1.27.2 Environment Requirements

Hardware Requirements

Table 1-131 lists the hardware requirements.

Table 1-131 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-132 lists the software requirements.

Table 1-132 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
MITgcm	4.1	**http://mitgcm.org/download/**
NetCDF-C	4.7.0	**https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0**
NetCDF-Fortran	4.4.5	**https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5**
HDF5	1.10.1	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz**
Test cases	Exp4	Provided by the software.

OS Requirements

Table 1-133 lists the OS requirements.

Table 1-133 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://www.openeuler.org/zh/download**
Kernel	4.17	Included in the OS image

1.27.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MITgcm software porting.

Table 1-134 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/BISHENG**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/HMPI**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/HDF5**	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
4	**/path/to/NETCDF**	Installation path of NetCDF-C.	document are examples only. Use the actual paths planned during the installation process.
		Installation path of NetCDF-Fortran.	
5	**/path/to/MITGCM**	Installation path of MITgcm.	

1.27.4 Configuring the Compilation Environment

1.27.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-135 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **1.27.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.27.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.27.4.4 Installing NetCDF-Fortran**.

NOTE

When installing the BiSheng Compiler, you need to specify the compilation parameters.

CC=clang CXX=clang++ FC=flang

 Copyright © Huawei Technologies Co., Ltd.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

1.27.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

NOTE

If the server has multiple MPIs, **CC**, **FC**, and **CXX** in the preceding command can be represented using absolute paths.

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify line 11835 in the file.
   ```
   wl="-Wl,"
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

 Copyright © Huawei Technologies Co., Ltd.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.27.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the installation.

```
make
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.27.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-foortran-4.4.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-
parallel-tests --enable-large-file-tests
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.27.5 Obtaining the Source Code

Procedure

Step 1 Download the MITgcm installation package.

http://mitgcm.org/download/

Step 2 Use an SFTP tool to upload the MITgcm installation package to the */path/to/MITGCM* directory on the server.

----End

1.27.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the MITgcm installation package directory.

```
cd /path/to/MITGCM
```

Step 3 Decompress the installation package.

```
tar -xvf MITgcm_c67o.tar.gz
cd MITgcm
```

Step 4 Configure the related files.

1. Go to the **build_options** directory.
   ```
   cd tools/build_options
   ```
2. Create and edit the **Linux_arm64_gfortran** file.
   ```
   vi linux_arm64_gfortran
   ```
3. Press **i** to enter the insert mode and add the following content:
   ```
   #!/bin/bash
   MPI='true'
   CC=mpicc
   FC=mpif77
   F90C=mpif90
   DEFINES='-DWORDLENGTH=4 -DNML_TERMINATOR -DALLOW_USE_MPI -DALWAYS_USE_MPI'
   EXTENDED_SRC_FLAG='-ffixed-line-length-132'
   F90FIXEDFORMAT='-ffixed-form'
   GET_FC_VERSION="--version"
   OMPFLAG='-fopenmp'
   NOOPTFLAGS='-O0 -g'
   NOOPTFILES=''
   CFLAGS='-O3 '
   FFLAGS="$FFLAGS -fconvert=big-endian -fimplicit-none"
   CFLAGS="$CFLAGS -fPIC"
   FFLAGS="$FFLAGS -fPIC"
   #- with FC 19, need to use this without -fPIC (which cancels -mcmodel option):
   # CFLAGS="$CFLAGS -mcmodel=medium"
   # FFLAGS="$FFLAGS -mcmodel=medium"
   #- might want to use '-fdefault-real-8' for fizhi pkg:
   #FFLAGS="$FFLAGS -fdefault-real-8 -fdefault-double-8"
   if test "x$IEEE" = x ; then #- with optimisation:
   #- full optimisation
   FOPTIM='-O3 -funroll-loops'
   NOOPTFILES="$NOOPTFILES ini_masks_etc.F"
   #- can use -O2 (safe optimisation) to avoid Pb with some gcc version of -O3:
   #FOPTIM='-O2 -funroll-loops'
   else
   # these may also be useful, but require specific gfortran versions:
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
# -Wnonstd-intrinsics for gfortran <= 4.3
# -Wintrinsics-std for gfortran >= 4.4
# -Wno-tabs for gfortran >= 4.3
# -Wno-unused-dummy-argument for gfortran >= 4.6
#FFLAGS="$FFLAGS -Waliasing -Wampersand -Wsurprising -Wline-truncation"
#- or simply:
FFLAGS="$FFLAGS -Wall"
#- to get plenty of warnings: -Wall -Wextra (older form: -Wall -W) or:
#FFLAGS="$FFLAGS -Wconversion -Wimplicit-interface -Wunused-labels"
if test "x$DEVEL" = x ; then #- no optimisation + IEEE :
FOPTIM='-O0'
else #- development/check options:
FOPTIM='-O0 -g -fbounds-check'
FOPTIM="$FOPTIM -ffpe-trap=invalid,zero,overflow -finit-real=inf"
fi
fi
F90FLAGS=$FFLAGS
F90OPTIM=$FOPTIM
# --------------------------
# NETCDF DIRECTORIES
# --------------------------
HDF5_HOME=/path/to/HDF5
NETCDF_HOME=/path/to/NETCDF
INCLUDEDIRS=''
INCLUDES="-I$NETCDF_HOME/include -I$HDF5_HOME/include"
LIBS="-L$NETCDF_HOME/lib -lnetcdff -lnetcdf -L$HDF5_HOME/lib -lhdf5_hl -lhdf5"
# ----------------------
# INCLUDE MPI DIRECTORIES
# ----------------------
MPI_HOME=/path/to/OPENMPI-4.0.1
MPI_INC_DIR=$MPI_HOME/include
INCLUDES="$INCLUDES -I$MPI_HOME/include"
INCLUDEDIRS="$INCLUDEDIRS $MPI_HOME/include"
MPIINCLUDEDIR="$MPI_INC_DIR"
```

4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 The following uses the **exp4** test case as an example to describe how to compile and install the software.

1. Go to the related directory.
   cd */path/to/MITGCM*/**MITgcm/verification/exp4/code**
2. Rename **SIZE.h_mpi** to **SIZE.h** to replace the original **SIZE.h** file.
   mv SIZE.h_mpi SIZE.h
3. Go to the related directory.
   cd */path/to/MITGCM*/**MITgcm/verification/exp4/build**
4. Create a **makefile** file.
   ../../../tools/genmake2 -mods=../code -mpi -of=../../../tools/build_options/linux_arm64_gfortran
5. Perform the compilation and installation. If the **mitgcmuv** file is generated, the compilation is successful.
   make depend
   make

----End

1.27.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.
cd */path/to/MITGCM*/**MITgcm/verification/exp4/run**

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Copy the test case to the current directory.

```
cp ../input/* ./
cp ../build/mitgcmuv ./
```

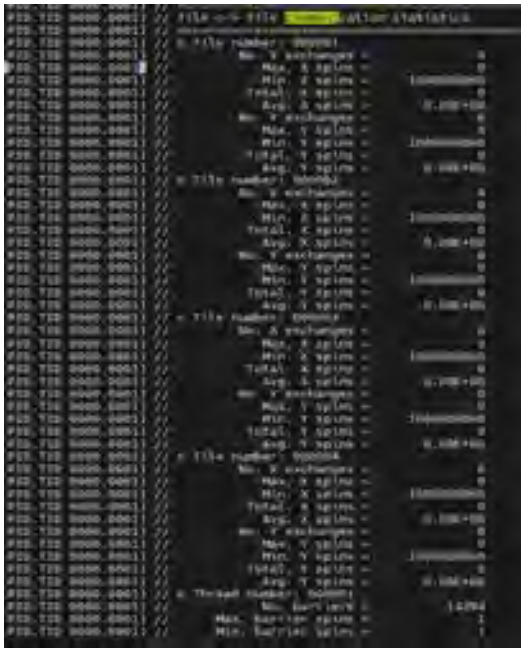
Step 4 Start the test.

```
time mpirun -n 4 --allow-run-as-root ./mitgcmuv
```

Step 5 View the **STDOUT.0000** output file.

```
vi STDOUT.0000
```

The following figure shows the output:



----End

1.27.8 Troubleshooting

1.27.8.1 After a Dependency Package Is Configured, an Error Message Is Displayed Indicating that the Dynamic Link Library Cannot Be Found

Symptom

After a dependency package is configured, an error message is displayed indicating that the dynamic link library cannot be found.

Key Process and Cause Analysis

The dynamic link library is not loaded in the environment variables.

Conclusion and Solution

Check the environment variables. The NetCDF path does not have the corresponding library. The NetCDF-C path is separated from the NetCDF-Fortran path. As a result, the NetCDF-Fortran library function cannot be found. Therefore, you need to reload the environment variables. If the program to be compiled uses Fortran and C at the same time, place the two dependency packages in the same directory.

1.28 NCEPLIBS 1.2.0 Porting Guide (openEuler 20.03)

1.28.1 Introduction

NCEPLIBS is a collection of libraries and utilities that are used by the weather model and other applications.

For more information about NCEPLIBS, visit the **NCEPLIBS official website**.

Programming languages: C/Fortran

Brief description: common dependency libraries in the meteorological field.

Open source license: Apache

Recommended Software Version

NCEPLIBS v1.2.0

1.28.2 Environment Requirements

Hardware Requirements

Table 1-136 lists the hardware requirements.

Table 1-136 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-137 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-137 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
HDF5	1.10.5	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz**
PnetCDF	1.11.2	**http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz**
NetCDF-C	4.7.0	**https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.0.tar.gz**
NetCDF-Fortran	4.4.5	**https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.5.tar.gz**
CMake	3.16.5	**https://cmake.org/files/v3.16/cmake-3.16.5.tar.gz**
NCEPLIBS	v1.2.0	**https://github.com/NOAA-EMC/NCEPLIBS/archive/refs/tags/v1.2.0.tar.gz**

OS Requirements

Table 1-138 lists the OS requirements.

Table 1-138 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	**https://repo.openeuler.org/openEuler-20.03-LTS-SP1/**
Kernel	4.19	Included in the OS image

1.28.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NCEPLIBS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-139 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler/BS**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/install/hucx** **/path/to/install/hmpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/HDF5**	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	**/path/to/PNETCDF**	Installation path of PnetCDF.	
5	**/path/to/NETCDF**	Installation path of NetCDF-C. Installation path of NetCDF-Fortran.	
6	**/path/to/CMAKE**	Test path of CMake.	
7	**/path/to/NCEPLIBS**	Installation path of NCEPLIBS.	

1.28.4 Configuring the Compilation Environment

1.28.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-140 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Configuring the Yum Source	See **1.28.4.2 Configuring the Yum Source**.
4	Installing HDF5	See **1.28.4.3 Installing HDF5**.
5	Installing PnetCDF	See **1.28.4.4 Installing PnetCDF**.
6	Installing NetCDF-C	See **1.28.4.5 Installing NetCDF-C**.
7	Installing NetCDF-Fortran	See **1.28.4.6 Installing NetCDF-Fortran**.
8	Installing CMake	See **1.28.4.7 Installing CMake**.

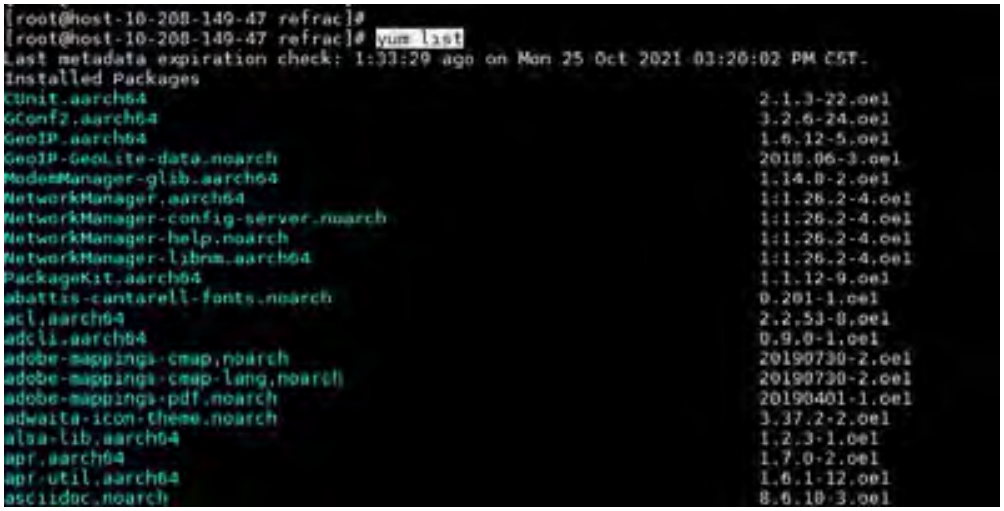
1.28.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether Yum is configured.

```
yum list
```



NOTE

The Yum source is configured by default on openEuler 20.03 (LTS SP1) and does not need to be configured additionally.

Step 3 Install the base dependencies.

```
yum install -y gcc gcc-c++ m4 wget openssl-devel git
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.28.4.3 Installing HDF5

NOTE

- If the **configure** file in the source packages of the HDF5 components specifies that the GNU software optimization level is **O3**, add the following optimization options before performing **Step 1**. For details, see **Step 4**.
  ```
  export optflags="-O3"
  export ldflags="-Wl,--build-id"
  export threads=$(nproc)
  ```
- If the **configure** file specifies that non-GNU software optimization is not required, you do not need to add optimization options.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -zxvf hdf5-1.10.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.5
```

Step 4 Perform the configuration.

```
CC=mpicc CFLAGS="$optflags -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FC=mpif90 FCFLAGS="$optflags -fPIC" FFLAGS="$optflags -fPIC" LDFLAGS="$ldflags" CXX=mpicxx ./configure --prefix=/path/to/HDF5 --enable-shared --enable-static --enable-fortran --enable-parallel --enable-optimization=high
```

```
                    Fortran: yes
                Fortran Compiler: /path/to/install/compi/bin/mpif90 ( Bisheng Compiler 1.3.1
g-0658acfa40ab))
                  Fortran Flags: -O3 -fPIC
               H5 Fortran Flags:
               AM Fortran Flags:
         Shared Fortran Library: yes
         Static Fortran Library: yes

                        C++: no

                       Java: no


Features:
---------
                  Parallel HDF5: yes
Parallel Filtered Dataset Writes: yes
                Large Parallel I/O: yes
                High-level library: yes
                      Threadsafety: no
               Default API mapping: v110
  With deprecated public symbols: yes
          I/O filters (external): deflate(zlib)
                             MPE:
                      Direct VFD: no
                         dmalloc: no
  Packages w/ extra debug output: none
                     API tracing: no
            Using memory checker: no
 Memory allocation sanity checks: no
          Function stack tracing: no
       Strict file format checks: no
    Optimization instrumentation: no
[root@host-10-208-84-248 hdf5-1.10.5]#
```

Step 5 Apply the patch package.

```
patch -p1 < libtool.patch
```

```
[root@host-10-208-84-248 hdf5-1.10.5]# vi libtool.patch
[root@host-10-208-84-248 hdf5-1.10.5]#
[root@host-10-208-84-248 hdf5-1.10.5]# patch -p1 < libtool.patch
patching file libtool
[root@host-10-208-84-248 hdf5-1.10.5]#
```

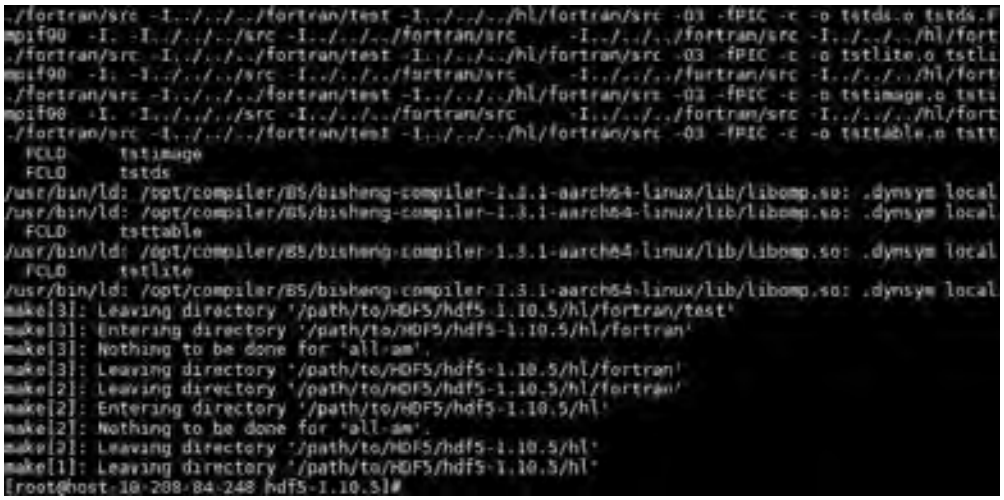
 Copyright © Huawei Technologies Co., Ltd.

Step 6 Create a **libtool.patch** file and edit the file.

1. Create a **libtool.patch** file.
   ```
   vi libtool.patch
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
   --- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
   +++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
   @@ -11832,7 +11832,7 @@
    pic_flag=""
    # How to pass a linker flag through the compiler.
   -wl=""
   +wl="-Wl,"
    # Compiler flag to prevent dynamic linking.
     link_static_flag=""
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make -j$threads
```

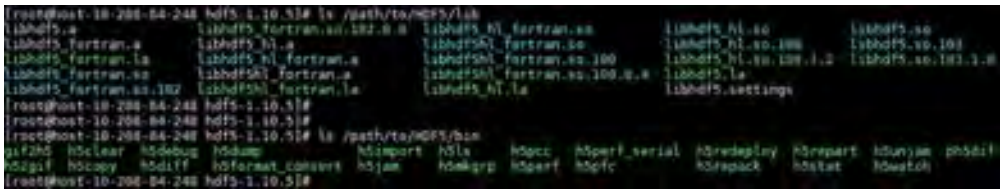



```
make install
```

Step 8 Verify the installation.

```
ls /path/to/HDF5/lib
ls /path/to/HDF5/bin
```

If the following information is displayed, the installation is successful:



Step 9 Load the environment variables.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.28.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

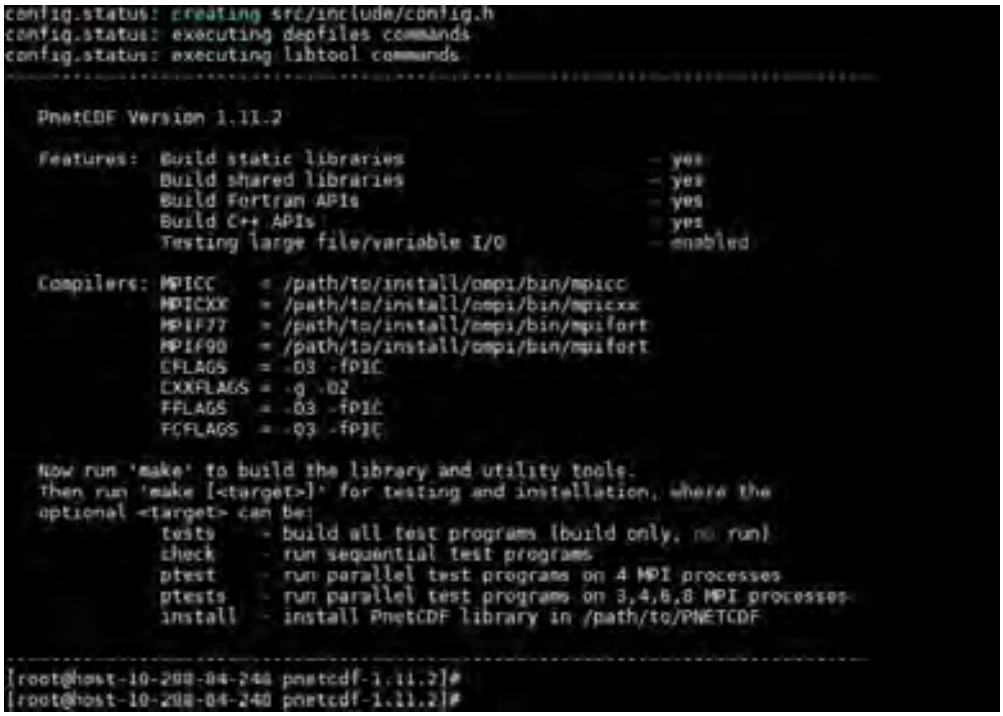
```
tar -zxvf pnetcdf-1.11.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pnetcdf-1.11.2
```

Step 4 Perform the configuration.

```
CC=mpicc FC=mpifort CXX=mpicxx CFLAGS="$optflags -fPIC" FFLAGS="$optflags -fPIC"
FCFLAGS="$optflags -fPIC" LD="$ldflags" ./configure --build=aarch64-linux --prefix=/path/to/PNETCDF
--enable-shared --enable-static --enable-fortran --enable-large-file-test
```



Step 5 Perform the installation.

```
make -j$threads
```

```
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/CXX'
Making all in nf_test
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf_test'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf_test'
Making all in nf90_test
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf90_test'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf90_test'
Making all in F90
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/F90'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/F90'
Making all in largefile
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/largefile'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/largefile'
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[1]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

make install

```
+-----------------------------------------------------------------------+
|                                                                       |
|  PnetCDF has been successfully installed under                        |
|         /path/to/PNETCDF                                              |
|                                                                       |
|  * PnetCDF header files have been installed in                        |
|         /path/to/PNETCDF/include                                      |
|  * PnetCDF library files have been installed in                       |
|         /path/to/PNETCDF/lib                                          |
|  * PnetCDF utility programs have been installed in                    |
|         /path/to/PNETCDF/bin                                          |
|  * PnetCDF man pages have been installed in                           |
|         /path/to/PNETCDF/share/man                                    |
|                                                                       |
|  To compile your PnetCDF programs, please add the following to the command |
|  line, so the compiler can find the PnetCDF header files:             |
|      -I/path/to/PNETCDF/include                                       |
|                                                                       |
|  Add the following line to link your program to PnetCDF library:      |
|      -L/path/to/PNETCDF/lib -lpnetcdf                                 |
|                                                                       |
|  Add the following to your run-time environment variable LD_LIBRARY_PATH, |
|  when linking your executable with the PnetCDF shared libraries.      |
|      /path/to/PNETCDF/lib                                             |
|                                                                       |
|                                                                       |
|  PnetCDF is jointly developed by a team at Northwestern University and |
|  Argonne National Laboratory.                                         |
|                                                                       |
|  Visit PnetCDF project web site for more information:                 |
|      https://parallel-netcdf.github.io                                |
|                                                                       |
+-----------------------------------------------------------------------+
make[3]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

Step 6 Verify the installation.

ls */path/to/PNETCDF*/**bin**
ls */path/to/PNETCDF*/**lib**

```
[root@host-10-200-84-248 pnetcdf-1.11.2]# ls /path/to/PNETCDF/bin
ncmpidiff  ncmpidump  ncmpigen  ncoffsets  ncvalidator  pnetcdf-config  pnetcdf_version
[root@host-10-200-84-248 pnetcdf-1.11.2]#
[root@host-10-200-84-248 pnetcdf-1.11.2]# ls /path/to/PNETCDF/lib
libpnetcdf.a  libpnetcdf.la  libpnetcdf.so  libpnetcdf.so.3  libpnetcdf.so.3.0.2  pkgconfig
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Load the environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.28.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y curl libcurl-devel zlib-devel
```

Step 3 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

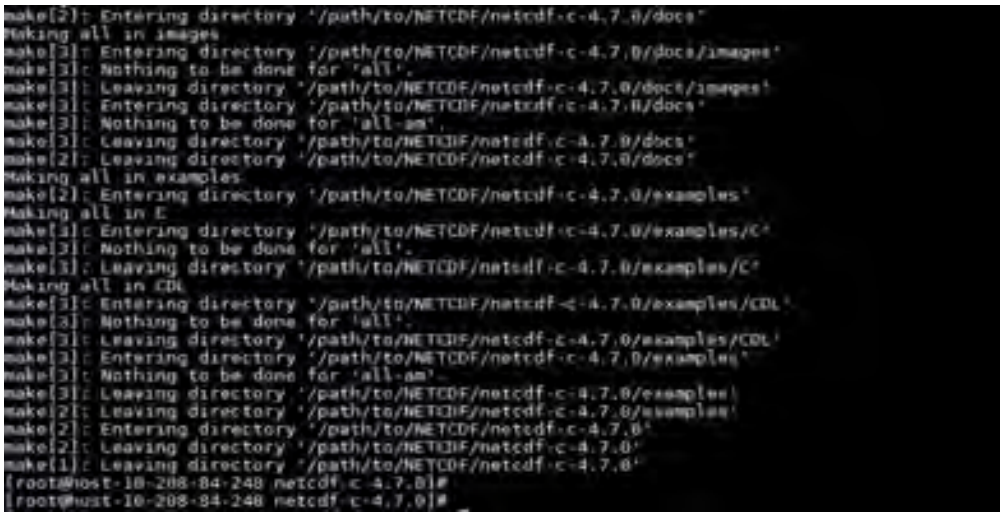
```
cd netcdf-c-4.7.0
```

Step 5 Perform the configuration.

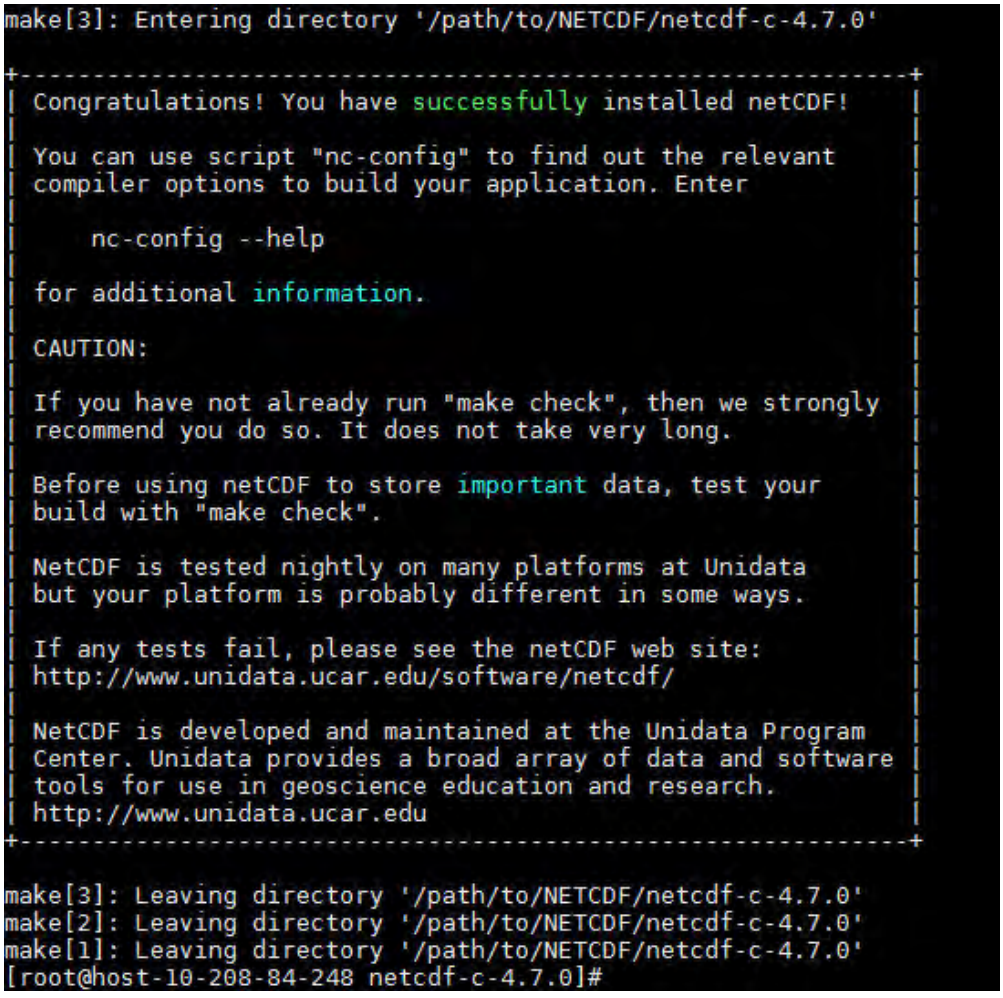
```
CC=mpicc CFLAGS="$optflags -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
LDFLAGS="$ldflags -L/opt/compiler/BS/bisheng-compiler-1.3.1-aarch64-linux/lib -L/path/to/HDF5/lib -L/
path/to/PNETCDF/lib" ./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-static --
enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-largefile --enable-large-
file-tests --disable-dap
```

Step 6 Perform the installation.

```
make -j$threads
```



make install



Step 7 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**

```
[root@host-10-208-84-248 ~]# ls /path/to/NETCDF/bin
nc-config  nccopy  ncdump  ncgen  ncgen3
[root@host-10-208-84-248 ~]#
[root@host-10-208-84-248 ~]# ls /path/to/NETCDF/lib
libh5bzip2.la  libnetcdf.a   libnetcdf.settings  libnetcdf.so.15      pkgconfig
libh5bzip2.so  libnetcdf.la  libnetcdf.so        libnetcdf.so.15.0.1
[root@host-10-208-84-248 ~]#
```

Step 8 Load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.28.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Perform the configuration.

```
CC=mpicc CFLAGS="$optflags -I/path/to/HDF5/include -I/path/to/NETCDF/include" FC=mpif90 FFLAGS="$optflags -fPIC -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="$optflags -fPIC -I/path/to/HDF5/include -I/path/to/NETCDF/include" F77=mpif77 LDFLAGS="$ldflags -L/opt/compiler/BS/bisheng-compiler-1.3.1-aarch64-linux -L/path/to/HDF5/lib -L/path/to/NETCDF/lib" ./configure --build=aarch64-linux --prefix=/path/to/NETCDF --enable-static --enable-shared --with-pic --enable-parallel-tests --enable-largefile --enable-large-file-tests
```

```
checking size of float... 4
checking size of double... 8
checking size of off_t... 8
checking size of size_t... 8
checking size of ptrdiff_t... 8
configure: generating header files and makefiles
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating nf-config
config.status: creating netcdf-fortran.pc
config.status: creating docs/Makefile
config.status: creating libsrc/Makefile
config.status: creating fortran/Makefile
config.status: creating nf_test/Makefile
config.status: creating nf03_test/Makefile
config.status: creating CMakeExtras/Makefile
config.status: creating examples/Makefile
config.status: creating examples/F90/Makefile
config.status: creating examples/F77/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: creating libsrc/nfconfig1.inc
config.status: libsrc/nfconfig1.inc is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 5 Apply the patch package.

```
patch -p1 < libtool.patch
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# vi libtool.patch
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# patch -p1 < libtool.patch
patching file libtool
Hunk #1 succeeded at 11835 (offset 3 lines).
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 6 Create a **libtool.patch** file and edit the file.

1. Create a **libtool.patch** file.
   vi libtool.patch
2. Press **i** to enter the edit mode and add the following content:

```
diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
--- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
+++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
@@ -11832,7 +11832,7 @@
 pic_flag=""
 # How to pass a linker flag through the compiler.
-wl=""
+wl="-Wl,"
 # Compiler flag to prevent dynamic linking.
 link_static_flag=""
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the installation.

make -j$threads

```
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/docs'
Making all in examples
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making all in F77
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
Making all in F90
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make  all-am
make[4]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Nothing to be done for 'all-am'.
make[4]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making all in CMakeExtras
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5'
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

make install

 Copyright © Huawei Technologies Co., Ltd.

```
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
Making install in F90
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make  install-am
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Nothing to be done for 'install-exec-am'.
make[4]: Nothing to be done for 'install-data-am'.
make[4]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making install in CMakeExtras
make[1]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 8 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**

```
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# ls /path/to/NETCDF/bin
nc-config  ncdump  ncgen  ncgen3  nf-config
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# ls /path/to/NETCDF/lib
libnetcdff.a  libnetcdff.la  libnetcdff.so  libnetcdff.so.6  libnetcdff.so.6.1.1  libnetcdff_exports.cmake
```

----End

1.28.4.7 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar -zxvf cmake-3.16.5.tar.gz
```

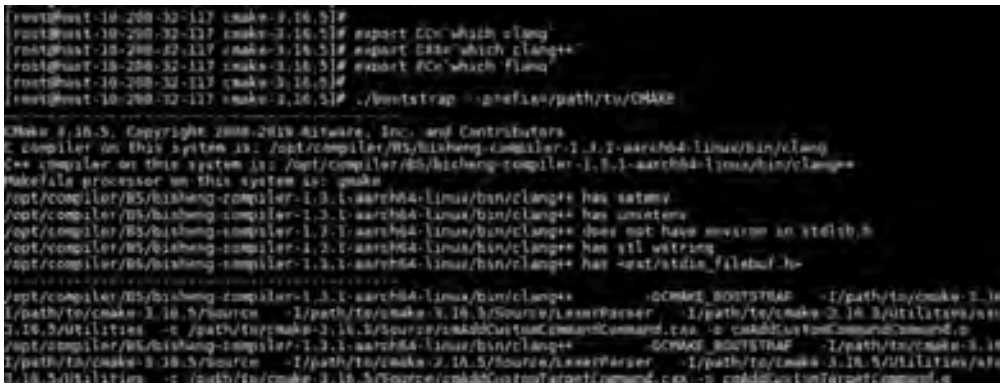
Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.16.5
```

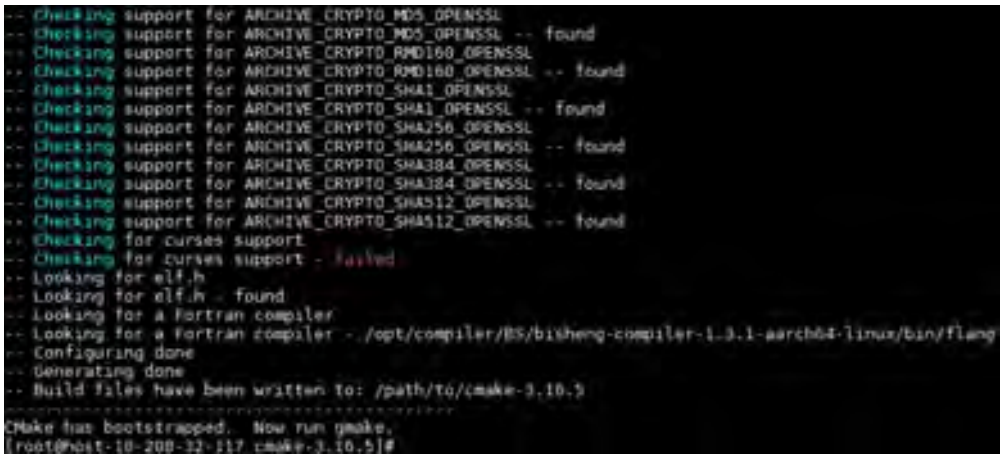
Step 4 Perform the configuration.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
./bootstrap --prefix=/path/to/CMAKE
```

Output for the start:



Output for the end:



Step 5 Perform the installation.

```
make
make install
```

```
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/ApplicationIcon.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Logo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SplashScreen.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/Windows_TemporaryKey.pfx
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo44x44.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/StoreLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/Windows/SmallLogo.png
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CPack.GenericLicense.txt
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/CTestScript.cmake.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/AppleInfo.plist
-- Installing: /path/to/CMAKE/share/cmake-3.16/Templates/TestDriver.cxx.in
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/indent/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/vim/syntax/cmake.vim
-- Installing: /path/to/CMAKE/share/cmake-3.16/editors/emacs/cmake-mode.el
-- Installing: /path/to/CMAKE/share/aclocal/cmake.m4
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cmake
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/cpack
-- Installing: /path/to/CMAKE/share/cmake-3.16/completions/ctest
[root@host-10-208-32-117 cmake-3.16.5]#
```

Step 6 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 7 Check the version after the installation.

```
cmake --version
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-32-117 to]# cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
[root@host-10-208-32-117 to]# CMAKE/bin/cmake --version
cmake version 3.16.5

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@host-10-208-32-117 to]#
```

----End

1.28.5 Obtaining the Source Code

Procedure

Step 1 Create the NCEPLIBS installation directory */path/to/NCEPLIBS*.

```
mkdir -p /path/to/NCEPLIBS
```

Step 2 Go to the **NCEPLIBS** directory.

```
cd /path/to/NCEPLIBS
```

Step 3 Download the NCEPLIBS installation package.

https://github.com/NOAA-EMC/NCEPLIBS/archive/refs/tags/v1.2.0.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the NCEPLIBS installation package and rename the package.

```
wget https://github.com/NOAA-EMC/NCEPLIBS/archive/refs/tags/v1.2.0.tar.gz  --no-check-certificate
mv v1.2.0.tar.gz NCEPLIBS-1.2.0.tar.gz
```

Step 4 Use an SFTP tool to upload the NCEPLIBS installation package to the */path/to/NCEPLIBS* directory on the server.

----End

1.28.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the NCEPLIBS installation package directory.

```
cd /path/to/NCEPLIBS
```

Step 3 Decompress the NCEPLIBS installation package.

```
tar -zxvf NCEPLIBS-1.2.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd NCEPLIBS-1.2.0
```

Step 5 Modify the download addresses in the **CMakeList** file.

1. Open the **CMakeList** file.

   vi */path/to/NCEPLIBS*/**NCEPLIBS-1.2.0/CMakeLists.txt**

2. Press **i** to enter the edit mode and change **https** in lines 153, 186, and 215 to **git**.

```
set(GIT_URL    "git://github.com/noaa-emc/nceplibs-${lib}")
set(GIT_URL    "git://github.com/noaa-emc/emc_crtm")
set(GIT_URL    "git://github.com/noaa-emc/emc_post")
```

NOTE

After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

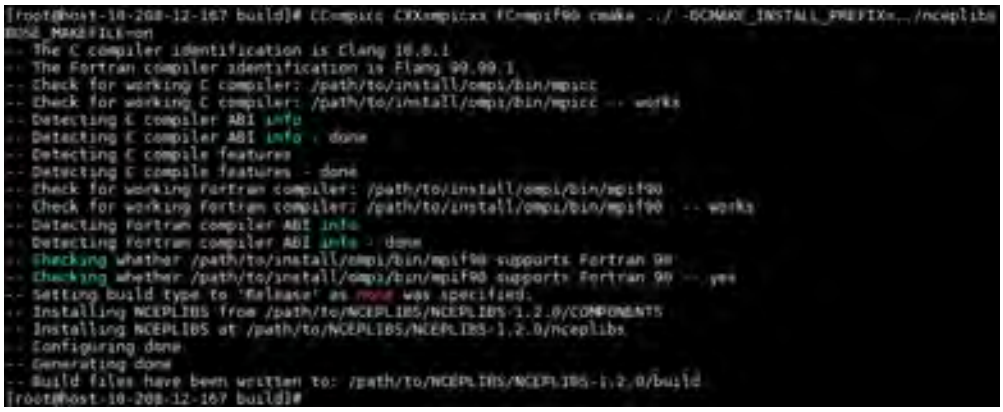
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Create the **build** and **nceplibs** directories, and go to the **build** directory.

```
mkdir -p build && mkdir nceplibs
cd build
```

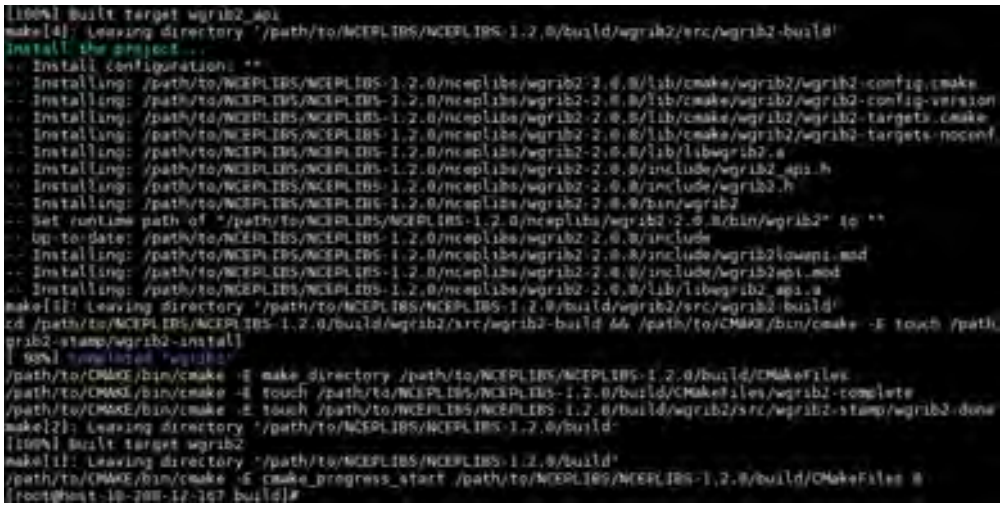
Step 7 Build the software.

```
CC=mpicc CXX=mpicxx FC=mpif90 cmake ../ -DCMAKE_INSTALL_PREFIX=../nceplibs -
DCMAKE_VERBOSE_MAKEFILE=on
```



Step 8 Perform the compilation and installation.

```
make
```



 Copyright © Huawei Technologies Co., Ltd.

NOTE

- If the sigio module reports an error during the installation, see **1.28.8.1 Error Reported When Compiling the sigio Module**.
- If the w3emc module reports an error during the installation, see **1.28.8.2 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the w3emc Module** and **1.28.8.3 Error "F90-F-0004-Unable to open MODULE file sigio_module.mod" Reported When Compiling the w3emc Module**.
- If the gfsio module reports an error during the installation, see **1.28.8.4 Error Reported When Compiling the gfsio Module** and **1.28.8.5 Error Reported When Compiling the sfcio Module**.
- If the sfcio module reports an error during the installation, see **1.28.8.5 Error Reported When Compiling the sfcio Module**.
- If the g2 module reports an error during the installation, see **1.28.8.6 Error Reported When Compiling the g2 Module**.
- If the nemsiogfs module reports an error during the installation, see **1.28.8.7 Error Reported When Compiling the nemsiogfs Module**.
- If the landsfcutil module reports an error during the installation, see **1.28.8.8 Error Reported When Compiling the landsfcutil Module**.
- If the nceppost module reports an error during the installation, see **1.28.8.9 Error "Failed to clone repository: 'git://github.com/noaa-emc/emc_post'" Reported When Compiling the nceppost Module**, **1.28.8.10 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the nceppost Module**, **1.28.8.11 Errors "F90-S-0021-Label field of continuation line is not blank" and "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module**, and **1.28.8.12 Error "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module**.
- If the wrf_io module reports an error during the installation, see **1.28.8.13 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wrf_io Module** and **1.28.8.14 Error "F90-S-0017-Unable to open include file: netcdf.inc" Reported When Compiling the wrf_io Module**.
- If the wgrib2 module reports an error during the installation, see **1.28.8.15 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wgrib2 Module** and **1.28.8.16 Errors "Could NOT find JPEG" and "Could NOT find Jasper" Reported When Compiling the wgrib2 Module**.

----End

1.28.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the library file installation directory and view the library file.

```
cd /path/to/NCEPLIBS/NCEPLIBS-1.2.0/nceplibs
```

If the following dependency library information is displayed, the installation is complete.

```
[root@host-10-208-12-167 nceplibs]# ll
total 72K
drwxr-xr-x. 5 root root 4.0K Dec  1 18:58 bacio-2.4.1
drwxr-xr-x. 7 root root 4.0K Dec  1 19:18 bufr-11.4.0
drwxr-xr-x. 4 root root 4.0K Dec  1 18:57 crtm-2.3.0
drwxr-xr-x. 5 root root 4.0K Dec  1 19:25 g2-3.4.1
drwxr-xr-x. 5 root root 4.0K Dec  1 19:26 g2tmpl-1.9.1
drwxr-xr-x. 4 root root 4.0K Dec  1 19:14 gfsio-1.4.1
drwxr-xr-x. 6 root root 4.0K Dec  1 19:26 ip-3.3.3
drwxr-xr-x. 5 root root 4.0K Dec  1 19:32 landsfcutil-2.4.1
drwxr-xr-x. 5 root root 4.0K Dec  1 19:41 nceppost-dceca26
drwxr-xr-x. 5 root root 4.0K Dec  1 18:59 nemsio-2.5.2
drwxr-xr-x. 4 root root 4.0K Dec  1 19:29 nemsiogfs-2.5.3
drwxr-xr-x. 4 root root 4.0K Dec  1 19:17 sfcio-1.4.1
drwxr-xr-x. 4 root root 4.0K Dec  1 19:00 sigio-2.3.2
drwxr-xr-x. 6 root root 4.0K Dec  1 19:12 sp-2.3.3
drwxr-xr-x. 6 root root 4.0K Dec  1 19:12 w3emc-2.7.3
drwxr-xr-x. 6 root root 4.0K Dec  1 18:58 w3nco-2.4.1
drwxr-xr-x. 5 root root 4.0K Dec  1 20:00 wgrib2-2.0.8
drwxr-xr-x. 4 root root 4.0K Dec  1 19:47 wrf_io-1.1.1
[root@host-10-208-12-167 nceplibs]#
```

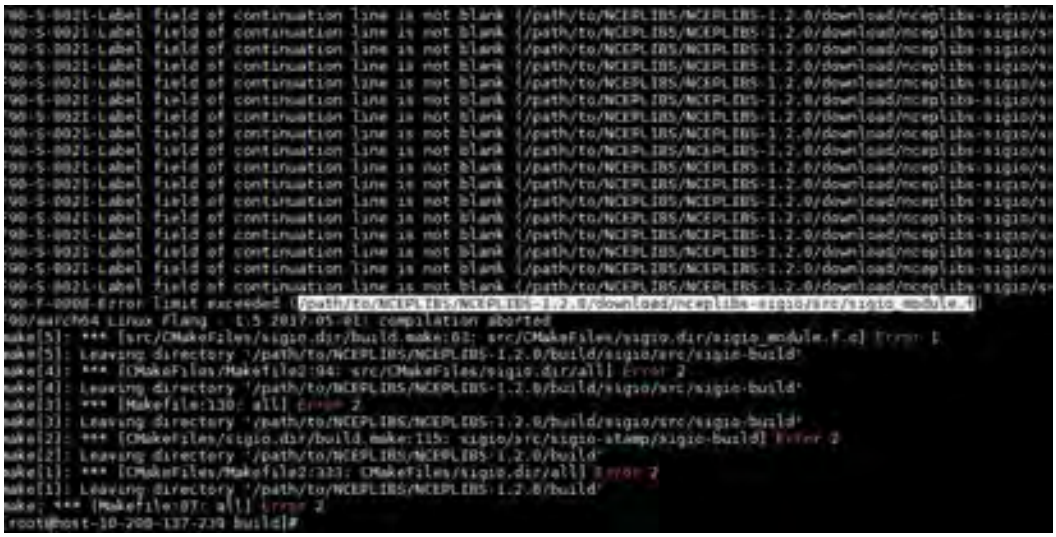
----End

1.28.8 Troubleshooting

1.28.8.1 Error Reported When Compiling the sigio Module

Symptom

During the NCEPLIBS compilation and installation, the sigio module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

Add a flang branch.

Step 1 Modify the **/nceplibs-sigio/src/CMakeLists.txt** file.

1. Modify the **/nceplibs-sigio/src/CMakeLists.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/nceplibs-sigio/src/CMakeLists.txt
2. Press **i** to enter the edit mode and add a new line above line 10:
   ```
   elseif(CMAKE_Fortran_COMPILER_ID MATCHES "^(Flang)$")
   set(CMAKE_Fortran_FLAGS
   "-g -ffree-form -fconvert=big-endian -funroll-loops ${CMAKE_Fortran_FLAGS}")
   set(CMAKE_Fortran_FLAGS_RELEASE "-O2")
   set(CMAKE_Fortran_FLAGS_DEBUG "-ggdb -Wall")
   ```

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

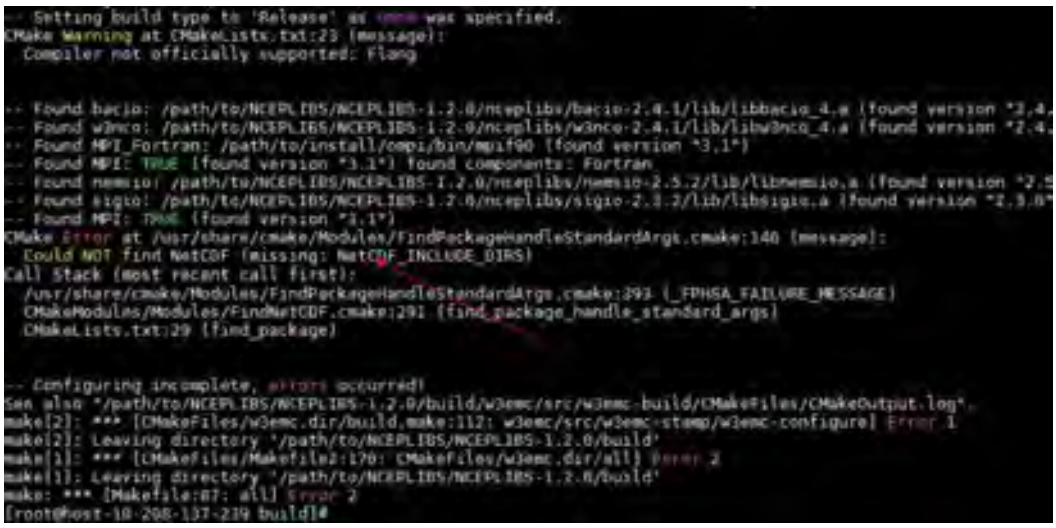
```
make
```

----End

1.28.8.2 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the w3emc Module

Symptom

During the NCEPLIBS compilation and installation, the w3emc module reports the following error:



Key Process and Cause Analysis

The NetCDF installation path is not added to the used file.

 Copyright © Huawei Technologies Co., Ltd.

Conclusion and Solution

Step 1 Modify the **CMakeCache.txt** file.

1. Open the **CMakeCache.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/w3emc/src/w3emc-build/CMakeCache.txt
2. Press **i** to enter the edit mode and modify the file as follows:
   //NetCDF Include directory paths
   NetCDF_INCLUDE_DIRS:STRING=**/path/to/NETCDF/include**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

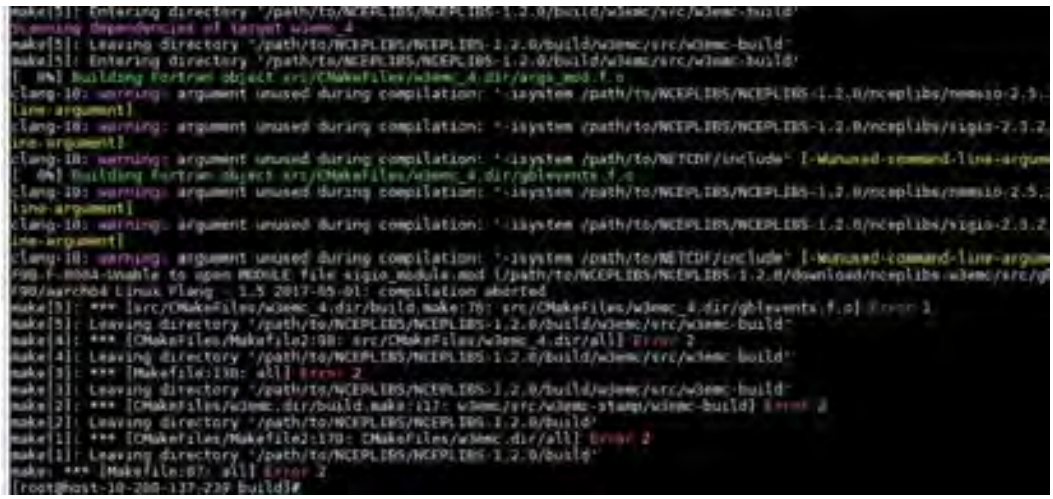
make

----End

1.28.8.3 Error "F90-F-0004-Unable to open MODULE file sigio_module.mod" Reported When Compiling the w3emc Module

Symptom

During the NCEPLIBS compilation and installation, the w3emc module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

Step 1 Modify the **/w3emc.dir/build.make** file.

1. Open the **/w3emc.dir/build.make** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/CMakeFiles/w3emc.dir/build.make
2. Press **i** to enter the edit mode and modify the content in line 117.
   cd /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/w3emc/src/w3emc-build && sed -i "s/isystem/I/g" `grep -rl "isystem"` && $(MAKE)

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Modify the **/nceplibs-w3emc/src/gblevents.f** file.

1. Modify the **/nceplibs-w3emc/src/gblevents.f** file.
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/nceplibs-w3emc/src/gblevents.f
   ```
2. Press **i** to enter the edit mode and modify the file as follows:

   Modify line 2463 as follows:

   ```
   INTEGER IDATE(8,2),JDATE(8,2),KDATE(8,2),KINDX(2),KINDX_2
   ```

   Replace the content from ELSE in line 2783 to ENDIF with the following:

   ```
   ELSE
   KINDX_2 = ABS(KINDX(2))
   COFS=
   $   ((KINDX_2*COFS_f(:,:,1))  +(KINDX(1)*COFS_f(:,:,2)))/3.
   COFV=
   $   ((KINDX_2*COFV_f(:,:,:,1))+(KINDX(1)*COFV_f(:,:,:,2)))/3.
   ENDIF
   ```

   > **NOTE**
   >
   > After performing **Step 2.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Perform the compilation again.

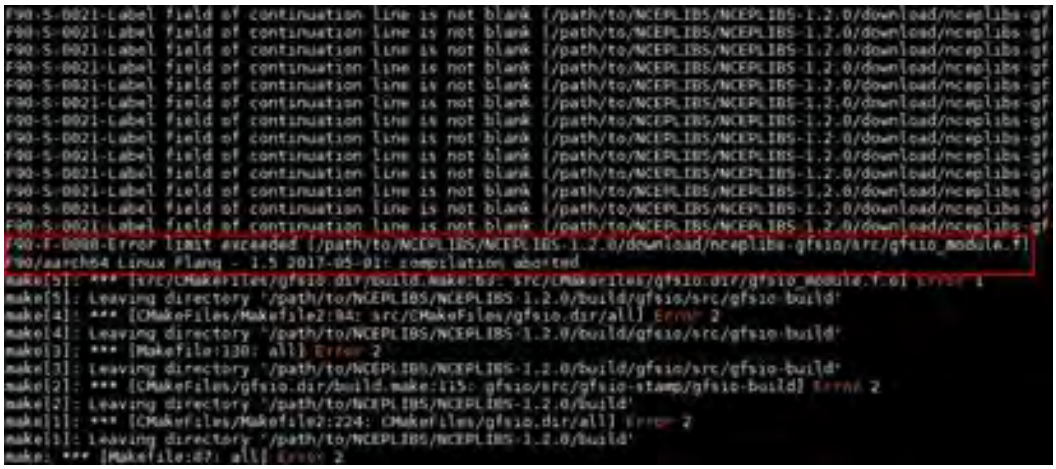
```
make
```

----End

1.28.8.4 Error Reported When Compiling the gfsio Module

Symptom

During the NCEPLIBS compilation and installation, the gfsio module reports the following error:



 Copyright © Huawei Technologies Co., Ltd.

Key Process and Cause Analysis

None

Conclusion and Solution

Add a flang branch.

Step 1 Modify the **/nceplibs-gfsio/src/CMakeLists.txt** file.

1. Modify the **/nceplibs-gfsio/src/CMakeLists.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/nceplibs-gfsio/src/CMakeLists.txt
2. Press **i** to enter the edit mode and add a new line above line 8:
   ```
   elseif(CMAKE_Fortran_COMPILER_ID MATCHES "^(Flang)$")
   set(CMAKE_Fortran_FLAGS "-g -fconvert=big-endian -ffree-form ${CMAKE_Fortran_FLAGS}")
   set(CMAKE_Fortran_RELEASE "-O2")
   ```

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

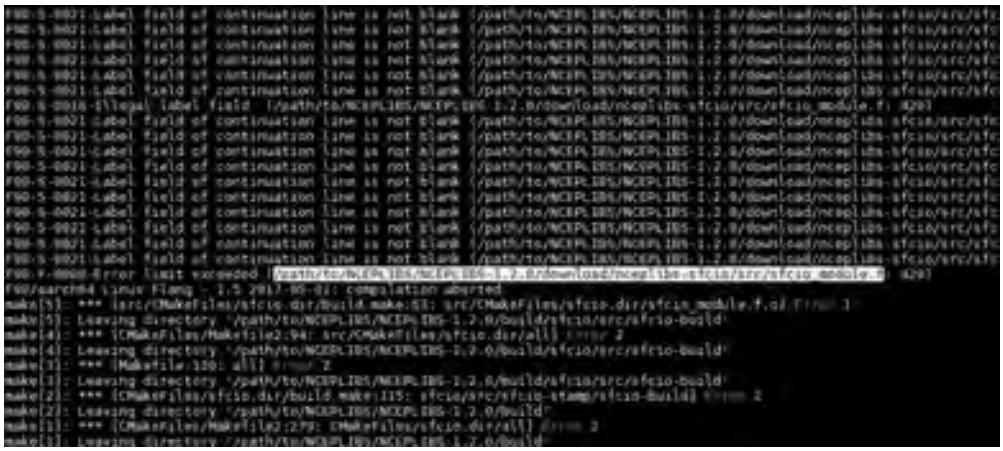
Step 2 Perform the compilation again.
make

----End

1.28.8.5 Error Reported When Compiling the sfcio Module

Symptom

During the NCEPLIBS compilation and installation, the sfcio module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

Add a flang branch.

Step 1 Modify the **/nceplibs-sfcio/src/CMakeLists.txt** file.

1. Modify the **/nceplibs-sfcio/src/CMakeLists.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/nceplibs-sfcio/src/CMakeLists.txt
2. Press **i** to enter the edit mode and add a new line above line 12:
   ```
   elseif(CMAKE_Fortran_COMPILER_ID MATCHES "^(Flang)$")
     set(CMAKE_Fortran_FLAGS
         "-g -ffree-form -fconvert=big-endian -funroll-loops ${CMAKE_Fortran_FLAGS}")
     set(CMAKE_Fortran_FLAGS_RELEASE "-O2")
     set(CMAKE_Fortran_FLAGS_DEBUG "-ggdb -Wall")
   ```

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

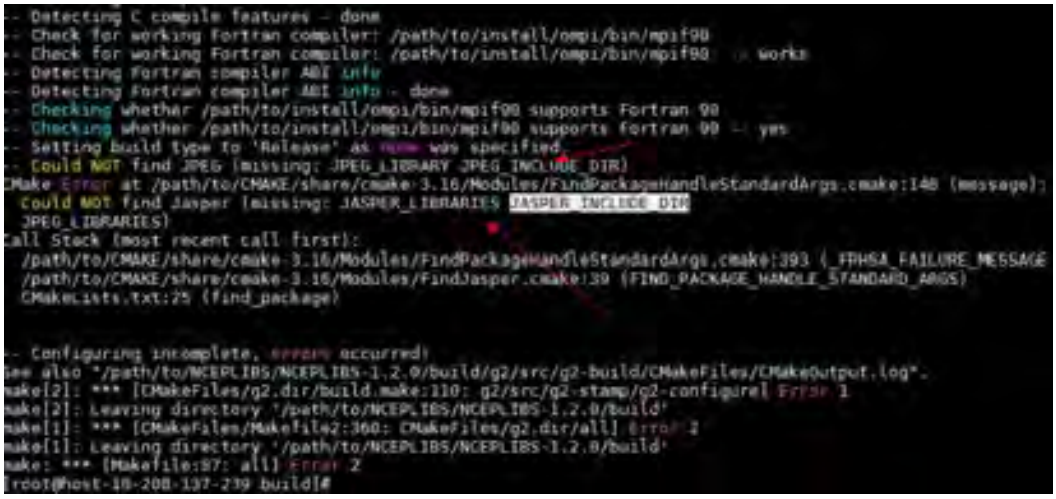
```
make
```

----End

1.28.8.6 Error Reported When Compiling the g2 Module

Symptom

During the NCEPLIBS compilation and installation, the g2 module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

Install and deploy JPEG and JASPER.

Step 1 Install JPEG and import the environment variables.

```
cd /path/to/JPEG
git clone git://github.com/LuaDist/libjpeg.git
cd libjpeg/
```

```
git branch
git checkout 6c0fcb8
./configure --prefix=/path/to/JPEG
make
make install
export PATH=/path/to/JPEG/bin:$PATH
export LD_LIBRARY_PATH=/path/to/JPEG/lib:$LD_LIBRARY_PATH
```

Step 2 Install JASPER and import the environment variables.

```
cd /path/to/JASPER
wget https://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.2.tar.gz
tar -zxvf jasper-1.900.2.tar.gz
cd jasper-1.900.2/
./configure --prefix=/path/to/JASPER
make all install
export PATH=/path/to/JASPER/bin:$PATH
export LD_LIBRARY_PATH=/path/to/JASPER/lib:$LD_LIBRARY_PATH
```

Step 3 Modify the **/g2/src/g2-build/CMakeCache.txt** file.

1. Open the **/g2/src/g2-build/CMakeCache.txt** file.
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/g2/src/g2-build/CMakeCache.txt
   ```
2. Press **i** to enter the edit mode and add the JASPER and JPEG installation paths.
   ```
   //Path to a file.
   JASPER_INCLUDE_DIR:PATH=/path/to/JASPER/include/
   //Path to a library.
   JASPER_LIBRARY_DEBUG:FILEPATH=/path/to/JASPER/lib/
   //Path to a library.
   JASPER_LIBRARY_RELEASE:FILEPATH=JASPER_LIBRARY_RELEASE-NOTFOUND
   //Path to a file.
   JPEG_INCLUDE_DIR:PATH=/path/to/JPEG/include
   //Path to a library.
   JPEG_LIBRARY_DEBUG:FILEPATH=/path/to/JPEG/lib
   //Path to a library.
   JPEG_LIBRARY_RELEASE:FILEPATH=JPEG_LIBRARY_RELEASE-NOTFOUND
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Perform the compilation again.

```
make
```

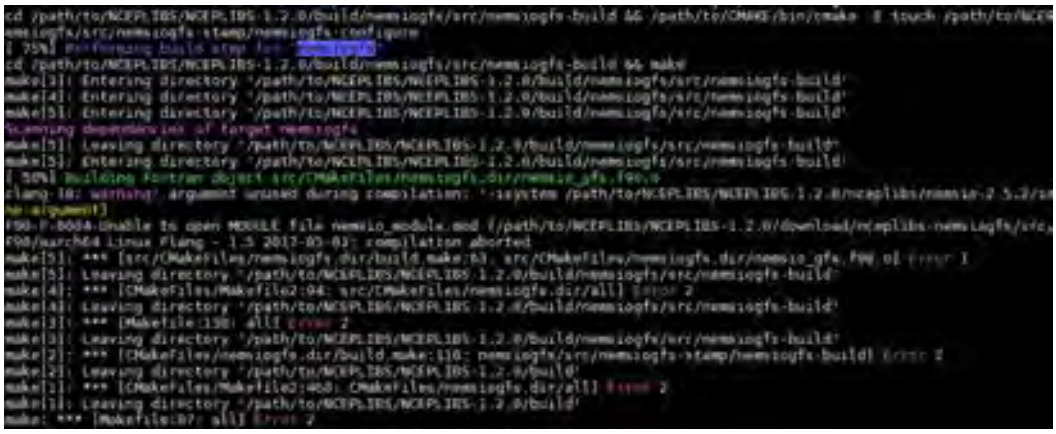
----End

1.28.8.7 Error Reported When Compiling the nemsiogfs Module

Symptom

During the NCEPLIBS compilation and installation, the nemsiogfs module reports the following error:

 Copyright © Huawei Technologies Co., Ltd.



Key Process and Cause Analysis

None

Conclusion and Solution

Step 1 Modify the **/nemsiogfs.dir/build.make** file.

1. Open the **/nemsiogfs.dir/build.make** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/CMakeFiles/nemsiogfs.dir/build.make
2. Press **i** to enter the edit mode and modify the content in line 116.
   cd /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/nemsiogfs/src/nemsiogfs-build && sed -i "s/isystem/I/g" `grep -rl "isystem"` && $(MAKE)

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

make

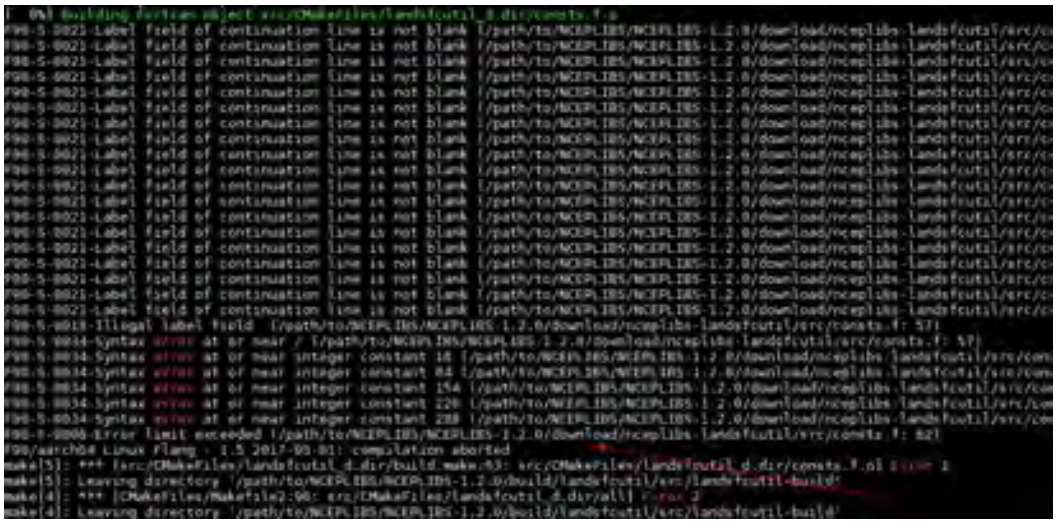
----End

1.28.8.8 Error Reported When Compiling the landsfcutil Module

Symptom

During the NCEPLIBS compilation and installation, the landsfcutil module reports the following error:

 Copyright © Huawei Technologies Co., Ltd.



Key Process and Cause Analysis

None

Conclusion and Solution

Add a flang branch.

Step 1 Modify the **/nceplibs-landsfcutil/src/CMakeLists.txt** file.

1. Modify the **/nceplibs-landsfcutil/src/CMakeLists.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/nceplibs-landsfcutil/src/CMakeLists.txt
2. Press **i** to enter the edit mode and add a new line above line 11:
   ```
   elseif(CMAKE_Fortran_COMPILER_ID MATCHES "^(Flang)$")
     set(CMAKE_Fortran_FLAGS "-g -ffree-form ${CMAKE_Fortran_FLAGS}")
     set(CMAKE_Fortran_FLAGS_RELEASE "-O3")
     set(CMAKE_Fortran_FLAGS_DEBUG "-ggdb -Wall")
     set(fortran_d_flags "-fdefault-real-8")
   ```

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

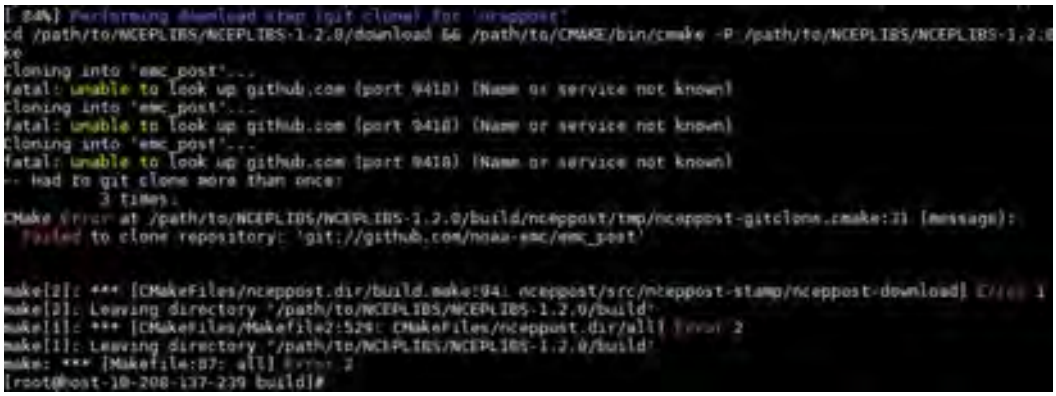
make

----End

1.28.8.9 Error "Failed to clone repository: 'git://github.com/noaa-emc/emc_post'" Reported When Compiling the nceppost Module

Symptom

During the NCEPLIBS compilation and installation, the nceppost module reports the following error:



Key Process and Cause Analysis

None

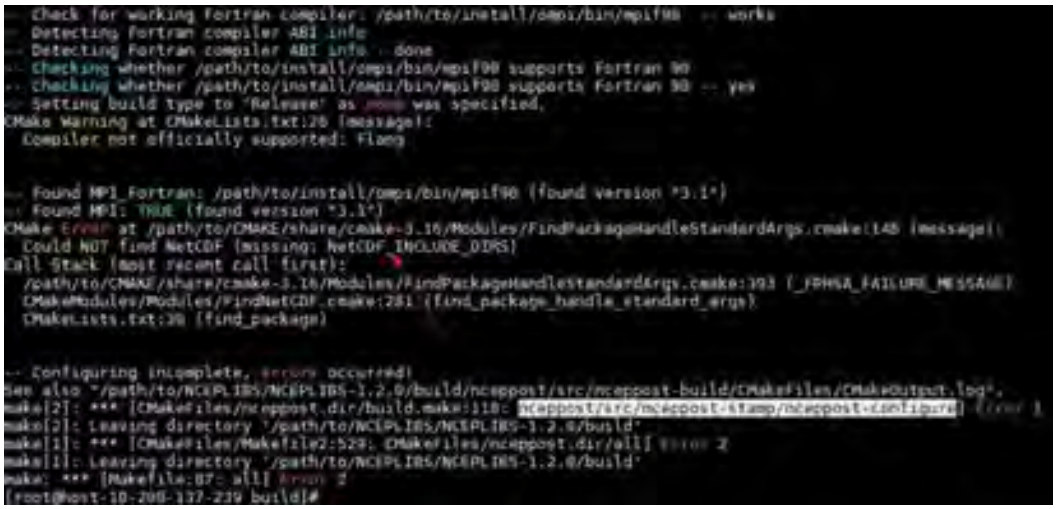
Conclusion and Solution

The github official website cannot be accessed due to a network problem. Perform the compilation and installation again.

1.28.8.10 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the nceppost Module

Symptom

During the NCEPLIBS compilation and installation, the nceppost module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

The NetCDF installation path is not added to the used file.

Step 1 Modify the **CMakeCache.txt** file.

1. Open the **CMakeCache.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/nceppost/src/nceppost-build/CMakeCache.txt
2. Press **i** to enter the edit mode and add the following information in bold:
   //NetCDF Include directory paths
   NetCDF_INCLUDE_DIRS:STRING=***/path/to/NETCDF***/**include**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

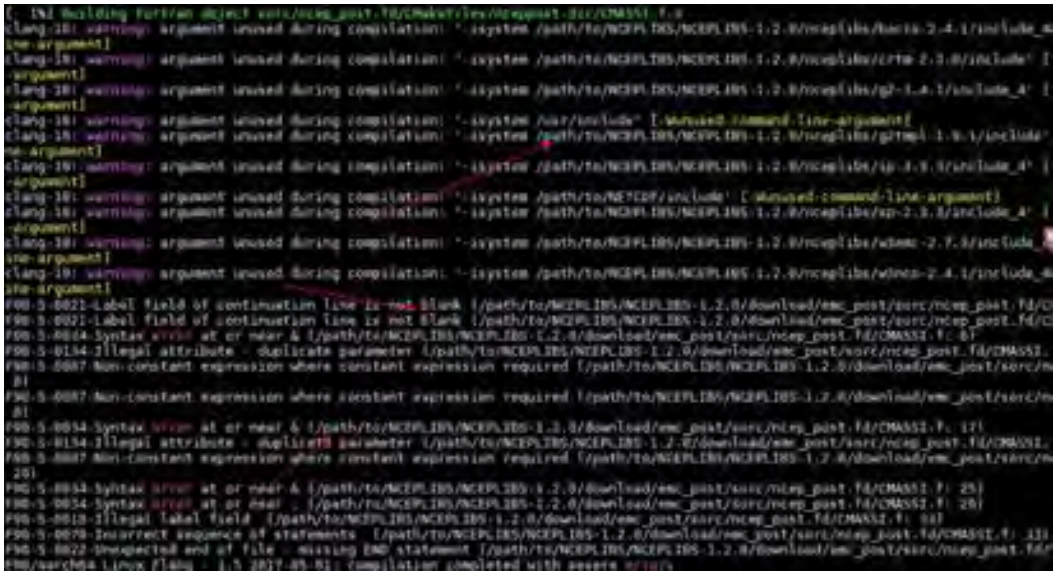
```
make
```

----End

1.28.8.11 Errors "F90-S-0021-Label field of continuation line is not blank" and "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module

Symptom

During the NCEPLIBS compilation and installation, the nceppost module reports the two errors: "F90-S-0021-Label field of continuation line is not blank" and "F90-S-0034-Syntax error at or near & (/path/to/NCEPLIBS/NCEPLIBE-1.2.0/download/emc_post/sorc/ncep_post.fd/CMASSI.f: 17)."



Key Process and Cause Analysis

None

Conclusion and Solution

Step 1 Modify the **/nceppost.dir/build.make** file.

1. Open the **/nceppost.dir/build.make** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/CMakeFiles/nceppost.dir/build.make
2. Press **i** to enter the edit mode and modify the content in line 123.

```
cd /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/nceppost/src/nceppost-build && sed -i "s/isystem/I/g"
`grep -rl "isystem"` && $(MAKE)
```

NOTE

After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

```
make
```

----End

1.28.8.12 Error "F90-S-0034-Syntax error at or near" Reported When Compiling the nceppost Module

Symptom

During the NCEPLIBS compilation and installation, the nceppost module reports the two errors: "F90-S-0034-Syntax error at or near & (/path/to/NCEPLIBS/NCEPLIBE-1.2.0/download/emc_post/sorc/ncep_post.fd/CMASSI.f: 6)" and "F90-S-0034-Syntax error at or near & (/path/to/NCEPLIBS/NCEPLIBE-1.2.0/download/emc_post/sorc/ncep_post.fd/CMASSI.f: 26)."

Key Process and Cause Analysis

None

Conclusion and Solution

Step 1 Modify the **/ncep_post.fd/CMakeLists.txt** file.

1. Open the **/ncep_post.fd/CMakeLists.txt** file.
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/download/emc_post/sorc/ncep_post.fd/CMakeLists.txt
   ```
2. Press **i** to enter the edit mode and add a new line above line 188:
   ```
   elseif(CMAKE_Fortran_COMPILER_ID MATCHES "^(Flang)$")
     set(CMAKE_Fortran_FLAGS "-g -ffree-form -fconvert=big-endian")
     set(CMAKE_Fortran_FLAGS_RELEASE "-O3")
     set(CMAKE_Fortran_FLAGS_DEBUG "-O0 -ggdb -fno-unsafe-math-optimizations -frounding-math
   -fsignaling-nans -ffpe-trap=invalid,zero,overflow -fbounds-check")
   ```

NOTE

After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

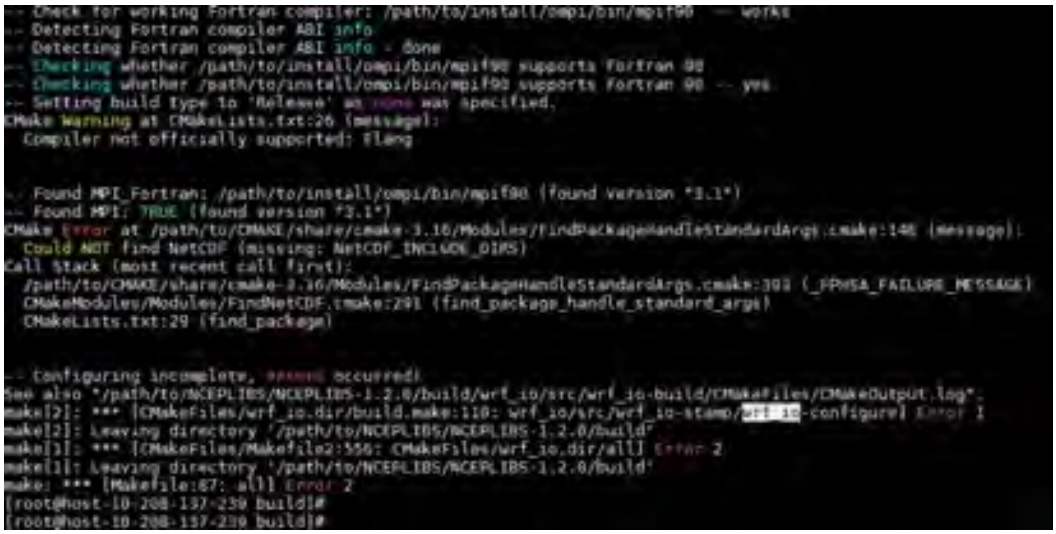
```
make
```

----End

1.28.8.13 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wrf_io Module

Symptom

During the NCEPLIBS compilation and installation, the wrf_io module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

The NetCDF installation path is not added to the used file.

Step 1 Modify the **CMakeCache.txt** file.

1. Open the **CMakeCache.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/wrf_io/src/wrf_io-build/CMakeCache.txt
2. Press **i** to enter the edit mode and modify the file as follows:
   //NetCDF Include directory paths
   NetCDF_INCLUDE_DIRS:STRING=**/path/to/NETCDF/include**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

```
make
```

----End

1.28.8.14 Error "F90-S-0017-Unable to open include file: netcdf.inc" Reported When Compiling the wrf_io Module

Symptom

During the NCEPLIBS compilation and installation, the wrf_io module reports the following error:

Key Process and Cause Analysis

None

Conclusion and Solution

Step 1 Modify the **/wrf_io.dir/build.make** file.

1. Open the **/wrf_io.dir/build.make** file.
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/CMakeFiles/wrf_io.dir/build.make
   ```
2. Press **i** to enter the edit mode and modify the content in line 115.
   ```
   cd /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/wrf_io/src/wrf_io-build && sed -i "s/isystem/I/g" `grep -rl "isystem"` && $(MAKE)
   ```

   NOTE

   After performing **Step 1.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

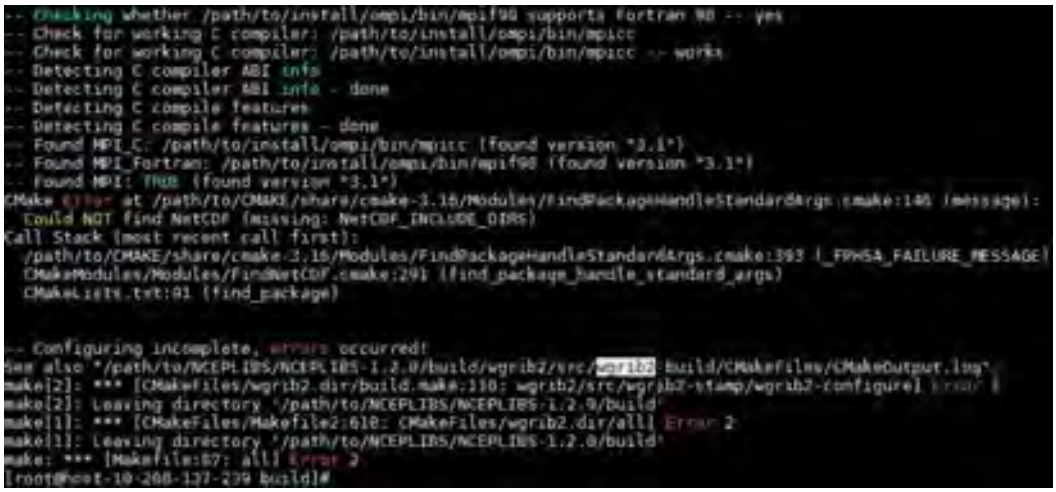
```
make
```

----End

1.28.8.15 Error "Could NOT find NetCDF (missing: NetCDF_INCLUDE_DIRS)" Reported When Compiling the wgrib2 Module

Symptom

During the NCEPLIBS compilation and installation, the wgrib2 module reports the following error:



Key Process and Cause Analysis

None

Conclusion and Solution

The NetCDF installation path is not added to the used file.

Step 1 Modify the **CMakeCache.txt** file.

1. Open the **CMakeCache.txt** file.
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/wgrib2/src/wgrib2-build/CMakeCache.txt
2. Press **i** to enter the edit mode and modify the file as follows:
   //NetCDF Include directory paths
   NetCDF_INCLUDE_DIRS:STRING=**/path/to/NETCDF/include**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

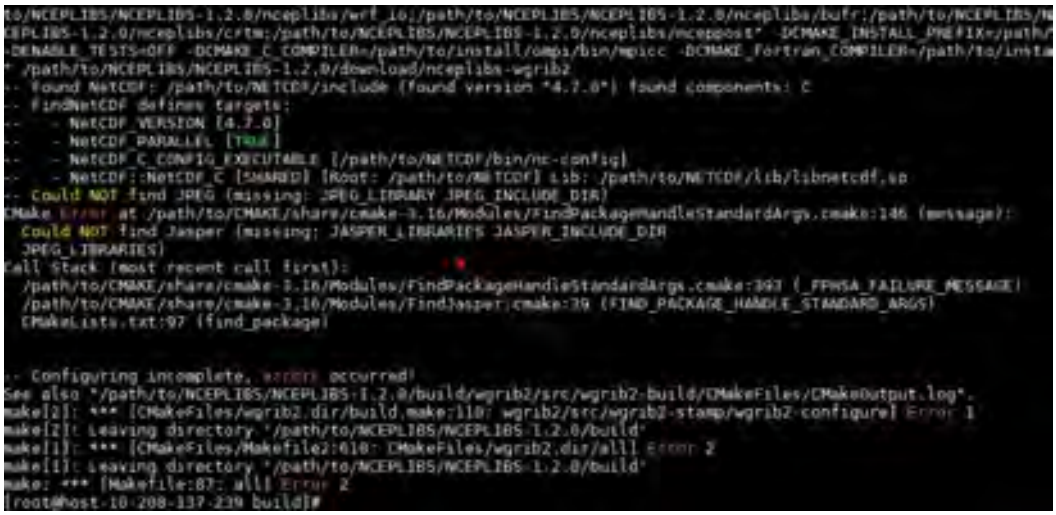
make

----End

1.28.8.16 Errors "Could NOT find JPEG" and "Could NOT find Jasper" Reported When Compiling the wgrib2 Module

Symptom

During the NCEPLIBS compilation and installation, the wgrib2 module reports the two errors: "Could NOT find JPEG (missing: JPEG_LIBRARY JPEG_INCLUDE_DIR)" and "Could NOT find Jasper (missing: JASPER_LIBRARIES JASPER_INCLUDE_DIR JPEG_LIBRARIES)."



Key Process and Cause Analysis

None

Conclusion and Solution

The JASPER and JPEG installation paths are not added to the used file.

Step 1 Modify the **CMakeCache.txt** file.

1. Open the **CMakeCache.txt** file.
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS-1.2.0/build/wgrib2/src/wgrib2-build/CMakeCache.txt
   ```
2. Press **i** to enter the edit mode and add the JASPER and JPEG installation paths.
   ```
   //Path to a file.
   JASPER_INCLUDE_DIR:PATH=/path/to/JASPER/include/
   //Path to a library.
   JASPER_LIBRARY_DEBUG:FILEPATH=/path/to/JASPER/lib/
   //Path to a library.
   JASPER_LIBRARY_RELEASE:FILEPATH=/path/to/JASPER/lib/libjasper.so

   //Path to a file.
   JPEG_INCLUDE_DIR:PATH=/path/to/JPEG/include
   //Path to a library.
   JPEG_LIBRARY_DEBUG:FILEPATH=/path/to/JPEG/lib
   //Path to a library.
   JPEG_LIBRARY_RELEASE:FILEPATH=JPEG_LIBRARY_RELEASE-NOTFOUND
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 2 Perform the compilation again.

```
make
```

----End

1.29 SWAN 41.31 Porting Guide (openEuler 20.03)

1.29.1 Introduction

SWAN is a third-generation wave model that computes random, short-crested wind-generated waves in coastal regions and inland waters. SWAN can be used on any scale relevant for wind-generated surface gravity waves.

For more information about SWAN, visit the **SWAN official website**.

Programming language: Fortran

Brief description: a wave model.

Recommended Software Version

SWAN 41.31

1.29.2 Environment Requirements

Hardware Requirements

Table 1-141 lists the hardware requirements.

Table 1-141 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-142 lists the software requirements.

Table 1-142 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
HDF5	1.10.5	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz**
PnetCDF	1.11.2	**http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz**
NetCDF-C	4.7.0	**https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
NetCDF-Fortran	4.4.5	**https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.5.tar.gz**
SWAN	41.31	**http://swanmodel.sourceforge.net/download/zip/swan4131.tar.gz**
Test case (**refrac.tar.gz**)	-	**http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz**

OS Requirements

Table 1-143 lists the OS requirements.

Table 1-143 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	**https://repo.openeuler.org/openEuler-20.03-LTS-SP1/**
Kernel	4.19	Included in the OS image

1.29.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SWAN software porting.

Table 1-144 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler/BS**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/install/hucx** **/path/to/install/hmpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/HDF5**	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the
4	**/path/to/PNETCDF**	Installation path of PnetCDF.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
5	**/path/to/NETCDF**	Installation path of NetCDF-C. Installation path of NetCDF-Fortran.	actual paths planned during the installation process.
6	**/path/to/SWAN**	Installation path of SWAN.	

1.29.4 Configuring the Compilation Environment

1.29.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-145 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Configuring the Yum Source	See **1.29.4.2 Configuring the Yum Source**.
4	Installing HDF5	See **1.29.4.3 Installing HDF5**.
5	Installing PnetCDF	See **1.29.4.4 Installing PnetCDF**.
6	Installing NetCDF-C	See **1.29.4.5 Installing NetCDF-C**.
7	Installing NetCDF-Fortran	See **1.29.4.6 Installing NetCDF-Fortran**.

 Copyright © Huawei Technologies Co., Ltd.

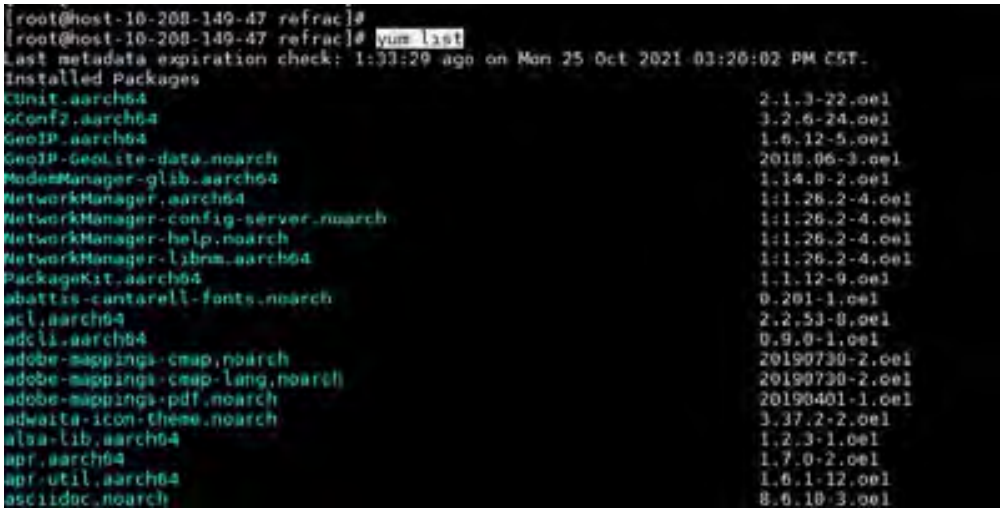
1.29.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether Yum is configured.

```
yum list
```



NOTE

The Yum source is configured by default on openEuler 20.03 (LTS SP1) and does not need to be configured additionally.

Step 3 Install the base dependencies.

```
yum install -y gcc gcc-c++ m4 wget openssl-devel git
```

----End

1.29.4.3 Installing HDF5

NOTE

- If the **configure** file in the source packages of the HDF5 components specifies that the GNU software optimization level is **O3**, add the following optimization options before performing **Step 1**. For details, see **Step 4**.

  ```
  export optflags="-O3"
  export ldflags="-Wl,--build-id"
  export threads=$(nproc)
  ```

- If the **configure** file specifies that non-GNU software optimization is not required, you do not need to add optimization options.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -zxvf hdf5-1.10.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.5
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Perform the configuration.

```
CC=mpicc CFLAGS="$optflags -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FC=mpif90 FCFLAGS="$optflags -fPIC" FFLAGS="$optflags -fPIC" LDFLAGS="$ldflags" CXX=mpicxx ./configure --prefix=/path/to/HDF5 --enable-shared --enable-static --enable-fortran --enable-parallel --enable-optimization=high
```

```
                    Fortran: yes
             Fortran Compiler: /path/to/install/compi/bin/mpif90 ( Bisheng Compiler 1.3.1
g-0658acfa40ab))
                Fortran Flags: -O3 -fPIC
             H5 Fortran Flags:
             AM Fortran Flags:
       Shared Fortran Library: yes
       Static Fortran Library: yes

                          C++: no

                         Java: no


Features:
---------
                  Parallel HDF5: yes
Parallel Filtered Dataset Writes: yes
             Large Parallel I/O: yes
             High-level library: yes
                   Threadsafety: no
            Default API mapping: v110
  With deprecated public symbols: yes
          I/O filters (external): deflate(zlib)
                            MPE:
                     Direct VFD: no
                        dmalloc: no
  Packages w/ extra debug output: none
                     API tracing: no
            Using memory checker: no
 Memory allocation sanity checks: no
           Function stack tracing: no
        Strict file format checks: no
     Optimization instrumentation: no
[root@host-10-208-84-248 hdf5-1.10.5]#
```

Step 5 Apply the patch package.

```
patch -p1 < libtool.patch
```

```
[root@host-10-208-84-248 hdf5-1.10.5]# vi libtool.patch
[root@host-10-208-84-248 hdf5-1.10.5]#
[root@host-10-208-84-248 hdf5-1.10.5]# patch -p1 < libtool.patch
patching file libtool
[root@host-10-208-84-248 hdf5-1.10.5]#
```

Step 6 Create a **libtool.patch** file and edit the file.

1. Create a **libtool.patch** file.
   ```
   vi libtool.patch
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
   --- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
   +++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
   @@ -11832,7 +11832,7 @@
    pic_flag=""
    # How to pass a linker flag through the compiler.
   -wl=""
   +wl="-Wl,"
    # Compiler flag to prevent dynamic linking.
      link_static_flag=""
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make -j$threads
```

 Copyright © Huawei Technologies Co., Ltd.

```
./fortran/src -I../../../fortran/test -I../../../hl/fortran/src -O3 -fPIC -c -o tstds.o tstds.f
mpif90  -I. -I../../../src -I../../../fortran/src     -I../../../fortran/src -I../../../hl/fort
./fortran/src -I../../../fortran/test -I../../../hl/fortran/src -O3 -fPIC -c -o tstlite.o tstli
mpif90  -I. -I../../../src -I../../../fortran/src     -I../../../fortran/src -I../../../hl/fort
./fortran/src -I../../../fortran/test -I../../../hl/fortran/src -O3 -fPIC -c -o tstimage.o tsti
mpif90  -I. -I../../../src -I../../../fortran/src     -I../../../fortran/src -I../../../hl/fort
./fortran/src -I../../../fortran/test -I../../../hl/fortran/src -O3 -fPIC -c -o tsttable.o tstt
  FCLD     tstimage
  FCLD     tstds
/usr/bin/ld: /opt/compiler/85/bisheng-compiler-1.3.1-aarch64-linux/lib/libomp.so: .dynsym local
/usr/bin/ld: /opt/compiler/85/bisheng-compiler-1.3.1-aarch64-linux/lib/libomp.so: .dynsym local
  FCLD     tsttable
/usr/bin/ld: /opt/compiler/85/bisheng-compiler-1.3.1-aarch64-linux/lib/libomp.so: .dynsym local
  FCLD     tstlite
/usr/bin/ld: /opt/compiler/85/bisheng-compiler-1.3.1-aarch64-linux/lib/libomp.so: .dynsym local
make[3]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran/test'
make[3]: Entering directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran'
make[2]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran'
make[2]: Entering directory '/path/to/HDF5/hdf5-1.10.5/hl'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl'
make[1]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl'
[root@host-10-208-84-248 hdf5-1.10.5]#
```

make install

```
+ /usr/bin/install -c ./ex_table_07.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_table_08.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_table_09.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_table_10.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_table_11.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_table_12.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./ex_ds1.c /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./image24pixel.txt /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./image8.txt /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./pal_rgb.h /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c run-hlc-ex.sh /path/to/HDF5/share/hdf5_examples/hl/c/.
+ /usr/bin/install -c ./run-hl-ex.sh /path/to/HDF5/share/hdf5_examples/hl/.
make[2]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/examples'
make[2]: Entering directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran'
make[3]: Entering directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran/examples'
../../../bin/mkdirs /path/to/HDF5/share/hdf5_examples/hl/fortran
+ /usr/bin/install -c ./exlite.f90 /path/to/HDF5/share/hdf5_examples/hl/fortran/.
+ /usr/bin/install -c ./ex_ds1.f90 /path/to/HDF5/share/hdf5_examples/hl/fortran/.
+ /usr/bin/install -c run-hlfortran-ex.sh /path/to/HDF5/share/hdf5_examples/hl/fortran/.
make[3]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran/examples'
make[2]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl/fortran'
make[1]: Leaving directory '/path/to/HDF5/hdf5-1.10.5/hl'
[root@host-10-208-84-248 hdf5-1.10.5]#
```

Step 8 Verify the installation.

ls */path/to/HDF5*/**lib**
ls */path/to/HDF5*/**bin**

If the following information is displayed, the installation is successful:

Step 9 Load the environment variables.

export PATH=*/path/to/HDF5*/**bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/HDF5*/**lib:$LD_LIBRARY_PATH**

----End

 Copyright © Huawei Technologies Co., Ltd.

1.29.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -zxvf pnetcdf-1.11.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pnetcdf-1.11.2
```

Step 4 Perform the configuration.

```
CC=mpicc FC=mpifort CXX=mpicxx CFLAGS="$optflags -fPIC" FFLAGS="$optflags -fPIC"
FCFLAGS="$optflags -fPIC" LD="$ldflags" ./configure --build=aarch64-linux --prefix=/path/to/PNETCDF
--enable-shared --enable-static --enable-fortran --enable-large-file-test
```

```
config.status: creating src/include/config.h
config.status: executing depfiles commands
config.status: executing libtool commands

   PnetCDF Version 1.11.2

   Features:  Build static libraries                  - yes
              Build shared libraries                  - yes
              Build Fortran APIs                      - yes
              Build C++ APIs                          - yes
              Testing large file/variable I/O         - enabled

   Compilers: MPICC    = /path/to/install/ompi/bin/mpicc
              MPICXX   = /path/to/install/ompi/bin/mpicxx
              MPIF77   = /path/to/install/ompi/bin/mpifort
              MPIF90   = /path/to/install/ompi/bin/mpifort
              CFLAGS   = -O3 -fPIC
              CXXFLAGS = -g -O2
              FFLAGS   = -O3 -fPIC
              FCFLAGS  = -O3 -fPIC

   Now run 'make' to build the library and utility tools.
   Then run 'make [<target>]' for testing and installation, where the
   optional <target> can be:
              tests     - build all test programs (build only, no run)
              check     - run sequential test programs
              ptest     - run parallel test programs on 4 MPI processes
              ptests    - run parallel test programs on 3,4,6,8 MPI processes
              install   - install PnetCDF library in /path/to/PNETCDF

[root@host-10-208-04-240 pnetcdf-1.11.2]#
[root@host-10-208-04-240 pnetcdf-1.11.2]#
```

Step 5 Perform the installation.

```
make -j$threads
```

```
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/CXX'
Making all in nf_test
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf_test'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf_test'
Making all in nf90_test
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf90_test'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/nf90_test'
Making all in F90
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/F90'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/F90'
Making all in largefile
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/largefile'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test/largefile'
make[2]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2/test'
make[1]: Entering directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

make install

```
+-----------------------------------------------------------------------+
|                                                                       |
|  PnetCDF has been successfully installed under                        |
|         /path/to/PNETCDF                                              |
|                                                                       |
|  * PnetCDF header files have been installed in                        |
|         /path/to/PNETCDF/include                                      |
|  * PnetCDF library files have been installed in                       |
|         /path/to/PNETCDF/lib                                          |
|  * PnetCDF utility programs have been installed in                    |
|         /path/to/PNETCDF/bin                                          |
|  * PnetCDF man pages have been installed in                           |
|         /path/to/PNETCDF/share/man                                    |
|                                                                       |
|  To compile your PnetCDF programs, please add the following to the command |
|  line, so the compiler can find the PnetCDF header files:             |
|      -I/path/to/PNETCDF/include                                       |
|                                                                       |
|  Add the following line to link your program to PnetCDF library:      |
|      -L/path/to/PNETCDF/lib -lpnetcdf                                 |
|                                                                       |
|  Add the following to your run-time environment variable LD_LIBRARY_PATH, |
|  when linking your executable with the PnetCDF shared libraries.      |
|      /path/to/PNETCDF/lib                                             |
|                                                                       |
|                                                                       |
|  PnetCDF is jointly developed by a team at Northwestern University and |
|  Argonne National Laboratory.                                         |
|                                                                       |
|  Visit PnetCDF project web site for more information:                 |
|      https://parallel-netcdf.github.io                                |
|                                                                       |
+-----------------------------------------------------------------------+
make[3]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[2]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
make[1]: Leaving directory '/path/to/PNETCDF/pnetcdf-1.11.2'
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

Step 6 Verify the installation.

ls */path/to/PNETCDF*/**bin**
ls */path/to/PNETCDF*/**lib**

```
[root@host-10-200-84-248 pnetcdf-1.11.2]# ls /path/to/PNETCDF/bin
ncmpidiff  ncmpidump  ncmpigen  ncoffsets  ncvalidator  pnetcdf-config  pnetcdf_version
[root@host-10-200-84-248 pnetcdf-1.11.2]#
[root@host-10-200-84-248 pnetcdf-1.11.2]# ls /path/to/PNETCDF/lib
libpnetcdf.a  libpnetcdf.la  libpnetcdf.so  libpnetcdf.so.3  libpnetcdf.so.3.0.2  pkgconfig
[root@host-10-200-84-248 pnetcdf-1.11.2]#
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Load the environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.29.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y curl libcurl-devel zlib-devel
```

Step 3 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

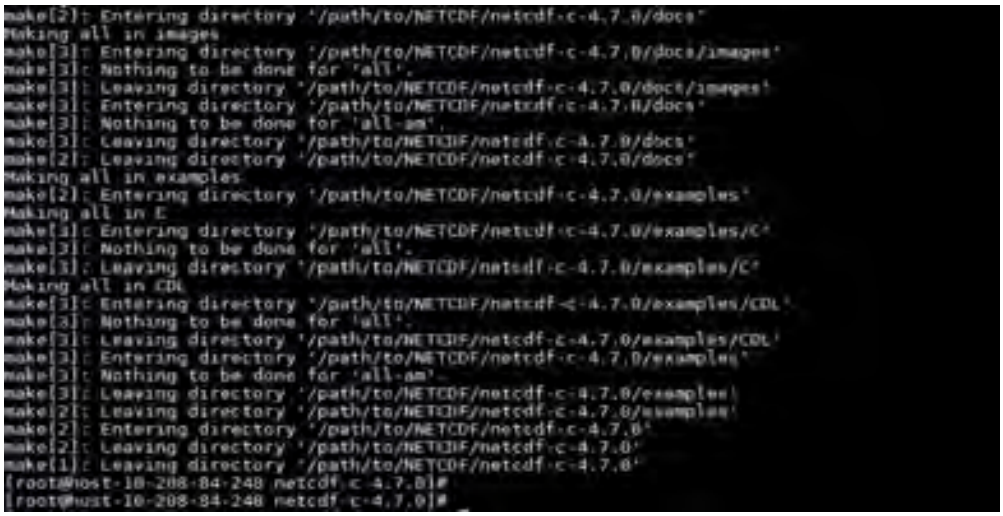
```
cd netcdf-c-4.7.0
```

Step 5 Perform the configuration.

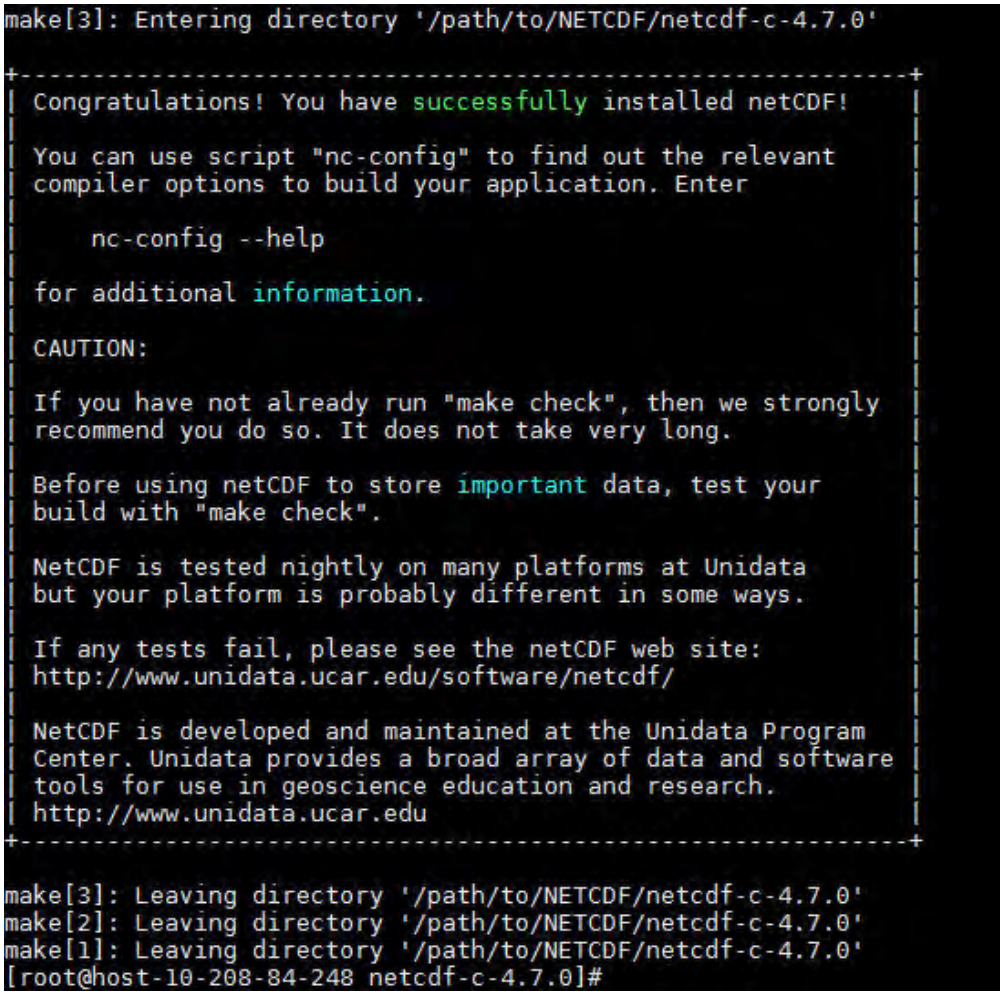
```
CC=mpicc CFLAGS="$optflags -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
LDFLAGS="$ldflags -L/opt/compiler/BS/bisheng-compiler-1.3.1-aarch64-linux/lib -L/path/to/HDF5/lib -L/path/to/PNETCDF/lib" ./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-largefile --enable-large-file-tests --disable-dap
```

Step 6 Perform the installation.

```
make -j$threads
```



make install



Step 7 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**

```
[root@host-10-208-84-248 ~]# ls /path/to/NETCDF/bin
nc-config  nccopy  ncdump  ncgen  ncgen3
[root@host-10-208-84-248 ~]#
[root@host-10-208-84-248 ~]# ls /path/to/NETCDF/lib
libh5bzip2.la  libnetcdf.a   libnetcdf.settings  libnetcdf.so.15      pkgconfig
libh5bzip2.so  libnetcdf.la  libnetcdf.so        libnetcdf.so.15.0.1
[root@host-10-208-84-248 ~]#
```

Step 8 Load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.29.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Perform the configuration.

```
CC=mpicc CFLAGS="$optflags -I/path/to/HDF5/include -I/path/to/NETCDF/include" FC=mpif90 FFLAGS="$optflags -fPIC -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="$optflags -fPIC -I/path/to/HDF5/include -I/path/to/NETCDF/include" F77=mpif77 LDFLAGS="$ldflags -L/opt/compiler/BS/bisheng-compiler-1.3.1-aarch64-linux -L/path/to/HDF5/lib -L/path/to/NETCDF/lib" ./configure --build=aarch64-linux --prefix=/path/to/NETCDF --enable-static --enable-shared --with-pic --enable-parallel-tests --enable-largefile --enable-large-file-tests
```

```
checking size of float... 4
checking size of double... 8
checking size of off_t... 8
checking size of size_t... 8
checking size of ptrdiff_t... 8
configure: generating header files and makefiles
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating nf-config
config.status: creating netcdf-fortran.pc
config.status: creating docs/Makefile
config.status: creating libsrc/Makefile
config.status: creating fortran/Makefile
config.status: creating nf_test/Makefile
config.status: creating nf03_test/Makefile
config.status: creating CMakeExtras/Makefile
config.status: creating examples/Makefile
config.status: creating examples/F90/Makefile
config.status: creating examples/F77/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: creating libsrc/nfconfig1.inc
config.status: libsrc/nfconfig1.inc is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 5 Apply the patch package.

```
patch -p1 < libtool.patch
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# vi libtool.patch
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
[root@host-10-208-84-248 netcdf-fortran-4.4.5]# patch -p1 < libtool.patch
patching file libtool
Hunk #1 succeeded at 11835 (offset 3 lines).
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 6 Create a **libtool.patch** file and edit the file.

1. Create a **libtool.patch** file.
   vi libtool.patch
2. Press **i** to enter the edit mode and add the following content:

```
diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
--- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
+++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
@@ -11832,7 +11832,7 @@
 pic_flag=""
 # How to pass a linker flag through the compiler.
-wl=""
+wl="-Wl,"
 # Compiler flag to prevent dynamic linking.
 link_static_flag=""
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the installation.

make -j$threads

```
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/docs'
Making all in examples
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making all in F77
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
Making all in F90
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make  all-am
make[4]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Nothing to be done for 'all-am'.
make[4]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making all in CMakeExtras
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5'
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

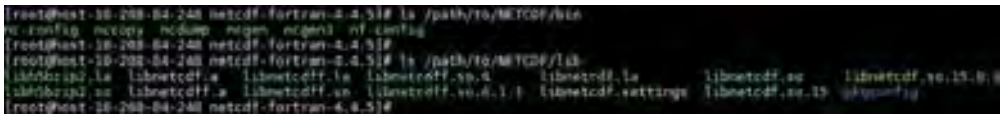
make install

 Copyright © Huawei Technologies Co., Ltd.

```
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F77'
Making install in F90
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make  install-am
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[4]: Nothing to be done for 'install-exec-am'.
make[4]: Nothing to be done for 'install-data-am'.
make[4]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples/F90'
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/examples'
Making install in CMakeExtras
make[1]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.4.5/CMakeExtras'
[root@host-10-208-84-248 netcdf-fortran-4.4.5]#
```

Step 8 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**



----End

1.29.5 Obtaining the Source Code

Procedure

Step 1 Create the SWAN installation directory */path/to/SWAN*.

mkdir -p */path/to/SWAN*

Step 2 Go to the **SWAN** directory.

cd */path/to/SWAN*

Step 3 Download the SWAN installation package.

http://swanmodel.sourceforge.net/download/zip/swan4131.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the SWAN installation package.

wget http://swanmodel.sourceforge.net/download/zip/swan4131.tar.gz --no-check-certificate

Step 4 Use an SFTP tool to upload the SWAN installation package to the */path/to/SWAN* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.29.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the SWAN installation package directory.

```
cd /path/to/SWAN
```

Step 3 Decompress the SWAN installation package.

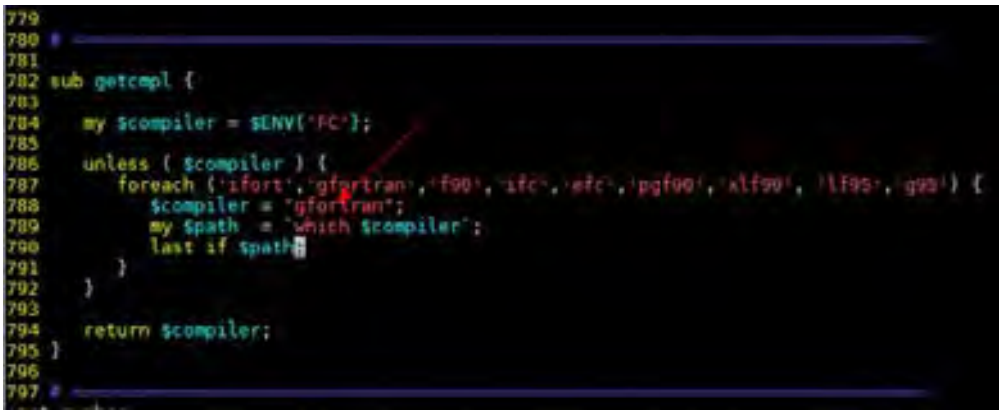
```
tar -zxvf swan4131.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd swan4131
```

Step 5 Modify the **platform.pl** file.

1. Open the **platform.pl** file.
   ```
   vi platform.pl
   ```
2. Press **i** to enter the edit mode and change the content of line 788 to **$compiler = "gfortran"**.



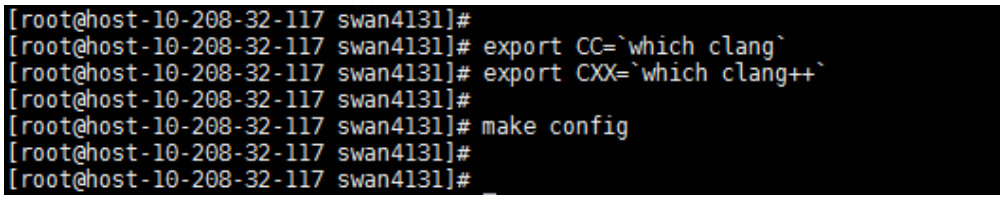
NOTE

After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Create a configuration file.

```
export CC=`which clang`
export CXX=`which clang++`
make config
```



Step 7 Load the environment variables.

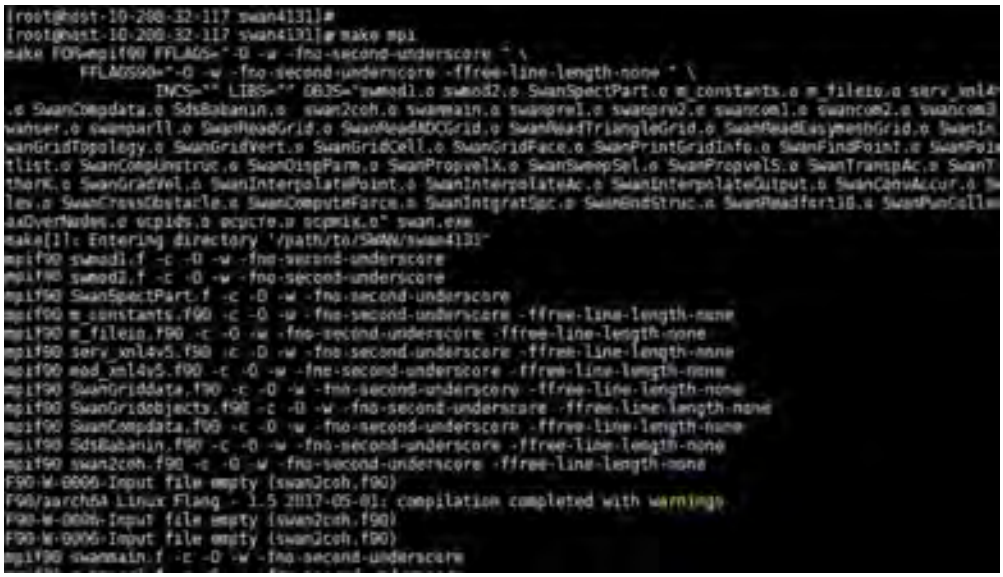
```
export NETCDFROOT=/path/to/NETCDF
```

Step 8 Create a **swan.exe** file.

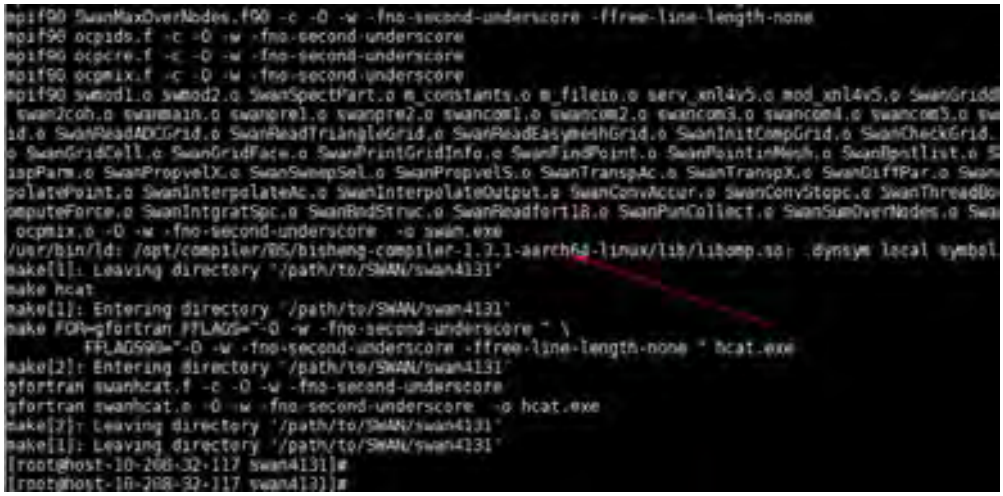
```
make mpi
```

 Copyright © Huawei Technologies Co., Ltd.

Output for the start:

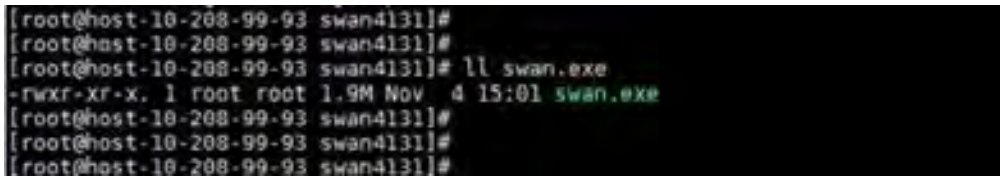


Output for the end:



Step 9 Check whether the **swan.exe** file is generated.

```
ll swan.exe
```



----End

1.29.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test case directory and upload the test case to the directory.

```
mkdir -p /path/to/SWAN/swan-testcse
cd /path/to/SWAN/swan-testcse
```

Step 3 Download the test case and upload it to the directory.

http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz

NOTE

If the server is connected to the Internet, you are advised to run the wget command to obtain the test case.

```
wget http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz --no-check-certif icate
```

Step 4 Decompress the test case file.

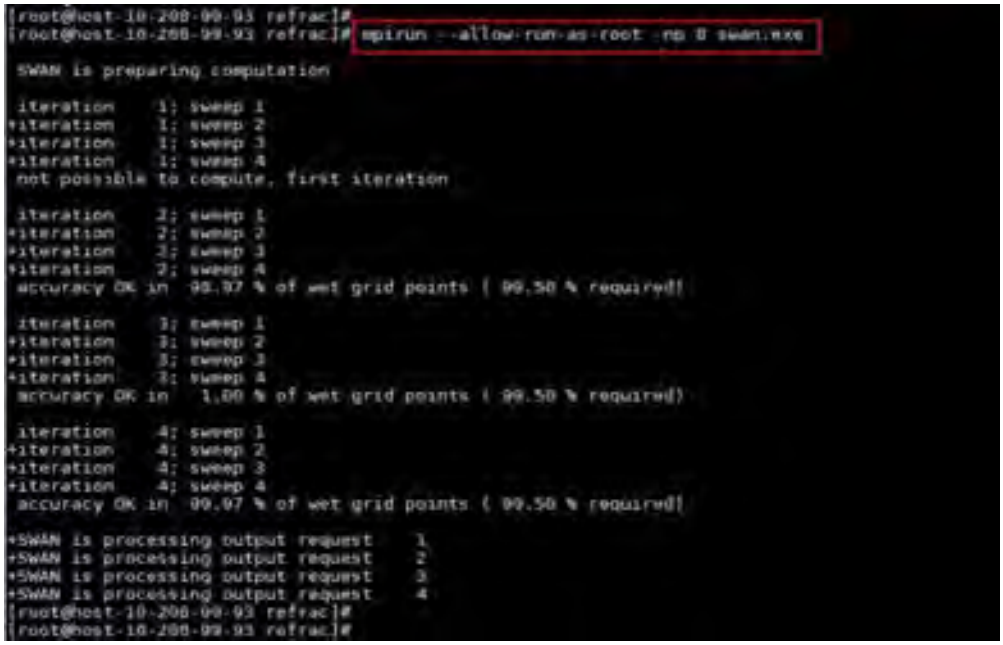
```
tar -zxvf refrac.tar.gz
```

Step 5 Go to the directory and create a soft link.

```
cd refrac
ln -s /path/to/SWAN/swan4131/swan.exe ./
```

Step 6 Run the test case.

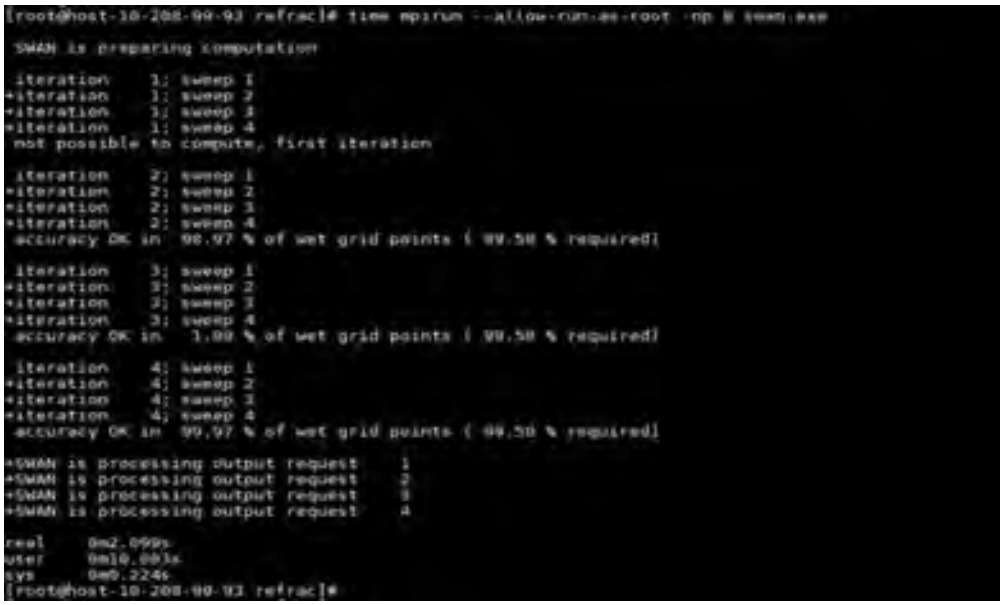
```
cp a11refr.swn INPUT
mpirun --allow-run-as-root -np 8 swan.exe
```




```
time mpirun --allow-run-as-root -np 8 swan.exe
```

The following figure shows an example of the CPU running time.

 Copyright © Huawei Technologies Co., Ltd.



NOTE

- In the preceding command, the **-np** parameter indicates the total number of processes used in the test. Run the **lscpu** command and set this parameter based on your requirements.
- The **real** time (unit: s) in the return result indicates the time. A smaller value indicates higher performance.

----End

1.30 WAVEWATCH III 6.07.1 Porting Guide (openEuler 21.03)

1.30.1 Introduction

WAVEWATCH III is a community wave modeling framework that includes the latest scientific advances in wave modeling and dynamics. WAVEWATCH III solves the random phase spectral action density balance equation for wavenumber-direction spectra. The model includes options for shallow-water (surf zone) applications, as well as wetting and drying of grid points.

For more information about WAVEWATCH III, visit the **WAVEWATCH III official website**.

Programming language: Fortran

Brief description: third-generation community wave modeling framework.

Open source license: custom open source license

Recommended Software Version

WAVEWATCH III 6.07.1

 Copyright © Huawei Technologies Co., Ltd.

1.30.2 Environment Requirements

Hardware Requirements

Table 1-146 lists the hardware requirements.

Table 1-146 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-147 lists the software requirements.

Table 1-147 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng
Hyper MPI	1.1.1	https://www.hikunpeng.com/en/developer/hpc/hypermpi
WAVEWATCH III	6.07.1	https://github.com/NOAA-EMC/WW3/releases/tag/6.07.1
HDF5	1.10.1	http://hdfgroup.org/HDF5/
NETCDF-C	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NETCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
ParMETIS	4.0.3	http://pkgs.freefem.org/ParMetis-4.0.3.tar.gz
Test cases	mww3_data_01	-

OS Requirements

Table 1-148 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-148 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://repo.openeuler.org/openEuler-21.03/**
Kernel	4.14.0-115	Included in the OS image

1.30.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WAVEWATCH III software porting.

Table 1-149 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/hypermpi**	Installation path of Hyper MPI.	See **Hyper MPI User Guide**.
2	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
3	**/path/to/HDF5**	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	**/path/to/NETCDF**	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	**/path/to/METIS**	Installation path of ParMETIS.	
6	**/path/to/WAVEWATCH**	Installation path of WAVEWATCH III.	
7	**/path/to/WW3_TEST**	Test path of WAVEWATCH III.	

1.30.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

1.30.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-150 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI User Guide**.
3	Installing HDF5	See **1.30.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.30.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.30.4.4 Installing NetCDF-Fortran**.
6	Installing ParMETIS	See **1.30.4.5 Installing ParMETIS**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hypermpi/hmpi/include
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

 Copyright © Huawei Technologies Co., Ltd.

```
source ~/.bashrc
```

----End

1.30.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

NOTE

If the server has multiple MPIs, **CC**, **FC**, and **CXX** in the preceding command can be represented using absolute paths.

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify line 11835 in the file.
   ```
   wl="-Wl,"
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.30.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.4.1.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap
```

Step 5 Perform the installation.

```
make -j
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.30.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Decompress the file.

```
cd netcdf-fortran-4.4.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-
static=yes --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -
Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc FC=mpif90
CXX=mpicxx
```

Step 6 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify lines 10118 and 10270 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.30.4.5 Installing ParMETIS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ParMETIS installation package.

```
tar -xvf parmetis-4.0.3.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd parmetis-4.0.3
```

Step 4 Modify the **metis/include/metis.h** file.

1. Open the **metis/include/metis.h** file.
   ```
   vi metis/include/metis.h
   ```
2. Press **i** to enter the insert mode and modify line 33 in the **metis/include/metis.h** file.
   ```
   #define IDXTYPEWIDTH 64
   ```

   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Install CMake.

```
yum install cmake -y
```

NOTE

If an error is reported during the installation, modify the Yum configuration file to change the Python version.

Step 6 Go to the **metis** directory and compile and install ParMETIS.

```
cd metis/
make config shared=1 prefix=/path/to/METIS
make install
```

Step 7 Go to the upper-layer directory and perform the compilation and installation.

```
cd ../
```

Add the following information in bold to the **CMakeLists.txt** file in the current directory:

```
# List of paths that the compiler will search for header files.
add_compile_options(-fPIC)
```

Save the file and exit.

```
make config shared=1 prefix=/path/to/METIS
make install
```

----End

1.30.5 Obtaining the Source Code

Procedure

Step 1 Download the WAVEWATCH III installation package.

https://polar.ncep.noaa.gov/waves/wavewatch/

Step 2 Use an SFTP tool to upload the WAVEWATCH III installation package to the */path/to/WAVEWATCH* directory on the server.

----End

1.30.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the WAVEWATCH III installation directory.

```
cd /path/to/WAVEWATCH
```

Step 3 Decompress the WAVEWATCH III source package.

```
tar -xvf WW3-6.07.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd WW3-6.07.1
```

Step 5 Create an environment variable file.

1. Create an environment variable file.

   ```
   vi env_ww3.sh
   ```

2. Press **i** to enter the edit mode and add the following content:

   ```
   ############## clang + hypermpi ##################
   export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
   export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
   export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:
   $LD_LIBRARY_PATH############## netcdf ##################
   export NETCDF_DIR=/path/to/NETCDF
   export NETCDF_LIBDIR=$NETCDF_DIR/lib
   export NETCDF_INCDIR=$NETCDF_DIR/include
   export PATH=$NETCDF_DIR/bin:$PATH
   export LD_LIBRARY_PATH=$NETCDF_LIBDIR:$LD_LIBRARY_PATH
   export WWATCH3_NETCDF=NC4
   export NETCDF_CONFIG=$NETCDF_DIR/bin/nc-config
   ############## metis ##################
   export METIS_PATH=/path/to/METIS
   export PATH=$METIS_PATH/bin:$PATH
   export LD_LIBRARY_PATH=$METIS_PATH/lib:$LD_LIBRARY_PATH
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Load the environment variables.

```
source env_ww3.sh
```

Step 7 Configure the WAVEWATCH III parameters (pay attention to the information in bold).

```
./model/bin/w3_setup model/
******************************
***   WAVEWATCH III setup    ***
******************************
[INFO] local env file wwatch3.env created in /home/WW3-6.07.1/model/bin/wwatch3.env
Setup file /home/WW3-6.07.1/model/bin/wwatch3.env not found
Default set up :
Printer (listings)       : printer
Auxiliary FORTRAN comp.  : gfortran
Auxiliary C compiler     : gcc
Scratch directory        : /path/to/WAVEWATCH/WW3-6.07.1/model/tmp
```

 Copyright © Huawei Technologies Co., Ltd.

```
Save sources            : yes
Save listings           : yes
Update settings ? [y/n] y
Creating new set-up :
Printer for listings [printer] :
Auxiliary FORTRAN compiler [gfortran] :flang
Auxiliary C compiler [gcc] :clang
Scratch space [/home/wdy/WW3-6.07.1/model/tmp] :
Save source code files (*.f)  [yes] :
Save listing files  [yes] :
Modified set up :
Printer (listings)      : printer
Auxiliary FORTRAN comp.  : flang
Auxiliary C compiler     : clang
Scratch directory       : /path/to/WAVEWATCH III/WW3-6.07.1/model/tmp
Save sources            : yes
Save listings           : yes
New settings OK ? [y/n] y
Setup makefile for auxiliary programs
......
Create required model subdirectories
Finished setting up WAVEWATCH III
```

----End

1.30.7 Running and Verification

NOTE

- Determine the number of running processes based on the node configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the nodes.
  - Enable SSH password-free login between the nodes.
  - Disable the firewall and ensure that clients and servers are not disconnected.

Running and Verification

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory and go to it.

```
mkdir -p /path/to/WW3_TEST
cd /path/to/WW3_TEST
```

Step 3 Download the test case file and copy it to the */path/to/WW3_TEST* directory.

Step 4 Decompress the file.

```
tar -xvf ww3_from_ftp.v6.07.tar.gz
cd cases
```

Step 5 Set the environment variables.

```
source /path/to/WAVEWATCH/WW3-6.07.1/env_ww3.sh
export ww3_dir=/path/to/WAVEWATCH/WW3-6.07.1/model
```

Step 6 Modify the **mww3_case_01** file.

1. Open the **mww3_case_01** file.

   ```
   vi mww3_case_01
   ```

2. Press **i** to enter the edit mode and modify the **mww3_case_01** file.

   Line 61:

   ```
   #cd /scratch1/portfolios/NCEPDEV/ocean/noscrub/Henrique.Alves/WW3_CASES/cases
   ```

   Line 65:

```
#module load intel mpt grads
```

Line 72:

```
compstr="Gnu"
```

Line 143:

```
#ww3_dir='echo $case_dir | sed 's/\/cases\/*//g''
```

Line 138:

```
proc=8
```

Line 420:

```
mpirun --allow-run-as-root --mca btl ^openib -np $proc --hostfile /path/to/WW3_TEST/cases/hostfile $path_e/ww3_multi
```

NOTE

After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. If you delete **--hostfile /path/to/WW3_TEST/cases/hostfile** in line 420, only one node is running.

NOTE

Set the **proc** parameter in line 138 based on the number of CPUs. You can also delete **-np $proc** in line 420 to use the maximum number of cores by default.

5. If the result of a dual-node test case is similar to that of a single-node test case (run the **top** command to monitor the second node and no running status is found on the second node), perform the following operations: (Alternative solution)

   Modify line 138 as follows:

   ```
   #proc=8
   ```

   Modify line 420 as follows:

   ```
   mpirun --allow-run-as-root --mca btl ^openib -np 16 -N 8 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH $proc --hostfile    /path/to/WW3_TEST/cases/hostfile $path_e/ww3_multi
   ```

   NOTE

   The **-np** parameter indicates the total number of processes, and the **-N** parameter indicates the number of processes used by each node. Set this parameter based on the server configuration.

Step 7 Modify the **comp.Gnu** and **link.Gnu** files.

1. Open the **comp.Gnu** file.

   vi ../WW3-6.07.1/model/bin/comp.Gnu

2. Press **i** to enter the edit mode.

   In line 93, change **comp=gfortran** to **comp=flang**.

   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Open the **link.Gnu** file.

   vi ../WW3-6.07.1/model/bin/link.Gnu

 Copyright © Huawei Technologies Co., Ltd.

5. Press **i** to enter the edit mode.

   In line 95, change **comp=gfortran** to **comp=flang**.

6. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Create a **hostfile** file.

1. Create **hostfile**.

```
vi hostfile
Node1
Node2
```

   NOTE

   Node1 and **Node2** are host names.

2. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

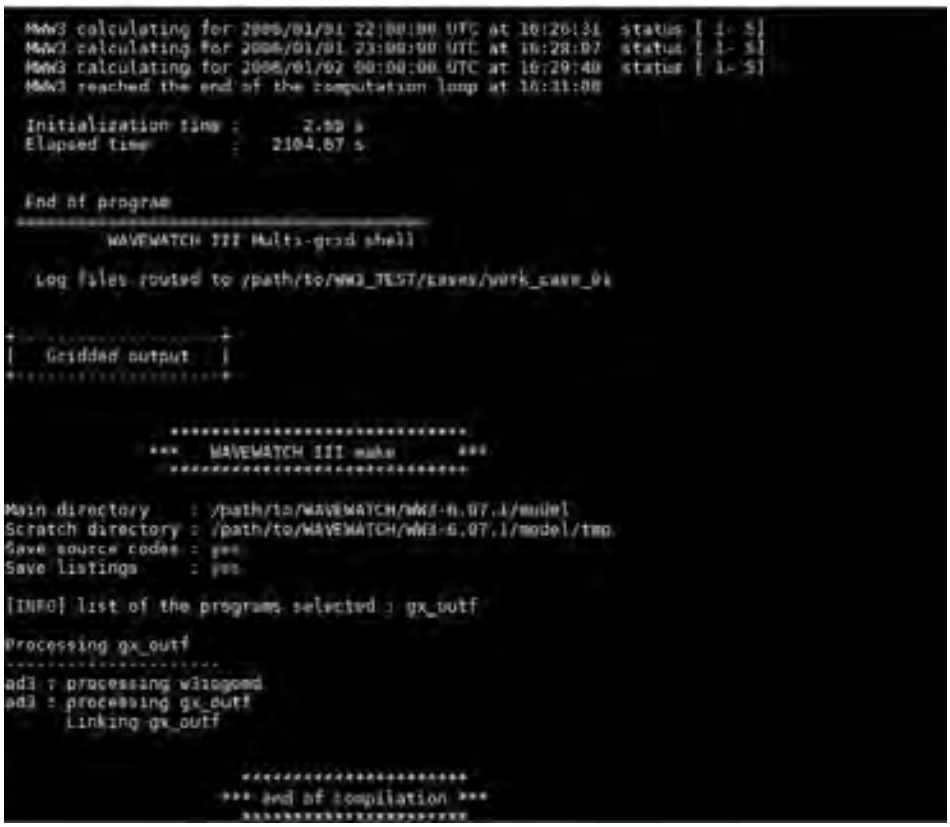
Step 9 In the window of **Node1**, run the following command to start the WAVEWATCH III program:

```
./mww3_case_01
```

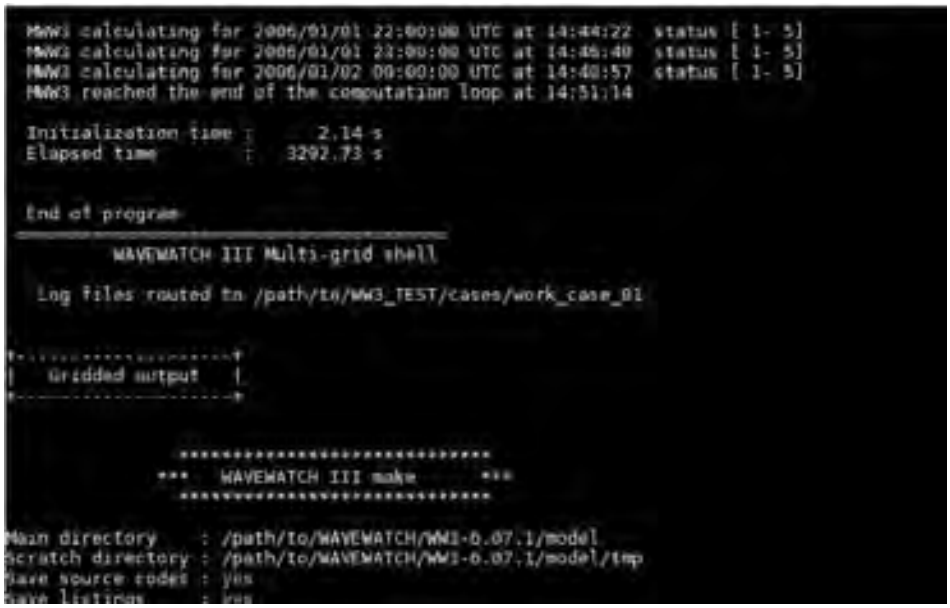
If the information shown in the following figure is displayed when the WAVEWATCH III program is complete, the WAVEWATCH III program completes properly.

Check the value of **Elapsed time** (unit: s) in the **work_case_01/log.mww3** log file. A smaller value indicates higher performance.

- Example of the dual-node running result



- Example of the single-node running result



----End

1.31 WRF 4.2 Porting Guide (openEuler 21.03)

1.31.1 Introduction

The Weather Research and Forecasting (WRF) Model is designed for fine-scale weather simulation and forecasting. It is one of major high-performance computing (HPC) application scenarios.

For more information about WRF, visit the **WRF official website**.

Programming languages: C/Fortran

Brief description: mesoscale weather forecasting model.

Open source license: public domain

Recommended Software Version

WRF v4.2

1.31.2 Environment Requirements

Hardware Requirements

Table 1-151 lists the hardware requirements.

Table 1-151 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-152 lists the software requirements.

Table 1-152 Software requirements

Item	Version	Download URL
ZLIB	1.2.11	https://udomain.dl.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
PNETCDF	1.9.0	http://cucis.ece.northwestern.edu/projects/PnetCDF
NETCDF-C	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NETCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
WRF	4.2	https://github.com/wrf-model/WRF/archive/refs/tags/v4.2.tar.gz
BiSheng Compiler	1.3.3	https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi
Test cases	conus12km	https://www2.mmm.ucar.edu/wrf/src/conus12km.tar.gz

OS Requirements

Table 1-153 lists the OS requirements.

Table 1-153 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

1.31.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WRF software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-154 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/hypermpi	Installation path of Hyper MPI.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/PNETCDF	Installation path of PnetCDF.	
3	/path/to/HDF5	Installation path of HDF5.	
4	/path/to/NETCDF	Installation path of NetCDF.	
		Installation path of NetCDF-Fortran.	
5	/path/to/ZLIB	Installation path of zlib.	
6	/path/to/WRF	Installation path of WRF.	
7	/path/to/conus12km	Test path of WRF.	

1.31.4 Configuring the Compilation Environment

1.31.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-155 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing zlib	See **1.31.4.2 Installing zlib**.
4	Installing PnetCDF	See **1.31.4.3 Installing PnetCDF**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
5	Installing HDF5	See **1.31.4.4 Installing HDF5**.
6	Installing NetCDF-C	See **1.31.4.5 Installing NetCDF-C**.
7	Installing NetCDF-Fortran	See **1.31.4.6 Installing NetCDF-Fortran**.

1.31.4.2 Installing zlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Add the environment variables.

```
export CC=clang
export CXX=clang++
export FC=flang
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

NOTE

After that, you do not need to add them again during subsequent installation steps.

Step 3 Decompress the installation package.

```
tar -vxf zlib-1.2.11.tar.gz
cd zlib-1.2.11
```

Step 4 Perform the compilation and installation.

```
./configure -prefix=/path/to/ZLIB
make
make install
```

Step 5 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/ZLIB/lib:$LD_LIBRARY_PATH
```

----End

1.31.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd parallel-netcdf-1.9.0
```

Step 4 Perform the configuration and installation.

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

 Copyright © Huawei Technologies Co., Ltd.

```
make
make install
```

Step 5 Load the environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.31.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Set the environment variables.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.31.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.4.1.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L$HDF5/lib" CPPFLAGS="-I$HDF5/include" CC=mpicc
--disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the compilation and installation.

```
make
make install
```

Step 6 After the installation is complete, load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.31.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$HDF5/include -I$NETCDF/include" LDFLAGS="-L
$HDF5/lib -L$NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-static=yes --enable-shared
CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion"
FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify lines 10118 and 10270 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 After the installation is complete, load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.31.5 Obtaining the Source Code

Procedure

Step 1 Download the WRF installation package **WRF-4.2.tar.gz**.

https://github.com/wrf-model/WRF/archive/refs/tags/v4.2.tar.gz

Step 2 Use an SFTP tool to upload the WRF installation package to the */path/to/WRF* directory on the server.

----End

1.31.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the WRF installation package.

```
tar -vxf WRF-4.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd WRF-4.2
```

Step 4 Modify the **arch/configure.defaults** file.

1. Open the **arch/configure.defaults** file.
   ```
   vi arch/configure.defaults
   ```
2. Press **i** to enter the edit mode and add the following content to the end of line 1978:

```
################################################### #########
#ARCH   Linux   aarch64,clang HYPERMPI#serial smpar dmpar dm+sm
DESCRIPTION     =       CLANG ($SFC/$SCC)
DMPARALLEL      =       1
OMPCPP          =       -D_OPENMP
OMP             =       -fopenmp
OMPCC           =       -fopenmp
SFC             =       flang
SCC             =       clang
CCOMP           =       clang
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC              =       CONFIGURE_FC
CC              =       CONFIGURE_CC
LD              =       $(FC)
RWORDSIZE       =       CONFIGURE_RWORDSIZE
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR  -DWRF_USE_CLM
CFLAGS_LOCAL    =       -mcpu=native -w -O3 -c -march=armv8.2-a
LDFLAGS_LOCAL   =
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 -funroll-loops -march=armv8.2-a
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       -g # -fbacktrace -ggdb-fcheck=bounds,do,mem,pointer -ffpe-
trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-0
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian
```

 Copyright © Huawei Technologies Co., Ltd.

```
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       -mcpu=native $(OMP) $(FCBASEOPTS_NO_G)
MODULE_SRCH_FLAG =
TRADFLAG        =       -traditional
CPP             =       /lib/cpp -P
AR              =       ar
ARFLAGS         =       ru
M4              =       m4 -G
RANLIB          =       ranlib
RLFLAGS         =
CC_TOOLS        =       $(SCC)
```

NOTE

After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the **phys/module_mp_SBM_polar_radar.F** file.

1. Open the **phys/module_mp_SBM_polar_radar.F** file.
   ```
   vi phys/module_mp_SBM_polar_radar.F
   ```
2. Press **i** to go to the edit mode and add the following content to line 452 of the file:
   ```
   external :: derf
   ```

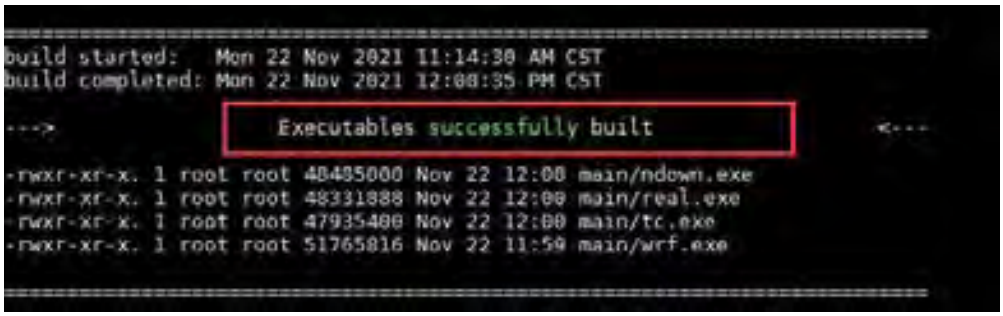
   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
echo 4 | ./configure
./compile  em_real 2>&1 | tee -a compile.log
```

If the following information is displayed, the WRF program is installed successfully.



----End

 Copyright © Huawei Technologies Co., Ltd.

1.31.7 Running and Verification

NOTE

- Determine the number of running processes based on the server configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the servers.
  - Enable password-free SSH login between the servers.
  - Disable the firewall and ensure that clients and servers are not disconnected.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the WRF test case **conus12km**.

https://www2.mmm.ucar.edu/wrf/src/conus12km.tar.gz

Step 3 Upload **conus12km.tar.gz** to the */path/to/conus12km* directory.

Step 4 Decompress the installation package.

```
tar -vxf conus12km.tar.gz
```

Step 5 Copy the binary file to the working directory and then go to the working directory.

```
cp -r /path/to/WRF/WRF-4.2/run/* conus12km
cd conus12km
ln -sf /path/to/WRF/WRF-4.2/main/*.exe ./
```

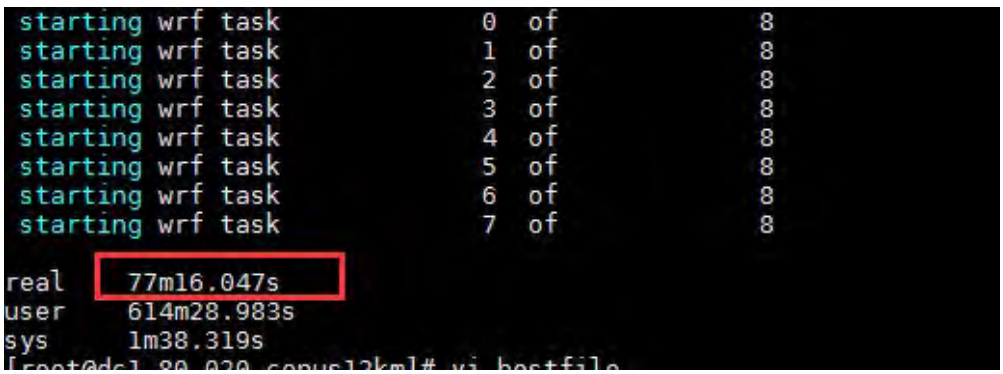
NOTE

When the system asks you whether to overwrite the original file, enter **no**.

Step 6 Start the WRF program.

```
time mpirun --allow-run-as-root -np 8 ./wrf.exe
```

Check the **real** value (unit: s). A smaller value indicates higher performance.



NOTE

The **-np** parameter indicates the total number of used processes.

Step 7 Check whether the program stops properly.

```
less rsl.out.0000
```

If the following information is displayed at the end of **rsl.out.0000**, the program is successfully executed and stops properly.

 Copyright © Huawei Technologies Co., Ltd.

Step 8 The environment variable configuration for multiple nodes is the same as that for a single node. Configure the **hostfile** file in the working directory.

1. Open the **hostfile** file.
   ```
   vi hostfile
   ```
2. Press **i** to enter the edit mode and configure the name of each node in the **hostfile** file.
   ```
   node1
   node2
   ```

   NOTE

   Set them based on the actual node names (host names).
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Start the running.
   ```
   time mpirun --allow-run-as-root -np 16 -N 8 -hostfile hostfile ./wrf.exe
   ```

   NOTE

   The **-N** parameter indicates the number of processes running on each server.

----End

1.32 CESM 2.1.1 Porting Guide (openEuler 21.03)

1.32.1 Introduction

The Community Earth System Model (CESM) is a fully coupled numerical simulation of the Earth system consisting of atmospheric, ocean, ice, land surface, carbon cycle, and other components. CESM provides state-of-the-art computer simulations of Earth's past, present, and future climate states.

For more information about CESM, visit the **CESM official website**.

Programming language: C++

Brief description: A fully-coupled global climate model.

Open source license: Artistic License 2.0

Recommended Software Version

CESM 2.1.1

1.32.2 Environment Requirements

Hardware Requirements

Table 1-156 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-156 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-157 lists the software requirements.

Table 1-157 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
CESM	2.1.1	https://github.com/ESCOMP/CESM/archive/refs/tags/cesm2.1.1.tar.gz
OpenBLAS	0.3.10	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.10.tar.gz
PnetCDF	1.12.2	https://parallel-netcdf.github.io/Release/pnetcdf-1.12.2.tar.gz
NetCDF	4.7.4	https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.4.tar.gz
NetCDF-Fortran	4.5.3	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.5.3.tar.gz
HDF5	1.12.0	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.12/hdf5-1.12.0/src/hdf5-1.12.0.tar.gz
curl	7.70.0	https://curl.se/download/curl-7.70.0.tar.gz
SZIP	2.1.1	https://distfiles.macports.org/szip/szip-2.1.1.tar.gz
zlib	1.2.11	https://udomain.dl.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz

OS Requirements

Table 1-158 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-158 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	Included in the OS image

1.32.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CESM software porting.

Table 1-159 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/bisheng	Installation path of BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/HMPI	Installation path of Hyper MPI.	
3	/path/to/SZIP	Installation path of szip.	
4	/path/to/ZLIB	Installation path of zlib.	
5	/path/to/CURL	Installation path of curl.	
6	/path/to/NETCDF	Installation path of NetCDF-C and NetCDF-Fortran.	
7	/path/to/HDF5	Installation path of HDF5.	
8	/path/to/PNETCDF	Installation path of PnetCDF.	
9	/path/to/OPENBLAS	Installation path of OpenBLAS.	
10	/path/to/CESM	Test path of CESM.	

1.32.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

1.32.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-160 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Setting the BiSheng Compiler and Hyper MPI Environment Variables	See **1.32.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables**.
4	Installing zlib	See **1.32.4.3 Installing zlib**.
5	Installing szip	See **1.32.4.4 Installing szip**.
6	Installing curl	See **1.32.4.5 Installing curl**.
7	Installing HDF5	See **1.32.4.6 Installing HDF5**.
8	Installing PnetCDF	See **1.32.4.7 Installing PnetCDF**.
9	Installing NetCDF-C	See **1.32.4.8 Installing NetCDF-C**.
10	Installing NetCDF-Fortran	See **1.32.4.9 Installing NetCDF-Fortran**.
11	Installing OpenBLAS	See **1.32.4.10 Installing OpenBLAS**.

1.32.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/bisheng/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bisheng/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CC=$MPICC
export CXX=$MPICXX
```

 Copyright © Huawei Technologies Co., Ltd.

```
export FC=$MPIFC
export F77=$MPIFC
```

----End

1.32.4.3 Installing zlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the zlib installation package.

```
tar -xzvf zlib-1.2.11.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd zlib-1.2.11
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/ZLIB
```

Step 5 Perform the compilation and installation.

```
make -j 64
make install
```

Step 6 Load the zlib environment variables.

```
export LD_LIBRARY_PATH=/path/to/ZLIB/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.4 Installing szip

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the szip installation package.

```
tar -zxvf szip-2.1.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd szip-2.1.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/SZIP
```

Step 5 Perform the compilation and installation.

```
make -j 64
make install
```

Step 6 Load the szip environment variables.

```
export LD_LIBRARY_PATH=/path/to/SZIP/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.32.4.5 Installing curl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the curl installation package.

```
tar -zxvf curl-7.70.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd curl-7.70.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/CURL --with-zlib=/path/to/ZLIB
```

Step 5 Perform the compilation and installation.

```
make -j 64
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/CURL/bin:$PATH
export LD_LIBRARY_PATH=/path/to/CURL/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.6 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.12.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.12.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpifort --with-zlib=/path/to/ZLIB --with-szlib=/path/to/SZIP
```

Step 5 Modify the **libtool** file.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j 64
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.7 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar xf pnetcdf-1.12.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pnetcdf-1.12.2
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" CXXFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" FFLAGS="-O3 -fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpifort
```

Step 5 Perform the compilation and installation.

```
make -j 64
make install
```

Step 6 Load the PnetCDF environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.8 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.4
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/CURL/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -L/path/to/CURL/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/PNETCDF/lib -Wl,-rpath=/path/to/CURL/lib"
export CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion"
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static -enable-pnetcdf --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpifort
```

Step 5 Perform the compilation and installation.

```
make -j 64
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.9 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar xf netcdf-fortran-4.5.3.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.5.3
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to /NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib" CFLAGS="-fPIC -L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="-fPIC -L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 5 Modify the **libtool** file.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and change all **wl=""** to **wl="-Wl,"**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j 64
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.32.4.10 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar zxvf OpenBLAS-0.3.10.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.10/
export CC=clang
export CXX=clang++
export FC=flang
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Perform the compilation and installation.

```
make FC=flang
make prefix=/path/to/OPENBLAS install
```

Step 5 Clear the compilation parameters.

```
unset CC CXX FC F77 CPPFLAGS LDFLAGS CFLAGS CXXFLAGS FCFLAGS
```

----End

1.32.5 Obtaining the Source Code

Procedure

Step 1 Download the CESM installation package **cesm-release-cesm2.1.1.tar.gz**.

https://github.com/ESCOMP/CESM/archive/refs/tags/cesm2.1.1.tar.gz

Step 2 Use an SFTP tool to upload the CESM installation package to the */path/to/CESM* directory on the server.

----End

1.32.6 Compilation and Installation

Prerequisites

Configure an external network so that the server can be connected to the Internet.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CESM installation package and go to the directory generated after the decompression.

```
tar xf CESM-cesm2.1.1.tar.gz
cd /path/to/CESM/CESM-cesm2.1.1
```

Step 3 Install Environment Modules, Git, and SVN.

```
yum install git svn environment-modules -y
```

Step 4 Add the verification information.

```
svn ls https://svn-ccsm-models.cgd.ucar.edu/ww3/release_tags
```

Step 5 Install other CESM components.

```
./manage_externals/checkout_externals
```

Step 6 Check whether the required components are installed.

```
./manage_externals/checkout_externals -S
```

The installation is successful if the following information in is displayed:

```
Processing externals description file : Externals.cfg
Processing externals description file : Externals_CLM.cfg
Processing externals description file : Externals_POP.cfg
Processing externals description file : Externals_CISM.cfg
Checking status of externals: clm, fates, ptclm, mosart, ww3, cime, cice, pop, cvmix, marbl, cism,
source_cism, rtm, cam,
./cime
./components/cam
```

 Copyright © Huawei Technologies Co., Ltd.

```
./components/cice
./components/cism
./components/cism/source_cism
./components/clm
./components/clm/src/fates
./components/clm/tools/PTCLM
./components/mosart
./components/pop
./components/pop/externals/CVMix
./components/pop/externals/MARBL
./components/rtm ./components/ww3
```

----End

1.32.7 Running and Verification

Single Node

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Modify the **config_machines.xml** file.

1. Open the **config_machines.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.1/cime/config/cesm/machines/config_machines.xml**
2. Press **i** to enter the insert mode and modify the following information in bold:

```
<machine MACH="centos7-linux">
<DESC>
Example port to centos7 linux system with gcc, netcdf, pnetcdf and mpich
using modules from http://www.admin-magazine.com/HPC/Articles/Environment-Modules
</DESC>
<NODENAME_REGEX>n2</NODENAME_REGEX>
<OS>LINUX</OS>
<PROXY> https://howto.get.out </PROXY>
<COMPILERS>gnu</COMPILERS>
<MPILIBS>openmpi</MPILIBS>
<PROJECT>none</PROJECT>
<SAVE_TIMING_DIR> </SAVE_TIMING_DIR>
<CIME_OUTPUT_ROOT>$ENV{HOME}/cesm/scratch</CIME_OUTPUT_ROOT>
<DIN_LOC_ROOT>$ENV{HOME}/cesm/inputdata</DIN_LOC_ROOT>
<DIN_LOC_ROOT_CLMFORC>$ENV{HOME}/cesm/inputdata/lmwg</DIN_LOC_ROOT_CLMFORC>
<DOUT_S_ROOT>$ENV{HOME}/cesm/archive/$CASE</DOUT_S_ROOT>
<BASELINE_ROOT>$ENV{HOME}/cesm/cesm_baselines</BASELINE_ROOT>
<CCSM_CPRNC>$ENV{HOME}/cesm/tools/cime/tools/cprnc/cprnc</CCSM_CPRNC>
<GMAKE>make</GMAKE>
<GMAKE_J>8</GMAKE_J>
<BATCH_SYSTEM>none</BATCH_SYSTEM>
<SUPPORTED_BY>me@my.address</SUPPORTED_BY>
<MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
<MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
<PROJECT_REQUIRED>FALSE</PROJECT_REQUIRED>
<mpirun mpilib="default">
<executable>mpirun</executable>
<arguments>
<arg name="ntasks">--allow-run-as-root --mca btl ^openib -np {{ total_tasks }} </arg>
</arguments>
</mpirun>
<module_system type="module" allow_error="true">
<init_path lang="perl">/usr/share/Modules/init/perl.pm</init_path>
<init_path lang="python">/usr/share/Modules/init/python.py</init_path>
<init_path lang="csh">/usr/share/Modules/init/csh</init_path>
<init_path lang="sh">/usr/share/Modules/init/sh</init_path>
<cmd_path lang="perl">/usr/bin/modulecmd perl</cmd_path>
<cmd_path lang="python">/usr/bin/modulecmd python</cmd_path>
<cmd_path lang="sh">module</cmd_path>
<cmd_path lang="csh">module</cmd_path>
<modules>
```

 Copyright © Huawei Technologies Co., Ltd.

```
<command name="purge"/>
</modules>
</module_system>
<environment_variables>
<env name="NETCDF_PATH">/path/to/NETCDF</env>
<env name="NETCDF_FORTRAN_PATH">/path/to/NETCDF</env>
<env name="PNETCDF_PATH">/path/to/PNETCDF</env>
<env name="NETCDF_C_PATH">/path/to/NETCDF</env>
<env name="MPI_PATH">/path/to/HMPI</env>
<env name="OMP_STACKSIZE">256M</env>
</environment_variables>
<resource_limits>
<resource name="RLIMIT_STACK">-1</resource>
</resource_limits>
</machine>
```

Table 1-161 Parameters in the **config_machines.xml** file

Parameter	Description
<NODENAME_REGEX>	Set this parameter based on the actual node names.
<COMPILERS>	Set it to **GNU**.
<MPILIBS>	Set it to **OpenMPI**.
<MAX_TASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<MAX_MPITASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<executable>	The default value is **mpiexec**. You need to change it to **mpirun**.
<arguments>	Set it to **--allow-run-as-root --mca btl ^openib -np {{ total_tasks }}**. The system automatically generates the total number of threads.
<environment_variables>	Set the dependency function path based on the installation location.

NOTE

Change the paths based on **Table 1-161**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Modify the **config_compilers.xml** file.

1. Open the **config_compilers.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.1/cime/config/cesm/machines/config_compilers.xml**
2. Press **i** to enter the insert mode and modify the following information in bold:

```
<compiler COMPILER="gnu">
<CFLAGS>
<base>  -O3 -mcpu=tsv110 -Hx,54,0x08 </base>
<append compile_threaded="true"> -fopenmp </append>
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
```

 Copyright © Huawei Technologies Co., Ltd.

```
</append>
<append DEBUG="FALSE"> -O </append>
</CFLAGS>
<CPPDEFS>
<!-- http://gcc.gnu.org/onlinedocs/gfortran/ -->
<append> -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU</append>
</CPPDEFS>
<CXX_LINKER>FORTRAN</CXX_LINKER>
<FC_AUTO_R8>
<base> -fdefault-real-8 </base>
</FC_AUTO_R8>
<FFLAGS>
<!-- -ffree-line-length-none and -ffixed-line-length-none need to be in FFLAGS rather than in
FIXEDFLAGS/FREEFLAGS
so that these are passed to cmake builds (cmake builds don't use FIXEDFLAGS and FREEFLAGS). -->
<base>  -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/
path/to/PNETCDF/include -I/path/to/HDF5/include -I/path/to/CURL/include -I/path/to/
OPENBLAS/include -I/path/to/ZLIB/include </base>
<append compile_threaded="true"> -fopenmp </append>
<!-- Ideally, we would also have 'invalid' in the ffpe-trap list. But at
least with some versions of gfortran (confirmed with 5.4.0, 6.3.0 and
7.1.0), gfortran's isnan (which is called in cime via the
CPRGNU-specific shr_infnan_isnan) causes a floating point exception
when called on a signaling NaN. -->
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds </
append>
<append DEBUG="FALSE"> -O </append>
</FFLAGS>
<FFLAGS_NOOPT>
<base> -O0 </base>
</FFLAGS_NOOPT>
<FIXEDFLAGS>
<base>  -ffixed-form </base>
</FIXEDFLAGS>
<FREEFLAGS>
<base> -ffree-form </base>
</FREEFLAGS>
<HAS_F2008_CONTIGUOUS>FALSE</HAS_F2008_CONTIGUOUS>
<LDFLAGS>
<append compile_threaded="true"> -fopenmp -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/
path/to/HDF5/lib -L/path/to/OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -
lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz -lcurl</append>
</LDFLAGS>
<MPICC> /path/to/HMPI/bin/mpicc  </MPICC>
<MPICXX> /path/to/HMPI/bin/mpicxx </MPICXX>
<MPIFC> /path/to/HMPI/bin/mpifort </MPIFC>
<SCC> /path/to/bisheng/bin/clang </SCC>
<SCXX> /path/to/bisheng/bin/clang++ </SCXX>
<SFC> /path/to/bisheng/bin/flang </SFC>
<SUPPORTS_CXX>TRUE</SUPPORTS_CXX>
<SLIBS>
<append> -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/path/to/
OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -
lhdf5_hl -lhdf5 -lm -lz -lcurl </append>
</SLIBS>
```

Table 1-162 Parameters in the **config_compilers.xml** file

Parameter	Description
<CFLAGS>	Add the **-mcpu=tsv110** parameter.
<FFLAGS>	Add the **-mcpu=tsv110** parameter and link the **include** paths of all dependency libraries.
<MPICC>	Specifies the mpicc path of Hyper MPI.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
<MPICXX>	Specifies the mpicxx location of Hyper MPI.
<MPIFC>	Specifies the mpifort location of Hyper MPI.
<SCC>	Specifies the clang path of the BiSheng Compiler.
<SCXX>	Specifies the clang++ path of the BiSheng Compiler.
<SFC>	Specifies the flang path of the BiSheng Compiler.
<LDFLAGS>	Use the **-L** parameter to link all dependency libraries.
<SLIBS>	Use the **-L** parameter to link all dependency libraries.

NOTE

Change the paths based on **Table 1-162**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Go to the working directory.

```
cd /path/to/CESM/CESM-cesm2.1.1/cime/scripts
```

Step 5 Create a test case.

```
./create_newcase --case mycase --compset X --res f19_g16 --mach=centos7-linux
```

The operation is successful if the following information in is displayed:

Step 6 Download the input data. No offline test case packages are involved. All components are automatically downloaded based on the command line.

```
cd mycase
./check_case
./check_input_data --download
```

NOTE

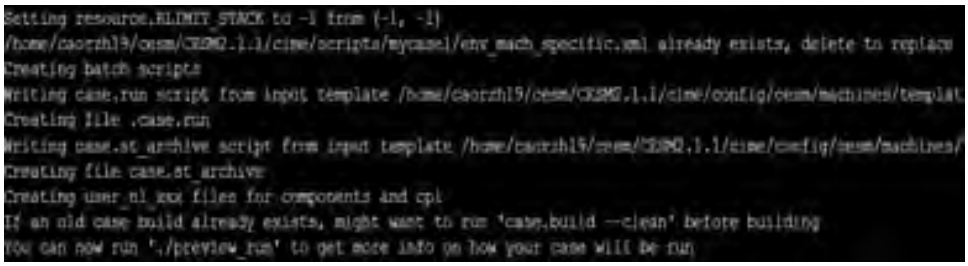
After a module is successfully downloaded, **SUCCESS** is displayed. You do not need to perform any operations before the whole download process automatically completes.

Step 7 Start the CESM program.

```
./case.setup
```

The operation is successful if the following information in is displayed:

 Copyright © Huawei Technologies Co., Ltd.



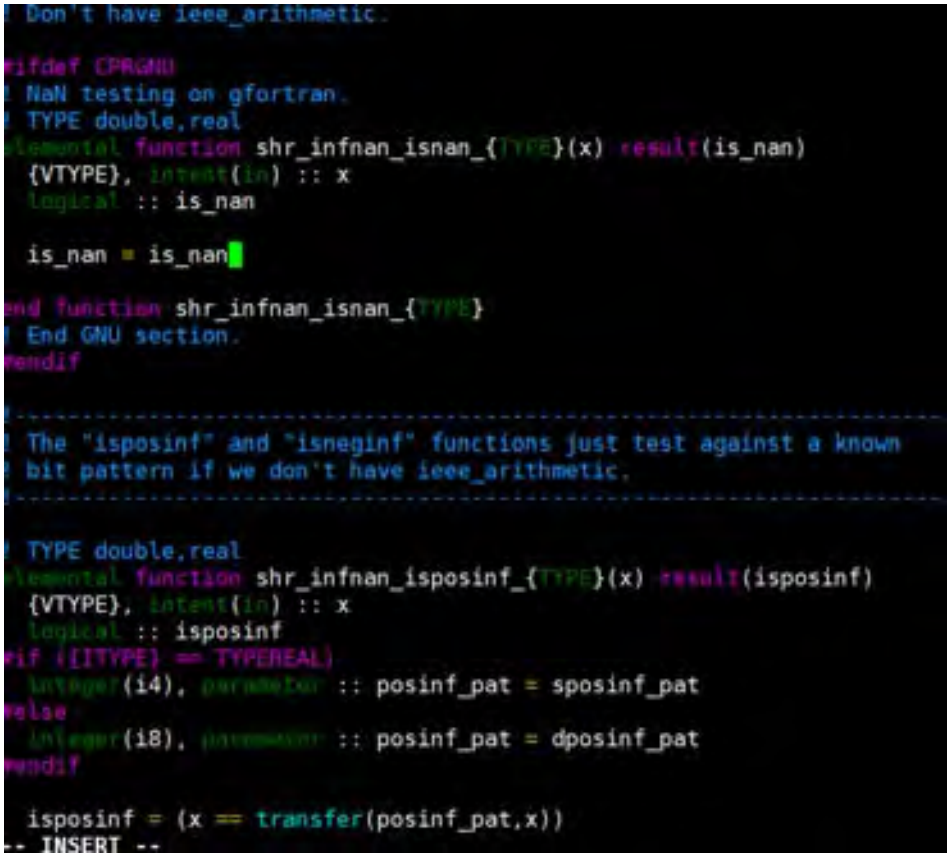
Step 8 Check the compilation options in the **Macro.make** file. The compilation options in the **Macro.make** file must be the same as those in the **configure.compilers.xml** file. The options are <CFLAGS>, <FFLAGS>, <MPICC>, <MPICXX>, <MPIFC>, <SCC>, <SCXX>, <SFC>, <LDFLAGS>, and <SLIBS>.

```
SUPPORTS_CXX := FALSE
ifeq ($(COMPILER),gnu)
FC_AUTO_R8 :=  -fdefault-real-8
HAS_F2008_CONTIGUOUS := FALSE
FFLAGS_NOOPT :=  -O0
SUPPORTS_CXX := TRUE
FFLAGS :=   -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/path/to/PNETCDF/include -I/path/to/HDF5/include -I/path/to/CURL/include -I/path/to/OPENBLAS/include -I/path/to/ZLIB/include
FIXEDFLAGS :=   -ffixed-form
SCC :=  /path/to/bisheng/bin/clang
SFC :=  /path/to/bisheng/bin/flang
MPICC :=  /path/to/HMPI/bin/mpicc
CFLAGS :=   -O3 -mcpu=tsv110 -Hx,54,0x08
MPIFC :=  /path/to/HMPI/bin/mpifort
MPICXX :=  /path/to/HMPI/bin/mpicxx
FREEFLAGS :=  -ffree-form
CXX_LINKER := FORTRAN
SCXX :=  /path/to/bisheng/bin/clang++
endif
ifeq ($(MODEL),pop)
CPPDEFS := $(CPPDEFS)  -D_USE_FLOW_CONTROL
endif
ifeq ($(COMPILER),gnu)
CPPDEFS := $(CPPDEFS)  -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU
SLIBS := $(SLIBS)  -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz -lcurl
ifeq ($(compile_threaded),true)
FFLAGS := $(FFLAGS)  -fopenmp
CFLAGS := $(CFLAGS)  -fopenmp
endif
ifeq ($(DEBUG),TRUE)
FFLAGS := $(FFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds
CFLAGS := $(CFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
endif
ifeq ($(DEBUG),FALSE)
FFLAGS := $(FFLAGS)  -O
CFLAGS := $(CFLAGS)  -O
endif
ifeq ($(compile_threaded),true)
LDFLAGS := $(LDFLAGS)  -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz -lcurl
endif
endif
```

Step 9 Modify the **CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in** file.

1. Open the **CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in** file.
   vi */path/to/CESM*/**CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in**

2. Press **i** to enter the insert mode and modify the content in line 227, as shown by the cursor in the following figure:



NOTE

After performing **Step 9.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

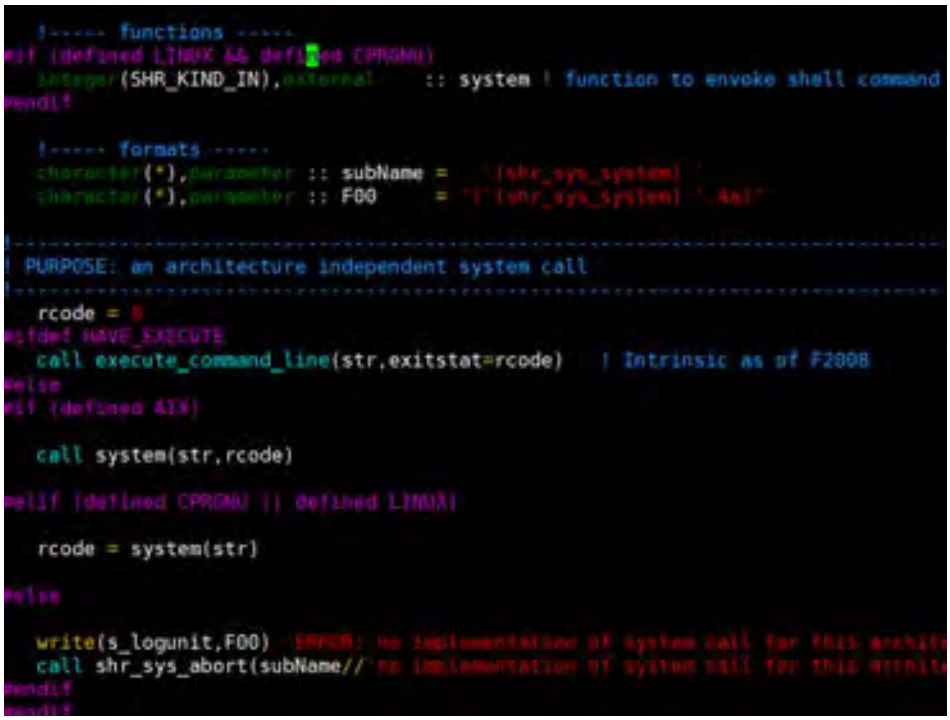
Step 10 Modify the **CESM-release-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90** file.

1. Open the **CESM-release-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90** file.

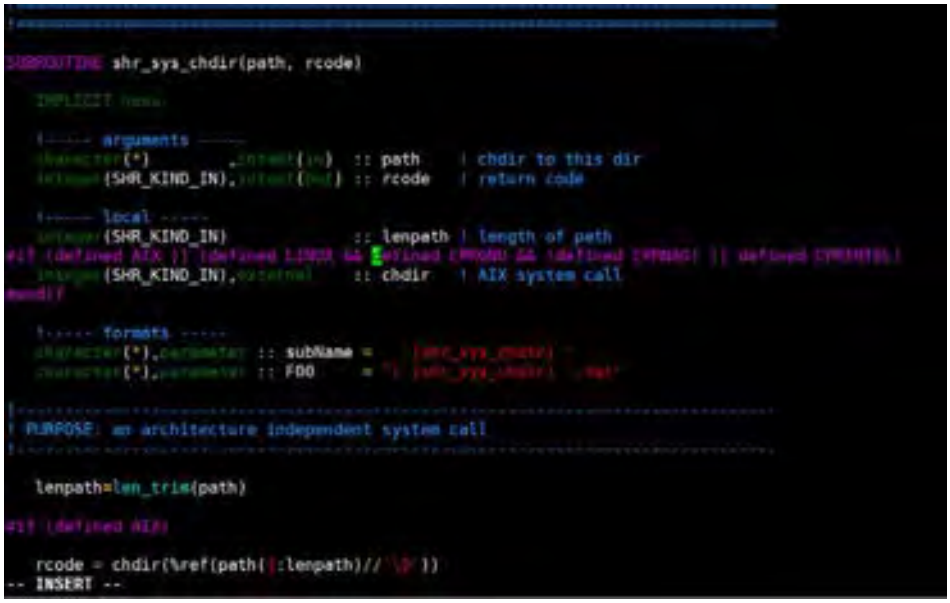
```
vi /path/to/CESM/CESM-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90
```

2. Press **i** to enter the insert mode and modify the file as follows:

   Delete the exclamation mark (!) before **defined CPRGNU** in line 70.



Delete the exclamation mark (!) before **defined CPRGNU** in line 115.



NOTE

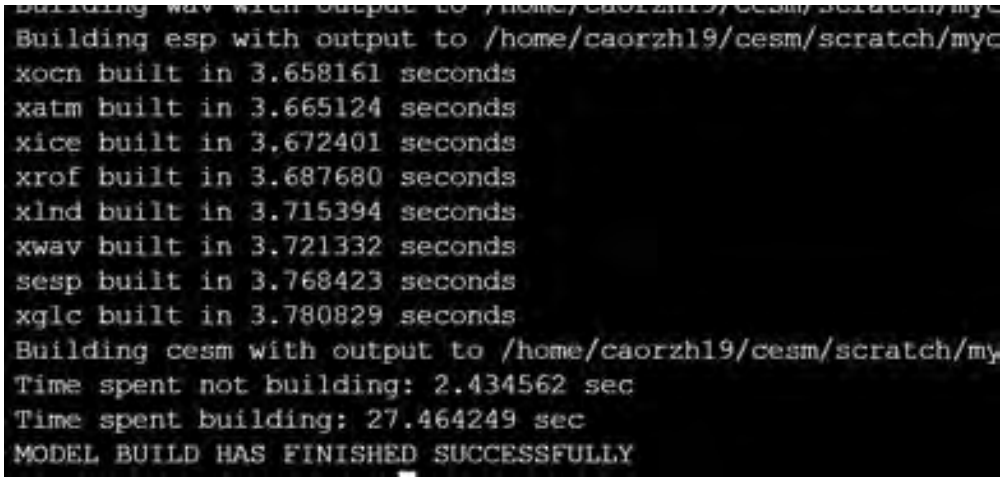
After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Build the **cesm.exe** file.

```
./case.build --skip-provenance-check
```

The operation is successful if the following information in is displayed:



NOTE

You need to soft link CMake3 in the system to CMake and specify the CMake path. If you are the root user and the CMake version is 3.0 or later, ignore **ln -sf /usr/bin/cmake3 /usr/bin/cmake**.

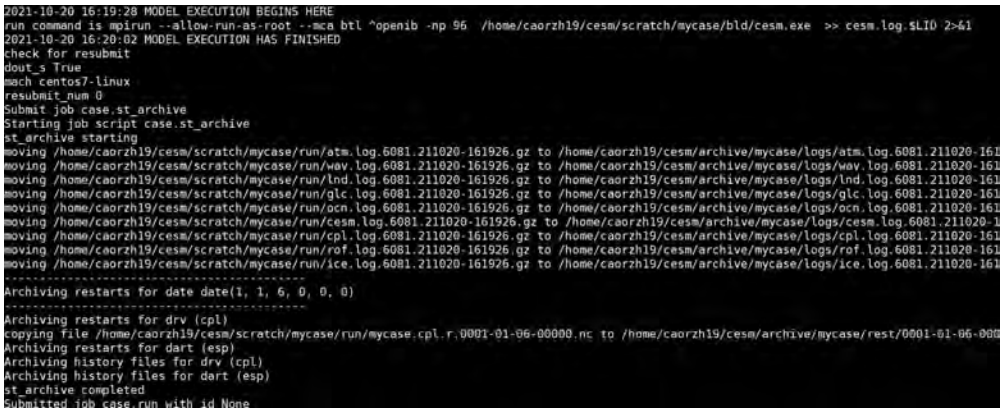
```
ln -sf /usr/bin/cmake3 /usr/bin/cmake
```

Step 12 Execute **cesm.exe**.

```
./case.submit
```

If the information shown in the following figure is displayed when the CESM program is complete, the CESM program completes properly.

View the logs displayed on the screen. You need to manually calculate the time difference. A smaller value indicates higher performance. As shown in the following figure, the start time is 2021-10-20 16:19:28 and the end time is 2021-10-20 16:20:02. Therefore, the execution duration of the test case is 34s.



----End

Multiple Nodes

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Modify the **config_machines.xml** file.

1. Open the **config_machines.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.1/cime/config/cesm/machines/config_machines.xml**

2. Press **i** to enter the insert mode and modify the following information in bold:

```
<machine MACH="centos7-linux">
<DESC>
Example port to centos7 linux system with gcc, netcdf, pnetcdf and mpich
using modules from http://www.admin-magazine.com/HPC/Articles/Environment-Modules
</DESC>
<NODENAME_REGEX>n2</NODENAME_REGEX>
<OS>LINUX</OS>
<PROXY> https://howto.get.out </PROXY>
<COMPILERS>gnu</COMPILERS>
<MPILIBS>openmpi</MPILIBS>
<PROJECT>none</PROJECT>
<SAVE_TIMING_DIR> </SAVE_TIMING_DIR>
<CIME_OUTPUT_ROOT>$ENV{HOME}/cesm/scratch</CIME_OUTPUT_ROOT>
<DIN_LOC_ROOT>$ENV{HOME}/cesm/inputdata</DIN_LOC_ROOT>
<DIN_LOC_ROOT_CLMFORC>$ENV{HOME}/cesm/inputdata/lmwg</DIN_LOC_ROOT_CLMFORC>
<DOUT_S_ROOT>$ENV{HOME}/cesm/archive/$CASE</DOUT_S_ROOT>
<BASELINE_ROOT>$ENV{HOME}/cesm/cesm_baselines</BASELINE_ROOT>
<CCSM_CPRNC>$ENV{HOME}/cesm/tools/cime/tools/cprnc/cprnc</CCSM_CPRNC>
<GMAKE>make</GMAKE>
<GMAKE_J>8</GMAKE_J>
<BATCH_SYSTEM>none</BATCH_SYSTEM>
<SUPPORTED_BY>me@my.address</SUPPORTED_BY>
<MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
<MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
<PROJECT_REQUIRED>FALSE</PROJECT_REQUIRED>
<mpirun mpilib="default">
<executable>mpirun</executable>
<arguments>
<arg name="ntasks">--allow-run-as-root --mca btl ^openib --machinefile path/to/machinefile -
np 192 -N 96 -x PATH=/path/to/bisheng/bin:/path/to/HMPI/bin:/path/to/NETCDF/bin:$PATH -x
LD_LIBRARY_PATH=/path/to/bisheng/lib:/path/to/HMPI/lib:/path/to/NETCDF/lib:
$LD_LIBRARY_PATH </arg>
</arguments>
</mpirun>
<module_system type="module" allow_error="true">
<init_path lang="perl">/usr/share/Modules/init/perl.pm</init_path>
<init_path lang="python">/usr/share/Modules/init/python.py</init_path>
<init_path lang="csh">/usr/share/Modules/init/csh</init_path>
<init_path lang="sh">/usr/share/Modules/init/sh</init_path>
<cmd_path lang="perl">/usr/bin/modulecmd perl</cmd_path>
<cmd_path lang="python">/usr/bin/modulecmd python</cmd_path>
<cmd_path lang="sh">module</cmd_path>
<cmd_path lang="csh">module</cmd_path>
<modules>
<command name="purge"/>
</modules>
</module_system>
<environment_variables>
<env name="NETCDF_PATH">/path/to/NETCDF </env>
<env name="NETCDF_FORTRAN_PATH">/path/to/NETCDF</env>
<env name="PNETCDF_PATH">/path/to/NETCDF</env>
<env name="NETCDF_C_PATH">/path/to/PNETCDF</env>
<env name="MPI_PATH">/path/to/HMPI/env>
<env name="OMP_STACKSIZE">256M</env>
</environment_variables>
<resource_limits>
<resource name="RLIMIT_STACK">-1</resource>
</resource_limits>
</machine>
```

Table 1-163 Parameters in the **config_machines.xml** file

Parameter	Description
<NODENAME_REGEX>	Set this parameter based on the actual node names.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
<COMPILERS>	Set it to **GNU**.
<MPILIBS>	Set it to **OpenMPI**.
<MAX_TASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<MAX_MPITASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<executable>	The default value is **mpiexec**. You need to change it to **mpirun**.
<arguments>	In a multi-node scenario, change the value of **ntasks**. For example, if two nodes with 96 cores each are used, set the total number of processes **ntasks** to 96 x 2 = 192. The value of **-np** is 192 and the value of **-N** (processes on each node) is 96. To run multiple nodes at the same time, add the compiler path and dependency library path. Separate the paths using the **-x** parameter.
<environment_variables>	Set the dependency function path based on the installation location.

NOTE

Change the paths based on **Table 1-163**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Modify the **config_compilers.xml** file.

1. Open the **config_compilers.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.1/cime/config/cesm/machines/config_compiler.xml**
2. Press **i** to enter the insert mode and modify the following information in bold:

```
<compiler COMPILER="gnu">
<CFLAGS>
<base>  -O3 -mcpu=tsv110 -Hx,54,0x08 </base>
<append compile_threaded="true"> -fopenmp </append>
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
</append>
<append DEBUG="FALSE"> -O </append>
</CFLAGS>
<CPPDEFS>
<!-- http://gcc.gnu.org/onlinedocs/gfortran/ -->
<append> -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU</append>
</CPPDEFS>
<CXX_LINKER>FORTRAN</CXX_LINKER>
<FC_AUTO_R8>
<base> -fdefault-real-8 </base>
</FC_AUTO_R8>
<FFLAGS>
<!-- -ffree-line-length-none and -ffixed-line-length-none need to be in FFLAGS rather than in
FIXEDFLAGS/FREEFLAGS
so that these are passed to cmake builds (cmake builds don't use FIXEDFLAGS and FREEFLAGS). -->
<base>  -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/
path/to/PNETCDF/include -I/path/to/HDF5/include -I/path/to/CURL/include -I/path/to/
```

 Copyright © Huawei Technologies Co., Ltd.

```
OPENBLAS/include -I/path/to/ZLIB/include </base>
<append compile_threaded="true"> -fopenmp </append>
<!-- Ideally, we would also have 'invalid' in the ffpe-trap list. But at
least with some versions of gfortran (confirmed with 5.4.0, 6.3.0 and
7.1.0), gfortran's isnan (which is called in cime via the
CPRGNU-specific shr_infnan_isnan) causes a floating point exception
when called on a signaling NaN. -->
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds </
append>
<append DEBUG="FALSE"> -O </append>
</FFLAGS>
<FFLAGS_NOOPT>
<base> -O0 </base>
</FFLAGS_NOOPT>
<FIXEDFLAGS>
<base>  -ffixed-form </base>
</FIXEDFLAGS>
<FREEFLAGS>
<base> -ffree-form </base>
</FREEFLAGS>
<HAS_F2008_CONTIGUOUS>FALSE</HAS_F2008_CONTIGUOUS>
<LDFLAGS>
<append compile_threaded="true"> -fopenmp -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/
path/to/HDF5/lib -L/path/to/OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -
lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz -lcurl</append>
</LDFLAGS>
<MPICC> /path/to/HMPI/bin/mpicc  </MPICC>
<MPICXX> /path/to/HMPI/bin/mpicxx </MPICXX>
<MPIFC> /path/to/HMPI/bin/mpifort </MPIFC>
<SCC> /path/to/bisheng/bin/clang </SCC>
<SCXX> /path/to/bisheng/bin/clang++ </SCXX>
<SFC> /path/to/bisheng/bin/flang </SFC>
<SUPPORTS_CXX>TRUE</SUPPORTS_CXX>
<SLIBS>
<append> -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/path/to/
OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -
lhdf5_hl -lhdf5 -lm -lz -lcurl </append>
</SLIBS>
```

Table 1-164 Parameters in the **config_compilers.xml** file

Parameter	Description
<CFLAGS>	Add the **-mcpu=tsv110** parameter.
<FFLAGS>	Add the **-mcpu=tsv110** parameter and link the **include** paths of all dependency libraries.
<MPICC>	Specifies the mpicc path of Hyper MPI.
<MPICXX>	Specifies the mpicxx location of Hyper MPI.
<MPIFC>	Specifies the mpifort location of Hyper MPI.
<SCC>	Specifies the clang path of the BiSheng Compiler.
<SCXX>	Specifies the clang++ path of the BiSheng Compiler.
<SFC>	Specifies the flang path of the BiSheng Compiler.
<LDFLAGS>	Use the **-L** parameter to link all dependency libraries.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
<SLIBS>	Use the **-L** parameter to link all dependency libraries.

NOTE

Change the paths based on **Table 1-164**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Go to the working directory.

```
cd /path/to/CESM/CESM-cesm2.1.1/cime/scripts
```

Step 5 Create a test case.

```
./create_newcase --case mycase --compset X --res f19_g16 --mach=centos7-linux
```

Information similar to the following is displayed.

Step 6 Download the input data.

```
cd mycase
./check_case
./check_input_data --download
```

NOTE

After a module is successfully downloaded, **SUCCESS** is displayed. You do not need to perform any operations before the whole download process automatically completes.

Step 7 Start the CESM program.

```
./case.setup
```

Step 8 Check the compilation options in the **Macro.make** file. The compilation options in the **Macro.make** file must be the same as those in the **configure.compilers.xml** file. The options are <CFLAGS>, <FFLAGS>, <MPICC>, <MPICXX>, <MPIFC>, <SCC>, <SCXX>, <SFC>, <LDFLAGS>, and <SLIBS>.

```
SUPPORTS_CXX := FALSE
ifeq ($(COMPILER),gnu)
FC_AUTO_R8 :=  -fdefault-real-8
HAS_F2008_CONTIGUOUS := FALSE
FFLAGS_NOOPT :=  -O0
SUPPORTS_CXX := TRUE
FFLAGS :=   -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/path/to/PNETCDF/include -I/path/to/HDF5/include -I/path/to/CURL/include -I/path/to/OPENBLAS/include -I/path/to/ZLIB/include
FIXEDFLAGS :=   -ffixed-form
SCC :=  /path/to/bisheng/bin/clang
SFC :=  /path/to/bisheng/bin/flang
MPICC :=  /path/to/HMPI/bin/mpicc
CFLAGS :=   -O3 -mcpu=tsv110 -Hx,54,0x08
MPIFC :=  /path/to/HMPI/bin/mpifort
```

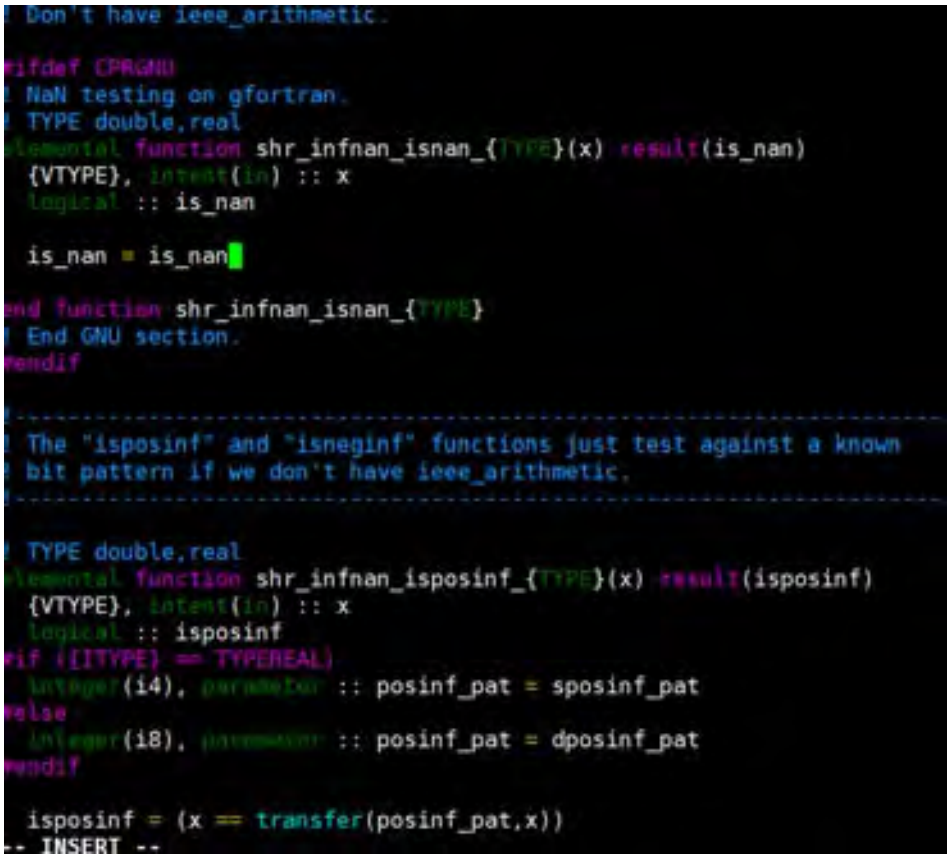
 Copyright © Huawei Technologies Co., Ltd.

```
MPICXX :=  /path/to/HMPI/bin/mpicxx
FREEFLAGS :=  -ffree-form
CXX_LINKER := FORTRAN
SCXX :=  /path/to/bisheng/bin/clang++
endif
ifeq ($(MODEL),pop)
CPPDEFS := $(CPPDEFS)  -D_USE_FLOW_CONTROL
endif
ifeq ($(COMPILER),gnu)
CPPDEFS := $(CPPDEFS)  -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU
SLIBS := $(SLIBS)  -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/path/to/
OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -
lhdf5_hl -lhdf5 -lm -lz -lcurl
ifeq ($(compile_threaded),true)
FFLAGS := $(FFLAGS)  -fopenmp
CFLAGS := $(CFLAGS)  -fopenmp
endif
ifeq ($(DEBUG),TRUE)
FFLAGS := $(FFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds
CFLAGS := $(CFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
endif
ifeq ($(DEBUG),FALSE)
FFLAGS := $(FFLAGS)  -O
CFLAGS := $(CFLAGS)  -O
endif
ifeq ($(compile_threaded),true)
LDFLAGS := $(LDFLAGS)  -L/path/to/NETCDF/lib -L/path/to/PNETCDF/lib -L/path/to/HDF5/lib -L/
path/to/OPENBLAS/lib -L/path/to/CURL/lib -L/path/to/ZLIB/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -
lhdf5_hl -lhdf5 -lm -lz -lcurl
endif
endif
```

Step 9 Modify the **CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in** file.

1. Open the **CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in** file.
   ```
   vi /path/to/CESM/CESM-cesm2.1.1/cime/src/share/util/shr_infnan_mod.F90.in
   ```
2. Press **i** to enter the insert mode and modify the content in line 227, as shown by the cursor in the following figure:

 Copyright © Huawei Technologies Co., Ltd.



NOTE

After performing Step 9.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

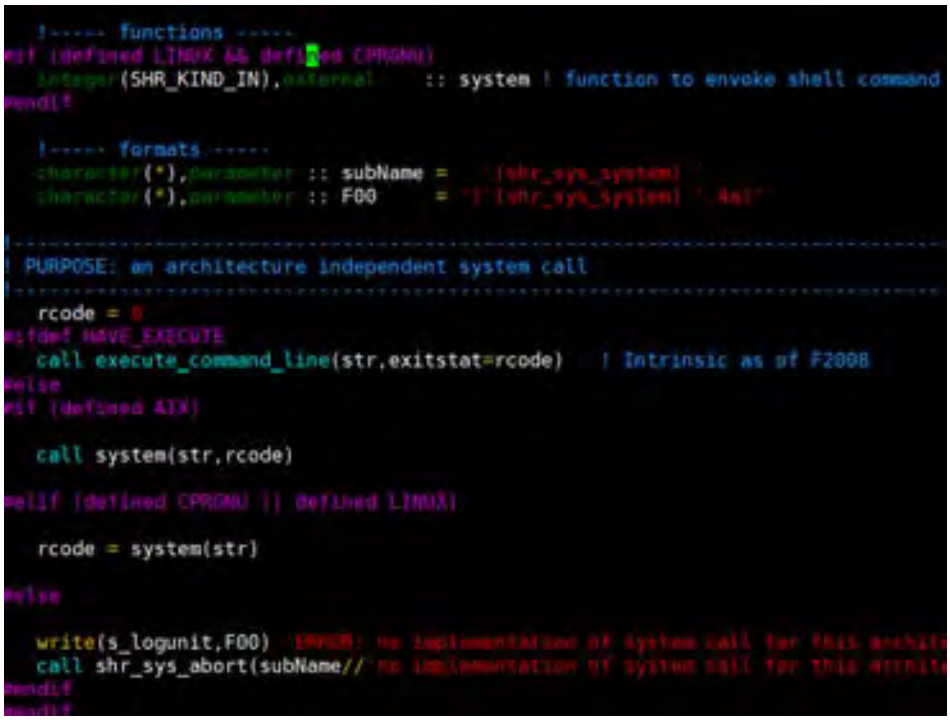
Step 10 Modify the **CESM-release-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90** file.

1. Open the **CESM-release-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90** file.

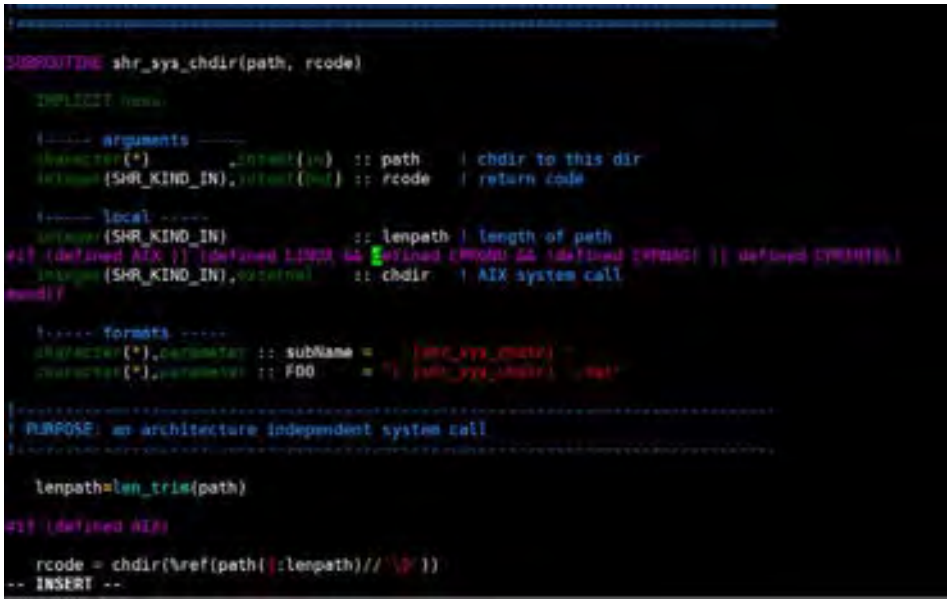
```
vi /path/to/CESM/CESM-cesm2.1.1/cime/src/share/util/shr_sys_mod.F90
```

2. Press **i** to enter the insert mode and modify the file as follows:

   Delete the exclamation mark (!) before **defined CPRGNU** in line 70.



Delete the exclamation mark (!) before **defined CPRGNU** in line 115.



NOTE

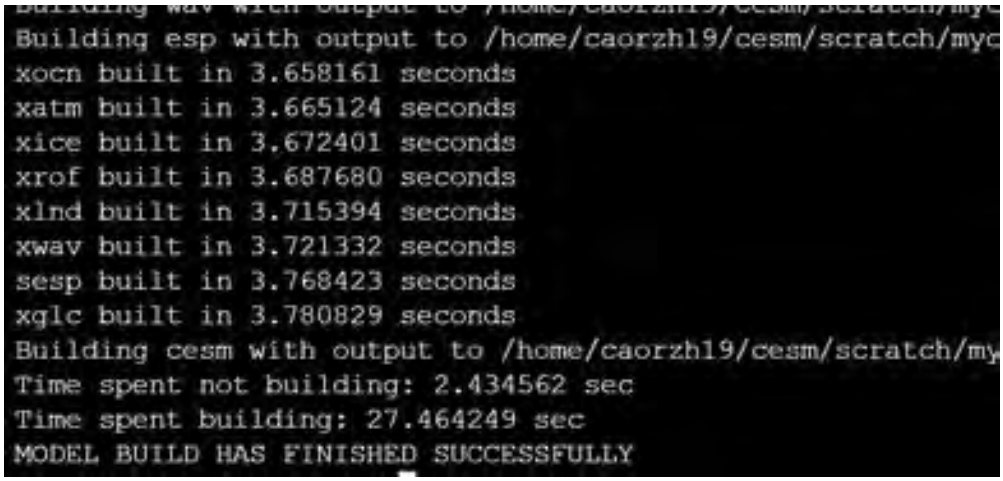
After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Build the **cesm.exe** file.

```
./case.build --skip-provenance-check
```

The operation is successful if the following information in is displayed:



NOTE

You need to soft link CMake3 in the system to CMake and specify the CMake path. If you are the root user and the CMake version is 3.0 or later, ignore **ln -sf /usr/bin/cmake3 /usr/bin/cmake**.

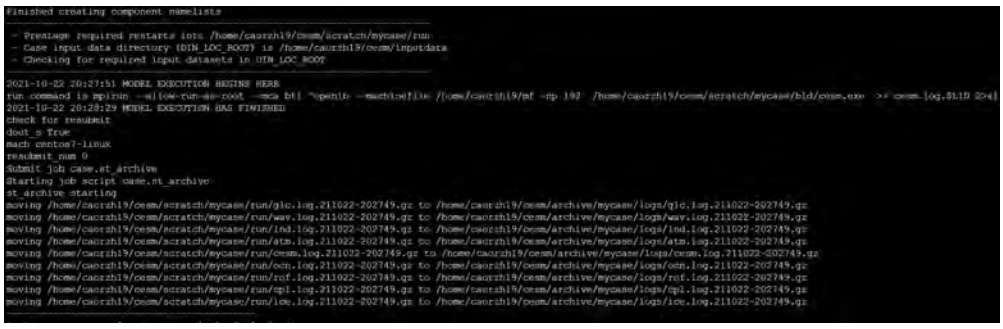
```
ln -sf /usr/bin/cmake3 /usr/bin/cmake
```

Step 12 Execute **cesm.exe**.

```
./case.submit
```

If the information shown in the following figure is displayed when the CESM program is complete, the CESM program completes properly.

View the logs displayed on the screen. You need to manually calculate the time difference. A smaller value indicates higher performance. As shown in the following figure, in a dual-node scenario, the start time is 2021-10-22 20:27:51 and the end time is 2021-10-22 20:28:29. Therefore, the execution duration of the test case is 38s.



----End

1.33 CESM 2.1.3 Porting Guide (openEuler 21.03)

1.33.1 Introduction

The Community Earth System Model (CESM) is a fully coupled numerical simulation of the Earth system consisting of atmospheric, ocean, ice, land surface, carbon cycle, and other components. CESM includes a climate model providing

state-of-art simulations of the Earth's past, present, and future. It is the successor of the Community Climate System Model (CCSM), specifically version 4 (CCSMv4), which provided the initial atmospheric component for CESM.

For more information about CESM, visit the **CESM official website**.

Programming language: Fortran

Brief description: A fully coupled numerical simulation of the Earth system.

Open source license: custom open source license

Recommended Software Version

CESM 2.1.3

1.33.2 Environment Requirements

Hardware Requirements

Table 1-165 lists the hardware requirements.

Table 1-165 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-166 lists the software requirements.

Table 1-166 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
CESM	2.1.3	**https://codeload.github.com/ESCOMP/CESM/tar.gz/refs/tags/release-cesm2.1.3**
OpenBLAS	0.3.6	**https://github.com/xianyi/OpenBLAS/releases**
HDF5	1.10.1	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz**
NetCDF-C	4.8.1	**https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.8.1**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
NetCDF-Fortran	4.5.3	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.5.3.tar.gz

OS Requirements

Table 1-167 lists the OS requirements.

Table 1-167 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	Included in the OS image

1.33.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CESM software porting.

Table 1-168 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/bisheng	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/HMPI	Installation path of Hyper MPI.	
3	/path/to/OPENBLAS	Installation path of OpenBLAS.	
4	/path/to/HDF5	Installation path of HDF5.	
5	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
6	/path/to/CESM	Installation path of CESM.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
7	/CESMroot	Data directory of CESM.	

1.33.4 Configuring the Compilation Environment

1.33.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-169 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Setting the BiSheng Compiler and Hyper MPI Environment Variables	See **1.33.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables**.
4	Installing OpenBLAS	See **1.33.4.3 Installing OpenBLAS**.
5	Installing HDF5	See **1.33.4.4 Installing HDF5**.
6	Installing NetCDF-C	See **1.33.4.5 Installing NetCDF-C**.
7	Installing NetCDF-Fortran	See **1.33.4.6 Installing NetCDF-Fortran**.

1.33.4.2 Setting the BiSheng Compiler and Hyper MPI Environment Variables

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/bisheng/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bisheng/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
export MPICC=mpicc
export MPICXX=mpicxx
```

 Copyright © Huawei Technologies Co., Ltd.

```
export MPIFC=mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
export F77=$MPIFC
```

----End

1.33.4.3 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Perform the configuration.

```
export CC=`which clang`
export CXX= `which clang++`
export FC=`which flang`
```

Step 5 Perform the compilation and installation.

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

1.33.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-typesdiscards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.33.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the offline NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.8.1.tar.gz
```

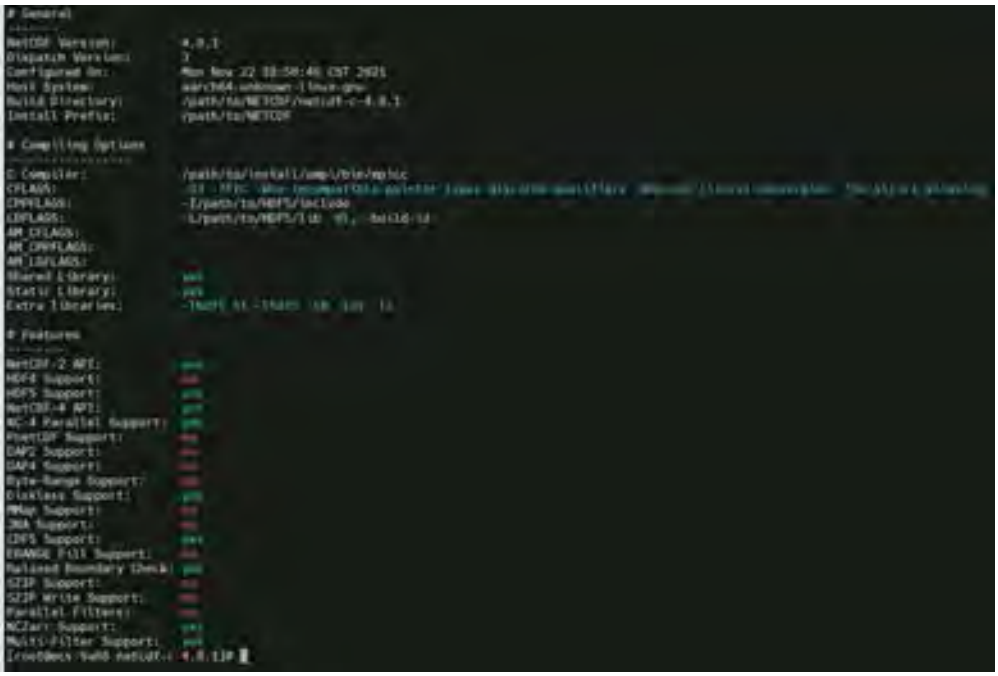
Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.8.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include" LDFLAGS="-L/path/to/HDF5/lib -Wl,--build-id" --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" --enable-fortran --enable-netcdf4 --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

The configuration is successful if the following information in is displayed:



Step 5 Perform the compilation and installation.

```
make
make install
```

After the installation is complete, the following information is displayed:

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

Step 7 Check whether NetCDF-C is installed.

```
nc-config --all
```

The installation is successful if the following information in is displayed:

----End

1.33.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the offline NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.5.3.tar.gz
```

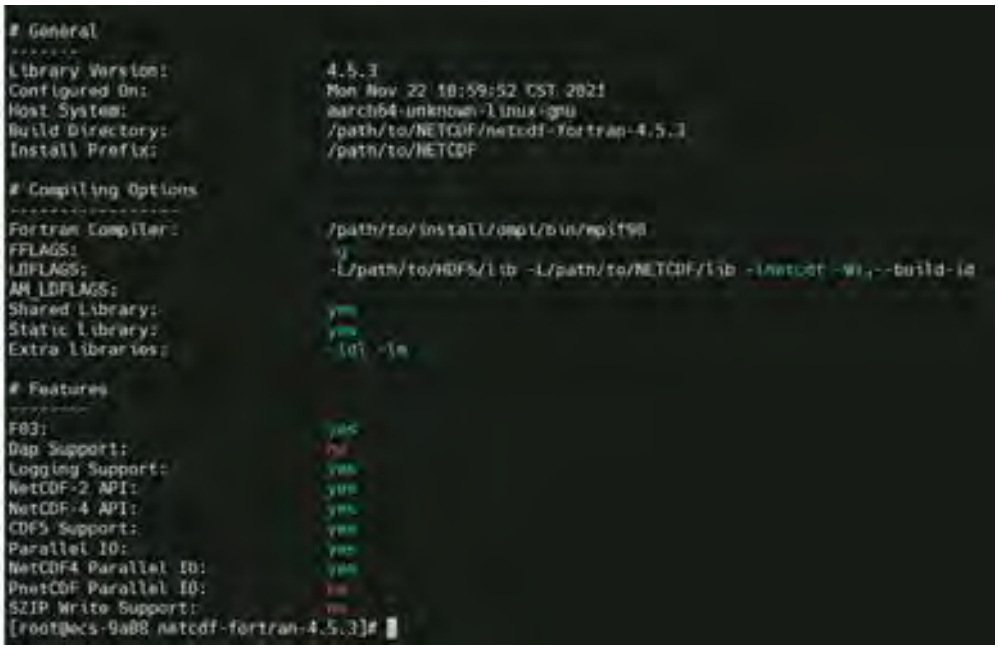
Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.5.3
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -lnetcdf -Wl,--build-id" --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC"  --enable-fortran --enable-netcdf4 --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx F77=mpif77
```

The configuration is successful if the following information in is displayed:



Step 5 Perform the compilation and installation.

```
make
make install
```

The installation is successful if the following information in is displayed:

```
 /usr/bin/mkdir -p '/path/to/NETCDF/bin'
 /usr/bin/install -c nf-config '/path/to/NETCDF/bin'
 /usr/bin/mkdir -p '/path/to/NETCDF/lib/pkgconfig'
 /usr/bin/install -c -m 644 netcdf-fortran.pc '/path/to/NETCDF/lib/pkgconfig'
 /usr/bin/mkdir -p '/path/to/NETCDF/lib'
 /usr/bin/install -c -m 644 libnetcdff.settings '/path/to/NETCDF/lib'
make  install-data-hook
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.5.3'

+-------------------------------------------------------------+
| Congratulations! You have successfully installed the netCDF |
| Fortran libraries.                                          |
|                                                             |
| You can use script "nf-config" to find out the relevant     |
| compiler options to build your application. Enter           |
|                                                             |
|     nf-config --help                                        |
|                                                             |
| for additional information.                                 |
|                                                             |
| CAUTION:                                                    |
|                                                             |
| If you have not already run "make check", then we strongly  |
| recommend you do so. It does not take very long.            |
|                                                             |
| Before using netCDF to store important data, test your      |
| build with "make check".                                    |
|                                                             |
| NetCDF is tested nightly on many platforms at Unidata       |
| but your platform is probably different in some ways.       |
|                                                             |
| If any tests fail, please see the netCDF web site:          |
| http://www.unidata.ucar.edu/software/netcdf/                |
|                                                             |
| NetCDF is developed and maintained at the Unidata Program   |
| Center. Unidata provides a broad array of data and software |
| tools for use in geoscience education and research.         |
| http://www.unidata.ucar.edu                                 |
+-------------------------------------------------------------+

make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.3'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.3'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.3'
[root@ecs-9a08 netcdf-fortran-4.5.3]#
```

Step 6 Set the NetCDF-Fortran environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

Step 7 Check whether NetCDF-Fortran is installed.

```
nf-config --all
```

The installation is successful if the following information in is displayed:

 Copyright © Huawei Technologies Co., Ltd.

----End

1.33.5 Obtaining the Source Code

Procedure

Step 1 Download the CSEM 2.1.3 installation package.

https://codeload.github.com/ESCOMP/CESM/tar.gz/refs/tags/release-cesm2.1.3

Step 2 Use an SFTP tool to upload the CESM installation package to the */path/to/CESM* directory on the server.

----End

1.33.6 Compilation and Installation

Prerequisites

Configure an external network so that the server can be connected to the Internet.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the CESM installation package directory.

```
cd /path/to/CESM
```

Step 3 Decompress the CESM installation package.

```
tar  -zxvf CESM-release-cesm2.1.3.tar.gz
```

Step 4 Go to the source code directory.

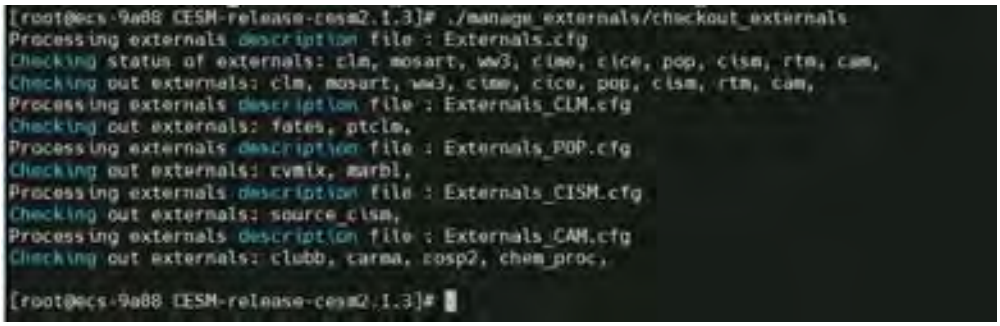
```
cd CESM-release-cesm2.1.3
```

Step 5 Download necessary components.

```
./manage_externals/checkout_externals
```

The command is executed successfully if the following information in is displayed:

 Copyright © Huawei Technologies Co., Ltd.



After the download is complete, the following directory files are displayed:

Step 6 Check whether the required components are installed.

```
./manage_externals/checkout_externals -S
```

The installation is successful if the following information in is displayed:

----End

1.33.7 Running and Verification

Single Node

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Modify the **config_machines.xml** file.

1. Open the **config_machines.xml** file.

   ```
   vi /path/to/CESM/CESM-release-cesm2.1.3/cime/config/cesm/machines/config_machines.xml
   ```

2. Press **i** to enter the edit mode and modify the following information in bold:

   ```
   <machine MACH="centos7-linux">
   <DESC>
   Example port to centos7 linux system with gcc, netcdf, pnetcdf and mpich
   using modules from http://www.admin-magazine.com/HPC/Articles/Environment-Modules
   </DESC>
   <NODENAME_REGEX>n2</NODENAME_REGEX>
   <OS>LINUX</OS>
   <PROXY> https://howto.get.out </PROXY>
   <COMPILERS>gnu</COMPILERS>
   <MPILIBS>openmpi</MPILIBS>
   <PROJECT>none</PROJECT>
   <SAVE_TIMING_DIR> </SAVE_TIMING_DIR>
   <CIME_OUTPUT_ROOT>/CESMroot/cesm/scratch</CIME_OUTPUT_ROOT>
   <DIN_LOC_ROOT>/CESMroot/cesm/inputdata</DIN_LOC_ROOT>
   <DIN_LOC_ROOT_CLMFORC>/CESMroot/cesm/inputdata/lmwg</DIN_LOC_ROOT_CLMFORC>
   ```

```
<DOUT_S_ROOT>/CESMroot/cesm/archive/$CASE</DOUT_S_ROOT>
<BASELINE_ROOT>/CESMroot/cesm/cesm_baselines</BASELINE_ROOT>
<CCSM_CPRNC>/CESMroot/cesm/tools/cime/tools/cprnc/cprnc</CCSM_CPRNC>
<GMAKE>make</GMAKE>
<GMAKE_J>8</GMAKE_J>
<BATCH_SYSTEM>none</BATCH_SYSTEM>
<SUPPORTED_BY>me@my.address</SUPPORTED_BY>
<MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
<MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
<PROJECT_REQUIRED>FALSE</PROJECT_REQUIRED>
<mpirun mpilib="default">
<executable>mpirun</executable>
<arguments>
<arg name="ntasks">--allow-run-as-root --mca btl ^openib -np {{ total_tasks }} </arg>
</arguments>
</mpirun>
<module_system type="module" allow_error="true">
<init_path lang="perl">/usr/share/Modules/init/perl.pm</init_path>
<init_path lang="python">/usr/share/Modules/init/python.py</init_path>
<init_path lang="csh">/usr/share/Modules/init/csh</init_path>
<init_path lang="sh">/usr/share/Modules/init/sh</init_path>
<cmd_path lang="perl">/usr/bin/modulecmd perl</cmd_path>
<cmd_path lang="python">/usr/bin/modulecmd python</cmd_path>
<cmd_path lang="sh">module</cmd_path>
<cmd_path lang="csh">module</cmd_path>
<modules>
<command name="purge"/>
</modules>
</module_system>
<environment_variables>
<env name="NETCDF_PATH">/path/to/NETCDF</env>
<env name="NETCDF_FORTRAN_PATH">/path/to/NETCDF</env>
<env name="NETCDF_C_PATH">/path/to/NETCDF</env>
<env name="MPI_PATH">/path/to/HMPI</env>
<env name="OMP_STACKSIZE">256M</env>
</environment_variables>
<resource_limits>
<resource name="RLIMIT_STACK">-1</resource>
</resource_limits>
</machine>
```

Table 1-170 Parameters in the **config_machines.xml** file

Parameter	Description
<NODENAME_REGEX>	Set this parameter based on the actual node names.
<COMPILERS>	Set it to **GNU**.
<MPILIBS>	Set it to **OpenMPI**.
<MAX_TASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<MAX_MPITASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<executable>	The default value is **mpiexec**. You need to change it to **mpirun**.
<arguments>	Set it to **--allow-run-as-root --mca btl ^openib -np {{ total_tasks }}**. The system automatically generates the total number of threads.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
<environment_variables>	Set the dependency function path based on the installation location.
<CIME_OUTPUT_ROOT> <DIN_LOC_ROOT> <DIN_LOC_ROOT_CLMFORC> <DOUT_S_ROOT> <BASELINE_ROOT> <CCSM_CPRNC>	The default value is **$ENV{HOME}**, which indicates the actual CESM data directory.

NOTE

Change the paths based on **Table 1-170**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Modify the **config_compilers.xml** file.

1. Open the **config_compilers.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.3/cime/config/cesm/machines/config_compilers.xml**
2. Press **i** to enter the edit mode and modify the following information in bold:

```
<compiler COMPILER="gnu">
<CFLAGS>
<base>  -O3 -mcpu=tsv110 -Hx,54,0x08 </base>
<append compile_threaded="true"> -fopenmp </append>
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
</append>
<append DEBUG="FALSE"> -O </append>
</CFLAGS>
<CPPDEFS>
<!-- http://gcc.gnu.org/onlinedocs/gfortran/ -->
<append> -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU</append>
</CPPDEFS>
<CXX_LINKER>FORTRAN</CXX_LINKER>
<FC_AUTO_R8>
<base> -fdefault-real-8 </base>
</FC_AUTO_R8>
<FFLAGS>
<!-- -ffree-line-length-none and -ffixed-line-length-none need to be in FFLAGS rather than in
FIXEDFLAGS/FREEFLAGS
so that these are passed to cmake builds (cmake builds don't use FIXEDFLAGS and FREEFLAGS). -->
<base>  -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/
path/to/HDF5/include -I/path/to/OPENBLAS/include </base>
<append compile_threaded="true"> -fopenmp </append>
<!-- Ideally, we would also have 'invalid' in the ffpe-trap list. But at
least with some versions of gfortran (confirmed with 5.4.0, 6.3.0 and
7.1.0), gfortran's isnan (which is called in cime via the
CPRGNU-specific shr_infnan_isnan) causes a floating point exception
when called on a signaling NaN. -->
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds </
append>
<append DEBUG="FALSE"> -O </append>
</FFLAGS>
<FFLAGS_NOOPT>
<base> -O0 </base>
</FFLAGS_NOOPT>
```

 Copyright © Huawei Technologies Co., Ltd.

```
<FIXEDFLAGS>
<base>  -ffixed-form </base>
</FIXEDFLAGS>
<FREEFLAGS>
<base> -ffree-form </base>
</FREEFLAGS>
<HAS_F2008_CONTIGUOUS>FALSE</HAS_F2008_CONTIGUOUS>
<LDFLAGS>
<append compile_threaded="true"> -fopenmp -L/path/to/NETCDF/lib  -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm –lz </append>
</LDFLAGS>
<MPICC> /path/to/HMPI/bin/mpicc  </MPICC>
<MPICXX> /path/to/HMPI/bin/mpicxx </MPICXX>
<MPIFC> /path/to/HMPI/bin/mpifort </MPIFC>
<SCC> /path/to/bisheng/bin/clang </SCC>
<SCXX> /path/to/bisheng/bin/clang++ </SCXX>
<SFC> /path/to/bisheng/bin/flang </SFC>
<SUPPORTS_CXX>TRUE</SUPPORTS_CXX>
<SLIBS>
<append> -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm –lz </append>
</SLIBS>
```

Table 1-171 Parameters in the **config_compilers.xml** file

Parameter	Description
<CFLAGS>	Add the **-mcpu=tsv110** parameter.
<FFLAGS>	Add the **-mcpu=tsv110** parameter and link the **include** paths of all dependency libraries.
<MPICC>	Specifies the mpicc path of Hyper MPI.
<MPICXX>	Specifies the mpicxx location of Hyper MPI.
<MPIFC>	Specifies the mpifort location of Hyper MPI.
<SCC>	Specifies the clang path of the BiSheng Compiler.
<SCXX>	Specifies the clang++ path of the BiSheng Compiler.
<SFC>	Specifies the flang path of the BiSheng Compiler.
<LDFLAGS>	Use the **-L** parameter to link all dependency libraries.
<SLIBS>	Use the **-L** parameter to link all dependency libraries.

NOTE

Change the paths based on **Table 1-171**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Go to the working directory.

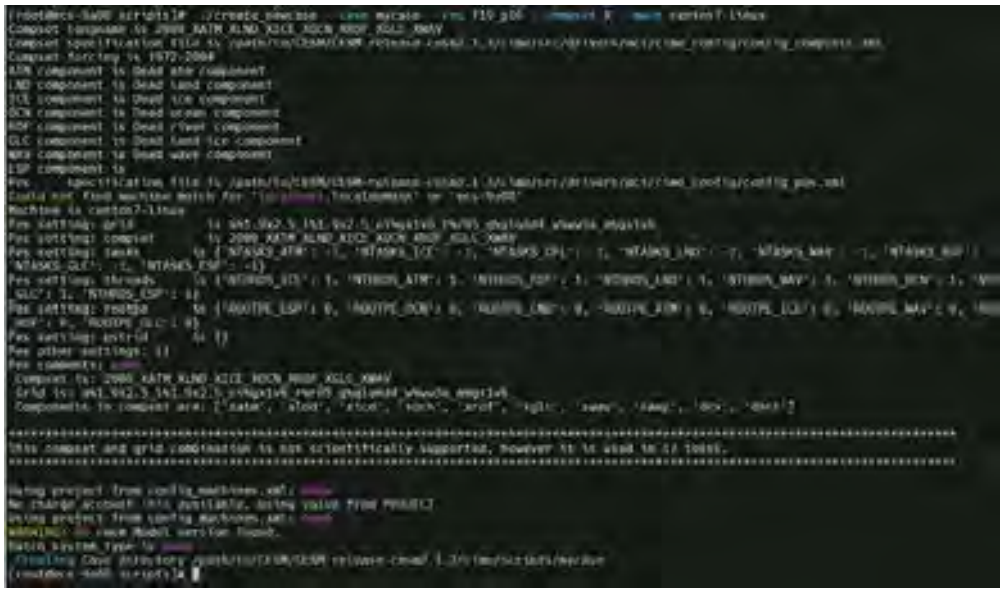
```
cd /path/to/CESM/CESM-cesm2.1.3/cime/scripts
```

Step 5 Create a test case.

```
./create_newcase --case mycase --compset X --res f19_g16 --mach=centos7-linux
```

The operation is successful if the following information in is displayed:

 Copyright © Huawei Technologies Co., Ltd.



Step 6 Download the input data. No offline test case packages are involved. All components are automatically downloaded based on the command line.

```
cd mycase
./check_case
./check_input_data --download
```

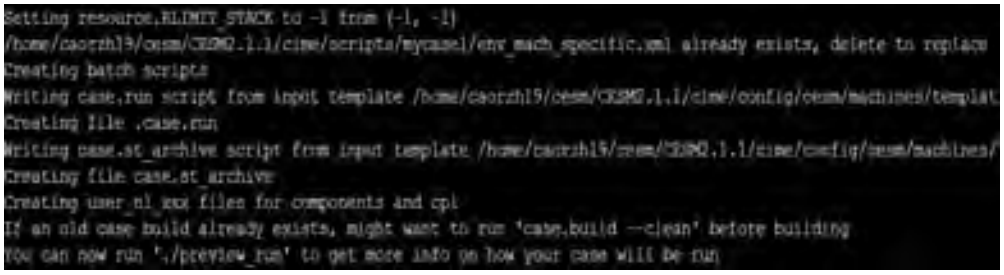
NOTE

After a module is successfully downloaded, **SUCCESS** is displayed. You do not need to perform any operations before the whole download process automatically completes.

Step 7 Start the CESM program.

```
./case.setup
```

The operation is successful if the following information in is displayed:



Step 8 Check the compilation options in the **Macro.make** file. The compilation options in the **Macro.make** file must be the same as those in the **config_compilers.xml** file. The options are <CFLAGS>, <FFLAGS>, <MPICC>, <MPICXX>, <MPIFC>, <SCC>, <SCXX>, <SFC>, <LDFLAGS>, and <SLIBS>.

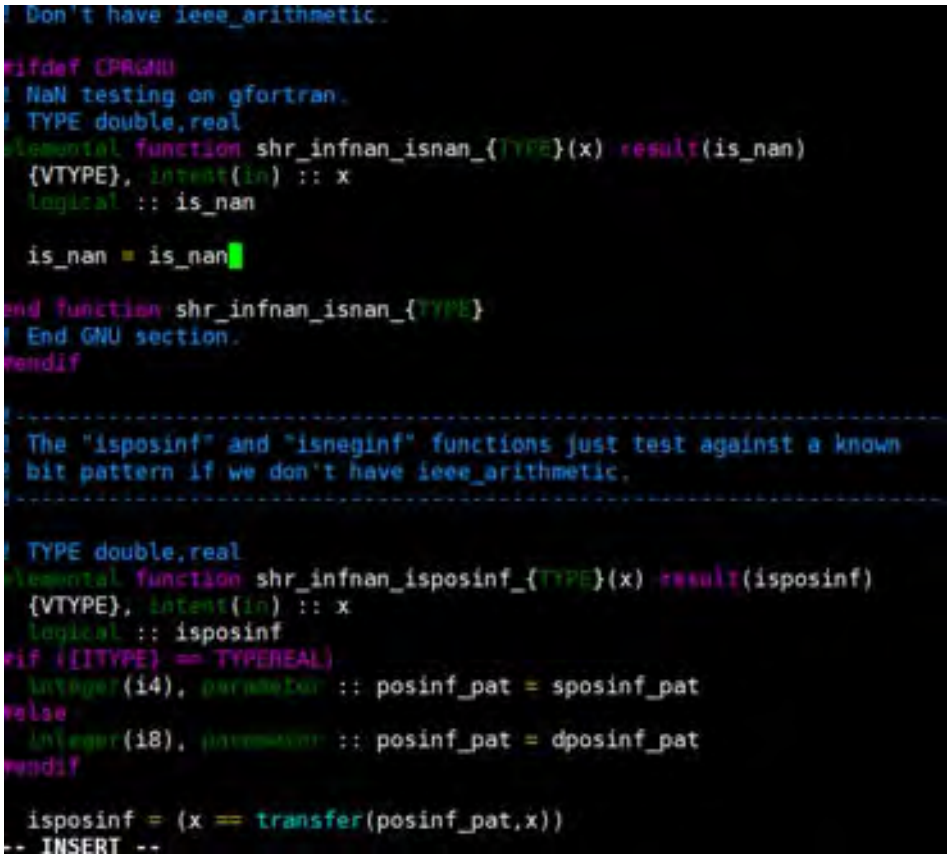
```
SUPPORTS_CXX := FALSE
ifeq ($(COMPILER),gnu)
FC_AUTO_R8 :=  -fdefault-real-8
HAS_F2008_CONTIGUOUS := FALSE
FFLAGS_NOOPT :=  -O0
SUPPORTS_CXX := TRUE
FFLAGS :=   -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/path/to/HDF5/include -I/path/to/OPENBLAS/include
FIXEDFLAGS :=   -ffixed-form
SCC :=  /path/to/bisheng/bin/clang
SFC :=  /path/to/bisheng/bin/flang
```

```
MPICC :=  /path/to/HMPI/bin/mpicc
CFLAGS :=   -O3 -mcpu=tsv110 -Hx,54,0x08
MPIFC :=  /path/to/HMPI/bin/mpifort
MPICXX :=  /path/to/HMPI/bin/mpicxx
FREEFLAGS :=  -ffree-form
CXX_LINKER := FORTRAN
SCXX :=  /path/to/bisheng/bin/clang++
endif
ifeq ($(MODEL),pop)
CPPDEFS := $(CPPDEFS)  -D_USE_FLOW_CONTROL
endif
ifeq ($(COMPILER),gnu)
CPPDEFS := $(CPPDEFS)  -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU
SLIBS := $(SLIBS)  -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -
lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz
ifeq ($(compile_threaded),true)
FFLAGS := $(FFLAGS)  -fopenmp
CFLAGS := $(CFLAGS)  -fopenmp
endif
ifeq ($(DEBUG),TRUE)
FFLAGS := $(FFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds
CFLAGS := $(CFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
endif
ifeq ($(DEBUG),FALSE)
FFLAGS := $(FFLAGS)  -O
CFLAGS := $(CFLAGS)  -O
endif
ifeq ($(compile_threaded),true)
LDFLAGS := $(LDFLAGS)  -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -
lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz
endif
endif
```

Step 9 Modify the **CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in** file.

1. Open the **CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in** file.
   ```
   vi /path/to/CESM/CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in
   ```
2. Press **i** to enter the edit mode and modify the content in line 227, as shown by the cursor in the following figure:

 Copyright © Huawei Technologies Co., Ltd.



NOTE

After performing Step 9.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

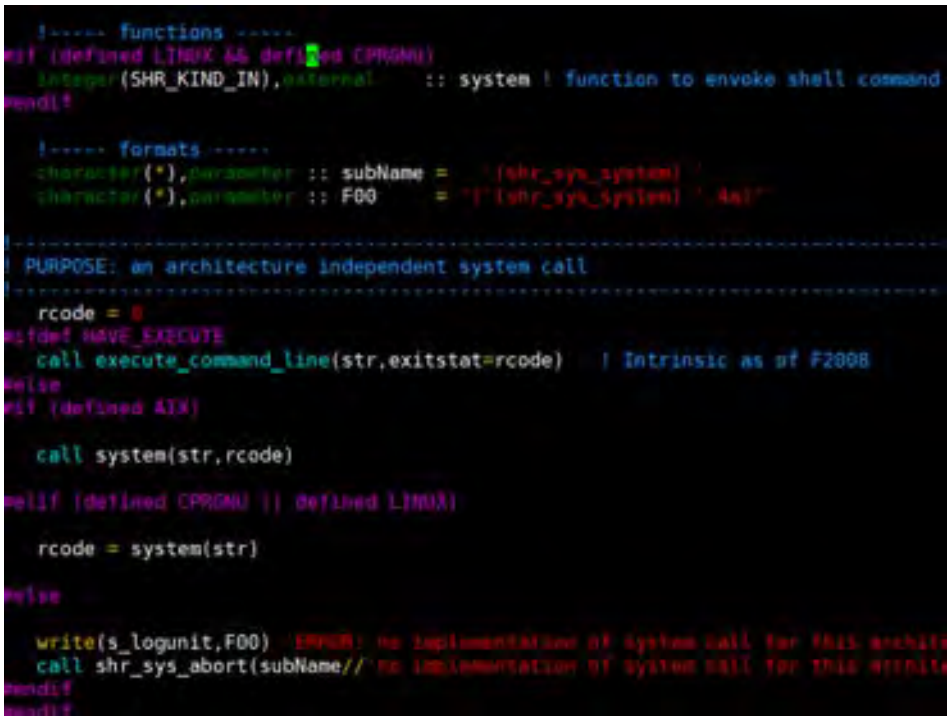
Step 10 Modify the **CESM-release-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90** file.

1. Open the **CESM-release-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90** file.

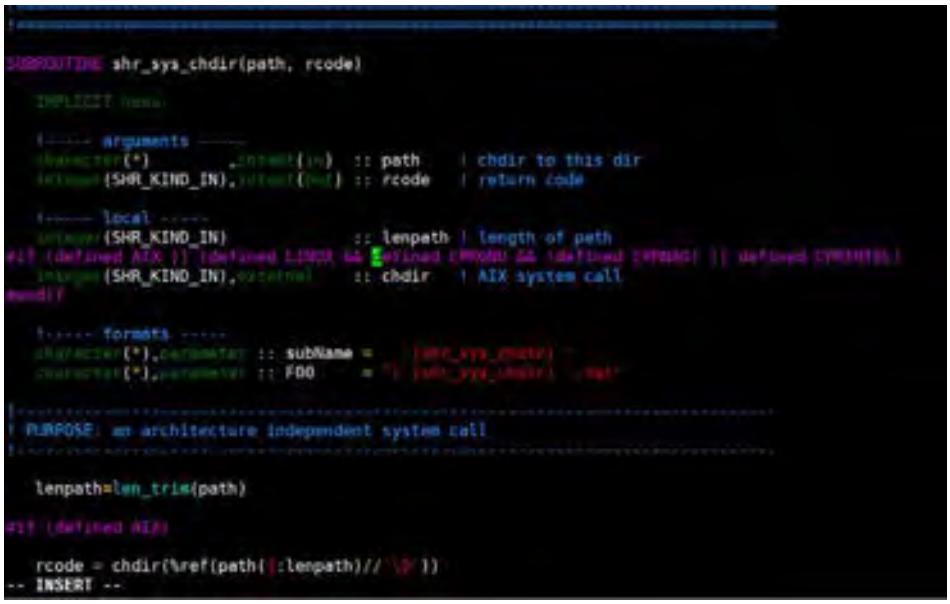
```
vi /path/to/CESM/CESM-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90
```

2. Press **i** to enter the edit mode and modify the file as follows:

   Delete the exclamation mark (!) before **defined CPRGNU** in line 70.



Delete the exclamation mark (!) before **defined CPRGNU** in line 115.



NOTE

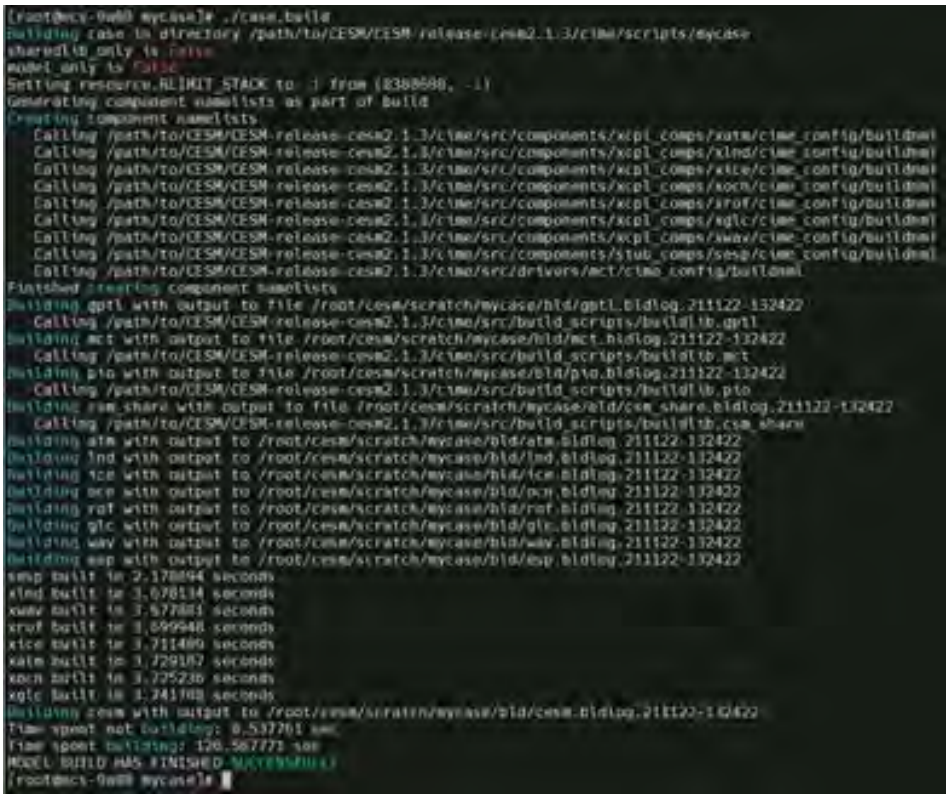
After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Build the **cesm.exe** file.

- Method 1:
  ```
  ./case.build --skip-provenance-check
  ```
- Method 2:
  ```
  ./case.build
  ```

  The operation is successful if the following information in is displayed:



NOTE

You need to soft link CMake3 in the system to CMake and specify the CMake path. If you are the root user and the CMake version is 3.0 or later, ignore **ln -sf /usr/bin/cmake3 /usr/bin/cmake**.

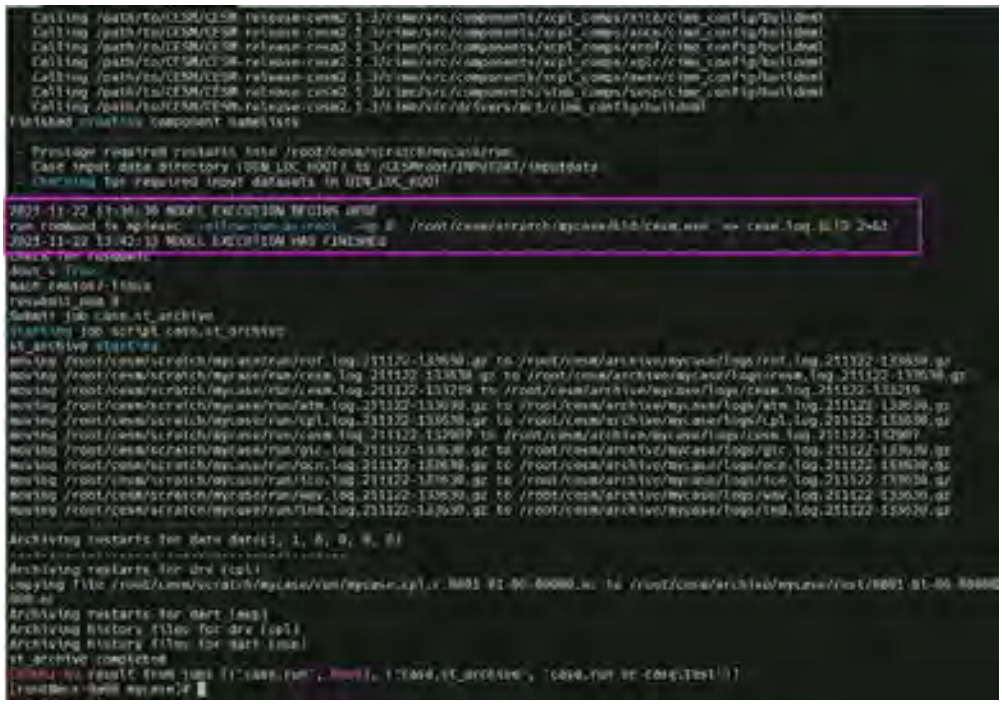
```
ln -sf /usr/bin/cmake3 /usr/bin/cmake
```

Step 12 Execute **cesm.exe**.

```
./case.submit
```

If the information shown in the following figure is displayed when the CESM program is complete, the CESM program completes properly.

View the logs displayed on the screen and manually calculate the time difference. A smaller value indicates higher performance. As shown in the following figure, the start time is 2021-10-20 16:19:28 and the end time is 2021-10-20 16:20:02. Therefore, the execution duration of the test case is 34s.



----End

Multiple Nodes

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Modify the **config_machines.xml** file.

1. Open the **config_machines.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.3/cime/config/cesm/machines/config_machines.xml**
2. Press **i** to enter the edit mode and modify the following information in bold:

```
<machine MACH="centos7-linux">
<DESC>
Example port to centos7 linux system with gcc, netcdf, pnetcdf and mpich
using modules from http://www.admin-magazine.com/HPC/Articles/Environment-Modules
</DESC>
<NODENAME_REGEX>n2</NODENAME_REGEX>
<OS>LINUX</OS>
<PROXY> https://howto.get.out </PROXY>
<COMPILERS>gnu</COMPILERS>
<MPILIBS>openmpi</MPILIBS>
<PROJECT>none</PROJECT>
<SAVE_TIMING_DIR> </SAVE_TIMING_DIR>
<CIME_OUTPUT_ROOT>/CESMroot/cesm/scratch</CIME_OUTPUT_ROOT>
<DIN_LOC_ROOT>/CESMroot/cesm/inputdata</DIN_LOC_ROOT>
<DIN_LOC_ROOT_CLMFORC>/CESMroot/cesm/inputdata/lmwg</DIN_LOC_ROOT_CLMFORC>
<DOUT_S_ROOT>/CESMroot/cesm/archive/$CASE</DOUT_S_ROOT>
<BASELINE_ROOT>/CESMroot/cesm/cesm_baselines</BASELINE_ROOT>
<CCSM_CPRNC>/CESMroot/cesm/tools/cime/tools/cprnc/cprnc</CCSM_CPRNC>
<GMAKE>make</GMAKE>
<GMAKE_J>8</GMAKE_J>
<BATCH_SYSTEM>none</BATCH_SYSTEM>
<SUPPORTED_BY>me@my.address</SUPPORTED_BY>
<MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
<MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
<PROJECT_REQUIRED>FALSE</PROJECT_REQUIRED>
<mpirun mpilib="default">
<executable>mpirun</executable>
<arguments>
```

```
<arg name="ntasks">--allow-run-as-root --mca btl ^openib --machinefile path/to/machinefile -
np 192 -N 96 -x PATH=/path/to/bisheng/bin:/path/to/HMPI/bin:/path/to/NETCDF/bin:$PATH -x
LD_LIBRARY_PATH=/path/to/bisheng/lib:/path/to/HMPI/lib:/path/to/NETCDF/lib:
$LD_LIBRARY_PATH </arg>
</arguments>
</mpirun>
<module_system type="module" allow_error="true">
<init_path lang="perl">/usr/share/Modules/init/perl.pm</init_path>
<init_path lang="python">/usr/share/Modules/init/python.py</init_path>
<init_path lang="csh">/usr/share/Modules/init/csh</init_path>
<init_path lang="sh">/usr/share/Modules/init/sh</init_path>
<cmd_path lang="perl">/usr/bin/modulecmd perl</cmd_path>
<cmd_path lang="python">/usr/bin/modulecmd python</cmd_path>
<cmd_path lang="sh">module</cmd_path>
<cmd_path lang="csh">module</cmd_path>
<modules>
<command name="purge"/>
</modules>
</module_system>
<environment_variables>
<env name="NETCDF_PATH">/path/to/NETCDF </env>
<env name="NETCDF_FORTRAN_PATH">/path/to/NETCDF</env>
<env name="NETCDF_C_PATH">/path/to/NETCDF</env>
<env name="MPI_PATH">/path/to/HMPI</env>
<env name="OMP_STACKSIZE">256M</env>
</environment_variables>
<resource_limits>
<resource name="RLIMIT_STACK">-1</resource>
</resource_limits>
</machine>
```

Table 1-172 Parameters in the **config_machines.xml** file

Parameter	Description
<NODENAME_REGEX>	Set this parameter based on the actual node names.
<COMPILERS>	Set it to **GNU**.
<MPILIBS>	Set it to **OpenMPI**.
<MAX_TASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<MAX_MPITASKS_PER_NODE>	The maximum value is the total number of CPU cores on the node.
<executable>	The default value is **mpiexec**. You need to change it to **mpirun**.
<arguments>	In a multi-node scenario, change the value of **ntasks**. For example, if two nodes with 96 cores each are used, set the total number of processes **ntasks** to 96 x 2 = 192. The value of **-np** is 192 and the value of **-N** (processes on each node) is 96. To run multiple nodes at the same time, add the compiler path and dependency library path. Separate the paths using the **-x** parameter.
<environment_variables>	Set the dependency function path based on the installation location.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
<CIME_OUTPUT_ROOT> <DIN_LOC_ROOT> <DIN_LOC_ROOT_CLMFORC> <DOUT_S_ROOT> <BASELINE_ROOT> <CCSM_CPRNC>	The default value is **$ENV{HOME}**, which indicates the actual CESM data directory.

NOTE

Change the paths based on **Table 1-172**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Modify the **config_compilers.xml** file.

1. Open the **config_compilers.xml** file.
   vi */path/to/CESM*/**CESM-cesm2.1.3/cime/config/cesm/machines/config_compiler.xml**
2. Press **i** to enter the edit mode and modify the following information in bold:

```
<compiler COMPILER="gnu">
<CFLAGS>
<base>  -O3 -mcpu=tsv110 -Hx,54,0x08 </base>
<append compile_threaded="true"> -fopenmp </append>
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
</append>
<append DEBUG="FALSE"> -O </append>
</CFLAGS>
<CPPDEFS>
<!-- http://gcc.gnu.org/onlinedocs/gfortran/ -->
<append> -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU</append>
</CPPDEFS>
<CXX_LINKER>FORTRAN</CXX_LINKER>
<FC_AUTO_R8>
<base> -fdefault-real-8 </base>
</FC_AUTO_R8>
<FFLAGS>
<!-- -ffree-line-length-none and -ffixed-line-length-none need to be in FFLAGS rather than in
FIXEDFLAGS/FREEFLAGS
so that these are passed to cmake builds (cmake builds don't use FIXEDFLAGS and FREEFLAGS). -->
<base>  -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/
path/to/HDF5/include -I/path/to/OPENBLAS/include </base>
<append compile_threaded="true"> -fopenmp </append>
<!-- Ideally, we would also have 'invalid' in the ffpe-trap list. But at
least with some versions of gfortran (confirmed with 5.4.0, 6.3.0 and
7.1.0), gfortran's isnan (which is called in cime via the
CPRGNU-specific shr_infnan_isnan) causes a floating point exception
when called on a signaling NaN. -->
<append DEBUG="TRUE"> -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds </
append>
<append DEBUG="FALSE"> -O </append>
</FFLAGS>
<FFLAGS_NOOPT>
<base> -O0 </base>
</FFLAGS_NOOPT>
<FIXEDFLAGS>
<base>  -ffixed-form </base>
</FIXEDFLAGS>
```

 Copyright © Huawei Technologies Co., Ltd.

```
<FREEFLAGS>
<base> -ffree-form </base>
</FREEFLAGS>
<HAS_F2008_CONTIGUOUS>FALSE</HAS_F2008_CONTIGUOUS>
<LDFLAGS>
<append compile_threaded="true"> -fopenmp -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/
path/to/OPENBLAS/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm –lz </append>
</LDFLAGS>
<MPICC> /path/to/HMPI/bin/mpicc  </MPICC>
<MPICXX> /path/to/HMPI/bin/mpicxx </MPICXX>
<MPIFC> /path/to/HMPI/bin/mpifort </MPIFC>
<SCC> /path/to/bisheng/bin/clang </SCC>
<SCXX> /path/to/bisheng/bin/clang++ </SCXX>
<SFC> /path/to/bisheng/bin/flang </SFC>
<SUPPORTS_CXX>TRUE</SUPPORTS_CXX>
<SLIBS>
<append> -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -
lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz </append>
</SLIBS>
```

Table 1-173 Parameters in the **config_compilers.xml** file

Parameter	Description
<CFLAGS>	Add the **-mcpu=tsv110** parameter.
<FFLAGS>	Add the **-mcpu=tsv110** parameter and link the **include** paths of all dependency libraries.
<MPICC>	Specifies the mpicc path of Hyper MPI.
<MPICXX>	Specifies the mpicxx location of Hyper MPI.
<MPIFC>	Specifies the mpifort location of Hyper MPI.
<SCC>	Specifies the clang path of the BiSheng Compiler.
<SCXX>	Specifies the clang++ path of the BiSheng Compiler.
<SFC>	Specifies the flang path of the BiSheng Compiler.
<LDFLAGS>	Use the **-L** parameter to link all dependency libraries.
<SLIBS>	Use the **-L** parameter to link all dependency libraries.

NOTE

Change the paths based on **Table 1-173**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Go to the working directory.

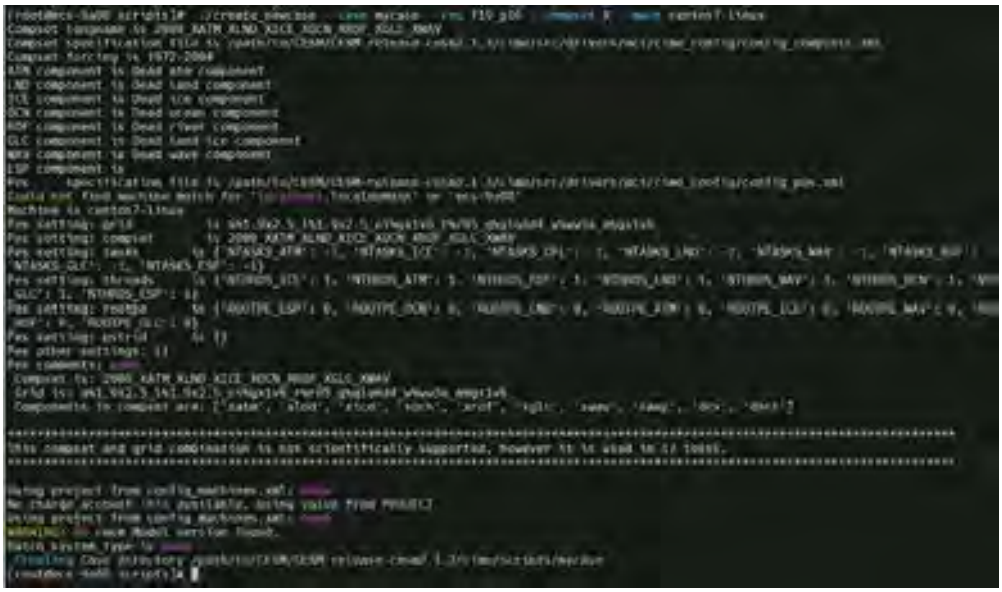
```
cd /path/to/CESM/CESM-cesm2.1.3/cime/scripts
```

Step 5 Create a test case.

```
./create_newcase --case mycase --compset X --res f19_g16 --mach=centos7-linux
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



Step 6 Download the input data.

```
cd mycase
./check_case
./check_input_data --download
```

NOTE

After a module is successfully downloaded, **SUCCESS** is displayed. You do not need to perform any operations before the whole download process automatically completes.

Step 7 Start the CESM program.

```
./case.setup
```

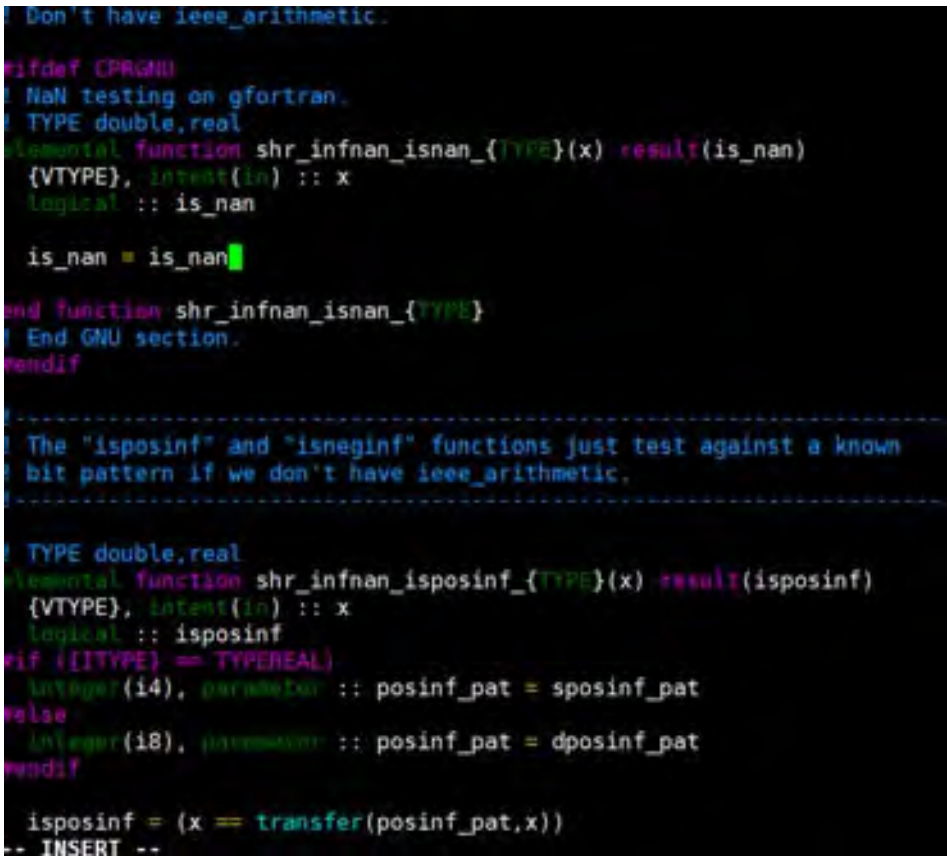
Step 8 Check the compilation options in the **Macro.make** file. The compilation options in the **Macro.make** file must be the same as those in the **config_compilers.xml** file. The options are <CFLAGS>, <FFLAGS>, <MPICC>, <MPICXX>, <MPIFC>, <SCC>, <SCXX>, <SFC>, <LDFLAGS>, and <SLIBS>.

```
SUPPORTS_CXX := FALSE
ifeq ($(COMPILER),gnu)
FC_AUTO_R8 :=  -fdefault-real-8
HAS_F2008_CONTIGUOUS := FALSE
FFLAGS_NOOPT :=  -O0
SUPPORTS_CXX := TRUE
FFLAGS :=   -fconvert=big-endian -O3 -mcpu=tsv110 -Hx,54,0x08 -I/path/to/NETCDF/include -I/path/to/HDF5/include -I/path/to/OPENBLAS/include
FIXEDFLAGS :=  -ffixed-form
SCC :=  /path/to/bisheng/bin/clang
SFC :=  /path/to/bisheng/bin/flang
MPICC :=  /path/to/HMPI/bin/mpicc
CFLAGS :=   -O3 -mcpu=tsv110 -Hx,54,0x08
MPIFC :=  /path/to/HMPI/bin/mpifort
MPICXX :=  /path/to/HMPI/bin/mpicxx
FREEFLAGS :=  -ffree-form
CXX_LINKER := FORTRAN
SCXX :=  /path/to/bisheng/bin/clang++
endif
ifeq ($(MODEL),pop)
CPPDEFS := $(CPPDEFS)  -D_USE_FLOW_CONTROL
endif
ifeq ($(COMPILER),gnu)
CPPDEFS := $(CPPDEFS)  -DFORTRANUNDERSCORE -DNO_R16 -DCPRGNU
SLIBS := $(SLIBS)  -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz
```

```
ifeq ($(compile_threaded),true)
FFLAGS := $(FFLAGS)  -fopenmp
CFLAGS := $(CFLAGS)  -fopenmp
endif
ifeq ($(DEBUG),TRUE)
FFLAGS := $(FFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=zero,overflow -fcheck=bounds
CFLAGS := $(CFLAGS)  -g -Wall -Og -fbacktrace -ffpe-trap=invalid,zero,overflow -fcheck=bounds
endif
ifeq ($(DEBUG),FALSE)
FFLAGS := $(FFLAGS)  -O
CFLAGS := $(CFLAGS)  -O
endif
ifeq ($(compile_threaded),true)
LDFLAGS := $(LDFLAGS)  -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -L/path/to/OPENBLAS/lib -lopenblas -lnetcdff -lnetcdf -lpnetcdf -lhdf5_hl -lhdf5 -lm -lz
endif
endif
```

Step 9 Modify the **CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in** file.

1. Open the **CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in** file.
   ```
   vi /path/to/CESM/CESM-cesm2.1.3/cime/src/share/util/shr_infnan_mod.F90.in
   ```
2. Press **i** to enter the edit mode and modify the content in line 227, as shown by the cursor in the following figure:



NOTE

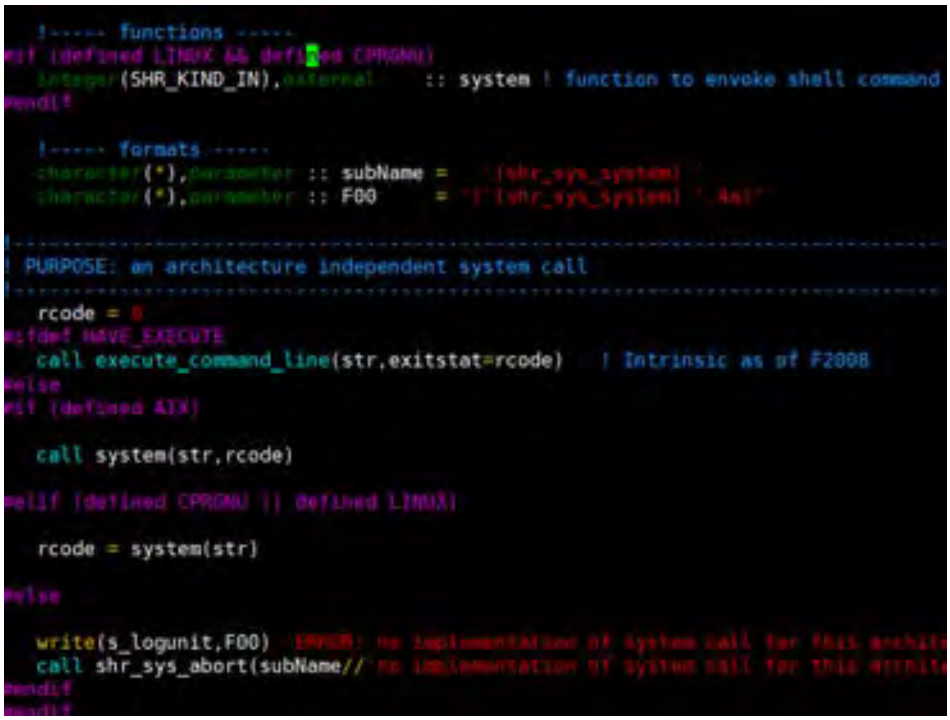
After performing Step 9.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

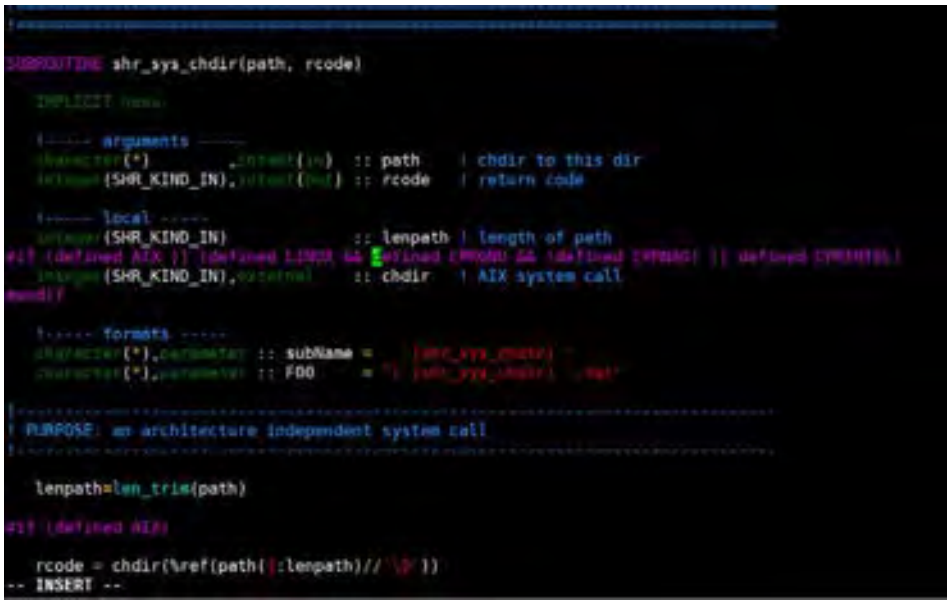
Step 10 Modify the **CESM-release-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90** file.

1. Open the **CESM-release-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90** file.
   vi */path/to/CESM*/**CESM-cesm2.1.3/cime/src/share/util/shr_sys_mod.F90**
2. Press **i** to enter the edit mode and modify the file as follows:

   Delete the exclamation mark (!) before **defined CPRGNU** in line 70.



   Delete the exclamation mark (!) before **defined CPRGNU** in line 115.



   NOTE

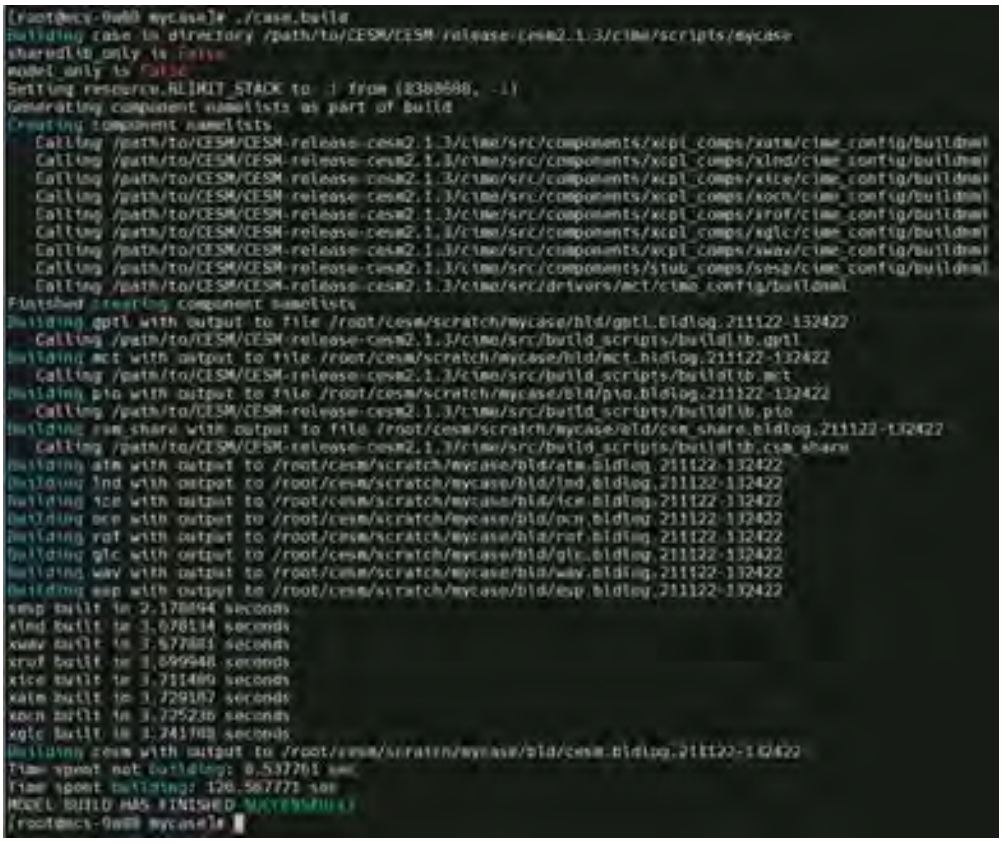
   After performing Step 10.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 11 Build the **cesm.exe** file.

- Method 1:
  ```
  ./case.build --skip-provenance-check
  ```
- Method 2:
  ```
  ./case.build
  ```

  The operation is successful if the following information in is displayed:



NOTE

You need to soft link CMake3 in the system to CMake and specify the CMake path. If you are the root user and the CMake version is 3.0 or later, ignore **ln -sf /usr/bin/cmake3 /usr/bin/cmake**.

```
ln -sf /usr/bin/cmake3 /usr/bin/cmake
```

Step 12 Execute **cesm.exe**.

```
./case.submit
```

If the information shown in the following figure is displayed when the CESM program is complete, the CESM program completes properly.

View the logs displayed on the screen and manually calculate the time difference. A smaller value indicates higher performance. As shown in the following figure, in a dual-node scenario, the start time is 2021-10-22 20:27:51 and the end time is 2021-10-22 20:28:29. Therefore, the execution duration of the test case is 38s.

----End

1.33.8 More Resources

GitHub developer forum:

https://esmci.github.io/cime/index.html

1.34 NEMO 3.6 Porting Guide (openEuler 21.03)

1.34.1 Introduction

Nucleus for European Modelling of the Ocean (NEMO) is a state-of-the-art modeling framework for research activities and forecasting services in ocean and climate sciences.

For more information about NEMO, visit the **NEMO official website**.

Programming language: C++

Brief description: A state-of-the-art modeling framework for research activities and forecasting services in ocean and climate sciences.

Open source license: Artistic License 2.0

Recommended Software Version

NEMO v3.6

1.34.2 Environment Requirements

Hardware Requirements

Table 1-174 lists the hardware requirements.

Table 1-174 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-175 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-175 Software requirements

Item	Version	Download URL
XIOS	1.0	https://forge.ipsl.jussieu.fr/ioserver/svn/XIOS1/branches/xios-1.0/
Blitz	1.0.1	https://codeload.github.com/blitzpp/blitz/tar.gz/refs/tags/1.0.1
PnetCDF	1.9.0	http://cucis.ece.northwestern.edu/projects/PnetCDF
NetCDF	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NetCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
NEMO	v3.6	http://forge.ipsl.jussieu.fr/nemo/browser/NEMO/releases/release-3.6
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
BiSheng Compiler	1.3.3	https://www.hikunpeng.com/en/developer/devkit/compiler?data=bisheng
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi
AMM12 test case file	v3.6	https://zenodo.org/record/2640767/files/AMM12_v3.6.tar?download=1

OS Requirements

Table 1-176 lists the OS requirements.

Table 1-176 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

1.34.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NEMO software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-177 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/hypermpi	Installation path of Hyper MPI.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/PNETCDF	Installation path of PnetCDF.	
3	/path/to/HDF5	Installation path of HDF5.	
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	/path/to/BLITZ	Installation path of Blitz.	
6	/path/to/XIOS	Installation path of XIOS.	
7	/path/to/NEMO	Installation path of NEMO.	

1.34.4 Configuring the Compilation Environment

1.34.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-178 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing PnetCDF	See **1.34.4.2 Installing PnetCDF**.
4	Installing HDF5	See **1.34.4.3 Installing HDF5**.
5	Installing NetCDF	See **1.34.4.4 Installing NetCDF**.
6	Installing NetCDF-Fortran	See **1.34.4.5 Installing NetCDF-Fortran**.
7	Installing Blitz	See **1.34.4.6 Installing Blitz**.
8	Installing XIOS	See **1.34.4.7 Installing XIOS**.

1.34.4.2 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.9.0.tar.bz
```

Step 3 Go to the directory generated after the decompression.

```
cd parallel-netcdf-1.9.0
```

Step 4 Add the environment variables.

```
export CC=clang
export CXX=clang++
export FC=flang
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

NOTE

After that, you do not need to add them again during subsequent installation steps.

Step 5 Perform the configuration and installation.

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
make
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.34.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Set the environment variables.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.34.4.4 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.4.1.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L$HDF5/lib" CPPFLAGS="-I$HDF5/include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the compilation and installation.

 Copyright © Huawei Technologies Co., Ltd.

```
make
make install
```

Step 6 After the installation is complete, load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.34.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$HDF5/include -I$NETCDF/include" LDFLAGS="-L$HDF5/lib -L$NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-static=yes --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify lines 10118 and 10270 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 After the installation is complete, load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.34.4.6 Installing Blitz

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Blitz installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
tar -xvf blitz-1.0.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd blitz-1.0.1/
```

Step 4 Perform the configuration.

```
autoreconf -fiv
./configure --prefix=/path/to/BLITZ --enable-fortran --enable-64bit
```

Step 5 Modify the **genstencils.py** file.

1. Open the **genstencils.py** file.
   ```
   vi blitz/generate/genstencils.py
   ```
2. Press **i** to enter the edit mode and modify line 9 in the **genstencils.py** file.
   ```
   print("Generating file %s"%sys.argv[1])
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make lib
make install
cp -rf ./src /path/to/BLITZ
```

Step 7 Load the environment variables.

```
export BLITZ_INCLUDEDIR=/path/to/BLITZ/include
export LD_LIBRARY_PATH=/path/to/BLITZ/lib:$LD_LIBRARY_PATH
```

----End

1.34.4.7 Installing XIOS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory generated after the decompression.

```
cd /path/to/XIOS
```

Step 3 Download the XIOS source code from the Internet.

```
svn co -r 703 https://forge.ipsl.jussieu.fr/ioserver/svn/XIOS1/branches/xios-1.0/ xios-1.0
```

Step 4 Install the system dependencies.

```
yum install -y perl-lib* boost-*
```

Step 5 Go to the directory generated after the decompression.

```
cd xios-1.0
```

Step 6 Modify the **bld.cfg** file.

1. Open the **bld.cfg** file.
   ```
   vi bld.cfg
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   search_src          true
   src::zzz .
   #src::date $PWD/extern/boost/src/date_time
   src::blitz $PWD/extern/blitz/src
   src::netcdf $PWD/extern/netcdf4
   ```

```
bld::lib xios
bld::target libxios.a
#bld::target generate_fortran_interface.exe
#bld::target xios_server.exe test_client.exe parse_xml.exe test_complete.exetest_xios_interface.exe
bld::exe_dep
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Create an **arch-AARCH64_GNU_LINUX.env** configuration file.

1. Create an **arch-AARCH64_GNU_LINUX.env** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.env
   ```
2. Press **i** to enter the edit mode. Create and edit the **arch-AARCH64_GNU_LINUX.env** file.
   ```
   export HDF5_INC_DIR="/path/to/HDF5/include"
   export HDF5_LIB_DIR="/path/to/HDF5/lib"
   export NETCDF_INC_DIR="/path/to/NETCDF/include"
   export NETCDF_LIB_DIR="/path/to/NETCDF/lib"
   export BOOST_INC_DIR="/usr/include/boost/"
   export BOOST_LIB_DIR="/usr/lib64/"
   export BLITZ_INC_DIR="/path/to/BLITZ/include"
   export BLITZ_LIB_DIR="/path/to/BLITZ/lib"
   ```

   NOTE

   Edit the file based on the actual environment variables and the dependency library installation path.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Create an **arch-AARCH64_GNU_LINUX.fcm** configuration file.

1. Create an **arch-AARCH64_GNU_LINUX.fcm** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.fcm
   ```
2. Press **i** to enter the edit mode. Create and edit the **arch-AARCH64_GNU_LINUX.fcm** file.
   ```
   ############################################################################
   ###########
   ####################       Projet xios - xmlioserver      ####################
   ############################################################################
   ###########
   %CCOMPILER      mpicc
   %FCOMPILER      mpif90
   %LINKER         mpif90
   %BASE_CFLAGS    -ansi -w -D_GLIBCXX_USE_CXX11_ABI=0
   %PROD_CFLAGS    -O3 -DBOOST_DISABLE_ASSERTS
   %DEV_CFLAGS     -g -O2
   %DEBUG_CFLAGS   -g
   %BASE_FFLAGS    -D__NONE__
   %PROD_FFLAGS    -O3
   %DEV_FFLAGS     -g -O2
   %DEBUG_FFLAGS   -g
   %BASE_INC       -D__NONE__
   %BASE_LD        -lstdc++
   %CPP            cpp
   %FPP            cpp -P
   %MAKE           gmake
   ```

   NOTE

   Edit the file based on the actual environment variables and the dependency library installation path.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Create an **arch-AARCH64_GNU_LINUX.path** configuration file.

 Copyright © Huawei Technologies Co., Ltd.

1. Create an **arch-AARCH64_GNU_LINUX.path** configuration file.
   ```
   vi arch/arch-AARCH64_GNU_LINUX.path
   ```
2. Press **i** to enter the edit mode. Create and edit the **arch-AARCH64_GNU_LINUX.path** file. Edit the file based on the actual environment variables and the dependency library installation path.
   ```
   NETCDF_INCDIR="-I /path/to/NETCDF/include"
   NETCDF_LIBDIR="-L /path/to/NETCDF/lib"
   NETCDF_LIB="-lnetcdff -lnetcdf"
   MPI_INCDIR="-I /path/to/OPENMPI/include"
   MPI_LIBDIR="-L /path/to/OPENMPI/lib"
   MPI_LIB="-lmpi"
   HDF5_INCDIR="-I /path/to/HDF5/include"
   HDF5_LIBDIR="-L /path/to/HDF5/lib"
   HDF5_LIB="-lhdf5_hl -lhdf5  -lz"
   BOOST_INCDIR="-I /usr/include/boost/"
   BOOST_LIBDIR="-L /usr/lib64/"
   BOOST_LIB=""
   BLITZ_INCDIR="-I /path/to/BLITZ/include"
   BLITZ_LIBDIR="-L /path/to/BLITZ/lib"
   BLITZ_LIB="-lblitz"
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Perform the first compilation and installation.

```
cd /path/to/XIOS/xios-1.0
chmod +x ./make_xios
cd extern
ln -s /path/to/BLITZ ./blitz
ln -s /path/to/NETCDF ./netcdf4
ln -s /usr/include/boost ./boost
cd ..
./make_xios --dev --job 32 --full --arch AARCH64_GNU_LINUX
```

----End

1.34.5 Obtaining the Source Code

Procedure

Step 1 Download the NEMO installation package **NEMO_releases_release-3.6-10083.zip**.

http://www.nemo-ocean.eu/

Step 2 Use an SFTP tool to upload the NEMO installation package to the */path/to/NEMO* directory on the server.

----End

1.34.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the NEMO installation directory.

```
cd /path/to/NEMO
```

Step 3 Set the NEMO environment variables.

```
echo "export NETCDF_DIR=/path/to/NETCDF" >> nemo-env.sh
echo "export HDF_DIR=/path/to/HDF5" >> nemo-env.sh
echo "export XIOS_DIR=/path/to/XIOS/xios-1.0" >> nemo-env.sh
source nemo-env.sh
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Decompress the installation package.

```
unzip NEMO_releases_release-3.6-15515.zip
```

Step 5 Go to the NEMO source code directory.

```
cd NEMO/releases/release-3.6/NEMOGCM
```

Step 6 Create an **arch-aarch64_gnu.fcm** file.

1. Create an **arch-aarch64_gnu.fcm** file.

   ```
   vi ARCH/arch-aarch64_gnu.fcm
   ```

2. Press **i** to enter the edit mode and add the following content:

   ```
   #generic gfortran compiler options for linux
   #NCDF_INC    netcdf include file
   #NCDF_LIB    netcdf library
   #FC          Fortran compiler command
   #FCFLAGS     Fortran compiler flags
   #FFLAGS      Fortran 77 compiler flags
   #LD          linker
   #LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
   #FPPFLAGS    pre-processing flags
   #AR          assembler
   #ARFLAGS     assembler flags
   #MK          make
   #USER_INC    additional include files for the compiler,  e.g. -I<include dir>
   #USER_LIB    additional libraries to pass to the linker, e.g. -l<library>
   #CC          C compiler used to compile conv for AGRIF
   #CFLAGS      compiler flags used with CC
   #
   #Note that:
   # - unix variables "$..." are accpeted and will be evaluated before calling fcm.
   # - fcm variables are starting with a % (and not a $)
   %NCDF_HOME           $NETCDF_DIR
   %HDF5_HOME           $HDF_DIR
   %XIOS_HOME           $XIOS_DIR
   %NCDF_INC            -I%NCDF_HOME/include -I%HDF5_HOME/include
   %NCDF_LIB            -L%HDF5_HOME/lib -L/%NCDF_HOME/lib -lnetcdf -lnetcdff
   %XIOS_INC            -I%XIOS_HOME/inc
   %XIOS_LIB            -L%XIOS_HOME/lib -lxios
   %CPP                 cpp
   %FC                  mpifort
   %FCFLAGS             -cpp -r8 -O3 -mcpu=native
   %FFLAGS              %FCFLAGS
   %LD                  mpif90
   %LDFLAGS             -lstdc++
   %FPPFLAGS            -P -C -traditional
   %AR                  ar
   %ARFLAGS             rs
   %MK                  gmake
   %USER_INC            %XIOS_INC %NCDF_INC
   %USER_LIB            %XIOS_LIB %NCDF_LIB
   %CC                  clang
   %CFLAGS              -O0
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Compile and install the software.

```
chmod 777 ./ -R
cd ./CONFIG/
./makenemo -m aarch64_gnu -j 32 -r AMM12 -n 'MY_AMM12' add_key "key_nosignedzero"
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.34.7 Running and Verification

NOTE

- Determine the number of running processes based on the server configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the servers.
  - Enable password-free SSH login between the servers.
  - Disable the firewall and ensure that clients and servers are not disconnected.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the working directory.

```
cd /path/to/NEMO/NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00
```

Step 3 Upload **AMM12_v3.6.tar** to */path/to/NEMO*/**NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00**.

Step 4 Decompress the file.

```
tar -vxf AMM12_v3.6.tar
gunzip *.gz */*.gz
rm domain_def.xml field_def.xml namelist_ref -f
ln -s ../../SHARED/domain_def.xml ./domain_def.xml
ln -s ../../SHARED/field_def.xml ./field_def.xml
ln -s ../../SHARED/namelist_ref ./namelist_ref
```

Step 5 Create a **host** file.

1. Create a **host** file.
   ```
   vi hostfile
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   node1
   node2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Start the NEMO program.

```
{ time mpirun --allow-run-as-root -np 16 -N 8 -hostfile hostfile --mca btl ^openib ./opa ; } 2>&1 | tee -a NEMO.log
```

Table 1-179 Parameter description

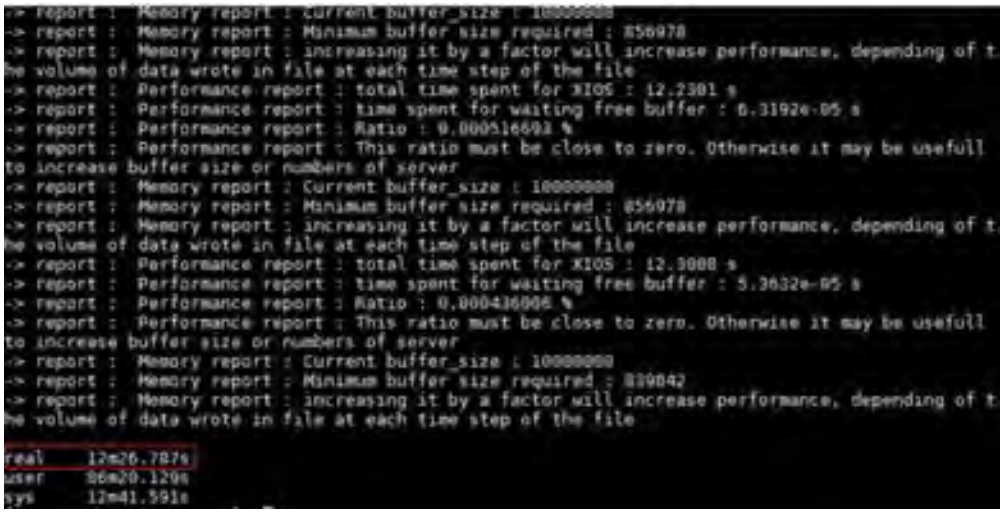
Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
--hostfile	Name of the node to be used.

NOTE

If you want to run the program on a single node, delete the **--hostfile** parameter.

 Copyright © Huawei Technologies Co., Ltd.

If the information shown in the following figure is displayed at the end of the **ocean.output** log generated by NEMO, the NEMO program stops normally.



Check the value of **real** (unit: s) in the **NEMO.log** file. A larger value indicates lower performance.

----End

1.35 ROMS 4.0 Porting Guide (openEuler 21.03)

1.35.1 Introduction

The Regional Ocean Modeling System (ROMS) is a 3D regional ocean modeling system developed by the Rutgers University, University of California Los Angeles. It is widely used to simulate the hydrodynamic and water environment in the ocean and estuary.

For more information about ROMS, visit the **ROMS official website**.

Programming language: Fortran

Brief description: 3D regional ocean modeling system.

Open source license: MIT/X License

Recommended Software Version

ROMS 4.0

1.35.2 Environment Requirements

Hardware Requirements

Table 1-180 lists the hardware requirements.

Table 1-180 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-181 lists the software requirements.

Table 1-181 Software requirements

Item	Version	Download URL
ROMS	4.0	https://www.myroms.org/svn/src/trunk
HDF5	1.10.1	https://hdfgroup.org/solutions/hdf5/
PnetCDF	1.9.0	https://parallel-netcdf.github.io/wiki/Download.html
NetCDF	4.4.1.1	https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
NetCDF-Fortran	4.4.1	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1
BiSheng Compiler	1.3.3	https://www.hikunpeng.com/en/developer/devkit/compiler?data=bisheng
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi
Test cases	roms_upwelling	Provided by the software.

OS Requirements

Table 1-182 lists the OS requirements.

Table 1-182 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

1.35.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ROMS software porting.

Table 1-183 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/hypermpi	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
2	/opt/compiler	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
3	/path/to/HDF5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	/path/to/PNETCDF	Installation path of PnetCDF.	
6	/path/to/ROMS	Installation path of ROMS.	

1.35.4 Configuring the Compilation Environment

1.35.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-184 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing HDF5	See **1.35.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.35.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.35.4.4 Installing NetCDF-Fortran**.
6	Installing PnetCDF	See **1.35.4.5 Installing PnetCDF**.

1.35.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar zxvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.35.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.4.1.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Set the NetCDF environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.35.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar zxvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-static=yes --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify lines 10118 and 10270 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 After the installation is complete, load the environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.35.4.5 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -xvf parallel-1.9.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pnetcdf-1.9.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

----End

1.35.5 Obtaining the Source Code

Procedure

Step 1 Download the ROMS installation package.

https://www.myroms.org/

NOTE

To download the ROMS installation package, you need to register an account on the ROMS website and log in to the website using the registered account.

Step 2 Use an SFTP tool to upload the ROMS installation package to the */path/to/ROMS* directory on the server.

----End

1.35.6 Compilation and Installation

Procedure

Step 1 Use SVN to check out and obtain the ROMS source code.

 Copyright © Huawei Technologies Co., Ltd.

Create a folder and check out the ROMS source code to the folder. To obtain the source code, you need to register an account.

```
cd /path/to/ROMS
mkdir ROMS_4.0
svn checkout --username Account name --password Password https://www.myroms.org/svn/src/trunk ROMS_4.0
```

Example:

Check out the source code using **HPC/Huawei12#$**:

svn checkout --username HPC --password Huawei12#$ https://www.myroms.org/svn/src/trunk ROMS_4.0

The source code file structure is as follows:

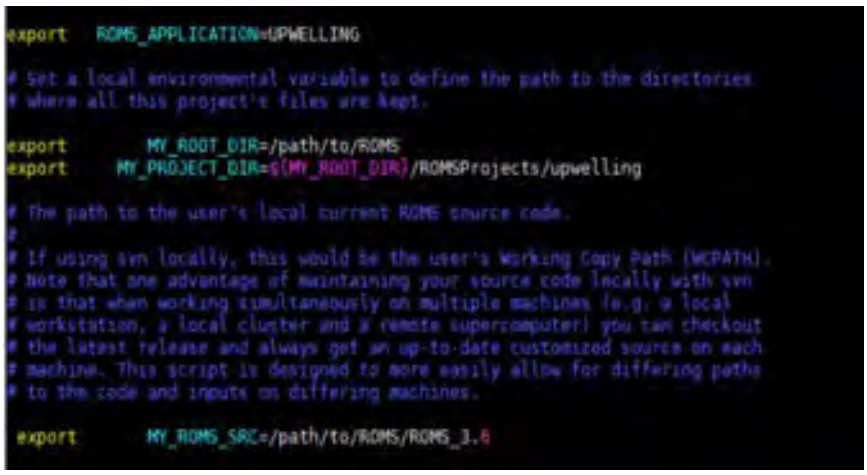
```
trunk/                          Main trunk directory
     /Atmosphere/               Atmosphere models root directory
               /COAMPS          COAMPS root directory (empty)
               /WRF             WRF root directory (empty)
     /Compilers/                make configuration files
     /Data/                     Input data root directory
          /ROMS/                ROMS data root directory
               /CDL             ROMS NetCDF metadata design
               /Forcing         Input test cases forcing NetCDF files
               /Grid            Input test cases grid NetCDF files
               /Initial         Input test cases initial conditions NetCDF files
     /Lib/                      External libraries
         /ARPACK                Arpack eigenvalue problems library
         /MCT                   Modeling Coupling Tool library
     /Master                    Main standalone and coupling programs
     /ROMS/                     ROMS root directory
          /Adjoint              Adjoint model
                  /Biology      Adjoint biology/ecosystem models
          /Bin                  Executable scripts
          /Drivers              Computational drivers
          /External             Standard input scripts
          /Functionals          Analytical expression header files
          /Include              Test cases configuration header files
          /Modules              Declaration modules
          /Nonlinear            Nonlinear model
                   /Biology     Nonlinear biology/ecosystem models
                   /Sediment    Nonlinear sediment transport model
          /Obsolete             Discontinued files
          /Programs             Support programs
          /SeaIce               Sea-ice model (empty)
          /Representer          Representer model
                     /Biology   Representer biology/ecosystem models
          /Tangent              Tangent linear model
                  /Biology      Tangent linear biology/ecosystem models
          /Utility              Generic utility files
          License_ROMS.text     Open source license
          Version               SVN Version information
     /User/                     ROMS User interface root directory
          /External             User standard input scripts
          /Functionals          User analytical expressions templates
          /Include              User application header files
     /WAVES/                    Waves models root directory
           /SWAN                SWAN root directory
                /External       SWAN input data and standard input files
                /Src            SWAN model
```

Step 2 Customize the build script.

1. To keep the source code unchanged, run the following commands to create a **ROMSProjects** folder for compiling and running ROMS. **upwelling** is the default test case included in the source package.

   ```
   cd /path/to/ROMS
   mkdir ROMSProjects
   cd ROMSProjects
   mkdir upwelling
   cd upwelling
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Copy the following files to the **upwelling** directory.
   ```
   cp /path/to/ROMS/ROMS_4.0/ROMS/Bin/build_roms.sh .
   cp /path/to/ROMS/ROMS_4.0/ROMS/External/roms_upwelling.in .
   cp /path/to/ROMS/ROMS_4.0/ROMS/External/varinfo.dat .
   cp /path/to/ROMS/ROMS_4.0/ROMS/Include/upwelling.h .
   ```
3. Edit the **build_roms.sh** file.
   a. Open the **build_roms.sh** file.
      ```
      vi build_roms.sh
      ```
   b. Press **i** to enter the edit mode.
      - Set the ROMS root directory and **upwelling** path and modify the following content:
        ```
        export ROMS_APPLICATION=UPWELLING
        export MY_ROOT_DIR=/path/to/ROMS
        export MY_PROJECT_DIR=${MY_ROOT_DIR}/ROMSProjects/upwelling
        export MY_ROMS_SRC=/path/to/ROMS/ROMS_4.0
        ```
        

      - Set compilation options and modify the following content:
        ```
        export USE_MPI=on
        export USE_MPIF90=on
        export which_MPI=openmpi
        export FORT=flang
        export USE_NETCDF4=on
        ```

 Copyright © Huawei Technologies Co., Ltd.

```
export          USE_MPI=on          # distributed-memory parallelism
export        USE_MPIF90=on         # compile with mpif90 script
#export       which_MPI=intel       # compile with mpiifort library
#export       which_MPI=mpich       # compile with MPICH library
#export       which_MPI=mpich2      # compile with MPICH2 library
#export       which_MPI=mvapich2    # compile with MVAPICH2 library
 export       which_MPI=openmpi     # compile with OpenMPI library

#export       USE_OpenMP=on         # shared-memory parallelism

 export            FORT=flang
#export            FORT=gfortran
#export            FORT=pgi

#export       USE_DEBUG=on          # use Fortran debugging flags
 export       USE_LARGE=on          # activate 64-bit compilation

# ROMS I/O choices and combinations. A more complete description of the
# available options can be found in the wiki: (https://myroms.org/wiki/IO).
# Most users will want to enable at least USE_NETCDF4 because that will
# instruct the ROMS build system to use nf-config to determine the
# necessary libraries and paths to link into the ROMS executable.

export      USE_NETCDF4=on          # compile with NetCDF-4 library
#export  USE_PARALLEL_IO=on         # Parallel I/O with NetCDF-4/HDF5
#export       USE_PIO=on           # Parallel I/O with PIO library
#export    USE_SCORPIO=on          # Parallel I/O with SCORPIO library
```

Comment out **export USE_MY_LIBS=no** and use **export USE_MY_LIBS=yes**.

```
#-----------------------------------------------------------------------
# If applicable, use my specified library paths.
#-----------------------------------------------------------------------

#export USE_MY_LIBS=no          # use system default library paths
export USE_MY_LIBS=yes          # use my customized library paths
```

c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Edit the **my_build_path.sh** file.

1. Open the **my_build_path.sh** file.
   cd */path/to/ROMS*/**ROMS_4.0/Compilers**
   vi my_build_path.sh
2. Press **i** to enter the edit mode and modify the file as follows:
   Change **gfortran** in line 83 to **flang**.
   Change **MPI_ROOT** in line 87 to **MPI_ROOT=/path/to/hypermpi/hmpi**.

```
flang )
  if [ "${which_MPI}" = "mpich2" ]; then
    export        MPI_ROOT=/opt/gfortransoft/mpich2
  elif [ "${which_MPI}" = "openmpi" ]; then
    export        MPI_ROOT=/path/to/hypermpi/hmpi
  elif [ "${which_MPI}" = "mvapich2" ]; then
    export        MPI_ROOT=/opt/gfortransoft/mvapich2
  fi
  export              PATH=${MPI_ROOT}/bin:$PATH
  export       MPI_INCDIR=${MPI_ROOT}/include
  ;;
```

   Change **gfortran** in lines 295 and 296 to **flang**.
   Change **MPI_ROOT** in line 311 to **MPI_ROOT=/path/to/hypermpi/hmpi**.

 Copyright © Huawei Technologies Co., Ltd.

```
flang )
  export      ESMF_COMPILER=flang
  if [ -n "${USE_DEBUG:+1}" ]; then
    export        ESMF_BOPT=g
  else
    export        ESMF_BOPT=O
  fi
  export          ESMF_ABI=64
  export         ESMF_COMM=$(which_MPI)
  export         ESMF_SITE=default

  export      ARPACK_LIBDIR=/opt/gfortransoft/serial/ARPACK
  if [ -n "${USE_MPI:+1}" ]; then
    if [ "$(which_MPI)" = "mpich2" ]; then
      export       MPI_SOFT=/opt/gfortransoft/mpich2
    elif [ "$(which_MPI)" = "openmpi" ]; then
      export       MPI_SOFT=/path/to/hypermpi/hmpi
    elif [ "$(which_MPI)" = "mvapich2" ]; then
      export       MPI_SOFT=/opt/gfortransoft/mvapich2
```

Change **NETCDF=** in line 330 to **NETCDF=/path/to/NETCDF**.

```
if [ ! -n "${SINGULARITY_COMMAND:+1}" ]; then
  if [ -n "${USE_NETCDF4:+1}" ]; then
    if [ -n "${USE_PARALLEL_IO:+1}" ] && [ -n "${USE_MPI:+1}" ]; then
      export       ESMF_DIR=${MPI_SOFT}/esmf_nc4
      export         NETCDF=${MPI_SOFT}/netcdf4
      export      NF_CONFIG=${NETCDF}/bin/nf-config
      export  NETCDF_INCDIR=${NETCDF}/include
      export        NETCDF4=1
    else
      export       ESMF_DIR=${MPI_SOFT}/esmf_nc4
      export         NETCDF=/path/to/NETCDF
      export      NF_CONFIG=${NETCDF}/bin/nf-config
```

NOTE

After performing **Step 3.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Place the **Linux-flang.mk** file in **/path/to/ROMS/ROMS_4.0/Compilers/**.

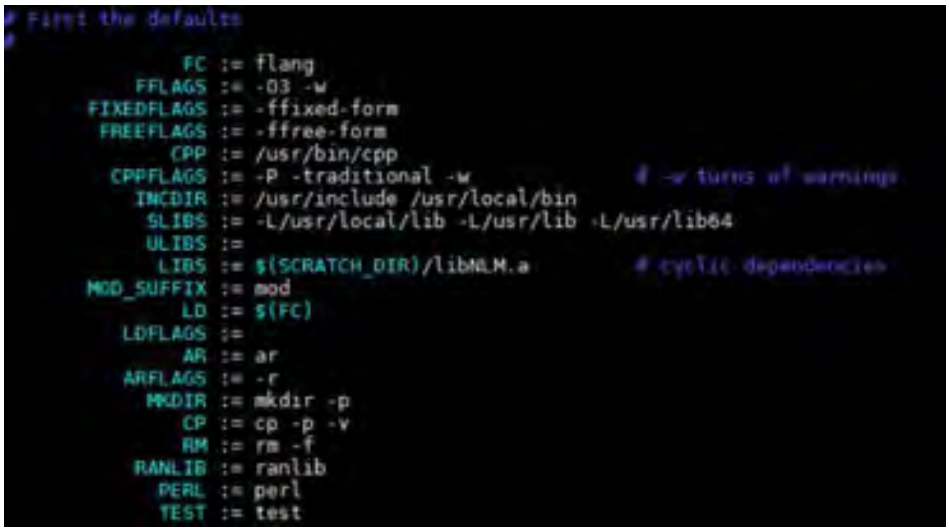
```
cd /path/to/ROMS/ROMS_4.0/Compilers/
```

Compile **Linux-flang.mk** as follows:

```
cp Linux-gfortran.mk ../
mv Linux-gfortran.mk Linux-flang.mk
vi Linux-flang.mk
```

Modify it as follows:

 Copyright © Huawei Technologies Co., Ltd.



Press **Esc** and type **:wq!** to save the file and exit.

Step 5 Modify the **roms_upwelling.in** file.

1. Open the **roms_upwelling.in** file.
   cd */path/to/ROMS*/**ROMSProjects/upwelling/**
   vi roms_upwelling.in
2. Press **i** to enter the edit mode and change the path of **varinfo.dat** in **VARNAME**.
   VARNAME = */path/to/ROMS*/ROMSProjects/upwelling/varinfo.dat

   Set **NtileI** and **NtileJ**. The product of **NtileI** and **NtileJ** must be equal to the value of **np** (number of cores).

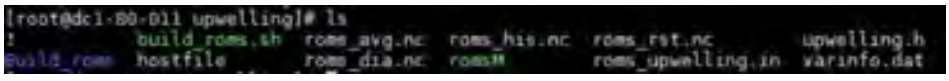
   NtileI=2

   NtileJ=4
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the installation.

chmod 777 build_roms.sh
chmod 777 */path/to/ROMS*/**ROMS_4.0/ROMS/Bin/cpp_clean**
chmod 777 */path/to/ROMS*/**ROMS_4.0/ROMS/Bin/sfmakedepend**
./build_roms.sh -j

If the executable file **romsM** is generated in the directory and no error is reported during the installation, the installation is successful.



----End

 Copyright © Huawei Technologies Co., Ltd.

1.35.7 Running and Verification

NOTE

- Determine the number of running processes based on the server configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the servers.
  - Enable password-free SSH login between the servers.
  - Disable the firewall and ensure that clients and servers are not disconnected.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the working directory.

```
cd /path/to/ROMS/ROMSProjects/upwelling
```

Step 3 Start the program.

```
mpirun --allow-run-as-root -n 8 ./romsM roms_upwelling.in
```

Check the value of **Node# CPU:** (unit: s) in the log file. A smaller value indicates higher performance.

Figure 1-25 shows the command output.

Figure 1-25 Example result

```
Elapsed wall CPU time for each process (seconds):

Node   #    0 CPU:      35.471
Total:                 285.053
Average:                35.632
Minimum:                35.471
Maximum:                35.690
```

----End

1.36 ncl 6.3.0 Porting Guide (openEuler 20.03)

1.36.1 Introduction

NCL is designed specifically for scientific data processing and visualization, widely used in the area of atmospheric modeling.

For more information about NCL, visit the **NCL official website**.

Programming languages: C/Fortran

Brief description: data processing and visualization.

Recommended Software Version

NCL 6.3.0

 Copyright © Huawei Technologies Co., Ltd.

1.36.2 Environment Requirements

Hardware Requirements

Table 1-185 lists the hardware requirements.

Table 1-185 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-186 lists the software requirements.

Table 1-186 Software requirements

Item	Version	Download URL
NCL	6.3.0	https://download-ib01.fedoraproject.org/pub/epel/7/SRPMS/Packages/n/ncl-6.3.0-6.el7.src.rpm
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
HDF5	1.8.12	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.12/src/hdf5-1.8.12.tar.gz
NetCDF	4.3.3.1	https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.3.3.1.tar.gz
NetCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.5.tar.gz
HDF	4.2.14-5	https://download-ib01.fedoraproject.org/pub/epel/8/Everything/SRPMS/Packages/h/hdf-4.2.14-5.el8.src.rpm
GDAL	2.2.4	http://download.osgeo.org/gdal/2.2.4/gdal-2.2.4.tar.gz
Jasper	1.900.2	https://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.2.tar.gz
G2Clib	1.6.0-patch	http://www.ncl.ucar.edu/Download/files/g2clib-1.6.0-patch.tar.gz
Udunits2	2.2.28	https://gfd-dennou.org/arch/ucar/unidata/pub/udunits/udunits-2.2.28.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 1-187 lists the OS requirements.

Table 1-187 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	https://repo.openeuler.org/openEuler-20.03-LTS-SP1/
Kernel	4.19	Included in the OS image

1.36.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NCL software porting.

Table 1-188 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/hdf5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/netcdf	Installation path of NetCDF-C.	
4	/path/to/netcdf	Installation path of NetCDF-Fortran.	
5	/path/to/gdal	Installation path of GDAL.	
6	/path/to/jasper	Installation path of Jasper.	
7	/path/to/g2clib	Installation path of G2Clib.	
8	/path/to/udunits2	Installation path of Udunits2.	

 Copyright © Huawei Technologies Co., Ltd.

1.36.4 Configuring the Compilation Environment

1.36.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-189 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **1.36.4.2 Installing the BiSheng Compiler**.
2	Installing HDF5	See **1.36.4.3 Installing HDF5**.
3	Installing NetCDF-C	See **1.36.4.4 Installing NetCDF-C**.
4	Installing NetCDF-Fortran	See **1.36.4.5 Installing NetCDF-Fortran**.
5	Installing HDF	See **1.36.4.6 Installing HDF**.
6	Installing GDAL	See **1.36.4.7 Installing GDAL**.
7	Installing Jasper	See **1.36.4.8 Installing Jasper**.
8	Installing G2Clib	See **1.36.4.9 Installing G2Clib**.
9	Installing Udunits2	See **1.36.4.10 Installing Udunits2**.

1.36.4.2 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export CC=clang
export CXX=clang++
export FC=flang
```

Step 3 Install the BiSheng Compiler. For details, see **Installing and Using BiSheng Compiler**.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.36.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export optflags="-O3"
export ldflags="-Wl,--build-id"
```

Step 3 Create an installation directory.

```
mkdir -p /path/to/hdf5
```

Step 4 Go to the installation directory.

```
cd /path/to/hdf5
```

Step 5 Obtain the source code.

```
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.12/src/hdf5-1.8.12.tar.gz
```

Step 6 Decompress the HDF5 installation package.

```
tar -zxvf hdf5-1.8.12.tar.gz
```

Step 7 Go to the directory generated after the decompression.

```
cd hdf5-1.8.12
```

Step 8 Perform the configuration.

```
./configure --prefix=/path/to/hdf5 --build=aarch64-unknown-linux-gnu --enable-shared --enable-static=yes --enable-fortran CC=clang CXX=clang++ FC=flang F77=flang CFLAGS="$optflags -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="$optflags -fPIC" FFLAGS="$optflags -fPIC" LDFLAGS="$ldflags"
```

Step 9 Create a **libtool.patch** file.

1. Create a **libtool.patch** file.
   ```
   vi libtool.patch
   ```

2. Press **i** to enter the insert mode and add the following content:

```
diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
--- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
+++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
@@ -11832,7 +11832,7 @@
pic_flag=""
# How to pass a linker flag through the compiler.
-wl=""
+wl="-Wl,"
# Compiler flag to prevent dynamic linking.
link_static_flag=""
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

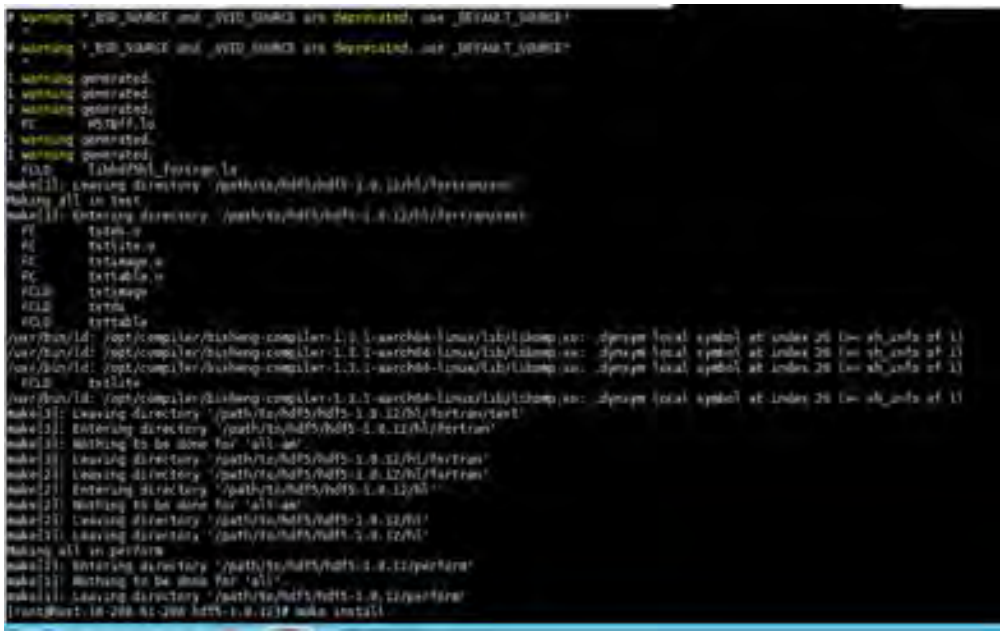
Step 10 Install the patch.

```
patch -p1 < libtool.patch
```

```
[root@host-10-208-61-208 hdf5-1.8.12]# vi libtool.patch
[root@host-10-208-61-208 hdf5-1.8.12]# patch -p1 < libtool.patch
patching file libtool
Hunk #1 succeeded at 10267 (offset -1565 lines).
[root@host-10-208-61-208 hdf5-1.8.12]# make -j 8
```

Step 11 Perform the compilation and installation.

```
make -j 8
```



```
make install
```

Step 12 Set the environment variables.

```
export PATH=/path/to/hdf5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/hdf5/lib:$LD_LIBRARY_PATH
```

----End

1.36.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

```
mkdir -p /path/to/netcdf
```

Step 3 Install the dependencies.

```
yum install curl libcurl-devel zlib-devel
```

Step 4 Go to the installation directory.

```
cd /path/to/netcdf
```

Step 5 Obtain the source package and rename it.

```
wget https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.3.3.1.tar.gz
mv v4.3.3.1.tar.gz netcdf-4.3.3.1.tar.gz
```

Step 6 Decompress the source package.

```
tar -vxf netcdf-4.3.3.1.tar.gz
```

Step 7 Go to the directory generated after the decompression.

```
cd netcdf-c-4.3.3.1/
```

Step 8 Configure the compilation.

```
./configure --prefix=/path/to/netcdf --build=aarch64-linux --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-largefile CC=clang CXX=clang++ FC=flang F77=flang CPPFLAGS="-I/path/to/hdf5/include" LDFLAGS="-L/path/to/hdf5/lib -Wl,-rpath=/path/to/hdf5/lib" CFLAGS="-L/path/to/hdf5/lib -I/path/to/hdf5/include"
```

 Copyright © Huawei Technologies Co., Ltd.

```
# General
-------
NetCDF Version:         4.3.3.1
Configured On:          Thu Dec 23 09:53:19 CST 2021
Host System:            aarch64-unknown-linux-gnu
Build Directory:        /path/to/netcdf/netcdf-c-4.3.3.1
Install Prefix:         /path/to/netcdf

# Compiling Options
-----------------
C Compiler:             /opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/bin/clang
CFLAGS:                 -L/path/to/hdf5/lib -I/path/to/hdf5/include
CPPFLAGS:               -I/path/to/hdf5/include
LDFLAGS:                -L/path/to/hdf5/lib -Wl,-rpath=/path/to/hdf5/lib
AM_CFLAGS:
AM_CPPFLAGS:
AM_LDFLAGS:
Shared Library:         yes
Static Library:         yes
Extra libraries:        -lhdf5_hl -lhdf5 -ldl -lm -lz -lcurl

# Features
--------
NetCDF-2 API:           yes
NetCDF-4 API:           yes
HDF4 Support:           no
HDF5 Support:           yes
PNetCDF Support:        no
Parallel Support:       no
DAP Support:            yes
Diskless Support:       yes
MMap Support:           no
JNA Support:            no
[root@host-10-208-167-174 netcdf-c-4.3.3.1]# make -j 8
```

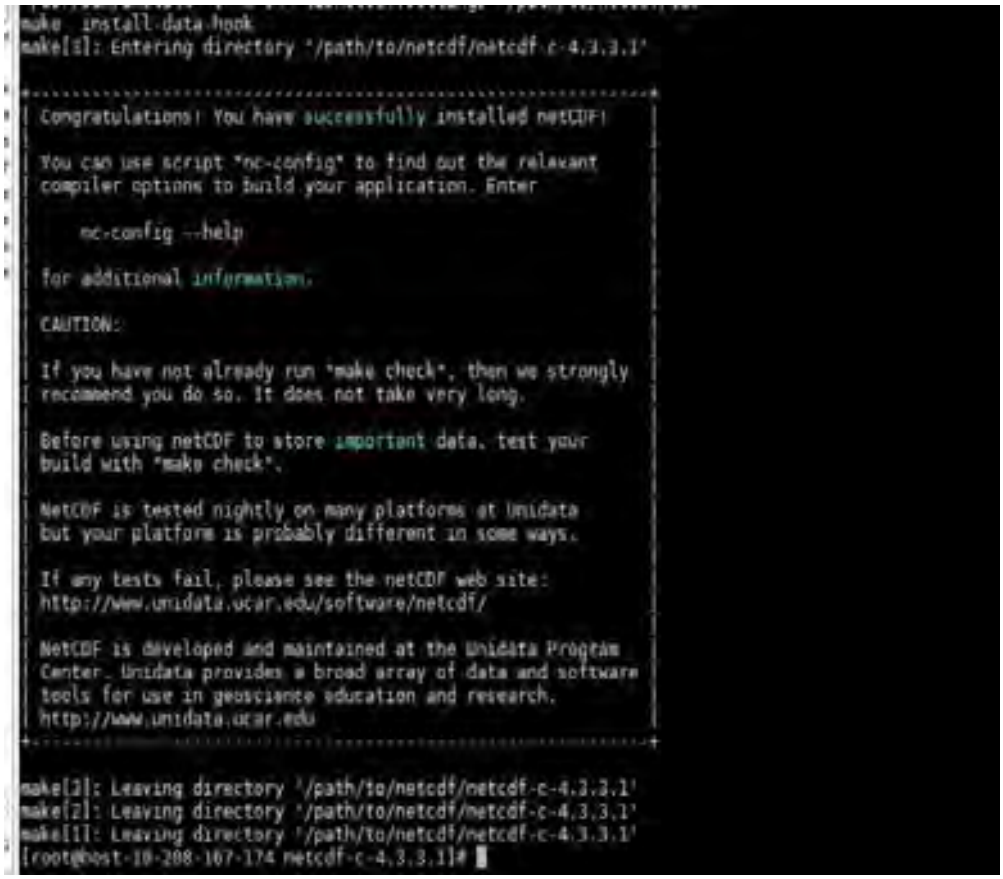
Step 9 Perform the compilation and installation.

make -j 8

```
make[4]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test/expectremote3'
Making all in expectremote4
make[4]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test/expectremote4'
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test/expectremote4'
make[4]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test'
make[4]: Nothing to be done for 'all-am'.
make[4]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test'
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test'
make[2]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/ncdap_test'
Making all in docs
make[2]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs'
Making all in images
make[3]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs/images'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs/images'
make[3]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs'
make[2]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/docs'
Making all in examples
make[2]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples'
Making all in C
make[3]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples/C'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples/C'
Making all in CDL
make[3]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples/CDL'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples/CDL'
make[3]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples'
make[2]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1/examples'
make[2]: Entering directory '/path/to/netcdf/netcdf-c-4.3.3.1'
make[2]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1'
make[1]: Leaving directory '/path/to/netcdf/netcdf-c-4.3.3.1'
[root@host-10-208-167-174 netcdf-c-4.3.3.1]#
```

make install

 Copyright © Huawei Technologies Co., Ltd.



Step 10 Set the environment variables.

```
export PATH=/path/to/netcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/netcdf/lib:$LD_LIBRARY_PATH
```

----End

1.36.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/netcdf
```

Step 3 Obtain the source package and rename it.

```
wget https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.5.tar.gz
mv v4.4.5.tar.gz netcdf-fortran-4.4.5.tar.gz
```

Step 4 Decompress the source package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

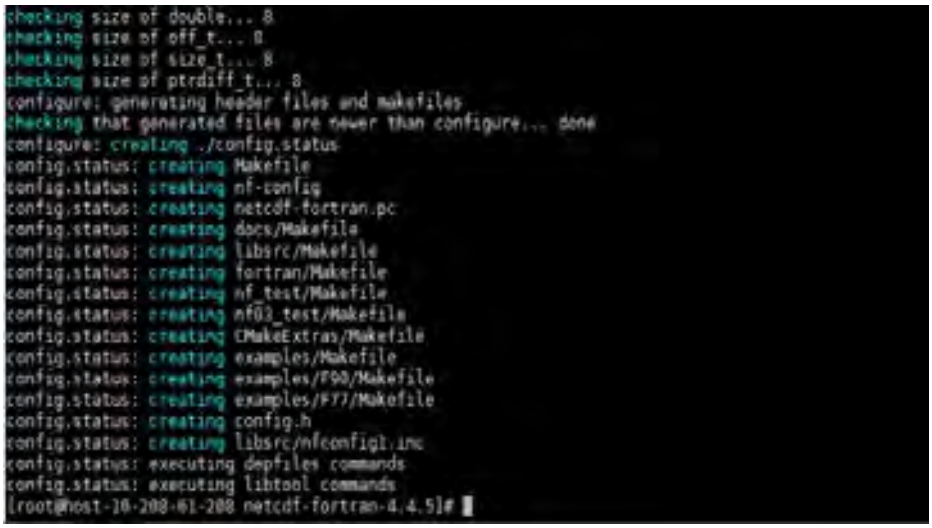
Step 5 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 6 Configure the compilation.

```
./configure --prefix=/path/to/netcdf --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=clang CXX=clang++ FC=flang F77=flang CPPFLAGS="-I/path/to/hdf5/include -I/path/to/netcdf/include " LDFLAGS="-L/path/to/hdf5/lib -L/path/to/
```

```
netcdf/lib -Wl,-rpath=/path/to/hdf5/lib -Wl,- rpath=/path/to/netcdf/lib" CFLAGS="-L/path/to/hdf5/lib -L/path/to/netcdf/lib -I/path/to/hdf5/include -I/path/to/netcdf/include -fPIC" CXXFLAGS="-L/path/to/hdf5/lib -L/path/to/netcdf/lib -I/path/to/hdf5/include -I/path/to/netcdf/include -fPIC" FCFLAGS="-L/path/to/hdf5/lib -L/path/to/netcdf/lib -I/path/to/hdf5/include - I/path/to/netcdf/include -fPIC"
```



Step 7 Apply the patch package.

1. Create a **libtool.patch** file.
   ```
   vi libtool.patch
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   diff -ruN hdf5-1.10.1/libtool hdf5-1.10.1-new/libtool
   --- hdf5-1.10.1/libtool 2019-12-02 20:02:56.270676503 +0000
   +++ hdf5-1.10.1-new/libtool 2019-12-02 20:03:05.062708034 +0000
   @@ -11832,7 +11832,7 @@
   pic_flag=""
   # How to pass a linker flag through the compiler.
   -wl=""
   +wl="-Wl,"
   # Compiler flag to prevent dynamic linking.
   link_static_flag=""
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

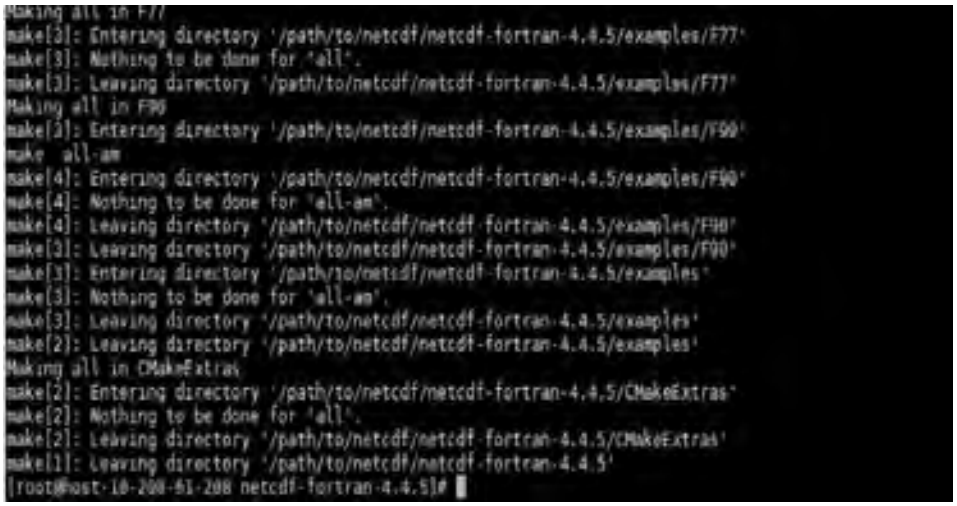
Step 8 Install the patch.

```
patch -p1 < libtool.patch
```

Step 9 Perform the compilation and installation.

```
make -j 8
```

 Copyright © Huawei Technologies Co., Ltd.



make install

Step 10 Set the environment variables.

export PATH=*/path/to/netcdf***/bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/netcdf***/lib:$LD_LIBRARY_PATH**

----End

1.36.4.6 Installing HDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the source package.

wget https://download-ib01.fedoraproject.org/pub/epel/8/Everything/SRPMS/Packages/h/hdf-4.2.14-5.el8.src.rpm

Step 3 Install the source package.

rpm -ivh hdf-4.2.14-5.el8.src.rpm

Step 4 Go to the directory of the executable file.

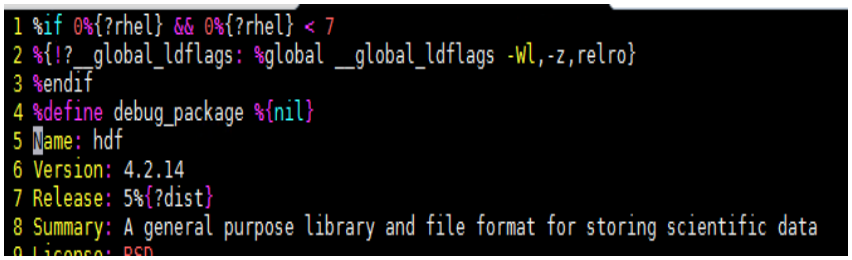
cd /root/rpmbuild/SPECS/

Step 5 Modify the **hdf.spec** file.

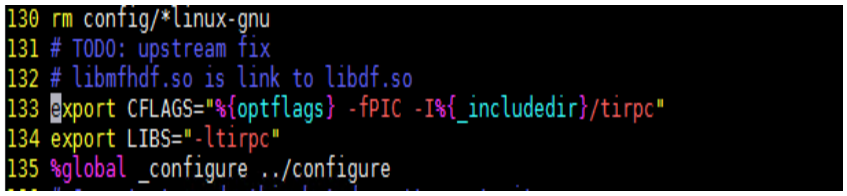
1. Open the **hdf.spec** file.
   vi hdf.spec
2. Press **i** to enter the edit mode.

Add the following content to line 4:

```
%define debug_package %{nil}
```



Add **-fPIC** to line 133.



NOTE

After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Modify the compilation configuration file in the source package in the **SOURCES** directory.

1. Open the compilation configuration file.

```
cd /root/rpmbuild/SOURCES/
tar -xvf hdf-4.2.14.tar.bz2
vim hdf-4.2.14/config/linux-gnu
```

2. Press **i** to enter the edit mode and modify the content in lines 24, 25, 29, and 30.

NOTE

After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Compress the source package again.

```
rm -rf hdf-4.2.14.tar.bz2
tar -zcvf hdf-4.2.14.tar.bz2 hdf-4.2.14
rm -rf hdf-4.2.14
```

Step 8 Go to the **SPECS** directory.

```
cd /root/rpmbuild/SPECS/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 Package the file.

```
rpmbuild -ba hdf.spec
```

NOTE

If an error is reported in this step, the dependency RPM package is missing when building the HDF package. If the required RPM package has been installed, this error does not occur and please ignore **Step 9** and **Step 10**. If the required RPM package has not been installed, install it by performing **Step 9** and **Step 10**. The required dependency packages may vary depending on the environment. Perform **Step 9** based on the site requirements.

Step 10 Install the dependencies using the **yum** command.

```
yum install hamcrest javapackages-tools junit libaec-devel libjpeg-devel libtirpc-devel slf4j
```

Step 11 Install the RPM package of the specified Java version.

NOTE

Install the Java package of the version specified in **Step 10**. If the version is too late, a compatibility problem may occur.

```
yum install java-1.8.0-openjdk-devel
```

Step 12 Compile and package the file.

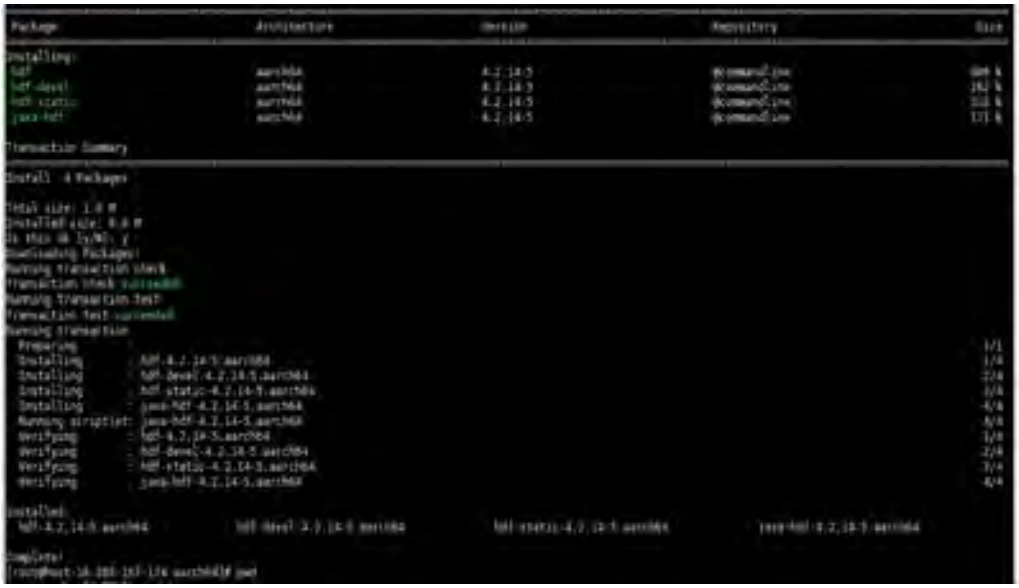
```
rpmbuild -ba hdf.spec
```

Step 13 Go to the installation package directory.

```
cd /root/rpmbuild/RPMS/aarch64
```

Step 14 Install HDF.

```
yum install hdf-4.2.14-5.aarch64.rpm hdf-devel-4.2.14-5.aarch64.rpm hdf-static-4.2.14-5.aarch64.rpm
java-hdf-4.2.14-5.aarch64.rpm
```



----End

1.36.4.7 Installing GDAL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

```
mkdir -p /path/to/gdal
```

Step 3 Go to the installation directory.

```
cd /path/to/gdal
```

Step 4 Obtain the source code.

```
wget http://download.osgeo.org/gdal/2.2.4/gdal-2.2.4.tar.gz
```

Step 5 Decompress the source package.

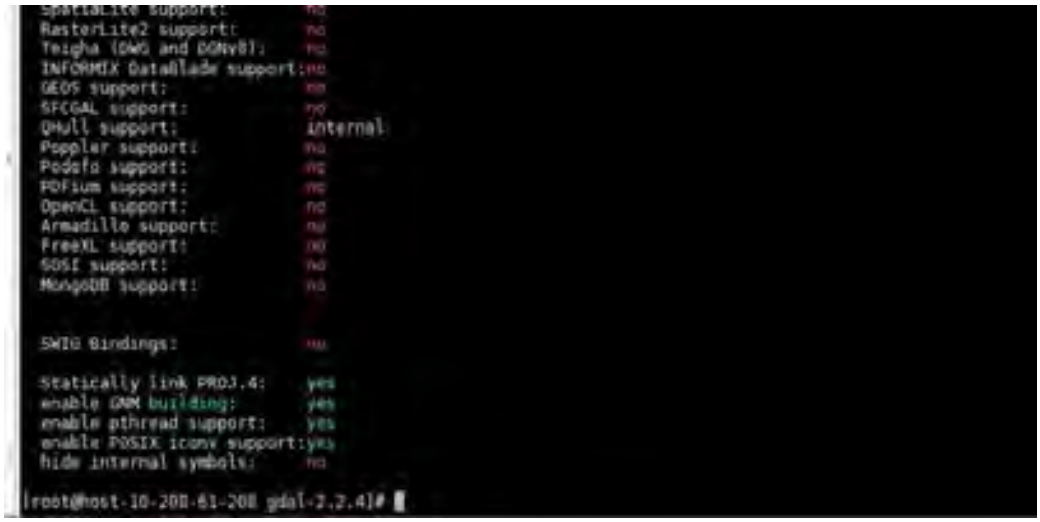
```
tar -zxvf gdal-2.2.4.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd gdal-2.2.4
```

Step 7 Configure the compilation.

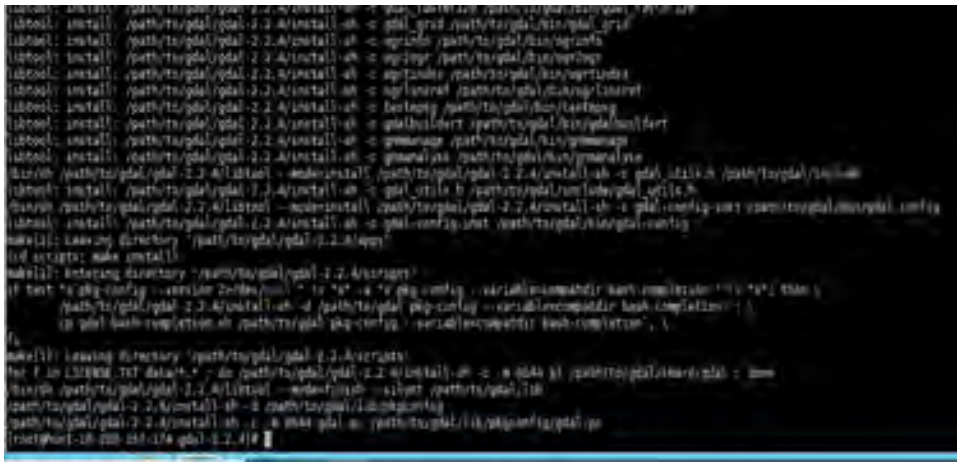
```
./configure --with-static-proj4=/usr --prefix=/path/to/gdal --without-pam --with-png=/usr --with-gif=internal --with-libtiff=internal --with-geotiff=internal --with-jpeg=/usr --with-libz=/usr --with-sqlite3=no --with-expat=no --with-curl=no --without-ld-shared --with-hdf4=no --with-hdf5=no --with-pg=no --without-grib --disable-shared --with-freexl=no --with-geos=no --with-openjpeg=no --with-mysql=no --with-ecw=no --with-fgdb=no --with-odbc=no --with-xml2=no --with-ogdi=no --with-pcraster=no --with-xerces=no
```



Step 8 Perform the compilation and installation.

```
make -j 8
```

```
make install
```



Step 9 Set the environment variables.

```
export PATH=/path/to/gdal/bin:$PATH
export LD_LIBRARY_PATH=/path/to/gdal/lib:$LD_LIBRARY_PATH
```

----End

1.36.4.8 Installing Jasper

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

```
mkdir -p /path/to/jasper
```

Step 3 Go to the installation directory.

```
cd /path/to/jasper
```

Step 4 Obtain the source package.

```
wget https://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.2.tar.gz
```

Step 5 Decompress the source package.

```
tar -zxvf jasper-1.900.2.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd jasper-1.900.2/
```

Step 7 Configure the compilation.

```
./configure --prefix=/path/to/jasper
```

Step 8 Perform the compilation and installation.

```
make -j 8
```

make install

Step 9 Set the environment variables.

export PATH=*/path/to/jasper***/bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/jasper***/lib:$LD_LIBRARY_PATH**

----End

1.36.4.9 Installing G2Clib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

mkdir -p */path/to/g2clib*

Step 3 Go to the installation directory.

cd /path/to/g2clib

Step 4 Obtain the source package.

wget http://www.ncl.ucar.edu/Download/files/g2clib-1.6.0-patch.tar.gz

Step 5 Decompress the source package.

tar -zxvf g2clib-1.6.0-patch.tar.gz

Step 6 Go to the directory generated after the decompression.

 Copyright © Huawei Technologies Co., Ltd.

```
cd g2clib-1.6.0-patch/
```

Step 7 Configure the compilation.

1. Open the compilation file.
   ```
   vi /path/to/g2clib/g2clib-1.6.0-patch/makefile
   ```
2. Press **i** to enter the edit mode and modify lines 22 and 33 of the file.

```
19 #
20
21 #INC=-I/nwprod/lib/include/
22 INC=-I/path/to/jasper/include -I/usr/include
23
24
25 #
26 #   This "C" source code contains many uses of the C++
27 #   comment style "//".  Please make sure you include the
28 #   appropriate compiler flag to allow the use of "//" comment indicators.
29 #
30 CFLAGS= $(INC) $(DEFS) -fPIC
31 #CFLAGS=  $(INC) $(DEFS)
32
33 CC=clang
34
35 LIB=libgrib2c.a
36 ARFLAGS=
```

NOTE

After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation and installation.

```
make all
mv libgrib2c.a /usr/local/lib
cp grib2.h /usr/local/include
```

----End

1.36.4.10 Installing Udunits2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

```
mkdir -p /path/to/udunits2
```

Step 3 Go to the installation directory.

```
cd /path/to/udunits2
```

Step 4 Obtain the source package.

```
wget https://artifacts.unidata.ucar.edu/repository/downloads-udunits/udunits-2.2.28.tar.gz
```

Step 5 Decompress the source package.

```
tar -zxvf udunits-2.2.28.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd udunits-2.2.28/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Set the environment variables.

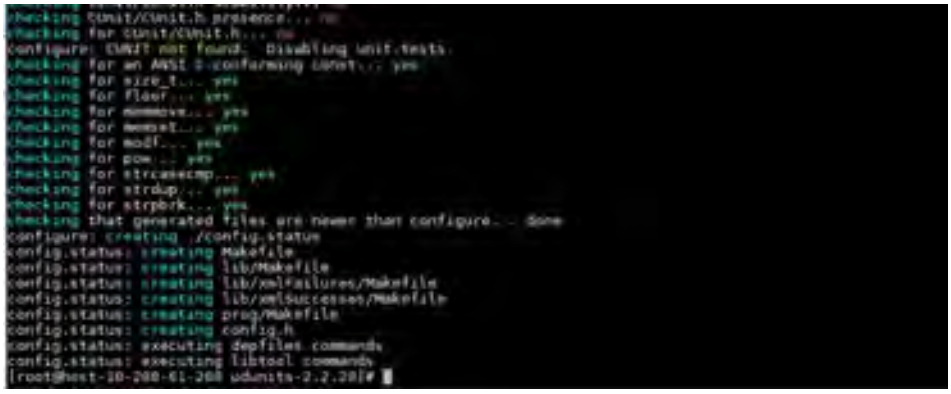
```
export PERL=""
```

Step 8 Install the dependencies.

```
yum install expat-devel
```

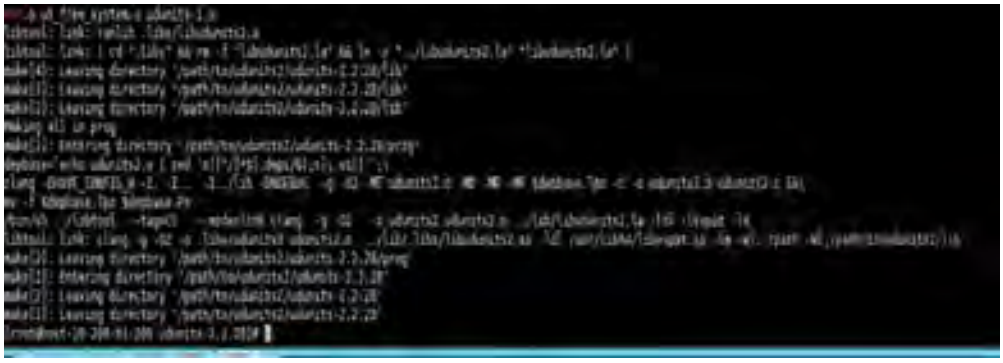
Step 9 Configure the compilation.

```
./configure --prefix=/path/to/udunits2
```

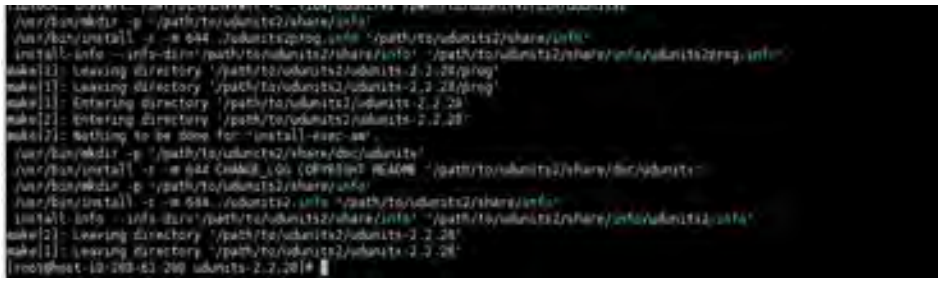


Step 10 Perform the compilation and installation.

```
make -j 8
```



```
make install
```



Step 11 Set the environment variables.

```
export PATH=/path/to/udunits2/bin:$PATH
export LD_LIBRARY_PATH=/path/to/udunits2/lib:$LD_LIBRARY_PATH
```

----End

1.36.5 Obtaining the Source Code

Procedure

Step 1 Download the NCL installation package.

https://download-ib01.fedoraproject.org/pub/epel/7/SRPMS/Packages/n/ncl-6.3.0-6.el7.src.rpm

Step 2 Use an SFTP tool to upload the NCL installation package to the **/root** directory on the server.

----End

1.36.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install flex-devel
```

Step 3 Obtain the source package.

```
wget https://download-ib01.fedoraproject.org/pub/epel/7/SRPMS/Packages/n/ncl-6.3.0-6.el7.src.rpm
```

Step 4 Install the source package.

```
rpm -ivh ncl-6.3.0-6.el7.src.rpm
```

Step 5 Go to the **SOURCES** directory, modify the NCL source package, and repackage the file.

```
cd ~/rpmbuild/SOURCES
tar -zxvf ncl_ncarg-6.3.0.tar.gz
```

Step 6 Edit the **ncl_ncarg-6.3.0/common/src/libncarg_c/yMakefile** file.

1. Open the **ncl_ncarg-6.3.0/common/src/libncarg_c/yMakefile** file.
   ```
   vi ncl_ncarg-6.3.0/common/src/libncarg_c/yMakefile
   ```
2. Press **i** to enter the insert mode and modify the content in line 35.
   ```
   #elif (defined(ia64) || defined(x86_64) || defined(amd64) || defined(aarch64))
   ```

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Edit the **ncl_ncarg-6.3.0/ni/src/lib/nio/yMakefile** file.

1. Open the **ncl_ncarg-6.3.0/ni/src/lib/nio/yMakefile** file.
   ```
   vi ncl_ncarg-6.3.0/ni/src/lib/nio/yMakefile
   ```
2. Press **i** to enter the insert mode and modify the content in line 85.
   ```
   #if defined(IRIX64) || defined(x86_64) || defined(__LP64__) || defined(aarch64)
   ```

   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Edit the **ncl_ncarg-6.3.0/ni/src/ncl/yMakefile** file.

1. Open the **ncl_ncarg-6.3.0/ni/src/ncl/yMakefile** file.
   ```
   vi ncl_ncarg-6.3.0/ni/src/ncl/yMakefile
   ```
2. Press **i** to enter the insert mode.

   Modify line 46:
   ```
   EXTRA_CCOPTIONS = -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -fsigned-char
   ```
   Modify line 113:
   ```
   #if defined(IRIX64) || defined(x86_64) || defined(ia64) || defined(__LP64__) || defined(aarch64)
   ```
   Modify lines 168 and 170.
   ```
   167 #elif defined(linux)
   168 CC_LD             = clang\+\+
   169 #else
   170 CC_LD             = clang\+\+
   171 #endif
   ```

   NOTE

   After performing **Step 8.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Edit the **ncl_ncarg-6.3.0/ncarview/src/lib/libncarg_ras/misc.c** file.

1. Open the **ncl_ncarg-6.3.0/ncarview/src/lib/libncarg_ras/misc.c** file.
   ```
   vi ncl_ncarg-6.3.0/ncarview/src/lib/libncarg_ras/misc.c
   ```
2. Press **i** to enter the insert mode.

   Add the following function declaration statements to lines 41 and 42, that is, add the statements before the **int** keyword of the first function.
   ```
   void _swapshort (register char *bp, register unsigned n);
   void _swaplong (register char *bp, register unsigned n);
   ```

   NOTE

   After performing **Step 9.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Edit the **ncl_ncarg-6.3.0/config/LINUX** file.

1. Open the **ncl_ncarg-6.3.0/config/LINUX** file.
   ```
   vi ncl_ncarg-6.3.0/config/LINUX
   ```
2. Press **i** to enter the insert mode and replace **clang** with **gcc** for compilation. The modification result is as follows:
   ```
   21 #define CCompiler   clang
   22 #define FCompiler   flang
   23 #define CtoFLibraries   -lflang -lm
   24 #define CtoFLibrariesUser  -lflang -lm
   ```

   NOTE

   After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Compress the NCL source package again.

```
cd /root/rpmbuild/SOURCES/
rm -rf ncl_ncarg-6.3.0.tar.gz
tar -zcvf ncl_ncarg-6.3.0.tar.gz ncl_ncarg-6.3.0
rm -rf ncl_ncarg-6.3.0
```

Step 12 Go to the **SOURCES** directory and modify the **Site.local.ncl** file.

1. Open the **Site.local.ncl** file.
   ```
   vi Site.local.ncl
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   #define YmakeRoot $(DESTDIR)@prefix@
   #define ManRoot $(DESTDIR)@mandir@
   #define LibRoot $(DESTDIR)@libdir@/ncarg
   #define SharePath $(DESTDIR)@datadir@
   #define BuildWithF90 TRUE
   #define IncSearch -I/usr/include/hdf -I/path/to/netcdf/include/ -I/path/to/udunits2/include/ -I/usr/include/freetype2 -I/path/to/gdal/include/ -I/path/to/hdf5/include/
   #define LibSearch -L@libdir@/hdf -L/path/to/jasper/lib -ljasper  -L/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib -lflang -lpgmath -lflangrti
   #define BuildNetCDF4 1
   #define NetCDF4lib -L/path/to/netcdf/lib -lnetcdf
   #define BuildCAIRO 1
   #define CAIROlib -lcairo -lfreetype
   #define BuildGDAL 1
   #define GDALlib -L/path/to/gdal/lib -lgdal -lproj
   #define BuildHDFEOS 0
   #define BuildHDFEOS5 0
   #define BuildTRIANGLE 0
   #define HDFlib -lmfhdf -ldf -ljpeg -lz
   #define HDF5lib -L/path/to/hdf5/lib -lhdf5_hl -lhdf5
   #define BuildUdunits 1
   #define UdUnitslib -L/path/to/udunits2/lib -ludunits2
   ```

   NOTE

   If the actual paths for installing NetCDF, Udunits2, GDAL, HDF5, Jasper, and BiSheng Compiler from their source code are different from the paths in the preceding file, change the installation paths in the preceding file to the actual ones.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

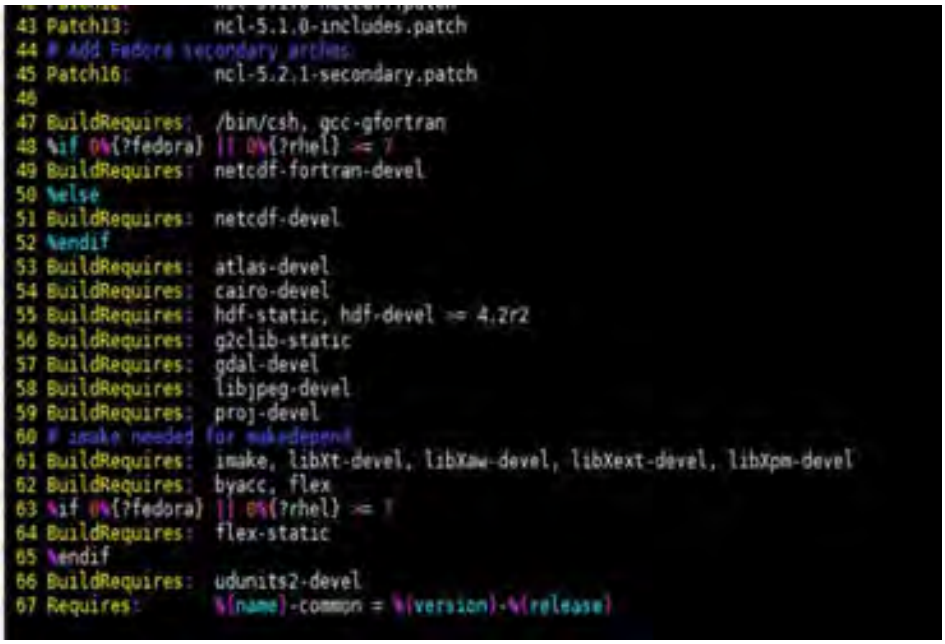
Step 13 Go to the **SPECS** directory.

```
cd /root/rpmbuild/SPECS
```

Step 14 Modify the **ncl.spec** file.

1. Open the **ncl.spec** file.
   ```
   vi ncl.spec
   ```
2. Press **i** to enter the insert mode.
   a. Add the following content at the beginning of the file and move the original first line to the next line.
      ```
      %define debug_package %{nil}
      ```
   b. Delete the **BuildRequires** command line corresponding to the dependency package for the source code installation.

      Before the modification:

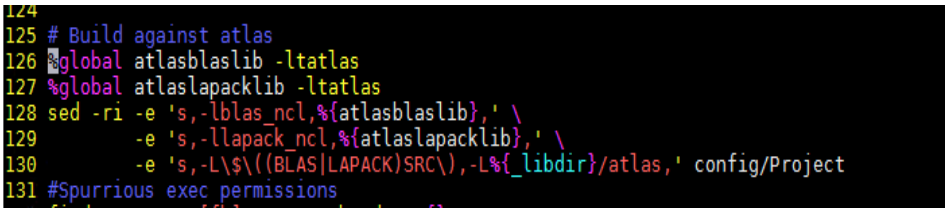


Delete lines 48, 49, 50, 51, 52, 56, 57, and 66.

c. Modify the definitions of the macros related to Atlas.

Before the modification:

Delete lines 126, 129, 130, 131, and 132.



d. Change the values of variables **F77**, **F77_LD**, and **CTOFLIBS**, as shown in the following figure. Modify the installation path of the BiSheng Compiler as required.

 Copyright © Huawei Technologies Co., Ltd.

```
169 #make Build CCOPTIONS="$RPM_OPT_FLAGS -fPIC -Werror-implicit-function-declaration" F77=gfortran F77_LD=gfortran\
170
171 make Build CCOPTIONS="$RPM_OPT_FLAGS -fPIC -fno-strict-aliasing -fopenmp" F77=flang F77_LD=flang\
172  CTOFLIBS="-L/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib -lflang" FCOPTIONS="$RPM_OPT_FLAGS -fPIC -fno-second-underscore -fno-range-check -fo
   penmp" \
173  COPT= FOPT=
174
175
176 %install
```

e. Add the following parameters so that the default dependency relationship is ignored during the package installation.

Modify line 60 and comment out this line.

```
#Requires:      udunits2
```

Add a new line of **AutoReqProv: no** above each of the four lines: **%description**, **%description common**, **%description devel**, and **%description examples**.

```
AutoReqProv: no
```

NOTE

After performing **Step 14.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 15 Compile and package the file.

```
rpmbuild -ba ncl.spec
```

NOTE

If an error is reported in this step, the dependency RPM package is missing when building the NCL package. If the required RPM package has been installed, this error does not occur and please ignore **Step 12**. If the RPM package has not been installed, install it by performing **Step 12**.

```
[root@host-10-208-167-174 SPECS]# rpmbuild -ba ncl.spec
error: Failed build dependencies:
        /bin/csh is needed by ncl-6.3.0-6.aarch64
        atlas-devel is needed by ncl-6.3.0-6.aarch64
        cairo-devel is needed by ncl-6.3.0-6.aarch64
        imake is needed by ncl-6.3.0-6.aarch64
        libXaw-devel is needed by ncl-6.3.0-6.aarch64
        libXext-devel is needed by ncl-6.3.0-6.aarch64
        libXpm-devel is needed by ncl-6.3.0-6.aarch64
        libXt-devel is needed by ncl-6.3.0-6.aarch64
        proj-devel is needed by ncl-6.3.0-6.aarch64
[root@host-10-208-167-174 SPECS]#
```

Step 16 Install the dependencies.

```
yum install /bin/csh atlas-devel cairo-devel imake libXaw-devel libXext-devel libXpm-devel libXt-devel proj-devel
```

Step 17 Package the file.

```
rpmbuild -ba ncl.spec
```

Step 18 Go to the **RPMS** directory and compile and install NCL.

```
cd /root/rpmbuild/RPMS/aarch64
yum install ncl*
```

Step 19 Set the environment variables.

```
export NCARG_FONTCAPS=/usr/lib64/ncarg/fontcaps
export NCARG_GRAPHCAPS=/usr/lib64/ncarg/graphcaps
export NCARG_ROOT=/usr
export NCARG_DATABASE=/usr/lib64/ncarg/database
export NCARG_LIB=/usr/lib64/ncarg
export NCARG_NCARG=/usr/share/ncarg
export EXTRA_CCOPTIONS=" -fsigned-char "$EXTRA_CCOPTIONS
```

----End

1.36.7 Running and Verification

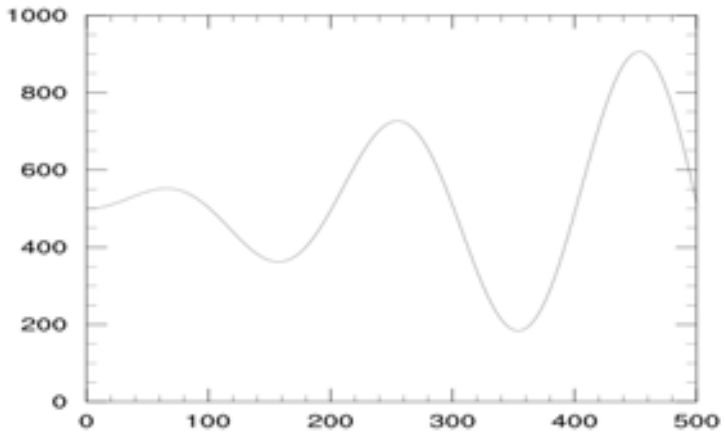
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Execute the test.

```
ng4ex xy01n -W png
```

If the **xy01n.png** image is generated in the current directory, the execution is successful, as shown in the following figure:



----End

1.36.8 Troubleshooting

 Copyright © Huawei Technologies Co., Ltd.

1.36.8.1 Error Reported When Packaging rpmbuild -ba ncl.spec

Symptom

If the following error information is displayed when packaging **rpmbuild -ba ncl.spec**:

```
/path/to/gdal/gdal-2.2.4/frmts/fits/fitsdataset.cpp:541: undefined reference to `ffinit'
/usr/bin/ld: /path/to/gdal/gdal-2.2.4/frmts/fits/fitsdataset.cpp:564: undefined reference to `ffclos'
/usr/bin/ld: /path/to/gdal/gdal-2.2.4/frmts/fits/fitsdataset.cpp:571: undefined reference to `ffcrim'
/usr/bin/ld: /path/to/gdal/gdal-2.2.4/frmts/fits/fitsdataset.cpp:578: undefined reference to `ffclos'
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)
```

Key Process and Cause Analysis

The compilation fails for multiple times.

Conclusion and Solution

Step 1 Go to the **/root/rpmbuild/BUILD/ncl_ncarg-6.3.0/ni/src/ncl** directory.

```
cd /root/rpmbuild/BUILD/ncl_ncarg-6.3.0/ni/src/ncl
```

Step 2 Run the following command:

```
make clean
```

Step 3 Go to the **/root/rpmbuild/SPECS** directory.

```
cd /root/rpmbuild/SPECS
```

Step 4 Package the file.

```
rpmbuild -ba ncl.spec
```

----End

1.37 PyFerret 7.6.0 Porting Guide (openEuler 21.03)

1.37.1 Introduction

Ferret is an interactive computer visualization and analysis environment designed to meet the needs of oceanographers and meteorologists analyzing large and complex gridded data sets.

For more information about Ferret, visit the **Ferret official website**.

Programming languages: C/Fortran

Brief description: ecosystem dynamics modeling software

Open source license: GPL

Recommended Software Version

PyFerret 7.6.0

 Copyright © Huawei Technologies Co., Ltd.

1.37.2 Environment Requirements

Hardware Requirements

Table 1-190 lists the hardware requirements.

Table 1-190 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-191 lists the software requirements.

Table 1-191 Software requirements

Item	Version	Download URL
GNU	9.3.0	https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/
HDF5	1.10.5	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz
NetCDF-C	4.7.0	https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.0.tar.gz
NetCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.5.tar.gz
PyFerret	7.6.0	https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0

OS Requirements

Table 1-192 lists the OS requirements.

Table 1-192 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	5.10	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

1.37.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PyFerret software porting.

Table 1-193 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	-	Installation path of GNU.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PyFerret	Installation path of PyFerret.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/HDF5	Installation path of HDF5.	
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	

1.37.4 Configuring the Compilation Environment

1.37.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-194 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Configuring the local yum source	See **1.37.4.2 Configuring the Local Yum Source**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing HDF5	See **1.37.4.3 Installing HDF5**.
4	Installing NetCDF-C	See **1.37.4.4 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.37.4.5 Installing NetCDF-Fortran**.

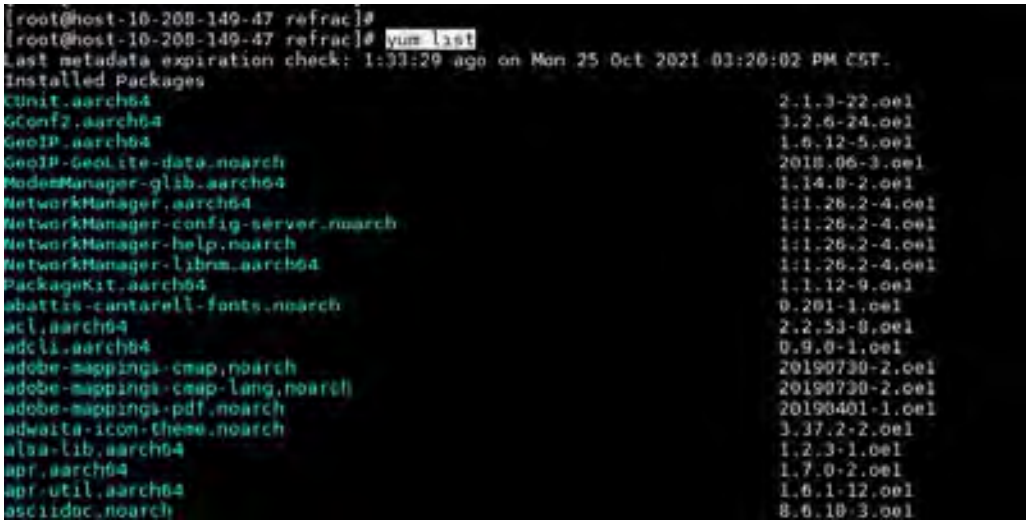
1.37.4.2 Configuring the Local Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether Yum is configured.

```
yum list
```



NOTE

The Yum source is configured by default on openEuler 21.03 (LTS SP1) and does not need to be configured additionally.

Step 3 Install the base dependencies.

```
yum install -y m4 wget openssl-devel
yum install -y python2 python2-devel python2-numpy libgfortran python2-qt4 python2-scipy
pip2 install pyshp
```

NOTE

Ensure that the Python version is the actual one.

----End

1.37.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an HDF5 installation directory, for example, */path/to/HDF5*.

```
mkdir -p /path/to/HDF5
```

Step 3 Go to the HDF5 installation directory.

```
cd /path/to/HDF5
```

Step 4 Obtain the HDF5 installation package.

```
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
```

Step 5 Decompress the HDF5 installation package.

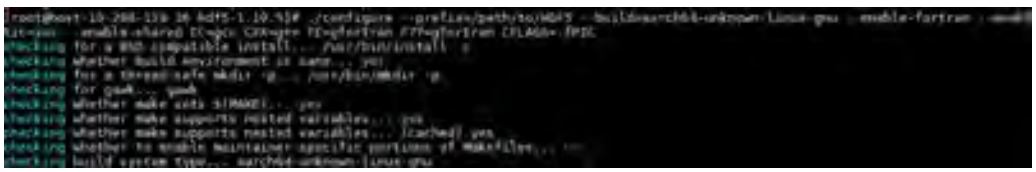
```
tar -zxvf hdf5-1.10.5.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd hdf5-1.10.5
```

Step 7 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-shared CC=gcc CXX=g++ FC=gfortran F77=gfortran CFLAGS=-fPIC
```



Step 8 Perform the compilation and installation.

```
make -j
```

```
[root@host-10-208-119-36 hdf5-1.10.5]# make -j 8
Making all in src
make[1]: Entering directory '/path/to/HDF5/hdf5-1.10.5/src'
make  all-am
make[2]: Entering directory '/path/to/HDF5/hdf5-1.10.5/src'
  CC       H5.lo
  CC       H5checksum.lo
  CC       H5dbg.lo
  CC       H5system.lo
  CC       H5timer.lo
  CC       H5trace.lo
  CC       H5A.lo
  CC       H5Abtree2.lo
```

```
make install
```

```
[root@host-10-208-119-36 hdf5-1.10.5]# make install
Making install in src
make[1]: Entering directory '/path/to/HDF5/hdf5-1.10.5/src'
make[2]: Entering directory '/path/to/HDF5/hdf5-1.10.5/src'
 /usr/bin/mkdir -p '/path/to/HDF5/lib'
 /bin/sh ../libtool   --mode=install /usr/bin/install -c   libhdf5.la '/path/to/HDF5
libtool: install: /usr/bin/install -c .libs/libhdf5.so.103.1.0 /path/to/HDF5/lib/libh
libtool: install: (cd /path/to/HDF5/lib && { ln -s -f libhdf5.so.103.1.0 libhdf5.so.
.1.0 libhdf5.so.103; }; })
libtool: install: (cd /path/to/HDF5/lib && { ln -s -f libhdf5.so.103.1.0 libhdf5.so
hdf5.so; }; })
libtool: install: /usr/bin/install -c .libs/libhdf5.lai /path/to/HDF5/lib/libhdf5.la
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 Verify the installation.

```
ls /path/to/HDF5/lib
ls /path/to/HDF5/bin
```

If the following information is displayed, the installation is successful:

Step 10 Load the environment variables.

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.37.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a NetCDF-C installation directory, for example, */path/to/NETCDF*.

```
mkdir -p /path/to/NETCDF
```

Step 3 Install the dependency.

```
yum install curl* -y
```

Step 4 Go to the NetCDF-C installation directory.

```
cd /path/to/NETCDF
```

Step 5 Obtain the NetCDF-C installation package.

```
wget https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.3 --no-check-certificate
```

Step 6 Rename and decompress the NetCDF-C installation package.

```
mv v4.7.3 netcdf-c-4.7.3.tar.gz
tar -zxvf netcdf-c-4.7.3.tar.gz
```

Step 7 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.3
```

Step 8 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-largefile CC=gcc CXX=g++ FC=gfortran F77=gfortran CPPFLAGS="-I/path/to/HDF5/include"  LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-  L/path/to/HDF5/lib -I/path/to/HDF5/include"
```

Step 9 Perform the compilation and installation.

```
make -j
```

```
[root@host-10-208-119-36 netcdf-c-4.7.3]# make -j 8
make  all-recursive
make[1]: Entering directory '/path/to/NETCDF/netcdf-c-4.7.3'
Making all in include
make[2]: Entering directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
Making all in h5_test
make[2]: Entering directory '/path/to/NETCDF/netcdf-c-4.7.3/h5_test'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-c-4.7.3/h5_test'
```

make install

```
[root@host-10-208-119-36 netcdf-c-4.7.3]# make install
Making install in include
make[1]: Entering directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
make[2]: Entering directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
make[2]: Nothing to be done for 'install-exec-am'.
 /usr/bin/mkdir -p '/path/to/NETCDF/include'
 /usr/bin/install -c -m 644 netcdf.h netcdf_meta.h netcdf_mem.h netcd
make[2]: Leaving directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-c-4.7.3/include'
```

Step 10 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**

If the following information is displayed, the installation is successful:

```
[root@host-10-208-119-36 netcdf-c-4.7.3]# ls /path/to/NETCDF/bin
nc-config  nccopy  ncdump  ncgen  ncgen3
[root@host-10-208-119-36 netcdf-c-4.7.3]# ls /path/to/NETCDF/lib
libh5bzip2.la  libnetcdf.a   libnetcdf.settings  libnetcdf.so.15      pkgconfig
libh5bzip2.so  libnetcdf.la  libnetcdf.so        libnetcdf.so.15.2.1
[root@host-10-208-119-36 netcdf-c-4.7.3]#
```

Step 11 Load the environment variables.

export PATH=*/path/to/NETCDF*/**bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/NETCDF*/**lib:$LD_LIBRARY_PATH**

----End

1.37.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the NetCDF-C installation directory you created in **1.37.4.5 Installing NetCDF-Fortran**.

cd */path/to/NETCDF*

Step 3 Obtain the NetCDF-Fortran installation package.

wget https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.5.2 --no-check-certificate

Step 4 Rename and decompress the NetCDF-Fortran installation package.

mv v4.5.2 netcdf-fortran-4.5.2.tar.gz
tar -zxvf netcdf-fortran-4.5.2.tar.gz

Step 5 Go to the directory generated after the decompression.

cd netcdf-fortran-4.5.2

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=gcc CXX=g++ FC=gfortran F77=gfortran CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

```
checking size of off_t... 8
checking size of size_t... 8
checking size of ptrdiff_t... 8
configure: generating header files and makefiles
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating nf-config
config.status: creating netcdf-fortran.pc
config.status: creating docs/Makefile
config.status: creating fortran/Makefile
config.status: creating nf_test/Makefile
config.status: creating nf_test4/Makefile
config.status: creating nf03_test/Makefile
config.status: creating nf03_test4/Makefile
config.status: creating CMakeExtras/Makefile
config.status: creating libsrc/Makefile
config.status: creating examples/Makefile
config.status: creating examples/F90/Makefile
config.status: creating examples/F77/Makefile
config.status: linking nf_test/ref_fills.nc to nf_test/fills.nc
config.status: executing depfiles commands
config.status: executing libtool commands
[root@host-10-208-119-36 netcdf-fortran-4.5.2]#
```

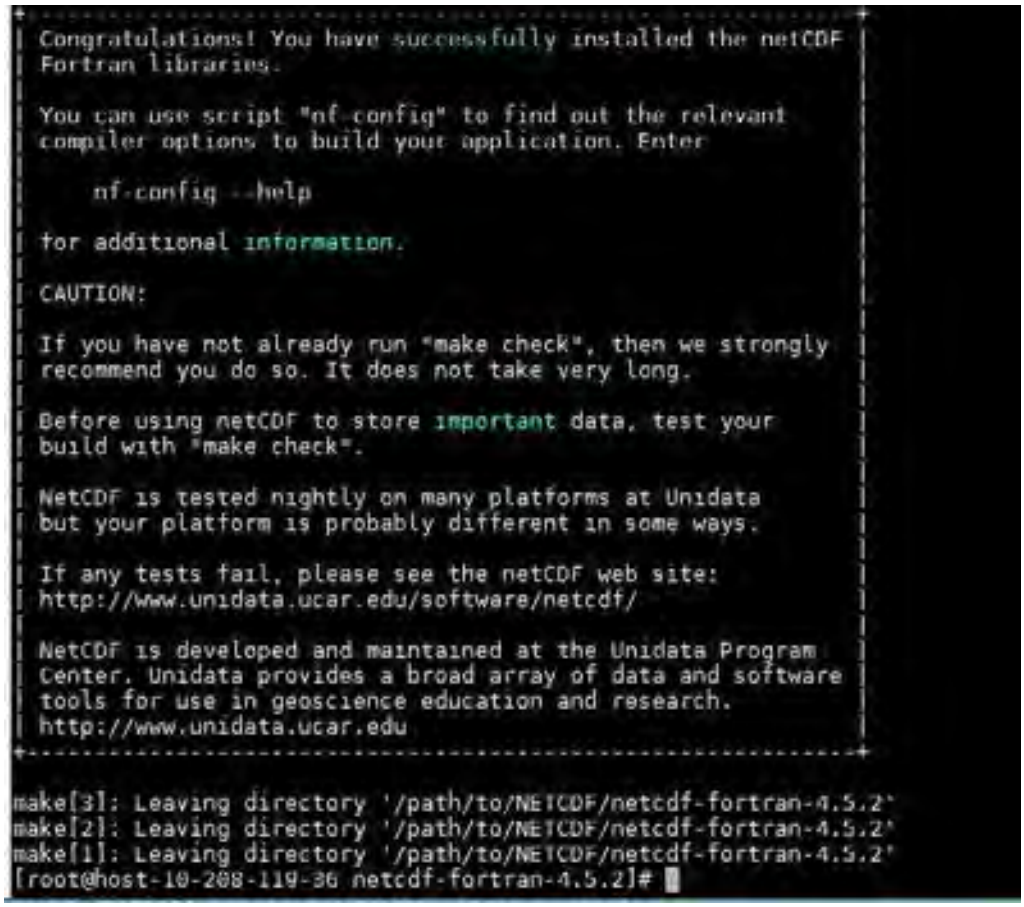
Step 7 Perform the compilation and installation.

```
make -j
```

```
make[3]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples/F90'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples/F90'
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples/F90'
make[2]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples'
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2/examples'
Making all in CMakeExtras
make[1]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.5.2/CMakeExtras'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2/CMakeExtras'
make[1]: Entering directory '/path/to/NETCDF/netcdf-fortran-4.5.2'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/path/to/NETCDF/netcdf-fortran-4.5.2'
[root@host-10-208-119-36 netcdf-fortran-4.5.2]#
```

```
make install
```

 Copyright © Huawei Technologies Co., Ltd.



Step 8 Verify the installation.

ls */path/to/NETCDF*/**bin**
ls */path/to/NETCDF*/**lib**

If the following information is displayed, the installation is successful:

----End

1.37.5 Obtaining the Source Code

Procedure

Step 1 Create the PyFerret installation directory */path/to/PyFerret*.

mkdir -p */path/to/PyFerret*

Step 2 Go to the PyFerret installation directory.

cd */path/to/PyFerret*

Step 3 Obtain the PyFerret installation package **PyFerret-7.6.0.tar.gz** and rename it.

wget https://codeload.github.com/NOAA-PMEL/PyFerret/tar.gz/refs/tags/v7.6.0 --no-check-certificate
mv v7.6.0 PyFerret-7.6.0.tar.gz

----End

 Copyright © Huawei Technologies Co., Ltd.

1.37.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the base dependencies.

```
yum -y install cairo*
yum -y install pango-devel pangomm pango-tests pangomm-devel pangomm-doc
```

Step 3 Decompress the installation package and go to the installation directory.

```
tar -zxvf PyFerret-7.6.0.tar.gz
cd PyFerret-7.6.0
```

Step 4 Copy the configuration file.

```
cp site_specific.mk.in site_specific.mk
```

Step 5 Modify the **site_specific.mk** configuration file.

1. Open the **site_specific.mk** file.
   ```
   vi site_specific.mk
   ```
2. Press **i** to enter the edit mode and add a new line above line 10:
   ```
   HOME=/path/to/PyFerret
   ```
   Before the modification:
   ```
   DIR_PREFIX = $(HOME)/git/PyFerret
   INSTALL_FER_DIR = $(HOME)/PyFerret
   BUILDTYPE = x86_64-linux
   PYTHON_EXE = python3.6
   HDF5_LIBDIR = /usr/local/hdf5/lib
   NETCDF_LIBDIR = /usr/local/netcdf/lib
   ```
   After the modification:
   ```
   HOME=/path/to/PyFerret
   DIR_PREFIX = $(HOME)/PyFerret-7.6.0
   INSTALL_FER_DIR = $(HOME)/pyferret
   BUILDTYPE = aarch64-linux
   PYTHON_EXE = python3.8
   HDF5_LIBDIR = /path/to/HDF5/lib
   NETCDF_LIBDIR = /path/to/NETCDF/lib
   ```

   NOTE

   After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Copy the configuration file.

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 7 Modify **platform_specific.mk.aarch64-linux**.

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 8 Go to the **ef_utility** directory.

```
cd external_functions/ef_utility/
```

Step 9 Copy the configuration file.

```
cp site_specific.mk.in site_specific.mk
```

Step 10 Modify the configuration file.

1. Open the configuration file.

 Copyright © Huawei Technologies Co., Ltd.

```
vi site_specific.mk
```

2. Press **i** to enter the edit mode and modify the following information in bold:
   BUILDTYPE = **aarch64-linux**
   PYTHON_EXE = **python3.8**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Copy the configuration file.

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 12 Modify the **platform_specific.mk.aarch64-linux** configuration file.

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 13 Return to the original directory.

```
cd -
```

Step 14 Perform the installation.

```
make
```

```
./binary   in_fnt24a.sym  fnt24.sym
./binary   in_fnt25a.sym  fnt25.sym
./binary   in_fnt26a.sym  fnt26.sym
./binary   in_fnt27a.sym  fnt27.sym
./binary   in_fnt28a.sym  fnt28.sym
./binary   in_fnt29a.sym  fnt29.sym
./binary   in_fnt30a.sym  fnt30.sym
./binary   in_fnt31a.chr  fnt31.chr
./binary   in_fnt32a.chr  fnt32.chr
./binary   in_fnt33a.sym  fnt33.sym
make[1]: Leaving directory '/path/to/PyFerret/PyFerret-7.6.0/bin/build_fonts/unix'
[root@host-10-208-119-36 PyFerret-7.6.0]#
```

```
make install
```

```
Create gzipped tar file /path/to/PyFerret/PyFerret-7.6.0/pyferret-latest-local.tar.gz (y/n)? y

-rw-r--r--. 1 root root 6590091 Dec 22 17:00 /path/to/PyFerret/PyFerret-7.6.0/pyferret-latest-lo

Cleaning up - removing /tmp/pyferret_110216

mkdir -p /path/to/PyFerret/pyferret
mv -f pyferret-latest-local.tar.gz /path/to/PyFerret/pyferret
( cd /path/to/PyFerret/pyferret ; tar xz --strip-components=1 -f pyferret-latest-local.tar.gz )
[root@host-10-208-119-36 PyFerret-7.6.0]#
```

----End

1.37.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the PyFerret installation directory.

```
cd /path/to/PyFerret
```

Step 3 Obtain the PyFerret installation package and rename it.

```
wget https://codeload.github.com/NOAA-PMEL/FerretDatasets/tar.gz/v7.6 --no-check-certificate
mv v7.6 FerretDatasets-7.6.tar.gz
```

Step 4 Decompress **FerretDatasets-7.6.tar.gz**.

```
tar -zxvf FerretDatasets-7.6.tar.gz
```

Step 5 Go to the PyFerret installation directory.

```
cd /path/to/PyFerret/pyferret/bin
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@host-10-208-99-93 PyFerret]# cd pyferret/bin
[root@host-10-208-99-93 bin]# ll
total 104K
drwxrwxr-x. 3 root root 4.0K Jun 25  2020 build_fonts
-rwxrwxr-x. 1 root root  587 Jun 25  2020 Fdata
lrwxrwxrwx. 1 root root    6 Nov  8 17:34 Fdesc -> Fdescr
-rwxrwxr-x. 1 root root  601 Jun 25  2020 Fdescr
-rwxrwxr-x. 1 root root  109 Jun 25  2020 Fenv
```

Step 6 Run the **Finstall** script to generate the corresponding file.

```
./Finstall
```

During the process, you need to type the PyFerret installation path, FerretDatasets package path, and **ferret_paths.sh** script generation path. This process is an interactive process, and you need to type the needed information at the following prompts:

- Proceed? **(y/n)**: Type **y** and press **Enter** to continue.
- **FER_DIR**: PyFerret installation path. Type */path/to/PyFerret/***pyferret**.
- **FER_DSETS**: FerretDatasets package path. Type */path/to/PyFerret/* **FerretDatasets-7.6**.
- **desired ferret_paths location**: **ferret_paths.sh** script path. Type */path/to/PyFerret/*.
- derret_paths link to create? (c/s/n): Type **n**.
- **python executable to use**: Type **python2.7**.

Step 7 Load the environment variables.

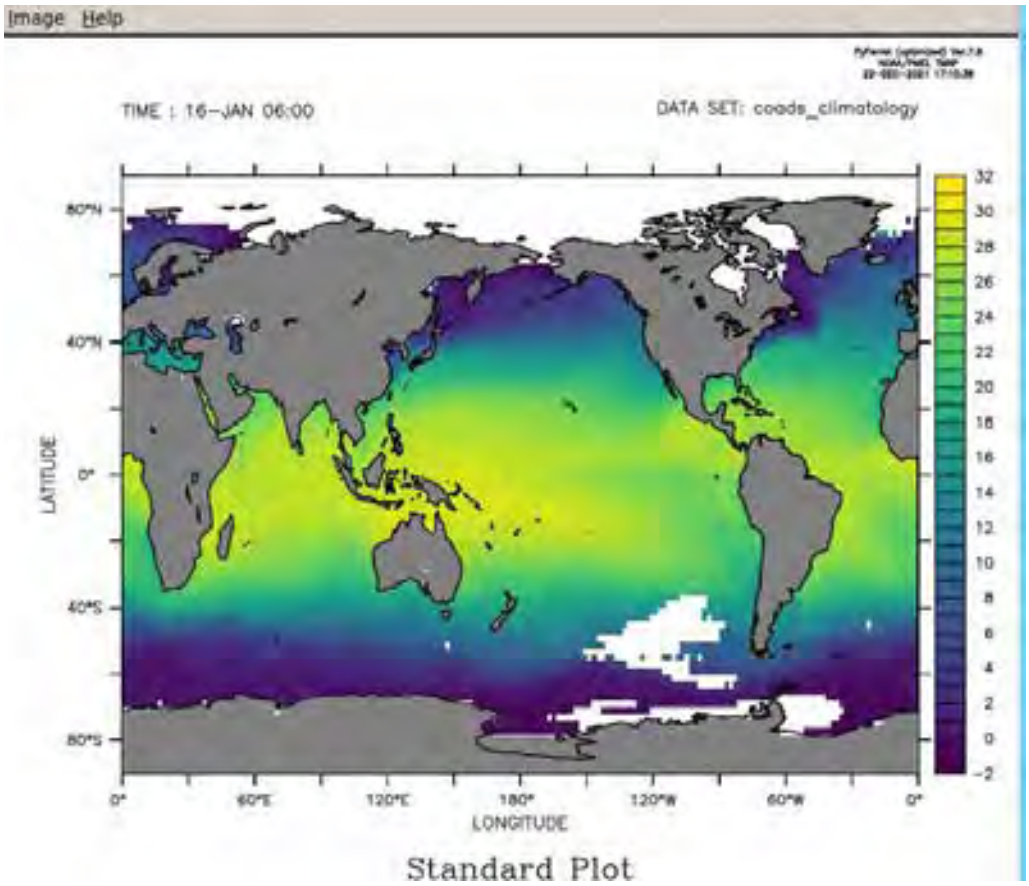
```
cd /path/to/PyFerret
source ferret_paths.sh
```

Step 8 Start the main program.

```
cd /path/to/PyFerret/pyferret/bin
./pyferret
yes? go /help mp_demo
yes? USE coads_climatology
yes? SHADE/X=0:360/Y=-90:90/L=1/TITLE="Standard Plot" sst
yes? GO fland
yes? GO land
yes? exit
```

An example of the command output is as follows:

 Copyright © Huawei Technologies Co., Ltd.



----End

1.37.8 Troubleshooting

1.37.8.1 Error "fatal error: 'hb.h' file not found" Reported During the Compilation

Symptom

A message is displayed stating "/usr/include/pango-1.0/pango/pango-coverage.h:28:10: fatal error: 'hb.h' file not found #include <hb.h>." That means the **hb.h** file is not found.

```
/usr/include/pango-1.0/pango/pango-coverage.h:28:10: fatal error: 'hb.h' file not found
#include <hb.h>
         ^~~~~~
1 error generated.
make[4]: *** [<builtin>: cairoCFerBind_saveWindow.o] Error 1
make[4]: Leaving directory '/path/to/PyFerret/PyFerret-7.6.0/fer/cferbind'
make[3]: *** [Makefile:17: all] Error 2
make[3]: Leaving directory '/path/to/PyFerret/PyFerret-7.6.0/fer/cferbind'
make[2]: *** [Makefile:117: fer_libraries] Error 1
make[2]: Leaving directory '/path/to/PyFerret/PyFerret-7.6.0/fer'
make[1]: *** [Makefile:39: optimized] Error 2
make[1]: Leaving directory '/path/to/PyFerret/PyFerret-7.6.0/fer'
make: *** [Makefile:25: optimized] Error 2
[root@host-10-200-84-240 PyFerret-7.6.0]#
```

Key Process and Cause Analysis

None

Conclusion and Solution

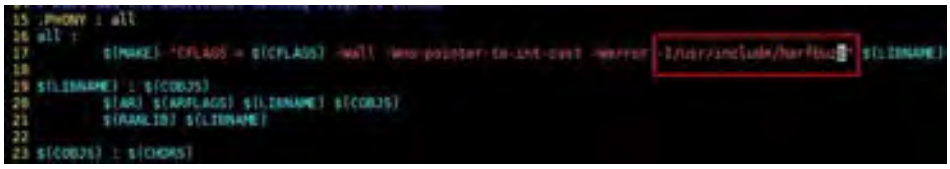
Step 1 Query the directory where **hb.h** is.

```
cd /usr
find -name hb.h
```

```
[root@host-10-208-160-246 PyFerret-7.6.0]# cd /usr
[root@host-10-208-160-246 usr]# find -name hb.h
./include/harfbuzz/hb.h
[root@host-10-208-160-246 usr]#
```

Step 2 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   cd /path/to/PyFerret/PyFerret-7.6.0
   vi /path/to/PyFerret/PyFerret-7.6.0/fer/cferbind/Makefile
   ```
2. Press **i** to enter the edit mode and add **-I/usr/include/harfbuzz** to line 17.



NOTE

After performing **Step 2.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Perform the compilation again.

```
make clean
make
```

----End

1.37.8.2 Error "cannot connect to X server" Reported During the Verification

Symptom

During the verification, a message is displayed stating "PipedViewerPQ: cannot connect to X server".

Key Process and Cause Analysis

X11-forwarding is not enabled.

Conclusion and Solution

Step 1 Exit from the current screen.

```
CTRL+ Z
```

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command:

```
yum install -y xorg-x11-xauth xorg-x11-fonts* xorg-x11-font-utils xorg-x11-fonts-Type1 xclock
```

Step 3 Modify the **/etc/ssh/sshd_config** file.

1. Open the **/etc/ssh/sshd_config** file.
   ```
   vi /etc/ssh/sshd_config
   ```
2. Press **i** to enter the edit mode and change the value of **X11Forwarding** in line 100 to **yes**.

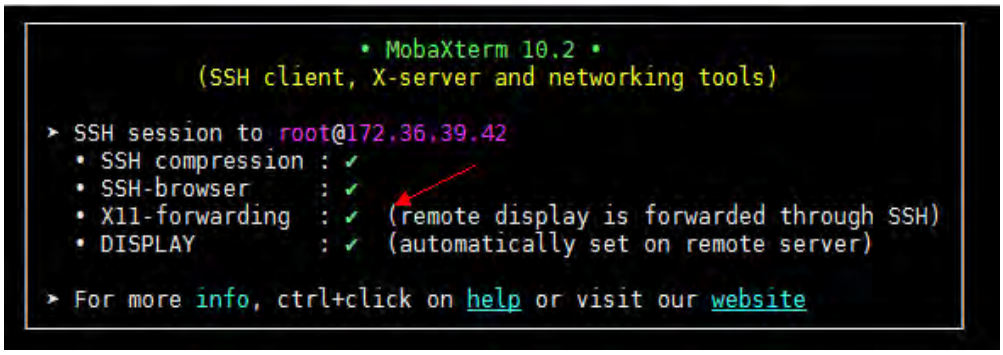
   NOTE

   After performing **Step 3.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Restart the SSHD service.

```
systemctl restart sshd
```

Open the SSH window again and connect to the server. If X11-forwarding is ticked, it is enabled successfully.



Step 5 Perform **Step 7** again.

----End

1.38 SMOKE 4.7 Porting Guide (openEuler 21.03)

1.38.1 Introduction

The Sparse Matrix Operator Kernel Emissions (SMOKE) modeling system is primarily an emissions processing system designed to create gridded, speciated, hourly emissions for input into a variety of air quality models, such as CMAQ, REMSAD, CAMX, and UAM. SMOKE supports area, biogenic, mobile (both onroad and nonroad), and point source emissions processing for criteria, particulate, and toxic pollutants. For biogenic emissions modeling, SMOKE uses the Biogenic Emission Inventory System, version 2.5 (BEIS2) and version 3.09 and 3.14 (BEIS3). SMOKE is also integrated with the onroad emissions model MOBILE6 and MOVES.

For more information about SMOKE, visit the **SMOKE official website**.

Programming language: Fortran 90

Brief description: A source emission tool for air modeling using sparse matrix.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: LGPL 3.0

Recommended Software Version

SMOKE 4.7

1.38.2 Environment Requirements

Hardware Requirements

Table 1-195 lists the hardware requirements.

Table 1-195 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-196 lists the software requirements.

Table 1-196 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
SMOKE	smoke_v47.Linux2_x86_64ifort.tar.gz smoke_v47.nctox.data.tar.gz smoke_install_v47.csh	https://github.com/CEMPD/SMOKE/releases
NetCDF-Fortran	4.4.1	https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.1
I/O API	3.2	https://github.com/cjcoats/ioapi-3.2
NetCDF-C	4.7.0	https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0
PnetCDF	1.8.0	https://parallel-netcdf.github.io/wiki/Download.html

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
Test cases	$SMK_HOME/data/ge_dat	Provided by the software.

OS Requirements

Table 1-197 lists the OS requirements.

Table 1-197 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

1.38.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SMOKE software porting.

Table 1-198 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/HDF5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	/path/to/SMOKE	Installation path of SMOKE.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
6	/path/to/PNETCDF	Installation path of PnetCDF.	

1.38.4 Configuring the Compilation Environment

1.38.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-199 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **1.38.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.38.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.38.4.4 Installing NetCDF-Fortran**.
6	Installing PnetCDF	See **1.38.4.5 Installing PnetCDF**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

1.38.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.5
```

Step 4 Perform the configuration.

```
CC=mpicc CXX=mpicxx FC=mpifort F77=mpif77 CFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC -DPIC" FFLAGS="-fPIC" \./configure --prefix=/path/to/SMOKE/HDF5 --enable-shared --enable-fortran --enable-static --enable-parallel
```

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.38.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Set the variables.

```
export CPPFLAGS="-I/path/toHDF5/include"
export LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/SMOKE/HDF5/lib"
export CFLAGS="-L/path/to/HDF5/lib -I/path/to/SMOKE/HDF5/include - "
export CXXFLAGS="-L/path/to/HDF5/lib -I/path/to/SMOKE/HDF5/include "
export FCFLAGS="-L/path/to/HDF5/lib -I/path/to/SMOKE/HDF5/include"
```

Step 5 Perform the configuration. (If the CC, FC, and CXX environment variables fail to be set, you can use an absolute path.)

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.38.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Set the variables.

```
export CPPFLAGS="-I/path/to/SMOKE/HDF5/include -I/path/to/SMOKE/NETCDF/include"
export LDFLAGS="-L/path/to/SMOKE/HDF5/lib -Wl,-rpath=/path/to/SMOKE/HDF5/lib -L/path/to/SMOKE/
NETCDF/lib -Wl,-rpath=/path/to/SMOKE/NETCDF/lib"
export CFLAGS="-L/path/to/SMOKE/HDF5/lib -I/path/to/SMOKE/HDF5/include -L/path/to/SMOKE/
NETCDF/lib -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/SMOKE/HDF5/lib -I/path/to/SMOKE/HDF5/include -L/path/to/SMOKE/
NETCDF/lib -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/SMOKE/HDF5/lib -I/path/to/SMOKE/HDF5/include -L/path/to/SMOKE/
NETCDF/lib -I/path/to/NETCDF/include"
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --enable-shared --with-pic --disable-doxygen --enable-largefile --
enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpifort CFLAGS="-O3 -fPIC -Wno-
incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" CXXFLAGS="-O3 -
fPIC -Wno-incompatible-pointer-types-discards-qualifiers" FCFLAGS="-O3 -fPIC" FFLAGS="-O3 -fPIC"
```

Step 6 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify lines 10118 and 10270 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

NOTE

After performing Step 6.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make
make install
```

----End

1.38.4.5 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd parallel-netcdf-1.9.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/SMOKE/PNETCDF CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC"
FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpif90
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Set the PnetCDF environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.38.5 Obtaining the Source Code

Procedure

Step 1 Download the SMOKE installation package.

https://github.com/CEMPD/SMOKE/releases

Step 2 Use an SFTP tool to upload the SMOKE installation package to the */path/to/SMOKE* directory on the server.

----End

1.38.6 Compilation and Installation

Installing the I/O API

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Install SMOKE in csh.

```
csh
```

Step 3 Set the environment variables.

```
setenv SMK_HOME /path/to/SMOKE
setenv LD_LIBRARY_PATH /path/to/BISHENG/bisheng-1.3.1/lib:/path/to/HMPI/hmpi-1.0.2/ompi/lib
setenv PATH /path/to/HMPI/hmpi-1.0.2/hmpi/bin:/path/to/BISHENG/bisheng-1.3.1/bin:$PATH
setenv LD_LIBRARY_PATH /path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

Step 4 Copy the installation packages to the installation path.

```
cp smoke_v47.gz $SMK_HOME
cp smoke_v47.nctox.data.tar.gz $SMK_HOME
cp smoke_install.csh $SMK_HOME
cp ioapi-3.2.tar.gz $SMK_HOME
cd $SMK_HOME
source smoke_install_v47.csh
setenv BIN Linux2_aarch64
```

Step 5 Decompress and rename the **ioapi** file.

```
tar -xvf ioapi-3.2-2020111.tar.gz
mv ioapi-3.2-2020111 ioapi-3.2
```

Step 6 Copy the configuration file.

```
cd ioapi-3.2
cp ioapi/Makeinclude.Linux2_ia64gfort ioapi/Makeinclude.Linux2_aarch64
```

Step 7 Modify the **Makeinclude.Linux2_aarch64** configuration file.

1. Open the **Makeinclude.Linux2_aarch64** configuration file.
   ```
   vi ioapi/Makeinclude.Linux2_aarch64
   ```
2. Press **i** to enter the edit mode and modify the compiler options (you can use an absolute path if necessary).
   ```
   CC = mpicc
   CXX = mpicxx
   FC = mpif90
   ```
   Comment out the line corresponding to the **FSFLAGS** keyword, for example, **#FSFLAGS = -save**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Copy the **Makefile** file.

```
cp ioapi/Makefile.nocpl ioapi/Makefile
```

Step 9 Copy the configuration file.

```
cp m3tools/Makefile.nocpl m3tools/Makefile
```

Step 10 Modify the **Makefile** configuration file.

1. Open the **Makefile** configuration file.
   ```
   vi m3tools/Makefile
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   LIBS = -L${OBJDIR} -lioapi -L/path/to/NETCDF/lib -lnetcdff –lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz $(OMPLIBS) $(ARCHLIB) $(ARCHLIBS)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Copy the configuration file.

```
cp Makefile.template Makefile
```

Step 12 Edit the configuration file.

1. Open the configuration file.
   ```
   vi Makefile
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the edit mode.

   Modify the following content and remove the comments. (If the path is incorrect, check whether the path in **BASEDIR** is correct.)

   ```
   BIN = Linux2_aarch64
   BASEDIR = ${PWD}
   INSTALL = ${HOME}
   LIBINST = $(INSTALL)/$(BIN)
   BININST = $(INSTALL)/$(BIN)
   CPLMODE = nocpl
   IOAPIDEFS = "-DIOAPI_NCF4"
   ```

   Modify **NCFLIBS**.

   ```
   NCFLIBS = -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 13 Compile the I/O API.

```
make BIN=Linux2_aarch64
```

----End

Installing SMOKE

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **$SMK_HOME/subsys/smoke/assigns/** directory.

```
cd $SMK_HOME/subsys/smoke/assigns/
```

Step 3 Modify the **ASSIGNS.nctox.cmaq.cb05_soa.us12-nc** file.

1. Open the **ASSIGNS.nctox.cmaq.cb05_soa.us12-nc** file.

   ```
   vi ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
   ```

2. Press **i** to enter the edit mode and modify the content in line 25.

   ```
   setenv BIN Linux2_aarch64
   ```

   NOTE

   After performing **Step 3.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Modify the **Makeinclude** file.

1. Open the **Makeinclude** file.

   ```
   cd $SMK_HOME/subsys/smoke/src
   vi Makeinclude
   ```

2. Press **i** to enter the edit mode and modify the following content. (If an error is reported for the link library, add the absolute path of the library in the format of **-L/***Absolute path*.)

   ```
   INSTDIR = ${OBJDIR}/${BIN}
   #EFLAG = -extend-source 132 -zero
   EFLAG = -ffixed-line-length-132 -fno-backslash
   IOLIB = -L$(IOBIN) -lioapi -L/path/to/NETCDF/lib -lnetcdf -lnetcdff
   IOBASE  = ${SMK_HOME}/ ioapi-3.2
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the **biog/normbeis361.f** file.

1. Open the **biog/normbeis361.f** file.

   ```
   vi biog/normbeis361.f
   ```

2. Press **i** to enter the insert mode and change the types of the IS_AG and IS_TAG functions in line 161 from **INTEGER** to **LOGICAL**.

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Modify the **emutil/gentpro.f** file.

1. Open the **emutil/gentpro.f** file.
```
vi emutil/gentpro.f
```
2. Press **i** to enter the edit mode and modify lines 1663 and 1695 of the file.
```
1663 (( PROF_MON( S,NP )), NP = 1,12 )
1695 ( ( PROF_DAY( S,NP ) ), NP = 1,31 )
```

> **NOTE**
>
> After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Create the directory.
```
mkdir ${SMK_HOME}/subsys/smoke/${BIN}
```

Step 8 Perform the compilation.
```
source $SMK_HOME/subsys/smoke/assigns/ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
make
```

----End

1.38.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Start the parallel execution.
```
cd $SMK_HOME/subsys/smoke/script/run/
```

Step 3 Go to cshell.
```
csh
```

Step 4 Test the execution time of each script.
```
time./Script name
```

For example: **time ./smk_rateperdistance_nctox.csh**

```
28.269u 0.071s 0:28.36 99.8%    0+0k 0+54016io 0pf+0w
54.079u 3.890s 0:32.54 178.1%   0+0k 0+108160io 0pf+0w
```

----End

1.39 SPECFEM3D GLOBE 7.0.0 Porting Guide (openEuler 21.03)

 Copyright © Huawei Technologies Co., Ltd.

1.39.1 Introduction

SPECFEM3D_GLOBE simulates global and regional (continental-scale) seismic wave propagation. Additional new model routines are provided for the Comprehensive Earth Model (CEM) project, generic point-profile models (PPM) and Gauss-Lobatto-Legendre based models (GLL), with complementary tools for postprocessing adjoint sensitivity kernels and gradient-based model updates. The structure of the software has been simplified to facilitate easier implementation of new 3D models. The code accommodates general moment tensor files, and provides complete information in the SAC headers, as explained in detail in the updated user manual. Loop-vectorization helps reduce the total number of memory accesses performed in each spectral element and improve code vectorization, enhancing numerical performance of the version.

For more information about SPECFEM3D GLOBE, visit the **SPECFEM3D GLOBE official website**.

Programming languages: C/Fortran

Brief description: SPECFEM3D GLOBE simulates global and regional (continental scale) seismic wave propagation.

Open source license: GPL 3.0

Recommended Software Version

SPECFEM3D GLOBE 7.0.0

1.39.2 Environment Requirements

Hardware Requirements

Table 1-200 lists the hardware requirements.

Table 1-200 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-201 lists the software requirements.

Table 1-201 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
SPECFEM3D GLOBE	7.0.0	https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz

OS Requirements

Table 1-202 lists the OS requirements.

Table 1-202 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

1.39.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SPECFEM3D GLOBE software porting.

Table 1-203 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
5	/path/to/SPECFEM3D GLOBE	Installation path of SPECFEM3D GLOBE.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

1.39.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-204 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

1.39.5 Obtaining the Source Code

Procedure

Step 1 Download the SPECFEM3D GLOBE installation package.

https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz

Step 2 Use an SFTP tool to upload the SPECFEM3D GLOBE installation package to the */path/to/SPECFEM3D GLOBE* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.39.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create and go to the main program installation directory.

```
mkdir -p /path/to/SPECFEM3D_GLOBE
cd /path/to/SPECFEM3D_GLOBE
```

Step 3 Decompress the source package and go to the source code directory.

```
tar -xzvf specfem3d_globe-7.0.0.tar.gz
cd specfem3d_globe-7.0.0
```

Step 4 Generate a configuration file.

```
./configure FC=mpif90 CC=mpicc CXX=mpicxx
```

Step 5 Modify the **Par_file** file.

1. Open the **Par_file** file.
   ```
   vi DATA/Par_file
   ```
2. Press **i** to enter the edit mode and modify lines 23 and 24 of the file.

   Before the modification:

   ```
   NPROC_XI = 2
   NPROC_ETA = 2
   ```

   After the modification:

   ```
   NPROC_XI = 8
   NPROC_ETA = 8
   ```

   NOTE

   After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 6 Perform the compilation.

```
make
```

----End

1.39.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/SPECFEM3D_ GLOBE/specfem3d_globe-7.0.0
```

Step 3 Perform the single-node test.

```
mpirun --allow-run-as-root -np 64 --mca btl ^openib ./bin/xmeshfem3D
mpirun --allow-run-as-root -np 64 --mca btl ^openib ./bin/xspecfem3D
vim OUTPUT_FILES/output_solver.txt
```

If the following information is displayed, the running is successful:

 Copyright © Huawei Technologies Co., Ltd.

Step 4 If the test must be performed on more than one node, edit the **hostfile** file.

1. Open the **hostfile** file.
   ```
   vi hostfile
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   Node1
   Node2
   ```

   NOTE

   Node1 and **Node2** are the node names. Change them based on the site requirements and ensure that the nodes can ping each other.
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the changes and exit.

Step 5 Perform the multiple-node test.

```
mpirun --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH --hostfile hostfile -np 32 –N 16 --mca btl ^openib ./bin/xmeshfem3D
mpirun --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH –np 32 -N 16 --mca btl ^openib ./bin/xspecfem3D
```

Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
--hostfile	Name of the node to be used.

----End

1.40 CMAQ 5.3.1 Porting Guide (openEuler 21.03)

1.40.1 Introduction

Community Multiscale Air Quality (CMAQ) is an active open-source development project that consists of a suite of programs for conducting air quality model simulations. CMAQ combines current knowledge in atmospheric science and air quality modeling, multi-processor computing techniques, and an open-source framework to deliver fast, technically sound estimates of ozone, particulates, toxics and acid deposition.

For more information about CMAQ, visit the **CMAQ official website**.

Programming language: Fortran

Brief description: An active open-source development project that consists of a suite of programs for conducting air quality model simulations.

Open source license: GPL 3.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

CMAQ 5.3.1

1.40.2 Environment Requirements

Hardware Requirements

Table 1-205 lists the hardware requirements.

Table 1-205 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-206 lists the software requirements.

Table 1-206 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
HPCX	2.6	https://www.mellanox.com/page/hpcx_eula?mtag=hpc-x&mrequest=downloads&mtype=hpc&mver=hpc-x&mname=v2.6/hpcx-v2.6.0-gcc-OFED-4.17-redhat7.6-aarch64.tbz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
NetCDF-C	4.7.0	https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0
NetCDF-Fortran	4.4.5	https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5
OPTIMIZED-ROUTINES	V20.02	https://codeload.github.com/ARM-software/optimized-routines/tar.gz/v20.02
IOAPI	3.2	https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Test cases	2016_12SE1	https://dataverse.unc.edu/dataset.xhtml?persistentId=doi:10.15139/S3/IQVABD

OS Requirements

Table 1-207 lists the OS requirements.

Table 1-207 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

1.40.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CMAQ software porting.

Table 1-208 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/CMAQ	Installation path of CMAQ.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	/path/to/HDF5	Installation path of HDF5.	
6	/path/to/MATH	Installation path of OPTIMIZED-ROUTINES.	

 Copyright © Huawei Technologies Co., Ltd.

1.40.4 Configuring the Compilation Environment

1.40.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-209 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **1.40.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.40.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.40.4.4 Installing NetCDF-Fortran**.
6	Installing OPTIMIZED-ROUTINES	See **1.40.4.5 Installing OPTIMIZED-ROUTINES**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.40.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

NOTE

If the server has multiple MPIs, **CC**, **FC**, and **CXX** in the preceding command can be represented using absolute paths.

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify line 11835 in the file.
   ```
   wl="-Wl,"
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.40.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the installation.

```
make
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.40.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-foortran-4.4.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-
parallel-tests --enable-large-file-tests
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

1.40.4.5 Installing OPTIMIZED-ROUTINES

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OPTIMIZED-ROUTINES installation package.

```
tar -xvf optimized-routines-20.02.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd optimized-routines-20.02
```

Step 4 Create a **MATH** directory.

```
mkdir -p /path/to/MATH
```

Step 5 Copy the configuration file.

```
cp config.mk.dist config.mk
```

Step 6 Modify the **config.mk** file.

1. Open the **config.mk** file.
   ```
   vi config.mk
   ```

2. Press **i** to enter the edit mode.

   Modify the content in line 13.

   ```
   CC = $(CROSS_COMPILE)clang
   ```

   Modify the content in line 19.

   ```
   HOST_CC = clang
   ```

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make
```

Step 8 Copy the library file.

```
cp -r ./build/* /path/to/MATH
```

Step 9 Add **MATH** to the environment variables.

```
export PATH=/path/to/MATH/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MATH/lib:$LD_LIBRARY_PATH
```

----End

1.40.5 Obtaining the Source Code

Procedure

Step 1 Download the CMAQ dependency package.

https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111

Step 2 Download the CMAQ installation package.

https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019

Step 3 Use an SFTP tool to upload the CMAQ installation package to the */path/to/CMAQ* directory on the server.

----End

1.40.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the CMAQ installation directory.

```
mkdir -p /path/to/CMAQ
```

Step 3 Decompress and rename the **ioapi** file.

```
tar -xvf ioapi-3.2-2020111.tar.gz
mv ioapi-3.2-2020111 ioapi-3.2
```

Step 4 Copy the configuration file.

```
cd ioapi-3.2
cp ioapi/Makeinclude.Linux2_ia64gfort ioapi/Makeinclude.Linux4_aarch64
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Modify the **Makeinclude.Linux4_aarch64** configuration file.

1. Open the **Makeinclude.Linux4_aarch64** configuration file.
   vi ioapi/Makeinclude.Linux4_aarch64
2. Press **i** to enter the edit mode.

   Modify the compiler options.

   ```
   CC = mpicc
   CXX = mpicxx
   FC = mpifort
   ```

   Comment out the parameters corresponding to some keywords, as shown in the following figure:

   ```
   AR      = ar
   CC      = mpicc
   CXX     = mpicxx
   FC      = mpif90
   M4      = m4
   LINK    = $(CC)

   MOD1 = -I       #% Module-include-directory command
   #D132 = -ffixed-line-length-132

   MFLAGS    = -ffast-math -funroll-loops  #  -Wall -Wsurprising -Waliasing -fbounds-check

   #OMPFLAGS  =  -fopenmp
   #OMPLIBS   =  -fopenmp
   COPTFLAGS = -O3 $(MFLAGS)
   FOPTFLAGS = -O3 $(MFLAGS)
   FPPFLAGS  = -DEF90=1
   #FSFLAGS   =  -save
   ARCHFLAGS = \
     -DAUTO_ARRAYS=1 \
     -DF90=1 -DFLOOR=1 \
     -DFSTR_L=int \
     -DIOAPI_NO_STDOUT=1 \
     -DNEED_ARGS=1
   PARFLAGS  =
   ARCHLIB   =  -dynamic -lm -lpthread -lc
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Copy the **Makefile** file and configure **HOME**.

```
cp ioapi/Makefile.nocpl ioapi/Makefile
export HOME=/path/to/CMAQ
```

Step 7 Copy the configuration file.

```
cp m3tools/Makefile.nocpl m3tools/Makefile
```

Step 8 Modify the **Makefile** configuration file.

1. Open the **Makefile** configuration file.
   vi m3tools/Makefile
2. Press **i** to enter the edit mode and modify the file as follows:
   LIBS = -L${OBJDIR} -lioapi -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to /HDF5/lib -lhdf5_hl -lhdf5 -lz $(OMPLIBS) $(ARCHLIB) $(ARCHLIBS)
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Copy the configuration file.

```
cp Makefile.template Makefile
```

Step 10 Modify the **Makefile** configuration file.

1. Open the **Makefile** configuration file.
   vi Makefile
2. Press **i** to enter the edit mode.

   Modify the following content and delete the comment tag (#).

   ```
   BIN = Linux4_aarch64
   BASEDIR = ${PWD}
   INSTALL = ${HOME}
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
LIBINST = $(INSTALL)/$(BIN)
BININST = $(INSTALL)/$(BIN)
CPLMODE = nocpl
IOAPIDEFS = "-DIOAPI_NCF4"
```

Modify the **NCFLIBS** parameter in lines 146 and 193.

```
NCFLIBS = -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz
```

NOTE

After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

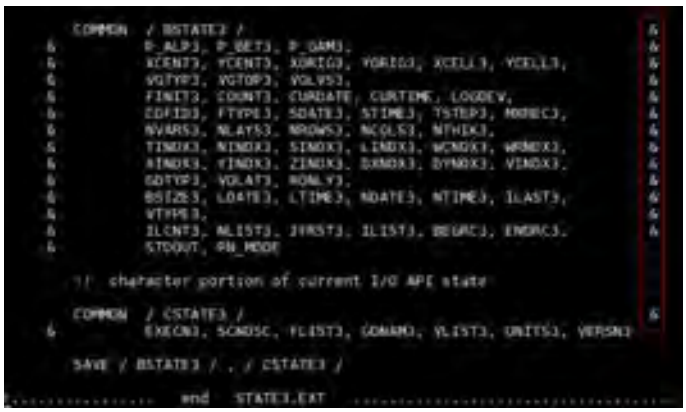
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Compile IOAPI.

```
make BIN=Linux4_aarch64
```

Step 12 Modify the **STATE3.EXT** file.

1. Open the **STATE3.EXT** file.
   ```
   vi ioapi/STATE3.EXT
   ```
2. Press **i** to enter the edit mode. Delete the ampersands (&) at the end of some lines in the **STATE3.EXT** file, as shown in the following figure:



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 13 Go to the C shell environment and configure HOME.

```
csh
setenv HOME /path/to/CMAQ
```

Step 14 Add the environment variables to the C shell environment.

```
setenv PATH /path/to/Bisheng131/bin:$PATH
setenv INCLUDE /path/to/Bisheng131/include
setenv LD_LIBRARY_PATH /path/to/Bisheng131/lib
setenv PATH /path/to/HMPI_1.0.2_aarch64/source/hmpi/bin:$PATH
setenv INCLUDE /path/to/HMPI_1.0.2_aarch64/source/hmpi/include:$INCLUDE
setenv LD_LIBRARY_PATH /path/to/HMPI_1.0.2_aarch64/source/hmpi/lib:$LD_LIBRARY_PATH
```

Step 15 Decompress the CMAQ installation package and go to the directory generated after the decompression.

```
cd /path/to/CMAQ
tar -xzvf CMAQ-CMAQv5.3.1_19Dec2019.tar.gz
cd CMAQ-CMAQv5.3.1_19Dec2019
```

Step 16 Modify the **bldit_project.csh** configuration file.

1. Open the **bldit_project.csh** configuration file.
   ```
   vi bldit_project.csh
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   set CMAQ_HOME = /path/to/CMAQ/CMAQ_Project
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 17 Create the file required for the initialization.

```
./bldit_project.csh
```

Step 18 Go to the working directory.

```
cd ../CMAQ_Project/
```

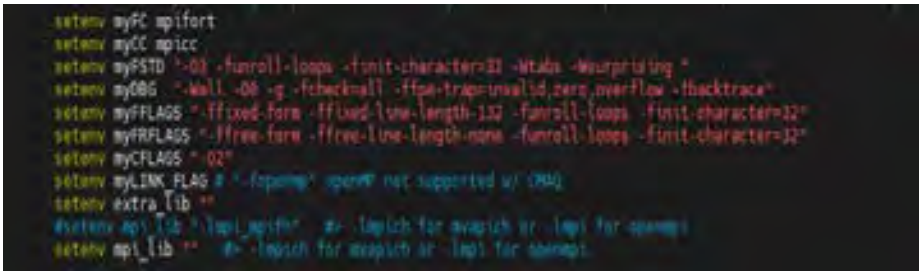
Step 19 Modify the **config_cmaq.csh** configuration file.

1. Open the **config_cmaq.csh** configuration file.
   ```
   vi config_cmaq.csh
   ```
2. Press **i** to enter the edit mode.

   In the **case gcc** area, change **case gcc** to **case bisheng**.

   ```
   setenv IOAPI_INCL_DIR /path/to/CMAQ/ioapi-3.2/ioapi/
   setenv IOAPI_LIB_DIR /path/to/CMAQ/ioapi-3.2/Linux4_aarch64/
   setenv NETCDF_LIB_DIR /path/to/NETCDF/lib/
   setenv NETCDF_INCL_DIR /path/to/NETCDF/include/
   setenv MPI_LIB_DIR /path/to/HMPI/
   setenv NETCDFF_LIB_DIR /path/to/NETCDF/lib/
   setenv NETCDFF_INCL_DIR /path/to/NETCDF/include/
   ```

   Modify the compiler parameters, as shown in the following figure. (A warning may be displayed during the compilation but it does not affect the compilation. You can also delete the related parameters according to the warning.)

   Comment out the following content:

   ```
   #if ( ! -e $NETCDFF_DIR/lib/libnetcdff.a ) then
   #    echo "ERROR: $NETCDFF_DIR/lib/libnetcdff.a does not exist in your CMAQ_LIB directory!!! Check your installation before proceeding with CMAQ build."
   #    exit
   # endif
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 20 Connect to the dependent libraries.

```
./config_cmaq.csh bisheng 1.3.1
```

Step 21 Go to the compilation directory and compile the main program.

```
cd CCTM/scripts/
cp /path/to/HMPI/include/mpif.h /path/to/CMAQ/CMAQ_Project/CCTM/scripts/
BLD_CCTM_v531_bisheng1.3.1
./bldit_cctm.csh bisheng 1.3.1
```

After the compilation is complete, the executable program **CCTM_v531.exe** is generated in the **BLD_CCTM_v531_bisheng1.3.1** directory.

----End

1.40.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the test case file.

```
tar -xvf CMAQv5.3.1_Benchmark_2Day_Input.tar.gz -C /path/to/CMAQ/CMAQ_Project/data
```

Step 3 Go to the test directory.

```
cd /path/to/CMAQ/CMAQ_Project/CCTM/scripts
```

Step 4 Configure the running script.

1. Open the running script.
   ```
   vim run_cctm_Bench_2016_12SE1.csh
   ```
2. Press **i** to enter the edit mode.

   Change the value of **compiler** to **bisheng** and that of **compilerVrsn** to **1.3.1**.

   ```
   setenv compiler bisheng
   setenv compilerVrsn 1.3.1
   ```

   Change the values of **NPCOL** and **NPROW**. The product of the two values is equal to the number of parallel cores. For example:

   ```
   @ NPCOL = 8; @ NPROW = 4
   ```

   Modify the INPDIR path, for example:

   ```
   setenv INPDIR /path/to/CMAQ/CMAQ_Project/data/CMAQv5.3.1_Benchmark_2Day_Input/2016_12SE1
   ```

   Modify the MPI running parameters. For example:

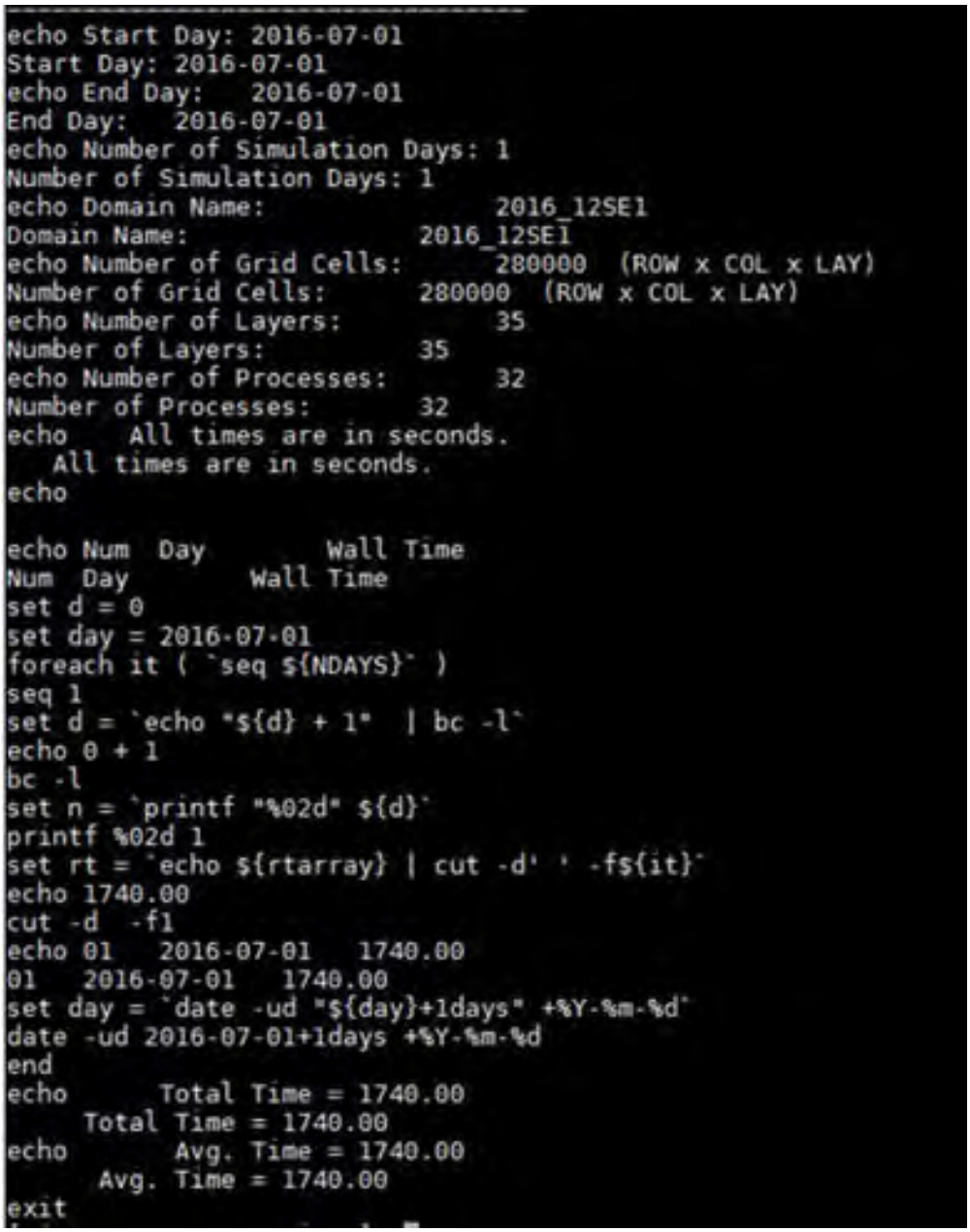
   ```
   /usr/bin/time -p mpirun -np 32  -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so --allow-run-as-root
   --mca btl ^openib $BLD/$EXEC
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the test.

```
csh -x /path/to/CMAQ/CMAQ_Project/CCTM/scripts/run_cctm_Bench_2016_12SE1.csh >& /path/to/
CMAQ/CMAQ_Project/CCTM/scripts/csh.log
```

Check the value of **Total Time** (unit: s) in the **csh.log** file. A smaller value indicates higher performance.

 Copyright © Huawei Technologies Co., Ltd.



----End

1.40.8 Troubleshooting

1.40.8.1 Error Reported When Executing the make Command During the OPTIMIZED-ROUTINES Installation

Symptom

When executing the **make** command during the OPTIMIZED-ROUTINES installation, an error is reported stating "/usr/bin/ld: cannot find -lm" and "/usr/bin/ld: cannot find –lc."

Key Process and Cause Analysis

The environment dependency is missing.

Conclusion and Solution

Run the following command:

```
yum install -y glibc-static
```

1.40.8.2 Error Reported When Running and Verifying a CMAQ Test

Symptom

When running and verifying a CMAQ test, a message is displayed indicating that the **libnetcdf.so** library is missing.

Key Process and Cause Analysis

The required variable is not included in the environment variables.

Conclusion and Solution

Add the environment variable to the test environment.

```
setenv LD_LIBRARY_PATH /path/to/CMAQ/NETCDF/lib:$LD_LIBRARY_PATH
```

1.41 POM POM2K Porting Guide (openEuler 21.03)

1.41.1 Introduction

The Princeton Ocean Model (POM), a sigma coordinate, free surface, ocean model, which includes a turbulence sub-model. It was developed in the late 1970's by Blumberg and Mellor, with subsequent contributions from other people. The model has been used for modeling of estuaries, coastal regions, basin and global oceans.

For more information about POM, visit the **POM official website**.

Programming language: Fortran

Brief description: A sigma coordinate, free surface, ocean model, which includes a turbulence sub-model.

Open source license: GPL 3.0

Recommended Software Version

POM POM2K

 Copyright © Huawei Technologies Co., Ltd.

1.41.2 Environment Requirements

Hardware Requirements

Table 1-210 lists the hardware requirements.

Table 1-210 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-211 lists the software requirements.

Table 1-211 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
POM	POM2K	http://www.ccpo.odu.edu/POMWEB/userinfo.htm
NetCDF-C	4.7.0	https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0

OS Requirements

Table 1-212 lists the OS requirements.

Table 1-212 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

1.41.3 Paths for Software Porting

This chapter lists the software installation paths involved in the POM software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-213 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/NETCDF	Installation path of NetCDF-C.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
4	/path/to/POM	Installation path of POM.	

1.41.4 Configuring the Compilation Environment

1.41.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-214 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing NetCDF-C	See **1.41.4.2 Installing NetCDF-C**.

 Copyright © Huawei Technologies Co., Ltd.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

1.41.4.2 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration. (If the CC, FC, and CXX environment variables fail to be set, you can use an absolute path.)

```
./configure --prefix=/path/to/NETCDF CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the compilation and installation.

```
make
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.41.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a POM directory.

 Copyright © Huawei Technologies Co., Ltd.

```
mkdir –p /path/to/POM
```

Step 3 Go to the POM directory.

```
cd /path/to/POM
```

Step 4 Download the POM software and the test case **GRID-DATA** (ensure that your computer is connected to the Internet):

```
wget -r -p -k -np -nc -e robots=off http://www.ccpo.odu.edu/POMWEB/GRID-DATA
```

----End

1.41.6 Generating an IC Data File

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/POM
```

Step 3 Move the POM2K and GRID-DATA directories to the installation directory of the main program (other files can be deleted).

```
cd www.ccpo.odu.edu/POMWEB/
mv POM2K ../../
mv GRID-DATA ../../
```

Step 4 Go to the directory where the GRID-DATA test case is stored.

```
cd /path/to/POM/GRID-DATA
```

Step 5 Create the **IC** and **include** directories.

```
mkdir IC
mkdir -p /path/to/POM/GRID-DATA/IC/include
```

Step 6 Copy the header files **gridborder**, **gridcom**, and **params** to the **include** directory.

```
cp gridborder /path/to/POM/GRID-DATA/IC/include
cp gridcom /path/to/POM/GRID-DATA/IC/include
cp params /path/to/POM/GRID-DATA/IC/include
```

Step 7 Modify the **rungrid** file.

1. Open the **rungrid** file.
   ```
   vi rungrid
   ```
2. Press **i** to enter the edit mode and modify the file as follows:

   Before the modification:

   ```
   echo ' Start Compiling ...'
   g77 -O3 GRID.f
   #pgf77 GRID.f
   a.out > printout
   #
   # -------------- output files for POM2K ------------------------
   /bin/mv fort.40 IC.dat # formatted IC,grid & constant wind
   # -------------- output files for MATLAB plotting -------------
   /bin/mv fort.43 ijk.dat # IM, JM, KB, Z
   /bin/mv fort.44 plt.dat # formatted grid
   /bin/mv fort.45 bnd.dat # boundary points (gridborder)
   /bin/mv fort.46 wnd.dat # wind velocity
   /bin/mv fort.47 dxy.dat # dx,dx (km)
   #
   /bin/rm a.out
   /bin/rm fort.*
   echo ' *** END GRID ***'
   ```

 Copyright © Huawei Technologies Co., Ltd.

After the modification:

```
echo ' Start Compiling ...'
flang -O3 GRID.f -I/path/to/POM/GRID-DATA/IC/include
#pgf77 GRID.f
#./a.out > printout
#
# ---------- output files for POM2K ------------------------
#mv fort.40 IC.dat # formatted IC,grid & constant wind
# ---------- output files for MATLAB plotting ----------
#mv fort.43 ijk.dat # IM, JM, KB, Z
#mv fort.44 plt.dat # formatted grid
#mv fort.45 bnd.dat # boundary points (gridborder)
#mv fort.46 wnd.dat # wind velocity
#mv fort.47 dxy.dat # dx,dx (km)
#
#/bin/rm a.out
#/bin/rm fort.*
echo ' *** END GRID ***'
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Copy the data files to the **IC** directory.

```
cp /path/to/POM/GRID-DATA/TOPO.dat /path/to/POM/GRID-DATA/IC
cp /path/to/POM/GRID-DATA/TS.dat /path/to/POM/GRID-DATA/IC
cp /path/to/POM/GRID-DATA/WIND.dat /path/to/POM/GRID-DATA/IC
cp /path/to/POM/GRID-DATA/GRID.f /path/to/POM/GRID-DATA/IC
cp /path/to/POM/GRID-DATA/IC.dat /path/to/POM/GRID-DATA/IC
```

Step 9 Set the environment variables.

```
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
```

Step 10 Grant the permission to **rungrid** and execute it.

```
chmod 755 rungrid
./rungrid
```

----End

1.41.7 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the POM installation directory.

```
cd /path/to/POM/POM2K
```

Step 3 Configure the related files.

1. Open the **pom2k.f** file.
   ```
   vi pom2k.f
   ```
2. Press **i** to enter the edit mode and change mode **1** to **3**.

   Before the modification:

   ```
   iproblem=1
   ```

   After the modification:

   ```
   iproblem=3
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Copy the header file **gridcom** in the installation package and the **IC.dat** file generated using the IC data to the current directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cp /path/to/POM/GRID-DATA/IC/include/gridcom ./
cp /path/to/POM/GRID-DATA/IC/IC.dat ./
```

Step 5 Modify the **runpom2k** file.

1. Open the **runpom2k** file.
   ```
   vi runpom2k
   ```
2. Press **i** to enter the edit mode.

   Add **#** at the beginning of lines 13 to 17 to comment out the lines.

   Delete **#** at the beginning of lines 21 to 25 to uncomment the lines.

   Add **#** at the beginning of line 37 to comment out the line.

   Change line 32 to the following content:

   ```
   flang -O3 pom2k.f /path/to/NETCDF/lib/libnetcdf.a -I/path/to/NETCDF/lib
   ```

   NOTE

   After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Grant the permission to **runpom2k** and execute it.

```
chmod 755 runpom2k
time ./runpom2k
```

The following is an example of the command output:

```
[zju@DC2-03-006 POM2K]$ time ./runpom2k
/usr/bin/ld: /home/zju/compiler/bisheng-1.3.
Warning: ieee_underflow is signaling
Warning: ieee_inexact is signaling
FORTRAN STOP

real    0m15.049s
user    0m14.995s
sys     0m0.033s
```

----End

1.42 FVCOM 4.1 Porting Guide (openEuler 21.03)

1.42.1 Introduction

FVCOM is a prognostic, unstructured-grid, finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model. The model consists of momentum, continuity, temperature, salinity and density equations and is closed physically and mathematically using turbulence closure submodels. The horizontal grid is comprised of unstructured triangular cells and the irregular bottom is presented using generalized terrain-following coordinates. The General Ocean Turbulent Model (GOTM) developed by Burchard's research group in Germany (Burchard, 2002) has been added to FVCOM to provide optional vertical turbulent closure schemes. FVCOM is solved numerically by a second-order accurate discrete flux calculation in the integral form of the governing equations over an unstructured triangular grid. This approach combines the best features of finite-element methods (grid flexibility) and finite-difference methods (numerical

 Copyright © Huawei Technologies Co., Ltd.

efficiency and code simplicity) and provides a much better numerical representation of both local and global momentum, mass, salt, heat, and tracer conservation. The ability of FVCOM to accurately solve scalar conservation equations in addition to the topological flexibility provided by unstructured meshes and the simplicity of the coding structure has make FVCOM ideally suited for many coastal and interdisciplinary scientific applications.

For more information about FVCOM, visit the **FVCOM official website**.

Programming languages: C/Fortran 90

Brief description: A prognostic, unstructured-grid, finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model.

Open source license: GPL 3.0

Recommended Software Version

FVCOM 4.1

1.42.2 Environment Requirements

Hardware Requirements

Table 1-215 lists the hardware requirements.

Table 1-215 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-216 lists the software requirements.

Table 1-216 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz**
FVCOM	4.1	**http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz**
metis	4.0	Provided by the software.

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
NetCDF-C	4.7.0	https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0
NetCDF-Fortran	4.4.5	https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5
HDF5	1.10.5	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz

OS Requirements

Table 1-217 lists the OS requirements.

Table 1-217 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

1.42.3 Paths for software porting

This chapter lists the software installation paths involved in the FVCOM software porting.

Table 1-218 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/HDF5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
5	/path/to/FVCOM	Installation path of FVCOM.	
6	/path/to/CASE	Path of FVCOM test cases.	

1.42.4 Configuring the Compilation Environment

1.42.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-219 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **1.42.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **1.42.4.3 Installing NetCDF-C**.
5	Installing NetCDF-Fortran	See **1.42.4.4 Installing NetCDF-Fortran**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

1.42.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11835 in the **libtool** file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

1.42.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the installation.

```
make
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.42.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the file.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-
fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-
pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--
build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the edit mode and modify line 11838 in the file.
   ```
   wl='-Wl,'
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

----End

1.42.5 Obtaining the Source Code

Procedure

Step 1 Download the FVCOM installation package.

http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz

Step 2 Use an SFTP tool to upload the FVCOM installation package to the */path/to/FVCOM* directory on the server.

----End

1.42.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the FVCOM installation package directory.

```
cd /path/to/FVCOM
```

Step 3 Decompress the installation package.

```
tar -xvf fvcom-4.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd FVCOM4.1
```

Step 5 Copy the **make.inc_example** file and rename it to **make.inc**.

```
cp Examples/Estuary/make.inc_example FVCOM_source/make.inc
```

Step 6 Create a soft link for the **make.inc** file.

```
ln -sf FVCOM_source/make.inc ./
```

Step 7 Edit the **make.inc** file.

1. Open the **make.inc** file.
   ```
   vi make.inc
   ```
2. Press **i** to enter the edit mode and modify the content in lines 51, 79, 80, 97, and 99.

   Line 51:

   ```
   TOPDIR = /path/to/FVCOM/FVCOM4.1/FVCOM_source
   ```

   NOTE

   Delete the space next to **/path/to/FVCOM/FVCOM4.1/FVCOM_source**.

   Lines 79 and 80:

   ```
   LIBDIR = -L$(INSTALLDIR)/lib -L../METIS_source/metis -L./libs/julian
   INCDIR = -I$(INSTALLDIR)/include -I../METIS_source/metis -I./libs/julian
   ```

   NOTE

   Delete the comment tag (#).

   In line 97, modify the IOLIBS content that is not commented out using **#**.

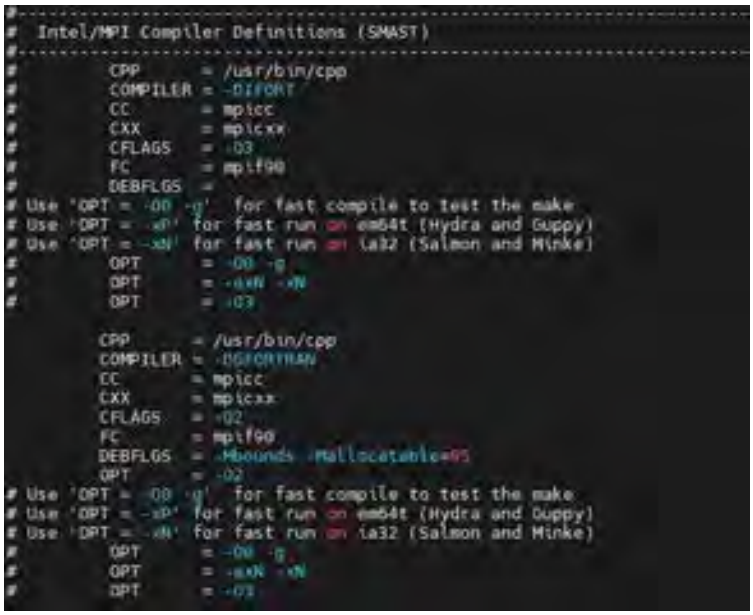
   ```
   IOLIBS = -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -lnetcdff -lnetcdf -lhdf5_hl -lhdf5 -lz -lcurl -lm
   ```

   In line 98, modify the IOINCS content that is not commented out using **#**.

 Copyright © Huawei Technologies Co., Ltd.

```
IOINCS = -I/path/to/NETCDF/include -I/path/to/HDF5/include
```

Comment out and add the content in the following figure:



NOTE

After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Create a compilation environment variable file.

1. Create an environment variable file.
   ```
   vi env-fvcom.sh
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   export CXX=mpicxx
   export CC=mpicc
   export MPI_HOME=/path/to/HMPI
   export PATH=$MPI_HOME/bin:$PATH
   export LD_LIBRARY_PATH=$MPI_HOME/lib:$LD_LIBRARY_PATH
   export HDF5=/path/to/HDF5
   export PATH=$HDF5/bin:$PATH
   export LD_LIBRARY_PATH=$HDF5/lib:$LD_LIBRARY_PATH
   export NETCDF=/path/to/NETCDF
   export PATH=$NETCDF/bin:$PATH
   export LD_LIBRARY_PATH=$NETCDF/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Load the environment variables.

```
source env-fvcom.sh
```

Step 10 Go to the METIS installation directory.

```
cd ./METIS_source
```

Step 11 Decompress the METIS installation package.

```
tar -zxvf metis.tgz
cd metis
sed -i 's/log2/log2_function/g' *.*
```

Step 12 Create the related directories.

 Copyright © Huawei Technologies Co., Ltd.

```
mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/lib
mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/include
mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/bin
```

Step 13 Compile and install the METIS library.

```
make
make install
```

Step 14 Go to the julian installation directory.

```
cd ../../FVCOM_source/libs
```

Step 15 Decompress the **julian.tgz** installation package.

```
tar -zxvf julian.tgz
```

Step 16 Go to the directory generated after the decompression.

```
cd julian
```

Step 17 Compile and install the julian library.

```
make
make install
```

Step 18 Go to the FVCOM installation directory.

```
cd /path/to/FVCOM/FVCOM4.1/FVCOM_source
```

Step 19 Edit the **mod_newinp.F** file.

1. Open the **mod_newinp.F** file.
   ```
   vi mod_newinp.F
   ```
2. Press **i** to enter the edit mode.

   Add the following information in bold below the first **contains** statement.

   ```
   !---------------------------------------
   ! functions
   !---------------------------------------
   contains
   Character( Len = 256 ) Function N_Fmt( c , n )
   Character( Len = * ) , Intent( IN ) :: c
   Integer , Intent( IN ) :: n
   integer :: i , j
   character( len = 16 ) :: cn
   i = index( c , '<' )
   j = index( c , '>' )
   write( cn , '(g0)' ) n
   N_Fmt = c(:i-1) // Trim(adjustL(cn)) // c(j+1:)
   End Function N_Fmt
   ```

   Modify the content in lines 352, 421, 494, and 567.

   Line 352:

   ```
   write(*,N_Fmt('(A20,<size>F10.4)',SIZE))trim(argname)//': ',fval(1:SIZE)
   ```

   Line 421:

   ```
   write(*,N_Fmt('(A20,<size>I10)',SIZE))trim(argname)//': ',ival(1:SIZE)
   ```

   Line 494:

   ```
   write(*,N_Fmt('(A20,<size>L10)',SIZE))trim(argname)//': ',cval(1:SIZE)
   ```

   Line 567:

   ```
   write(*,N_Fmt('(A20,<size>A10)',SIZE))trim(argname)//': ',sval(1:SIZE)
   ```

   NOTE

   After performing **Step 19.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 20 Modify the **mod_scal.F**, **internal_step.F**, **adv_t.F**, and **adv_s.F** files. Modify the statement that starts with **IF(BACKWARD_ADVECTION** in the file. Change **==** to **.eqv** and **/=** to **.neqv**.

```
sed -i 's/\/=\.TRUE\./\.neqv\.\.TRUE\./g' mod_scal.F
sed -i 's/==\.TRUE/\.eqv\.\.TRUE/g' internal_step.F
sed -i 's/==\.FALSE\./\.eqv\.\.FALSE\./g' adv_t.F
sed -i 's/==\.FALSE\./\.eqv\.\.FALSE\./g' adv_s.F
```

Step 21 Edit the **mod_utils.F** file.

1. Open the **mod_utils.F** file.
   ```
   vi mod_utils.F
   ```
2. Press **i** to enter the edit mode and change the **ISNAN** function to **IEEE USENAN**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 22 Compile and install the FVCOM main program.

```
make
```

After the installation is successful, you can run the **ll** command to view the generated executable file **fvcom** in the current directory.

----End

1.42.7 Running and Verification

Procedure (Using Estuary as an Example)

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test directory.

```
mkdir -p /path/to/FVCOM/test
```

Step 3 Go to the test directory.

```
cd /path/to/FVCOM/test
```

Step 4 Copy the test case to the current directory.

```
cp -r /path/to/FVCOM/FVCOM4.1/Examples/Estuary/ ./
```

Step 5 Go to the **run** directory.

```
cd ./Estuary/run
```

Step 6 Edit the test case input file.

1. Open the test case input file.
   ```
   vi tst_run.nml
   ```
2. Press **i** to enter the edit mode and add single quotation marks to **tst_grd.dat**.
   ```
   GRID_FILE = 'tst_grd.dat'
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Go to the **/path/to/FVCOM/test/Estuary/tstinp** directory and run the following command:

```
dos2unix tst_grd.dat
```

Step 8 Go to the **/path/to/FVCOM/test/Estuary/run** directory and run the following command:

```
time mpirun -n 24 --allow-run-as-root --bind-to core /path/to/FVCOM/FVCOM4.1/FVCOM_source/
fvcom --casename=tst
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 View the current directory. After the command is executed successfully, an **.nc** result file is generated.

ll

```
-rw-r--r--. 1 zju zju 4.0K Nov 19 19:02 tst_0001.nc
-rw-------. 1 zju zju  11K Nov 16 23:02 tst_run.nml
```

----End

1.42.8 Troubleshooting

1.42.8.1 Programs Not Supported by the BiSheng Compiler

Symptom

When the BiSheng Compiler is used for compilation, some of the programs are not supported by the compiler.

Key Process and Cause Analysis

The program versions are too early or the programs have non-standard versions of libraries.

Conclusion and Solution

Replace the libraries with the standard function libraries. If the ISNAN function is not recognized by the BiSheng Compiler, replace it with the standard library **ieee_is_nan** of F 2003.

1.42.8.2 isnan Function Reports an Error

Symptom

The isnan function reports an error.

Key Process and Cause Analysis

None

Conclusion and Solution

When locating the error function, no result is found if you enter **isnan**. The source program function name is **ISNAN**. If you enter **ISNAN**, the function is located successfully.

1.42.8.3 Version Conflict

Symptom

A version conflict occurs.

 Copyright © Huawei Technologies Co., Ltd.

Key Process and Cause Analysis

A version conflict occurs.

Conclusion and Solution

When running a program on a machine that contains multiple versions of compilers and MPIs, use absolute paths to specify the required compilers and MPIs.

1.43 gadget2 2.3.5 Porting Guide (Kylin V10)

1.43.1 Introduction

Gadget is the Globally applicable Area Disaggregated General Ecosystem Toolbox.

Gadget is a statistical model of marine ecosystems. Gadget models can be both very data- and computationally-intensive. Various scripts have been written to extract data from data warehouses using mfdb, a specialized R-package, that can handle the complexities of the disparate datasets that can be used by Gadget when estimating model parameters. To speed up processing time, a parallel processing optimizer (called Paramin) has been developed. In addition to these a specialized R-package, Rgadget, has been developed to aid in the development of Gadget models by providing tools to estimate likelihood weights, defining model skeletons, and model diagnostics and prognosis.

For more information, visit the **gadget2 page at GitHub**.

Programming language: C++

Brief description: Globally applicable Area Disaggregated General Ecosystem Toolbox.

Open source license: GPLv2

Recommended Software Version

gadget2 2.3.5

1.43.2 Environment Requirements

Hardware Requirements

Table 1-220 lists the hardware requirements.

Table 1-220 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 1-221 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz
gadget2	2.3.5	https://github.com/gadget-framework/gadget2/archive/refs/tags/v2.3.5.tar.gz

OS Requirements

Table lists the OS requirements.

Table 1-222 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

1.43.3 Paths for Software Porting

This chapter lists the software installation paths involved in the gadget2 software porting.

Table 1-223 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/gadget2	Installation path of gadget2	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

1.43.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-224 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

1.43.5 Obtaining the Source Code

Procedure

Step 1 Download the gadget2 installation package **gadget2-2.3.5.tar.gz**.

https://github.com/gadget-framework/gadget2/archive/refs/tags/v2.3.5.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/gadget2* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.43.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/gadget2
```

Step 3 Decompress the installation package.

```
tar -xvf gadget2-v2.3.5.tar.gz
cd gadget2-2.3.5
```

Step 4 Perform the compilation and installation.

```
make && make install
```

----End

1.43.7 Running and Verification

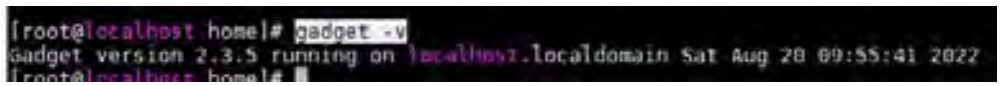
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify the version.

```
gadget -v
```

If the following information is displayed, the installation is successful:



----End

1.44 GrADS 2.2.1 Porting Guide (CentOS 8.2)

1.44.1 Introduction

The Grid Analysis and Display System (GrADS) is an interactive desktop tool that is used for easy access, manipulation, and visualization of earth science data. GrADS has two data models for handling gridded and station data. GrADS supports many data file formats, including binary (stream or sequential), GRIB (version 1 and 2), NetCDF, HDF (version 4 and 5), and BUFR (for station data). GrADS has been implemented worldwide on a variety of commonly used OSs and is freely distributed over the Internet.

GrADS uses a 5-dimensional data environment: the four conventional dimensions (longitude, latitude, vertical level, and time) plus an optional 5th dimension for grids that is generally implemented but designed to be used for ensembles. Datasets are placed within the 5-D space by use of a data descriptor file. GrADS handles grids that are regular, non-linearly spaced, gaussian, or of variable resolution. Data from different datasets may be graphically overlaid, with correct spatial and time registration. Operations are executed interactively by entering

 Copyright © Huawei Technologies Co., Ltd.

Fortran-like expressions at the command line. A rich set of built-in functions are provided, but users may also add their own functions as external routines written in any programming language.

Data may be displayed using a variety of graphical techniques: line and bar graphs, scatter plots, smoothed contours, shaded contours, streamlines, wind vectors, grid boxes, shaded grid boxes, and station model plots. Graphics may be output in PostScript or image formats. GrADS provides geophysically intuitive defaults, but the user has the option to control all aspects of graphics output.

GrADS has a programmable interface (scripting language) that allows for sophisticated analysis and display applications. Use scripts to display buttons and dropmenus as well as graphics, and then take action based on user point-and-clicks. GrADS can be run in batch mode, and the scripting language facilitates using GrADS to do long overnight batch jobs.

Programming language: C

Brief description: An interactive desktop tool.

Recommended Software Version

GrADS 2.2.1

1.44.2 Environment Requirements

Hardware Requirements

Table 1-225 lists the hardware requirements.

Table 1-225 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 1-226 lists the OS requirements.

Table 1-226 OS requirements

Item	Version	Download URL
CentOS	8.2	https://www.centos.org/download/
Kernel	4.18.0-193.el8.aarch64	Included in the OS image

Software Requirements

Table 1-227 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-227 Software requirements

Item	Version	Download URL
GrADS	2.2.1	ftp://cola.gmu.edu/grads/2.2/grads-2.2.1-src.tar.gz
libgd	2.25	https://sourceforge.net/projects/opengrads/files/supplibs/2.2.1/supplibs-2.2.1.tar.gz
Libsx	2.05	https://sourceforge.net/projects/opengrads/files/supplibs/2.2.1/supplibs-2.2.1.tar.gz
Test case	-	ftp://cola.gmu.edu/grads/example.tar.gz

1.44.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GrADS software porting.

Table 1-228 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	For details, see "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/libgd	Installation path of libgd.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/libsx	Installation path of libsx.	
4	/path/to/grads	Installation path of GrADS.	

1.44.4 Configuring the Compilation Environment

1.44.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

The server must be connected to the Internet.

Configuration Process

Table 1-229 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing dependencies	See **1.44.4.2 Installing Dependencies**.
3	Installing libgd	See **1.44.4.3 Installing libgd**.
4	Installing libsx	See **1.44.4.4 Installing libsx**.

1.44.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y libtiff* libXaw-devel fontconfig* readline* udunits* gd gd-devel cairo* hdf5*
```

----End

1.44.4.3 Installing libgd

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the **supplibs** package.

```
tar xvf supplibs-2.2.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd supplibs-2.2.1/src/gd
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/libgd --build=aarch64-unknown-linux-gnu
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.44.4.4 Installing libsx

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory generated after the decompression.

```
cd supplibs-2.2.1/src/libsx
```

Step 3 Modify the configuration.

```
vim libsx_defs
```

Modify the **libsx_defs** file. Change the value of **XLIBS** in line 103 to the following value:

```
XLIBS = -L/path/to/libgd/lib  -lXpm -l$(XAWLIB) -lXmu -lXt -lX11
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/libsx --build=aarch64-unknown-linux-gnu
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

1.44.5 Obtaining the Source Code

Procedure

Step 1 Download the GrADS installation package **grads-2.2.1-src.tar.gz**.

ftp://cola.gmu.edu/grads/2.2/grads-2.2.1-src.tar.gz

Step 2 Use an SFTP tool to upload the GrADS installation package to the */path/to/grads* directory on the server.

----End

1.44.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package and go to the installation directory.

```
tar xvf grads-2.2.1-src.tar.gz
cd grads-2.2.1
```

Step 3 Perform the configuration.

```
./configure
mkdir lib
cp /path/to/libsx/lib/libsx.so ./lib
mkdir include
cp /path/to/libsx /include/libsx.h ./include
```

Step 4 Modify the configuration.

```
vim src/Makefile
```

 Copyright © Huawei Technologies Co., Ltd.

Change the value of **LIBS** in line 398 of **Makefile**.

```
LIBS=-lm -lreadline
```

Change the value of **gui_libs** in line 483 of **Makefile**.

```
gui_libs = -L/path/to/libsx/lib -lsx  -lXext -lXaw -lXpm -lXmu -lXt
```

```
vim src/grads.c
```

Delete **static** from line 51 in the **grads.c** file.

```
struct gacmn gcmn;
```

```
vim src/gauser.c
```

Modify line 2815 in the **gauser.c** file.

```
#if (USENETCDF == 1 || USEHDF == 1  || USEHDF5 ==1)
```

Step 5 Compile GrADS.

```
make && make install
```

Step 6 Recompile GrADS.

After the compilation and installation are complete (although no error is reported during the installation), the following error is reported when running GrADS: "grads: symbol lookup error: /home/ivan/WRF/lib/grads-2.2.1/lib/libgxdCairo.so: undefined symbol: gxdbkq."

The cause is that **libgxdXXX.so** is not linked to **libgradspy.so**. Therefore, it is necessary to link to **libgradspy.so**. (However, do not link to **libgradspy.so** in the initial compilation because it has not been generated.)

In the **grads-2.2.1/src/Makefile** file, change the value of **gd_ldflags** in line 475.

```
gd_ldflags= -L/path/to/grads-2.2.1/lib  -lgradspy
```

Perform the compilation and installation again.

```
make clean
cd lib
rm -rf libgxd*
rm -rf libgxp*
cd ..
make && make install
```

----End

1.44.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export PATH=/path/to/grads-2.2.1/bin:$PATH
export LD_LIBRARY_PATH=/path/to/grads-2.2.1/lib:$LD_LIBRARY_PATH
export GADDIR=/path/to/grads-2.2.1/data
```

 Copyright © Huawei Technologies Co., Ltd.

```
export GASCRP=/path/to/grads-2.2.1/src
export GAUDPT=/path/to/grads-2.2.1/udpt
```

Step 3 Create a configuration file.

```
vim /path/to/grads-2.2.1/udpt
```

Add the following content to the file:

```
# Type     Name     Full path to shared object file
# ----     ----     -------------------------------
gxdisplay  Cairo    /path/to/grads-2.2.1/lib/libgxdCairo.so
gxdisplay  X11      /path/to/grads-2.2.1/lib/libgxdX11.so
gxdisplay  gxdummy  /path/to/grads-2.2.1/lib/libgxdummy.so
*
gxprint    Cairo    /path/to/grads-2.2.1/lib/libgxpCairo.so
gxprint    GD       /path/to/grads-2.2.1/lib/libgxpGD.so
gxprint    gxdummy  /path/to/grads-2.2.1/lib/libgxdummy.so
*
#function   dothis   /home/username/grads/udp/dothis.so
```

Step 4 Decompress the test case file.

```
tar -zxvf example.tar.gz
```

Step 5 Run GrADS.

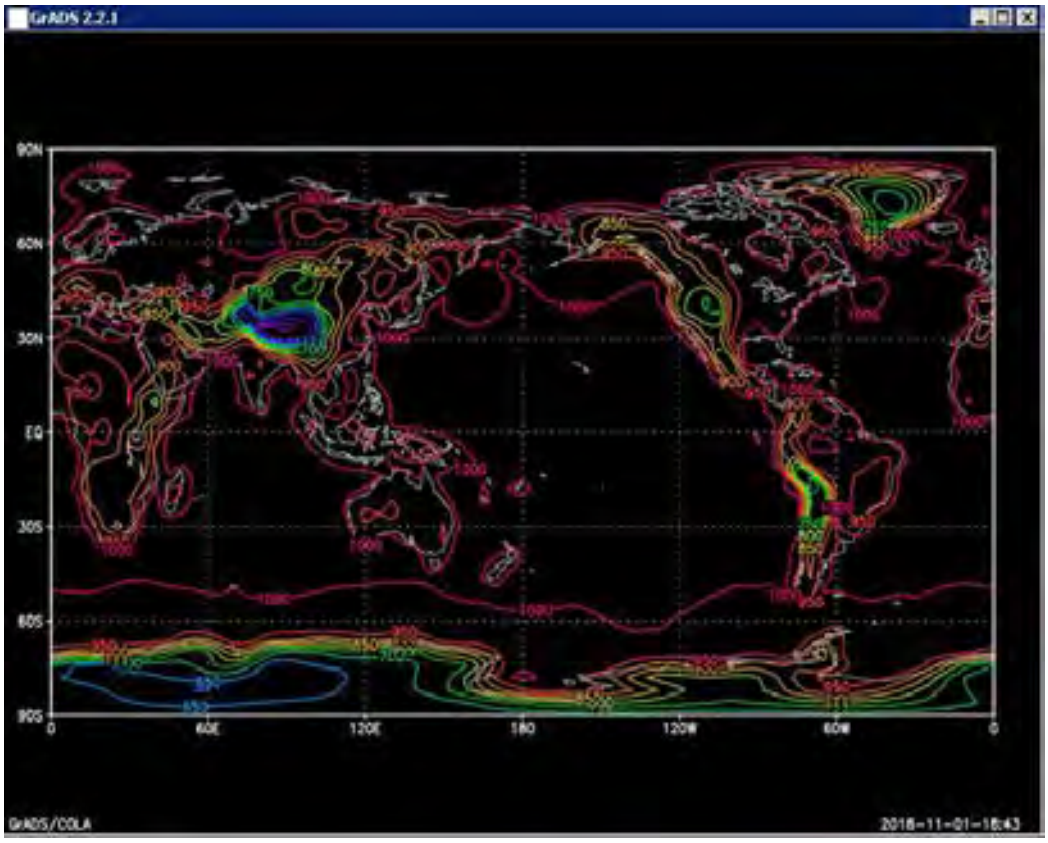
```
grads
```

After you log in to the system, the system asks you whether to use the Landscape mode. Press **Enter**.

In GrADS, run the following command to open the test file:

```
open model.ctl
query file
d ps
```

If the following information is displayed, the running is successful.



If the preceding information is displayed, GrADS is successfully installed.

----End

1.45 wgrib 1.8.2 Porting Guide (Kylin V10)

1.45.1 Introduction

wgrib is a program to manipulate, inventory and decode GRIB files. The program is known to work on machines ranging from 486 to Cray supercomputers.

For more information, visit the **wgrib website**.

Programming language: C

Brief description: A program to manipulate, inventory and decode GRIB files.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.8.2

1.45.2 Environment Requirements

Hardware Requirements

Table 1-230 lists the hardware requirements.

Table 1-230 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software requirements

Table 1-231 lists the software requirements.

Table 1-231 Software requirements

Item	Version	Download URL
wgrib	1.8.2	**https://ftp.cpc.ncep.noaa.gov/wd51we/wgrib/wgrib.tar.v1.8.2**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 1-232 lists the OS requirements.

Table 1-232 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

1.45.3 Paths for Software Porting

This section describes the software installation paths involved in the wgrib software porting.

Table 1-233 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/wgrib	Installation path of wgrib.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

1.45.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-234 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in User Guide (GCC for openEuler).

 Copyright © Huawei Technologies Co., Ltd.

1.45.5 Obtaining the Source Code

Procedure

Step 1 Download the installation package **wgrib.tar.v1.8.2**.

https://ftp.cpc.ncep.noaa.gov/wd51we/wgrib/wgrib.tar.v1.8.2

Step 2 Use an SFTP tool to upload the installation package to the */path/to/wgrib* directory on the server.

----End

1.45.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/wgrib
tar -xvf wgrib.tar.v1.8.2
```

Step 3 Perform the compilation.

```
make
```

----End

1.45.7 Running and Verification

Procedure

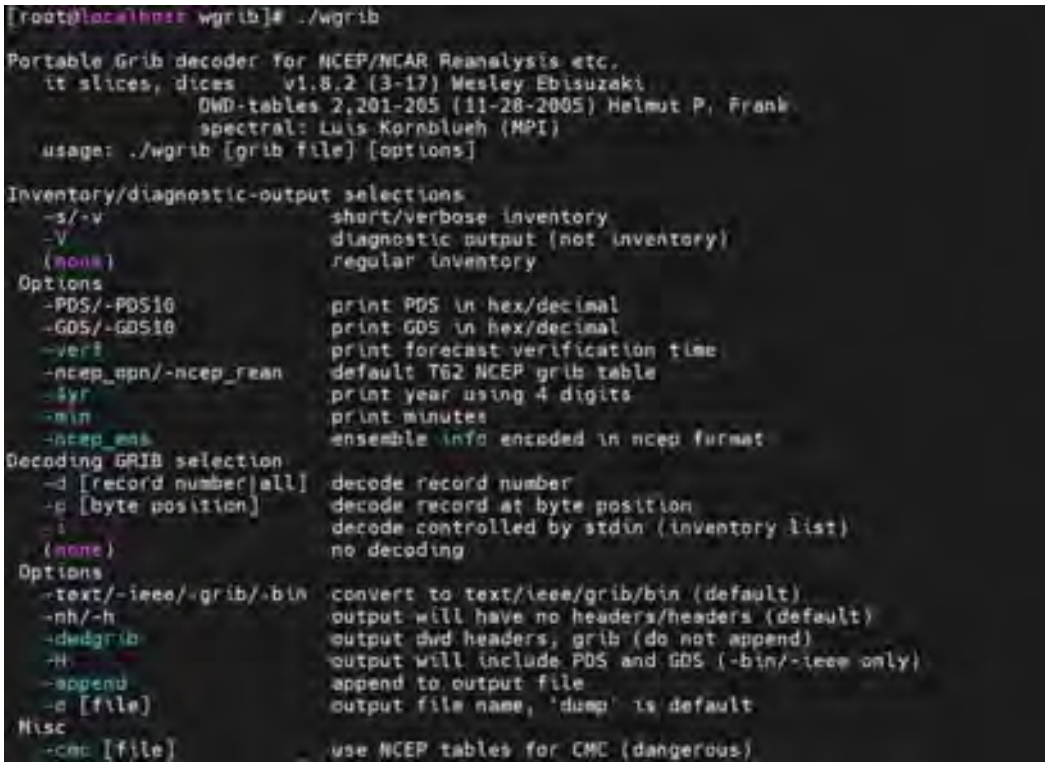
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/wgrib
./wgrib
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

1.45.8 Change History

Table 1-235

Date	Description
2023-04-28	This issue is the first official release.

1.46 wgrib2 3.1.1 Porting Guide (Kylin V10)

1.46.1 Introduction

wgrib2 is a processor for grib2 files. It is a utility and library for manipulating grib files. The utility was designed to be used to reduce the need for custom Fortran programs to read, write, and manipulate grib files.

For more information, visit the **wgrib2 website**.

Programming language: C

Brief description: wgrib2 is a processor for grib2 files. It is a utility and library for manipulating grib files.

Open source license: custom open source license

Recommended Software Version

3.1.1

1.46.2 Environment Requirements

Hardware Requirements

Table 1-236 lists the hardware requirements.

Table 1-236 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 1-237 lists the software requirements.

Table 1-237 Software requirements

Item	Version	Download URL
wgrib2	3.1.1	https://www.ftp.cpc.ncep.noaa.gov/wd51we/wgrib2/wgrib2.tgz.v3.1.1

OS Requirements

Table 1-238 lists the OS requirements.

Table 1-238 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

1.46.3 Paths for Software Porting

This chapter lists the software installation paths involved in the wgrib2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 1-239 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/wgrib2	Installation path of wgrib2.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

1.46.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-240 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See "Installing the BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

1.46.5 Obtaining the Source Code

Procedure

Step 1 Download the installation package **wgrib2.tgz.v3.1.1**.

https://www.ftp.cpc.ncep.noaa.gov/wd51we/wgrib2/wgrib2.tgz.v3.1.1

Step 2 Use an SFTP tool to upload the installation package to the */path/to/wgrib2* directory on the server.

----End

1.46.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/wgrib2
tar -xvf wgrib2.tgz.v3.1.1
cd wgrib2
```

Step 3 Modify the source code.

```
sed -i "860s/configure/configure --build=aarch64-unknown-linux-gnu/g" makefile
sed -i "936s/configure/configure --build=aarch64-unknown-linux-gnu/g" makefile
```

Step 4 Perform the compilation and installation.

```
make
make lib
```

----End

1.46.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/wgrib2/wgrib2
./wgrib2
```

If the following information is displayed, the verification is successful.

----End

1.46.8 Change History

Table 1-241

Date	Description
2023-04-28	This issue is the first official release.

1.47 WPS 3.9 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

1.47.1 Introduction

The Weather Research and Forecasting (WRF) Model is designed for fine-scale weather simulation and forecasting. It is one of major high-performance computing (HPC) application scenarios. The WRF Preprocessing System (WPS) is the preprocessing system of WRF.

For more information, visit the **WRF official website**.

Programming language: C/Fortran

Brief description: A weather research and forecasting model and its preprocessing system.

Open source license: custom open source license

Recommended Software Version

3.9

1.47.2 Environment Requirements

Hardware Requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Item	Version	Download URL
CentOS	8.2	**https://www.centos.org/download/**
Kernel	4.18.0-193.el8.aarch64	Included in the OS image

Software Requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**
HMPI	1.2.0	**https://github.com/kunpengcompute**
HDF5	1.8.20	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.20/src/hdf5-1.8.20.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
PnetCDF	1.11.2	https://parallel-netcdf.github.io/Release/pnetcdf-1.11.2.tar.gz
NetCDF-C	4.7.4	https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.4.tar.gz
NetCDF-Fortran	4.5.2	https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.5.2.tar.gz
WRF	3.9	https://github.com/wrf-model/WRF/archive/refs/tags/V3.9.tar.gz
WPS	3.9	https://github.com/wrf-model/WPS/archive/refs/tags/v3.9.tar.gz

1.47.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WRF software porting.

Table 1-242 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bisheng_compiler	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/install/HMPI	Installation path of HMPI.	
3	/path/to/install/HDF5	Installation path of HDF5.	
4	/path/to/install/PNETCDF	Installation path of PnetCDF.	
5	/path/to/install/NETCDF	Installation path of NetCDF-C and NetCDF-Fortran.	
6	/path/to/install/WRF	Installation path of WRF.	

1.47.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

1.47.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-243 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See "Installing the BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
2	Installing HMPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
3	Installing HDF5	See **1.47.4.2 Installing HDF5**.
4	Installing PnetCDF	See **1.47.4.3 Installing PnetCDF**.
5	Installing NetCDF-C	See **1.47.4.4 Installing NetCDF-C**.
6	Installing NetCDF-Fortran	See **1.47.4.5 Installing NetCDF-Fortran**.

1.47.4.2 Installing HDF5

Procedure

Step 1 Download the HDF5 source package.

https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.20/src/hdf5-1.8.20.tar.gz

Step 2 Upload the source package to its installation directory on the server, for example, **/path/to/download/hdf5**.

Step 3 Go to the source package directory.

```
cd /path/to/download/hdf5
```

Step 4 Decompress the installation package.

```
tar -zxvf hdf5-1.8.20.tar.gz
```

Step 5 Switch to the directory of the **configure** script.

```
cd hdf5-1.8.20
```

Step 6 Install HDF5.

```
./configure --prefix=/path/to/install/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared  CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CFLAGS=-fPIC CXXFLAGS=-fPIC FCFLAGS=-fPIC
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Modify the **libtool** file in the current directory.

```
vi libtool
```

In line 10118, change **wl=""** to **wl="-Wl,"**.

In line 10267, change **wl=""** to **wl="-Wl,"**.

The line numbers may vary according to each HDF5 version. Search for **wl=""** to determine the line numbers.

Step 8 Perform the compilation and installation.

```
make -j 16
make install
```

Step 9 Set the environment variables.

```
export PATH=/path/to/install/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/HDF5/lib:$LD_LIBRARY_PATH
```

----End

1.47.4.3 Installing PnetCDF

Procedure

Step 1 Download the PnetCDF source package.

https://parallel-netcdf.github.io/Release/pnetcdf-1.11.2.tar.gz

Step 2 Upload the source package to its installation directory on the server, for example, **/path/to/download/pnetcdf**.

Step 3 Go to the source package directory.

```
cd /path/to/download/pnetcdf
```

Step 4 Decompress the installation package.

```
tar -zxvf pnetcdf-1.11.2.tar.gz
```

Step 5 Switch to the directory of the **configure** script.

```
cd pnetcdf-1.11.2
```

Step 6 Install PnetCDF.

```
./configure --prefix=/path/to/install/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 7 Perform the compilation and installation.

```
make -j 16
make install
```

Step 8 Set the environment variables.

```
export PATH=/path/to/install/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

1.47.4.4 Installing NetCDF-C

Procedure

Step 1 Download the NetCDF-C source package.

https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.4.tar.gz

Step 2 Upload the source package to its installation directory on the server, for example, **/path/to/download/netcdf-c**.

Step 3 Go to the source package directory.

```
cd /path/to/download/netcdf-c
```

Step 4 Decompress the installation package.

```
tar -zxvf netcdf-c-4.7.4.tar.gz
```

Step 5 Switch to the directory of the **configure** script.

```
cd netcdf-c-4.7.4
```

Step 6 Install NetCDF-C.

```
./configure --prefix=/path/to/install/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/install/HDF5/include -I/path/to/install/PNETCDF/include" LDFLAGS="-L/path/to/install/HDF5/lib -L/path/to/install/PNETCDF/lib -Wl,-rpath=/path/to/install/HDF5/lib -Wl,-rpath=/path/to/install/PNETCDF/lib" CFLAGS="-L/path/to/install/HDF5/lib -L/path/to/install/PNETCDF/lib -I/path/to/install/HDF5/include -I/path/to/install/PNETCDF/include"
```

Step 7 Perform the compilation and installation.

```
make -j 16
make install
```

Step 8 Set the environment variables.

```
export PATH=/path/to/install/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.47.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Download the NetCDF-Fortran source package.

https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.5.2.tar.gz

Step 2 Upload the source package to its installation directory on the server, for example, **/path/to/download/netcdf-fortran**.

Step 3 Go to the source package directory.

```
cd /path/to/download/netcdf-fortran
```

Step 4 Decompress the installation package.

```
tar -zxvf netcdf-fortran-4.5.2.tar.gz
```

Step 5 Switch to the directory of the **configure** script.

```
cd netcdf-fortran-4.5.2
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Install NetCDF-Fortran.

```
./configure --prefix=/path/to/install/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS=" -fPIC  -I/path/to/install/HDF5/include -I/path/to/install/NETCDF/include" LDFLAGS="-L/path/to/install/HDF5/lib -L/path/to/install/NETCDF/lib -Wl,-rpath=/path/to/install/HDF5/lib -Wl,-rpath=/path/to/install/NETCDF/lib" CFLAGS=" -fPIC -L/path/to/install/HDF5/lib -L/path/to/install/NETCDF/lib -I/path/to/install/HDF5/include -I/path/to/install/NETCDF/include" CXXFLAGS=" -fPIC -L/path/to/install/HDF5/lib -L/path/to/install/NETCDF/lib -I/path/to/install/HDF5/include -I/path/to/install/NETCDF/include" FCFLAGS=" -fPIC -L/path/to/install/HDF5/lib -L/path/to/install/NETCDF/lib -I/path/to/install/HDF5/include -I/path/to/install/NETCDF/include"
```

Step 7 Modify the **libtool** file in the current directory.

```
vi libtool
```

In line 11787, change **wl=""** to **wl="-Wl,"**.

In line 11939, change **wl=""** to **wl="-Wl,"**.

The line numbers may vary according to each NetCDF-Fortran version. Search for **wl=""** to determine the line numbers.

Step 8 Perform the compilation and installation.

```
make
make install
```

----End

1.47.5 Obtaining the Source Code

Procedure

Step 1 Download the WRF source code.

https://github.com/wrf-model/WRF/archive/refs/tags/V3.9.tar.gz

Step 2 Upload the source package to the */path/to/WRF* directory on the server.

Step 3 Download the WPS source code.

https://github.com/wrf-model/WPS/archive/refs/tags/v3.9.tar.gz

Step 4 Upload the source package to the */path/to/WPS* directory on the server.

----End

NOTE

In this section, the example installation directory is **/path/to/wrf**. WRF and WPS have no special requirements for the installation directory. You are advised to deploy them in a shared directory. Replace **/path/to/wrf** with the actual one.

1.47.6 Compilation and Installation

1.47.6.1 Compiling and Installing WRF

Procedure

Step 1 Decompress the WRF installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
tar -zxvf WRF-3.9.tar.gz
```

Step 2 Switch to the WRF source code directory.

```
cd WRF-3.9
```

Step 3 Configure the pre-compilation environment.

```
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
export NETCDF=/path/to/install/NETCDF
export HDF5=/path/to/install/HDF5
export PNETCDF=/path/to/install/PNETCDF
export CPPFLAGS="-I$HDF5/include -I$PNETCDF/include -I$NETCDF/include"
export LDFLAGS="-L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib -lnetcdf -lnetcdff -lpnetcdf -lhdf5_hl -lhdf5 -lz"
```

Step 4 Modify the **arch/configure_new.defaults** file.

```
vim arch/configure_new.defaults
```

Add the following content below **#insert new stanza here**:

```
################################################### #########
#ARCH   Linux   aarch64,clang HYPERMPI#serial smpar dmpar dm+sm
DESCRIPTION     =       CLANG ($SFC/$SCC)
DMPARALLEL      =       1
OMPCPP          =       -D_OPENMP
OMP             =       -fopenmp
OMPCC           =       -fopenmp
SFC             =       flang
SCC             =       clang
CCOMP           =       clang
DM_FC           =       mpif90
DM_CC           =       mpicc -DMPI2_SUPPORT
FC              =       CONFIGURE_FC
CC              =       CONFIGURE_CC
LD              =       $(FC)
RWORDSIZE       =       CONFIGURE_RWORDSIZE
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR  -DWRF_USE_CLM
CFLAGS_LOCAL    =       -mcpu=native -w -O3 -c -march=armv8.2-a -I/usr/include/tirpc/
LDFLAGS_LOCAL   = -ltirpc
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 -funroll-loops -march=armv8.2-a
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       -g # -fbacktrace -ggdb-fcheck=bounds,do,mem,pointer -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-0
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       -mcpu=native $(OMP) $(FCBASEOPTS_NO_G)
MODULE_SRCH_FLAG =
TRADFLAG        =       -traditional
CPP             =       /lib/cpp -P
AR              =       ar
ARFLAGS         =       ru
M4              =       m4 -G
RANLIB          =       ranlib
RLFLAGS         =
CC_TOOLS        =       $(SCC)
################################################################
```

Step 5 Create a configuration file.

```
./configure
```

Type **4** and press **Enter**.

Type **1** and press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the compilation and installation.

```
./compile em_real 2>&1 | tee -a compile.log
```

Step 7 Verify the installation.

```
ls main
```

If executable files such as **wrf.exe** are generated in the command output, the compilation is successful.

----End

1.47.6.2 Compiling and Installing WPS

Procedure

Step 1 Decompress the WPS installation package.

```
tar -zxvf WPS-3.9.tar.gz
```

Step 2 Switch to the WPS source code directory.

```
cd WPS-3.9
```

Step 3 Configure the pre-compilation environment.

```
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
export NETCDF=/path/to/install/NETCDF
export HDF5=/path/to/install/HDF5
export PNETCDF=/path/to/install/PNETCDF
export CPPFLAGS="-I$HDF5/include -I$PNETCDF/include -I$NETCDF/include"
export LDFLAGS="-L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib -lnetcdf -lnetcdff -lpnetcdf -lhdf5_hl -lhdf5 -lz"
unset MPI MPI_DIR MPI_INC
export WRF_DIR=/path/to/install/WRF
```

Step 4 Modify the **arch/configure.defaults** file.

```
vi arch/configure.defaults
```

Modify lines 128 to 136. See the red lines for the modification.

Step 5 Perform the compilation.

```
echo 3 | ./configure
```

Step 6 Modify the **configure.wps** file.

```
vi configure.wps
```

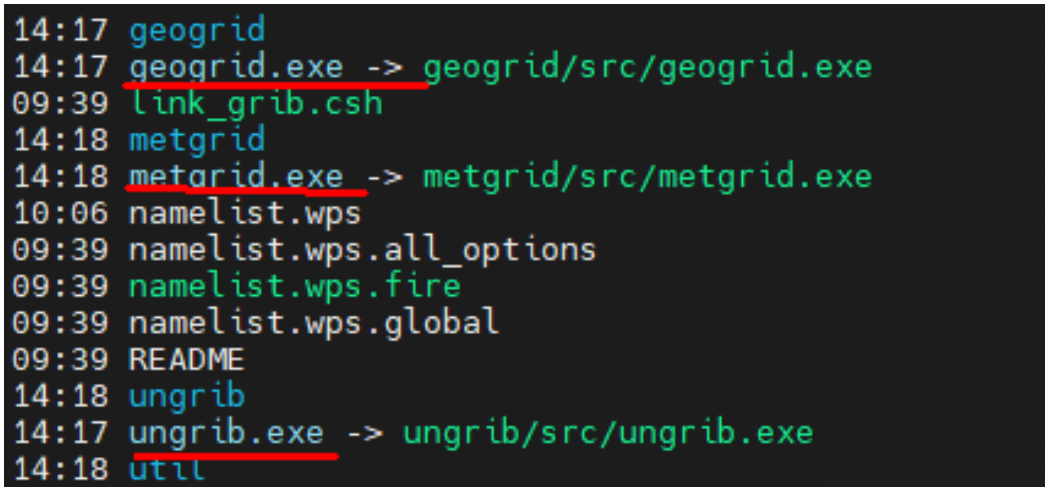
Change **WRF_DIR** to the WRF path and add **-lomp** to the end of **WRF_LIB**.

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Perform the compilation.

```
./compile
```

The following information is displayed. After the compilation is successful, three **.exe** executable files are generated in the current directory.



----End

1.47.7 Running and Verification

Procedure

Step 1 Copy **wrf.exe** in the */path/to/WRF/main* directory to the test directory.

```
cp /path/to/WRF/main/wrf.exe /path/to/WRFTEST/
```

Step 2 Copy the test cases to the test directory.

```
cp /path/to/TESTCASE /path/to/WRFTEST/
```

Step 3 Run WRF.

```
mpirun –allow-run-as-root -np 128 ./wrf.exe
```

Check the generated **rsl** file.

----End

 Copyright © Huawei Technologies Co., Ltd.

1.47.8 Change History

Table 1-244

Date	Description
2023-05-04	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

2 Manufacturing

2.1 OpenFOAM 1906 Porting Guide (CentOS 7.6)
2.2 SU2 Porting Guide (CentOS 7.6)
2.3 PALABOS 2.1r0 Porting Guide (CentOS 7.6)
2.4 deal.II 9.1.1 Porting Guide (CentOS 7.6)
2.5 AMG 1.2 Porting Guide (CentOS 7.6)
2.6 Code_Saturne 6.0.2 Porting Guide (CentOS 7.6)
2.7 FDS 6.7.4 Porting Guide (CentOS 7.6)
2.8 Elmer 8.4 Porting Guide (CentOS 7.6)
2.9 Code_Aster 14.6.0 Porting Guide (CentOS 7.6)
2.10 OpenFOAM 1906 Porting Guide (openEuler 21.03)
2.11 Code_Aster 14.6.0 Porting Guide (openEuler 21.03)
2.12 OpenLB 1.4-0 Porting Guide (GCC for openEuler)
2.13 OpenLB 1.4-0 Porting Guide (BiSheng Compiler)
2.14 Kokkos 3.5.00 Porting Guide (Kylin V10)
2.15 Pencil Code 2021.02.20 Porting Guide (Kylin V10)
2.16 SPARTA 7Jan2022 Porting Guide (Kylin V10)

2.1 OpenFOAM 1906 Porting Guide (CentOS 7.6)

2.1.1 Introduction

Open Source Field Operation and Manipulation (OpenFOAM) is a physical field computing software developed by OpenCFD, a subsidiary of ESI. The software analyzes fluid, heat transfer, molecular dynamics, electromagnetic fluid and solid stress, and provides a visualized interface for operations from grid division to post-

 Copyright © Huawei Technologies Co., Ltd.

processing. It supports data pre-processing, post-processing, and solver customization, and is often used for computational fluid dynamics (CFD).

There are two versions of the OpenFOAM installation package provided on the official OpenFOAM website (https://openfoam.org/):

1. OpenFOAM-dev: Current development source code package of OpenFOAM, which includes the source code of the next major version of OpenFOAM.
2. OpenFOAM-7: Compilation source code on Linux distributions, which includes all latest patches. The number indicates the major version. For example, the latest major version is OpenFOAM-7.

For more information about OpenFOAM, visit the **OpenFOAM official website**.

Programming language: C++

Brief description: distributed memory object caching system

Open-source protocol: GNU general public license

NOTE

For OpenFOAM v1906 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port OpenFOAM to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

OpenFOAM v1906

2.1.2 Environment Requirements

Hardware Requirements

Table 2-1 lists the hardware requirements.

Table 2-1 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
Network	1 x GE Ethernet port
Computing network	1 x single-port EDR/HDR IB NIC

OS Requirements

Table 2-2 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-2 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0	Included in the OS image

2.1.3 Paths for Software Porting

This chapter lists the software installation paths involved in the OpenFOAM software porting.

Table 2-3 Paths for software porting

No.	Software Installation Path	Usage	Description
1	/path/to/OPENFOAM	Installation path of OpenFOAM	This installation path is only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

2.1.4 Configuring the Compilation Environment

2.1.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-4 Configuration process

No.	Configuration Item	Description
1	Setting Up the Basic Environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing ClusterShell	See **2.1.4.2 Installing ClusterShell**.

 Copyright © Huawei Technologies Co., Ltd.

2.1.4.2 Installing ClusterShell

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to download the ClusterShell RPM installation package **clustershell-1.8.2-1.el7.noarch.rpm** and its dependency **python2-clustershell-1.8.2-1.el7.noarch.rpm**:

```
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/c/clustershell-1.8.2-1.el7.noarch.rpm
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/p/python2-clustershell-1.8.2-1.el7.noarch.rpm
```

Step 3 Run the following commands to install ClusterShell:

```
yum install python-setuptools –y
rpm -ivh python2-clustershell-1.8.2-1.el7.noarch.rpm clustershell-1.8.2-1.el7.noarch.rpm
```

NOTE

ClusterShell is installed to clear the cache of each node when multiple nodes are running. If only one node is running, you do not need to install ClusterShell.

----End

2.1.5 Obtaining the Source Code

Step 1 Download the OpenFOAM installation source code package **OpenFOAM-v1906.tgz** and the auxiliary third-party software installation package **ThirdParty-v1906.tgz**.

URLs:

https://sourceforge.net/projects/openfoam/files/v1906/OpenFOAM-v1906.tgz

https://sourceforge.net/projects/openfoam/files/v1906/ThirdParty-v1906.tgz

Step 2 Use an SFTP tool to copy the OpenFOAM source code package and auxiliary third-party software installation package to the */path/to/OPENFOAM* directory on the server.

----End

2.1.6 Compiling and Installing OpenFOAM

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the system dependency package:

```
yum install -y zlib-devel texinfo gstreamer-plugins-base-devel libXext-devel libGLU-devel libXt-devel libXrender-devel libXinerama-devel libpng-devel libXrandr-devel libXi-devel libXft-devel libjpeg-turbo-devel libXcursor-devel readline-devel ncurses-devel mpfr-devel gmp-devel libmpc*
```

Step 3 Run the following commands to go to the directory of the installation package, and decompress the OpenFOAM source code package and third-party software installation package:

```
cd /path/to/OPENFOAM
tar zxvf OpenFOAM-v1906.tgz
tar zxvf ThirdParty-v1906.tgz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to edit the OpenFOAM installation configuration file:

1. Open the OpenFOAM installation parameter file.
   vi */path/to/OPENFOAM*/**OpenFOAM-v1906/etc/bashrc**
2. Modify the compiler and MPI settings in the OpenFOAM installation configuration file.
   ```
   export WM_COMPILER_TYPE=system
   export WM_COMPILER=Gcc
   export WM_MPLIB=SYSTEMOPENMPI
   ```

   NOTE

   The three parameters indicate that the GNU compiler and Open MPI installed in the system are used for the compilation and installation. If you need to use another compiler or MPI, modify the parameters as prompted.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 5 Run the following commands to modify the compilation parameters:

```
sed -i 's/\-O3/\-O3 \-march=armv8-a/g' /path/to/OPENFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Gcc/c++Opt
sed -i 's/\-O3/\-O3 \-march=armv8-a/g' /path/to/OPENFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Gcc/cOpt
```

Step 6 Run the following command to make the OpenFOAM configuration file **bashrc** take effect:

```
source /path/to/OPENFOAM/OpenFOAM-v1906/etc/bashrc
No completion added for /path/to/OPENFOAM/OpenFOAM-v1906/platforms/linuxARM64GccDPInt32Opt/bin
... incorrect platform, or not yet compiled?
```

Step 7 Run the following commands to perform compilation and installation:

```
cd /path/to/OPENFOAM/OpenFOAM-v1906/
./Allwmake -j 16 -s -k -q
```

NOTE

- **-j** indicates that all cores are used for the compilation. You can also specify the number of cores, for example, **-j 8**.
- **-s** indicates that log printing is disabled.
- **-k** indicates that errors during compilation are ignored.
- **-q** indicates that subdirectories are collected to schedule and accelerate the compilation.
- This step takes approximately 2.5 hours. After the step is complete, the **bin** and **lib** directories are generated in the */path/to/OPENFOAM*/**platforms/linuxARM64GccDPInt32Opt** directory. The executable files and library files required for running are available in the two directories.

----End

2.1.7 Running and Verifying OpenFOAM

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands on PuTTY to load environment variables:

```
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

NOTE

This step declares the MPI environment variables. If the MPI environment variables have been declared, skip this step.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to create the **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 4 Run the following commands to load the OpenFOAM application:

```
Node_list=$(cat /path/to/HOSTFILE | xargs | tr ' ' ',')
clush -w ${Node_list} "cp /root/.bashrc /root/.bashrc_bak;echo source /path/to/OPENFOAM/
OpenFOAM-v1906/etc/bashrc >> /root/.bashrc"
```

Step 5 Run the following command to copy the case file to any location in the computing environment:

```
cp -r /path/to/OPENFOAM/OpenFOAM-v1906/tutorials/incompressible/pisoFoam/LES/motorBike/
motorBike ./
```

Step 6 Run the following command to modify the **RunFunctions** file:

1. Open **RunFunctions**.

   vi */path/to/OPENFOAM*/**OpenFOAM-v1906/bin/tools/RunFunctions**

2. Press **i** to go to the insert mode.
   - Multi-node:

     Add **--allow-run-as-root -x PATH -x LD_LIBRARY_PATH --hostfile** */path/to/HOSTFILE* to the end of the **mpirun** command, save the modification, and exit.

     Before the modification:

     ```
     $mpirun -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
     ```

     After the modification:

     ```
     $mpirun --allow-run-as-root -x PATH -x LD_LIBRARY_PATH --hostfile /path/to/HOSTFILE -n
     $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
     ```

   - Single-node:

     Add **--allow-run-as-root** to the end of the **mpirun** command.

     Before the modification:

     ```
     $mpirun -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
     ```

     After the modification:

     ```
     $mpirun  --allow-run-as-root -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
     ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 7 Run the following command to modify the configuration file:

```
sed -ri "s/\(20 8 8\)/\(64 12 8\)/" motorBike/system/blockMeshDict
```

- Two-node scenario:

  ```
  sed -ri "s/8/256/" motorBike/ system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  sed -ri "s/\(4 2 1\)/\(32 4 2\)/"  motorBike/system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  ```

- Single-node scenario:

  ```
  sed -ri "s/8/128/" motorBike/ system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  sed -ri "s/\(4 2 1\)/\(16 4 2\)/"  motorBike/system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  ```

NOTE

- If there are two nodes and 128 tasks are concurrently executed on each node, then 128 x 2 = 256, 32 x 4 x 2 = 256.
- If there is only one node and 128 tasks are concurrently executed on the node, then 128 x 1 = 128, 16 x 4 x 2 = 128.
- You can run the **lscpu |grep ^CPU\(s\)** command to view the number of concurrent tasks (128) on the single node. You can also change the number based on the actual configuration of the node.

Step 8 Run the following commands to go to the test case folder and execute the calculation script:

```
cd motorBike
./Allrun
```

During the computing, the following information is displayed:

```
Running blockMesh on /hpcstore/ysf/openfoam-test/1020_motorBike
Running decomposePar on /hpcstore/ysf/openfoam-test/1020_motorBike
Running snappyHexMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
Restore 0/ from 0.orig/ for processor directories
Running renumberMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
Running potentialFoam (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
Running checkMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
Running simpleFoam (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
```

Step 9 After the calculation is complete and the system exits properly, a calculation log file is generated. You can run the following command to view the log file:

```
vi log.simpleFoam
forceCoeffs forceCoeffs1 execute:
   Coefficients
      Cd       : 0.413353     (pressure: 0.398578     viscous: 0.0147753)
      Cs       : 0.0100002    (pressure: 0.00999416   viscous: 6.01357e-06)
      Cl       : 0.0683827    (pressure: 0.0683764    viscous: 6.2544e-06)
      CmRoll       : -0.0101679       (pressure: -0.0102318   viscous: 6.39234e-05)
      CmPitch      : 0.15315 (pressure: 0.146948     viscous: 0.00620151)
      CmYaw        : 0.0118663 (pressure: 0.0120074    viscous: -0.000141087)
      Cd(f)    : 0.196509
      Cd(r)    : 0.216844
      Cs(f)    : 0.0168664
      Cs(r)    : -0.00686619
      Cl(f)    : 0.187341
      Cl(r)    : -0.118959
ensightWrite ensightWrite write: ( k omega p U )
End

Finalising parallel run
```

----End

2.1.8 Troubleshooting

Problem 1: ".fatal error: boost/config/compiler/gcc.hpp: No such file or directory" Is Displayed During Compilation

Symptom

When Allwmake is run to compile the Computational Geometry Algorithms Library (CGAL), the compilation fails and ".fatal error: boost/config/compiler/gcc.hpp: No such file or directory" is displayed.

Possible Cause

The CGAL fails to be installed due to environment variable or CMake installation failure. If the error occurs during the installation of a module of OpenFOAM, you only need to install the module again.

Procedure

Step 1 Delete the installed executable file and path.

```
rm -rf /path/to/OPENFOAM/OpenFOAM-v1906/platform/arch/boost_1_64_0
```

Step 2 Install the script independently when running **Thirdparty-v1906**.

```
cd /path/to/OPENFOAM/Thirdparty-v1906
./makeCGAL
```

----End

Problem 2: "*esmump.h* No such file or directory" Is Displayed During Compilation

Symptom

When Allwmake is run to compile Scotch, the compilation fails and "*esmump.h* No such file or directory" is displayed.

Possible Cause

An error occurs during script running. As a result, esmump fails to be installed.

Procedure

Step 1 Go to the Scotch directory and manually compile esmumps.

```
cd /path/to/OPENFOAM/ThirdParty-v1906/scotch_6.0.6/src
make esmumps
```

Step 2 Run the Allwmake script again to continue the compilation and installation.

```
foam
./Allwmake -j
```

----End

Problem 3: " cannot stat `../bin/d[agm]*'" Is Displayed During Compilation

Symptom

When Allwmake is run to compile Scotch, the compilation fails and the following message is displayed:

cp: cannot stat `../bin/d[agm]*': No such file or directory

make: [install] Error 1 (ignored)

Possible Cause

This problem is not caused by **d[agm]**. Generally, this problem occurs when the Scotch directory is not cleared before the installation program is rerun. You can run the **make distclean** command in the Scotch source code directory and then manually install the Scotch program.

Procedure

 Copyright © Huawei Technologies Co., Ltd.

Step 1 Delete the directory generated by the failed installation.

NOTE

linuxARM64GccDPInt32 is named based on the installation environment architecture and compiler. You need to change it to the actual path.

```
rm -rf /path/to/OPENFOAM/ThirdParty-v1906/platforms/linuxARM64GccDPInt32/scotch_6.0.6/
```

Step 2 Go to the Scotch source-code directory and manually compile and install the software.

NOTE

To manually install Scotch, you need to set the installation path to the installation path that is deleted previously.

```
make scotch
make esmumps
make ptscotch
make prefix=/path/to/OPENFOAM/ThirdParty-v1906/platforms/linuxARM64GccDPInt32/scotch_6.0.6/
install
```

Step 3 Run the Allwmake script again to continue the compilation and installation.

```
foam
./Allwmake -j
```

----End

Problem 4: The Case Cannot Stop Properly and "ls: cannot access processor*: No such file or directory" Is Displayed

Symptom

Generally, case running takes several minutes. If the case is stopped within one minute, open the log file to check whether the computing is running properly. If the log file contains the error information, "mpirun has detected an attempt to run as root." is displayed.

Possible Cause

When running the **mpirun** command as the **root** user, you need to add a parameter.

Procedure

Step 1 Modify the **RunFunctions** file.

```
vi /path/to/OPENFOAM/OpenFOAM-v1906/bin/tools/RunFunctions
```

Step 2 Press **i** to enter the insert mode and add **--allow-run-as-root** to the end of the **mpirun** command.

Step 3 Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 4 Run the test case again.

```
./Allclean
./Allrun
```

----End

 Copyright © Huawei Technologies Co., Ltd.

2.2 SU2 Porting Guide (CentOS 7.6)

2.2.1 Introduction

SU2 is a high-precision partial differential equation solver developed by the Department of Aeronautics and Astronautics of Stanford University. Different from CFL3D of NASA, SU2 is an unstructured grid solver developed based on C++ (similar to OpenFOAM). It carries out not only serial computing but also parallel computing, and has the function of deformation grid (that is, dynamic grid) and grid adaptive function. To improve the computing efficiency, the code adopts multiple grids to accelerate the convergence.

SU2 has multiple spatial discrete formats and multiple flux limiters, which can be used to calculate the flow rate from low speed to high speed, it can also simulate multiple physical fields, multicomponent flow, combustion, aerodynamic noise, two-phase flow, magnetohydrodynamic, plasma flow, and optimization.

For more information, visit the **official SU2 website**.

Programming language: C++

Brief description: a high-precision partial differential equation solver

Open source license: GPL 2.1

2.2.2 Environment Requirements

Hardware Requirements

Table 2-5 lists the hardware requirements.

Table 2-5 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-6 lists the software requirements.

Table 2-6 Software requirements

Item	Version	Download Address
SU2	6.0.1	**https://su2code.github.io/**
	7.0.4	**https://github.com/su2code/SU2/archive/v7.0.4.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
HDF5 (optional)	1.10.5	https://www.hdfgroup.org/downloads/hdf5/source-code/
Test case	Inviscid_Bump	https://github.com/su2code/su2code.github.io

OS Requirements

Table 2-7 lists the OS requirements.

Table 2-7 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

2.2.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SU2 software porting.

Table 2-8 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/SU2	Installation path of SU2	
4	/path/to/CASE	Path for storing the SU2 test case	

2.2.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

2.2.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-9 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **2.2.4.2 Installing HDF5**.

2.2.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the HDF5 installation package:

```
tar zxvf hdf5-1.10.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.5
```

Step 4 Run the following command to perform configuration:

```
CC=mpicc ./configure --prefix=/path/to/HDF5/ --enable-parallel
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following command to configure the compilation environment:

```
export HDF5_ROOT=/path/to/HDF5/
```

----End

2.2.5 SU2 6.0.1 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

2.2.5.1 Obtaining the Source Code

Procedure

Step 1 Download the SU2 installation package **SU2-6.0.1.tar.gz**.

URL: **https://su2code.github.io/**

Step 2 Use an SFTP tool to upload the SU2 installation package to the */path/to/SU2* directory on the server.

----End

2.2.5.2 Compiling and Installing SU2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install numpy.aarch64 -y
```

Step 3 Run the following commands to decompress the SU2 installation package:

```
tar zxvf SU2-6.0.1.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd SU2-6.0.1
```

Step 5 Run the following command to run the **bootstrap** script:

```
./bootstrap
```

Step 6 Run the following command to switch to the SU2 installation directory:

```
cd /path/to/SU2/
```

Step 7 Run the following command to perform configuration:

```
SU2-6.0.1/configure --prefix=/path/to/SU2/ CXXFLAGS="-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110" --enable-mpi --with-cc=mpicc --with-cxx=mpicxx --with-HDF5-lib=/path/to/HDF5/lib --with-HDF5-include=/path/to/HDF5/include
```

Step 8 Run the following commands to perform compilation and installation:

```
make -j 32
make install
```

Step 9 Run the following commands to add the environment variables:

```
export SU2_RUN=/path/to/SU2/bin
export SU2_HOME=/path/to/SU2
export PATH=$PATH:$SU2_RUN
export PYTHONPATH=$PYTHONPATH:$SU2_RUN
```

----End

2.2.6 SU2 7.0.4 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

2.2.6.1 Obtaining the Source Code

Procedure

Step 1 Download the SU2 installation package **SU2-7.0.4.tar.gz**.

https://github.com/su2code/SU2/archive/v7.0.4.tar.gz

Step 2 Use an SFTP tool to upload the SU2 installation package to the */path/to/SU2* directory on the server.

----End

2.2.6.2 Compiling and Installing SU2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install numpy.aarch64 -y
```

Step 3 Run the following command to decompress the SU2 installation package:

```
tar zxvf SU2-7.0.4.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd SU2-7.0.4
```

Step 5 Execute the **bootstrap** script.

```
./bootstrap
```

Step 6 Run the following commands to switch to the SU2 installation directory:

```
mkdir ../SU2-7.0.4-gnu-9.3.build
cd ../SU2-7.0.4-gnu-9.3.build
```

Step 7 Run the following command to perform configuration:

```
../SU2-7.0.4/configure --prefix=/path/to/SU2/ CXXFLAGS="-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110" --enable-mpi --with-cc=mpicc --with-cxx=mpicxx
```

Step 8 Run the following commands to compile and install SU2:

```
make -j 32
make install
```

Step 9 Run the following commands to add the environment variables:

```
export SU2_RUN=/path/to/SU2/bin
export SU2_HOME=/path/to/SU2
export PATH=$PATH:$SU2_RUN
export PYTHONPATH=$PYTHONPATH:$SU2_RUN
```

----End

2.2.7 Running and Verifying SU2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Download the test case package **su2code.github.io-master.zip**.

NOTICE

The parameters in the case configuration files of SU2 6.*X.X* and SU2 7.*X.X* are greatly changed. You need to obtain the case configuration file of the corresponding version from the corresponding source code package.

Step 3 Use an SFTP tool to upload the SU2 test case package to the */path/to/CASE* directory on the server.

Step 4 Run the following command to decompress the test case package:

```
unzip su2code.github.io-master.zip
```

Step 5 Run the following command to switch to the directory generated after the package is decompressed:

```
cd su2code.github.io-master
```

Step 6 Run the following command to create a **testcase** folder:

```
mkdir testcase
```

Step 7 Run the following commands to copy the **testcase** folder and rename it **Inviscid_Bump**:

```
cd testcase
cp -rf /path/to/CASE/su2code.github.io-master/testcase /Inviscid_Bump
```

Step 8 Run the following command to go to the **Inviscid_Bump** test case package:

```
cd ../Inviscid_Bump
```

Step 9 Run the following command to start SU2:

```
mpirun --allow-run-as-root -np 16 SU2_CFD inv_channel.cfg
```

Information similar to the following is displayed.

```
----------------------- Residual Evolution Summary -----------------------
log10[Maximum residual]: -7.391.
Maximum residual point 511, located at (3, 0.003).
-------------------------------------------------------------------------
 Iter    Time(s)     Res[Rho]     Res[RhoE]      CL(Total)      CD(Total)
  600   0.087015    -9.260167     -3.862403       0.000291       0.000007
  601   0.087013    -9.271532     -3.873152       0.000291       0.000007
  602   0.087011    -9.282237     -3.883903       0.000291       0.000007
  603   0.087010    -9.293729     -3.894743       0.000291       0.000007
  604   0.087008    -9.305008     -3.905581       0.000291       0.000007
  605   0.087006    -9.316490     -3.916410       0.000291       0.000007
  606   0.087006    -9.327466     -3.927381       0.000291       0.000007
  607   0.087004    -9.338974     -3.938270       0.000291       0.000007
  608   0.087002    -9.350370     -3.949267       0.000291       0.000007
  609   0.087000    -9.361588     -3.960211       0.000291       0.000007
  610   0.086998    -9.373061     -3.971327       0.000291       0.000007
  611   0.086997    -9.384581     -3.982358       0.000291       0.000007
------------------------- File Output Summary -------------------------
Writing comma-separated values (CSV) surface files.
Loading solution output data locally on each rank.
Sorting output data across all ranks.
Writing binary SU2 native restart file.
-------------------------------------------------------------------------
------------------------- Solver Postprocessing -------------------------
Deleted CNumerics container.
Deleted CIntegration container.
Deleted CSolver container.
Deleted CIteration container.
```

 Copyright © Huawei Technologies Co., Ltd.

```
Deleted CInterpolator container.
Deleted CTransfer container.
Deleted CGeometry container.
Deleted CFreeFormDefBox class.
Deleted CSurfaceMovement class.
Deleted CVolumetricMovement class.
Deleted CConfig container.
Deleted COutput class.
-------------------------------------------------------------------------
Completed in 53.260358 seconds on 16 cores.
------------------------ Exit Success (SU2_CFD) ------------------------
```

----End

2.2.8 Troubleshooting

Problem 1: An Error Is Reported During SU2 6.0.1 Compilation

Symptom

An error message "error:'VALID_REF' was not declared in this scope; did you mean'INVALID_REF'" is displayed when SU2 is compiled.

Possible Cause

The code has a bug.

Procedure

Download the official release of 6.0.x.

Problem 2: An Error Is Reported During SU2 7.0.4 Compilation

Symptom

An error message "configure: error: cannot find python-config for /usr/bin/python." is displayed when SU2 is compiled.

Possible Cause

The dependency **python-devel** does not exist.

Procedure

Run the following command to install the dependency before **configuring**:

```
yum install python-devel -y
```

2.2.9 More Information

Official SU2 website:

https://su2code.github.io/

SU2 usage guide:

https://su2code.github.io/docs/Execution/#collapse-3

2.3 PALABOS 2.1r0 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

2.3.1 Introduction

The Palabos library is a common computational fluid dynamics (CFD) framework. Its kernel is based on the lattice Boltzmann method (LBM). It can be used as a research tool or an engineering tool. It provides a friendly programming interface and can be easily used to simulate fluid flow.

For more information about Palabos, visit the **official Palabos website**.

Programming language: C++

Brief description: Palabos is fluid dynamics simulation software.

Open source license: GPL 3.0

Recommended Software Versions

Palabos v2.1r0

2.3.2 Environment Requirements

Hardware Requirements

Table 2-10 lists the hardware requirements.

Table 2-10 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software requirements

Table 2-11 lists the software requirements.

Table 2-11 Software requirements

Item	Version	Download Address
PALABOS	v2.1r0	**https://gitlab.com/unigespc/palabos/-/archive/v2.1r0/palabos-v2.1r0.zip**
ImageMagick	6.7.8.9-15.el7_2	RPM package contained in the OS image
Python	3.6.3	**https://www.python.org/downloads/source/**
Test case	cavity3d	Benchmark case provided by the software

OS Requirements

Table 2-12 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-12 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

2.3.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Palabos software porting.

Table 2-13 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PALABOS	Installation path of Palabos	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/PYTHON3	Installation path of Python	

2.3.4 Configuring the Compilation Environment

2.3.4.1 Configuration Description

Prerequisites

Use SFTP to upload the installation packages to the planned directories on the server.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 2-14 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Python	See **2.3.4.2 Installing Python**.
3	Installing ImageMagick	See **2.3.4.3 Installing ImageMagick**.

2.3.4.2 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependencies required by Python:

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel
```

Step 3 Run the following command to decompress the Python installation package:

```
tar -xvf Python-3.6.3.tgz
```

Step 4 Run the following command to switch to the directory containing the decompressed files:

```
cd Python-3.6.3
```

Step 5 Run the following command perform configuration:

```
./configure --prefix=/path/to/PYTHON3
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 7 Run the following commands to set the Python environment variables:

```
export PATH=/path/to/PYTHON3/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON3/lib:$LD_LIBRARY_PATH
```

----End

2.3.4.3 Installing ImageMagick

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the ImageMagick installation package:

 Copyright © Huawei Technologies Co., Ltd.

```
yum install ImageMagick-6.7.8.9-15.el7_2.aarch64 -y
```

----End

2.3.5 Obtaining the Source Code

Procedure

Step 1 Download the Palabos installation package **palabos-v2.1r0.zip**.

https://gitlab.com/unigespc/palabos/-/archive/v2.1r0/palabos-v2.1r0.zip

Step 2 Use an SFTP tool to upload the Palabos installation package to the */path/to/PALABOS* directory on the server.

----End

2.3.6 Compiling and Installing Palabos

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/PALABOS
```

Step 3 Copy the installation package to the main program installation directory.

```
cp palabos-v2.1r0.zip /path/to/PALABOS
```

Step 4 Run the following command to go to the program installation directory:

```
cd /path/to/PALABOS
```

Step 5 Run the following commands to decompress the installation package:

```
unzip palabos-v2.1r0.zip
```

Step 6 Run the following command to go to the **cavity2d** case directory:

```
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/cavity2d
```

Step 7 Run the following command to compile and install the main program based on the test case:

```
make -j 16
```

After the installation is successful, run the **ll** command to view the generated **cavity2d** binary executable file in the current directory.

NOTE

Palabos builds a binary executable file based on the case model. The following uses the **cavity2d** case file as an example.

----End

2.3.7 Running and Verifying Palabos

 Copyright © Huawei Technologies Co., Ltd.

2.3.7.1 Running and Verifying the cavity2d Case

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to set the environment variables:

```
export PATH=/path/to/GNU/bin: /path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 4 Run the following command to go to the **cavity2d** case directory:

```
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/cavity2d
```

Step 5 Run the following command to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x
OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 ./cavity2d
```

NOTE

Run the following command on a single node to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib -np 96 ./cavity2d
```

After the case is executed, you can run the **ll** command to view the generated GIF image file in the **tmp** directory.

----End

2.3.7.2 Running and Verifying the boussinesqThermal3d Case

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to set the environment variables:

```
export PATH=/path/to/GNU/bin: /path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to go to the **cavity2d** case directory:

```
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/boussinesqThermal3d
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
```

Step 6 Run the following command to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x
OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 ./rayleighBenard3D 100
```

NOTE

Run the following command on a single node to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib -np 96 ./rayleighBenard3D 100
```

After the case is executed, you can run the **ll** command to view the generated VTI file in the **tmp** directory.

Figure 2-1 shows an example of the command output:

Figure 2-1 Result example

```
1355000 : Writing gif.
average=1; stdDev/average=1.00779e-14
average=1; stdDev/average=1.00177e-14
4520 : Writing VTK.
1356000 : Writing gif.
average=1; stdDev/average=1.00311e-14
Simulation is over.
N=30
number of processors: 96
simulation time: 9715.61
total time: 9715.69
total iterations: 1356340
Msus: 16.1035

real    162m6.303s
user    15326m24.693s
sys     230m34.920s
```

NOTE

In the preceding figure, **VTK** is the VTI file. The file name extension is not displayed.

----End

2.3.8 Support and Other Resources

Official Palabos website:

https://palabos.unige.ch/

PALABOS installation guide:

https://gitlab.com/unigespc/palabos/-/tree/v2.1r0

2.4 deal.II 9.1.1 Porting Guide (CentOS 7.6)

2.4.1 Introduction

deal.II was originally meant to be a successor to the differential equation analysis library, a C++ library designed to use the adaptive finite element method for

partial differential equation calculations. It uses state-of-the-art programming techniques to provide you with modern interfaces to complex data structures and algorithms that you need.

deal.II aims to quickly develop modern finite element code by using various tools frequently used in adaptive grids and finite element programs. Writing such procedures is not easy, and successful programs tend to become very large and complex. You are recommended to do this with a library that handles the details of grid processing and refinement, degree of freedom, grid input, and output of graphical format results. Similarly, the way that multiple space sizes are supported at a time allows programs to be written independently of the space size and run without causing memory consumption.

For more information about deal.II, visit the **official deal.II website**.

Programming language: C++

Brief description: an open source finite element software program used to solve partial differential equations

Open source license: GNU Lesser General Public version 2.1

Recommended Software Version

deal.II 9.1.1

2.4.2 Environment Requirements

Hardware Requirements

Table 2-15 lists the hardware requirements.

Table 2-15 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-16 lists the software requirements.

Table 2-16 Software requirements

Item	Version	Download Address
deal.II	9.1.1	**https://www.dealii.org/download.html**

OS Requirements

Table 2-17 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-17 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

2.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the deal.II software porting.

Table 2-18 Paths for software porting

No.	Software Installation Path	Usage	Description
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/DEAL.II	Installation path of deal.II	The installation path provided here is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

2.4.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-19 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

2.4.5 Obtaining the Source Code

Procedure

Step 1 Download the **dealii-9.1.1.tar.gz** installation package.

https://www.dealii.org/download.html

Step 2 Use an SFTP tool to upload the deal.II installation package to the */path/to/DEAL.II* directory on the server.

----End

2.4.6 Compiling and Installing deal.II

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install blas lapack zlib boost -y
```

Step 3 Run the following command to decompress the deal.II installation package:

```
tar -xzvf dealii-9.1.1.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd dealii-9.1.1
```

Step 5 Run the following commands to create and access the directory for compiling deal.II:

```
mkdir build
cd build
```

Step 6 Run the following commands to perform compilation and installation:

```
CC=mpicc CXX=mpicxx FC=mpif90 cmake -DDEAL_II_WITH_THREADS=OFF -
DDEAL_II_COMPONENT_DOCUMENTATION=ON -DDEAL_II_WITH_MPI=ON -
DCMAKE_INSTALL_PREFIX=/path/to/DEALL.II ../
```

Step 7 Run the following command to modify the **build.make** file:

1. Open **build.make**.
   ```
   vi tests/quick_tests/CMakeFiles/mpi.debug.run.dir/build.make
   ```
2. Press **i** to enter the insert mode and add the content in bold to line 52 in the **build.make** file.
   ```
   cd /storage/software/TaiShan/dealii/src/build/tests/quick_tests && /storage/software/TaiShan/
   openmpi/openmp    i-4.0.1-gcc9.1-hcoll/bin/mpiexec --allow-run-as-root -mca btl ^openib -np 2 ./
   mpi.debug > mpi.debug-OK 2>&1 || ( ec    ho mpi.debug:\ RUN\ failed.\ Output: && cat mpi.debug-
   OK && rm mpi.debug-OK && exit 1 )
   ```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

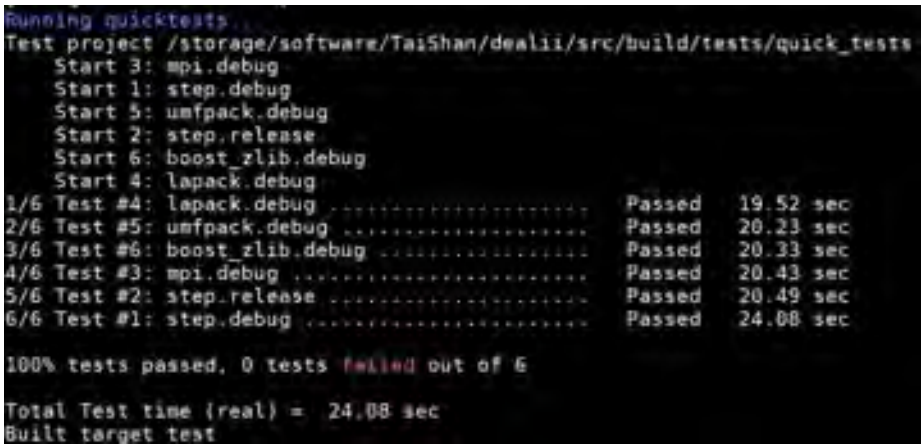
Step 8 Run the following commands to check the setting:

```
make -j 32
make install
make test
```

If **100% tests passed** is displayed, the program is successfully deployed, as shown in **Figure 2-2**.

 Copyright © Huawei Technologies Co., Ltd.

Figure 2-2 Result example



----End

2.4.7 Running and Verifying deal.II

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/deal.II/examples/step-23
```

Step 3 Run the following commands to compile the **step-23** test case:

```
cmake .
make
```

Step 4 Run the following command to create the **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 5 Run the following command to perform the **step-23** test case:

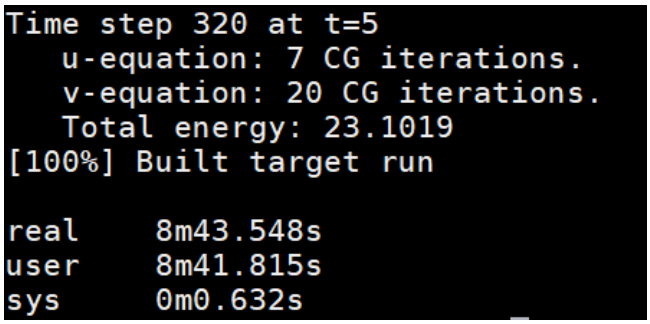
- Single-node:
  ```
  { time make run ; } 2>&1 |tee -a test.out
  ```
- Multi-node:
  ```
  { time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 2 ./step-23 ; } 2>&1 |tee -a test.out
  ```

Check the value of **real** in the **test.out** log. A smaller value indicates better single-core performance.

Figure 2-3 is an example of the output result.

 Copyright © Huawei Technologies Co., Ltd.

Figure 2-3 Result example



----End

2.4.8 More Information

Official deal.II installation guide:

https://www.dealii.org/9.1.1/readme.html#optional

2.5 AMG 1.2 Porting Guide (CentOS 7.6)

2.5.1 Introduction

Algebraic Multigrid Benchmark (AMG) is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. AMG is written in ISO-C. It is an SPMD code which uses MPI and OpenMP threading within MPI tasks. Parallelism is achieved by data decomposition. The driver provided with AMG achieves this decomposition by simply subdividing driver provided with AMG achieves this decomposition by simply subdividing the grid into logical P x Q x R (in 3D) chunks of equal size.

For details about AMG, visit the **official AMG website**.

Programming language: C

Brief description: A parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids.

Open source license: GPL

Recommended Software Version

AMG AMG-1.2

2.5.2 Environment Requirements

Hardware Requirements

Table 2-20 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-20 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-21 lists the software requirements.

Table 2-21 Software requirements

Item	Version	How to Obtain
AMG	AMG-1.2	https://github.com/LLNL/AMG/tree/1.2

OS Requirements

Table 2-22 lists the OS requirements.

Table 2-22 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

2.5.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AMG software porting.

Table 2-23 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/AMG	Installation path of AMG	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

2.5.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-24 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

2.5.5 Obtaining the Source Code

Procedure

Step 1 Download the AMG installation package **AMG-1.2.zip**.

https://github.com/LLNL/AMG/tree/1.2

Step 2 Use an SFTP tool to upload the AMG installation package to the */path/to/AMG* directory on the server.

----End

2.5.6 Compiling and Installing AMG

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the AMG installation package:

```
unzip AMG-1.2.zip
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd /path/to/AMG/AMG-1.2
```

Step 4 Run the following commands to perform compilation:

```
make
```

NOTE

In **Makefile.include**, **-DHYPRE_USING_PERSISTENT_COMM** optimizes the MPI and **-DHYPRE_HOPSCOTCH** optimizes the OpenMP.

----End

2.5.7 Running and Verifying AMG

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 3 Run the following command to go to the **test** directory. After the compilation, the **amg** binary file is generated.

```
cd /path/to/AMG/AMG-1.2/test
```

Step 4 Run the following command to run the **amg** binary file:

```
mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 amg -P 4 6 4 -n 100 100 100 -printstats
```

NOTE

- **-p** *<Px> <Py> <Pz>*: defines the processor topology for each part. The total number of MPI processes is Px * Py * Pz.
- **-n** *<nx> <ny> <nz>*: defines the size of each processor for problems of the multi-dimensional data set. The **-n** option allows each MPI process to specify the size of a local problem, generating the global problem size of **<Pz> * <nz>** for **<Px> * <nx>** and **<Py> * <ny>**.
- **-printstats**: displays detailed information about the AMG preprocessor and the number of iterations.
- **-N**: indicates the number of tasks on each node. The value of Px * Py * Pz must be equal to the value of -N multiplied by the number of nodes.
- If only one node is running, run the following command:
  ```
  mpirun --allow-run-as-root -mca btl ^openib -x OMP_NUM_THREADS=1 -np 96 amg -P 4 6 4 -n 100 100 100 -printstats
  ```

The following is an example of the command output:

```
BoomerAMG SOLVER PARAMETERS:
  Maximum number of cycles:         1
  Stopping Tolerance:               0.000000e+00
```

 Copyright © Huawei Technologies Co., Ltd.

```
 Cycle type (1 = V, 2 = W, etc.):  1
 Relaxation Parameters:
  Visiting Grid:                   down   up  coarse
         Number of sweeps:           2    2     1
  Type 0=Jac, 3=hGS, 6=hSGS, 9=GE:     18   18     9
  Point types, partial sweeps (1=C, -1=F):
               Pre-CG relaxation (down):   0   0
                Post-CG relaxation (up):   0   0
                         Coarsest grid:   0
=============================================
Problem 1: AMG Setup Time:
=============================================
PCG Setup:
  wall clock time = 6.333697 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 6.350000 seconds
  cpu MFLOPS      = 0.000000
FOM_Setup: nnz_AP / Setup Phase Time: 4.452531e+08
=============================================
Problem 1: AMG-PCG Solve Time:
=============================================
PCG Solve:
  wall clock time = 34.855871 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 34.860000 seconds
  cpu MFLOPS      = 0.000000
Iterations = 23
Final Relative Residual Norm = 6.624473e-09
FOM_Solve: nnz_AP * Iterations / Solve Phase Time: 1.860870e+09
Figure of Merit (FOM_1): 1.506966e+09
```

----End

2.6 Code_Saturne 6.0.2 Porting Guide (CentOS 7.6)

2.6.1 Introduction

Code_Saturne is the free, open-source software developed and released by EDF to solve computational fluid dynamics (CFD) applications.

It solves the Navier-Stokes equations for 2D, 2D-axisymmetric and 3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not.

For details about Code_Saturne, visit the **official Code_Saturne website**.

Programming language: C

Brief description: free, open-source CFD software developed and released by EDF

Open source license: GNU GPL license

Recommended Software Version

Code_Saturne 6.0.2

2.6.2 Environment Requirements

Hardware Requirements

Table 2-25 lists the hardware requirements.

Table 2-25 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-26 lists the software requirements.

Table 2-26 Software requirements

Item	Version	Download Address
Code_Saturne	v6.0.2	**https://github.com/code-saturne/code_saturne/**
HDF5	1.10.3	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.3/src/hdf5-1.10.3.tar.gz**
MED	4.0.0	**https://sources.voidlinux-ppc.org/med-4.0.0/med-4.0.0.tar.gz**
SCOTCH	6.0.9	**https://www.labri.fr/perso/pelegrin/scotch/distrib/scotch_6.0.9.tar.gz**
METIS	5.1.0	**https://src.fedoraproject.org/lookaside/pkgs/metis/metis-5.1.0.tar.gz/**

OS Requirements

Table 2-27 lists the OS requirements.

Table 2-27 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

2.6.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Code_Saturne software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-28 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CODE_SATURNE	Installation path of Code_Saturne	The planned installation paths are only examples. You are advised to deploy the software in shared paths. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/HDF5	Installation path of HDF5	
4	/path/to/MED	Installation path of MED	
5	/path/to/SCOTCH	Installation path of SCOTCH	
6	/path/to/METIS	Installation path of METIS	

2.6.4 Configuring the Compilation Environment

2.6.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-29 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing the BLAS dependency	See **2.6.4.2 Installing the BLAS Dependency**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing PyQt4	See **2.6.4.3 Installing PyQt4**.
4	Installing HDF5	See **2.6.4.4 Installing HDF5**.
5	Installing MED	See **2.6.4.5 Installing MED**.
6	Installing SCOTCH	See **2.6.4.6 Installing SCOTCH**.
7	Installing METIS	See **2.6.4.7 Installing METIS**.

2.6.4.2 Installing the BLAS Dependency

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install blas blas-devel -y
```

----End

2.6.4.3 Installing PyQt4

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to install PyQt4:

```
yum install PyQt4.aarch64 PyQt4-devel.aarch64 -y
```

----End

2.6.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -zxvf hdf5-1.10.3.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.3
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to configure HDF5:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared
```

Step 5 Run the following commands to compile and install HDF5:

```
make -j
make install
```

Step 6 Run the following commands to set the HDF5 environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

2.6.4.5 Installing MED

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the MED installation package:

```
tar -zxvf med-4.0.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd med-4.0.0
```

Step 4 Run the following command to configure MED:

```
./configure --prefix=/path/to/MED --with-hdf5=/path/to/HDF5 --disable-python
```

Step 5 Run the following commands to compile and install MED:

```
make -j
make install
```

Step 6 Run the following commands to set the MED environment variables:

```
export PATH=/path/to/MED/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MED/lib:$LD_LIBRARY_PATH
```

----End

2.6.4.6 Installing SCOTCH

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the SCOTCH installation package:

```
tar -zxvf scotch_6.0.9.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd scotch_6.0.9
```

Step 4 Run the following command to create a soft link for **Makefile.inc**:

```
cd src
ln -s ./Make.inc/Makefile.inc.x86-64_pc_linux2.shlib Makefile.inc
```

Step 5 Run the following commands to compile and install SCOTCH:

 Copyright © Huawei Technologies Co., Ltd.

```
make scotch
```

Step 6 Run the following commands to set the SCOTCH environment variables:

```
export PATH=/path/to/SCOTCH/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SCOTCH/lib:$LD_LIBRARY_PATH
```

----End

2.6.4.7 Installing METIS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the METIS installation package:

```
tar -zxvf metis-5.1.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd metis-5.1.0
```

Step 4 Run the following commands to compile and install METIS:

```
make config prefix=/path/to/METIS
make install
```

Step 5 Run the following commands to set the METIS environment variables:

```
export PATH=/path/to/METIS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/METIS/lib:$LD_LIBRARY_PATH
```

----End

2.6.5 Obtaining the Source Code

Procedure

Step 1 Download the Code_Saturne installation package **v6.0.2.tar.gz**.

https://github.com/code-saturne/code_saturne/

Step 2 Use an SFTP tool to upload the Code_Saturne installation package to the */path/to/Code_Saturne* directory on the server.

----End

2.6.6 Compiling and Installing Code_Saturne

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the Code_Saturne installation package:

```
tar -xvf v6.0.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd code_saturne-6.0.2
```

Step 4 Run the following command to start bootstrap:

```
./sbin/bootstrap
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following command to perform configuration:

```
CC=mpicc FC=mpifort CXX=mpicxx ./configure --prefix=/path/to/CODE_SATURNE --with-blas-lib=/usr/lib64 --with-hdf5=/path/to/HDF5 --with-scotch=/path/to/SCOTCH --with-metis=/path/to/METIS --with-med=/path/to/MED --enable-shared --with-mpi CXXFLAGS="-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110"
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 32
make install
```

Step 7 Run the following command to set the Code_Saturne environment variable:

```
export PATH=/path/to/CODE_SATURNE/bin:$PATH
```

Step 8 Run the following command to check whether Code_Saturne is installed successfully:

```
which code_saturne
```

If the following information is displayed, the installation is successful:

```
/path/to/CODE_SATURNE/bin/code_saturne
```

----End

2.6.7 Running and Verifying Code_Saturne

2.6.7.1 Running and Verifying Code_Saturne on a Single Node

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Create a case directory and go to the directory.

```
cd /path/to/CODE_SATURNE/
mkdir TESTCASE
cd TESTCASE
```

Step 3 Run the following command to create a case:

```
code_saturne create -s test -c case1
```

NOTE

In the previous command, *test* indicates the user-defined study name and *case1* indicates the user-defined case name. You can change them based on the actual situation. After this command is run, a **test** directory is generated in the current directory, and the **case1**, **MESH**, and **POST** directories are generated in the **test** directory.

Step 4 Run the following command to copy the test case file:

```
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/mesh/*  test/MESH -rf
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/case1/setup.xml test/case1/DATA -rf
```

Step 5 Run the following command to set the MPI environment variable:

```
export CS_MPIEXEC_OPTIONS="--allow-run-as-root --mca btl ^openib"
```

Step 6 Run the following command to run Code_Saturne on a single node:

```
code_saturne run -n 18 --nt=1 --param=/path/to/CODE_SATURNE/TESTCASE/test/case1/DATA/setup.xml --case=/path/to/CODE_SATURNE/TESTCASE/test/case1
```

 Copyright © Huawei Technologies Co., Ltd.

Table 2-30 Parameter description

Parameter	Description
-n <nprocs>	Number of MPI processes.
--nt=NTHREADS	Number of OpenMP threads of each task.
--param=<param>	Path and name of the parameter file.
--case=<case>	Case path.

If the following information is displayed, the initialization is successful:

```
*****************************
 Post-calculation operations
*****************************
```

Step 7 Run the following command to go to the directory of test cases:

```
cd /path/to/CODE_SATURNE/TESTCASE/test/case1/RESU/year+month+day-hour+minute_round
```

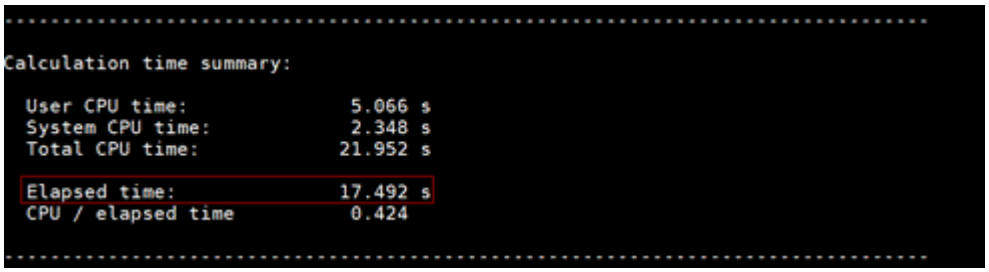
NOTE

A log is generated in the **RESU** directory each time a job is run. The log directory format is *YearMonthDay-HourMinute_Sequence number of job executions in the same minute*. For example, the **20200330-2104** directory contains the log of the first job execution at 21:04 on March 30, 2020, and the **20200330-2104_1** directory contains the log of the second job execution at 21:04 on March 30, 2020.

Elapsed time: *XX* **s** in the **performance.log** file indicates the performance. A smaller value indicates a better performance.

Figure 2-4 shows the command output:

Figure 2-4 An example result



----End

2.6.7.2 Running and Verifying Code_Saturne on Multiple Nodes

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Create a case directory and go to the directory.

```
cd /path/to/CODE_SATURNE/
mkdir TESTCASE
cd TESTCASE
```

Step 3 Run the following command to create a case:

```
code_saturne create -s test -c case1
```

NOTE

In the previous command, *test* indicates the user-defined study name and *case1* indicates the user-defined case name. You can change them based on the actual situation. After this command is run, a **test** directory is generated in the current directory, and the **case1**, **MESH**, and **POST** directories are generated in the **test** directory.

Step 4 Run the following command to copy the test case file:

```
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/mesh/*  test/MESH -rf
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/case1/setup.xml test/case1/DATA -rf
```

Step 5 Run the following command to set the MPI environment variable:

```
export CS_MPIEXEC_OPTIONS="--allow-run-as-root --mca btl ^openib"
```

Step 6 Run the following command to initialize the case:

```
code_saturne run --initialize --param=/path/to/CODE_SATURNE /TESTCASE/test/case1/DATA/setup.xml --case=/path/to/CODE_SATURNE /TESTCASE/test/case1
```

If the following information is displayed, the initialization is successful:

```
*****************************
  Preprocessing calculation
*****************************
```

Step 7 Run the following command to create the **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

In the previous command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.

Step 8 Run the following command to go to the directory of test cases:

```
cd /path/to/CODE_SATURNE/TESTCASE/test/case1/RESU/year+month+day-hour+minute_round
```

NOTE

A log is generated in the **RESU** directory each time a job is run. The log directory format is *YearMonthDay-HourMinute_Sequence number of job executions in the same minute*. For example, the **20200330-2104** directory contains the log of the first job execution at 21:04 on March 30, 2020, and the **20200330-2104_1** directory contains the log of the second job execution at 21:04 on March 30, 2020.

Step 9 Run the following command to run the test case on multiple nodes:

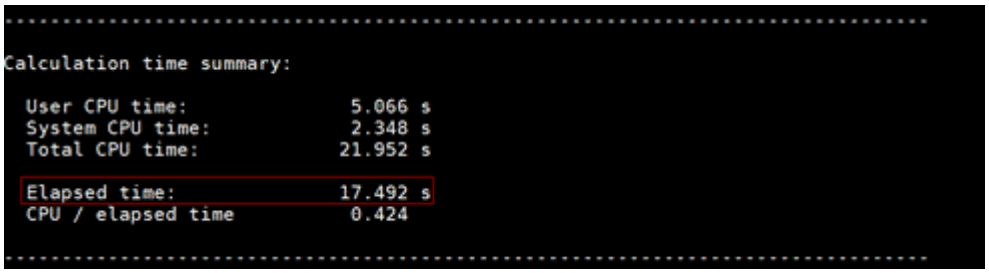
```
time mpirun --allow-run-as-root --mca btl ^openib -N 96 -x OMP_NUM_THREADS=1 --hostfile /path/to/HOSTFILE ./cs_solver --mpi
```

Elapsed time: *XX* **s** in the **performance.log** file indicates the performance. A smaller value indicates a better performance.

Figure 2-5 shows the command output:

 Copyright © Huawei Technologies Co., Ltd.

Figure 2-5 An example result



----End

2.6.8 More Resources

Official Code_Saturne website:

https://www.code-saturne.org/cms/web/

2.7 FDS 6.7.4 Porting Guide (CentOS 7.6)

2.7.1 Introduction

FDS, developed by the National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce and the VTT Technical Research Center of Finland, is a large eddy simulation software for low-speed flows. It focuses on the smoke and heat transfer caused by fires and aims to solve the actual fire problems in fire engineering. It also provides tools for studying basic dynamics and combustion in a fire.

For more information about FDS, visit the **official FDS website**.

Programming language: Fortran

Brief description: fire smoke and heat transfer simulation tool

Recommended Software Version

FDS 6.7.4

2.7.2 Environment Requirements

Hardware Requirements

Table 2-31 lists the hardware requirements.

Table 2-31 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 2-32 lists the software requirements.

Table 2-32 Software requirements

Item	Version	Download Address
FDS	6.7.4	https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4
Test case	weak_scaling_test_128	Test case provided by the software

OS Requirements

Table 2-33 lists the OS requirements.

Table 2-33 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

2.7.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FDS software porting.

Table 2-34 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/FDS	Installation path of FDS	The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

2.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-35 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

2.7.5 Obtaining the Source Code

Procedure

Step 1 Download the FDS installation package **fds-FDS6.7.4.tar.gz**.

https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4

Step 2 Use an SFTP tool to upload the FDS installation package to the */path/to/FDS* directory on the server.

----End

2.7.6 Compiling and Installing FDS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to decompress the FDS installation package:

```
tar zxvf fds-FDS6.7.4.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd fds-FDS6.7.4
```

Step 5 Run the following commands to delete the **-m64** compilation option.

```
cd /path/to/FDS/fds-FDS6.7.4/Build
sed -i "s/-m64//g" ./makefile
```

The compilation option before modification:

 Copyright © Huawei Technologies Co., Ltd.

```
mpi_gnu_linux_64 : FFLAGS = -m64 -O2 -std=f2008
```

The compilation option after modification:

```
mpi_gnu_linux_64 : FFLAGS =  -O2 -std=f2008
```

Step 6 Run the following command to switch to the FDS compilation directory:

```
cd /path/to/FDS/fds-FDS6.7.4/Build/mpi_gnu_linux_64
```

Step 7 Run the following command to run the automatic compilation script:

```
./make_fds.sh
```

Step 8 Run the following commands to add the FDS environment variable:

```
export PATH=/path/to/FDS/fds-FDS6.7.4/Build/mpi_gnu_linux_64:$PATH
```

----End

2.7.7 Running and Verifying FDS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the **FDS** directory:

```
cd /path/to/FDS/fds-FDS6.7.4/
```

Step 3 Run the following command to create a **testcase** directory:

```
mkdir testcase
```

Step 4 Run the following command to go to the **testcase** directory:

```
cd testcase
```

Step 5 Run the following command to set the number of OpenMP threads to **1**:

```
export OMP_NUM_THREADS=1
```

Step 6 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 7 Run the following commands to run FDS:

- Single node

  ```
  `which mpirun` -np 128 --mca btl ^openib --allow-run-as-root ../../Build/mpi_gnu_linux_64/fds_mpi_gnu_linux_64 ../../Validation/MPI_Scaling_Tests/FDS_Input_Files/weak_scaling_test_128.fds
  ```

  If the command is successfully run on a single node, the following information is displayed:

  ```
  STOP: FDS completed successfully (CHID: weak_scaling_test_128)
  ```

- Multiple nodes

  ```
  `which mpirun` -N 128 -x OMP_NUM_THREADS -x PATH -x LD_LIBRARY_PATH --hostfile /path/to/HOSTFILE --mca btl ^openib --allow-run-as-root ../../Build/mpi_gnu_linux_64/fds_mpi_gnu_linux_64 ../../Validation/MPI_Scaling_Tests/FDS_Input_Files/weak_scaling_test_288.fds
  ```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> In the preceding case, the value of **X** in **weak_scaling_test_X.fds** must be greater than the total number of CPU cores used for the running. For example, if two nodes need to use test cases whose **X** is greater than or equal to 256, set **weak_scaling_test_X.fds** to **weak_scaling_test_288.fds**.

Check the value of **real** in the **fds.log** log. The unit is second. A smaller value indicates better performance.

Figure 2-6 shows an example.

Figure 2-6 Result example

```
 Time Step:     60, Simulation Time:      6.00 s
 Time Step:     70, Simulation Time:      7.00 s
 Time Step:     80, Simulation Time:      8.00 s
 Time Step:     90, Simulation Time:      9.00 s
 Time Step:    100, Simulation Time:     10.00 s

STOP: FDS completed successfully (CHID: weak_scaling_test_128)

real    50m51.274s
user    4560m45.142s
sys     305m24.382s
```

----End

2.7.8 Troubleshooting

Failure 1: FDS Runtime Error

Symptom

An error is reported during FDS running, indicating that too many files are opened. The error message is "Fortran runtime error: Cannot open file 'weak_scaling_test_128_0022_04.s3d': Too many open files."

Possible Causes

When the FDS is running, a large number of files are read and written. By default, the maximum number of files that can be opened is 1024, which cannot meet the FDS running requirements.

Procedure

Tens of millions of files can be opened at the same time on a Taishan server. You can run the **ulimit -n** command to set the maximum number of files that can be opened at the same time. Example:

```
ulimit -n 10240
```

2.7.9 More Information

FDS home page:

https://pages.nist.gov/fds-smv/

FDS usage guidelines:

https://pages.nist.gov/fds-smv/manuals.html

 Copyright © Huawei Technologies Co., Ltd.

2.8 Elmer 8.4 Porting Guide (CentOS 7.6)

2.8.1 Introduction

Elmer is a finite element software for numerical solutions of partial differential equations. Elmer can handle any number of equations, so it is very suitable for simulating multiphysical problems. Elmer includes physical models of fluid dynamics, structural mechanics, electromagnetics, heat transfer and acoustics. Users can also write their own equations that can be dynamically linked to the main program.

For more information about Elmer, visit the **official Elmer website**.

Programming languages: Fortran, C++, C

Brief description: multiphysical simulation software

Open source license: GPLv2

Recommended Software Version

Elmer 8.4

2.8.2 Environment Requirements

Hardware Requirements

Table 2-36 lists the hardware requirements.

Table 2-36 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-37 lists the software requirements.

Table 2-37 Software requirements

Item	Version	How to Obtain
Elmer	8.4	**https://codeload.github.com/ElmerCSC/elmerfem/tar.gz/scc20**
CMake	3.8.1	**https://cmake.org/download/**
OpenBLAS	0.3.9	**https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Official test case	-	https://sources.debian.org/data/non-free/e/elmer-doc/2014.02.06-1~bpo70%2B1/ElmerTutorialsFiles_nonGUI.tar.gz

OS Requirements

Table 2-38 lists the OS requirements.

Table 2-38 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

2.8.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Elmer software porting.

Table 2-39 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/ELMER	Installation path of Elmer.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/CMAKE	Installation path of Cmake.	
4	/path/to/OPENBLAS	Installation path of OpenBLAS.	

2.8.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

2.8.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-40 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **2.8.4.2 Installing CMake**.
3	Installing OpenBLAS	See **2.8.4.3 Installing OpenBLAS**.

2.8.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the CMake installation package:

```
tar zxvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd cmake-3.8.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make install
```

Step 6 Run the following commands to set the CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

2.8.4.3 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

```
tar xzvf OpenBLAS-0.3.9.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd OpenBLAS-0.3.9/
```

Step 4 Run the following commands to set the environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 96
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following commands to set the OpenBLAS environment variables:

```
export LIBRARY_PATH=/path/to/OPENBLAS/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

2.8.5 Obtaining the Source Code

Procedure

Step 1 Download the Elmer source code package **elmerfem-scc20.tar.gz**.

https://codeload.github.com/ElmerCSC/elmerfem/tar.gz/scc20

Step 2 Download the official test case package **ElmerTutorialsFiles_nonGUI.tar.gz**

https://sources.debian.org/data/non-free/e/elmer-doc/2014.02.06-1~bpo70%2B1/ElmerTutorialsFiles_nonGUI.tar.gz

Step 3 Use an SFTP tool to upload the downloaded installation package to the */path/to/ELMER* directory on the server.

----End

2.8.6 Compiling and Installing Elmer

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Elmer installation package:

```
tar -xvf elmerfem-scc20.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd elmerfem-scc20
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following commands to create a folder:

```
mkdir build
cd build
```

Step 5 Run the following command to perform compilation:

```
cmake -DCMAKE_INSTALL_PREFIX=/path/to/ELMER -DCMAKE_C_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_Fortran_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" ../
```

Step 6 Run the following commands to perform installation:

```
make -j 32
make install
```

Step 7 Run the following commands to set the environment variables:

```
export PATH=/path/to/ELMER/bin:$PATH
export ELMER_HOME=/path/to/ELMER
```

----End

2.8.7 Running and Verifying Elmer

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

- In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 3 Run the following commands to decompress the official package:

```
cd /path/to/ELMER/
tar -xvf ElmerTutorialsFiles_nonGUI.tar.gz
```

Step 4 Run the following command to enter the test case:

```
cd tutorials_files/FlowResistance
```

NOTE

- **FlowResistance** is one of the official test cases.
- For details about how to run other test cases, see the **official manual**.

Step 5 Run the following command to generate a grid:

```
ElmerGrid 1 2 hole.grd
```

Step 6 Run the following command to run the case:

```
time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x PATH -x LD_LIBRARY_PATH -x ELMER_HOME -N 128 ElmerSolver
```

NOTE

Run the following command to run the case in single-node mode:

```
time mpirun --allow-run-as-root -mca btl ^openib -np 128 ElmerSolver
```

After the command is run, the following information is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
ElmerSolver: *** Elmer Solver: ALL DONE ***
ElmerSolver: The end
SOLVER TOTAL TIME(CPU,REAL):       10.23       2.86
ELMER SOLVER FINISHED AT: 2020/05/12 17:16:54
```

----End

2.8.8 Troubleshooting

Problem 1: An Error Is Reported When make install Is Run

Symptom

"Error: Array specification at (1) has more than 7 dimensions" is displayed when the **make install** command is run.

Possible Cause

The environment configuration is incorrect during compilation.

Procedure

The GFortran version is too early. Add the following command to perform the compilation and installation again when setting environment variables:

```
export FC= /path/to/GNU/bin/gfortran
```

2.8.9 More Information

For more information, visit the official Elmer website:

http://www.csc.fi/elmer

2.9 Code_Aster 14.6.0 Porting Guide (CentOS 7.6)

2.9.1 Introduction

Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics. It runs on Linux and was public released based on General Public License in October 2001. Code_Aster has now 1,500,000 lines of source code, most of which are developed in Fortran and Python, and is being constantly developed and updated. Most of the Code_Aster fields have been validated by independent comparisons with analytical or experimental results, benchmarks towards other codes.

Code_Aster is a solver, based on the theory of the mechanics of the continuous media, which uses the method of the finite elements to solve different types of mechanical, thermal, acoustic, seismic, and other problems. Beyond the standard functionalities of a FEM software for solid mechanics, Code_Aster also compiles specific research in various fields: fatigue, damage, fracture, contact, geomaterials, porous media, and multiphysics coupling.

Currently, the Code_Aster software has been widely used to evaluate and maintain power plants and power networks, and to analyze and calculate bridge data.

Salome is an open-source software that provides a generic platform for pre- and post-processing for numerical simulation. Salome-Meca is a standalone

 Copyright © Huawei Technologies Co., Ltd.

application, which is an external project based on the SALOME platform. Salome-Meca represents integration of the Code_Aster solver in the SALOME platform.

Programming languages: C/Fortran/Python

Brief description: Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics.

Open source license: GPL

NOTE

For Code_Aster 14.6.0 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port Code_Aster to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

Code_Aster 14.6.0

2.9.2 Environment Requirements

Hardware Requirements

Table 2-41 lists the hardware requirements.

Table 2-41 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-42 lists the software requirements.

Table 2-42 Software requirements

Item	Version	Download Address
Code_Aster	14.6.0	**https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz**

OS Requirements

Table 2-43 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-43 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

2.9.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Code_Aster software porting.

Table 2-44 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/CODE-ASTER	Installation path of Code_Aster.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

2.9.4 Configuring the Compilation Environment

2.9.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 2-45 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
4	Configuring the Yum source	See **2.9.4.2 Configuring the Yum Source**.

2.9.4.2 Configuring the Yum Source

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to download the **Base** source configuration file:

```
wget --no-check-certificate -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.tools.huawei.com/repository/conf/CentOS-AltArch-7.repo
```

Step 3 Run the following commands to configure the EPEL source configuration file:

```
echo '[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.tools.huawei.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0' > /etc/yum.repos.d/epel.repo
```

Step 4 Run the following commands to refresh the Yum source cache:

```
yum clean all
yum makecache
```

----End

 Copyright © Huawei Technologies Co., Ltd.

2.9.5 Obtaining the Source Code

Procedure

Step 1 Download the Code_Aster installation package **aster-full-src-14.6.0-1.noarch.tar.gz**.

Download address: **https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz**

Step 2 Use an SFTP tool to upload the Code_Aster installation package to the */path/to/CODE-ASTER* directory on the server.

----End

2.9.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependencies:

```
yum -y install zlib* lapack* blas* python3.aarch64 python3-devel.aarch64 cmake3*  boost_python* boost boost-devel numpy* python36-numpy flex bison
```

Step 3 Run the following command to go to the main program installation directory:

```
cd /path/to/CODE-ASTER
```

Step 4 Run the following command to decompress the main program installation package:

```
tar -xvf aster-full-src-14.6.0-1.noarch.tar.gz
```

Step 5 Run the following command to go to the directory generated after decompression:

```
cd aster-full-src-14.6.0
```

Step 6 Run the following commands to modify the **setup.py** file:

1. Open the **setup.py** file.
   ```
   vi setup.py
   ```
2. Press **i** to enter the insert mode and add the **aarch64** option.

 Copyright © Huawei Technologies Co., Ltd.

```
344     common_libs = ['pthread', 'z']
345     cfg['PLATFORM'] = plt
346     if plt.startswith('linux'):
347         plt = 'linux'
348     if plt == 'win32':
349         cfg['IFDEF'] = 'WIN32'
350     elif plt in ('linux', 'cygwin'):
351         cfg['ARCH'] = 'x86'
352         if machine.endswith('64'):
353             cfg['IFDEF'] = 'LINUX64'
354             if machine in ('x86_64', 'ia64', 'ppc64', 'aarch64'):
355                 cfg['ARCH'] = machine
356             else:  # force to x86_64
357                 cfg['ARCH'] = 'x86_64'
358         else:
359             cfg['IFDEF'] = 'LINUX'
360     elif plt == 'darwin':
361         cfg['ARCH'] = 'x86'
362         if machine.endswith('64'):
363             cfg['IFDEF'] = 'DARWIN64'
364             if machine in ('x86_64', 'ia64', 'ppc64'):
365                 cfg['ARCH'] = machine
366             else:  # force to x86_64
367                 cfg['ARCH'] = 'x86_64'
368         else:
369             cfg['IFDEF'] = 'DARWIN'
370     elif plt.startswith('freebsd'):
371         common_libs = []
372         cfg['IFDEF'] = 'FREEBSD'
373         cfg['ARCH'] = 'x86'
374         if machine.endswith('64'):
375             if machine in ('x86_64', 'ia64', 'ppc64'):
376                 cfg['ARCH'] = machine
377             else:  # force to x86_64
378                 cfg['ARCH'] = 'x86_64'
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

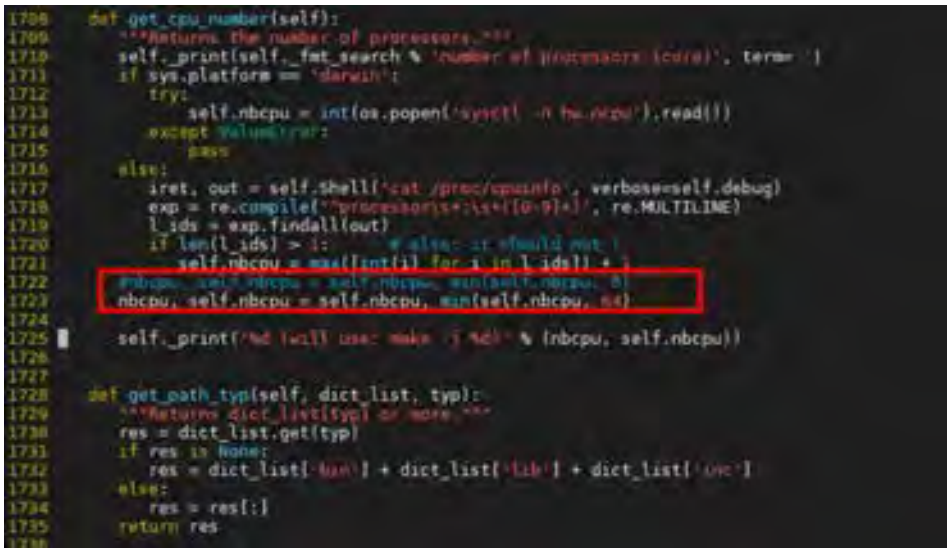
Step 7 Run the following commands to modify the **as_setup.py** file:

1. Open the **as_setup.py** file.
   vi as_setup.py
2. Press **i** to enter the insert mode.
   a. Add the **aarch64** option to line 1538.
      'aarch64': 'shell|script|aarch64',

```
1519 #--------------------------------------------------------------------------
1520 #--------------------------------------------------------------------------
1521 #--------------------------------------------------------------------------
1522 class FIND_TOOLS:
1523     """Some utilities to find files, libraries...
1524     """
1525     _fmt_search  = _('Checking for %s... ')
1526     _fmt_test    = _('Checking if %s... ')
1527     _fmt_chkpref = _('Checking prefix for %s (%s)... ')
1528     _fmt_home    = _('adjust to %s')
1529     _fmt_locate  = _('(locate)... ')
1530     _fmt_yes     = _('yes')
1531     _fmt_no      = _('no')
1532     dict_rearch = {
1533         'x86'    : 'shell|script|80.86',
1534         'i86pc'  : 'shell|script|80.86',
1535         'x86_64' : 'shell|script|x86.64',
1536         'ppc64'  : 'shell|script|64-bit PowerPC',
1537         'ia64'   : 'shell|script|ia.64',
1538         'aarch64': 'shell|script|aarch64',
1539     }
```

   b. Modify the number of compilation cores to increase the compilation speed.

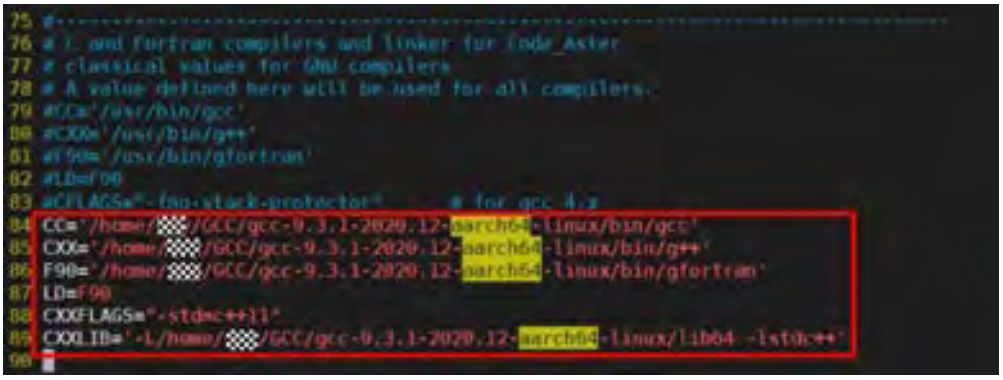
 Copyright © Huawei Technologies Co., Ltd.



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following commands to modify the **setup.cfg** file:

1. Open the **setup.cfg** file.
   ```
   vi setup.cfg
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   CC='/path/to/gcc/bin/gcc'
   CXX='/ path/to/gcc/bin/g++'
   F90='/ path/to/gcc/bin/gfortran'
   LD=F90
   CXXFLAGS="-std=c++11"
   CXXLIB='-L/ path/to/gcc/lib64 -lstdc++'
   ```



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following commands to modify the med configuration:

```
sed '186 a\ \  conf_cmd = ("export F77=$F90; "' -i products.py
sed -i '188d' products.py
```

Step 10 Run the following command to go to the Code_Aster source code directory:

```
cd SRC/
```

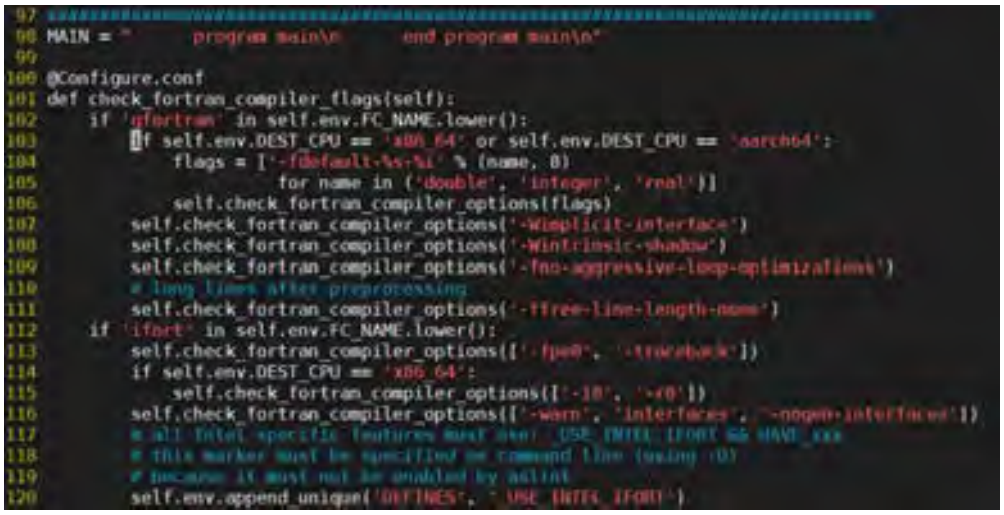
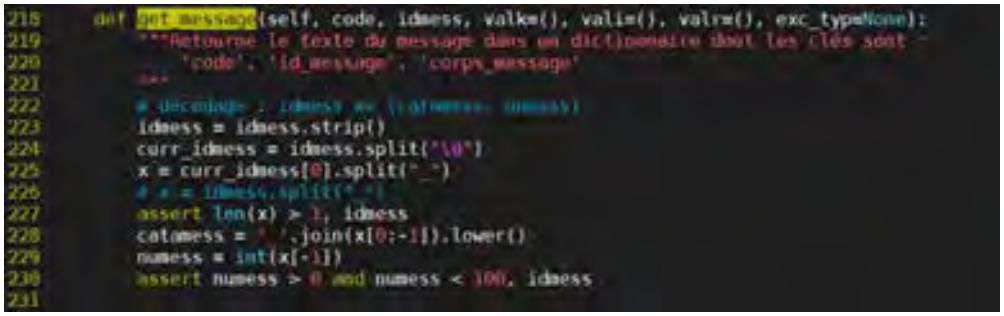
Step 11 Run the following command to decompress the Code_Aster installation package:

```
tar xvf aster-14.6.0.tgz
```

Step 12 Run the following commands to edit the files:

1. Open the **/bibfor/wscript** file.
   ```
   vi aster-14.6.0/bibfor/wscript
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and modify line 103 as follows:
   ```
   if self.env.DEST_CPU == 'x86_64' or self.env.DEST_CPU == 'aarch64':
   ```
   
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Open the **/bibpyt/Utilitai/Utmess.py** file.
   ```
   vi ./aster-14.6.0/bibpyt/Utilitai/Utmess.py
   ```
5. Press **i** to enter the insert mode and modify line 224 as follows:
   ```
   curr_idmess = idmess.split("\0")
   x = curr_idmess[0].split("_")
   # x = idmess.split("_")
   ```
   
6. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 13 Run the following command to decompress the Code_Aster installation package:

```
tar cvzf aster-14.6.0.tgz aster-14.6.0
```

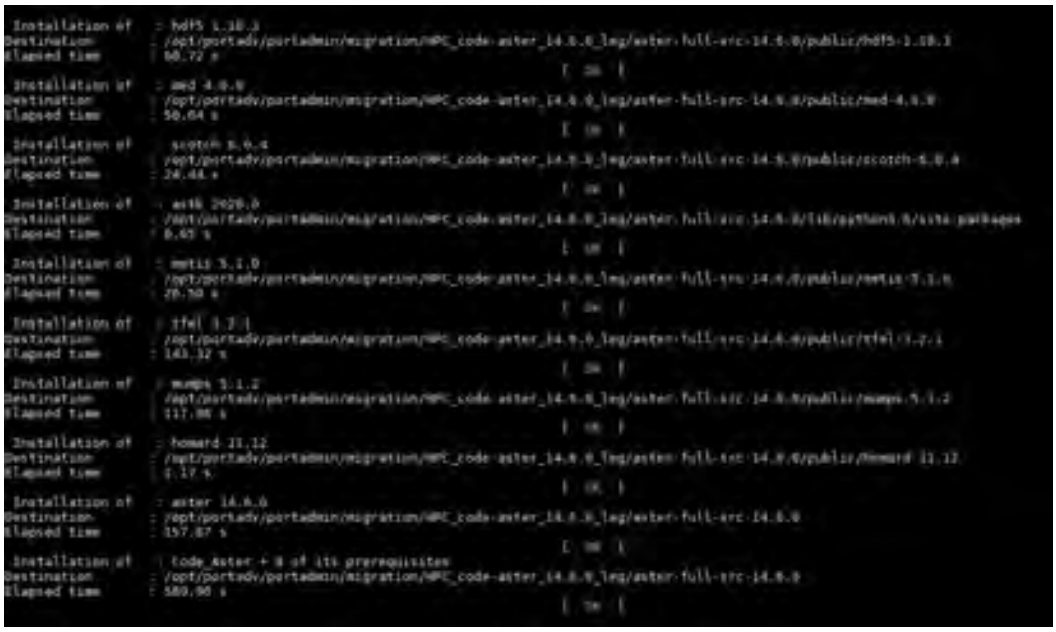
Step 14 Run the following command to delete the Code_Aster installation package:

```
rm -rf aster-14.6.0
```

Step 15 Run the following command to perform the compilation:

```
cd /path/to/CODE-ASTER/aster-full-src-14.6.0
echo y | python3 setup.py --prefix=/path/to/CODE-ASTER
```

 Copyright © Huawei Technologies Co., Ltd.



----End

2.9.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the **CODE-ASTER** directory:

```
cd /path/to/CODE-ASTER
```

Step 3 Run the following command to set the environment variables:

```
source /path/to/CODE-ASTER/profile.sh
```

Step 4 Run the following command to obtain the **forma01a.export** file:

```
as_run --get_export forma01a > forma01a.export
```

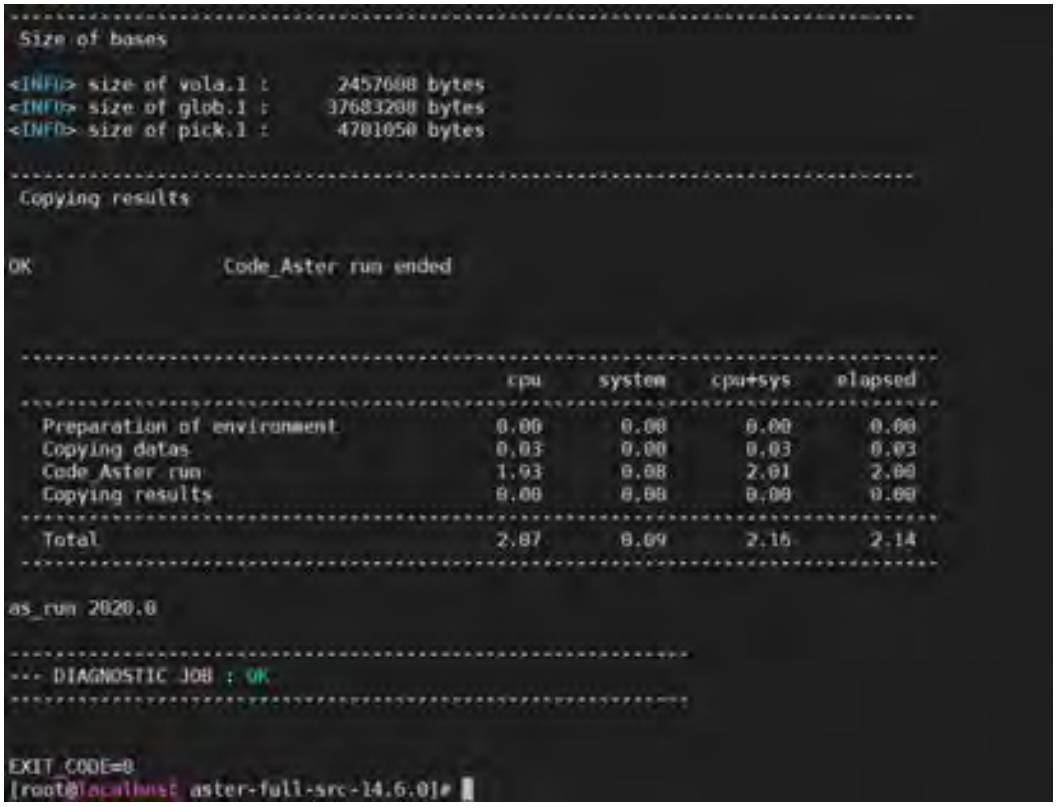
Step 5 Run the following commands to modify the **forma01a.export** file:

1. Open the **forma01a.export** file.
   ```
   vi forma01a.export
   ```
2. Press **i** to enter the insert mode and change the value of **memory_limit** in line 5 to **2048.0**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to execute the configuration file:

```
as_run forma01a.export
```



----End

2.9.8 More Resources

Code_Aster official website:

https://www.code-aster.org

2.10 OpenFOAM 1906 Porting Guide (openEuler 21.03)

2.10.1 Introduction

Open Source Field Operation and Manipulation (OpenFOAM) is the physical field computing software developed by OpenCFD, a subsidiary of ESI. The software analyzes fluid, heat transfer, molecular dynamics, electromagnetic fluid and solid stress, and provides a visualized interface for operations from grid division to post-processing. It supports data pre-processing, post-processing, and solver customization, and is often used for computational fluid dynamics (CFD).

For more information about OpenFOAM, visit the **OpenFOAM official website**.

Programming language: C++

Brief description: Open-source field computing and processing software.

Open source license: GNU General Public License

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

OpenFOAM v1906

2.10.2 Environment Requirements

Hardware Requirements

Table 2-46 lists the hardware requirements.

Table 2-46 Hardware requirements

Item	Description
CPU	Kunpeng 920 7260 processor

Software Requirements

Table 2-47 lists the software requirements.

Table 2-47 Software requirements

Item	Version	Download Address
OpenFOAM	v1906	https://sourceforge.net/projects/openfoam/files/v1906/OpenFOAM-v1906.tgz
ThirdParty	v1906	https://sourceforge.net/projects/openfoam/files/v1906/ThirdParty-v1906.tgz
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
Test cases	tutorials / incompressible/ simpleFoam/ motorBike	Included in the OpenFOAM installation package

OS Requirements

Table 2-48 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-48 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	5.10	Included in the OS image

2.10.3 Paths for Software Porting

This chapter lists the software installation paths involved in the OpenFOAM software porting.

Table 2-49 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	**/path/to/bisheng**	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on your requirements.
2	**/path/to/hmpi**	Installation path of Hyper MPI.	
3	**/path/to/OpenFOAM**	Installation path of OpenFOAM.	

2.10.4 Configuring the Compilation Environment

2.10.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-50 Configuration process

No.	Configuration Item	Remarks
1	Configuring the installation path	See **2.10.4.2 Configuring the Installation Path**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing the BiSheng Compiler	See **2.10.4.3 Installing the BiSheng Compiler**.
3	Installing Hyper MPI	See **2.10.4.4 Installing Hyper MPI**.
4	Installing the Dependencies	See **2.10.4.5 Installing the Dependencies**.

2.10.4.2 Configuring the Installation Path

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Write the installation path to the environment variables.

```
export BISHENG=/path/to/bisheng
export OPENFOAM=/path/to/OpenFOAM
```

NOTE

The installation paths provided here are only an example. Shared paths are recommended. You need to change the installation paths in subsequent commands in this document based on your requirements.

----End

2.10.4.3 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the BiSheng Compiler installation package.

```
tar -zxvf bisheng-compiler-1.3.1-aarch64-linux.tar.gz
```

Step 3 Set the environment variables.

```
export PATH=$BISHENG/bin:$PATH
export INCLUDE=$BISHENG/include:$INCLUDE
export LD_LIBRARY_PATH=$BISHENG/lib:$LD_LIBRARY_PATH
```

NOTE

In the preceding command, **$BISHENG** is the BiSheng Compiler installation path. Change it based on your site requirements.

----End

 Copyright © Huawei Technologies Co., Ltd.

2.10.4.4 Installing Hyper MPI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/hmpi* directory.

```
cd /path/to/hmpi
```

Step 3 Decompress the Hyper MPI installation package.

```
tar -zxvf HMPI_1.0.2_aarch64.tar.gz
```

Step 4 Go to the HUCX source code directory.

```
cd HMPI_1.0.2_aarch64/source/hucx
```

NOTE

Hyper MPI consists of HUCX and Hyper MPI.

Step 5 Compile HUCX.

1. If you use the BiSheng Compiler directly, HUCX cannot be compiled. You need to delete or comment out the content in **src/ucs/arch/aarch64/cpu.h**.

```
static inline void ucs_arch_clear_cache(void *start, void *end)
{
    uintptr_t ptr;
    unsigned ctr_el0, dcache;
    asm volatile ("mrs\t%0, ctr_el0":"=r" (ctr_el0));
    dcache = sizeof(int) << ((ctr_el0 >> 16) & 0xf);

    for (ptr = ucs_align_down((uintptr_t)start, dcache); ptr < (uintptr_t)end; ptr += dcache) {
        asm volatile ("dc cvac, %0" :: "r" (ptr) : "memory");
    }
    asm volatile ("dsb ish" ::: "memory");
}
```

2. Run the following commands:

```
./autogen.sh
CC=clang CXX=clang++ FC=flang F77=flang ./configure --prefix=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hucx --enable-mt --enable-devel-headers
```

NOTE

If the numactl plugin does not exist, add the **--disable-numa** parameter. Otherwise, an error is reported and you need to run the following command:

CC=clang CXX=clang++ FC=flang F77=flang ./configure --prefix=*/path/to/hmpi/HMPI_1.0.2_aarch64/source/hucx* **--enable-mt --enable-devel-headers --disable-numa**

```
make
make install
```

NOTE

If the "limits file not found" error is reported, run the **cd src/tools/perf/lib/** command based on the installation directory:

```
ln -s /path/to/bisheng/1.3.2.b023/include/c++/v1/*
```

Step 6 Go to the HMPI source code directory.

```
cd HMPI_1.0.2_aarch64/source/hmpi
```

Step 7 Compile HMPI.

```
./autogen.pl
CC=clang CXX=clang++ F77=flang F90=flang FC=flang \
```

 Copyright © Huawei Technologies Co., Ltd.

```
./configure --prefix=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hmpi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hucx
make
make install
```

NOTE

- If the "C and C++ compilers are not link compatible" error is reported, check the log as prompted. A possible cause is that the libstdc library is missing. Use Yum to install the library.
  ```
  yum install libstdc++.aarch64
  ```
- For the **libstdc++.so** file problem, install the file first.

Step 8 Set the environment variables.

```
export PATH=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hmpi/bin:$PATH
export LD_LIBRARY_PATH=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hmpi/lib:$LD_LIBRARY_PATH
export OPAL_PREFIX=/path/to/hmpi/HMPI_1.0.2_aarch64/source/hmpi
```

----End

2.10.4.5 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
sudo yum install -y zlib-devel texinfo gstreamer-plugins-base-devel libXext-devel libGLU-devel libXt-devel libXrender-devel libXinerama-devel libpng-devel libXrandr-devel libXi-devel libXft-devel libjpeg-turbo-devel libXcursor-devel readline-devel ncurses-devel mpfr-devel gmp-devel libmpc*
```

----End

2.10.5 Obtaining the Source Code

Procedure

Step 1 Download the OpenFOAM installation package.

Download URLs:

https://sourceforge.net/projects/openfoam/files/v1906/OpenFOAM-v1906.tgz

https://sourceforge.net/projects/openfoam/files/v1906/ThirdParty-v1906.tgz

Step 2 Use an SFTP tool to upload the OpenFOAM installation package to the **/path/to/OpenFOAM** directory on the server.

----End

2.10.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the OpenFOAM installation package directory.

```
cd /path/to/OpenFOAM
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the OpenFOAM installation package.

```
tar -xvf OpenFOAM-v1906.tgz
tar -xvf ThirdParty-v1906.tgz
```

Step 4 Go to the directory generated after the decompression.

```
cd OpenFOAM-v1906
```

Step 5 Create a **List.H.patch** file and edit it.

1. Create a **List.H.patch** file.
   ```
   vi List.H.patch
   ```
2. Press **i** to enter the insert mode and type the following content:
   ```
   --- ./src/OpenFOAM/containers/Lists/List/List.H 2019-11-08 01:12:53.000000000 +0800
   +++ ../OpenFOAM-v1912/src/OpenFOAM/containers/Lists/List/List.H 2019-12-23 17:49:24.000000000
   +0800
   @@ -57,7 +56,6 @@
   {
   // Forward declarations
   -
   class Istream;
   class Ostream;
   @@ -114,6 +112,12 @@
   public:
   +    // Related types
   +
   +        //- Declare type of subList
   +        typedef SubList<T> subList;
   +
   +
   // Static Member Functions
   //- Return a null List
   @@ -201,12 +205,6 @@
   ~List();
   -    // Related types
   -
   -        //- Declare type of subList
   -        typedef SubList<T> subList;
   -
   -
   // Member Functions
   // Edit
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Create an **IListStream.H.patch** file and edit it.

1. Create an **IListStream.H.patch** file.
   ```
   vi IListStream.H.patch
   ```
2. Press **i** to enter the insert mode and type the following content:
   ```
   --- ./src/OpenFOAM/db/IOstreams/memory/IListStream.H    2019-11-08 01:12:53.000000000 +0800
   +++ ../OpenFOAM-v1912/src/OpenFOAM/db/IOstreams/memory/IListStream.H    2019-12-23
   17:49:24.000000000 +0800
   @@ -115,7 +115,7 @@
   reset_gbuffer();
   }
   -        //- Transfer contents to other list
   +        //- Transfer contents to other List
   inline void swap(List<char>& list)
   {
   List<char>::swap(list);
   @@ -128,7 +128,7 @@
   /*---------------------------------------------------------------------------*\
   Class IListStream Declaration
   -\*---------------------------------------------------d---------------------------*/
   +\*---------------------------------------------------------------------------*/
   //- An ISstream with internal List storage
   class IListStream
   @@ -158,7 +158,7 @@
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
//- Move construct from List
IListStream
(
-           List<char>&& buffer,
+           ::Foam::List<char>&& buffer,  // Fully qualify (issue #1521)
streamFormat format=ASCII,
versionNumber version=currentVersion,
const Foam::string& name="input"
@@ -184,7 +184,7 @@
{}
-    // Member functions
+    // Member Functions
//- The current get position in the buffer
using allocator_type::size;
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Apply the patch.

```
patch -p1 < List.H.patch
patch -p1 < IListStream.H.patch
```

Step 8 Modify the compiler and MPI settings in the OpenFOAM installation configuration file.

Change **export WM_COMPILER=/path/to/clang** to **export WM_COMPILER=Clang**.

NOTE

Change **/path/to/clang** to the **clang** file path of the BiSheng Compiler.

Step 9 Modify the compilation parameters.

```
sed -i 's/\-O3/\-O3 \-march=armv8-a/g' /path/to/OpenFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Clang/c++Opt
sed -i 's/\-O3/\-O3 \-march=armv8-a/g' /path/to/OpenFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Clang/cOpt
```

Step 10 Make the OpenFOAM configuration file **bashrc** take effect.

```
source etc/bashrc
```

Step 11 Perform the compilation and installation.

```
./Allwmake -j -k
```

----End

2.10.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/OpenFOAM/tutorials/incompressible/pisoFoam/LES/motorBike/motorBike
```

Step 3 Set the grid precision as required.

```
sed -ri "s/\(20 8 8\)/\(2048 512 64\)/" ./system/blockMeshDict
```

NOTE

You are advised to increase the grid precision when multiple nodes are running.

Step 4 Set the number of running CPU cores.

```
sed -ri "s/8/256/" system/
{decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
sed -ri "s/\(4 2 1\)/\(16 4 4\)/" system/
{decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
```

NOTE

In this example, two nodes are configured and the number of CPU cores is 256 (16 x 4 x 4 = 256). You can change it based on your requirements.

Step 5 Start the running.

- Single node
  ```
  ./Allrun
  ```
  The following figure shows the running result when a single node uses 6 CPU cores. The results of other configurations are similar.

  NOTE

  If you start the running as the root user, modify the **RunFunctions** file (refer to the steps for the multiple-node running). Add **--allow-run-as-root** to the end of the **mpirun** command.

- Dual nodes

  a. Create a **hostfile** file.
  ```
  vi hostfile
  ```

  b. Press **i** to enter the insert mode.
  ```
  Node1
  Node2
  ```

  NOTE

  Node1 and **Node2** are the names of the nodes where the code is executed. Change them based on your requirements and ensure that the nodes can ping each other. Enable SSH password-free login between the nodes.

  c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

  d. Modify the **RunFunctions** file.
  ```
  vi /path/to/OPENFOAM/bin/tools/RunFunctions
  ```

  e. Press **i** to enter the insert mode and modify the file as follows:
  ```
  $mpirun -x PATH=$PATH –x LD_LIBRARY_PATH=$LD_LIBRARY_PATH -x WM_PROJECT_DIR=
  $WM_PROJECT -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
  ```
  So far, the configuration required for multi-node running is complete.

  f. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

  g. Start the running.
  ```
  ./Allrun
  ```

Step 6 After the computing is complete and the software exits, a computing log file is generated. Run the **vi log.simpleFoam** command to open the **log.simpleFoam**

file. Check the value of **ExecutionTime** (unit: second) in the **log.simpleFoam** file. A smaller value indicates better performance.

----End

2.11 Code_Aster 14.6.0 Porting Guide (openEuler 21.03)

2.11.1 Introduction

Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics. It runs on Linux and was public released based on General Public License in October 2001. Code_Aster has now 1,500,000 lines of source code, most of which are developed in Fortran and Python, and is being constantly developed and updated. Most of the Code_Aster fields have been validated by independent comparisons with analytical or experimental results, benchmarks towards other codes.

Code_Aster is a solver, based on the theory of the mechanics of the continuous media, which uses the method of the finite elements to solve different types of mechanical, thermal, acoustic, seismic, and other problems. Beyond the standard functionalities of a FEM software for solid mechanics, Code_Aster also compiles specific research in various fields: fatigue, damage, fracture, contact, geomaterials, porous media, and multiphysics coupling.

Currently, the Code_Aster software has been widely used to evaluate and maintain power plants and power networks, and to analyze and calculate bridge data.

Salome is an open-source software that provides a generic platform for pre- and post-processing for numerical simulation. Salome-Meca is a standalone application, which is an external project based on the SALOME platform. Salome-Meca represents integration of the Code_Aster solver in the SALOME platform.

Programming languages: C/Fortran/Python

Brief description: Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics.

Open source license: GPL

Recommended Software Version

Code_Aster 14.6.0

2.11.2 Environment Requirements

Hardware Requirements

Table 2-51 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-51 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 2-52 lists the software requirements.

Table 2-52 Software requirements

Item	Version	Download URL
Code_Aster	14.6.0	**https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz**
GCC for openEuler	9.3.1	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2020.12-aarch64-linux.tar.gz**
Hyper MPI	1.1.0	**https://www.hikunpeng.com/en/developer/hpc/hypermpi**

OS Requirements

Table 2-53 lists the OS requirements.

Table 2-53 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://repo.openeuler.org/openEuler-21.03/**
Kernel	5.10.1	Included in the OS image

2.11.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Code_Aster software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-54 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/HMPI	Installation path of Hyper MPI.	
3	/path/to/CODE-ASTER	Installation path of Code_Aster.	

2.11.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-55 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.

2.11.5 Obtaining the Source Code

Procedure

Step 1 Download the Code_Aster installation package **aster-full-src-14.6.0-1.noarch.tar.gz**.

https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz

Step 2 Use an SFTP tool to upload the Code_Aster installation package to the */path/to/CODE-ASTER* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

2.11.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum -y install zlib* lapack* blas* python3.aarch64 python3-devel.aarch64 cmake3 boost-python* boost boost-devel numpy* python-numpy flex bison tcl tk
```

Step 3 Go to the installation directory of the main program, decompress the installation package, and go to the directory.

```
cd /path/to/CODE-ASTER
tar -xvf aster-full-src-14.6.0-1.noarch.tar.gz
cd aster-full-src-14.6.0
```

Step 4 Modify the **setup.py** file.

1. Open **setup.py**.
   ```
   vi setup.py
   ```
2. Press **i** to go to the insert mode and add the **aarch64** option to line 354.

```
348     if plt == 'win32':
349         cfg['IFDEF'] = 'WIN32'
350     elif plt in ('linux', 'cygwin'):
351         cfg['ARCH'] = 'x86'
352         if machine.endswith('64'):
353             cfg['IFDEF'] = 'LINUX64'
354             if machine in ('x86_64', 'ia64', 'ppc64','aarch64'):
355                 cfg['ARCH'] = machine
356             else:  # force to x86_64
357                 cfg['ARCH'] = 'x86_64'
358         else:
359             cfg['IFDEF'] = 'LINUX'
360     elif plt == 'darwin':
```

NOTE

After performing Step 4.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Modify the **as_setup.py** file.

1. Open **as_setup.py**.
   vi as_setup.py
2. Press **i** to go to the insert mode and add the **aarch64** option to line 1538.
   'aarch64': 'shell|script|aarch64',

```
1533        'x86'    : 'shell|script|80.86',
1534        'i86pc'  : 'shell|script|80.86',
1535        'x86_64' : 'shell|script|x86.64',
1536        'ppc64'  : 'shell|script|64-bit PowerPC',
1537        'ia64'   : 'shell|script|ia.64',
1538        'aarch64': 'shell|script|aarch64',
1539    }
1540 #------------------------------------------------------------
1541    def __init__(self, **kargs):
1542        self.print      = kargs.get('log', None) or print
```

NOTE

Note that the number of indented spaces in each statement must be the same.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Modify the **setup.cfg** file.

1. Open **setup.cfg**.
   vi setup.cfg
2. Press **i** to go to the insert mode and add the following content to line 94.
   CC='/opt/aarch64/compiler/gcc-9.3.1-2020.12-aarch64-linux/gcc'
   CXX='/opt/aarch64/compiler/gcc-9.3.1-2020.12-aarch64-linux/g++'
   F90='/opt/aarch64/compiler/gcc-9.3.1-2020.12-aarch64-linux/gfortran'
   LD=F90
   CXXFLAGS="-std=c++11"
   CXXLIB='-L/opt/aarch64/compiler/gcc-9.3.1-2020.12-aarch64-linux/lib64 -lstdc++'

NOTE

Modify the file based on the actual installation location of GCC for openEuler.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the **products.py** file.

1. Open **products.py**.
   vi products.py
2. Press **i** to enter the insert mode and modify the content in line 187.
   conf_cmd = ("export F77=$F90; "

```
184     ldflags = '-Wl,--no-as-needed '
185 ldflags += cfg['OTHERLIB'] + ' ' + cfg['CXXLIB']
186 disable_shared = ''
187 conf_cmd = ("export F77=$F90; "
188              "export CXXFLAGS='-std=gnu++98'; "
189              "{4} ; "
190              "./configure {1} "
191              "--disable-mesgerr --with-hdf5={2} --prefix={3}"
192              .format(ldflags, disable_shared, cfg['HOME_HDF'],
193                      cfg['HOME_MED'], cfg['PYTHON_ALIAS']))
194
195 # ----- setup instance
196 setup=SETUP(
197     product=product,
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

Note that the number of indented spaces in each statement must be the same.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Go to the Code_Aster source package and decompress the ASTER installation package.

```
cd SRC/
tar xvf aster-14.6.0.tgz
```

Step 9 Modify the **/bibfor/wscript** file.

1. Open **/bibfor/wscript**.
   ```
   vi aster-14.6.0/bibfor/wscript
   ```
2. Press **i** to enter the insert mode and modify the content in line 103.
   ```
   if self.env.DEST_CPU == 'x86_64' or self.env.DEST_CPU == 'aarch64':
   ```

```
def check_fortran_compiler_flags(self):
    if 'gfortran' in self.env.FC_NAME.lower():
        if self.env.DEST_CPU == 'x86_64' or self.env.DEST_CPU == 'aarch64':
            flags = ['-fdefault-%s-%i' % (name, 8)
                     for name in ('double', 'integer', 'real')]
            self.check_fortran_compiler_options(flags)
        self.check_fortran_compiler_options('-Wimplicit-interface')
        self.check_fortran_compiler_options('-Wintrinsic-shadow')
```

NOTE

Note that the number of indented spaces in each statement must be the same.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Modify the **/bibpyt/Utilitai/Utmess.py** file.

1. Open **/bibpyt/Utilitai/Utmess.py**.
   ```
   vi ./aster-14.6.0/bibpyt/Utilitai/Utmess.py
   ```
2. Press **i** to enter the insert mode and modify the content in line 224:
   ```
   curr_idmess = idmess.split("\0")
   x = curr_idmess[0].split("_")
   ```

```
222         # décodage : idmess -> (catamess, numess)
223         idmess = idmess.strip()
224         curr_idmess = idmess.split("\0")
225         x = curr_idmess[0].split("_")
226         assert len(x) > 1, idmess
227         catamess = '_'.join(x[0:-1]).lower()
228         numess = int(x[-1])
229         assert numess > 0 and numess < 100, idmess
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

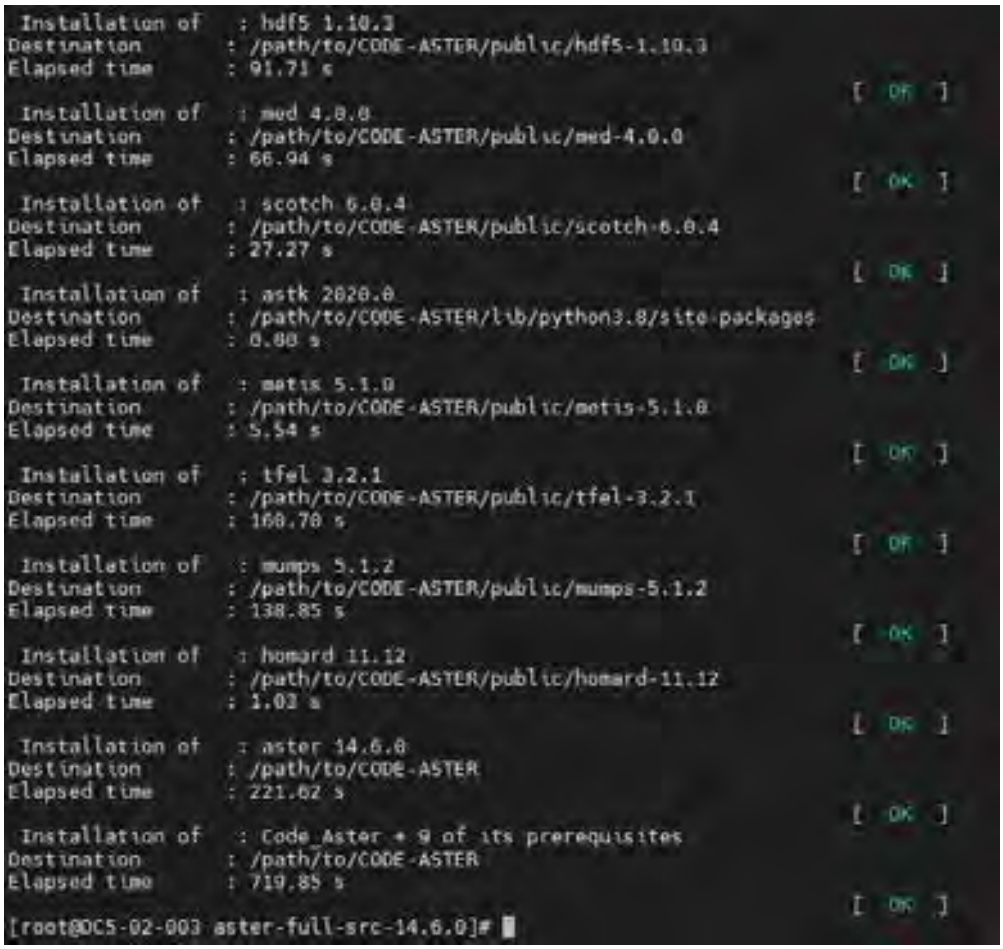
Step 11 Update the ASTER source package and delete temporary files.

```
tar cvzf aster-14.6.0.tgz aster-14.6.0
rm -rf aster-14.6.0
```

Step 12 Perform the compilation.

```
cd /path/to/CODE-ASTER/aster-full-src-14.6.0
echo y | python3 setup.py --prefix=/path/to/CODE-ASTER
```

 Copyright © Huawei Technologies Co., Ltd.



----End

2.11.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **CODE-ASTER** directory.

```
cd /path/to/CODE-ASTER
```

Step 3 Set the environment variables.

```
source /path/to/CODE-ASTER/14.6/share/aster/profile.sh
export PATH=/path/to/CODE-ASTER/bin/:$PATH
```

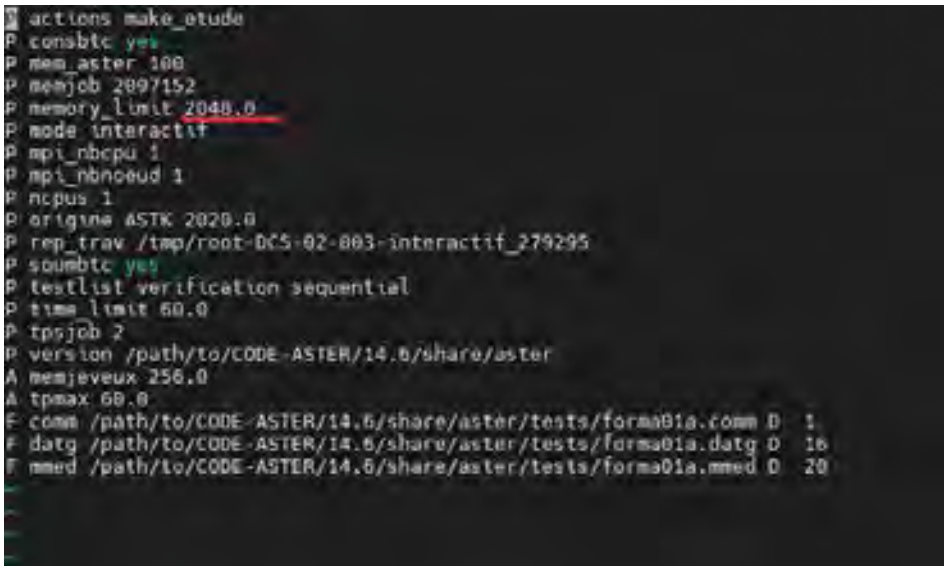
Step 4 Obtain the **forma01a.export** file.

```
as_run --get_export forma01a > forma01a.export
```

Step 5 Modify the **forma01a.export** file.

1. Open the **forma01a.export** file.
   ```
   vi forma01a.export
   ```
2. Change the value of **memory_limit** in line 5 to **2048.0**.

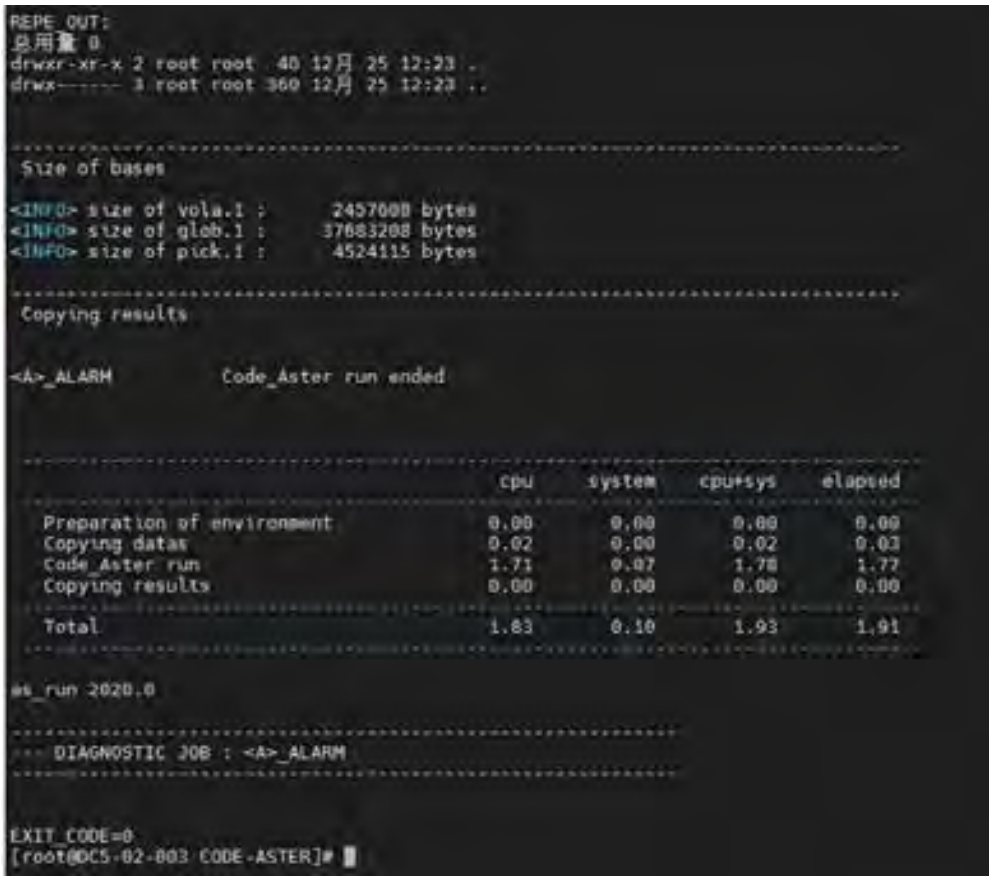
 Copyright © Huawei Technologies Co., Ltd.



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Execute the configuration file.

```
as_run forma01a.export
```



NOTE

If the command output contains **<A>_ALARM**, the CODE-ASTER version is too early and the system automatically prompts you to update the version.

 Copyright © Huawei Technologies Co., Ltd.

Meaning of the alarm: You are using an early version of CODE-ASTER. Please upgrade it to the latest version to use the enhanced features.

----End

2.12 OpenLB 1.4-0 Porting Guide (GCC for openEuler)

2.12.1 Introduction

The OpenLB project provides a C++ package for the implementation of Lattice Boltzmann methods that is general enough to address a vast range of transport problems, e.g. in computational fluid dynamics. The source code is publicly available and constructed in a well readable, modular way. This allows for a fast implementation of both simple academic test problems and advanced engineering applications. It is also easily extensible to include new physical content.

For more information about OpenLB, visit the **OpenLB official website**.

Programming language: C++

Brief description: A C++ package for the implementation of Lattice Boltzmann methods.

Open source license: GNU General Public License V2

Recommended Software Version

OpenLB 1.4-0

2.12.2 Environment Requirements

Hardware Requirements

Table 2-56 lists the hardware requirements.

Table 2-56 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 2-57 lists the software requirements.

Table 2-57 Software requirements

Item	Version	Download Address
OpenLB	1.4-0	**https://www.openlb.net/download/**
GCC for openEuler	9.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz**
HUCX source package	1.1.1	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.1.1-huawei.tar.gz**
XUCG source package	1.1.1	**https://github.com/kunpengcompute/xucg/archive/refs/tags/v1.1.1-huawei.tar.gz**
Hyper MPI source package	1.1.1	**https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz**

OS Requirements

Table 2-58 lists the OS requirements.

Table 2-58 OS requirements

Item	Version	Download Address
Kylin OS	Kylin V10 SP2	**https://www.kylinos.cn**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Provided by the OS.

2.12.3 Paths for Software Porting

This chapter lists software installation paths involved in the OpenLB software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-59 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/aarch64/KPGCC	Installation path of the GCC for openEuler compiler.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/download/hmpi	Download path of Hyper MPI.	
3	/path/to/download/hucx	Download path of HUCX.	
4	/path/to/install/hmpi	Installation path of Hyper MPI.	
5	/path/to/install/hucx	Installation path of HUCX.	

2.12.4 Configuring the Compilation Environment

2.12.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-60 Configuration process

No.	Configuration Item	Remarks
1	Configuring the GCC for openEuler compilation environment	See **2.12.4.2 Configuring the GCC for openEuler Compilation Environment**.
2	Installing Hyper MPI	See "Prerequisites" and "Compiling the Hyper MPI Software Package Using GCC" in the "Installing Hyper MPI Using Source Code" section of the **Hyper MPI User Guide**

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Setting environment variables	See 2.12.4.3 Setting Environment Variables.

2.12.4.2 Configuring the GCC for openEuler Compilation Environment

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the GCC for openEuler software package and upload it to the target server.

https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz

Step 3 Create an installation directory (**/opt/aarch64/KPGCC** is used as an example):

```
mkdir –p /opt/aarch64/KPGCC
```

Step 4 Copy the GCC for openEuler software package to the installation directory.

```
cp -rf gcc-9.3.1-2021.09-aarch64-linux.tar.gz /opt/aarch64/KPGCC
```

Step 5 Decompress the GCC for openEuler software package.

```
tar -zxvf gcc-9.3.1-2021.09-aarch64-linux.tar.gz
```

Step 6 Set the environment variables of the compiler.

```
export PATH=/opt/aarch64/KPGCC/gcc-9.3.1-2021.09-aarch64-linux/bin:$PATH
export INCLUDE=/opt/aarch64/KPGCC/gcc-9.3.1-2021.09-aarch64-linux/include:$INCLUDE
export LD_LIBRARY_PATH=/opt/aarch64/KPGCC/gcc-9.3.1-2021.09-aarch64-linux/
lib64:$LD_LIBRARY_PATH
```

Step 7 Verify the GCC for openEuler version.

```
gcc -v
```

If the command output contains the GCC for openEuler version information, the installation is successful.

----End

2.12.4.3 Setting Environment Variables

Procedure

Set the environment variables. The setting takes effect temporarily. To make the setting permanently valid, add the following command to the **~/.bashrc** file:

```
export PATH=/path/to/install/hmpi/bin:/path/to/install/hucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/hmpi/lib:/path/to/install/hucx/lib$LD_LIBRARY_PATH
```

2.12.5 Installing OpenLB

 Copyright © Huawei Technologies Co., Ltd.

2.12.5.1 Installing OpenLB

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the OpenLB installation package **olb-1.4r0.tgz**.

https://www.openlb.net/download/

Step 3 Upload the installation package to the installation directory on the server.

Step 4 Decompress the installation package.

```
tar –zxvf olb-1.4r0.tgz
```

Step 5 Go to the directory generated after the decompression.

```
cd olb-1.4r0
```

Step 6 Install OpenLB.

```
make -j4 samples
```

If a large amount of compilation information is displayed and no error is reported, the installation is successful.

----End

2.12.5.2 Modifying the Configuration File

By default, the program performs non-parallel computing. To perform parallel computing, modify the configuration file.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the **config.mk** file in the **olb-1.4r0** directory.

```
vim config.mk
```

Step 3 Press **i** to enter the insert mode and modify the file as follows:

```
#CXX    := g++          # Default
CXX    := mpicxx
#CC      := gcc          # Default
CC      := mpicc
OPTIM    := -03 -Wall –march=native –mtune=native
DEBUG    := -g –Wall –DOLB_DEBUG
CXXFLAGS    := $(OPTIM)
CXXFLAGA    += -std=c++14
ARPRG    := ar
LDFLAGS    :=
#PARALLEL_MODE    := OFF
PARALLEL_MODE    := HYBRID
MPIFLAGS    :=
OMPFLAGS    := -fopenmp
BUILDTYPE    := generic
FEATURES    :=
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

 Copyright © Huawei Technologies Co., Ltd.

2.12.5.3 Running Test Cases

2.12.5.3.1 Overview

All test cases can be compiled by running the **make samples** command. For details about how to execute an independent test case, for example, **cylinder2d**, see **2.12.5.3.2 Non-parallel Computing** and **2.12.5.3.3 Parallel Computing**.

2.12.5.3.2 Non-parallel Computing

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where the test case is stored.

```
cd examples/laminar/cylinder2d
```

Step 3 Compile the test case.

```
make
```

Step 4 Run the program to perform the computing.

```
./cylinder2d
```

The generated VTK and gnuplot results are stored in the **/tmp** directory.

----End

2.12.5.3.3 Parallel Computing

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where the test case is stored.

```
cd examples/laminar/cylinder2d
```

Step 3 Compile the test case.

```
make
```

Step 4 Run the program to perform the computing. The following uses **-np=64** and **OMP_NUM_THREADS=1** as an example.

```
mpirun –allow-run-as-root –np 64 –x OMP_NUM_THREADS=1 ./cylinder2d
```

The generated VTK and gnuplot results are stored in the **/tmp** directory.

----End

2.12.5.4 (Optional) Visualization Tool

You can use visualization tools, such as Paraview and gnuplot, to read the VTK file generated by OpenLB. In the following example, gnuplot is installed.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Install gnuplot using Yum.

```
yum install gnuplot
```

- If an error message is displayed, for example, "lua.h no such file or directory", perform **Step 3** to **Step 7**.
- If the installation is successful, no further action is required.

Step 3 Check the Kylin OS Yum source.

```
vim /etc/yum.repo.d
```

Step 4 Press **i** to enter the insert mode and check whether **baseurl** is the following address. If it is not, correct it.

```
baseurl = http://update.cs2c.com.cn/NS/V10/V10SP2/os/adv/lic/base/$basearch/
```

Step 5 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Reinstall gnuplot.

```
yum install gnuplot
```

Step 7 If the installation still fails, manually install gnuplot and the dependencies including lua and readline-devel.

----End

2.13 OpenLB 1.4-0 Porting Guide (BiSheng Compiler)

2.13.1 Introduction

The OpenLB project provides a C++ package for the implementation of Lattice Boltzmann methods that is general enough to address a vast range of transport problems, e.g. in computational fluid dynamics. The source code is publicly available and constructed in a well readable, modular way. This allows for a fast implementation of both simple academic test problems and advanced engineering applications. It is also easily extensible to include new physical content.

For more information about OpenLB, visit the **OpenLB official website**.

Programming language: C++

Brief description: A C++ package for the implementation of Lattice Boltzmann methods.

Open source license: GNU General Public License V2

Recommended Software Version

OpenLB 1.4-0

2.13.2 Environment Requirements

Hardware Requirements

Table 2-61 lists the hardware requirements.

Table 2-61 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 2-62 lists the software requirements.

Table 2-62 Software requirements

Item	Version	Download Address
OpenLB	1.4-0	**https://www.openlb.net/download/**
BiSheng Compiler	2.1.0	**https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng**
HUCX source package	1.1.1	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.1.1-huawei.tar.gz**
XUCG source package	1.1.1	**https://github.com/kunpengcompute/xucg/archive/refs/tags/v1.1.1-huawei.tar.gz**
Hyper MPI source package	1.1.1	**https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz**

OS Requirements

Table 2-63 lists the OS requirements.

Table 2-63 OS requirements

Item	Version	Download Address
Kylin OS	Kylin V10 SP2	**https://www.kylinos.cn**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Provided by the OS.

2.13.3 Paths for Software Porting

This chapter lists software installation paths involved in the OpenLB software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-64 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/opt/aarch64/compiler	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/download/hmpi	Download path of Hyper MPI.	
3	/path/to/download/hucx	Download path of HUCX.	
4	/path/to/install/hmpi	Installation path of Hyper MPI.	
5	/path/to/install/hucx	Installation path of HUCX.	

2.13.4 Configuring the Compilation Environment

2.13.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-65 Configuration process

No.	Configuration Item	Remarks
1	Configuring the BiSheng compiler environment	See **2.13.4.2 Configuring the BiSheng Compiler Environment**.
2	Installing Hyper MPI	See **2.13.4.3 Installing Hyper MPI**.
3	Setting environment variables	See **2.13.4.4 Setting Environment Variables**.

 Copyright © Huawei Technologies Co., Ltd.

2.13.4.2 Configuring the BiSheng Compiler Environment

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the BiSheng Compiler software package and upload it to the target server.

https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng

Step 3 Create an installation directory (**/opt/aarch64/compiler** is used as an example):

```
mkdir –p /opt/aarch64/compiler
```

Step 4 Copy the BiSheng Compiler software package to the installation directory.

```
cp bisheng-compiler-2.1.0-aarch64-linux.tar.gz/opt/aarch64/compiler
```

Step 5 Decompress the BiSheng Compiler software package.

```
tar –zxvf bisheng-compiler-2.1.0-aarch64-linux.tar.gz
```

Step 6 Set the environment variables of the compiler.

```
export PATH=/opt/aarch64/compiler/bisheng-compiler-2.1.0-aarch64-linux/bin:$PATH
export INCLUDE=/opt/aarch64/compiler/bisheng-compiler-2.1.0-aarch64-linux/include:$INCLUDE
export LD_LIBRARY_PATH=/opt/aarch64/compiler/bisheng-compiler-2.1.0-aarch64-linux/
lib64:$LD_LIBRARY_PATH
```

Step 7 (Optional) Clear the hash table in the current window.

NOTE

If another version of LLVM compiler exists in the system, run the following command immediately after installing the BiSheng Compiler to prevent the **clang** command from being captured by the hash algorithm. If the **clang** command is captured by the hash algorithm, the BiSheng Compiler or the open-source LLVM compiler cannot be used.

```
hash -r
```

Step 8 Check the BiSheng Compiler version.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

2.13.4.3 Installing Hyper MPI

Prerequisites

The following dependencies have been installed:

- perl-Data-Dumper
- autoconf
- automake
- libtool 2.4.2
- BiSheng Compiler 2.1.0
- numactl

 Copyright © Huawei Technologies Co., Ltd.

- binutils
- systemd-devel
- valgrind

Compiling HUCX

Step 1 Use PuTTY to log in to the server as a Hyper MPI installation and maintenance user, for example, **hmpi_master**.

Step 2 Download the HUCX source package **hucx-1.1.1-huawei.tar.gz**.

https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.1.1-huawei.tar.gz

Step 3 Download the XUCG source package **xucg-1.1.1-huawei.tar.gz**.

https://github.com/kunpengcompute/xucg/archive/refs/tags/v1.1.1-huawei.tar.gz

Step 4 Upload the two source packages to the server.

Step 5 Decompress the two source packages.

```
tar -zxvf hucx-1.1.1-huawei.tar.gz
tar -zxvf xucg-1.1.1-huawei.tar.gz
```

Step 6 Go to the directory where the HUCX source package is stored, for example, **/path/to/download/hucx**.

```
cd /path/to/download/hucx
```

Step 7 Copy the content of the XUCG source package to the **src/ucg** directory in the HUCX source package.

```
cp -r xucg-1.1.1-huawei/* hucx-1.1.1-huawei/src/ucg
```

Step 8 Go to the directory where the **autogen.sh** script is stored.

```
cd hucx-1.1.1-huawei/
```

Step 9 Install the source package.

```
./autogen.sh
./contrib/configure-opt --prefix=/path/to/install/hucx CC=clang CXX=clang++ FC=flang
```

NOTE

If the **numactl** plugin has not been installed, specify the **--disable-numa** parameter.

```
./contrib/configure-opt --prefix=/path/to/install/hucx --disable-numa CC=clang CXX=clang++ FC=flang
```

```
for file in `find . -name Makefile`;do sed -i "s/-Werror//g" $file;done
for file in `find . -name Makefile`;do sed -i "s/-implicit-function-declaration//g" $file;done
make -j32
make -j32 install
```

NOTE

The **-j** parameter following **make** is a parallel compilation parameter, indicating the number of CPU cores involved in the compilation process. Specify this parameter based on the actual number of CPU cores.

Step 10 (Optional) The BiSheng Compiler depends on the system C++ library. If the system C++ library is incomplete, the compilation may fail. If an error message is displayed indicating that the header file (for example **limits.h**) cannot be found during the compilation, install the C++ library.

 Copyright © Huawei Technologies Co., Ltd.

```
yum install gcc*
yum install gcc-*
```

Alternatively, add a compilation option.

```
make –stdlib=libstdc++
```

Step 11 (Optional) If the error message "clang-12 error: linker command failed with exit code 1" is displayed during the compilation, run the following command:

```
#make –stdlib=libstdc++
```

----End

Compiling Hyper MPI

Step 1 Use PuTTY to log in to the server as a Hyper MPI installation and maintenance user, for example, **hmpi_master**.

Step 2 Download the Hyper MPI source package **hmpi-1.1.1-huawei.tar.gz**.

https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz

Step 3 Upload the source package to the server.

Step 4 Decompress the software package.

```
tar -zxvf hmpi-1.1.1-huawei.tar.gz
```

Step 5 Go to the directory where the source package is stored, for example, **/path/to/download/hmpi**.

```
cd /path/to/download/hmpi
```

Step 6 Go to the directory where the **autogen.pl** script is stored.

```
cd hmpi-1.1.1-huawei/
```

Step 7 Install the Hyper MPI source package.

```
./autogen.pl
./configure --prefix=/path/to/install/hmpi --enable-mpi1-compatibility --with-ucx=/path/to/install/hucx
CC=clang CXX=clang++ FC=flang
make
make install
```

----End

2.13.4.4 Setting Environment Variables

Procedure

Set the environment variables. The setting takes effect temporarily. To make the setting permanently valid, add the following command to the **~/.bashrc** file:

```
export PATH=/path/to/install/hmpi/bin:/path/to/install/hucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/hmpi/lib:/path/to/install/hucx/lib$LD_LIBRARY_PATH
```

2.13.5 Installing OpenLB

 Copyright © Huawei Technologies Co., Ltd.

2.13.5.1 Installing OpenLB

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the OpenLB installation package **olb-1.4r0.tgz**.

https://www.openlb.net/download/

Step 3 Upload the installation package to the installation directory on the server.

Step 4 Decompress the installation package.

```
tar –zxvf olb-1.4r0.tgz
```

Step 5 Go to the directory generated after the decompression.

```
cd olb-1.4r0
```

Step 6 Install OpenLB.

```
make -j4 samples
```

If a large amount of compilation information is displayed and no error is reported, the installation is successful.

----End

2.13.5.2 Modifying the Configuration File

By default, the program performs non-parallel computing. To perform parallel computing, modify the configuration file, that is, add the LDFLAGS parameter to the BiSheng Compiler.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the **config.mk** file in the **olb-1.4r0** directory.

```
vim config.mk
```

Step 3 Press **i** to enter the insert mode and modify the file as follows:

```
#CXX    := g++          # Default
CXX    := mpicxx
#CC      := gcc          # Default
CC      := mpicc
#OPTIM    := -03 -Wall –march=native –mtune=native      # Default
OPTIM    := -03 -Wall
DEBUG    := -g –Wall –DOLB_DEBUG
CXXFLAGS    := $(OPTIM)
CXXFLAGA    += -std=c++14
ARPRG    := ar
LDFLAGS    := -Wl,-z,muldefs      # Allow repeated defining.
#PARALLEL_MODE    := OFF
PARALLEL_MODE    := HYBRID
MPIFLAGS    :=
OMPFLAGS    := -fopenmp
BUILDTYPE    := generic
FEATURES    :=
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

 Copyright © Huawei Technologies Co., Ltd.

2.13.5.3 Running Test Cases

2.13.5.3.1 Non-parallel Computing

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where the test case is stored.

```
cd examples/laminar/cylinder2d
```

Step 3 Compile the test case.

```
make
```

Step 4 Run the program to perform the computing.

```
./cylinder2d
```

The generated VTK and gnuplot results are stored in the **/tmp** directory.

----End

2.13.5.3.2 Parallel Computing

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where the test case is stored.

```
cd examples/laminar/cylinder2d
```

Step 3 Compile the test case.

```
make
```

Step 4 Run the program to perform the computing. The following uses **-np=64** and **OMP_NUM_THREADS=1** as an example.

```
mpirun –allow-run-as-root –np 64 –x OMP_NUM_THREADS=1 ./cylinder2d
```

The generated VTK and gnuplot results are stored in the **/tmp** directory.

----End

2.13.5.4 (Optional) Visualization Tool

You can use visualization tools, such as Paraview and gnuplot, to read the VTK file generated by OpenLB. In the following example, gnuplot is installed.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install gnuplot using Yum.

```
yum install gnuplot
```

- If an error message is displayed, for example, "lua.h no such file or directory", perform **Step 3** to **Step 7**.
- If the installation is successful, no further action is required.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Check the Kylin OS Yum source.

```
vim /etc/yum.repo.d
```

Step 4 Press **i** to enter the insert mode and check whether **baseurl** is the following address. If it is not, correct it.

```
baseurl = http://update.cs2c.com.cn/NS/V10/V10SP2/os/adv/lic/base/$basearch/
```

Step 5 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Reinstall gnuplot.

```
yum install gnuplot
```

Step 7 If the installation still fails, manually install gnuplot and the dependencies including lua and readline-devel.

----End

2.14 Kokkos 3.5.00 Porting Guide (Kylin V10)

2.14.1 Introduction

Kokkos Core implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms. For that purpose it provides abstractions for both parallel execution of code and data management. Kokkos is designed to target complex node architectures with N-level memory hierarchies and multiple types of execution resources. It currently can use CUDA, HIP, SYCL, HPX, OpenMP and C++ threads as backend programming models with several other backends in development.

For more information, visit the **Kokkos page at GitHub**.

Programming language: C++

Brief description: Writes performance portable applications targeting all major HPC platforms.

Open source license: custom open source license

Recommended Software Version

Kokkos 3.5.00

2.14.2 Environment Requirements

Hardware Requirements

Table 2-66 lists the hardware requirements.

Table 2-66 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 2-67 Software requirements

Item	Version	Download URL
Kokkos	3.5.00	https://github.com/kokkos/kokkos/archive/refs/tags/3.5.00.tar.gz
CMake	3.21.1	https://github.com/Kitware/CMake/archive/refs/tags/v3.21.1.tar.gz

OS Requirements

Table 2-68 lists the OS requirements.

Table 2-68 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

2.14.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Kokkos software porting.

Table 2-69 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.
2	/path/to/KOKKOS	Installation path of Kokkos	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/CMAKE	Installation path of CMake	paths planned during the installation process.

2.14.4 Configuring the Compilation Environment

2.14.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-70 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
4	Installing CMake	See **2.14.4.2 Installing CMake**.

2.14.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/CMAKE
wget https://github.com/Kitware/CMake/archive/refs/tags/v3.21.1.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf v3.21.1.tar.gz
```

Step 4 Go to the directory generated after decompression.

```
cd cmake-3.21.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Install the software.

```
./bootstrap --prefix=/path/to/CMAKE
make -j
make install
```

Step 6 Set the environment variable.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

2.14.5 Obtaining the Source Code

Procedure

Step 1 Download the Kokkos installation package **3.5.00.tar.gz**.

https://github.com/kokkos/kokkos/archive/refs/tags/3.5.00.tar.gz

Step 2 Use an SFTP tool to upload the Kokkos installation package to the */path/to/KOKKOS* directory on the server.

----End

2.14.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/KOKKOS
```

Step 3 Decompress the installation package.

```
tar xvf 3.5.00.tar.gz
```

Step 4 Go to the directory generated after decompression.

```
cd kokkos-3.5.00
```

Step 5 Create a build directory.

```
mkdir build && cd build
```

Step 6 Create a build file.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/kokkos/install/KOKKOS -DKokkos_ARCH_HSW=On -
DKokkos_ENABLE_TESTS=On
```

Step 7 Modify the build file.

```
cd ..
sed -i "s/\-march=core\-avx2/\-march=armv8\-a/g" `grep -rl "\-march=core\-avx2" ./`
sed -i "s/\-mtune=core\-avx2/\-mtune=tsv110/g" `grep -rl "\-mtune=core\-avx2" ./`
cd build
```

Step 8 Perform the compilation and installation.

```
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

2.14.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to perform the verification. If no error information is displayed, the verification is successful.

```
make test
```

----End

2.15 Pencil Code 2021.02.20 Porting Guide (Kylin V10)

2.15.1 Introduction

The Pencil Code is primarily designed to deal with weakly compressible turbulent flows, which is why we use high-order first and second derivatives. To achieve good parallelization, we use explicit (as opposed to compact) finite differences. Typical scientific targets include driven MHD turbulence in a periodic box, convection in a slab with non-periodic upper and lower boundaries, a convective star embedded in a fully nonperiodic box, accretion disc turbulence in the shearing sheet approximation, self-gravity, non-local radiation transfer, dust particle evolution with feedback on the gas, etc. A range of artificial viscosity and diffusion schemes can be invoked to deal with supersonic flows. For direct simulations regular viscosity and diffusion is being used.

For more information, visit the **Pencil Code page at GitHub**.

Programming language: Fortran

Brief description: A high-order finite-difference code for compressible hydrodynamic flows with magnetic fields and particles.

Open source license: custom open source license

Recommended Software Version

Pencil Code 2021.02.20

2.15.2 Environment Requirements

Hardware Requirements

Table 2-71 lists the hardware requirements.

Table 2-71 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 2-72 Software requirements

Item	Version	Download URL
Pencil Code	2021.02.20	https://github.com/pencil-code/pencil-code/archive/refs/tags/v2021.02.20.tar.gz

OS Requirements

Table 2-73 lists the OS requirements.

Table 2-73 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

2.15.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Pencil Code software porting.

Table 2-74 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/pencil-code	Installation path of the Pencil Code	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

2.15.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-75 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

2.15.5 Obtaining the Source Code

Procedure

Step 1 Download the Pencil Code installation package **v2021.02.20.tar.gz**.

https://github.com/pencil-code/pencil-code/archive/refs/tags/v2021.02.20.tar.gz

Step 2 Use an SFTP tool to upload the Pencil Code installation package to the */path/to/pencil-code* directory on the server.

----End

2.15.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/pencil-code
```

Step 3 Decompress the installation package.

```
tar -xvf v2021.02.20.tar.gz
mv pencil-code-2021.02.20 pencil-code
```

Step 4 Go to the directory generated after the decompression.

```
cd pencil-code
source sourceme.sh
```

Step 5 Create a run directory and copy the configuration file.

```
cd ..
mkdir -p data/myuser/myrun/src
```

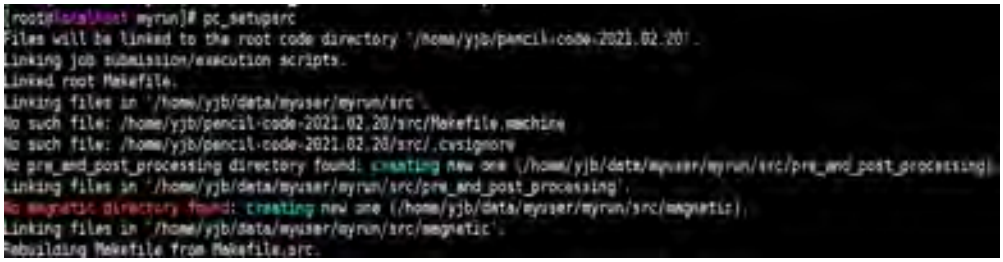
 Copyright © Huawei Technologies Co., Ltd.

```
cd data/myuser/myrun
cp $PENCIL_HOME/samples/1d-tests/jeans-x/*.in ./
cp $PENCIL_HOME/samples/1d-tests/jeans-x/src/*.local src/
```

Step 6 Set the symbolic link to the original Pencil Code directory.

```
pc_setupsrc
```

Information similar to the following is displayed:



Step 7 Modify the configuration file.

```
cd $PENCIL_HOME/config/compilers/separate/
sed -i 's/gcc/clang/g' GNU-gcc.conf
sed -i 's/gfortran/flang/g' GNU-gfortran.conf
```

Step 8 Perform the compilation.

```
cd /path/to/data/myuser/myrun
pc_build
```

----End

2.15.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

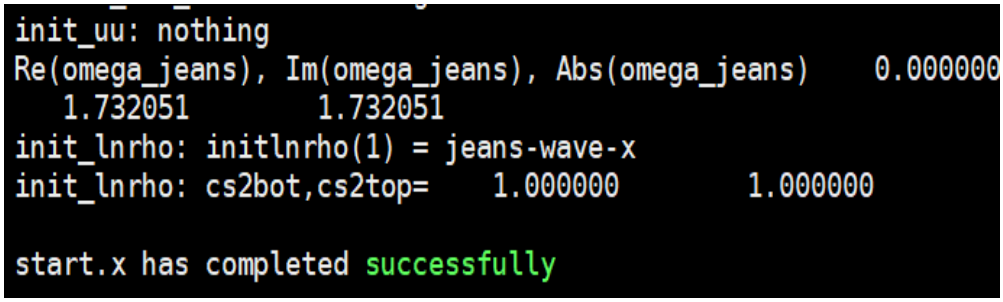
Step 2 Create an empty data directory.

```
cd /path/to/data/myuser/myrun
mkdir data
cd data
```

Step 3 Start the running.

```
pc_run
```

Information similar to the following is displayed:



After the running is complete, an empty **COMPLETED** file is generated in the current directory.

 Copyright © Huawei Technologies Co., Ltd.

```
Sat Apr  9 09:27:55 2022
Running
  pc_deprecated_slice_links
[root@localhost myrun]# ll
total 56
-rw-r--r-- 1 root root     0 Apr  9 09:27 COMPLETED
```

----End

2.16 SPARTA 7Jan2022 Porting Guide (Kylin V10)

2.16.1 Introduction

SPARTA stands for Stochastic PArallel Rarefied-gas Time-accurate Analyzer. SPARTA is a parallel DSMC or Direct Simulation Monte Carlo code for performing simulations of low-density gases in 2D or 3D. Particles advect through a hierarchical Cartesian grid that overlays the simulation box. The grid is used to group particles by grid cell for purposes of performing collisions and chemistry. Physical objects with triangulated surfaces can be embedded in the grid, creating cut and split grid cells. The grid is also used to efficiently find particle/surface collisions.

For more information, visit the **SPARTA official website** and **SPARTA page at GitHub**.

Programming language: C++

Brief description: A parallel DSMC or Direct Simulation Monte Carlo code for performing simulations of low-density gases in 2D or 3D.

Open source license: GPL 2.0

Recommended Software Version

SPARTA 7Jan2022

2.16.2 Environment Requirements

Hardware Requirements

Table 2-76 lists the hardware requirements.

Table 2-76 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 2-77 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz

OS Requirements

Table lists the OS requirements.

Table 2-78 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html

2.16.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SPARTA software porting.

Table 2-79 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.
2	/path/to/sparta	Installation path of SPARTA	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

2.16.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 2-80 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.

2.16.5 Obtaining the Source Code

Procedure

Step 1 Download the SPARTA source package **sparta-7Jan2022.tar.gz**.

https://github.com/sparta/sparta/archive/refs/tags/7Jan2022.tar.gz

Step 2 Use an SFTP tool to upload the SPARTA source package to the */path/to/sparta* directory on the server.

----End

2.16.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/sparta
```

Step 3 Decompress the source package.

```
tar -xvf sparta-7Jan2022.tar.gz
```

Step 4 Create a **build** directory and an **install** directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd sparta-7Jan2022/
mkdir build install
```

Step 5 Perform the compilation and installation.

```
cmake -DCMAKE_INSTALL_PREFIX=../install /path/to/sparta/sparta-7Jan2022/cmake
make
make install
```

----End

2.16.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test case directory.

```
cd /path/to/sparta/sparta-7Jan2022/examples/spiky
```

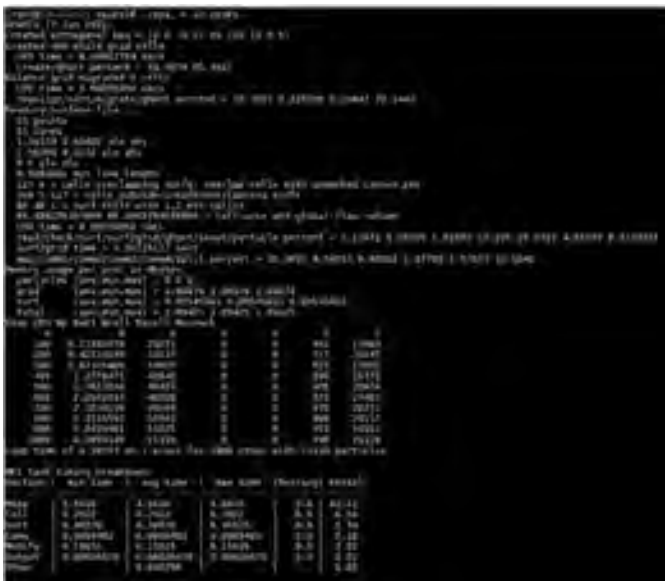
Step 3 Copy the generated SPARTA executable file to the current directory.

```
cp /path/to/sparta/sparta-7Jan2022/install/bin/spa_ ./
```

Step 4 Run the test case.

```
./spa_ < in.spiky
```

The running is successful if the following information is displayed:



----End

3 Education and Scientific Research

3.1 UDUNITS 2.2.28 Porting Guide (CentOS 7.6)
3.2 GROMACS 2019.3 Porting Guide (CentOS 7.6)
3.3 LAMMPS 5 Jun 2019 Porting Guide (CentOS 7.6)
3.4 Quantum ESPRESSO 6.4.1 Porting Guide (CentOS 7.6)
3.5 NAMD 2.13 Porting Guide (CentOS 7.6)
3.6 VASP 5.4.4 Porting Guide (CentOS 7.6)
3.7 ABINIT 8.10.3 Porting Guide (CentOS 7.6)
3.8 AmberTools 19 Porting Guide (CentOS 7.6)
3.9 CP2K 7.1 Porting Guide (CentOS 7.6)
3.10 NWChem 6.8.1 Porting Guide (CentOS 7.6)
3.11 ROOT 6.20 Porting Guide (CentOS 7.6)
3.12 Geant4 10.6 Porting Guide (CentOS 7.6)
3.13 DL_POLY 1.10 Porting Guide (CentOS 7.6)
3.14 GAMESS Porting Guide (CentOS 7.6)
3.15 ESPResSo 4.1.4 Porting Guide (CentOS 7.6)
3.16 MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)
3.17 MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6)
3.18 Octopus 10.3 Porting Guide (CentOS 7.6)
3.19 Psi4 1.3.2 Porting Guide (CentOS 7.6)
3.20 SIESTA 4.0.2 Porting Guide (CentOS 7.6)
3.21 Wannier90 3.1.0 Porting Guide (CentOS 7.6)
3.22 WanT 2.6.1 Porting Guide (CentOS 7.6)
3.23 GROMACS 2019.3 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

3.24 NAMD 2.14 Porting Guide (CentOS 8.2)
3.25 CP2K 7.1 Porting Guide (openEuler 21.03)
3.26 DL_POLY 1.10 Porting Guide (openEuler 21.03)
3.27 GROMACS 2019.3 Porting Guide (openEuler 21.03)
3.28 LAMMPS 5 Jun 2019 Porting Guide (openEuler 21.03)
3.29 NWChem 6.8.1 Porting Guide (openEuler 21.03)
3.30 VASP 5.4.4 Porting Guide (openEuler 21.03)
3.31 Wannier90 3.1.0 Porting Guide (openEuler 21.03)
3.32 QE 6.4.1 Porting Guide (openEuler 21.03)
3.33 MOOSE 1.0.0 Porting Guide (openEuler 21.03)
3.34 SIESTA 4.0.2 Porting Guide (openEuler 21.03)
3.35 UDUNITS 2.2.28 Porting Guide (openEuler 21.03)
3.36 ROOT 6.20 Porting Guide (openEuler 21.03)
3.37 LAMMPS 10 Feb 2021 Porting Guide (CentOS 8.2)
3.38 QE 6.8 Porting Guide (CentOS 8.2)
3.39 ABINIT 8.10.3 Porting Guide (openEuler 21.03)
3.40 AmberTools 21 Porting Guide (openEuler 21.03)
3.41 NAMD 2.13 Porting Guide (openEuler 21.03)
3.42 miniGhost 1.0 Porting Guide (openEuler 21.03)
3.43 MiniFE 2.2.0 Porting Guide (CentOS 7.6)
3.44 miniGhost v1.0 Porting Guide (CentOS 7.6)
3.45 ELK 8.4.30 Porting Guide (Kylin V10)
3.46 MOPAC 22.0.4 Porting Guide (Kylin V10)
3.47 QMCPACK 3.14.0 Porting Guide (Kylin V10)
3.48 OpenCV 4.5.5 Porting Guide (openEuler 20.03)
3.49 Phono3py 2.3.2 Porting Guide (Kylin V10)
3.50 QDK 2.3.12 Porting Guide (Kylin V10)
3.51 Towhee 8.2.3 Porting Guide (Kylin V10)
3.52 CosmoMC June2021 Porting Guide (Kylin V10)
3.53 ParaFEM 5.0.3 Porting Guide (Kylin V10)
3.54 VASPKIT 0.52 Porting Guide (Kylin V10)
3.55 MNE-Python Porting Guide (openEuler 20.03)
3.56 MiniDFT 1.1.1 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

3.57 dugksFoam 1.0 Porting Guide (Kylin V10)
3.58 DUNS 4.7.1 Porting Guide (Kylin V10)
3.59 CalculiX 2.19.0 Porting Guide (Kylin V10)
3.60 BerkeleyGW 3.0.1 Porting Guide (Kylin V10)
3.61 Amber 20 Porting Guide (CentOS 8.2)
3.62 Galaxy 22.01 Porting Guide (Kylin V10)
3.63 NWChem 6.8.1 Porting Guide (Kylin V10)
3.64 T-Flows 0.9 Porting Guide (Kylin V10)
3.65 yambo 5.1.1 Porting Guide (openEuler 20.03)

3.1 UDUNITS 2.2.28 Porting Guide (CentOS 7.6)

3.1.1 Introduction

The UDUNITS-2 package provides support for units of physical quantities. Its three main components are: 1) a C library for units of physical quantities (udunits2lib); 2) a utility; for obtaining the definition of a unit and for converting numeric values between compatible units (udunits2prog), and 3) an extensive database of units.

For more information about UDUNITS-2, visit the official UDUNITS-2 website.

Programming language: C/Fortran

Brief description: database.

Open source license: GPLv3

Recommended Software Version

UDUNITS 2.2.28

3.1.2 Environment Requirements

Hardware Requirements

Table 3-1 lists the hardware requirements.

Table 3-1 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-2 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-2 Software requirements

Item	Version	Download Address
GNU	9.3.0	https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/
HPC-X	2.6	https://www.mellanox.com/page/hpcx_eula?mtag=hpc-x&mrequest=downloads&mtype=hpc&mver=hpc-x&mname=v2.6/hpcx-v2.6.0-gcc-OFED-4.17-redhat7.6-aarch64
UDUNITS	2.2.28	https://gfd-dennou.org/arch/ucar/unidata/pub/udunits/udunits-2.2.28.tar.gz

OS Requirements

Table 3-3 lists the OS requirements.

Table 3-3 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.1.3 Paths for Software Porting

This chapter lists the software installation paths involved in the UDUNITS software porting.

Table 3-4 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/UDUNITS	Installation path of UDUNITS.	The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

3.1.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

Configuration Process

Table 3-5 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.1.5 Obtaining the Source Code

Procedure

Step 1 Obtain UDUNITS installation package **udunits-2.2.28.tar.gz** from the following address:

https://gfd-dennou.org/arch/ucar/unidata/pub/udunits/udunits-2.2.28.tar.gz

Step 2 Use an SFTP tool to upload the UDUNITS installation package to the */path/to/UDUNITS* directory on the server.

----End

3.1.6 Compiling and Installing UDUNITS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/UDUNITS
```

Step 3 Run the following command to decompress the package:

```
tar -zxvf udunits-2.2.28.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd udunits-2.2.28
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/UDUNITS CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 6 Run the following commands to perform compilation:

```
make -j 16
make install
```

----End

3.1.7 Running and Verifying UDUNITS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Check the installation path. If the following file is generated, the installation is successful:

```
./bin:
udunits2

./include:
converter.h  udunits2.h  udunits.h

./lib:
libudunits2.a  libudunits2.la  libudunits2.so  libudunits2.so.0  libudunits2.so.0.1.0

./share:
doc  info  udunits
```

----End

3.2 GROMACS 2019.3 Porting Guide (CentOS 7.6)

3.2.1 Introduction

Molecular dynamics (MD) simulation is the most widely used method for calculating a large complex system. Due to rapid development of molecular simulation since 1970, many force fields applicable to biochemical molecular systems, polymers, metals, and non-metallic materials had been systematically established. This greatly improves the capability and accuracy of computing complex system structures and some thermodynamic and spectral properties. MD simulation is a computational method developed by applying these force fields and according to Newton's mechanics of motion.

GROningen MAchine for Chemical Simulations (GROMACS) is a computational engine for MD simulation and energy minimization, which simulates hundreds to millions of atomic systems using Newton's equilibrium equations. It is primarily

 Copyright © Huawei Technologies Co., Ltd.

designed for biochemical molecules like proteins, lipids and nucleic acids that have a large number of complicated bonded interactions, but since it is extremely fast at calculating the nonbonded interactions (that usually dominate simulations), many groups are also using it for research on non-biological systems, such as polymers. Compared with other MD simulation software, GROMACS has the following unique advantages:

1. GROMACS is free of charge and complies with the GNU Lesser General Public License (LGPL). You can find the open-source code of GROMACS on GitHub.
2. GROMACS provides higher performance than the other software of similar types and has been optimized in code.
3. GROMACS is user-friendly for topology files and parameter setting files. The format of GROMACS is similar to that of Python.
4. The environment of GROMACS develops well, and many simulation tools support GROMACS well.

For more information about GROMACS, visit the **GROMACS official website**.

Programming language: C++

Brief description: MD simulation and energy minimization computing engine

Open source license: LGPL Version 2.1

NOTE

For GROMACS 2019.3 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port GROMACS to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

GROMACS 2019.3

3.2.2 Environment Requirements

Hardware Requirements

Table 3-6 lists the hardware requirements.

Table 3-6 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-7 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-7 Software requirements

Item	Version	Download Address
GROMACS	2019.3	http://manual.gromacs.org/documentation/2019.3/download.html
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/releases
FFTW	3.3.8	http://www.fftw.org/download.html
CMake	3.8.1	https://cmake.org/download/#previous
Test case	water_GMX50_bare	https://ftp.gromacs.org/pub/benchmarks/

OS Requirements

Table 3-8 lists the OS requirements.

Table 3-8 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.2.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GROMACS software porting.

Table 3-9 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are
3	/path/to/FFTW	Installation path of FFTW	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/CMAKE	Installation path of CMake	examples only. Use the actual paths planned during the installation process.
5	/path/to/GROMACS	Installation path of GROMACS	
6	/path/to/CASE	Planned path for storing the GROMACS test cases	

3.2.4 Configuring the Compilation Environment

3.2.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-10 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.2.4.2 Installing OpenBLAS**.
3	Installing FFTW	See **3.2.4.3 Installing FFTW**.
4	Installing CMake	See **3.2.4.4 Installing CMake**.

3.2.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to decompress the OpenBLAS installation package:

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.6
```

Step 4 Run the following commands to set the environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to compile and install OpenBLAS:

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following commands to set the OpenBLAS environment variables:

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.2.4.3 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd fftw-3.3.8
```

Step 4 Run the following command to run the **bootstrap.sh** script:

```
./bootstrap.sh
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon --enable-float
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j40
make install
```

Step 7 Run the following commands to set FFTW environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.2.4.4 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the CMake installation package:

NOTE

The CMake version must be 3.4.3 or later for GROMACS 2019.3.

```
tar zxvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd cmake-3.8.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make install
```

Step 6 Run the following command to set the CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.2.5 Obtaining the Source Code

Procedure

Step 1 Download the GROMACS installation package **gromacs-2019.3.tar.gz**.

http://manual.gromacs.org/documentation/2019.3/download.html

Step 2 Use an SFTP tool to upload the GROMACS installation package to the */path/to/GROMACS* directory on the server.

----End

3.2.6 Compiling and Installing GROMACS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the GROMACS installation package:

```
tar -xvf gromacs-2019.3.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd Gromacs-2019.3
```

Step 4 Run the following commands to create a **build** directory and switch to the directory:

```
mkdir build
cd build
```

Step 5 Run the following command to perform configuration:

```
FLAGS="-mcpu=tsv110"; CFLAGS=$FLAGS CXXFLAGS=$FLAGS LDFLAGS="-lgfortran" CC=mpicc
CXX=mpicxx \
/path/to/CMAKE/bin/cmake \
-DCMAKE_INSTALL_PREFIX=/path/to/GROMACS/gromacs-2019.3 \
-DBUILD_SHARED_LIBS=on \
```

 Copyright © Huawei Technologies Co., Ltd.

```
-DBUILD_TESTING=on \
-DREGRESSIONTEST_DOWNLOAD=off \
-DGMX_BUILD_OWN_FFTW=off \
-DGMX_SIMD=ARM_NEON_ASIMD \
-DGMX_DOUBLE=off \
-DGMX_EXTERNAL_BLAS=on \
-DGMX_EXTERNAL_LAPACK=on \
-DGMX_FFT_LIBRARY=fftw3 \
-DGMX_BLAS_USER=/path/to/OPENBLAS/OpenBLAS-0.3.6/lib/libopenblas.a \
-DGMX_LAPACK_USER=/path/to/OPENBLAS/OpenBLAS-0.3.6/lib/libopenblas.a \
-DFFTWF_LIBRARY=/path/to/FFTW/libfftw3f.so \
-DFFTWF_INCLUDE_DIR=/path/to/FFTW/include \
-DGMX_GPU=off \
-DGMX_MPI=on \
-DGMX_OPENMP=on \
-DGMX_X11=off \
../
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j40 V=1
make -j40 install
```

Step 7 Run the following command to check whether an executable file is generated:

```
ll /path/to/GROMACS/bin/gmx_mpi
-rwxr-xr-x 1 root root 83656 Jul 31 02:51 gmx_mpi
```

Step 8 Run the following commands to set environment variables:

```
export PATH=/path/to/GROMACS/bin:$PATH
```

----End

3.2.7 Running and Verifying GROMACS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the case is stored:

```
cd /path/to/CASE
```

Step 3 Decompress the compute instance files:

```
tar xvf water_GMX50_bare.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd water-cut1.0_GMX50_bare/0768
```

Step 5 Run the following command to create a **topol.tpr** file:

```
gmx_mpi grompp -f pme.mdp
```

Step 6 Run the following command to view whether the **topol.tpr** file is created:

```
ll topol.tpr
-rw-r--r-- 1 root root 18448672 Jan 11 16:41 topol.tpr
```

Step 7 Run the following command to start the GROMACS test:

```
mpirun --allow-run-as-root --mca btl ^openib  -np 96 gmx_mpi mdrun -dlb yes -v -nsteps 10000 -resethway -noconfout -pin on -ntomp 1 -s topol.tpr
```

View the value of **ns/day** in **Performance** in the **md.log** file. The unit is **ns/day**. The higher the value is, the better the performance is.

The following is an example of the test result:

 Copyright © Huawei Technologies Co., Ltd.

```
Part of the total run time spent waiting due to load imbalance: 1.1%.
Steps where the load balancing was limited by -rdd, -rcon and/or -dds: X 0 % Y 0 %
Average PME mesh/force load: 1.033
Part of the total run time spent waiting due to PP/PME imbalance: 2.1 %
               Core t (s)   Wall t (s)        (%)
       Time:    14806.100      154.231     9600.0
                 (ns/day)    (hour/ns)
Performance:        5.603        4.283
GROMACS reminds you: "Come on boys, Let's push it hard" (P.J. Harvey)
```

----End

3.2.8 More Resources

Arm website:

https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/resources/porting-and-tuning/building-gromacs-with-arm-compiler-deprecated

Zenodo website:

https://zenodo.org/record/3243834#.XUJg-vIzZhE

3.3 LAMMPS 5 Jun 2019 Porting Guide (CentOS 7.6)

3.3.1 Introduction

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a molecular dynamics program from Sandia National Laboratories. LAMMPS makes use of Message Passing Interface (MPI) for parallel communication and is free and open-source software, distributed under the terms of the GNU General Public License. The latest LAMMPS version supports CUDA and OpenCL-based GPU computing. LAMMPS was originally developed under a Cooperative Research and Development Agreement (CRADA) between the United States Department of Energy and three laboratories from private sector firms.

For more information about LAMMPS, visit the **LAMMPS official website**.

Programming language: C++

Brief description: open-source molecular dynamics program

Open source license: GPL 2.0

NOTE

For LAMMPS 5 Jun 2019 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port LAMMPS to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

LAMMPS June 5, 2019

 Copyright © Huawei Technologies Co., Ltd.

3.3.2 Environment Requirements

Hardware Requirements

Table 3-11 lists the hardware requirements.

Table 3-11 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-12 lists the software requirements.

Table 3-12 Software requirements

Item	Version	Download Address
LAMMPS	June 5, 2019	**https://lammps.sandia.gov/tars/**
FFTW	3.3.8	**http://www.fftw.org/fftw-3.3.8.tar.gz**
Test case	in.lj	Included in the **bench** directory of the LAMMPS source package

OS Requirements

Table 3-13 lists the OS requirements.

Table 3-13 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

3.3.3 Paths for Software Porting

This chapter lists software installation paths involved in the LAMMPS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-14 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	N/A	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FFTW	Installation path of FFTW	The installation path listed in this table is only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
3	/path/to/LAMMPS	Installation path of LAMMPS	
4	/path/to/CASE	Path of LAMMPS test cases.	

3.3.4 Configuring the Compilation Environment

3.3.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-15 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing FFTW	See **3.3.4.2 Installing FFTW**.

3.3.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to compile and install FFTW.

```
make -j 96
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.3.5 Obtaining the Source Code

Procedure

Step 1 Download the LAMMPS installation package **lammps-5Jun19.tar.gz** from:

https://lammps.sandia.gov/tars/

Step 2 Use an SFTP tool to upload the LAMMPS installation package to the */path/to/LAMMPS* directory on the server.

----End

3.3.6 Compiling and Installing LAMMPS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory, in which the LAMMPS installation package is stored:

```
cd /path/to/LAMMPS
```

Step 3 Run the following command to decompress the LAMMPS installation package:

```
tar -xvf lammps-5Jun19.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd lammps-5Jun2019
```

Step 5 Run the following command to switch to the **src** directory:

```
cd src
```

Step 6 Perform the following operations to modify the **MAKE/OPTIONS/Makefile.g++_openmpi** file:

1. Open the **MAKE/OPTIONS/Makefile.g++_openmpi** file.
   ```
   vi MAKE/OPTIONS/Makefile.g++_openmpi
   ```

2. Press **i** to enter the insert mode and modify the content in lines 54, 55, and 56.
```
FFT_INC =    -DFFT_FFTW -I/path/to/FFTW/include
FFT_PATH =   -L/path/to/FFTW/lib
FFT_LIB =    -lfftw3
```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following commands to perform compilation:
```
make yes-std
make no-lib
make -j 96 g++_openmpi
```

----End

3.3.7 Running and Verifying LAMMPS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create the working directory:
```
mkdir -p path/to/CASE
```

Step 3 Perform the following commands to switch to the working directory and copy the computing instance and binary files to the working directory.
```
cd /path/to/CASE
cp /path/to/LAMMPS/lammps-stable_5Jun2019/bench/in.lj  ./
cp /path/to/LAMMPS/lammps-stable_5Jun2019/src/lmp_g++_openmpi ./
```

Step 4 Run the following command:
```
mpirun --allow-run-as-root -np 96  --mca btl ^openib  ./lmp_g++_openmpi -in in.lj
```

Check the value (in the unit of **timesteps/s**) of **Performance** in the **log.lammps** log file. A higher value indicates better performance.

The following is an example of the test result.
```
Performance: 593019.799 tau/day, 1372.731 timesteps/s
98.1% CPU use with 96 MPI tasks x no OpenMP threads
MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.028215   | 0.040367   | 0.051175   |   3.4 | 55.41
Neigh   | 0.0036562  | 0.0049894  | 0.0061901  |   1.2 |  6.85
Comm    | 0.013968   | 0.026072   | 0.039602   |   4.8 | 35.79
Output  | 7.905e-05  | 0.000128   | 0.00023195 |   0.0 |  0.18
Modify  | 0.00056847 | 0.00080538 | 0.00099514 |   0.0 |  1.11
Other   |            | 0.0004857  |            |       |  0.67
```

----End

3.4 Quantum ESPRESSO 6.4.1 Porting Guide (CentOS 7.6)

3.4.1 Introduction

Quantum ESPRESSO is a software suite for ab initio quantum chemistry methods for electronic-structure calculation and materials modeling, distributed for free

 Copyright © Huawei Technologies Co., Ltd.

and as free software free under the GNU General Public License. It is based on density functional theory (DFT), plane wave (PW) basis sets and pseudopotentials (including norm conservation and ultrasoft). Quantum ESPRESSO is an open initiative of the CNR-IOM DEMOCRITOS National Simulation Center in Trieste, Italy, and its partners, in collaboration with different centers worldwide such as MIT, Princeton University, the University of Minnesota or the Ecole Polytechnique Fédérale de Lausanne. The initiative is coordinated by the Quantum ESPRESSO Foundation, which is composed of a host of research centers and groups worldwide. Quantum ESPRESSO, written mainly in Fortran-90 with some parts in C or in Fortran-77, was built out of the merging and re-engineering of different independently-developed core packages, plus a set of packages designed to be inter-operable with the core components. The basic packages include Pwscf, which solves the self-consistent Kohn and Sham equations and obtains for a periodic solid, CP, which carries out Car-Parrinello molecular dynamics, and PostProc, which allows data analysis and plotting. Regarding the additional packages, it is noteworthy to point out atomic for the pseudopotential generation, PHonon package, which implements density-functional perturbation theory (DFPT) for the calculation of second- and third-order derivatives of the energy with respect to atomic displacements and NEB.

For more information about Quantum ESPRESSO, visit the **Quantum ESPRESSO official website**.

Programming language: Fortran

Brief description: A software suite of ab initio quantum chemistry methods for electronic structure calculation and materials modeling.

Open source license: GPL 2.0

NOTE

For Quantum ESPRESSO 6.4.1 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port Quantum ESPRESSO to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

QE 6.4.1

3.4.2 Environment Requirements

Hardware Requirements

Table 3-16 lists the hardware requirements.

Table 3-16 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-17 lists the software requirements.

Table 3-17 Software requirements

Item	Version	Download Address
QE	6.4.1	https://github.com/QEF/q-e/tags
OpenBLAS	0.3.7	https://github.com/xianyi/OpenBLAS/tree/v0.3.7
ScaLAPACK	2.0.2	http://www.netlib.org/scalapack/
Test case	AUSURF112	https://github.com/QEF/benchmarks/tree/master/AUSURF112

OS Requirements

Table 3-18 lists the OS requirements.

Table 3-18 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Quantum ESPRESSO software porting.

Table 3-19 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/SCALAPACK	Installation path of ScaLapack	examples only. Use the actual paths planned during the installation process.
4	/path/to/QE	Installation path of Quantum ESPRESSO	
5	/path/to/CASE	Path for storing the Quantum ESPRESSO test cases	

3.4.4 Configuring the Compilation Environment

3.4.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-20 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.4.4.2 Installing OpenBLAS**.
3	Installing ScaLAPACK	See **3.4.4.3 Installing ScaLAPACK**.

3.4.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the OpenBLAS installation package:

```
tar xvf OpenBLAS-0.3.7.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd OpenBLAS-0.3.7
```

Step 4 Run the following commands to declare the compilation environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following command to load environment variables:

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.4.4.3 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the ScaLapack installation package:

```
tar -xvf scalapack.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd scalapack-2.0.2
```

Step 4 Run the following command to generate the **SLmake.inc** file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following command to edit the **SLmake.inc** file:

1. Open **SLmake.inc**.
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the insert mode and modify lines 58 and 59 of the file.
   ```
   BLASLIB      = -L/path/to/OPENBLAS/lib -lopenblas
   LAPACKLIB     = -L/path/to/OPENBLAS/lib -lopenblas
   ```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to perform compilation and installation:

```
make
```

Step 7 Run the following command to copy the generated link library to the installation directory:

```
cp libscalapack.a /path/to/SCALAPACK
```

Step 8 Run the following command to load environment variables:

```
export LD_LIBRARY_PATH=/path/to/SCALAPACK:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.4.5 Obtaining the Source Code

Procedure

Step 1 Download the installation package **q-e-qe-6.4.1.tar.gz**.

https://github.com/QEF/q-e/tags

Step 2 Use an SFTP tool to upload the QE installation package to the */path/to/QE* directory on the server.

----End

3.4.6 Compiling and Installing Quantum ESPRESSO

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the Quantum ESPRESSO installation package:

```
cd /path/to/QE
tar -xvf q-e-qe-6.4.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd q-e-qe-6.4.1
```

Step 4 Run the following commands to set the BLAS, LAPACK, and ScaLAPACK libraries:

```
export BLAS_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export LAPACK_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export SCALAPACK_LIBS="-L/path/to/SCALAPACK -lscalapack"
```

Step 5 Run the following commands to perform configuration:

```
./configure F90=gfortran F77=gfortran  MPIF90=mpifort MPIF77=mpifort  CC=mpicc \
FCFLAGS="-O3" CFLAGS="-O3" \
--with-scalapack=yes \
--prefix=/path/to/QE
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 20 pwall
make install
```

Step 7 Run the following command to load environment variables:

```
export PATH=/path/to/QE/bin:$PATH
```

----End

3.4.7 Running and Verifying Quantum ESPRESSO

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create and go to the working directory:

```
mkdir -p /path/to/CASE
cd /path/to/CASE
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Download the test cases and use an SFTP tool to upload the test cases to the */path/to/CASE* directory on the server.

https://github.com/QEF/benchmarks/tree/master

Step 4 Run the following commands to decompress the test case package and copy the required files to the */path/to/CASE* directory:

```
unzip benchmarks-master.zip
cp benchmarks-master/AUSURF112/* ./
```

Step 5 Run the following command to create a **hostfile** file:

1. Open **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   node1
   node2
   ```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 6 Open another login window and use PuTTY to log in to node 2 as the **root** user.

Step 7 On node 2, run the following command to configure environment variables:

1. Open **.bashrc**.
   ```
   vi /root/.bashrc
   ```
2. Press **i** to enter the insert mode, and add the following content to the end of the **.bashrc** file:
   ```
   export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
   export PATH=/path/to/QE/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 8 On node 1, run the following command to start the Quantum ESPRESSO program:

```
mpirun --allow-run-as-root -np 192 -N 96 -x OMP_NUM_THREADS=1 -hostfile hostfile --mca btl ^openib pw.x -input ./ausurf.in 2>&1 | tee -a qe.log
```

Table 3-21 Parameters

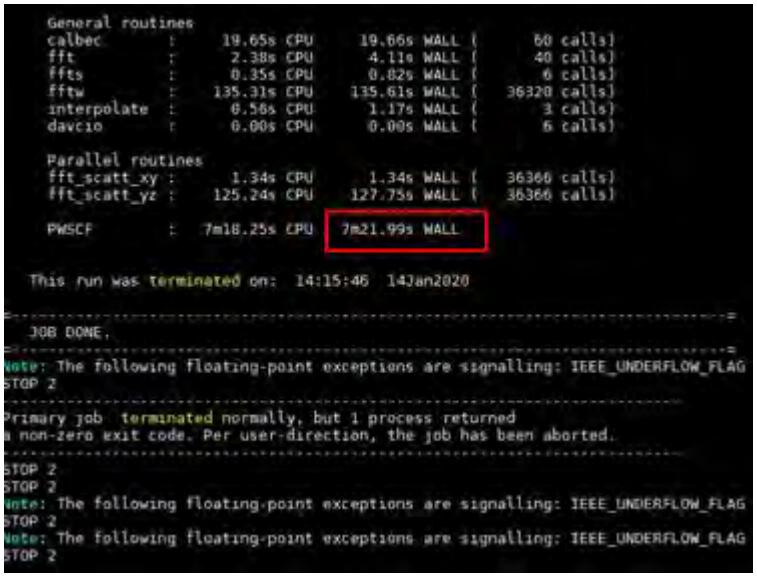
Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
-x OMP_NUM_THREADS=1	Number of threads in each process. Set the parameter to **1**.
--hostfile	Name of the node to be used.

If the information shown in **Figure 3-1** is displayed at the end of the **qe.log** file generated by Quantum ESPRESSO, the Quantum ESPRESSO program ends properly.

Check the value of **WALL** in the **qe.log** file. The unit is second. A higher the value indicates a lower performance. **Figure 3-1** shows the output result.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-1 Test example



----End

3.5 NAMD 2.13 Porting Guide (CentOS 7.6)

3.5.1 Introduction

NAnoscale Molecular Dynamics (NAMD) is a parallel molecular dynamics application that has been used to make breakthroughs in understanding the structure and dynamics of large biomolecular complexes NAMD uses empirical force fields, such as Amber, CHARMM, and Dreiding, to calculate atomic trajectories by numerically solving motion equations. NAMD is used to predict the dynamic behavior and important properties of biomolecules, such as dispersive factor and cohesive energy.

For more information about NAMD, visit the **official NAMD website**.

Programming language: C

Brief description: The parallel molecular dynamics code for fast simulation of macromolecular systems.

Open source license: custom open source license

Recommended Software Version

NAMD 2.13

3.5.2 Environment Requirements

Hardware Requirements

Table 3-22 lists the hardware requirements.

Table 3-22 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-23 lists the software requirements.

Table 3-23 Software requirements

Item	Version	Download Address
NAMD	2.13	https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD
CHARM	6.9.0	https://charm.cs.illinois.edu/distrib/charm-6.9.0.tar.gz
TCL	8.5.9	http://www.ks.uiuc.edu/Research/namd/libraries/tcl8.5.9-linux-arm64-threaded.tar.gz
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
Test case	Apoa1	http://www.ks.uiuc.edu/Research/namd/utilities/apoa1.tar.gz

OS Requirements

Table 3-24 lists the OS requirements.

Table 3-24 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.5.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NAMD software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-25 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CHARM	Installation path of CHARM	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/FFTW	Installation path of FFTW	
4	/path/to/TCL	Installation path of TCL	
5	/path/to/NAMD	Installation path of NAMD	
6	/path/to/CASE	Path for storing the NAMD test case	

3.5.4 Configuring the Compilation Environment

3.5.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-26 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CHARM	See **Installing CHARM**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing FFTW	See **3.5.4.2 Installing FFTW**.
4	Installing TCL	See **Installing TCL**.

3.5.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd fftw-3.3.8
```

Step 4 Run the following command to run the **bootstrap.sh** script:

```
./bootstrap.sh
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/FFTW --enable-threads --enable-openmp --enable-float
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j40
make -j40 install
```

Step 7 Run the following commands to set the FFTW environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.5.5 Obtaining the Source Code

Procedure

Step 1 Download the NAMD installation package **NAMD_2.13_Source.tar.gz**.

https://www.ks.uiuc.edu/Research/namd/

Step 2 Use an SFTP tool to upload the NAMD installation package to the */path/to/NAMD* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.5.6 Compiling and Installing NAMD

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create the NAMD-related directories:

```
cd /path/to/NAMD
mkdir NAMD-2.13
cd NAMD-2.13
mkdir build install
export BUILD_DIR=`pwd`/build
export INSTALL_DIR=`pwd`/install
```

Step 3 Run the following commands to set the environment variables.

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 4 Run the following commands to compile and install CHARM:

```
cd /path/to/NAMD/NAMD-2.13/build
tar -xvf charm-6.9.0.tar.gz
cd charm-6.9.0
mkdir ${INSTALL_DIR}/charm690/
cp src/arch/mpi-linux-x86_64 src/arch/mpi-linux-arm8 -rf
grep -rl 'm64' src/arch/mpi-linux-arm8 | xargs sed -i 's/\-m64//g'
./build charm++ mpi-linux-arm8 smp --with-production --destination=/path/to/NAMD/NAMD-2.13/
install/charm690/mpi-linux-arm8
```

Step 5 Run the following commands to decompress the NAMD installation package:

```
cd /path/to/NAMD/NAMD-2.13/build
tar -zxvf NAMD_2.13_Source.tar.gz
cd NAMD_2.13_Source
```

Step 6 Run the following command to create **Linux-ARM64.fftw3**:

```
cp arch/Linux-x86_64.fftw3 arch/Linux-ARM64.fftw3
```

Step 7 Run the following commands to upload and decompress the TCL file:

```
cd /path/to/NAMD/NAMD-2.13
tar -xvf tcl8.5.9-linux-arm64-threaded.tar.gz
```

Step 8 Run the following commands to modify the **Make.charm** file:

1. Open **Make.charm**.
   ```
   cd /path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source
   vi Make.charm
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   CHARMBASE = **/path/to/NAMD/NAMD-2.13/install/charm690**

   NOTE

   Change the value of **CHARMBASE** to the actual installation path of CHARM.
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following commands to modify the **Linux-ARM64-g++.arch** file:

1. Open **Linux-ARM64-g++.arch**.
   ```
   cd /path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source
   vi arch/Linux-ARM64-g++.arch
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   NAMD_ARCH = Linux-ARM64
   CHARMARCH = mpi-linux-arm8

 Copyright © Huawei Technologies Co., Ltd.

```
#FLOATOPTS = -O2 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer -march=armv7-a -
mcpu=cortex-a9 -mtune=cortex-a9
FLOATOPTS = -O3 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer -march=armv8-
a

CXX = g++ -std=c++11
CXXOPTS = $(FLOATOPTS)
CXXNOALIASOPTS = $(FLOATOPTS) -fno-strict-aliasing

CC = gcc
COPTS = $(FLOATOPTS)
```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 10 Run the following commands to configure NAMD:

```
./config Linux-ARM64-g++ --with-fftw3 --fftw-prefix /path/to/FFTW --with-tcl --tcl-prefix /path/to/
NAMD/NAMD-2.13/tcl8.5.9-linux-arm64-threaded --charm-arch mpi-linux-arm8
cd Linux-ARM64-g++
```

Step 11 Run the following commands to compile and install NAMD:

```
make -j
```

Step 12 Run the following command to check whether an executable file is generated:

```
ll /path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++/namd2
-rwxr-xr-x 1 root root 16140248 Aug  7 15:50 namd2
```

----End

3.5.7 Running and Verifying NAMD

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to set the environment variable for the NAMD executable file:

```
export PATH=/path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++/:$PATH
```

Step 3 Run the following command to go to the test case directory:

```
cd /path/to/CASE
```

Step 4 Run the following command to decompress the case file:

```
tar -xf apoa1.tar.gz
```

Step 5 Run the following command to modify the case file:

1. Open the test case file.
   ```
   vi apoa1/apoa1.namd
   ```
2. Press **i** to enter the insert mode and add the following content to the end of the file:
   ```
   FFTWEstimate yes
   FFTWUseWisdom no
   ```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to start a test:

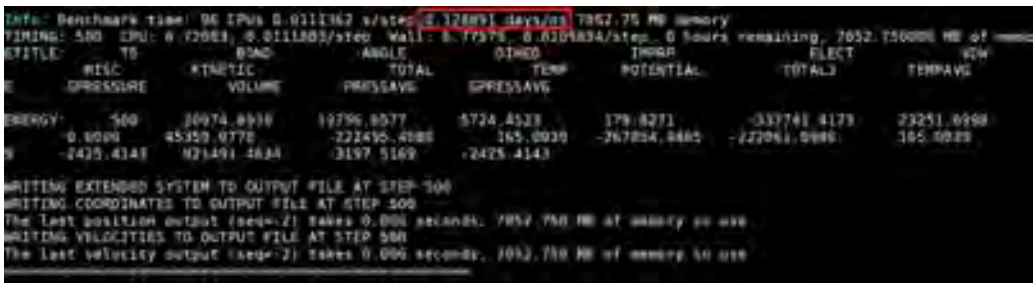
```
cd apoa1
mpirun --allow-run-as-root --mca btl ^openib -np 1 namd2 +ppn 96 apoa1.namd +setcpuaffinity 2>&1
|tee -a namd.log
```

 Copyright © Huawei Technologies Co., Ltd.

Check the value of **days/ns** in the last **Info:Benchmark time** in the **namd.log** file. A smaller value indicates better performance. You can also convert the value to the value of **ns/days**. A higher value indicates better performance.

Figure 3-2 shows an example of the test result.

Figure 3-2 Result example



----End

3.5.8 More Information

Official Arm website:

https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/resources/porting-and-tuning/building-namd-with-arm-compiler-deprecated

3.6 VASP 5.4.4 Porting Guide (CentOS 7.6)

3.6.1 Introduction

Vienna Ab-initio Simulation Package (VASP) is a software package developed by the Hafner team of the University of Vienna for electronic structure calculation and quantum mechanics-molecular dynamics simulation. VASP is one of the most popular commercial software used in material simulation and computational physical science study.

For more information about MySQL, visit the **official VASP website**.

Programming language: Fortran

Brief description: electronic structure calculation and quantum mechanics-molecular dynamics simulation.

Open source license: open source commercial software

Recommended Software Version

VASP 5.4.4

 Copyright © Huawei Technologies Co., Ltd.

3.6.2 Environment Requirements

Hardware Requirements

Table 3-27 lists the hardware requirements.

Table 3-27 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software requirements

Table 3-28 lists the software requirements.

Table 3-28 Software requirements

Item	Version	Download Address
VASP	5.4.4	**https://www.vasp.at/**
FFTW	3.3.8	**http://www.fftw.org/download.html**
OpenBLAS	0.3.6	**https://github.com/xianyi/OpenBLAS/tree/v0.3.6**
ScaLAPACK	2.0.2	**http://www.netlib.org/scalapack/**
VTST	179	**http://theory.cm.utexas.edu/vtsttools/download.html**
Wannier90	1.2	**https://github.com/wannier-developers/wannier90/archive/refs/tags/v1.2.tar.gz**
vasp-small	Test case	-

OS Requirements

Table 3-29 lists the OS requirements.

Table 3-29 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

3.6.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VASP software porting.

Table 3-30 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/SCALAPACK	Installation path of ScaLAPACK	
4	/path/to/FFTW	Installation path of FFTW	
5	/path/to/VASP	Installation path of VASP	
6	/path/to/CASE	Test path of VASP	

3.6.4 Configuring the Compilation Environment

3.6.4.1 Configuration Description

Prerequisites

Use SFTP to upload the installation packages to the planned directories on the server.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-31 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.6.4.2 Installing OpenBLAS**.
3	Installing ScaLAPACK	See **3.6.4.3 Installing ScaLAPACK**.
4	Installing FFTW	See **3.6.4.4 Installing FFTW**.

3.6.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the OpenBLAS installation package:

```
unzip  OpenBLAS-develop.zip
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd OpenBLAS-develop
```

Step 4 Run the following command to perform configuration:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

3.6.4.3 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar -xvf  scalapack.tgz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

 Copyright © Huawei Technologies Co., Ltd.

```
cd scalapack-2.0.2
```

Step 4 Run the following command to generate the **SLmake.inc** file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following command to edit the **SLmake.inc** file:

1. Run the following command to edit the case file:
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the insert mode and modify lines 58 and 59 of the file.
   ```
   BLASLIB      = -L/path/to/OPENBLAS/lib -lopenblas
   LAPACKLIB    = -L/path/to/OPENBLAS/lib -lopenblas
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform compilation and installation:

```
make
```

Step 7 Run the following command to copy the generated link library to the installation directory:

```
mkdir -p /path/to/SCALAPACK
cp libscalapack.a /path/to/SCALAPACK
```

----End

3.6.4.4 Installing FFTW

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
tar -xvf  fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make install
```

----End

3.6.5 Obtaining the Source Code

Procedure

Step 1 Download the VASP installation package **vasp.5.4.4.tar.gz**, VTSTcode package **vtstcode-179.tgz**, and Wannier90-1.2 installation package **wannier90-1.2**.

Download address: **https://www.vasp.at/**

Download address: **http://theory.cm.utexas.edu/vtsttools/download.html**

Download address: **https://github.com/wannier-developers/wannier90/archive/refs/tags/v1.2.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the VASP installation package to the */path/to/VASP* directory on the server.

----End

3.6.6 Compiling and Installing VASP

3.6.6.1 Basic Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the VASP installation package:

```
cd /path/to/VASP
tar -xvf vasp.5.4.4.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd  vasp.5.4.4
```

Step 4 Run the following commands to set environment variables:

```
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:/path/to/OPENBLAS/lib:/path/to/SCALAPACK: /path/to/FFTW/lib:$LD_LIBRARY_PATH
```

Step 5 Run the following command to create and edit the **makefile.include** file:

1. Create a **makefile.include** file.
   ```
   vi makefile.include
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   ##Precompiler options
    CPP_OPTIONS= -DHOST=\"LinuxGNU\" \
                 -DMPI -DMPI_BLOCK=8000 \
                 -Duse_collective \
                 -DscaLAPACK \
                 -DCACHE_SIZE=5000 \
                 -Davoidalloc \
                 -Duse_bse_te \
                 -Dtbdyn \
                 -Duse_shmem
    CPP        = gcc -E -P -C -w $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)
    FC         = mpif90
    FCL        = mpif90
    FREE       = -ffree-form -ffree-line-length-none
    FFLAGS     = -w
    OFLAG      = -O3 -ffp-contract=fast -fdec-math -ffpe-trap=invalid,zero,overflow -ffpe-summary=none
    OFLAG_IN   = $(OFLAG)
    DEBUG      = -O0
    LLIBS      = -L/path/to/SCALAPACK -lscalapack -L/path/to/OPENBLAS/lib -lopenblas
    FFTW       ?= /path/to/FFTW
    LLIBS      += -L$(FFTW)/lib -lfftw3
    INCS       = -I$(FFTW)/include
    OBJECTS    = fftmpiw.o fftmpi_map.o  fftw3d.o  fft3dlib.o
    OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
    OBJECTS_O2 += fft3dlib.o
    # For what used to be vasp.5.lib
    CPP_LIB    = $(CPP)
    FC_LIB     = $(FC)
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
CC_LIB     = mpicc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
FREE_LIB   = $(FREE)
OBJECTS_LIB= linpack_double.o getshmem.o
# For the parser library
CXX_PARS   = g++
LIBS       += parser
LLIBS      += -Lparser -lparser -lstdc++
# Normally no need to change this
SRCDIR     = ../../src
BINDIR     = ../../bin
```

NOTE

Ensure that no space exists at the end of a line or after a backslash (\).

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform compilation and installation:

```
make std
```

If you need to generate **vasp_gam** and **vasp_ncl**, run the following command:

```
make all
```

----End

3.6.6.2 (Optional) Extended Installation

3.6.6.2.1 Installing the VTST Transition State Tools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Upload the VTSTcode installation package to the directory where the VASP is installed.

Step 3 Run the following commands to decompress the VTST installation package:

```
cd /path/to/VASP
tar -xzvf vtstcode-179.tgz
```

Step 4 Run the following command to back up the **chain.F** file in the **src** directory of the VASP installation package:

```
cp vasp-5.4.4/src/chain.F / vasp-5.4.4/src/chain.F_bak
```

Step 5 Run the following command to copy files in the **vtstcode-179** directory to the **src** directory:

```
cp vtstconde-179/* vasp-5.4.4/src/
```

Step 6 Run the following commands to modify the **src/main.F** file:

1. Run the following command to edit the case file:
   ```
   vi vasp-5.4.4/src/main.F
   ```
2. Press **i** to enter the insert mode. Pay attention to the content in bold in line 3147.

   Before the modification:

   CALL CHAIN_FORCE(T_INFO%NIONS,DYN%POSION,TOTEN,TIFOR, & **LATT_CUR%A,LATT_CUR%B,IO%IU6**)

   After the modification:

```
CALL CHAIN_FORCE(T_INFO%NIONS,DYN%POSION,TOTEN,TIFOR, & TSIF,LATT_CUR%A,LATT_CUR%B,IO%IU6)
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to modify the compilation configuration file:

1. Open the compilation configuration file.
   ```
   vi vasp-5.4.4/src/.objects
   ```
2. Press **i** to enter the insert mode and add the following content between line 71 and line 72:
   ```
   bfgs.o dynmat.o instanton.o lbfgs.o sd.o cg.o dimer.o bbm.o \
   fire.o lanczos.o neb.o qm.o opt.o \
   ```

   NOTE

   Ensure that no space exists after a backslash (\). Replace spaces at the beginning of a line with tabs.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 See **Step 6** to perform compilation and installation.

```
make all
```

----End

3.6.6.2.2 Installing the Wannier90 Interfaces

Wannier90 has built-in VASP interfaces. VASP 5.4.*X* can only be connected to Wannier90-v1.2.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the **wannier90-1.2** installation package in the VASP installation directory:

```
cd /path/to/VASP
tar -xvf v1.2.tar.gz
```

Step 3 Run the following commands to replace **make.sys** with **make.sys.gfort**:

```
cd wannier90-1.2
cp config/make.sys.gfort make.sys
```

Step 4 Run the following command to modify the **make.sys** file:

1. Open **make.sys**.
   ```
   vi make.sys
   ```
2. Press **i** to enter the insert mode and add the following line to the end of the **make.sys** file:
   ```
   LIBS = -L/path/to/SCALAPACK -L/path/to/OPENBLAS/lib -lscalapack -lopenblas
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to perform installation:

```
make wannier lib test
```

The **libwannnier.a** function library is generated, which needs to be added to the VASP installation and compilation.

Step 6 Run the following commands to modify the **makefile.include** file:

1. Open **makefile.include**.
   cd */path/to/VASP*/**vasp-5.4.4**
   vi makefile.include
2. Press **i** to enter the insert mode and modify the file as follows:
   Modify line 10 as follows:
   -Duse_shmem -Dtbdyn -DVASP2WANNIER90
   Add the following information before line 24:
   WANNIER90 = ***/path/to/VASP***/**wannier90-1.2/libwannier.a**
   Add the following interface parameter to **LLIBS**:
   LLIBS = **$(WANNIER90) -L***/path/to/SCALAPACK* **-lscalapack -L***/path/to/SCALAPACK*/**lib –lopenblas**
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to perform compilation:
make all

----End

3.6.7 Running and Verifying VASP

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create and go to the working directory:
mkdir -p */path/to/CASE*
cd */path/to/CASE*

Step 3 Prepare the test case described in **Table 3-28** and copy the case to the */path/to/CASE* directory.

Step 4 Run the following commands to set environment variables:
export PATH=*/path/to/GNU*/**install/bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/GNU*/**install/lib64:$LD_LIBRARY_PATH**
export PATH=*/path/to/OPENMPI*/**bin:/home/vasp/bin:$PATH**
export LD_LIBRARY_PATH=*/path/to/OPENMPI*/**lib:***/path/to/OPENBLAS*/**lib:***/path/to/SCALAPACK*: */path/to/FFTW*/**lib:$LD_LIBRARY_PATH**

Step 5 Run the following command to create a **host** file:

1. Create **host**.
   vi hostfile
2. Press **i** to enter the insert mode and add the following content:
   node1
   node2
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to start the VASP program:
mpirun --allow-run-as-root -np 96 -x OMP_NUM_THREADS=1 --hostfile hostfile vasp_std 2>&1 | tee -a vasp.out

Table 3-32 Parameters

Parameter	Description
-np	Number of cores to be used.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
--hostfile	Name of the node to be used.

If the information shown in **Figure 3-3** is displayed at the end of the **OUTCAR** log generated by VASP, the VASP program ends normally.

Check the value of **real time** in the **OUTCAR** log. The unit is second. A smaller value indicates better performance.

Figure 3-3 shows the command output.

Figure 3-3 Result example

```
------------------------------------------------------------------------------


   LOOP+:  cpu time 2396.1689: real time  630.2601
  4ORBIT:  cpu time    0.0000: real time    0.0000

total amount of memory used by VASP on root node    43360. kBytes
========================================================================

  base      :      30000. kBytes
  nonl-proj :        637. kBytes
  fftplans  :       5353. kBytes
  grid      :       7240. kBytes
  one-center:          3. kBytes
  wavefun   :        127. kBytes


General timing and accounting informations for this job:
========================================================

                 Total CPU time used (sec):     2408.223
                           User time (sec):     2407.893
                         System time (sec):        0.330
                        Elapsed time (sec):      641.773

                  Maximum memory used (kb):       87424.
                  Average memory used (kb):          0.

                          Minor page faults:        11558
                          Major page faults:            0
                 Voluntary context switches:        29207
END)
```

----End

3.7 ABINIT 8.10.3 Porting Guide (CentOS 7.6)

3.7.1 Introduction

ABINIT is a software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials. Starting from the quantum equations of density functional theory, you can build up to advanced applications with perturbation theories based on DFT, and many-body Green's functions (GW and DMFT). ABINIT can calculate molecules, nanostructures and solids with any chemical composition, and comes with several complete and robust tables of atomic potentials.

For more information, visit the **official ABINIT website**.

 Copyright © Huawei Technologies Co., Ltd.

Programming language: C/Fortran

Brief description: Software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials

Open source license: GPL 3.0

Recommended Software Version

ABINIT 8.10.3

3.7.2 Environment Requirements

Hardware Requirements

Table 3-33 lists the hardware requirements.

Table 3-33 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-34 lists the software requirements.

Table 3-34 Software requirements

Item	Version	Download Address
Abinit	8.10.3	**https://www.abinit.org/packages**
AtomPAW	4.0.1.0	**https://www.abinit.org/fallbacks**
BigDFT	1.7.1.25	**https://www.abinit.org/fallbacks**
OpenBLAS	0.3.6	**https://github.com/xianyi/OpenBLAS/tree/v0.3.6**
libXC	3.0.0	**https://www.abinit.org/fallbacks**
PnetCDF	1.9.0	**https://parallel-netcdf.github.io/wiki/Download.html**
NetCDF-C	4.7.0	**https://www.unidata.ucar.edu/downloads/netcdf/index.jsp**
NetCDF-Fortran	4.4.5	**https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5**
HDF5	1.10.1	**http://hdfgroup.org/HDF5/**
Wannier90	2.0.1.1	**https://www.abinit.org/fallbacks**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
Test case	tbasepar_1.files	Test case provided by the software

OS Requirements

Table 3-35 lists the OS requirements.

Table 3-35 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

3.7.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ABINIT software porting.

Table 3-36 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PNETCDF	Installation path of PnetCDF	
4	/path/to/HDF5	Installation path of HDF5	
5	/path/to/NETCDF	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
6	/path/to/ABINIT	Installation path of ABINIT	

 Copyright © Huawei Technologies Co., Ltd.

3.7.4 Configuring the Compilation Environment

3.7.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-37 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.7.4.2 Installing OpenBLAS**.
3	Installing PnetCDF	See **3.7.4.3 Installing PnetCDF**.
4	Installing HDF5	See **3.7.4.4 Installing HDF5**.
5	Installing NetCDF-C	See **3.7.4.5 Installing NetCDF-C**.
6	Installing NetCDF-Fortran	See **3.7.4.6 Installing NetCDF-Fortran**.

3.7.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install BLAS and LAPACK:

```
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

NOTE

If you use a non-default compiler to install OpenBLAS, download OpenBLAS and use the specified compiler to install it. Specify the OpenBLAS path for compilation and installation in **compilation and installation**.

Step 3 Run the following command to decompress the OpenBLAS installation package:

```
tar xvf OpenBLAS-0.3.6.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd OpenBLAS-0.3.6
```

Step 5 Run the following command to perform the configuration:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 6 Run the following commands to perform the compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 7 Run the following commands to set the OpenBLAS environment variable:

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.7.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd parallel-netcdf-1.9.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --
enable-large-file-test
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the PnetCDF environment variables:

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

3.7.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd hdf5-1.10.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/pat/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the HDF5 environment variable:

```
export LD_LIBRARY_PATH=/pat/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

3.7.4.5 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

3.7.4.6 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make install
```

----End

3.7.5 Obtaining the Source Code

Procedure

Step 1 Download the ABINIT installation package **abinit-8.10.3.tar.gz**.

https://www.abinit.org/

Step 2 Use an SFTP tool to upload the ABINIT installation package to the */path/to/ABINIT* directory on the server.

----End

3.7.6 Compiling and Installing ABINIT

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to set environment variables:

```
export MPI_HOME=$MPI_DIR
export CPPFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path-to/NETCDF/include -I $MPI_HOME/include"
export CFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path-to/NETCDF/include -I $MPI_HOME/include "
export CXXFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path-to/NETCDF/include -I $MPI_HOME/include "
export FCFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export FFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS=" -L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -L/path/to/NETCDF/lib"
```

Step 3 Run the following commands to create an **extral** directory for storing third-party math libraries:

```
cd /path/to/ABINIT
mkdir extral
```

Step 4 Upload the following software packages to the **extral** directory:

- atompaw-4.0.1.0.tar.gz
- bigdft-1.7.1.25.tar.gz
- libxc-3.0.0.tar.gz
- wannier90-2.0.1.1.tar.gz

Step 5 Run the following command to decompress the ABINIT installation package:

```
tar -zxvf abinit-8.10.3.tar.gz
```

Step 6 Run the following command to go to the directory generated after the decompression:

```
cd abinit-8.10.3
```

Step 7 Run the following commands to create and access the directory for compiling the ABINIT software:

 Copyright © Huawei Technologies Co., Ltd.

```
mkdir build
cd build
```

Step 8 Run the following command to perform the configuration:

```
../configure FC=mpif90 --enable-mpi --with-mpi-prefix=/path/to/OPENMPI/ --with-dft-flavor="libxc
+atompaw" --with-trio-flavor="netcdf" --with-tardir=/path/to/ABINIT/extral --with-netcdf-incs="-I/
path/to/NETCDF/include" --with-netcdf-libs="-L/path/to/NETCDF/lib -lnetcdf -L/path/to/NETCDF/lib -
lnetcdff" FCFLAGS="-g -O2 -ffree-line-length-none" --with-linalg-libs="-L/path/to/OPENBLAS/ -lblas -
llapack"
```

Step 9 Run the following commands to perform the compilation and installation:

```
FC=mpif90 CC=mpicc CXX=mpicxx make multi multi_nprocs=96
```

Step 10 Run the following commands to add the environment variables:

```
export ABI_TESTS=/path/to/ABINIT/abinit-8.10.3/tests/
export ABI_TUTORIAL=$ABI_TESTS/tutorial/
export ABI_TUTORESPFN=$ABI_TESTS/tutorespfn/
export ABI_TUTOPARAL=$ABI_TESTS/tutoparal/
export ABI_TUTOPLUGS=$ABI_TESTS/tutoplugs/
export ABI_PSPDIR=$ABI_TESTS/Psps_for_tests/
export PATH=/path/to/ABINIT/abinit-8.10.3/build/src/98_main/:$PATH
```

----End

3.7.7 Running and Verifying ABINIT

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a case directory and copy the case files:

```
cd $ABI_TUTORIAL/Input
mkdir Work_paral
cd Work_paral
cp ../tbasepar_1.files .
cp ../tbasepar_1.in .
```

Step 3 Run the following command:

```
abinit < tbasepar_1.files > log 2> err
```

Step 4 Run the following commands:

1. Create a folder.
   ```
   mkdir tmp
   ```
2. Add a test case file.
   ```
   vi tbase1_x.files
   ```
3. Press **i** to enter the insert mode and add the following content:
   ```
   tbasepar_1.in
   tbasepar_1.out
   tbasepar_1i
   tbasepar_1o
   ./tmp/tbasepar_1 # Change the value to a path that can be accessed by each node, for example, the created tmp path. You are advised to set the value to an absolute path.
   ../../Psps_for_tests/HGH/82pb.4.hgh
   ```
4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
5. Perform the computation.
   ```
   mpirun -np 2 ../../../../build/src/98_main/abinit < tbasepar_1.files >& tbasepar_1.log &
   ```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

Run the following command as the root user:

```
mpirun --allow-run-as-root -np 2 ../../../../build/src/98_main/abinit < tbasepar_1.files >& tbasepar_1.log &
```

Check the value of **overall_cpu_time** (in seconds) in the **tbasepar_1.log** file. A smaller value indicates better performance.

Figure 3-4 shows the command output.

Figure 3-4 Result example

```
 Calculation completed.
.Delivered   1 WARNINGs and   1 COMMENTs to log file.

--- !FinalSummary
program: abinit
version: 8.10.3
start_datetime: Thu Aug 22 14:44:22 2019
end_datetime: Thu Aug 22 14:44:27 2019
overall_cpu_time:            9.5
overall_wall_time:           9.7
exit_requested_by_user: no
timelimit: 0
pseudos:
    Pb  : 8b5eec768eff7fd35f37bcf891d99906
usepaw: 0
mpi_procs: 2
omp_threads: 1
num_warnings: 1
num_comments: 1
...
```

----End

3.7.8 Troubleshooting

Problem 1: An Error Is Reported When Running the configure Command

Symptom

When executing the **configure** command, an error message "error: external libxc support does not work" is displayed.

Possible Causes

--with-libxc-libs= is not written in sequence.

Procedure

Add the following information in sequence after **Step 8**:

```
--with-libxc-libs=-lxcf90 -lxc---
```

Problem 2: An Error Is Reported During ABINIT Compilation (make)

Symptom

When compiling ABINIT by running the **make** command, an error message "Error: Line truncated at (1) [-Werror=line-truncation]" is displayed.

Possible Causes

 Copyright © Huawei Technologies Co., Ltd.

This error occurs because the code does not limit the number of characters to 132.

Procedure

In **Step 8**, add **-ffree-line-length-none** to the **FCFLAGS="-g -O2 -ffree-line-length-none"** command.

Problem 3: An Error Is Reported During ABINIT Compilation (make)

Symptom

When compiling ABINIT by running the **make** command, an error message "File'mpi.mod'opened at (1) is not a GNU Fortran module file" is displayed.

Possible Causes

This error occurs because the GCC version does not match the Open MPI compiler version.

Procedure

Unifying the compilers used by Open MPI and ABINIT can solve the problem.

NOTE

During the installation, Open MPI, BLAS-LAPACK, and NetCDF must use the same compiler. Otherwise, errors may occur.

Problem 4: An Error Is Reported During ABINIT Compilation (make)

Symptom

When compiling ABINIT by running the **make** command, an error message "abinit-8.10.3/config/gnu/missing: line 81: automake-1.16: command not found" is displayed.

Possible Causes

This error occurs because the aclocal and automake versions are set to 1.16 when creating the **Makefile** file.

Procedure

Check the automake version of the system and replace it with the system version in **Makefile**.

Step 1 Run the following command to open the **Makefile** file:

```
vi abinit-8.10.3/build/Makefile
```

Step 2 Press **i** to enter the insert mode and modify the system version in the **Makefile** file. Pay attention to the information in bold.

```
AUTOMAKE = ${SHELL} /storage/softwares/TaiShan/abinit/abinit-8.10.3/config/gnu/missing automake-1.16
```

Step 3 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

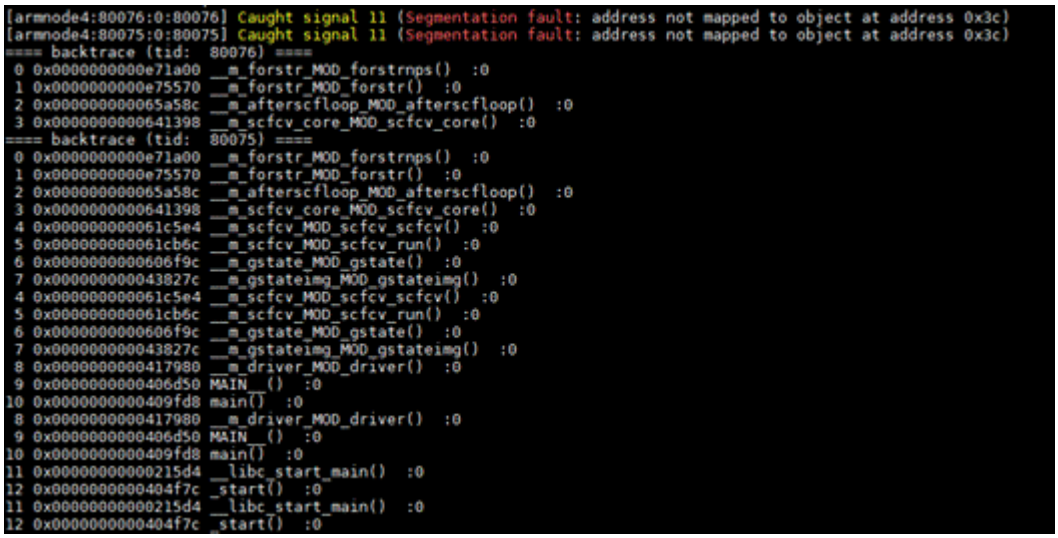
----End

Problem 5: An Error Is Reported When the Case Is Run

Symptom

 Copyright © Huawei Technologies Co., Ltd.

An error is reported when the case is executed. The error information is similar to the segmentation fault:



Possible Causes

The segmentation error is caused by the inconsistency between the compiler used to compile ABINIT and the compiler used to compile Open MPI 4.0.1 and BLAS.

Procedure

Installing the AtomPAW can optimize the memory read/write of data.

Step 1 Run the following command to access the directory where the AtomPAW installation package is located:

```
cd /path/to/ABINIT/extral
```

Step 2 Run the following command to decompress the source package:

```
tar xvf atompaw-4.0.1.0.tar.gz
```

Step 3 Run the following command to go to the source code directory:

```
cd atompaw-4.0.1.0
```

Step 4 Run the following commands to perform the compilation and installation:

```
./configure
make
make install
```

----End

3.7.9 More Information

ABINIT official installation guide:

https://wiki.abinit.org/doku.php?id=howto:getting_started#installation_of_abinit

Official installation guide of the ABINIT third-party link libraries:

https://wiki.abinit.org/doku.php?id=hpc:linking_libraries

 Copyright © Huawei Technologies Co., Ltd.

3.8 AmberTools 19 Porting Guide (CentOS 7.6)

3.8.1 Introduction

AmberTools is a series of modeling, molecular mechanics, and dynamics simulation programs developed by professor Coleman at the University of California for biomolecules. There are many programs, such as building modules, which arrange solvent moisture molecules and perform charge fitting, and analysis tools can be used to perform NMR refinement and track analysis of dynamic calculations.

For details about AmberTools, visit the **official AmberTools website**.

Programming language: C/C++/Fortran

Brief description: Molecular mechanics and dynamics simulation program.

Open source license: GPL 3.0

Recommended Software Version

AmberTools 19

3.8.2 Environment Requirements

Hardware Requirements

Table 3-38 lists the hardware requirements.

Table 3-38 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-39 lists the software requirements.

Table 3-39 Software requirements

Item	Version	Download Address
AmberTools	19	**http://ambermd.org/GetAmber.php#amber**
Test case	nab	Provided by the AmberTools 19 software

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 3-40 lists the OS requirements.

Table 3-40 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.8.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AmberTools software porting.

Table 3-41 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/AMBER	Installation path of AmberTools	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

3.8.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-42 Configuration process

No.	Operation	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.8.5 Obtaining the Source Code

Procedure

Step 1 Download the AmberTools installation package **AmberTools19.tar.bz2**.

http://ambermd.org/GetAmber.php#amber

Step 2 Use an SFTP tool to upload the AmberTools installation package to the */path/to/AMBER* directory on the server.

----End

3.8.6 Compiling and Installing AmberTools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to set environment variables:

```
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following commands to decompress the AmberTools installation package:

```
cd /path/to/AMBER
tar -xvf AmberTools19.tar.bz2
```

Step 4 Run the following commands to update the **config.guess** and **config.sub** scripts in the **fftw3-3.3** directory:

```
cd /path/to/AMBER/amber18/AmberTools/src/fftw-3.3
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD
vi config.guess
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD
vi config.sub
```

Step 5 Run the following commands to update the **config.guess** and **config.sub** scripts in the **xblas** directory:

```
cd /path/to/AMBER/amber18/AmberTools/src/xblas/config
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD
vi config.guess
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD
vi config.sub
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following commands to perform serial compilation and installation:

```
cd /path/to/AMBER/amber18
./configure --skip-python -nosse gnu
test -f /path/to/AMBER/amber18/amber.sh && source /path/to/AMBER/amber18/amber.sh
make install
```

Step 7 Run the following command to perform a serial test:

```
make test
```

Step 8 Run the following commands to perform parallel compilation and installation:

```
cd /path/to/AMBER/amber18
./configure --skip-python -nosse -mpi gnu
make install
export DO_PARALLEL="mpirun --allow-run-as-root -np 96 -mca btl ^openib"
./configure --skip-python -nosse -openmp gnu
make openmp
```

----End

3.8.7 Running and Verifying AmberTools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the case directory of AmberTools 18:

```
cd /path/to/AMBER/amber18/AmberTools/benchmarks/nab
```

Step 3 Run the following commands to modify the **bench_amber** file:

1. Open **bench_amber**.

   ```
   vi bench_amber
   ```

2. Press **i** to enter the insert mode and modify the **bench_amber** file.

   ```
   #!/bin/sh
   sander=sander.MPI
   cat<<EOF  > mdin
         benchmark a short md
   &cntrl
         igb=1, cut=20.0, rgbmax=20.0,
         tempi=50.0, temp0=100.0, tautp=0.4, ntt=1,
         ntb=0, nstlim=1000, ntpr=10,
         ntx=1, irest=0,
         ntc=2, ntf=2, tol=0.0000001,
   /
   EOF
   $DO_PARALLEL $sander -O -i mdin -p $1.top -c $1.mc.x -o $1.md.o
   /bin/rm mdin restrt mdinfo
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

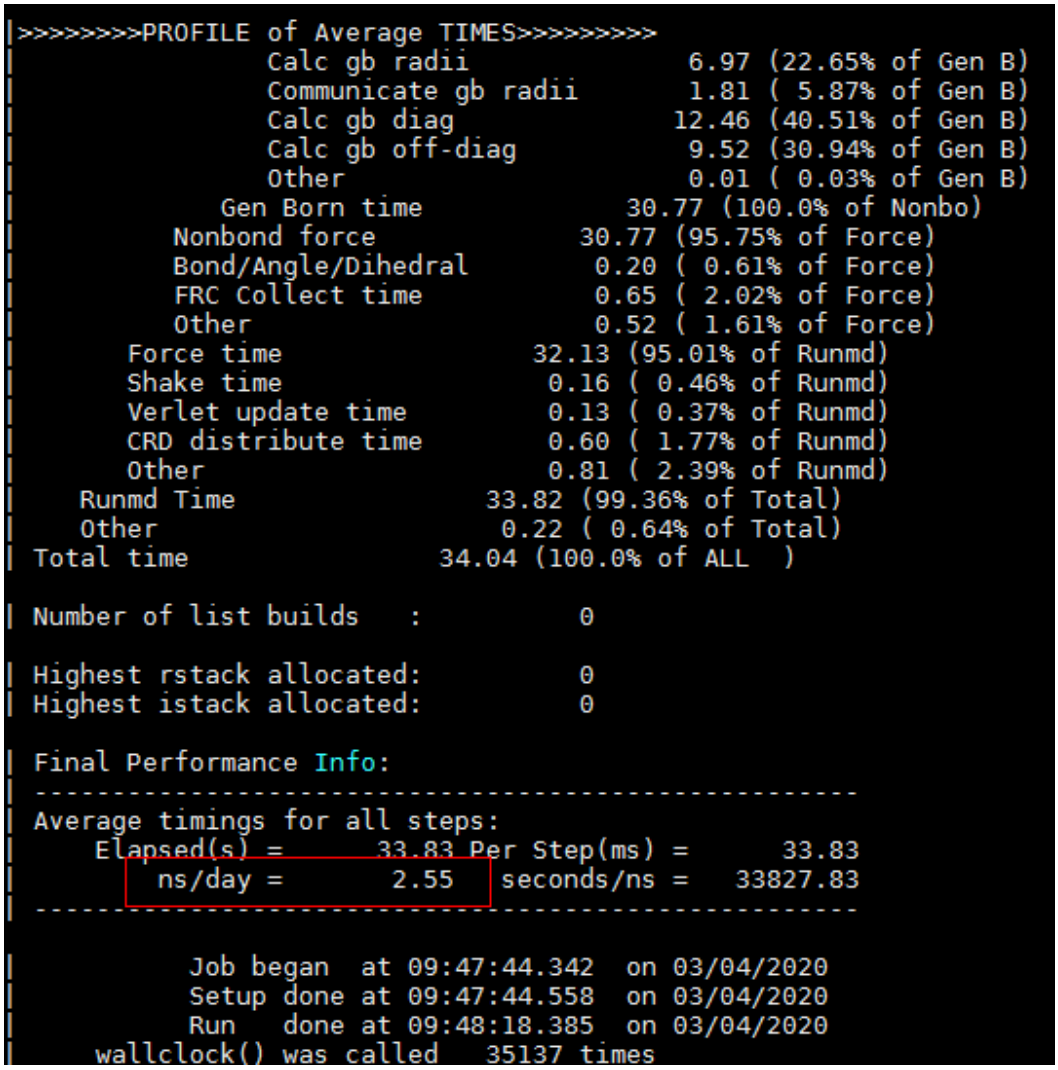
Step 4 Run the following command to perform parallel computing:

```
./bench_amber halfam0
```

View the **ns/day** value in the **halfam0.md.o** log file. A larger value indicates better performance.

Figure 3-5 shows an example of the output result.

Figure 3-5 An example result



----End

3.8.8 Troubleshooting

Problem 1: An Error Is Reported When configure Is Run

Symptom:

An error message "Error: FFTW configure returned 1" is displayed when running the **configure** command.

Possible causes:

- The FFTW3 certificate provided by AmberTools has expired.
- The **-nosse** parameter is not added when the **configure** command is executed.

Procedure:

Obtain the latest **config.guess** and **config.sub** files based on the website displayed in the error message. Run the **vi** command to update the **config.guess**

 Copyright © Huawei Technologies Co., Ltd.

and **config.sub** files of FFTW3. Add the **-nosse** parameter when running the **configure** command.

3.9 CP2K 7.1 Porting Guide (CentOS 7.6)

3.9.1 Introduction

CP2K is a quantum chemistry and solid-state physics package that can perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It provides a general framework for different modeling methods. It supports DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, and MNDO), and classical force fields (AMBER and CHARMM). CP2K can use NEB or dimer method to do simulations of molecular dynamics, metadynamics, Monte Carlo, Ellenfast dynamics, vibration analysis, core level spectroscopy, energy minimization, and transition state optimization.

For more information about CP2K, visit the **official CP2K website**.

Programming language: Fortran 2008

Brief description: a quantum chemistry and solid-state physics package used to perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems

Open source license: GPL

NOTE

For CP2K 7.1 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port CP2K to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

CP2K 7.1

3.9.2 Environment Requirements

Hardware Requirements

Table 3-43 lists the hardware requirements.

Table 3-43 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-44 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-44 Software requirements

Item	Version	Download Address
CP2K	7.1	https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz
Libint	2.6.0	https://github.com/evaleev/libint/archive/v2.6.0.tar.gz
Libxc	4.3.4	https://www.cp2k.org/static/downloads/libxc-4.3.4.tar.gz
FFTW	3.3.8	https://www.cp2k.org/static/downloads/fftw-3.3.8.tar.gz
LAPACK	3.8.0	https://www.cp2k.org/static/downloads/lapack-3.8.0.tgz
ScaLAPACK	2.1.0	https://www.cp2k.org/static/downloads/scalapack-2.1.0.tgz
CMake	3.16.4	https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz
GMP	6.2.0	https://gmplib.org/download/gmp/gmp-6.2.0.tar.xz
Boost	1.72	https://boostorg.jfrog.io/artifactory/main/release/1.72.0/source/boost_1_72_0.tar.gz
DBCSR	2.0.1	https://github.com/cp2k/dbcsr/releases/download/v2.0.1/dbcsr-2.0.1.tar.gz
ELPA	2019.05.001	https://www.cp2k.org/static/downloads/elpa-2019.05.001.tar.gz
Spglib	1.12.2	https://github.com/spglib/spglib/archive/v1.11.2.1.tar.gz
GSL	2.6	http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz
PLUMED	2.5.2	https://www.cp2k.org/static/downloads/plumed-2.5.2.tgz
Test case	H2O-256.inp	Provided by the software.

OS Requirements

Table 3-45 lists the OS requirements.

Table 3-45 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

3.9.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CP2K software porting.

Table 3-46 Paths for software porting

No.	Software Installation Path	Usage	Description
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/GMP	Installation path of GMP.	
4	/path/to/BOOST	Installation path of Boost.	
5	/path/to/LIBINT	Installation path of Libint.	
6	/path/to/FFTW	Installation path of FFTW.	
7	/path/to/LAPACK	Installation path of LAPACK.	
8	/path/to/SCALAPACK	Installation path of ScaLAPACK.	
9	/path/to/ELPA	Installation path of ELPA.	
10	/path/to/SPGLIB	Installation path of Spglib.	
11	/path/to/LIBXC	Installation path of Libxc.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Description
12	/path/to/GSL	Installation path of GSL.	
13	/path/to/PLUMED	Installation path of PLUMED.	
14	/path/to/EXTRA	Path of storing the library files generated by Libint, FFTW, LAPACK, ScaLAPACK, ELPA, Spglib, Libxc, GSL, and PLUMED.	

3.9.4 Configuring the Compilation Environment

3.9.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-47 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **3.9.4.2 Installing CMake**.
3	Installing GMP	See **3.9.4.3 Installing GMP**.
4	Installing Boost	See **3.9.4.4 Installing Boost**.
5	Installing Libint	See **3.9.4.5 Installing Libint**.
6	Installing FFTW	See **3.9.4.6 Installing FFTW**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
7	Installing LAPACK	See **3.9.4.7 Installing LAPACK**.
8	Installing ScaLAPACK	See **3.9.4.8 Installing ScaLAPACK**.
9	Installing ELPA	See **3.9.4.9 Installing ELPA**.
10	Installing Spglib	See **3.9.4.10 Installing Spglib**.
11	Installing Libxc	See **3.9.4.11 Installing Libxc**.
12	Installing GSL	See **3.9.4.12 Installing GSL**.
13	Installing PLUMED	See **3.9.4.13 Installing PLUMED**.

3.9.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the CMake installation package:

```
tar -xvf cmake-3.16.4.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd cmake-3.16.4
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Run the following command to perform compilation and installation:

```
make -j&&make install
```

Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.9.4.3 Installing GMP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GMP installation package:

```
tar -xvf gmp-6.2.0.tar.xz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following commands to perform configuration:

```
cd gmp-6.2.0
./configure --prefix=/path/to/GMP
```

Step 4 Run the following command to perform compilation and installation:

```
make -j&&make install
```

----End

3.9.4.4 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Boost installation package:

```
tar -xvf  boost_1_72_0.tar.gz
```

Step 3 Run the following command to perform configuration:

```
./bootstrap.sh
```

Step 4 Run the following command to perform compilation and installation:

```
./b2 install --prefix=/path/to/BOOST
```

----End

3.9.4.5 Installing Libint

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependency package using Yum:

```
yum install gmp-devel.aarch64 libudev* -y
```

Step 3 Run the following commands to create the installation directory:

```
mkdir -p /path/to/EXTRA
mkdir -p /path/to/EXTRA/mathlib
```

Step 4 Run the following command to decompress the Libint installation package:

```
tar -xvf libint-2.6.0.tar.gz
```

Step 5 Run the following commands to perform configuration:

```
cd libint-2.6.0
./autogen.sh
```

Step 6 Run the following commands to create a **build** directory and switch to the directory:

```
mkdir build
cd build
```

Step 7 Run the following commands to configure the environment:

```
export LDFLAGS="-L/path/to/GMP/gmp-6.2/lib -L/path/to/BOOST/boost_1_72/lib"
export CPPFLAGS="-I/path/to/BOOST/boost_1_72/include/ -I/path/to/GMP/gmp-6.2/include"
```

Step 8 Run the following command to perform configuration:

```
../configure CXX=mpicxx --enable-eri=1 --enable-eri2=1 --enable-eri3=1 --with-max-am=4 --with-eri-max-am=4,3 --with-eri2-max-am=6,5 --with-eri3-max-am=6,5 --with-opt-am=3 --enable-generic-code --disable-unrolling --with-libint-exportdir=libint_cp2k_lmax4
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 Run the following command to perform compilation:

```
make export
```

Step 10 Run the following commands to decompress the **libint_cp2k_lmax4.tgz** installation package:

```
tar -xvf libint_cp2k_lmax4.tgz
cd libint_cp2k_lmax4
```

Step 11 Run the following command to perform configuration:

```
./configure --prefix=/path/to/EXTRA/libint2 CC=mpicc CXX=mpicxx FC=mpifort --enable-fortran --enable-shared
```

Step 12 Run the following commands to perform compilation and installation:

```
make -j 32
make install
```

----End

3.9.4.6 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform configuration:

```
./configure CC=gcc F77=gfortran --enable-shared --enable-threads --enable-openmp --enable-mpi MPICC=mpicc --prefix= /path/to/EXTRA/fftw3
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End

3.9.4.7 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the LAPACK installation package:

```
tar -xvf lapack-3.8.0.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cp make.inc.example make.inc
```

Step 4 Run the following command to perform compilation and installation:

```
make -j
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following command to copy the static library to another directory:

```
cp *.a /path/to/EXTRA/mathlib
```

----End

3.9.4.8 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar  -xvf scalapack-2.1.0.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd scalapack-2.1.0
```

Step 4 Run the following command to copy and paste the file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following commands to modify the **SLmake.inc** file:

1. Open **SLmake.inc**.
   ```
   vim SLmake.inc
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   BLASLIB     = /path/to/EXTRA/librefblas.a
   LAPACKLIB  = /path/to/EXTRA/liblapack.a
   LIBS        = $(LAPACKLIB) $(BLASLIB)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 6 Run the following command to perform compilation:

```
make -j
```

Step 7 Run the following command to copy the generated static library to another directory:

```
cp *.a /path/to/EXTRA/mathlib
```

----End

3.9.4.9 Installing ELPA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the ELPA installation package:

```
tar -xvf elpa-2019.05.001.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd elpa-2019.05.001
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/EXTRA/elpa --enable-openmp --enable-shared=no LIBS="//path/to/EXTRA/mathlib/libscalapack.a /path/to/EXTRA/mathlib/liblapack.a /path/to/EXTRA/mathlib/librefblas.a" --disable-sse --disable-sse-assembly --disable-avx --disable-avx2
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to perform compilation:

```
make
make install
```

----End

3.9.4.10 Installing Spglib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Spglib installation package:

```
tar -xvf spglib-1.12.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd spglib-1.12.2
```

Step 4 Run the following commands to create a **build** directory and switch to the directory:

```
mkdir build
cd build
```

Step 5 Run the following command to perform configuration:

```
cmake .. -DCMAKE_INSTALL_PREFIX="/path/to/EXTRA/spglib112"
```

Step 6 Run the following commands to perform compilation:

```
make -j
make install
```

----End

3.9.4.11 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Libxc installation package:

```
tar -xvf libxc-4.3.4.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd libxc-4.3.4
```

Step 4 Run the following command to perform configuration:

```
./configure FC=gfortran CC=gcc --prefix=/path/to/EXTRA/libxc434
```

Step 5 Run the following commands to perform compilation:

```
make
make install
```

----End

3.9.4.12 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GSL installation package:

```
tar -xvf gsl-2.6.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd gsl-2.6
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/EXTRA/gsl
```

Step 5 Run the following commands to perform compilation:

```
make
make install
```

Step 6 Run the following commands to set environment variables:

```
export LIBRARY_PATH=/path/to/EXTRA/gsl/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/EXTRA/gsl/lib:$LD_LIBRARY_PATH
```

----End

3.9.4.13 Installing PLUMED

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PLUMED installation package:

```
tar -xvf plumed-2.5.2.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd plumed-2.5.2
```

Step 4 Run the following command to perform configuration:

```
./configure CXX=mpicxx CC=mpicc FC=mpifort --prefix=/path/to/EXTRA/plumed252 --enable-external-blas --enable-gsl --enable-external-lapack LDFLAGS=-L/path/to/EXTRA/mathlib LIBS="-lrefblas –llapack"
```

Step 5 Run the following commands to perform compilation:

```
make
make install
```

----End

3.9.5 Obtaining the Source Code

Procedure

Step 1 Download the CP2K installation package **cp2k-7.1.tar.gz**.

https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the CP2K installation package to the */path/to/CP2K* directory on the server.

----End

3.9.6 Compiling and Installing CP2K

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the CP2K installation package:

```
tar xvf cp2k-7.1.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd cp2k-7.1.0/arch
```

Step 4 Run the following command to modify the configuration file:

1. Open the configuration file.

   ```
   vi Linux-x86-64-gfortran.psmp
   ```

2. Press **i** to enter the insert mode and modify the file.

   ```
   GNU_PATH      = /path/to/EXTRA/
   MATHLIBPATH  = /path/to/EXTRA/mathlib
   include     $(GNU_PATH)/plumed252/lib/plumed/src/lib/Plumed.inc.static
   ELPA_VER    = 2019.05.001
   ELPA_INC    = $(GNU_PATH)/elpa/include/elpa_openmp-$(ELPA_VER)
   ELPA_LIB    = $(GNU_PATH)/elpa/lib
   FFTW_INC    = $(GNU_PATH)/fftw3/include
   FFTW_LIB    = $(GNU_PATH)/fftw3/lib
   LIBINT_INC   = $(GNU_PATH)/libint2/include
   LIBINT_LIB   = $(GNU_PATH)/libint2/lib
   LIBXC_INC    = $(GNU_PATH)/libxc434/include
   LIBXC_LIB    = $(GNU_PATH)/libxc434/lib
   SPGLIB_INC   = $(GNU_PATH)/spglib112/include
   SPGLIB_LIB   = $(GNU_PATH)/spglib112/lib
   CFLAGS       = -O2 -g -mtune=native
   DFLAGS       = -D__ELPA -D__FFTW3 -D__LIBINT -D__LIBXC
   DFLAGS       += -D__MPI_VERSION=3 -D__PLUMED2 -D__SPGLIB
   DFLAGS       += -D__parallel -D__SCALAPACK
   FCFLAGS      = $(CFLAGS) $(DFLAGS)
   FCFLAGS      += -ffree-form -ffree-line-length-none
   FCFLAGS      += -fopenmp
   FCFLAGS      += -ftree-vectorize -funroll-loops -std=f2008
   FCFLAGS      += -I$(ELPA_INC)/elpa -I$(ELPA_INC)/modules
   FCFLAGS      += -I$(FFTW_INC) -I$(LIBINT_INC) -I$(LIBXC_INC)
   LDFLAGS      = $(FCFLAGS)
   LIBS        = $(PLUMED_DEPENDENCIES) -lgsl -lgslcblas -lz
   LIBS        += $(ELPA_LIB)/libelpa_openmp.a
   LIBS        += $(LIBXC_LIB)/libxcf03.a
   LIBS        += $(LIBXC_LIB)/libxc.a
   LIBS        += $(LIBINT_LIB)/libint2.a
   LIBS        += $(SPGLIB_LIB)/libsymspg.a
   LIBS        += $(FFTW_LIB)/libfftw3.a
   LIBS        += $(FFTW_LIB)/libfftw3_threads.a
   LIBS        += $(MATHLIBPATH)/libscalapack.a
   LIBS       += $(MATHLIBPATH)/liblapack.a
   LIBS       += $(MATHLIBPATH)/librefblas.a
   LIBS       += -ldl -lpthread -lstdc++
   ```

Step 5 Run the following commands to go to the directory:

```
cd /path/to/CP2K/exts/dbcsr
cp dbcsr-2.1.0-rc12.tar.gz ./
```

Step 6 Run the following commands to decompress the **dbcsr-2.1.0-rc12.tar.gz** installation package and go to the directory generated after the decompression:

```
tar -xvf dbcsr-2.1.0-rc12.tar.gz
cd dbcsr-2.1.0-rc12
```

Step 7 Run the following command to move all the files in **dbcsr-2.1.0-rc12** to **dbcsr**:

```
mv * ../
```

Step 8 Run the following command to compile and install CP2K:

```
make -j 16 ARCH=Linux-x86-64-gfortran VERSION=psmp
```

----End

3.9.7 Running and Verifying CP2K

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to load the environment variable:

```
export PATH=/path/to/CP2K/exe/Linux-x86-64-gfortran:$PATH
```

Step 3 Run the following command as the root user to start the computation:

```
mpirun --allow-run-as-root -np 128 -x OMP_NUM_THREADS=1 cp2k.psmp H2O-256.inp > cp2k.H2O-256.inp.log
```

The time difference between **Figure 3-6** and **2** is 927s.

Figure 3-6 Output example (1)

```
 **** **** ******  **  PROGRAM STARTED AT                2020-06-18 20:51:55.469
***** ** ***  *** **   PROGRAM STARTED ON                           RSA320V2-7
**    ****   ******    PROGRAM STARTED BY                                 root
***** **    ** ** **   PROGRAM PROCESS ID                              115140
 **** **  *******  **  PROGRAM STARTED IN /storage/liu/software/CP2K/cp2k-7.1.0
                                         /cp2k-7.1.0/benchmarks/QS
```

Figure 3-7 Output example (2)

```
The number of warnings for this run is : 2

 -----------------------------------------------------------------------------
 **** **** ******  **  PROGRAM ENDED AT                  2020-06-18 21:07:22.935
***** ** ***  *** **   PROGRAM RAN ON                              RSA320V2-7
**    ****   ******    PROGRAM RAN BY                                     root
***** **    ** ** **   PROGRAM PROCESS ID                              115140
 **** **  *******  **  PROGRAM STOPPED IN /storage/liu/software/CP2K/cp2k-7.1.0
                                         /cp2k-7.1.0/benchmarks/QS
(END)
```

----End

3.9.8 More Information

CP2K installation guide:

https://www.cp2k.org/howto:compile

3.10 NWChem 6.8.1 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

3.10.1 Introduction

NWChem is computational chemistry software running on high-performance parallel supercomputers and common workstation clusters. NWChem uses standard quantum mechanics to describe electron wave functions or densities, to calculate the properties of molecules and periodic systems, and to simulate classical molecular dynamics and free energy.

For more information about NWChem, visit the **NWChem official website**.

Programming language: Python

Brief description: scalable computational chemistry tools

Open source license: ECL 2.0

NOTE

For NWChem 6.8.1 on CentOS 7.6, you are advised to use the dedicated software porting function of the Kunpeng DevKit to port NWChem to the Kunpeng platform. For details, see **Dedicated Software Porting**.

Recommended Software Version

NWChem 6.8.1

3.10.2 Environment Requirements

Hardware Requirements

Table 3-48 lists the hardware requirements.

Table 3-48 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-49 lists the software requirements.

Table 3-49 Software Requirements

Item	Version	How to Obtain
NWChem	6.8.1	**https://codeload.github.com/NWChemgit/NWChem/tar.gz/6.8.1-release**
GlobalArray	5.6.5	**https://github.com/edoapra/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz**
Test case	siosi8.nw	Provided by the software.

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 3-50 lists the OS requirements.

Table 3-50 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.10.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NWChem software porting.

Table 3-51 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/NWCHEM	Installation path of NWChem	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

3.10.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-52 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.10.5 Obtaining the Source Code

Procedure

Step 1 Download the **NWChem-6.8.1-release.tar.gz** installation package.

https://codeload.github.com/NWChemgit/NWChem/tar.gz/6.8.1-release

Step 2 Use an SFTP tool to upload the NWChem installation package to the */path/to/NWCHEM* directory on the server.

----End

3.10.6 Compiling and Installing NWChem

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to set environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 3 Run the following command to decompress the NWChem installation package:

```
cd /path/to/NWCHEM
tar -zxvf NWChem-6.8.1-release.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd NWChem-6.8.1-release
```

Step 5 Run the following commands to decompress the GlobalArray installation package to the **src/tools** directory:

```
cd src/tools
cp ga-5.6.5.tar.gz /path/to/NWCHEM/NWChem-6.8.1-release/src/tools ./
tar -zxvf ga-5.6.5.tar.gz
```

Step 6 Run the following command to perform configuration:

```
export NWCHEM_TOP=/path/to/NWCHEM/NWChem-6.8.1-release
export NWCHEM_TARGET=LINUX64
export NWCHEM_MODULES=all
export USE_MPI=y
export USE_INTERNALBLAS=y
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Run the following commands to perform compilation and installation:

```
cd /path/to/NWCHEM/NWChem-6.8.1-release/src
make nwchem_config
make FC=gfortran >& make.log
```

Step 8 Run the following command to add the environment variables:

```
export PATH=/path/to/NWCHEM/NWChem-6.8.1-release/bin/LINUX64:$PATH
```

----End

3.10.7 Running and Verifying NWChem

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the case folder:

```
cd /path/to/NWCHEM/NWChem-6.8.1-release/web/benchmarks/dft
```

Step 3 Run the following command to create a **hostfile** file:

1. Create **hostfile**.

   vi hostfile

2. Press **i** to enter the insert mode and add the following content:

   ```
   Node1
   Node2
   ```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the settings and exit.

Step 4 Run the following command to clear the cache:

```
clush -w Node[1-2] "sync;echo 3 > /proc/sys/vm/drop_caches"
```

Step 5 Run the following command to perform calculation:

```
mpirun --allow-run-as-root -N 96 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH -mca btl ^openib --hostfile hostfile nwchem siosi8.nw -v 2>&1 | tee   NWChem_examples.log
```

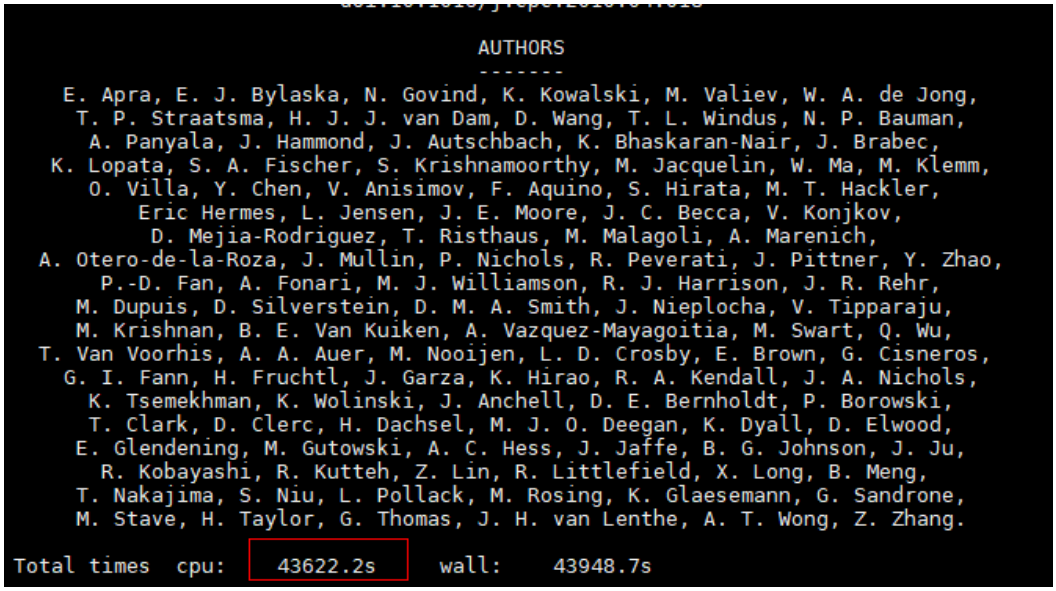
NOTE

- **--hostfile hostfile**: specifies the list of node names to be used. In this example, two nodes are used.
- **-N 96**: indicates that each node runs 96 processes. (The 96-core CPU of a single node is used as an example.)

Check the value of **Total times cpu** (in seconds) in **NWChem_examples.log**. A smaller value indicates better performance.

Figure 3-8 shows an example of the output result:

Figure 3-8 An example result



----End

3.10.8 Troubleshooting

Problem 1: An Error Is Reported When make NWChem_config Is Run

Symptom:

The error message "guess-mpidefs: command not found..." was displayed when **make NWChem_config** was run.

Possible causes:

The GlobalArray software package is not decompressed to the **src/tools** directory, and the device cannot access the Internet.

Procedure:

Decompress the GlobalArray software package to the **src/tools** directory or connect the device to the Internet.

3.11 ROOT 6.20 Porting Guide (CentOS 7.6)

3.11.1 Introduction

ROOT is a modular scientific software toolkit. It provides all the functions required for big data processing, statistical analysis, visualization, and storage.

For details about ROOT, visit the **official ROOT website**.

Programming language: C++/Python

Brief description: Modular scientific software toolkit

Open source license: GPL 3.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Version

ROOT 6.20

3.11.2 Environment Requirements

Hardware Requirements

Table 3-53 lists the hardware requirements.

Table 3-53 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-54 lists the software requirements.

Table 3-54 Software requirements

Item	Version	Download Address
ROOT	6.20	https://root.cern/download/root_v6.20.00.source.tar.gz
CMake	3.17	https://gitlab.kitware.com/cmake/cmake/-/archive/v3.17.0-rc2/cmake-v3.17.0-rc2.tar.gz
zstd	1.3.4	https://github.com/facebook/zstd/archive/refs/tags/v1.3.4.tar.gz
Test case	benchmarks.C	Provided by the software
	simple	Provided by the software

OS Requirements

Table 3-55 lists the OS requirements.

Table 3-55 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

3.11.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ROOT software porting.

Table 3-56 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/ZSTD	Installation path of zstd	
4	/path/to/ROOT	Installation path of ROOT	

3.11.4 Configuring the Compilation Environment

3.11.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-57 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **3.11.4.2 Installing CMake**.
3	Installing zstd	See **3.11.4.3 Installing zstd**.

 Copyright © Huawei Technologies Co., Ltd.

3.11.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependencies:

```
yum install -y gcc gcc-c++ libcurl-devel.aarch64 cmake binutils libX11-devel libXpm-devel libXft-devel libXext-devel openssl-devel pcre-devel mesa-libGL-devel mesa-libGLU-devel glew-devel ftgl-devel mysql-devel fftw-devel cfitsio-devel graphviz-devel avahi-compat-libdns_sd-devel libldap-dev python-devel libxml2-devel gsl-static
```

Step 3 Run the following commands to decompress the CMake installation package:

```
tar -xzvf cmake-v3.17.0-rc2.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd cmake-v3.17.0-rc2
```

Step 5 Run the following commands to perform compilation and installation:

```
./configure --prefix=/path/to/CMAKE
make -j16
make install
```

Step 6 Run the following command to set the CMake environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.11.4.3 Installing zstd

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the zstd installation package:

```
tar -zxvf zstd-1.3.4.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd zstd-1.3.4
```

Step 4 Run the following commands to perform compilation and installation:

```
make -j16
make install
```

Step 5 Run the following commands to set the zstd environment variables:

```
export LD_LIBRARY_PATH=/path/to/ZSTD/zstd-1.3.4/lib:$LD_LIBRARY_PATH
```

----End

3.11.5 Obtaining the Source Code

Procedure

Step 1 Download the root installation package **root_v6.20.00.source.tar.gz**.

https://root.cern/download/root_v6.20.00.source.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the ROOT installation package to the */path/to/ROOT* directory on the server.

----End

3.11.6 Compiling and Installing ROOT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/ROOT
```

Step 3 Run the following commands to decompress the ROOT installation package:

```
tar -xzvf root_v6.20.00.source.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd root-6.20.00
```

Step 5 Run the following command to create and access the directory for compiling the ROOT software:

```
mkdir obj
cd obj
```

Step 6 Ensure that the server is connected to the Internet and run the following commands to compile and install the software:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=/opt/xrootd -DENABLE_PERL=FALSE
make -j16
make install
```

Step 7 Run the following command to add the environment variables:

```
source ./bin/thisroot.sh
```

----End

3.11.7 Running and Verifying ROOT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/ROOT/root-6.20.00/obj/tutorials/legacy
```

Step 3 Run the following command to open ROOT:

```
root
```

Step 4 Run the following command to perform a benchmark test:

```
.x benchmarks.C
```

Check the value of **ROOTMARKS** in the command output. A larger value indicates better single-core performance.

Figure 3-9 shows the command output.

Figure 3-9 Result example

```
---------------ROOT 6.20/00 benchmarks summary (in ROOTMARKS)-----
   For comparison, a Pentium IV 2.4Ghz is benchmarked at 600 ROOTMARKS
hsimple     =  300.34 RealMARKS,  = 1032.00 CpuMARKS
hsum        =  419.53 RealMARKS,  =  847.06 CpuMARKS
fillrandom  = 1737.90 RealMARKS,  =  600.00 CpuMARKS
fit1        =  174.61 RealMARKS,  =  180.00 CpuMARKS
tornado     = 3385.81 RealMARKS,  =     inf CpuMARKS
na49        = 67236.56 RealMARKS,  = 83400.00 CpuMARKS
geometry    = 1688.39 RealMARKS,  = 3600.00 CpuMARKS
na49view    = 1345.04 RealMARKS,  = 3000.00 CpuMARKS
ntuple1     = 1432.93 RealMARKS,  =  840.00 CpuMARKS

**********************************************************
* Your machine is estimated at 1018.89 ROOTMARKS   *
**********************************************************
```

Step 5 Run the following command to exit ROOT.

```
.q
```

Step 6 Run the following command to go to the **proof** directory:

```
cd /path/to/ROOT/root-6.20.00/obj/tutorials/proof
```

Step 7 Run the following command to open ROOT:

```
root
```

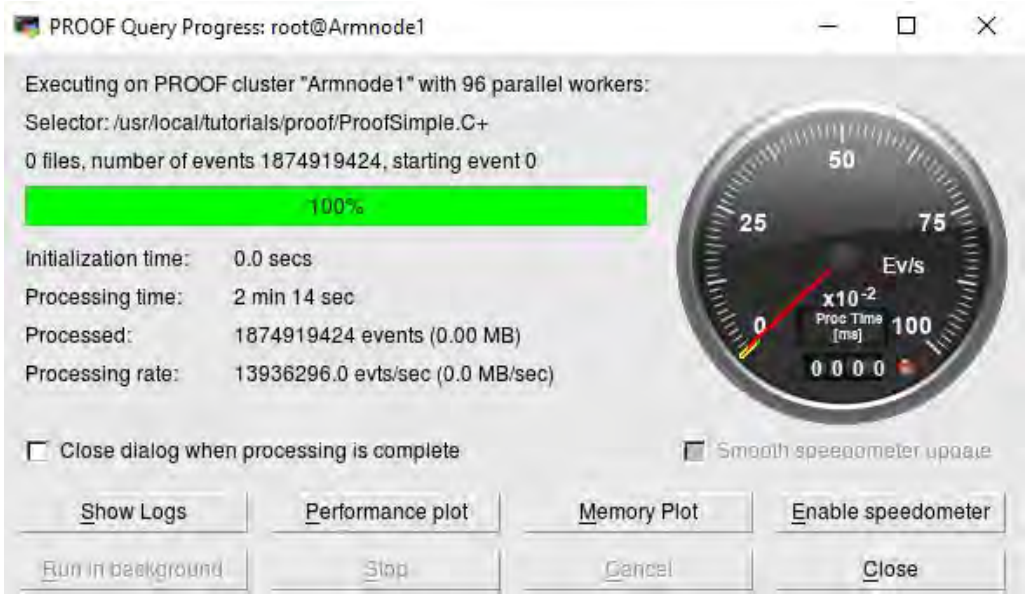
Step 8 Run the following commands to perform a simple test:

```
.L runProof.C
runProof("simple(nevt=10000000000000000,nhist=100)","lite://")
```

The test progress is displayed. When the progress reaches **100%**, check the value of **Processing rate**. A larger value indicates better multi-core performance.

Figure 3-10 shows an example of the dialog box.

Figure 3-10 Example



----End

 Copyright © Huawei Technologies Co., Ltd.

3.11.8 Troubleshooting

Problem 1: An Error Is Reported When Running CMake

Symptom:

"Error in <TInterpreter::InspectMembers>: TClass and cling disagree on the size of the class TBaseClass" is displayed when running CMake.

Possible Causes:

The GNU version is incorrect.

Procedure:

Check whether the GNU version matches the compilation environment and whether the **/usr/local/lib** library is correct. For details, see **Step 5** and **Step 4**.

Problem 2: An Error is Reported When Running runProof

Symptom:

"SysError in <TUnixSystem::UnixTcpConnect>: connect (localhost:40000) (Connection refused)" is displayed when running **runProof("simple(nevt=10000000000000000,nhist=100)")**.

Possible Causes:

You do not need to start the xproofd daemon the runProof is run on a local server.

Procedure:

Add the **lite://** parameter when running runProof. For details, see **Step 8**.

3.11.9 More Resources

Official ROOT installation guide:

https://root.cern/install/build_from_source/

Official ROOT forum:

https://root-forum.cern.ch/

3.12 Geant4 10.6 Porting Guide (CentOS 7.6)

3.12.1 Introduction

Geant4 is a platform for the simulation of the passage of particles through matter using Monte Carlo methods. It is used in high energy physics and nuclear experiments, medical, accelerator and space physics studies.

For more information about Geant4, visit the **official Geant4 website**.

Programming language: C++

Brief description: Monte Carlo application package

 Copyright © Huawei Technologies Co., Ltd.

Open source license: GPL 3.0

Recommended Version

Geant4 10.6

3.12.2 Environment Requirements

Hardware Requirements

Table 3-58 lists the hardware requirements.

Table 3-58 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-59 lists the software requirements.

Table 3-59 Software requirements

Item	Version	How to Obtain
Geant4	10.6	**http://geant4.web.cern.ch/support/download**
Data files	10.6	**http://geant4.web.cern.ch/support/download**
CMake	3.17	**https://cmake.org/download/**
Test computing instance	exMPI01	Provided by the software.

OS Requirements

Table 3-60 lists the OS requirements.

Table 3-60 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

3.12.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Geant4 software porting.

Table 3-61 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/GEANT4	Installation path of Geant4	
4	/path/to/DATA	Installation path of data file packages	

3.12.4 Configuring the Compilation Environment

3.12.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-62 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **3.12.4.2 Installing CMake**.

 Copyright © Huawei Technologies Co., Ltd.

3.12.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Decompress the CMake installation package.

```
tar -xzvf cmake-3.17.0-rc2.tar.gz
```

Step 3 Switch to the directory generated after the package is decompressed.

```
cd cmake-3.17.0-rc2
```

Step 4 Configure CMake.

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Compile and install CMake.

```
make -j16
make install
```

Step 6 Set CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.12.5 Obtaining the Source Code

Procedure

Step 1 Download the Geant4 source package **geant4.10.06.p01.tar.gz**.

http://geant4.web.cern.ch/support/download

Step 2 Download 12 data file packages.

http://geant4.web.cern.ch/support/download

Step 3 Use an SFTP tool to upload the Genet4 source package to the **/path/to/GEANT4** directory on the server and upload the 12 data file packages to the **/path/to/DATA** directory on the server.

----End

3.12.6 Compiling and Installing Geant4

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Configure OpenGL.

```
yum install -y mesa*
yum install -y freeglut*
yum install -y *GLEW*
yum install -y openssl-devel
```

Step 3 Switch to the **DATA** directory.

```
cd /path/to/DATA
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Create a script for decompression.

1. Create a script.
   ```
   vi run.sh
   ```
2. Press **i** to enter the insert mode and add the following content to decompress data packages in batches:
   ```
   #!/bin/bash
   for wav in ./*.tar.gz; do
       echo "Extracting wav from $wav"
       tar -xvf $wav
   done
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the script and exit.

Step 5 Add execution permissions for the script.

```
chmod +x run.sh
```

Step 6 Decompress the data files.

```
sh run.sh
```

Step 7 Set environment variables for data files.

1. Open **data_source**.
   ```
   vi data_source
   ```
2. Press **i** to enter the insert mode and add the following content to set environment variables:
   ```
   export G4SAIDXSDATA=/path/to/DATA/G4SAIDDATA2.0
   export G4LEDATA=/path/to/DATA/G4EMLOW7.9.1
   export G4LEVELGAMMADATA=/path/to/DATA/PhotonEvaporation5.5
   export G4INCLDATA=/path/to/DATA/G4INCL1.0
   export G4PARTICLEXSDATA=/path/to/DATA/G4PARTICLEXS2.1
   export G4PIIDATA=/path/to/DATA/G4PII1.3
   export G4RADIOACTIVEDATA=/path/to/DATA/RadioactiveDecay5.4
   export G4REALSURFACEDATA=/path/to/DATA/RealSurface2.1.1
   export G4ABLADATA=/path/to/DATA/G4ABLA3.1
   export G4NEUTRONHPDATA=/path/to/DATA/G4NDL4.6
   export G4ENSDFSTATEDATA=/path/to/DATA/G4ENSDFSTATE2.2
   export G4TENDLDATA=/path/to/DATA/G4TENDL1.3.2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to make the environment variables take effect:

```
source data_source
```

Step 9 Decompress the Geant4 installation package.

```
tar -xvf geant4.10.06.p01.tar.gz
```

Step 10 Create a **build** folder and switch to the folder.

```
mkdir geant4.10.06.p01/build
cd geant4.10.06.p01/build
```

Step 11 Compile Geant4.

```
CC=mpicc CXX=mpicxx CFLAGS='-O3 -finline-functions -march=armv8.2-a -flto' cmake -
DCMAKE_INSTALL_PREFIX=/path/to/GEANT4/geant4.10.06.p01-install -
DGEANT4_USE_OPENGL_X11=ON -DGEANT4_BUILD_MULTITHREADED=ON ../
```

Step 12 Run the following commands to install Geant4:

```
make -j16
make install
```

Step 13 Switch to the parallel module directory.

```
cd /path/to/GEANT4/geant4.10.06.p01/examples/extended/parallel/MPI/source
```

Step 14 Create a **build** directory and switch to the directory.

 Copyright © Huawei Technologies Co., Ltd.

```
mkdir build
cd build
```

Step 15 Compile the parallel module.

```
CC=mpicc CXX=mpicxx CFLAGS='-O3 -finline-functions -march=armv8.2-a -flto'  cmake -
DCMAKE_INSTALL_PREFIX=/path/to/GEANT4/geant4.10.06.p01-install/lib64  ../
```

Step 16 Install the parallel module.

```
make -j16
make install
```

----End

3.12.7 Running and Verifying Geant4

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the test computing instance is stored:

```
cd /path/to/GEANT4/geant4.10.06.p01/examples/extended/parallel/MPI/examples
```

Step 3 Copy all the content in the **exMPI01** folder to the **exMPI01_bak** folder:

```
cp -r exMPI01 exMPI01_bak
```

Step 4 Switch to the test directory.

```
cd exMPI01
mkdir build
cd build
```

Step 5 Run the following commands to perform compilation and installation:

```
CC=mpicc CXX=mpicxx CFLAGS='-O3 -finline-functions -march=armv8.2-a -flto' cmake -
DCMAKE_INSTALL_PREFIX=/path/to/GEANT4/geant4.10.06.p01/examples/extended/parallel/MPI/
examples/exMPI01-test -DGeant4_DIR=/path/to/GEANT4/geant4.10.06.p01-install/lib64/Geant4-10.6.1 -
DG4mpi_DIR=/path/to/GEANT4/geant4.10.06.p01-install/lib64/lib64/G4mpi-10.6.1 ../
make -j16
```

Step 6 Run the following command to perform the test:

```
mpirun --allow-run-as-root -np 96 -mca btl ^openib ./exMPI01 run.mac > geant4-128core.log 2>&1
```

In the **geant4-128core.log** file, check the value of **time** (in seconds). A smaller value indicates higher performance.

Figure 3-11 shows an example of the test result.

Figure 3-11 Result example

----End

 Copyright © Huawei Technologies Co., Ltd.

3.12.8 Troubleshooting

An Error Reported During the Compilation of the Geant4 Parallel Module

Symptom

When the geant4 parallel module was compiled, "G4MPImanager.hh:117:9: error: 'MPI' does not name a type; did you mean 'M_PI'?" is reported.

Possible Causes

By default, g++ binding is disabled for OpenMPI 4.0.1.

Procedure

Recompile OpenMPI 4.0.1 and add the **--enable-mpi-cxx** parameter when configuring OpenMPI. For details, see Step 5 in "Installing OpenMPI" in **Basic Environment Setup Guide**.

3.12.9 More Information

Official Geant4 website:

http://geant4-data.web.cern.ch

3.13 DL_POLY 1.10 Porting Guide (CentOS 7.6)

3.13.1 Introduction

DL_POLY is a general purpose classical molecular dynamics (MD) simulation software developed at Daresbury Laboratory by I.T. Todorov and W. Smith.

For details about DL_POLY, visit the **official DL_POLY website**.

Recommended Software Version

DL_POLY 1.10

3.13.2 Environment Requirements

Hardware Requirements

Table 3-63 lists the hardware requirements.

Table 3-63 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-64 lists the software requirements.

Table 3-64 Software requirements

Item	Version	Download Address
DL_POLY	V1.10	https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz
Test case	TEST41.tar.gz	https://gitlab.com/DL_POLY_Classic/tests/-/blob/master/TEST41.tar.gz

OS Requirements

Table 3-65 lists the OS requirements.

Table 3-65 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.13.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DL_POLY software porting.

Table 3-66 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/DL_POLY	Installation path of DL_POLY.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

3.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-67 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

3.13.5 Obtaining the Source Code

Procedure

Step 1 Download the DL_POLY installation package **dl_poly-RELEASE-1-10.tar.gz**.

https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz

Step 2 Use an SFTP tool to upload the DL_POLY installation package to the */path/to/DL_POLY* directory on the server.

----End

3.13.6 Compiling and Installing DL_POLY

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the DL_POLY installation package:

```
tar -xzf dl_poly-RELEASE-1-10.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd dl_poly-RELEASE-1-10/source
```

Step 4 Run the following command to copy the **Makefile** file:

```
cp ../build/MakePAR ./Makefile
```

Step 5 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vim Makefile
   ```
2. Press **i** to enter the insert mode and change the content in line 65 to the following:
   ```
   FC=mpif90 FFLAGS="-c -O3 -march=armv8.2-a -mtune=tsv110" \
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 6 Run the following command to perform compilation and installation:

```
make dlpoly
```

Step 7 Run the following command to set the DL_POLY environment variable:

```
export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:$PATH
```

Step 8 Run the following command to check whether DL_POLY is installed successfully:

```
which DLPOLY.X
```

If the following information is displayed, the installation is successful:

```
/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute/DLPOLY.X
```

----End

3.13.7 Running and Verifying DL_POLY

3.13.7.1 Running and Verifying DL_POLY on a Single Node

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the TEST41 test case package:

```
tar zxf TEST41.tar.gz
```

Step 3 Run the following command to go to the TEST41 test case directory:

```
cd TEST41/NHC
```

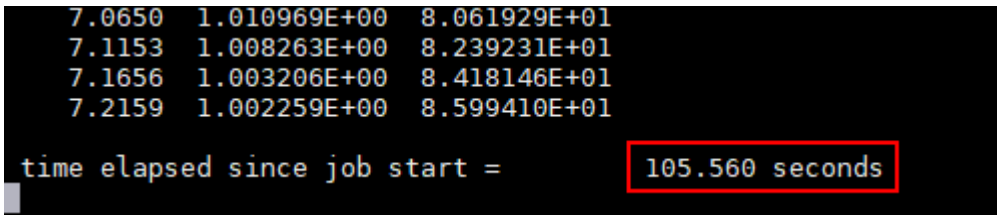
Step 4 Run the following command to run the test case in single-node mode:

```
mpirun --allow-run-as-root -np 128 -mca btl ^openib DLPOLY.X
```

time elapsed since job start = *XX* **seconds** in the **OUTPUT** file indicates the performance. A smaller value of *XX* indicates better performance.

Figure 3-12 is an example of the output result.

Figure 3-12 An example result



----End

3.13.7.2 Running and Verifying DL_POLY in Multi-node Mode

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the TEST41 test case package:

```
tar zxf TEST41.tar.gz
```

Step 3 Run the following command to go to the TEST41 test case directory:

```
cd TEST41/NHC
```

Step 4 Run the following command to create a **hostfile** file and add node information:

```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

NOTE

In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.

Step 5 On *node 2*, run the following command to configure environment variables:

```
vi /root/.bashrc
```

Step 6 Press **i** to enter the insert mode, and add the following content to the end of the **.bashrc** file:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export  LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:$PATH
```

Step 7 Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

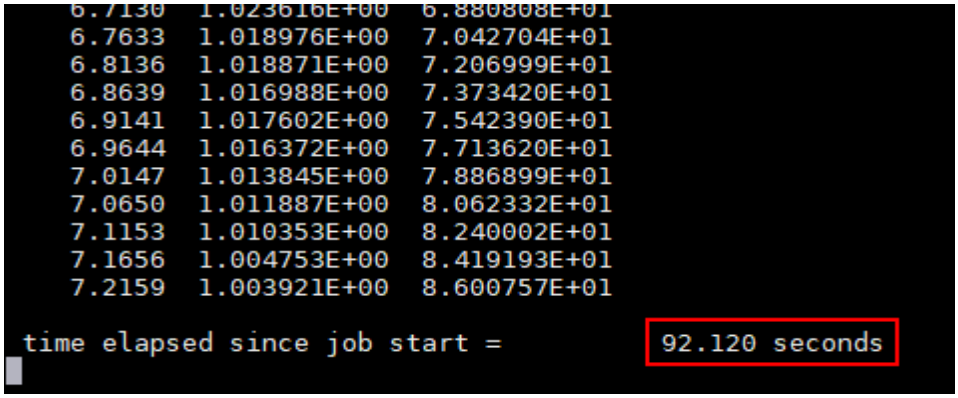
Step 8 Run the following command to run the test case in multi-node mode:

```
mpirun --allow-run-as-root -np 256 -N 128 -x OMP_NUM_THREADS=1 --hostfile /path/to/HOSTFILE -mca btl ^openib DLPOLY.X
```

time elapsed since job start = *XX* **seconds** in the **OUTPUT** file indicates the performance. A smaller value of *XX* indicates better performance.

Figure 3-13 is an example of the output result.

Figure 3-13 An example result



----End

3.13.8 More Information

For more information, visit the official DL_POLY website:

https://www.scd.stfc.ac.uk/Pages/DL_POLY.aspx

3.14 GAMESS Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

3.14.1 Introduction

The General Atomic and Molecular Electronic Structure System (GAMESS) is a program for *ab initio* quantum chemistry. GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wave functions include Configuration Interaction, second order perturbation Theory, and Coupled-Cluster approaches, as well as the Density Functional Theory approximation. Excited states can be computed by CI, EOM, or TD-DFT procedures. Nuclear gradients are available for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modeled by the discrete Effective Fragment potentials, or continuum models such as the Polarizable Continuum Model. Numerous relativistic computations are available, including infinite order two component scalar relativity corrections, with various spin-orbit coupling options. The Fragment Molecular Orbital method permits use of many of these sophisticated treatments on very large systems, by dividing the computation into small fragments. Nuclear wavefunctions can also be computed in VSCF or with explicit treatment of nuclear orbitals by the NEO code.

For more information, visit the **GAMESS official website**.

Programming language: C/Fortran 90

Brief description: GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF.

Open source license: GPL 3.0

Recommended Software Version

GAMESS does not have the version number. It is versioned based on the date of the public release. The version used in this document is R2 released on September 30, 2020.

3.14.2 Environment Requirements

Hardware Requirements

Table 3-68 lists the hardware requirements.

Table 3-68 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-69 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-69 Software requirements

Item	Version	Download Address
GAMESS	-	**https://www.msg.chem.iastate.edu/GAMESS/download/register**
HDF5	1.10.1	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz**
NetCDF-C	4.7.0	**https://www.unidata.ucar.edu/downloads/netcdf/**
NetCDF-Fortran	4.4.5	**https://www.unidata.ucar.edu/downloads/netcdf/**
Test case	runall	Provided by the software.

OS Requirements

Table 3-70 lists the OS requirements.

Table 3-70 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

3.14.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GAMESS software porting.

Table 3-71 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be
3	/path/to/NETCDF	Installation path of NetCDF-C	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
		Installation path of NetCDF-Fortran	replaced with the actual paths planned during the installation process.
4	/path/to/GAMESS	Installation path of GAMESS	

3.14.4 Configuring the Compilation Environment

3.14.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-72 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing HDF5	See **4.1**.
3	Installing NetCDF-C	See **3.14.4.3 Installing NetCDF-C**.
4	Installing NetCDF-Fortran	See **3.14.4.4 Installing NetCDF-Fortran**.

3.14.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -zxvf hdf5-1.10.1.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to create the HDF5 installation directory:

```
mkdir -p /path/to/HDF5
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following commands to set HDF5 environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

3.14.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L$HDF5/lib" CPPFLAGS="-I$HDF5/include" CC=mpicc --disable-dap
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

3.14.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

 Copyright © Huawei Technologies Co., Ltd.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$HDF5/include -I$NETCDF/include" LDFLAGS="-L$HDF5/lib -L$NETCDF/lib" CC=mpicc FC=mpif90 F77=mpif90
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End

3.14.5 Obtaining the Source Code

Academic and industrial users can obtain GAMESS site licenses free of charge. During the download, the system will ask you to agree to this license. This license recognizes Gordon Group's exclusive rights to GAMESS and also prohibits you from making copies of the GAMESS code for any purpose. You can only use the code in your own organization.

Procedure

Step 1 Register to apply for a license:

https://www.msg.chem.iastate.edu/GAMESS/download/register/

Step 2 After completing the registration, you will receive an email containing the download password. You can use any browser to download the compressed tar file or precompiled binary file of the source code.

----End

3.14.6 Compiling and Installing GAMESS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to check whether the required software is installed:

```
rpm -qa |grep gcc
gcc-4.8.5-36.el7.aarch64
gcc-gfortran-4.8.5-36.el7.aarch64
libgcc-4.8.5-36.el7.aarch64
gcc-c++-4.8.5-36.el7.aarch64
```

Step 3 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/GAMESS
```

Step 4 Run the following command to copy the installation package to the main program installation directory:

```
cp gamess-current.tar.gz /path/to/GAMESS
```

Step 5 Run the following command to go to the main program installation directory:

```
cd /path/to/GAMESS
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following commands to decompress the installation package and grant the root permission on the package:

```
tar -zxvf gamess-current.tar.gz
chown -R root:root gamess
```

Step 7 Run the following command to install the GAMESS main program:

1. Go to the GAMESS installation directory.
   ```
   cd /path/to/GAMESS/GAMESS
   ./config
   ```
2. Press **Enter** and enter **linux64**.
3. Press **Enter** three times and enter **00**.
4. Press **Enter** and enter **gfortran**.
5. Press **Enter** and enter the gfortran version number obtained in **Step 2**.
6. Press **Enter** twice and enter **none**.
7. Press **Enter** three times and enter **sockets**.
8. Press **Enter** and enter **no**.
9. Press **Enter** and enter **no**.
10. Press **Enter** and enter **no**.
11. Press **Enter**.

    The configuration file **install.info** is generated in */path/to/GAMESS*.

Step 8 Run the following commands to compile the GAMESS main program:

1. Compile DDI.
   ```
   cd ddi
   ./compddi
   mv ddikick.x ../
   cd ..
   ```
2. Compile GAMESS. After the command is executed for approximately 10 minutes, many configuration files are generated in the **object** directory.
   ```
   ./compall
   ```
3. Set the GAMESS link to generate **gamess.00.x**.
   ```
   ./lked gamess 00
   ```
4. Change the run environment.
   a. Create a **tmp** directory for storing temporary files.
      ```
      mkdir tmp
      ```
   b. Open the configuration file.
      ```
      vi rungms
      ```
   c. Press **i** to enter the insert mode and add the following content:
      ```
      set SCR=/path/to/GAMESS/tmp
      set USERSCR==/path/to/GAMESS/tmp
      set GMSPATH==/path/to/GAMESS
      ```
   d. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

3.14.7 Running and Verifying GAMESS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to check the host name:

```
hostname
```

Step 3 Run the following command to change the host name in the **runall** file:

1. Open **runall**.
   ```
   vi runall
   ```
2. Press **i** to enter the insert mode and change the value of **si.msg.chem.iastate.edu** in line 23 to **hostname**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to run the test case:

```
time ./runall 00
```

Step 5 Move the 47 log files in the */path/to/GAMESS* directory to the */path/to/GAMESS/tests/standard* directory.

Step 6 Run the following command to check whether the test case is executed successfully:

1. Switch to the test case directory.
   ```
   cd tests/standard
   ```
2. Modify the **checktst** file.
   a. Open **checktst**.
      vim checktst
   b. Press **i** to enter the insert mode and change **48** in line 39 to **47**.
   c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
3. Run the check command.
   ```
   time ./checktst
   ```
   If the following information is displayed, GAMESS is installed successfully:
   ```
   All 47 testresults are correct!
   ```

----End

3.14.8 More Resources

GAMESS official website:

https://www.msg.chem.iastate.edu/GAMESS/

3.15 ESPResSo 4.1.4 Porting Guide (CentOS 7.6)

3.15.1 Introduction

ESPResSo is a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models. It can be used to simulate systems such as polymers, liquid crystals, colloids, polyelectrolytes, ferrofluids and biological systems, for example DNA and lipid membranes.

For more information about ESPResSo, visit the **ESPResSo official website**.

Programming languages: C++/Python

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models.

Open source license: GPL 3.0

Recommended Software Version

ESPResSo 4.1.4

3.15.2 Environment Requirements

Hardware Requirements

Table 3-73 lists the hardware requirements.

Table 3-73 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-74 lists the software requirements.

Table 3-74 Software requirements

Item	Version	Download Address
ESPResSo	4.1.4	https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
Boost	1.72.0	https://github.com/boostorg/boost/archive/refs/tags/boost-1.72.0.tar.gz
CMake	3.8.1	https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz

OS Requirements

Table 3-75 lists the OS requirements.

Table 3-75 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

3.15.3 Paths for Software Porting

This chapter describes the software installation paths involved in the ESPResSo software porting.

Table 3-76 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Planned installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/ESPRESSOMD	Installation path of ESPResSo.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/FFTW	Installation path of FFTW.	
5	/path/to/CMAKE	Installation path of CMake.	
6	/path/to/BOOST	Installation path of Boost.	

3.15.4 Configuring the Compilation Environment

3.15.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-77 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing dependencies with Yum	See **3.15.4.2 Installing Dependencies with Yum**.
4	Installing FFTW	See **3.15.4.3 Installing FFTW**.
5	Installing CMake	See **3.15.4.4 Installing CMake**.
6	Installing Boost	See **3.15.4.5 Installing Boost**.

3.15.4.2 Installing Dependencies with Yum

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to configure the HUAWEI CLOUD mirror source:

```
wget --no-check-certificate -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.tools.huawei.com/repository/conf/CentOS-AltArch-7.repo
echo '[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.tools.huawei.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0' > epel.repo
```

Step 3 Run the following commands to configure the software source cache:

```
yum clean all
yum makecache
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to install Python 3, GSL, Git, NumPy, and Cython:

```
yum install -y python36 python3-devel gsl-devel git python36-numpy.aarch64  python36-Cython.aarch64
```

----End

3.15.4.3 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j 128
make install
```

Step 6 Run the following command to configure the FFTW environment variables required for installing the main program:

```
export CPATH=/path/to/FFTW/include:$CPATH
```

----End

3.15.4.4 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the CMake installation package:

```
tar -xvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd CMake-3.8.1
```

Step 4 Run the following commands to perform the compilation and installation:

```
./bootstrap --prefix=/path/to/CMAKE
make -j 16
make install
```

Step 5 Run the following command to configure CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.15.4.5 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Boost installation package:

```
tar -xvf boost_1_72_0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd boost_1_72_0
```

Step 4 Run the following command to perform the configuration:

```
./bootstrap.sh --with-toolset=gcc --with-libraries=all --with-python=/usr/bin/python3 --with-python-root=/usr --with-python-version=3.6 --prefix=/path/to/BOOST
```

Step 5 Run the following commands to modify the **project-config.jam** file:

1. Open the **project-config.jam** file.
   ```
   vi project-config.jam
   ```
2. Press **i** to enter the insert mode and modify the file.
   - Change the semicolon (;) in line 21 to a colon (:) and add two lines after line 21.

     Before the modification:

     ```
     21    using python : 3.6 : "/usr" ;
     ```

     After the modification:

     ```
     21    using python : 3.6 : "/usr" :
     22    "/usr/include/python3.6m" :
     23    "/usr/lib" ;
     ```
   - Add the following content below the last line of the file:
     ```
     using mpi ;
     ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform the compilation and installation:

```
./b2 -j32
./b2 install -j32
```

Step 7 Run the following commands to create a **BOOST_ROOT** variable and set the environment variable:

```
export BOOST_ROOT=/path/to/BOOST
export LD_LIBRARY_PATH=/path/to/BOOST/lib:$LD_LIBRARY_PATH
```

----End

3.15.5 Obtaining the Source Code

Procedure

Step 1 Download the ESPResSo installation package **espresso-4.1.4.tar.gz**.

https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz

Step 2 Use an SFTP tool to upload the ESPResSo installation package to the */path/to/ESPRESSOMD* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.15.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/ESPRESSOMD
```

Step 3 Run the following command to decompress the installation package:

```
tar -xvf espresso-4.1.4.tar.gz
```

Step 4 Run the following command to go to the directory generated after decompression:

```
cd espresso
```

Step 5 Run the following commands to modify the **cmake/FindFFTW3.cmake** file:

1. Open the **cmake/FindFFTW3.cmake** file.
   ```
   vi cmake/FindFFTW3.cmake
   ```
2. Press **i** to go to the edit mode. Add the following two lines before line 25:
   ```
   SET(CMAKE_INCLUDE_PATH ${CMAKE_INCLUDE_PATH} "/path/to/FFTW/include")
   SET(CMAKE_LIBRARY_PATH ${CMAKE_LIBRARY_PATH} "/path/to/FFTW/lib")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Create a **build** directory and build ESPResSo in the directory.

```
mkdir build
cd build
```

Step 7 Run the following commands to build the software:

```
cmake ..
```

Step 8 Run the following commands to modify the **CMakeCache.txt** file:

1. Open the **CMakeCache.txt** file.
   ```
   vi CMakeCache.txt
   ```
2. Press **i** to go to the edit mode and add compilation options to line 37 of the **CMakeCache.txt** file.
   ```
   CMAKE_CXX_FLAGS:STRING=-O3 -march=armv8.2-a -mcpu=tsv110 -Wl,-rpath=/path/to/KPGCC/
   gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/
   lib64/libhpc -lmathlib -lm
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following command to configure the environment variables of the **mathlib** library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:
$LD_LIBRARY_PATH
```

Step 10 Run the following commands to perform the compilation and installation:

```
make
```

NOTE

After the compilation is successful, the **EspressoCore.so** dynamic library is generated in the */path/to/ESPRESSOMD*/**espresso/build/src/core** directory. You can run the **ldd** command to list the dependencies of the dynamic library.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.15.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to run the test case contained in the source package:

```
{ time -p mpirun --allow-run-as-root -np 128 -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ./pypresso ../samples/constraints.py > run.log ;} 2>>run.log
```

The following is an example output of the log file:

```
(73.85, 103.92, 20.80)
(70.08, 105.04, 25.90)
(67.86, 112.22, 28.00)
(68.77, 113.43, 25.38)
(69.88, 116.07, 21.36)
real 228.98
user 29172.59
sys 55.19
```

----End

3.16 MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)

3.16.1 Introduction

Multiphysics Object Oriented Simulation Environment (MOOSE) is an object-oriented finite element framework for the development of tightly coupled multiphysics solvers. MOOSE makes use of the PETSc non-linear solver package and libMesh to provide the finite element discretization.

MOOSE features high parallelism (greater than 100,000 cores), complete multiphysics, and the use of both continuous and discrete Galerkin methods.

For more information about MOOSE, visit the **MOOSE official website**.

Programming language: C/C++

Brief description: a piece of object-oriented finite element multiphysics environment simulation software.

Open source license: GPL 2.0

Recommended Software Version

MOOSE 1.0.0

 Copyright © Huawei Technologies Co., Ltd.

3.16.2 Environment Requirements

Hardware Requirements

Table 3-78 lists the hardware requirements.

Table 3-78 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-79 lists the software requirements.

Table 3-79 Software requirements

Item	Version	Download Address
MOOSE	1.0.0	https://github.com/idaholab/moose/releases/tag/v1.0.0
CMake	3.8.1	https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz
Python	3.8.2	https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz
Test cases	hmg	Contained in the software.

OS Requirements

Table 3-80 lists the OS requirements.

Table 3-80 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.16.3 Paths for Software Porting

This chapter describes the software installation paths involved in the MOOSE software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-81 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/PYTHON	Installation path of Python.	
4	/path/to/MOOSE	Installation path of MOOSE.	

3.16.4 Configuring the Compilation Environment

3.16.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-82 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
4	Installing CMake	See **3.16.4.2 Installing CMake**.
5	Installing Python	See **3.16.4.3 Installing Python**.

3.16.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the CMake installation package:

```
cd /path/to/CMAKE
tar -xvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd cmake-3.8.1
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j16
make install
```

Step 6 Run the following command to load the environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.16.4.3 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the source package to the **/path/to/PYTHON** directory.

Step 3 Run the following commands to decompress the Python installation package:

```
cd /path/to/PYTHON
tar -xvf Python-3.8.2.tgz
```

Step 4 Run the following command to go to the directory generated after decompression:

```
cd Python-3.8.2
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/PYTHON
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Run the following command to back up Python:

```
mv /usr/bin/python /usr/bin/python-bak
```

Step 8 Run the following command to create a soft link for Python:

```
ln -s /path/to/PYTHON/bin/python3.8 /usr/bin/python
```

----End

3.16.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to configure Git:

```
git config --global http."sslVerify" false
```

Step 3 Run the following command to download the MOOSE source code:

```
git clone https://github.com/idaholab/moose.git
```

----End

3.16.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to configure the environment variables:

```
export CC=mpicc
export CXX=mpicxx
export F90=mpif90
export F77=mpif77
export FC=mpif90
```

Step 3 Run the following command to go to the **scripts** directory of MOOSE:

```
cd /path/to/MOOSE/moose/scripts
```

Step 4 Run the following command to install petsc:

```
./update_and_rebuild_petsc.sh
```

Step 5 Run the following command to install libmesh:

```
./update_and_rebuild_libmesh.sh
```

Step 6 Run the following command to go to the **test** directory:

```
cd /path/to/MOOSE/moose/test
```

Step 7 Run the following command to perform the compilation:

```
make -j 16
```

----End

3.16.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to go to the test directory:

```
cd /path/to/MOOSE/moose/test/tests/preconditioners/hmg
```

Step 3 Run the following command to perform the test:

```
{time -p mpirun --allow-run-as-root -n 128 /path/to/MOOSE/moose/test/moose_test-opt -i diffusion_hmg.i Mesh/dmg/dim=3 Mesh/dmg/nz=100 Outputs/file_base=diffusion_hmg_3d_out -log_view --error --error-unused --error-override --no-gdb-backtrace ;} > moose.log 2>&1
```

Step 4 Run the following command to restore Python:

```
mv /usr/bin/python-bak /usr/bin/python
```

----End

3.17 MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6)

3.17.1 Introduction

MPB is a free and open-source software package for computing the band structures, or dispersion relations, and electromagnetic modes of periodic dielectric structures, on both serial and parallel computers.

Meep is a free finite-difference time-domain (FDTD) simulation software package developed at MIT to model electromagnetic systems, along with the MPB eigenmode package.

For more information about MPB and Meep, visit the **MPB** and **Meep** official websites.

Programming languages: C++/Python

Brief description: a piece of finite-difference time-domain electromagnetic simulation software.

Open source license: GNU general public license

Recommended Software Version

MPB 1.11.1 and Meep 1.17.1

3.17.2 Environment Requirements

Hardware Requirements

Table 3-83 lists the hardware requirements.

Table 3-83 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-84 lists the software requirements.

Table 3-84 Software requirements

Item	Version	Download Address
Meep	1.17.1	https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
BLAS	3.8.0	http://www.netlib.org/blas/blas.tgz
LAPACK	3.9.0	https://codeload.github.com/Reference-LAPACK/lapack/tar.gz/v3.9.0
HARMINV	1.4.1	https://codeload.github.com/NanoComp/harminv/tar.gz/v1.4.1
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
libctl	4.5.0	https://codeload.github.com/NanoComp/libctl/tar.gz/v4.5.0
MPB	1.11.1	https://github.com/NanoComp/mpb/archive/v1.11.1.tar.gz
libGDSII	0.21	https://codeload.github.com/HomerReid/libGDSII/tar.gz/v0.21

OS Requirements

Table 3-85 lists the OS requirements.

Table 3-85 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.17.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Meep software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-86 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/SRC	Path of the source package.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/MEEP	Installation path of Meep.	
5	/path/to/DEPENDENCIES	Installation path of dependencies.	
6	/path/to/CASE	Path for storing test cases.	

3.17.4 Configuring the Compilation Environment

3.17.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-87 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing HDF5	See **3.17.4.2 Installing HDF5**.
4	Installing BLAS	See **3.17.4.3 Installing BLAS**.
5	Installing LAPACK	See **3.17.4.4 Installing LAPACK**.
6	Installing HARMINV	See **3.17.4.5 Installing HARMINV**.
7	Installing FFTW	See **3.17.4.6 Installing FFTW**.
8	Installing libctl	See **3.17.4.7 Installing libctl**.
9	Installing MPB	See **3.17.4.8 Installing MPB**.
10	Installing libGDSII	See **3.17.4.9 Installing libGDSII**.

3.17.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the HDF5 installation package:

```
cd /path/to/SRC
tar -xvf hdf5-1.10.1.tar
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform the configuration:

```
./configure --enable-parallel --enable-shared --prefix=/path/to/DEPENDENCIES CC=mpicc CXX=mpic++
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

Step 6 Run the following commands to configure environment variables:

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/DEPENDENCIES/bin:$PATH
export LD_LIBRARY_PATH=/path/to/DEPENDENCIES/lib:$LD_LIBRARY_PATH
```

----End

3.17.4.3 Installing BLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the BLAS installation package:

```
cd /path/to/SRC
tar -xvf blas.tgz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd BLAS-3.8.0
```

Step 4 Run the following commands to perform the compilation and installation:

```
`which gfortran` -c -O3 -fomit-frame-pointer -funroll-loops *.f
ar rv libblas.a *.o
```

Step 5 Run the following command to move the compiled static library to the planned directory:

```
cp libblas.a /path/to/DEPENDENCIES/lib/
```

----End

3.17.4.4 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the LAPACK installation package:

```
cd /path/to/SRC
tar -xvf lapack-3.9.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd lapack-3.9.0
```

Step 4 Run the following commands to perform the compilation and installation:

```
cp make.inc.example make.inc
make lapacklib -j96
```

Step 5 Run the following command to move the compiled static library to the planned directory:

```
cp liblapack.a /path/to/DEPENDENCIES/lib/
```

----End

3.17.4.5 Installing HARMINV

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following commands to decompress the HARMINV installation package:

```
cd /path/to/SRC
tar -xvf harminv-1.4.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd harminv-1.4.1
```

Step 4 Run the following command to perform the configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared LDFLAGS="-L/path/to/DEPENDENCIES/lib"
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

----End

3.17.4.6 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
cd /path/to/SRC
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to execute the **bootstrap.sh** script:

```
./bootstrap.sh
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/DEPENDENCIES --enable-shared --enable-static --enable-fma --enable-neon --enable-mpi
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

----End

3.17.4.7 Installing libctl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the libctl installation package:

```
cd /path/to/SRC
tar -xvf libctl-4.5.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd libctl-4.5.0
```

Step 4 Run the following command to install the system dependencies:

```
yum install -y guile-devel
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following command to perform the configuration:

```
sh autogen.sh --enable-shared --prefix=/path/to/DEPENDENCIES
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

----End

3.17.4.8 Installing MPB

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the MPB installation package:

```
cd /path/to/SRC
tar -xvf mpb-1.11.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd mpb-1.11.1
```

Step 4 Run the following command to install the system packages:

```
yum install -y guile-devel
```

Step 5 Run the following command to configure the environment variables of the **mathlib** library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:
$LD_LIBRARY_PATH
```

Step 6 Run the following command to perform the serial configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared CC=mpicc LDFLAGS="-L/path/to/
DEPENDENCIES/lib -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc/ -L/path/to/
KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm" CPPFLAGS="-I/path/to/
DEPENDENCIES/include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-hermitian-eps --
with-libctl=/path/to/DEPENDENCIES/share/libctl/
```

Step 7 Run the following commands to perform the serial compilation and installation:

```
make -j
make -j install
```

Step 8 (Optional) Run the following command to perform the parallel configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared CC=mpicc LDFLAGS="-L/path/to/
DEPENDENCIES/lib -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc/ -L/path/to/
KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm" CPPFLAGS="-I/path/to/
DEPENDENCIES/include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-hermitian-eps --
with-mpi --with-libctl=/path/to/DEPENDENCIES/share/libctl/
```

Step 9 (Optional) Run the following commands to perform the parallel compilation and installation:

```
make -j
make -j install
```

NOTE

Meep can only be linked to the serial version of MPB. For the linkage to the parallel version of MPB, you need to perform the compilation and installation.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.17.4.9 Installing libGDSII

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the libGDSII installation package:

```
cd /path/to/SRC
tar -xvf libGDSII-0.21.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd libGDSII-0.21
```

Step 4 Run the following command to perform the configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

----End

3.17.5 Obtaining the Source Code

Procedure

Step 1 Download the Meep installation package **meep-1.17.1.tar.gz**.

https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1

Step 2 Use an SFTP tool to upload the Meep installation package to the */path/to/SRC* directory on the server.

----End

3.17.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Meep installation package:

```
cd /path/to/SRC
tar -xvf meep-1.17.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd meep-1.17.1
```

Step 4 Run the following command to perform the configuration:

```
sh autogen.sh --prefix=/path/to/MEEP --enable-shared --with-mpi --with-openmp PYTHON=python3 MPICC=`which mpicc` MPICXX=`which mpic++` LDFLAGS="-L/path/to/DEPENDENCIES/lib -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc/ -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm" CPPFLAGS="-I/path/to/DEPENDENCIES/include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-libctl=/path/to/DEPENDENCIES/share/libctl/
```

Step 5 Run the following commands to compile and install Meep:

```
make -j
make -j install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following commands to view the generated executable file:

```
ll /path/to/MEEP/bin
-rwxr-xr-x 1 root root 1139520 Feb  3 11:43 meep
```

----End

3.17.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to configure the environment variables of the Meep executable file:

```
export PATH=/path/to/MEEP/bin:$PATH
```

Step 3 Run the following command to copy the test case file to the planned directory:

```
cp /path/to/SRC/meep-1.17.1/scheme/examples/mie-scattering.ctl /path/to/CASE
```

Step 4 Run the following command to go to the test case directory:

```
cd /path/to/CASE
```

Step 5 Run the following command to execute Meep:

```
mpirun --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 -np 128 meep mie-scattering.ctl |tee mie.out
```

Step 6 Run the following commands to extract the input data to a separate file:

```
grep flux1: mie.out |cut -d, -f2 -> input.dat
grep flux2: mie.out |cut -d, -f2 -> scatt.dat
```

After the commands are executed, check the value of **Elapsed run time** in the last line of the **mie.out** file. A smaller value indicates a higher performance.

----End

3.17.8 More Resources

MPB official website:

http://ab-initio.mit.edu/wiki/index.php/MIT_Photonic_Bands

Meep official website:

http://ab-initio.mit.edu/wiki/index.php/Meep

3.18 Octopus 10.3 Porting Guide (CentOS 7.6)

3.18.1 Introduction

Octopus is a pseudopotential real space package that simulates the electron-ion dynamics of one, two, and three-dimensional finite systems subject to time-dependent electromagnetic fields. The program is based on time-dependent density functional theory (TDDFT) in the Kohn-Sham scheme. Octopus can calculate static and dynamic polarizabilities and first hyperpolarizabilities, static

 Copyright © Huawei Technologies Co., Ltd.

magnetic susceptibilities, absorption spectra, and perform molecular dynamics simulations with Ehrenfest and Car–Parrinello methods.

For more information about Octopus, visit the **Octopus official website**.

Programming languages: C/C++

Brief description: a piece of quantum mechanics simulation software.

Recommended Software Version

Octopus 10.3

3.18.2 Environment Requirements

Hardware Requirements

Table 3-88 lists the hardware requirements.

Table 3-88 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-89 lists the software requirements.

Table 3-89 Software requirements

Item	Version	Download Address
Octopus	10.3	**https://octopus-code.org/down.php?file=10.3/octopus-10.3.tar.gz**
FFTW	3.3.8	**http://www.fftw.org/fftw-3.3.8.tar.gz**
GSL	2.6	**http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz**
Libxc	4.3.4	**http://forge.abinit.org/fallbacks/libxc-4.3.4.tar.gz**

OS Requirements

Table 3-90 lists the OS requirements.

Table 3-90 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

3.18.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Octopus software porting.

Table 3-91 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/FFTW	Installation path of FFTW.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/GSL	Installation path of GSL.	
5	/path/to/LIBXC	Installation path of Libxc.	
6	/path/to/OCTOPUS	Installation path of Octopus.	
7	/path/to/TESTCASE	Path for storing test cases.	

3.18.4 Configuring the Compilation Environment

3.18.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-92 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
2	Installing LAPACK	See **3.18.4.2 Installing Dependencies**.
3	Installing BLAS	See **3.18.4.2 Installing Dependencies**.
4	Installing FFTW	See **3.18.4.3 Installing FFTW**.
5	Installing GSL	See **3.18.4.4 Installing GSL**.
6	Installing Libxc	See **3.18.4.5 Installing Libxc**.

3.18.4.2 Installing Dependencies

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install lapack-devel blas-devel -y
```

----End

3.18.4.3 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/FFTW--enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to perform the compilation and installation:

```
make
make install
```

Step 6 Run the following commands to configure environment variables:

```
export LIBRARY_PATH=/path/to/FFTW/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.18.4.4 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GSL installation package:

```
tar -xvf gsl-2.6.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd gsl-2.6
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/GSL/gsl
```

Step 5 Run the following commands to perform the compilation:

```
make -j 32
make install
```

Step 6 Run the following commands to configure environment variables:

```
export LIBRARY_PATH=/path/to/GSL/gsl/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/GSL/gsl/lib:$LD_LIBRARY_PATH
```

----End

3.18.4.5 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Libxc installation package:

```
tar -xvf libxc-4.3.4.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd libxc-4.3.4
```

Step 4 Run the following command to perform the configuration:

```
./configure FC=mpifort CC=mpicc --prefix=/path/to/LIBXC/libxc434
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to perform the compilation:

```
make
make install
```

----End

3.18.5 Obtaining the Source Code

Procedure

Step 1 Download the Octopus installation package **octopus-10.3.tar.gz**.

https://octopus-code.org/down.php?file=10.3/octopus-10.3.tar.gz

Step 2 Use an SFTP tool to upload the Octopus installation package to the */path/to/OCTOPUS* directory on the server.

----End

3.18.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Octopus package:

```
tar -xvf octopus-10.3.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd octopus-10.3
```

Step 4 Run the following command to perform the configuration:

```
./configure CC=mpicc CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' FC=mpifort FCFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' LIBS='-Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm' --enable-mpi --prefix=/path/to/OCTOPUS --with-gsl-prefix=/path/to/GSL/gsl --with-libxc-prefix=/path/to/LIBXC/libxc434 --with-fftw-prefix=/path/to/FFTW
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j 32
make install
```

Step 6 Run the following command to configure the environment variables:

```
export PATH=/path/to/OCTOPUS/bin:$PATH
```

----End

3.18.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the test case directory:

```
cd /path/to/TESTCASE
```

Step 3 Run the following commands to create an input file named **inp** for calculating the ground state of the hydrogen atom:

 Copyright © Huawei Technologies Co., Ltd.

1. Open the **inp** file.
   ```
   vi inp
   ```
2. Press **i** to go to the insert mode and add the following content:
   ```
   CalculationMode = gs
   %Coordinates
   'H'|0|0|0
   %
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to execute the test case file:

```
mpirun --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -np 128 -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 octopus
```

Step 5 Run the following command to check the calculation result:

```
cat static/info
```

```
Eigenvalues [H]
 #st  Spin   Eigenvalue      Occupation
   1   --    -0.233015       1.000000

Energy [H]:
      Total       =        -0.44637971
      Free        =        -0.44637971
      -----------
      Ion-ion     =         0.00000000
      Eigenvalues =        -0.23301542
      Hartree     =         0.28415398
      Int[n*v_xc] =        -0.30429904
      Exchange    =        -0.19375648
      Correlation =        -0.03975286
      vanderWaals =         0.00000000
      Delta XC    =         0.00000000
      Entropy     =         1.38629436
      -TS         =        -0.00000000
      Photon ex.  =         0.00000000
      Kinetic     =         0.41780783
      External    =        -0.91483206
      Non-local   =         0.00000000
      Int[n*v_E]  =         0.00000000
```

----End

3.19 Psi4 1.3.2 Porting Guide (CentOS 7.6)

3.19.1 Introduction

Psi4 is an open source suite of ab initio quantum chemistry programs designed for efficient, high-accuracy simulations of molecular properties. It routinely performs computations with more than 2500 basis functions on multi-core machines.

With computationally demanding portions written in C++, exports of many C++ classes into Python via Pybind11, and a flexible Python driver, Psi4 strives to be friendly to both users and developers.

For more information about Psi4, visit the **Psi4 official website**.

Programming languages: C++/Python

 Copyright © Huawei Technologies Co., Ltd.

Brief description: an open-source quantum chemistry software package.

Open source license: GNU general public license

Recommended Software Version

Psi4 1.3.2

3.19.2 Environment Requirements

Hardware Requirements

Table 3-93 lists the hardware requirements.

Table 3-93 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-94 lists the software requirements.

Table 3-94 Software requirements

Item	Version	Download Address
Psi4	1.3.2	https://github.com/psi4/psi4/archive/v1.3.2.tar.gz
CMake	3.8.1	https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/archive/v0.3.6.tar.gz
gau2grid	1.3.0	https://github.com/dgasmith/gau2grid/archive/v1.3.0.tar.gz
libint	release-1-2-0	https://github.com/evaleev/libint/archive/release-1-2-0.tar.gz
Libxc	4.3.0	https://gitlab.com/libxc/libxc/-/archive/4.3.0/libxc-4.3.0.tar.gz
pybind11	2.6.2	https://github.com/pybind/pybind11/archive/v2.6.2.tar.gz

OS Requirements

Table 3-95 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-95 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

3.19.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Psi4 software porting.

Table 3-96 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/PSI4	Installation path of Psi4.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/CMAKE	Installation path of CMake.	
5	/path/to/OPENBLAS	Installation path of OpenBLAS.	
6	/path/to/GAU2GRID	Installation path of gau2grid.	
7	/path/to/LIBINT	Installation path of Libint.	
8	/path/to/LIBXC	Installation path of Libxc.	
9	/path/to/PYBIND11	Installation path of pybind11.	

3.19.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

3.19.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-97 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing system dependencies	See **3.19.4.2 Installing System Dependencies**.
4	Installing CMake	See **3.19.4.3 Installing CMake**.
5	Installing OpenBLAS	See **3.19.4.4 Installing OpenBLAS**.
6	Installing gau2grid	See **3.19.4.5 Installing gau2grid**.
7	Installing Libint	See **3.19.4.6 Installing Libint**.
8	Installing Libxc	See **3.19.4.7 Installing Libxc**.
9	Installing pybind11	See **3.19.4.8 Installing pybind11**.

 Copyright © Huawei Technologies Co., Ltd.

3.19.4.2 Installing System Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install the system dependencies:

```
yum install -y numpy python3 python3-devel python3-libs python3-pip
pip3 install QCElemental QCEngine deepdiff
```

----End

3.19.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the CMake installation package:

```
cd /path/to/CMAKE
tar -xvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd cmake-3.8.1
```

Step 4 Run the following command to perform the configuration:

```
mkdir install
./configure --prefix=/path/to/CMAKE/cmake-3.8.1/install
```

Step 5 Run the following command to perform the compilation and installation:

```
make -j
make install
```

Step 6 Run the following command to configure environment variables:

```
export PATH=/path/to/CMAKE/cmake-3.8.1/install/bin:$PATH
```

----End

3.19.4.4 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the OpenBLAS installation package:

```
cd /path/to/OPENBLAS
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd OpenBLAS-0.3.6
```

Step 4 Run the following commands to configure environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to modify the **Makefile.system** file:

 Copyright © Huawei Technologies Co., Ltd.

1. Open the **Makefile.system** file.
   ```
   vi Makefile.system
   ```
2. Type **:1161** to go to line 1161 and press **i** to go to the insert mode.

   Before the modification:
   ```
   COMMON_OPT = -O2
   ```
   After the modification:
   ```
   COMMON_OPT = -O2 -lgfortran
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform the compilation and installation:
```
mkdir install
make -j
make PREFIX=/path/to/OPENBLAS/OpenBLAS-0.3.6/install install
```

Step 7 Run the following commands to configure environment variables:
```
export MATH_ROOT=/path/to/OPENBLAS/OpenBLAS-0.3.6/install
```

----End

3.19.4.5 Installing gau2grid

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the gau2grid installation package:
```
cd /path/to/GAU2GRID
tar -xvf gau2grid-1.3.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:
```
cd gau2grid-1.3.0
```

Step 4 Run the following commands to perform the compilation:
```
mkdir /path/to/GAU2GRID/build
cmake -H. -B/path/to/GAU2GRID/build -DCMAKE_INSTALL_PREFIX=/path/to/GAU2GRID/install
```

Step 5 Run the following commands to perform the installation:
```
cd /path/to/GAU2GRID/build
make -j
make install
```

----End

3.19.4.6 Installing Libint

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Libint installation package:
```
cd /path/to/LIBINT
tar -xvf libint-release-1-2-0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:
```
cd libint-release-1-2-0
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following commands to perform the compilation:

```
mkdir /path/to/LIBINT/build
cmake -H. -B/path/to/LIBINT/build -DCMAKE_INSTALL_PREFIX=/path/to/LIBINT/install
```

Step 5 Run the following commands to perform the installation:

```
cd /path/to/LIBINT/build
make -j
make install
```

----End

3.19.4.7 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Libxc installation package:

```
cd /path/to/LIBXC
tar -xvf libxc-4.3.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd libxc-4.3.0
```

Step 4 Run the following commands to perform the compilation:

```
mkdir /path/to/LIBXC/build
cmake -H. -B/path/to/LIBXC/build -DCMAKE_INSTALL_PREFIX=/path/to/LIBXC/install
```

Step 5 Run the following commands to perform the installation:

```
cd /path/to/LIBXC/build
make -j
make install
```

----End

3.19.4.8 Installing pybind11

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the pybind11 installation package:

```
cd /path/to/PYBIND11
tar -xvf pybind11-2.6.2.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd pybind11-2.6.2
```

Step 4 Run the following commands to perform the compilation:

```
mkdir /path/to/PYBIND11/build
cmake -H. -B/path/to/PYBIND11/build -DCMAKE_INSTALL_PREFIX=/path/to/PYBIND11/install
```

Step 5 Run the following commands to perform the installation:

```
cd /path/to/PYBIND11/build
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.19.5 Obtaining the Source Code

Procedure

Step 1 Download the Psi4 installation package **psi4-1.3.2.tar.gz**.

https://github.com/psi4/psi4/archive/v1.3.2.tar.gz

Step 2 Use an SFTP tool to upload the Psi4 installation package to the */path/to/PSI4* directory on the server.

----End

3.19.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Psi4 installation package:

```
cd /path/to/PSI4
tar -xvf psi4-1.3.2.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd psi4-1.3.2
```

Step 4 Run the following commands to perform the compilation:

```
mkdir build install
cmake -H. -B/path/to/PSI4/build -DCMAKE_INSTALL_PREFIX=/path/to/PSI4/install -Dpybind11_DIR=/
path/to/PYBIND11/install/share/cmake/pybind11 -Dgau2grid_DIR=/path/to/GAU2GRID/install/share/
cmake/gau2grid -DLibint_DIR=/path/to/LIBINT/install/share/cmake/Libint -DLibxc_DIR=/path/to/LIBXC/
install/share/cmake/Libxc
```

Step 5 Run the following commands to perform the installation:

```
cd /path/to/PSI4/build
make -j
make install
```

Step 6 Run the following commands to view the generated executable file:

```
ll /path/to/PSI4/install/bin
-rwxr-xr-x 1 root root 12139 Mar  9 10:47 psi4
```

----End

3.19.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to configure the environment variables of the Psi4 executable file:

```
export PATH=/path/to/PSI4/install/bin:$PATH
```

Step 3 Run the following command to go to the test case directory:

```
cd /path/to/PSI4/install/share/psi4/samples/opt-multi-frozen-dimer-c2h
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to run Psi4:

```
psi4 input.dat
```

After the running is complete, check the value of **Wall Time** in the **timer.dat** file. A smaller value indicates a higher performance.

----End

3.20 SIESTA 4.0.2 Porting Guide (CentOS 7.6)

3.20.1 Introduction

SIESTA is both a method and its computer program implementation, to perform efficient electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids.

For more information about SIESTA, visit the **SIESTA official website**.

Programming languages: C/C++

Brief description: a piece of material calculation simulation software.

Recommended Software Version

SIESTA 4.0.2

3.20.2 Environment Requirements

Hardware Requirements

Table 3-98 lists the hardware requirements.

Table 3-98 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-99 lists the software requirements.

Table 3-99 Software requirements

Item	Version	Download Address
SIESTA	4.0.2	**https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz**
ScaLAPACK	2.0.2	**http://www.netlib.org/scalapack/scalapack-2.0.2.tgz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 3-100 lists the OS requirements.

Table 3-100 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.20.3 Paths for Software Porting

This chapter describes the software installation paths involved in the SIESTA software porting.

Table 3-101 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/SIESTA	Installation path of SIESTA.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
4	/path/to/SCALAPACK	Installation path of ScaLAPACK.	
5	/path/to/TESTCASE	Path for storing test cases.	

3.20.4 Configuring the Compilation Environment

3.20.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-102 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.
2	Installing LAPACK	See **3.20.4.2 Installing Dependencies**.
3	Installing BLAS	See **3.20.4.2 Installing Dependencies**.
4	Installing ScaLAPACK	See **3.20.4.3 Installing ScaLAPACK**.

3.20.4.2 Installing Dependencies

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install lapack-devel blas-devel -y
```

----End

3.20.4.3 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar -xvf scalapack-2.0.2.tgz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd scalapack-2.0.2
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to generate an **SLmake.inc** file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following command to perform the compilation and installation:

```
make
```

Step 6 Run the following commands to copy the generated link library to the installation directory:

```
mkdir -p /path/to/SCALAPACK
cp libscalapack.a /path/to/SCALAPACK
```

----End

3.20.5 Obtaining the Source Code

Procedure

Step 1 Download the SIESTA installation package **siesta-4.0.2.tar.gz**.

https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz

Step 2 Use an SFTP tool to upload the SIESTA installation package to the */path/to/SIESTA* directory on the server.

----End

3.20.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the SIESTA package:

```
tar -xvf siesta-4.0.2.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd siesta-4.0.2
```

Step 4 Run the following commands to create a directory:

```
mkdir kpgcc_hmpi_compiler
```

Step 5 Run the following commands to go to the directory and run the script:

```
cd kpgcc_hmpi_compiler
bash ../Src/obj_setup.sh
```

Step 6 Run the following command to perform the configuration:

```
../Src/configure CC=mpicc CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' FC=mpifort FCFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' --enable-mpi --build=arm-linux --prefix=/path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler
```

Step 7 Run the following commands to modify the **arch.make** file:

1. Open the **arch.make** file.
   ```
   vi arch.make
   ```
2. Press **i** to enter the insert mode and modify lines 38, 39, and 45 of the file.
   ```
   SCALAPACK_LIBS=/path/to/SCALAPACK/libscalapack.a
   MATH_LIBS= -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
LIBS=$(SCALAPACK_LIBS) $(BLACS_LIBS) $(LAPACK_LIBS) $(BLAS_LIBS) $(NETCDF_LIBS) $
(MATH_LIBS)
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to perform the compilation and installation:

```
make
```

Step 9 Run the following command to configure the environment variables:

```
export PATH=/path/to/SIESTA/:$PATH
```

----End

3.20.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to copy the **Tests** directory of the source package to the test case directory:

```
cd /path/to/SIESTA
cp -r ./siesta-4.0.2/Tests /path/to/TESTCASE
```

Step 3 Run the following command to go to the test case directory:

```
cd /path/to/TESTCASE/Tests/born
```

Step 4 Run the following command to execute the test case file:

```
make SIESTA="mpirun --allow-run-as-root -np 128 siesta"
```

If the **born.out** and **work** files are generated, the execution is successful.

```
[root@2280node23-test:/Oceanfile/RR3/kunpeng/application/siesta/kpgcc_hmpi_siesta/sie
born.fdf  born.out  born.pseudos  born.xml  completed_work  FC.fdf  makefile  work
```

----End

3.20.8 Troubleshooting

Problem 1: An error is reported when performing step 7 of SIESTA compilation and installation

Symptom:

During the SIESTA compilation and installation, an error is reported in step 7:

```
configure: error: cannot guess build type; you must specify one
make: *** [FoX/.config] Error 1
```

Possible Causes:

```
checking build system type... /Oceanfile/RR3/kunpeng/application/siesta/siesta-4.0.2/Src/FoX/config/
config.guess: unable to guess system type
This script, last modified 2006-12-08, has failed to recognize
the operating system you are using. It is advised that you
download the most up to date version of the config scripts from
http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess
and
http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub
```

Procedure:

 Copyright © Huawei Technologies Co., Ltd.

1. Perform the following operations:
   - http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess
   - http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub
2. Replace the **config.guess** and **config.sub** files in the **/path/to/SIESTA/siesta-4.0.2/SRC/Fox/config** directory. Run the **make clean** command and run the **make** command for compilation again.

3.21 Wannier90 3.1.0 Porting Guide (CentOS 7.6)

3.21.1 Introduction

Wannier90 is a computer package written in Fortran90. It is used to obtain maximally-localised Wannier functions in order to calculate bandstructures, Fermi surfaces, dielectric properties, sparse Hamiltonians and many things besides.

Many electronic structure codes have an interface to Wannier90, including Quantum ESPRESSO, Abinit, VASP, Siesta, Wien2k, Fleur, OpenMX, and GPAW. There are several post-processing codes that are able to use the output of Wannier90 for further analysis and calculation.

Programming language: Fortran

Brief description: Wannier90 is a computer package, written in Fortran90, used to obtain maximally-localised Wannier functions.

Open source license: GPL 2.0

Recommended Software Version

Wannier90 3.1.0

3.21.2 Environment Requirements

Hardware Requirements

Table 3-103 lists the hardware requirements.

Table 3-103 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-104 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-104 Software requirements

Item	Version	Download Address
Wannier90	3.1.0	**https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0**

OS Requirements

Table 3-105 lists the OS requirements.

Table 3-105 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

3.21.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Wannier90 software porting.

Table 3-106 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/WANNIER90	Installation path of Wannier90.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.21.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

3.21.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-107 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.

3.21.4.2 Installing the Dependency Libraries and Tools Using Yum

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the LAPACK and BLAS libraries as well as the gnuplot interactive graphing tool.

```
yum install lapack* blas* gnuplot -y
```

NOTE

- The LAPACK and BLAS libraries are installed in **/usr/lib64** by default.
- The gnuplot is installed in **/usr** by default.

Step 3 Run the following commands to configure gnuplot environment variables:

```
export GNUTERM=dumb
export GNUPLOT=/usr
export PATH=/usr/bin:$PATH
export MANPATH=/usr/share/man/man1:$MANPATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.21.5 Obtaining the Source Code

Procedure

Step 1 Download the Wannier90 installation package **wannier90-3.1.0.tar.gz**.

https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0

Step 2 Use an SFTP tool to upload the Wannier90 installation package to the */path/to/WANNIER90* directory on the server.

----End

3.21.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/WANNIER90
```

Step 3 Run the following command to decompress the main program installation package:

```
tar -xvf wannier90-3.1.0.tar.gz
```

Step 4 Run the following command to go to the directory generated after decompression:

```
cd wannier90-3.1.0
```

Step 5 Run the following command to perform build Wannier90:

```
cp ./config/make.inc.gfort ./make.inc
```

Step 6 Run the following commands to modify the **make.inc** file:

1. Open the **make.inc** file.
   ```
   vi make.inc
   ```
2. Press **i** to go to the insert mode.
   - Uncomment lines 7 and 8, and modify line 8 as follows:
     ```
     MPIF90 = mpif90
     ```
   - Modify line 10 as follows:
     ```
     FCOPTS = -O3 -march=armv8.2-a -mcpu=tsv110
     ```
   - Uncomment line 21 and modify it as follows:
     ```
     LIBDIR = /usr/lib64
     ```
   - Modify line 37 as follows:
     ```
     LIBS = -L$(LIBDIR) -llapack -lblas -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm
     ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to configure the environment variables of the **mathlib** library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:$LD_LIBRARY_PATH
```

Step 8 Run the following command to compile and install Wannie90:

```
make -j 16
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

After the compilation and installation are complete, two executable files **postw90.x** and **wannier90.x** are generated.

----End

3.21.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

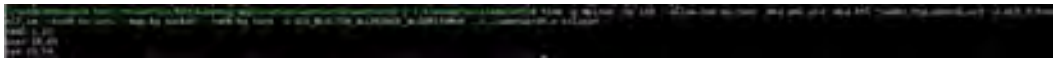
Step 2 Run the following command to go to the test directory:

```
cd /path/to/WANNIER90/wannier90-3.1.0/examples/example03
```

Step 3 Run the following command to perform the test:

```
time -p mpirun -np 128 --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x
UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x silicon
```

After the command is successfully executed, a **silicon.wout** file is generated. The following figure shows an example of the command output.



Step 4 Run the following commands to modify the **silicon.win** file:

1. Open the **silicon.win** file.
   ```
   vi silicon.win
   ```
2. Press **i** to go to the insert mode. Add the following two lines to the end of line 2:
   ```
   restart          = plot
   bands_plot       = true
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

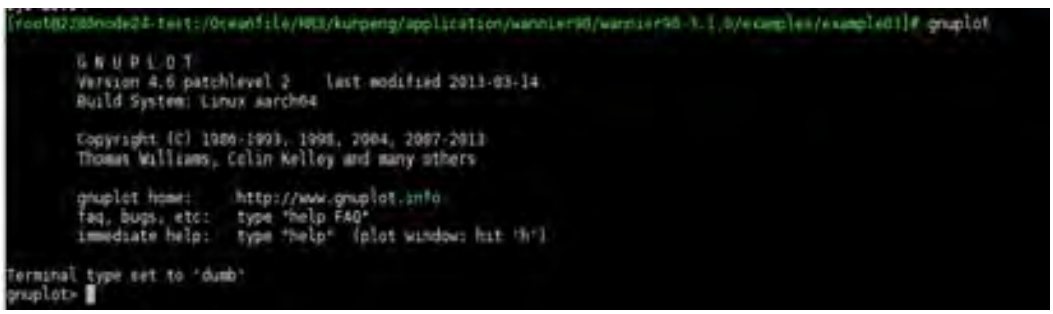
Step 5 Run the test command of **Step 3** again. After the command is successfully executed, the **silicon_band.gnu** and **silicon_band.dat** files are generated in the current directory.

```
time -p mpirun -np 128 --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x
UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x silicon
```

Step 6 Run the following command to use the gnuplot interactive graphing tool:

```
gnuplot
```

The following information is displayed:



```
load '/path/to/WANNIER90/wannier90-3.1.0/examples/example03/silicon_band.gnu'
```

The following information is displayed:

----End

3.21.8 More Resources

Wannier90 official website:

http://www.wannier.org

3.22 WanT 2.6.1 Porting Guide (CentOS 7.6)

3.22.1 Introduction

WanT simulates quantum transport properties (conductance) of nanojunctions using a real space Wannier function description of the Hamiltonian of the system. The core methodology combines state-of-the-art Density Functional Theory (DFT), plane-wave, norm-conserving pseudopotentials calculations with a Green's functions method based on the Landauer formalism to describe quantum conductance. A crucial step in the calculation is the use of the maximally-localized Wannier function representation to introduce naturally the ground-state electronic structure into the lattice Green's function approach. The WanT package operates, in principles, as a simple post-processing of any standard electronic structure code. The WanT code is currently interfaced to the codes in the Quantum-ESPRESSO distribution, Abinit, and CRYSTAL09. Interfaces with CP2K and GAMESS are under testing in the development version of the code.

Programming language: Fortran

Brief description: The WanT package operates, in principles, as a simple post-processing of any standard electronic structure code.

Open source license: GPL 2.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

WanT 2.6.1

3.22.2 Environment Requirements

Hardware Requirements

Table 3-108 lists the hardware requirements.

Table 3-108 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-109 lists the software requirements.

Table 3-109 Software requirements

Item	Version	Download Address
WanT	2.6.1	https://github.com/QEF/want/archive/want-2.6.1.tar.gz
QE	4.6.1	https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz

OS Requirements

Table 3-110 lists the OS requirements.

Table 3-110 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.22.3 Paths for Software Porting

This chapter describes the software installation paths involved in the WanT software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-111 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/WANT	Installation path of WanT.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/QE	Installation path of QE.	

3.22.4 Configuring the Compilation Environment

3.22.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-112 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Huawei Kunpeng compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Setting up the Huawei MPI environment	See **Hyper MPI 1.1.0 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

3.22.4.2 Installing QE

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the LAPACK and BLAS libraries:

```
yum install lapack* blas* -y
```

Step 3 Run the following commands to decompress the QE installation package:

```
cd /path/to/QE
tar -xvf q-e-qe-6.4.1.tar.gz
```

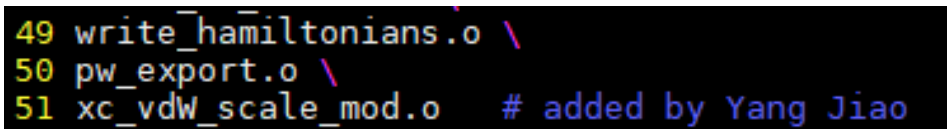
Step 4 Run the following command to go to the directory generated after decompression:

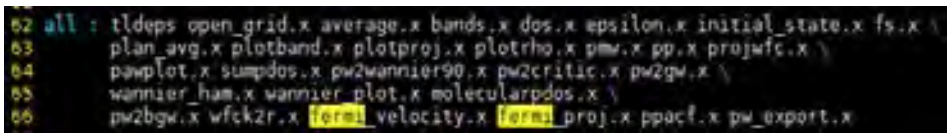
```
cd q-e-qe-6.4.1
```

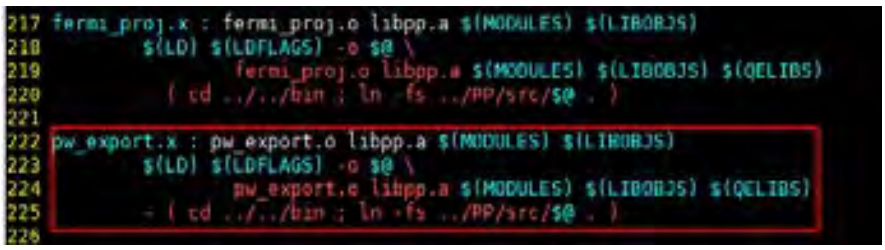
Step 5 Run the following command to perform the configuration:

```
./configure F90=gfortran F77=gfortran MPIF90=mpifort MPIF77=mpifort CC=mpicc FCFLAGS="-O3" CFLAGS="-O3" --with-scalapack=yes --build=aarch64-unknown-linux-gnu --prefix=/path/to/QE
```

Step 6 Run the following command to modify the **PP/src/Makefile** file:

1. Open the **PP/src/Makefile** file.
   ```
   vi PP/src/Makefile
   ```
2. Press **i** to go to the insert mode.
   - Add the following line below line 49:
     ```
     pw_export.o \
     ```
     

   - Add the following content to the end of line 66:
     ```
     pw_export.x
     ```
     

   - Add four lines below line 221:
     i. In common mode, move the cursor to line 217 and press **4yy** to copy lines 217 to 220.
     ii. Move the cursor to line 222 and press **p** to paste.
     iii. Modify the content of lines 222 and 224, as shown in the following figure:
     

 Copyright © Huawei Technologies Co., Ltd.

NOTE

If you add lines 222 to 225 in insert mode, the font color of part of the content does not change. That is, the modification does not take effect and the executable file of **pw_export.x** fails to be compiled. Therefore, copy the first four lines and keep them consistent before modifying part of the content.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following commands to perform the compilation and installation.

```
make -j 16 pwall
make install
```

----End

3.22.5 Obtaining the Source Code

Procedure

Step 1 Download the **want-want-2.6.1.tar.gz** installation package.

Download address: **https://github.com/QEF/want/archive/want-2.6.1.tar.gz**

Step 2 Use an SFTP tool to upload the WanT installation package to the */path/to/WANT* directory on the server.

----End

3.22.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/WANT
```

Step 3 Run the following command to decompress the main program installation package:

```
tar -xvf want-want-2.6.1.tar.gz
```

Step 4 Run the following command to go to the directory generated after decompression:

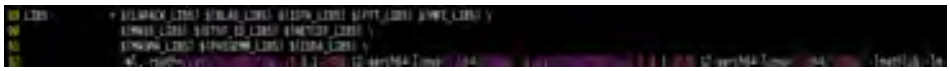
```
cd want-want-2.6.1
```

Step 5 Run the following command to perform the configuration:

```
./configure CC=gcc CFLAGS="-O3 -march=armv8.2-a -mcpu=tsv110" --build=aarch64-unknown-linux-gnu
```

Step 6 Run the following commands to modify the **make.sys** file and link the **mathlib** math library of the optimized GCC for openEuler version to the main program.

1. Open the **make.sys** file.
   ```
   vi make.sys
   ```
2. Press **i** to go to the insert mode.
   - Add a newline character (\) at the end of line 91.
   - Add line 92 and add the following content:
     ```
     -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm
     ```



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to configure the environment variables of the **mathlib** library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:$LD_LIBRARY_PATH
```

Step 8 Run the following command to perform the compilation and installation:

```
make all
```

After the compilation and installation are complete, the following executable files are generated in the **bin** directory:

----End

3.22.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the **tests** directory:

```
cd /path/to/WANT/want-want-2.6.1/tests
```

Step 3 Run the following commands to modify the **environment.conf** file:

1. Open the **environment.conf** file.
   ```
   vi environment.conf
   ```
2. Press **i** to go to the insert mode.
   - Modify line 28 as follows:
     ```
     QE_BIN=/path/to/QE/q-e-qe-6.4.1/bin
     ```
   - Modify line 32 as follows:
     ```
     TMPDIR=/path/to/WANT/want-want-2.6.1
     ```
   - Modify line 34 as follows:
     ```
     PARA_PREFIX="time -p mpirun -np 121 --allow-run-as-root --mca coll ^hcoll -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8"
     ```
   - Modify line 35 as follows:
     ```
     OMP_NUM_THREADS=1
     ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to go to the **test01** directory:

```
cd test01
```

Step 5 Run the following commands to modify the **scf.in**, **nscf.in**, and **want.in** files:

1. Open the **scf.in**, **nscf.in**, and **want.in** files.
   ```
   vi Input file
   ```
2. Press **i** to go to the insert mode.
   a. Add **diagonalization = 'cg'** to the **&electrons** part of **scf.in** and **nscf.in**.

```
&electrons
   mixing_mode = 'plain'
   mixing_beta = 0.7
   conv_thr =  1.0d-8
   diagonalization  = 'cg'
```

b. Add **nbnd=16** to the **nscf.in** file.

```
&system
   ibrav=  2, celldm(1) =10.599478, nat=  2, ntyp= 1,
   ecutwfc =30.0, nbnd = 16
   nr1 = 32
   nr2 = 32
   nr3 = 32
```

c. In the **want.in** file, comment out **dftdata_fmt**, that is, delete the exclamation mark (!) before **dftdata_fmt**.

```
&CONTROL
    prefix      = 'silicon'
    postfix     = '_WanT'
    work_dir    = './SCRATCH'
    dftdata_fmt = 'pw_export'
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to execute the test:

```
./run.sh all
```

If the test is executed successfully, the following information is displayed:

```
running SCF calculation... real 2.27
user 140.05
sys 51.27
 done
running NSCF calculation... real 68.02
user 7822.20
sys 319.17
 done
running EXPORT calculation... done
running DISENTANGLE calculation... real 1.89
user 139.30
sys 13.75
 done
running WANNIER calculation... real 95.18
user 11426.93
sys 16.13
 done
running BANDS calculation... real 1.06
user 48.02
sys 11.81
 done
```

----End

3.22.8 More Resources

WanT official website:

https://github.com/QEF/want

 Copyright © Huawei Technologies Co., Ltd.

3.23 GROMACS 2019.3 Porting Guide (CentOS 8.2)

3.23.1 Introduction

GROningen MAchine for Chemical Simulations (GROMACS) is a computational engine for MD simulation and energy minimization, which simulates hundreds to millions of atomic systems using Newton's equilibrium equations. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a large number of complicated bonded interactions, but since it is extremely fast at calculating the nonbonded interactions (that usually dominate simulations), many groups are also using it for research on non-biological systems, such as polymers. Compared with other MD simulation software, GROMACS has the following unique advantages:

- GROMACS is free of charge and complies with the GNU Lesser General Public License (LGPL) protocol. You can find the open-source code of GROMACS on GitHub.
- GROMACS provides higher performance than other software and has been optimized in code.
- GROMACS is user-friendly for topology files and parameter setting files. The format of GROMACS is similar to that of Python.
- The environment of GROMACS develops well, and many simulation tools support GROMACS well.

For more information about GROMACS, visit the **GROMACS official website**.

Programming language: C++

Brief description: GROMACS is a computational engine for MD simulation and energy minimization.

Open source license: LGPL Version 2.1

Recommended Software Version

GROMACS 2019.3

3.23.2 Environmental Requirements

Hardware Requirements

Table 3-113 lists the hardware requirements.

Table 3-113 Hardware

Item	Remarks
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-114 lists the software requirements.

Table 3-114 Software requirements

Item	Version	Download Address
GROMACS	2019.3	http://ftp.gromacs.org/pub/gromacs/gromacs-2019.3.tar.gz
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.6.tar.gz
FFTW	3.3.8	https://fftw.org/pub/fftw/fftw-3.3.8.tar.gz
CMake	3.8.1	https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
Test cases	water_GMX50_bare	https://ftp.gromacs.org/pub/benchmarks/water_GMX50_bare.tar.gz

OS Requirements

Table 3-115 lists the OS requirements.

Table 3-115 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

3.23.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GROMACS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-116 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/opt/compiler	Installation path of the BiSheng Compiler.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/HMPI	Installation path of the Hyper MPI.	
4	/usr/local/cuda-11.4	Installation path of NVIDIA CUDA.	
5	/path/to/OPENBLAS	Installation path of OpenBLAS.	
6	/path/to/FFTW	Installation path of FFTW.	
7	/path/to/CMAKE	Installation path of CMake.	
8	/path/to/GROMACS	Installation path of GROMACS.	
9	/path/to/CASE	Path of GROMACS test cases.	

3.23.4 Configuring the Compilation Environment

3.23.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-117 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Disabling the Nouveau driver	See **3.23.4.2 Disabling the Nouveau Driver**.
3	Installing NVIDIA CUDA	See **3.23.4.3 Installing NVIDIA CUDA**.
4	Installing the BiSheng Compiler	See **3.23.4.4 Installing the BiSheng Compiler**.
5	Installing Hyper MPI	See **3.23.4.5 Installing Hyper MPI**.
6	Installing OpenBLAS	See **3.23.4.6 Installing OpenBLAS**.
7	Installing FFTW	See **3.23.4.7 Installing FFTW**.
8	Installing CMake	See **3.23.4.8 Installing CMake**.

3.23.4.2 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

3.23.4.3 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3** to **Step 5**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Install the NVIDIA CUDA component.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Configure the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 5 Perform the verification.

```
nvcc --version
```

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

3.23.4.4 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the BiSheng Compiler installation package.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-
compiler-1.3.3-aarch64-linux.tar.gz
```

Step 3 Create a BiSheng Compiler installation directory (**/opt/compiler** is used as an example).

```
mkdir -p /opt/compiler
```

Step 4 Copy the BiSheng Compiler installation package to the installation directory.

```
cp -r bisheng-compiler-1.3.3-aarch64-linux.tar.gz /opt/compiler
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Go to the installation package directory.

```
cd /opt/compiler
```

Step 6 Decompress the installation package.

```
tar -zxvf bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 7 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 8 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

3.23.4.5 Installing Hyper MPI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the HUCX and XUCG source packages.

```
wget -c https://github.com/kunpengcompute/hucx/archive/refs/heads/huawei.zip -O hucx-huawei.zip
wget -c https://github.com/kunpengcompute/xucg/archive/refs/heads/huawei.zip -O xucg-huawei.zip
```

Step 3 Decompress the HUCX and XUCG source packages.

```
unzip hucx-huawei.zip
unzip xucg-huawei.zip
```

Step 4 Copy the content in the XUCG source package to the **src/ucg** directory of the HUCX source package.

```
cp -r xucg-huawei/* hucx-huawei/src/ucg/
```

Step 5 Go to the **hucx-huawei** directory.

```
cd hucx-huawei
```

Step 6 Configure the environment variables.

```
export SEC_CFLAGS="-Wno-error=deprecated-declarations -DHAVE___CLEAR_CACHE=1 -Wno-error=asm-operand-widths -Wno-error=implicit-int-float-conversion -Wno-error=format -Wno-error=tautological-pointer-compare"
```

Step 7 Create an HUCX installation directory.

```
mkdir -p /path/to/ucx
```

Step 8 Install HUCX.

```
./autogen.sh
./contrib/configure-opt --prefix=/path/to/ucx CFLAGS="$SEC_CFLAGS" CXXFLAGS="$SEC_CFLAGS"
make -j32
make install
```

Step 9 Obtain the Hyper MPI source package.

```
wget -c https://github.com/kunpengcompute/hmpi/archive/refs/heads/huawei.zip -O hmpi-huawei.zip
```

Step 10 Decompress the Hyper MPI source package.

```
unzip hmpi-huawei.zip
```

 Copyright © Huawei Technologies Co., Ltd.

Step 11 Go to the **hmpi-huawei** directory.

```
cd hmpi-huawei
```

Step 12 Install Hyper MPI.

```
./autogen.pl
./configure --prefix=/path/to/ompi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=/path/to/ucx
make -j16
make install
```

Step 13 Configure the environment variables.

```
export PATH=/path/to/ompi/bin:/path/to/ucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/ompi/lib:/path/to/ucx/lib:$LD_LIBRARY_PATH
export OPAL_PREFIX=/path/to/ompi/
export INCLUDE=/path/to/ompi:$INCLUDE
```

----End

3.23.4.6 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -zxvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Compile and install OpenBLAS.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 5 Set the OpenBLAS environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.23.4.7 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the FFTW installation package.

```
wget https://fftw.org/pub/fftw/fftw-3.3.8.tar.gz
```

Step 3 Decompress the FFTW installation package.

```
tar –zxvf fftw-3.3.8.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 5 Execute the **bootstrap.sh** script.

```
./bootstrap.sh
```

Step 6 Perform the configuration and installation.

 Copyright © Huawei Technologies Co., Ltd.

```
./configure --prefix=/path/to/FFTW --enable-single --enable-float --enable-neon --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
make clean
./configure --prefix=/path/to/FFTW --enable-long-double --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
make clean
./configure --prefix=/path/to/FFTW --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
```

Step 7 Configure the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.23.4.8 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the CMake installation package.

```
wget https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz
```

Step 3 Decompress the CMake installation package.

```
tar -zxvf cmake-3.8.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cmake-3.8.1
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/CMAKE
```

Step 6 Perform the compilation and installation.

```
make -j64
make install
```

Step 7 Configure the environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.23.5 Obtaining the Source Code

Procedure

Step 1 Download the GROMACS installation package.

http://ftp.gromacs.org/pub/gromacs/gromacs-2019.3.tar.gz

Step 2 Use an SFTP tool to upload the GROMACS installation package to the */path/to/GROMACS* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.23.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/GROMACS
```

Step 3 Decompress the installation package.

```
tar -xvf gromacs-2019.3.tar.gz
```

Step 4 Go to the directory generated after the decompression and create a **build** directory.

```
cd gromacs-2019.3
mkdir build
```

Step 5 Edit the configuration file.

1. Open the configuration file.
   ```
   vi cmake/gmxManageNvccConfig.cmake
   ```
2. Press **i** to enter the insert mode and comment out lines 116, 117, 120, and 121.

> **NOTE**
>
> After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Perform the configuration.

```
sed -i '212s/return fftw_version;/return 0;/g' src/gromacs/utility/binaryinformation.cpp
sed -i '214s/return fftwf_version;/return 0;/g' src/gromacs/utility/binaryinformation.cpp
sed -i '457s/int fftwflags = FFTW_DESTROY_INPUT;/int fftwflags = 0;/g' src/gromacs/fft/fft5d.cpp
sed -i '587s/FFTWPREFIX(cleanup)();/fftwf_cleanup_threads();/g' src/gromacs/fft/fft_fftw3.cpp
cd build
FLAGS="-march=armv8.2-a -mcpu=tsv110"; CFLAGS=$FLAGS CXXFLAGS=$FLAGS LDFLAGS="-fuse-ld=lld" CC=mpicc CXX=mpicxx \
cmake -DCMAKE_INSTALL_PREFIX=/path/to/GROMACS \
-DBUILD_SHARED_LIBS=on \
-DBUILD_TESTING=on \
-DREGRESSIONTEST_DOWNLOAD=off \
-DGMX_BUILD_OWN_FFTW=off \
-DGMX_SIMD=ARM_NEON_ASIMD \
-DGMX_DOUBLE=off \
-DGMX_EXTERNAL_BLAS=on \
-DGMX_EXTERNAL_LAPACK=on \
-DGMX_FFT_LIBRARY=fftw3 \
-DGMX_BLAS_USER=/path/to/OPENBLAS/lib/libopenblas.a \
-DGMX_LAPACK_USER=/path/to/OPENBLAS/lib/libopenblas.a \
-DFFTWF_LIBRARY=/path/to/FFTW/lib/libfftw3f.so \
-DFFTWF_INCLUDE_DIR=/path/to/FFTW/include \
-DGMX_GPU=on \
```

 Copyright © Huawei Technologies Co., Ltd.

```
-DGMX_OPENMP=on \
-DGMX_X11=off  \
-DGMX_MPI=on \
-DHWLOC_LIBRARIES=/usr/lib64 \
-DCUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda \
../
```

Step 7 Perform the compilation.

```
make -j40 V=1
make -j40 install
```

Step 8 Check whether an executable file is generated.

```
ll /path/to/GROMACS/bin/gmx_mpi
```

If the command output contains the **gmx_mpi** file information, an executable file is successfully generated.

Step 9 Configure the environment variables.

```
export PATH=/path/to/GROMACS/bin:$PATH
```

----End

3.23.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/CASE
```

Step 3 Decompress the test case file.

```
tar xvf water_GMX50_bare.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd water-cut1.0_GMX50_bare/0768
```

Step 5 Create a **topol.tpr** file.

```
gmx_mpi grompp -f pme.mdp
```

Step 6 Check whether a **topol.tpr** file is generated.

```
ll topol.tpr
```

If the command output contains the **topol.tpr** file information, an executable file is successfully generated.

Step 7 Start a GROMACS test.

```
mpirun --allow-run-as-root   -np 1  gmx_mpi mdrun -pin on  -ntomp 64 -v -nsteps 10000 -resetstep 8000  -noconfout  -pme gpu   -nstlist 400  -s topol.tpr
```

View the value of **ns/day** under **Performance** in the **md.log** file. A higher value indicates a higher performance.

The following is an example of the test result:

```
               Core t (s)   Wall t (s)        (%)
       Time:      915.374       14.303     6400.0
                 (ns/day)    (hour/ns)
Performance:       24.175        0.993
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.24 NAMD 2.14 Porting Guide (CentOS 8.2)

3.24.1 Introduction

Nanoscale Molecular Dynamics (NAMD) is computer software for molecular dynamics simulation, written using the Charm++ parallel programming model. It is noted for its parallel efficiency and is often used to simulate large systems (millions of atoms).

For more information about NAMD, visit the **NAMD official website**.

Programming language: C++

Brief description: NAMD is computer software for molecular dynamics simulation.

Open source license: custom open source license

Recommended Software Version

NAMD 2.14

3.24.2 Environment Requirements

Hardware Requirements

Table 3-118 lists the hardware requirements.

Table 3-118 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-119 lists the software requirements.

Table 3-119 Software requirements

Item	Version	Download Address
NAMD	2.14	**https://www.ks.uiuc.edu/Research/namd**
FFTW	3.3.8	**http://www.fftw.org/fftw-3.3.8.tar.gz**
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
Hyper MPI	1.1.1	https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
Test cases	stmv_nve_cuda	https://www.ks.uiuc.edu/Research/namd/utilities/stmv.tar.gz

OS Requirements

Table 3-120 lists the OS requirements.

Table 3-120 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

3.24.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NAMD software porting.

Table 3-121 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FFTW	Installation path of FFTW.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/NAMD	Installation path of NAMD.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/CASE	Path of NAMD test cases.	

3.24.4 Configuring the Compilation Environment

3.24.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-122 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing FFTW and TCL	See **3.24.4.2 Installing FFTW and TCL**.
3	Disabling the Nouveau driver	See **3.24.4.3 Disabling the Nouveau Driver**.
4	Installing NVIDIA CUDA	See **3.24.4.4 Installing NVIDIA CUDA**.
5	Installing the BiSheng Compiler	See **3.24.4.5 Installing the BiSheng Compiler**.

3.24.4.2 Installing FFTW and TCL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install FFTW and TCL.

 Copyright © Huawei Technologies Co., Ltd.

```
yum -y install fftw-devel tcl-devel
```

----End

3.24.4.3 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

3.24.4.4 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the NVIDIA CUDA component.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 3 Install the NVIDIA CUDA component.

```
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Perform the verification.

```
ls -l /usr/local/cuda-11.4/
```

Step 5 Configure the CUDA environment variables.

```
export CUDA_HOME=/usr/local/cuda-11.4
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.24.4.5 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the BiSheng Compiler installation package.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 3 Create a BiSheng Compiler installation directory (**/opt/compiler** is used as an example).

```
mkdir -p /opt/compiler
```

Step 4 Copy the BiSheng Compiler installation package to the installation directory.

```
cp -r bisheng-compiler-1.3.3-aarch64-linux.tar.gz /opt/compiler
```

Step 5 Go to the installation package directory.

```
cd /opt/compiler
```

Step 6 Decompress the installation package.

```
tar -zxvf bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 7 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 8 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

3.24.5 Obtaining the Source Code

Procedure

Step 1 Download the NAMD installation package. (You need to register an account for the download.)

https://www.ks.uiuc.edu/Research/namd

Step 2 Use an SFTP tool to upload the NAMD installation package to the **/path/to/NAMD** directory on the server.

----End

3.24.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/NAMD
```

Step 3 Decompress the NAMD installation package.

```
tar -xvf NAMD_2.14_Source.tar.gz
```

Step 4 Go to the **NAMD_2.14_Source** directory generated after the decompression.

```
cd NAMD_2.14_Source
```

Step 5 Decompress the **charm-6.10.2.tar** installation package.

```
tar xvf charm-6.10.2.tar
```

Step 6 Go to the **charm-6.10.2** directory generated after the decompression.

```
cd charm-6.10.2
```

Step 7 Perform the configuration.

```
./build charm++ multicore-arm8 clang  flang  --with-production
```

Step 8 Go to the directory generated after the decompression.

```
cd  multicore-arm8-flang-clang/tests/charm++/megatest/
```

Step 9 Install the components.

```
make pgm
./pgm +p4
```

Step 10 Modify the FFTW and TCL configuration items.

1. Copy the FFTW3 template.
   ```
   cd /path/to/NAMD
   cp arch/Linux-x86_64.fftw3 arch/Linux-ARM64.fftw
   ```
2. Open the **arch/Linux-ARM64.fftw** file.
   ```
   vi arch/Linux-ARM64.fftw
   ```
3. Press **i** to enter the insert mode and change **FFTDIR** to **/usr**.
   ```
   FFTDIR=/usr
   ```
4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
5. Open the **arch/Linux-ARM64.tcl** file.
   ```
   vi arch/Linux-ARM64.tcl
   ```
6. Press **i** to enter the insert mode, change **TCLDIR** to **/usr**, and change the library file version to **-ltcl8.6**.
   ```
   TCLDIR=/usr
   TCLLIB=-L$(TCLDIR)/lib -ltcl8.6 -ldl -lpthread
   ```
7. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 11 Compile NAMD.

```
./config Linux-ARM64-g++ --charm-arch multicore-arm8-flang-clang --with-cuda --cxx clang++ --cc clang
cd Linux-ARM64-g++/
make
```

An **namd2** executable program is generated in the current directory after the compilation is complete.

----End

3.24.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory.

```
mkdir -p path/to/CASE
```

Step 3 Go to the working directory.

```
cd /path/to/CASE
```

Step 4 Obtain the test case and the binary file.

```
wget https://www.ks.uiuc.edu/Research/namd/utilities/stmv.tar.gz
wget https://www.ks.uiuc.edu/Research/namd/2.13/benchmarks/stmv_nve_cuda.namd
```

Step 5 Decompress **stmv.tar.gz**.

```
tar xvf stmv.tar.gz
```

Step 6 Copy **stmv_nve_cuda.namd** to the **stmv** directory.

```
cp stmv_nve_cuda.namd stmv
```

Step 7 Run the test case to start a test.

```
cd stmv
/path/to/NAMD/NAMD_2.14_Source/Linux-ARM64-g++/namd2 +p96 +setcpuaffinity +maffinity
+isomalloc_sync +devices 0,1 stmv_nve_cuda.namd
```

View the value of **day/ns** in the log. A smaller value indicates a higher performance.

----End

3.25 CP2K 7.1 Porting Guide (openEuler 21.03)

3.25.1 Introduction

CP2K is a quantum chemistry and solid-state physics package that can perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It provides a general framework for different modeling methods. It supports DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, and MNDO), and classical force fields (AMBER and CHARMM). CP2K can use NEB or dimer method to do simulations of molecular dynamics, metadynamics, Monte Carlo, Ellenfast dynamics, vibration analysis, core level spectroscopy, energy minimization, and transition state optimization.

For more information, visit the **official CP2K website**.

Programming language: Fortran 2008

Brief description: A quantum chemistry and solid-state physics package used to perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems.

Open source license: GPL

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

CP2K 7.1

3.25.2 Environment Requirements

Hardware Requirements

Table 3-123 lists the hardware requirements.

Table 3-123 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-124 lists the software requirements.

Table 3-124 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi
CP2K	7.1	https://github.com/cp2k/cp2k/releases/tag/v7.1.0
LIBINT	2.6.0	https://github.com/evaleev/libint/archive/v2.6.0.tar.gz
LIBXC	4.3.4	https://www.abinit.org/fallbacks
FFTW	3.3.8	https://www.cp2k.org/static/downloads/
LAPACK	3.8.0	https://www.cp2k.org/static/downloads/
ScaLAPACK	2.1.0	https://www.cp2k.org/static/downloads/
CMake	3.16.4	https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz
GMP	6.2.0	https://gmplib.org/download/gmp/gmp-6.2.0.tar.xz
Boost	1.72	https://github.com/boostorg/boost/archive/refs/tags/boost-1.72.0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
DBCSR	2.0.1	**https://github.com/cp2k/dbcsr/releases/download/v2.0.1/dbcsr-2.0.1.tar.gz**
ELPA	2019.05.001	**https://www.cp2k.org/static/downloads/elpa-2019.05.001.tar.gz**
SPGLIB	1.12.2	**https://github.com/spglib/spglib/archive/v1.11.2.1.tar.gz**
GSL	2.6	**http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz**
PLUMED	2.5.2	**https://www.cp2k.org/static/downloads/plumed-2.5.2.tgz**
OPENSSL	1.1.1l	**https://www.openssl.org/source/openssl-1.1.1l.tar.gz**
Test cases	H2O-256.inp	Test case provided by the software: **cp2k-7.1.0/benchmarks/QS/**.

OS Requirements

Table 3-125 lists the OS requirements.

Table 3-125 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://repo.openeuler.org/openEuler-21.03/**
Kernel	4.14.0-115	Included in the OS image

3.25.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CP2K software porting.

Table 3-126 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/BISHENG**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

2	**/path/to/HMPI**	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	**/path/to/CMAKE**	Installation path of CMake.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	**/path/to/GMP**	Installation path of GMP.	
5	**/path/to/BOOST**	Installation path of Boost.	
6	**/path/to/LIBINT**	Installation path of Libint.	
7	**/path/to/FFTW**	Installation path of FFTW.	
8	**/path/to/LAPACK**	Installation path of LAPACK.	
9	**/path/to/SCALAPACK**	Installation path of ScaLAPACK.	
10	**/path/to/ELPA**	Installation path of ELPA.	
11	**/path/to/SPGLIB**	Installation path of Spglib.	
12	**/path/to/LIBXC**	Installation path of Libxc.	
13	**/path/to/GSL**	Installation path of GSL.	
14	**/path/to/PLUMED**	Installation path of PLUMED.	
15	**/path/to/EXTRA**	Path of storing the library files generated by Libint, FFTW, LAPACK, ScaLAPACK, ELPA, Spglib, Libxc, GSL, and PLUMED.	
16	**/path/to/OPENSSL**	Installation path of OpenSSL.	
17	**/path/to/EXTRA/MATH**	Path of LAPACK and ScaLAPACK library files.	

 Copyright © Huawei Technologies Co., Ltd.

3.25.4 Configuring the Compilation Environment

3.25.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-127 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing OpenSSL	See **3.25.4.2 Installing OpenSSL**.
4	Installing CMake	See **3.25.4.3 Installing CMake**.
5	Installing GMP	See **3.25.4.4 Installing GMP**.
6	Installing Boost	See **3.25.4.5 Installing Boost**.
7	Installing Libint	See **3.25.4.6 Installing Libint**.
8	Installing FFTW	See **3.25.4.7 Installing FFTW**.
9	Installing LAPACK	See **3.25.4.8 Installing LAPACK**.
10	Installing ScaLAPACK	See **3.25.4.9 Installing ScaLAPACK**.
11	Installing ELPA	See **3.25.4.10 Installing ELPA**.
12	Installing Spglib	See **3.25.4.11 Installing Spglib**.
13	Installing Libxc	See **3.25.4.12 Installing Libxc**.
14	Installing GSL	See **3.25.4.13 Installing GSL**.
15	Installing PLUMED	See **3.25.4.14 Installing PLUMED**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/bisheng-compiler-1.3.3-aarch64-linux/lib:
$LD_LIBRARY_PATH
export INCLUDE=/path/to/HMPI/hmpi/include
export PATH=/path/to/HMPI/hmpi/bin:/path/to/HMPI/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/HMPI/hmpi/
export LD_LIBRARY_PATH=/path/to/HMPI/hmpi/lib:/path/to/HMPI/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

3.25.4.2 Installing OpenSSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Configure the environment variables.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 3 Decompress the OpenSSL installation package.

```
tar -zxvf openssl-1.1.1l.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd openssl-1.1.1l
```

Step 5 Perform the configuration.

```
./Configure --prefix=/path/to/OPENSSL
```

Step 6 Perform the installation.

```
make -j&&make install
```

----End

3.25.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar -xvf cmake-3.16.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

 Copyright © Huawei Technologies Co., Ltd.

```
cd cmake-3.16.4
```

Step 4 Perform the configuration.

```
export OPENSSL_ROOT_DIR=/path/to/OPENSSL
export OPENSSL_INCLUDE_DIR=/path/to/OPENSSL/include
export OPENSSL_CRYPTO_DIR=/path/to/OPENSSL/lib
./configure --prefix=/path/to/CMAKE
```

Step 5 Perform the installation.

```
make -j&&make install
```

Step 6 Set the environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.25.4.4 Installing GMP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the GMP installation package.

```
tar -xvf gmp-6.2.0.tar.xz
```

Step 3 Go to the directory generated after the decompression.

```
cd gmp-6.2.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/GMP --enable-cxx
```

Step 5 Perform the installation.

```
make -j && make install
```

----End

3.25.4.5 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Boost installation package.

```
tar -xvf boost_1_72_0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd boost_1_72_0
```

Step 4 Perform the configuration.

```
./bootstrap.sh --with-toolset=clang
```

Step 5 Perform the installation.

```
./b2 install --prefix=/path/to/BOOST
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.25.4.6 Installing Libint

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies using the **yum** command.

```
yum install gmp-devel.aarch64 libudev* -y
```

Step 3 Create an installation directory.

```
mkdir -p /path/to/EXTRA
mkdir -p /path/to/EXTRA/mathlib
```

Step 4 Decompress the Libint installation package.

```
tar -xvf libint-2.6.0.tar.gz
```

Step 5 Go to the directory generated after the decompression and perform the configuration.

```
cd libint-2.6.0
./autogen.sh
```

Step 6 Create a **build** directory and switch to the directory.

```
mkdir build
cd build
```

Step 7 Configure the environment variables.

```
export LDFLAGS="-L/path/to/GMP/gmp-6.2/lib -L/path/to/BOOST/boost_1_72/lib"
export CPPFLAGS="-I/path/to/BOOST/boost_1_72/include/ -I/path/to/GMP/gmp-6.2/include"
```

Step 8 Perform the configuration.

```
../configure CXX=mpicxx --enable-eri=1 --enable-eri2=1 --enable-eri3=1 --with-max-am=4 --with-eri-max-am=4,3 --with-eri2-max-am=6,5 --with-eri3-max-am=6,5 --with-opt-am=3 --enable-generic-code --disable-unrolling --with-libint-exportdir=libint_cp2k_lmax4
```

Step 9 Perform the compilation.

```
make export
```

Step 10 Configure the environment variables.

```
export LDFLAGS="-L/path/to/GMP/lib -L/path/to/BOOST/lib"
export CPPFLAGS="-I/path/to/BOOST/include/ -I/path/to/GMP/include"
```

Step 11 Decompress the **libint_cp2k_lmax4.tgz** installation package.

```
tar -xvf libint-v2.6.0-cp2k-lmax-4.tgz
cd  libint-v2.6.0-cp2k-lmax-4
```

Step 12 Perform the configuration.

```
./configure --prefix=/path/to/EXTRA/LIBINT CC=mpicc CXX=mpicxx FC=mpifort --enable-fortran --enable-shared
```

Step 13 Perform the installation.

```
make -j 32
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.25.4.7 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

```
./configure CC=clang F77=flang --enable-shared --enable-threads --enable-openmp --enable-mpi MPICC=mpicc --prefix=/path/to/EXTRA/FFTW
```

Step 5 Perform the compilation and installation.

```
make -j && make install
```

----End

3.25.4.8 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the LAPACK installation package.

```
tar -xvf lapack-3.8.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd lapack-3.8.0
```

Step 4 Generate a **make.inc** file.

```
cp make.inc.example make.inc
```

Step 5 Edit the **make.inc** file.

1. Open the **make.inc** file.
   ```
   vi make.inc
   ```
2. Press **i** to enter the edit mode and modify the file as follows:

   Modify the content in line 11 to the following:

   ```
   CC = clang
   ```

   Modify the content in line 22 to the following:

   ```
   FORTRAN = flang
   ```

   Modify the content in line 30 to the following:

   ```
   LOADER = flang
   ```

   Modify the content in line 81 to the following:

   ```
   libblas.a
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Edit the **INSTALL/second_INT_ETIME.f** file.

1. Open the **INSTALL/second_INT_ETIME.f** file.
   ```
   vi INSTALL/second_INT_ETIME.f
   ```
2. Press **i** to enter the edit mode and add the following function below line 35 in the **second_INT_ETIME.f** file.
   ```
   REAL Function etime(time)
   REAL time(2)
   Call Cpu_Time(etime)
   time(1) = etime
   time(2) = 0
   End Function
   ```
   *Comment out the following lines:
   ```
   *REAL       ETIME
   *INTRINSIC   ETIME
   ```
   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Edit the **INSTALL/dsecnd_INT_ETIME.f** file.

1. Open the **INSTALL/dsecnd_INT_ETIME.f** file.
   ```
   vi INSTALL/dsecnd_INT_ETIME.f
   ```
2. Press **i** to enter the edit mode and add the following function below line 35 in the **dsecnd_INT_ETIME.f** file.
   ```
   REAL Function etime(time)
   REAL time(2)
   Call Cpu_Time(etime)
   time(1) = etime
   time(2) = 0
   End Function
   ```
   *Comment out the following lines:
   ```
   *REAL       ETIME
   *INTRINSIC   ETIME
   ```
   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the installation.
```
make -j
```

Step 9 Copy the static library to another directory.
```
cp *.a /path/to/EXTRA/mathlib
```

----End

3.25.4.9 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ScaLAPACK installation package.

```
tar -xvf scalapack-2.1.0.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd scalapack-2.1.0
```

Step 4 Run the following command to copy and paste the file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Modify the **SLmake.inc** file.

1. Open the **SLmake.inc** file.
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   BLASLIB=/path/to/LAPACK/libblas.a
   LAPACKLIB=/path/to/LAPACK/liblapack.a
   LIBS=$(LAPACKLIB) $(BLASLIB)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation.

```
make
```

Step 7 Copy the generated static library to another directory.

```
cp *.a /path/to/EXTRA/mathlib
```

----End

3.25.4.10 Installing ELPA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ELPA installation package.

```
tar -xvf elpa-2019.05.001.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd elpa-2019.05.001
```

Step 4 Perform the configuration.

```
export CC=`which mpicc`
export CXX=`which mpicxx`
export FC=`which mpifort`
export LDFLAGS="-L/path/to/EXTRA/mathlib/"
export CFLAGS="-L/path/to/EXTRA/mathlib/"
export CXXFLAGS="-L/path/to/EXTRA/mathlib/"
export FCFLAGS="-L/path/to/EXTRA/mathlib/"
export LIBS="/path/to/CP2K/EXTRA/mathlib/libscalapack.a /path/to/CP2K/EXTRA/mathlib/liblapack.a /path/to/CP2K/EXTRA/mathlib/libblas.a"
./configure --prefix=/path/to/EXTRA/ELPA --enable-openmp --enable-shared=no --disable-sse --disable-sse-assembly --disable-avx --disable-avx2
```

Step 5 Perform the installation.

```
make
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.25.4.11 Installing Spglib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Spglib installation package.

```
tar -xvf  spglib-1.12.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd spglib-1.12.2
```

Step 4 Perform the configuration.

```
cmake -DCMAKE_INSTALL_PREFIX="/path/to/EXTRA/SPGLIB"
```

Step 5 Perform the installation.

```
make -j &&make install
```

----End

3.25.4.12 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Libxc installation package.

```
tar -xvf libxc-4.3.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd libxc-4.3.4
```

Step 4 Perform the configuration.

```
./configure  AR=ar  FC=flang  F77=flang  F90=flang  CC=clang
--prefix=/path/to/EXTRA/LIBXC
```

Step 5 Perform the installation.

```
make
make install
```

----End

3.25.4.13 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the GSL installation package.

```
tar -xvf gsl-2.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd gsl-2.6
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/EXTRA/GSL
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the compilation and installation.

```
make
make install
```

NOTE

If "error adding symblis:DSO missing from command line" is reported during the compilation, check the environment variables. The environment variables you configured in **3.25.4.10 Installing ELPA** affect the compilation in **3.25.4.13 Installing GSL**. Solution: Run the following command to delete the LDFLAGS, CFLAGS, CXXFLAGS, FCFLAGS, and LIBS environment variables:

```
unset LDFLAGS CFLAGS CXXFLAGS FCFLAGS LIBS
```

Step 6 Set the environment variables.

```
export LIBRARY_PATH=/path/to/EXTRA/GSL/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/EXTRA/GSL/lib:$LD_LIBRARY_PATH
```

----End

3.25.4.14 Installing PLUMED

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PLUMED installation package.

```
tar -xvf plumed-2.5.2.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd plumed-2.5.2
```

Step 4 Perform the configuration.

```
export LDFLAGS=-L/path/to/EXTRA/mathlib
export LIBS="/path/to/EXTRA/mathlib/libscalapack.a /path/to/EXTRA/mathlib/liblapack.a /path/to/EXTRA/mathlib/libblas.a"
./configure CXX=mpicxx CC=mpicc FC=mpifort --prefix=/path/to/EXTRA/PLUMED --enable-external-blas --enable-gsl --enable-external-lapack
```

Step 5 Perform the installation.

```
make
make install
```

----End

3.25.5 Obtaining the Source Code

Procedure

Step 1 Download the CP2K installation package.

https://github.com/cp2k/cp2k/releases/tag/v7.1.0

Step 2 Use an SFTP tool to upload the CP2K installation package to the */path/to/CP2K* directory on the server.

----End

3.25.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CP2K installation package.

```
tar -xvf cp2k-7.1.tar.bz2
```

Step 3 Go to the directory generated after the decompression.

```
cd cp2k-7.1/arch
```

Step 4 Modify the **Linux-x86-64-gfortran.psmp** file.

1. Open the **Linux-x86-64-gfortran.psmp** file.

   ```
   vi Linux-x86-64-gfortran.psmp
   ```

2. Press **i** to enter the edit mode and modify the following information in bold:

   ```
   #Tested with: GFortran 7.4.0, MPICH 3.3, LAPACK 3.8.0, ScaLAPACK 2.0.2, FFTW 3.3, Libint 2.6.0, Libxc
   4.3.4, libxsmm 1.14, ELPA 2019.05.001, PLUMED 2.5.2, SPGLIB 1.12.2
   # Author: Matthias Krack (matthias.krack@psi.ch, PSI, December 2019)
   CC         = mpicc
   FC         = mpif90
   LD         = mpif90
   AR         = ar -r
   GNU_PATH = /path/to/install
   MATHLIBPATH = /path/to/EXTRA/mathlib
   include /path/to/install/PLUMED/lib/plumed/src/lib/Plumed.inc.static
   ELPA_INC     = $(GNU_PATH)/ELPA/include/elpa_openmp-2019.05.001
   ELPA_LIB     = $(GNU_PATH)/ELPA/lib
   FFTW_INC     = $(GNU_PATH)/FFTW/include
   FFTW_LIB     = $(GNU_PATH)/FFTW/lib
   LIBINT_INC   = $(GNU_PATH)/LIBINT/include
   LIBINT_LIB   = $(GNU_PATH)/LIBINT/lib
   LIBXC_INC    = $(GNU_PATH)/LIBXC/include
   LIBXC_LIB    = $(GNU_PATH)/LIBXC/lib
   SPGLIB_INC   = $(GNU_PATH)/SPGLIB/include
   SPGLIB_LIB   = $(GNU_PATH)/SPGLIB/lib
   PLUMED_LIB   =  $(GNU_PATH)/PLUMED/lib
   PLUMED_INC   =  $(GNU_PATH)/PLUMED/include
   GSLBLAS_LIB =  $(GNU_PATH)/GSL/lib
   GSLBLAS_INC =  $(GNU_PATH)/GSL/include
   CFLAGS       = -O1 -g -mtune=native -fno-unroll-loops -fstack-protector
   DFLAGS       = -D__ELPA -D__FFTW3 -D__LIBINT -D__LIBXC
   DFLAGS      += -D__MPI_VERSION=3 -D__PLUMED2 -D__SPGLIB
   DFLAGS      += -D__parallel -D__SCALAPACK
   FCFLAGS      = $(CFLAGS) $(DFLAGS)
   FCFLAGS     += -ffree-form -ffree-line-length-none
   FCFLAGS     += -fopenmp -fno-omit-frame-pointer
   FCFLAGS     += -ftree-vectorize -funroll-loops -std=f2008
   FCFLAGS     += -I$(ELPA_INC)/elpa -I$(ELPA_INC)/modules
   FCFLAGS     += -I$(FFTW_INC) -I$(LIBINT_INC) -I$(LIBXC_INC) -I$(GSLBLAS_INC) -I$(PLUMED_INC)
   LDFLAGS      = $(FCFLAGS) -static-flang-libs
   LIBS         = -L$(GSLBLAS_LIB) -L$(PLUMED_LIB) -lgsl -lgslcblas -lz -lplumed -lplumedKernel
   LIBS        += $(ELPA_LIB)/libelpa_openmp.a
   LIBS        += $(LIBXC_LIB)/libxcf03.a
   LIBS        += $(LIBXC_LIB)/libxc.a
   LIBS        += $(LIBINT_LIB)/libint2.a
   LIBS        += $(SPGLIB_LIB)/libsymspg.a
   LIBS        += $(FFTW_LIB)/libfftw3.a
   LIBS        += $(FFTW_LIB)/libfftw3_threads.a
   LIBS        += $(MATHLIBPATH)/libscalapack.a
   LIBS         += $(MATHLIBPATH)/liblapack.a
   LIBS        += $(MATHLIBPATH)/libblas.a
   LIBS        += -ldl -lpthread -lstdc++
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Go to the **cp2k-7.1** directory.

```
cd ../
```

Step 6 Perform the installation and compilation.

```
export LD_LIBRARY_PATH=/path/to/PLUMED/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/GSL/lib:$LD_LIBRARY_PATH
make -j 16 ARCH=Linux-x86-64-gfortran VERSION=psmp
```

----End

3.25.7 Running and Verification

NOTE

If multiple nodes are running, configure the network environment:

- In the **/etc/hosts** file, add the IP addresses and host names of the nodes.
- Enable SSH password-free login between the nodes.
- Disable the firewall and ensure that clients and servers are not disconnected.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Load the environment variables.

```
export PATH=/path/to/CP2K/cp2k-7.1.0/exe/Linux-x86-64-gfortran:$PATH
```

Step 3 Create a **hostfile** file.

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   Node1
   Node2
   ```

   NOTE

   Node1 and **Node2** are the node names. Change them based on the site requirements and ensure that the nodes can ping each other.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

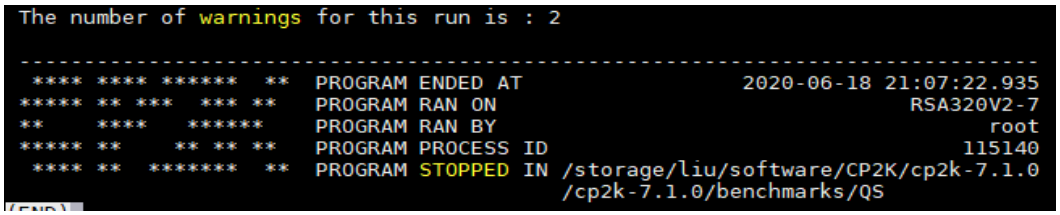
Step 4 Run the following commands as the root user:

- Single node
  ```
  mpirun --allow-run-as-root -np 96 -x OMP_NUM_THREADS=1 cp2k.psmp /path/to/CP2K/
  cp2k-7.1.0/benchmarks/QS/H2O-256.inp > cp2k.H2O-256.inp.log
  ```
- Dual nodes
  ```
  mpirun --allow-run-as-root -np 192 -N 96 -x PATH=$PATH -x LD_LIBRARY_PATH=
  $LD_LIBRARY_PATH -hostfile hostfile -x OMP_NUM_THREADS=1 cp2k.psmp /path/to/CP2K/
  cp2k-7.1.0/benchmarks/QS/H2O-256.inp > cp2k.H2O-256.inp.log
  ```

Figure 3-14 Output example (1)

```
  **** **** ******  **  PROGRAM STARTED AT                2020-06-18 20:51:55.469
***** ** ***  *** **   PROGRAM STARTED ON                          RSA320V2-7
**    ****   ******    PROGRAM STARTED BY                               root
***** **    ** ** **   PROGRAM PROCESS ID                             115140
 **** **  *******  **  PROGRAM STARTED IN /storage/liu/software/CP2K/cp2k-7.1.0
                                         /cp2k-7.1.0/benchmarks/QS
```

Figure 3-15 Output example (2)



The time difference between **Figure 3-14** and **Figure 3-15** and is 927s.

Table 3-128 Parameter description

Parameter	Description
-np	Total number of running MPI processes.
-N	Number of processes running on each server.
-hostfile	Name of the node to be used.

----End

3.26 DL_POLY 1.10 Porting Guide (openEuler 21.03)

3.26.1 Introduction

DL_POLY is a piece of general purpose classical molecular dynamics (MD) simulation software developed at Daresbury Laboratory by I.T. Todorov, W. Smith, A.M. Elena, and others.

For more information about DL_POLY, visit the **DL_POLY official website**.

Programming language: Fortran

Brief description: DL_POLY is a piece of general purpose classical molecular dynamics (MD) simulation software.

Recommended Software Version

DL_POLY v1.10

3.26.2 Environment Requirements

Hardware Requirements

Table 3-129 lists the hardware requirements.

Table 3-129 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-130 lists the software requirements.

Table 3-130 Software requirements

Item	Version	Download URL
DL_POLY	v1.10	https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz
Test cases	TEST41.tar.gz	https://gitlab.com/DL_POLY_Classic/tests/-/blob/master/TEST41.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz

OS Requirements

Table 3-131 lists the OS requirements.

Table 3-131 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

3.26.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DL_POLY software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-132 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/DL_POLY**	Installation path of DL_POLY.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
3	**/path/to/hypermpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.

3.26.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-133 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hypermpi/hmpi/include:$INCLUDE
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

3.26.5 Obtaining the Source Code

Procedure

Step 1 Download the DL_POLY installation package.

https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz

Step 2 Use an SFTP tool to upload the DL_POLY installation package to the */path/to/DL_POLY* directory on the server.

----End

3.26.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the DL_POLY installation package directory.

```
cd /path/to/DL_POLY
```

Step 3 Decompress the DL_POLY installation package.

```
tar -xzf dl_poly-RELEASE-1-10.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd dl_poly-RELEASE-1-10/source
```

Step 5 Copy the **Makefile** file.

```
cp ../build/MakePAR ./Makefile
```

Step 6 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode and modify line 65 of **Makefile**.
   ```
   FC=mpif90 FFLAGS="-c -O3 -march=armv8.2-a -mtune=tsv110" \
   ```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make dlpoly
```

Step 8 Set the DL_POLY environment variables.

```
export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:$PATH
```

Step 9 Check whether DL_POLY is installed successfully.

```
which DLPOLY.X
```

If the following information is displayed, the installation is successful:

```
/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute/DLPOLY.X
```

----End

3.26.7 Running and Verification

Single Node

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the TEST41 test case package.

```
tar -zxf TEST41.tar.gz
```

Step 3 Go to the TEST41 test case directory.

```
cd TEST41/NHC
```

Step 4 Run the test case on the single node.

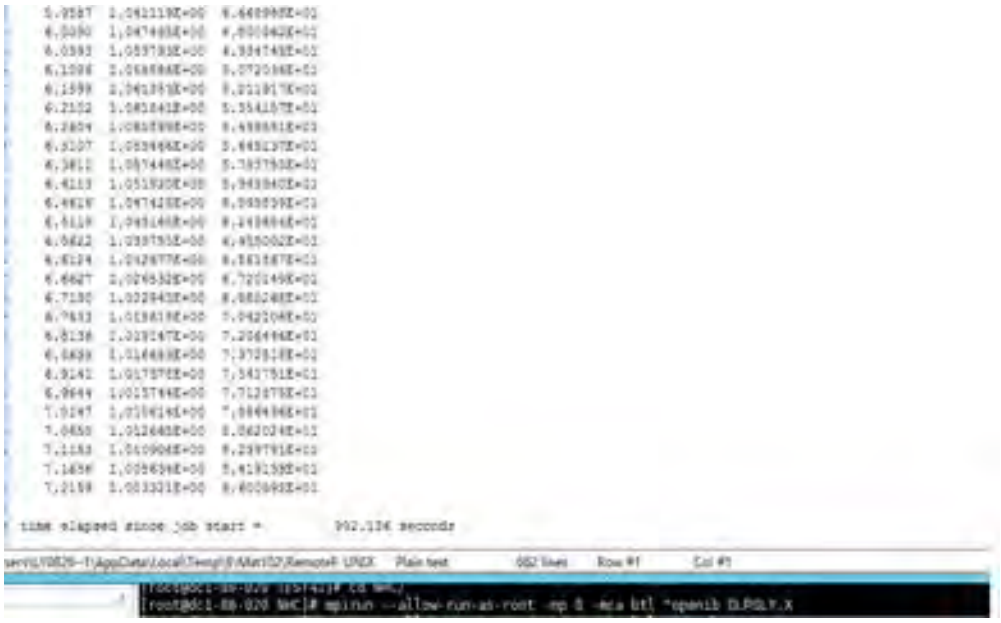
```
mpirun --allow-run-as-root -np 8 -mca btl ^openib DLPOLY.X
```

> **NOTE**
>
> The **-np** parameter indicates the total number of processes, and the **-N** parameter indicates the number of processes used by each node. Set the parameters based on the actual node configuration.

time elapsed since job start = *XX* **seconds** in the **OUTPUT** file indicates the performance. A smaller value of *XX* indicates higher performance.

The following is an example of the output result.

 Copyright © Huawei Technologies Co., Ltd.



----End

Multiple Nodes

NOTE

If multiple nodes are running, configure the network environment:

- In the **/etc/hosts** file, add the IP addresses and host names of the nodes.
- Enable SSH password-free login between the nodes.
- Disable the firewall and ensure that clients and servers are not disconnected.

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the TEST41 test case package.

```
tar -zxf TEST41.tar.gz
```

Step 3 Go to the TEST41 test case directory.

```
cd TEST41/NHC
```

Step 4 Run the following command to create the **hostfile** file and add node information:

```
echo -e 'node1|nnode2|n...|nnodex' > /path/to/HOSTFILE
```

NOTE

In the preceding command, *node1*, *node2*, ..., and *nodex* indicate the host names of the nodes. You can run the **hostname** command to query the host names. **\n** indicates a newline character.

Step 5 In the window of **node2**, set the environment variables.

```
vi /root/.bashrc
export PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.1-aarch64-linux/lib:$LD_LIBRARY_PATH
export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:$PATH
export INCLUDE=/path/to/hypermpi/hmpi/include
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the test case on the multiple nodes.

mpirun --allow-run-as-root -np *16* **-N** *8* **-x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH -x OMP_NUM_THREADS=1 --hostfile** */path/to/HOSTFILE* **-mca btl ^openib DLPOLY.X**

```
   5.9085   1.029963E+00   4.538330E+01
   5.9587   1.034407E+00   4.665902E+01
   6.0090   1.041681E+00   4.796549E+01
   6.0593   1.049816E+00   4.930428E+01
   6.1096   1.055420E+00   5.067266E+01
   6.1599   1.059505E+00   5.206903E+01
   6.2102   1.062367E+00   5.349214E+01
   6.2604   1.062243E+00   5.493821E+01
   6.3107   1.060903E+00   5.640575E+01
   6.3610   1.060489E+00   5.789619E+01
   6.4113   1.055105E+00   5.940260E+01
   6.4616   1.050095E+00   6.092547E+01
   6.5119   1.045202E+00   6.246492E+01
   6.5622   1.041734E+00   6.402306E+01
   6.6124   1.033177E+00   6.559217E+01
   6.6627   1.025465E+00   6.717334E+01
   6.7130   1.022875E+00   6.877442E+01
   6.7633   1.018795E+00   7.039309E+01
   6.8136   1.020053E+00   7.203794E+01
   6.8639   1.018109E+00   7.370399E+01
   6.9141   1.019254E+00   7.539643E+01
   6.9644   1.017927E+00   7.711135E+01
   7.0147   1.016900E+00   7.884936E+01
   7.0650   1.013769E+00   8.060696E+01
   7.1153   1.012209E+00   8.238692E+01
   7.1656   1.007656E+00   8.418400E+01
   7.2159   1.005724E+00   8.600290E+01

 time elapsed since job start =       17903.049 seconds
```

NOTE

The **-np** parameter indicates the total number of processes, and the **-N** parameter indicates the number of processes used by each node. Set the parameters based on the actual node configuration. Alternatively, you do not need to specify the two parameters and the maximum number of processes will be used. The **-x OMP_NUM_THREADS** parameter indicates the number of threads to be used.

----End

3.27 GROMACS 2019.3 Porting Guide (openEuler 21.03)

3.27.1 Introduction

GROningen MAchine for Chemical Simulations (GROMACS) is a computational engine for MD simulation and energy minimization, which simulates hundreds to millions of atomic systems using Newton's equilibrium equations. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a large number of complicated bonded interactions, but since it is extremely fast at calculating the nonbonded interactions (that usually dominate simulations),

many groups are also using it for research on non-biological systems, such as polymers. Compared with other MD simulation software, GROMACS has the following unique advantages:

- GROMACS is free of charge and complies with the GNU Lesser General Public License (LGPL) protocol. You can find the open-source code of GROMACS on GitHub.
- GROMACS provides higher performance than other software and has been optimized in code.
- GROMACS is user-friendly for topology files and parameter setting files. The format of GROMACS is similar to that of Python.
- The environment of GROMACS develops well, and many simulation tools support GROMACS well.

For more information, visit the **official GROMACS website**.

Programming language: C++

Brief description: GROMACS is a computational engine for MD simulation and energy minimization.

Open source license: LGPL Version 2.1

Recommended Software Version

GROMACS 2019.3

3.27.2 Environment Requirements

Hardware Requirements

Table 3-134 lists the hardware requirements.

Table 3-134 Hardware requirements

Item	Description
CPU	Kunpeng 920 7260 processor

Software Requirements

Table 3-135 lists the software requirements.

Table 3-135 Software requirements

Item	Version	Download URL
GROMACS	2019.3	**http://manual.gromacs.org/documentation/2019.3/download.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/releases
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
CMake	3.8.1	https://cmake.org/download/#previous
Test cases	water_GMX50_bare	https://ftp.gromacs.org/pub/benchmarks/

OS Requirements

Table 3-136 lists the OS requirements.

Table 3-136 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	Included in the OS image

3.27.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GROMACS software porting.

Table 3-137 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/path/to/BISHENG**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/HMPI**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/OPENBLAS**	Installation path of OpenBLAS.	The installation paths listed in this table are only examples. Shared

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
4	**/path/to/FFTW**	Installation path of FFTW.	paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
5	**/path/to/CMAKE**	Test path of CMake.	
6	**/path/to/GROMACS**	Installation path of GROMACS.	
7	**/path/to/CASE**	Path of GROMACS test cases.	

3.27.4 Configuring the Compilation Environment

3.27.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-138 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing OpenBLAS	See **3.27.4.2 Installing OpenBLAS**.
4	Installing FFTW	See **3.27.4.3 Installing FFTW**.
5	Installing CMake	See **3.27.4.4 Installing CMake**.

 Copyright © Huawei Technologies Co., Ltd.

3.27.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Set the environment variables.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 5 Compile and install OpenBLAS.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Set the OpenBLAS environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.27.4.3 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Execute the **bootstrap.sh** script.

```
./bootstrap.sh
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon --
enable-float
```

Step 6 Perform the compilation and installation.

```
make -j 96
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.27.4.4 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar zxvf cmake-3.8.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.8.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Perform the installation.

```
make
make install
```

Step 6 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.27.5 Obtaining the Source Code

Procedure

Step 1 Download the GROMACS installation package.

http://manual.gromacs.org/documentation/2019.3/download.html

Step 2 Use an SFTP tool to upload the GROMACS installation package to the */path/to/GROMACS* directory on the server.

----End

3.27.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the GROMACS installation package directory.

```
cd /path/to/GROMACS
```

Step 3 Decompress the GROMACS installation package.

```
tar -xvf gromacs-2019.3.tar.gz
```

Step 4 Go to the directory generated after the decompression and create a **build** directory.

```
cd gromacs-2019.3
mkdir build
cd build
```

Step 5 Perform the configuration.

```
FLAGS="-mcpu=tsv110"; CFLAGS=$FLAGS CXXFLAGS=$FLAGS LDFLAGS="-lflang" CC=mpicc
CXX=mpicxx \
```

 Copyright © Huawei Technologies Co., Ltd.

```
/path/to/CMAKE/bin/cmake \
-DCMAKE_INSTALL_PREFIX=/path/to/GROMACS/ \
-DBUILD_SHARED_LIBS=on \
-DBUILD_TESTING=on \
-DREGRESSIONTEST_DOWNLOAD=off \
-DGMX_BUILD_OWN_FFTW=off \
-DGMX_SIMD=ARM_NEON_ASIMD \
-DGMX_DOUBLE=off \
-DGMX_EXTERNAL_BLAS=on \
-DGMX_EXTERNAL_LAPACK=on \
-DGMX_FFT_LIBRARY=fftw3 \
-DGMX_BLAS_USER=/path/to/OPENBLAS/lib/libopenblas.a \
-DGMX_LAPACK_USER=/path/to/OPENBLAS/lib/libopenblas.a \
-DFFTWF_LIBRARY=/path/to/FFTW/lib/libfftw3f.so \
-DFFTWF_INCLUDE_DIR=/path/to/FFTW/include \
-DGMX_GPU=off \
-DGMX_MPI=on \
-DGMX_OPENMP=on \
-DGMX_X11=off \
../
```

Step 6 Perform the compilation and installation.

```
make -j40 V=1
make -j40 install
```

Step 7 Check whether an executable file is generated.

```
ll /path/to/GROMACS/bin/gmx_mpi
```

Step 8 Set the environment variables.

```
export PATH=/path/to/GROMACS/bin:$PATH
```

----End

3.27.7 Running and Verification

Single Node

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory and upload the test case file.

```
mkdir -p path/to/CASE
```

Step 3 Decompress the test case file.

```
tar xvf water_GMX50_bare.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd water-cut1.0_GMX50_bare/0768
```

Step 5 Create a **topol.tpr** file.

```
gmx_mpi grompp -f pme.mdp
```

Step 6 Check whether a **topol.tpr** file is generated.

```
ll topol.tpr
-rw-r--r-- 1 root root 18448672 Jan 11 16:41 topol.tpr
```

Step 7 Start the running.

```
mpirun --allow-run-as-root --mca btl ^openib -np 24 -x OMP_NUM_THREADS=4 gmx_mpi mdrun -dlb
yes -v -nsteps 10000 -resethway -noconfout -pin on -ntomp 4 -s topol.tpr
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

np indicates the number of MPI processes. If hyper-threading is not enabled, the number of MPI processes multiplied by the number of OMP threads must be less than or equal to the number of CPU cores. For example, in the command, the number of MPI processes (24) multiplied by the number of OMP threads (4) must be less than or equal to the number of CPU cores in the environment.

View the value of **ns/day** under **Performance** in the **md.log** file. A larger value indicates higher performance.

The following is an example of the test result.

```
Part of the total run time spent waiting due to load imbalance: 1.1%.
Steps where the load balancing was limited by -rdd, -rcon and/or -dds: X 0 % Y 0 %
Average PME mesh/force load: 1.033
Part of the total run time spent waiting due to PP/PME imbalance: 2.1 %
        Core t (s) Wall t (s) (%)
 Time: 14806.100 154.231 9600.0
        (ns/day) (hour/ns)
Performance: 5.603 4.283
GROMACS reminds you: "Come on boys, Let's push it hard" (P.J. Harvey)
```

----End

Dual Nodes

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory and upload the test case file.

```
mkdir -p path/to/CASE
```

Step 3 Decompress the test case file.

```
tar xvf water_GMX50_bare.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd water-cut1.0_GMX50_bare/0768
```

Step 5 Create a **host** file.

```
vi host
hostname1
hostname2
```

NOTE

hostname1 and **hostname2** are the host names of the two nodes.

Step 6 Start the running.

```
mpirun --allow-run-as-root --hostfile host --mca btl ^openib -np 48 -N 24 -x OMP_NUM_THREADS=4 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH  gmx_mpi mdrun -dlb yes -v -nsteps 10000 -resethway -noconfout -pin on -ntomp 4 -s topol.tpr
```

Table 3-139 Parameter description

Parameter	Description
--hostfile	Node file to be used.
-np	Total number of running MPI processes.
-N	Number of processes running on each server.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

- If hyper-threading is not enabled, the number of MPI processes multiplied by the number of OMP threads must be less than or equal to the number of CPU cores. In this example, the **np** value of the dual nodes is 48, *N* indicates the number of processes on each node, the value of **OMP_NUM_THREADS** is 4, and the number of used CPU cores is 192 (2 x 24 x 4).
- The test cases are stored in the shared directory. Ensure that the **PATH** and **LD_LIBRARY_PATH** variables must be the same as those in **3.27.4 Configuring the Compilation Environment**.

View the value of **ns/day** under **Performance** in the **md.log** file. A larger value indicates higher performance.

The following is an example of the test result.

```
Part of the total run time spent waiting due to load imbalance: 1.3%.
Steps where the load balancing was limited by -rdd, -rcon and/or -dds: X 0 % Y 0 %
Average PME mesh/force load: 1.017
Part of the total run time spent waiting due to PP/PME imbalance: 2.2 %
       Core t (s) Wall t (s) (%)
 Time: 13667.657    71.196  19197.2
       (ns/day) (hour/ns)
Performance: 11.603 1.965
GROMACS reminds you: "Come on boys, Let's push it hard" (P.J. Harvey)
```

----End

3.28 LAMMPS 5 Jun 2019 Porting Guide (openEuler 21.03)

3.28.1 Introduction

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a classical molecular dynamics simulation code designed to run efficiently on parallel computers. LAMMPS uses MPI to implement multi-machine parallel computing. Its latest version supports GPU computing based on CUDA and OpenCL. It is an open source code, distributed freely under the terms of the GNU Public License (GPL).

For more information about LAMMPS, visit the **LAMMPS official website**.

Programming language: C++

Brief description: a classical molecular dynamics simulation code designed to run efficiently on parallel computers.

Open source license: GPL 2.0

Recommended Software Version

LAMMPS 5 Jun 2019

 Copyright © Huawei Technologies Co., Ltd.

3.28.2 Environment Requirements

Hardware Requirements

Table 3-140 lists the hardware requirements.

Table 3-140 Hardware requirements

Item	Description
CPU	Kunpeng 920 7260 processor

Software Requirements

Table 3-141 lists the software requirements.

Table 3-141 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://github.com/benjycui/bisheng/archive/refs/tags/bisheng@1.3.3.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
LAMMPS	5 Jun 2019	https://lammps.sandia.gov/tars/
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
Test cases	in.lj	lammps-stable_5Jun2019/bench

OS Requirements

Table 3-142 lists the OS requirements.

Table 3-142 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	Included in the OS image

3.28.3 Paths for Software Porting

This chapter lists software installation paths involved in the LAMMPS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-143 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	**/path/to/BISHENG**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/HMPI**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/FFTW**	Installation path of FFTW.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	**/path/to/LAMMPS**	Installation path of LAMMPS.	
5	**/path/to/CASE**	Path of LAMMPS test cases.	

3.28.4 Configuring the Compilation Environment

3.28.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-144 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing FFTW	See **3.28.4.2 Installing FFTW**.

 Copyright © Huawei Technologies Co., Ltd.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source  ~/.bashrc
```

----End

3.28.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Perform the compilation and installation.

```
make -j 96
make install
```

Step 6 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 7 Press **i** to enter the edit mode and run the following commands to load the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

Step 8 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.28.5 Obtaining the Source Code

Procedure

Step 1 Download the LAMMPS installation package.

https://lammps.sandia.gov/tars/

Step 2 Use an SFTP tool to upload the LAMMPS installation package to the */path/to/LAMMPS* directory on the server.

----End

3.28.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the LAMMPS installation package directory.

```
cd /path/to/LAMMPS
```

Step 3 Decompress the LAMMPS installation package.

```
tar -xvf lammps-5Jun2019.tar.gz
```

Step 4 Go to the **src** directory.

```
cd lammps-5Jun19/src
```

Step 5 Modify the **MAKE/Makefile.mpi** file.

1. Open the **MAKE/Makefile.mpi** file.
   ```
   vi MAKE/Makefile.mpi
   ```
2. Press **i** to enter the edit mode and modify the **CCFLAGS**, **LINKFLAGS**, **FFT_INC**, **FFT_PATH**, and **FFT_LIB** lines in the **MAKE/Makefile.mpi** file.
   ```
   CC = mpicxx
   CCFLAGS = -g -O3 -I/path/to/BISHENG/include/c++/v1 -L/path/to/BISHENG/lib -std=c++11 -stdlib=libc
   ++ -lc++ -lc++abi
   SHFLAGS = -fPIC
   DEPFLAGS = -M
   LINK = mpicxx
   LINKFLAGS =  -g -O -L/path/to/BISHENG/lib -std=c++11 -stdlib=libc++ -lc++ -lc++abi
   LIB =
   SIZE = size
   ......
   ARCHIVE = ar
   ARFLAGS = -rc
   SHLIBFLAGS = -shared
   FFT_INC = -DFFT_FFTW -I/path/to/FFTW/include
   FFT_PATH = -L/path/to/FFTW/lib
   FFT_LIB = -lfftw3
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation.

```
make yes-KSPACE
make yes-MPIIO
make yes-MOLECULE
make yes-MANYBODY
make yes-USER-REAXC
make mpi -j
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.28.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory.

```
mkdir -p /path/to/CASE
```

Step 3 Go to the working directory, and copy the test cases and binary files to the working directory.

```
cd /path/to/CASE
cp /path/to/LAMMPS/lammps-5Jun19/bench/in.lj ./
cp /path/to/LAMMPS/lammps-5Jun19/src/lmp_mpi ./
```

Step 4 Start the running.

- Run single-node commands on CentOS.

  ```
  mpirun --allow-run-as-root -np 96 --mca btl ^openib ./lmp_mpi -in in.lj >>test_OneNode.log
  ```

- Run single-node commands on openEuler.

  ```
  mpirun --allow-run-as-root -np 96 -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
  UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x
  UCX_BUILTIN_BCAST_ALGORITHM=3 -x UCX_BUILTIN_BARRIER_ALGORITHM=5 -x
  UCX_BUILTIN_ALLREDUCE_ALGORITHM=10 ./lmp_mpi -in in.lj >> ./test_OneNode.log
  ```

  Output the result to the **test_OneNode.log** file in the current directory and check the value of **Performance** (unit: timesteps/s). A larger value indicates higher performance.

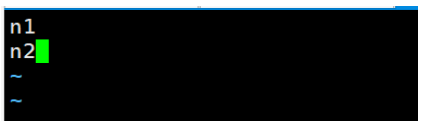
  The following is an example of the test result.

  ```
  Performance: 1134386.210 tau/day, 2625.894 timesteps/s
  99.7% CPU use with 96 MPI tasks x no OpenMP threads
  MPI task timing breakdown:
  Section |  min time  |  avg time  |  max time  |%varavg| %total
  ---------------------------------------------------------------
  Pair    | 0.021651   | 0.023624   | 0.025878   |   0.6 | 62.03
  Neigh   | 0.002734   | 0.0029155  | 0.0031491  |   0.2 |  7.66
  Comm    | 0.007533   | 0.010058   | 0.012192   |   1.1 | 26.41
  Output  | 5.6281e-05 | 0.00050681 | 0.00062442 |   0.0 |  1.33
  Modify  | 0.00056003 | 0.00061947 | 0.00070919 |   0.0 |  1.63
  Other   |            | 0.0003584  |            |       |  0.94
  ```

- Run dual-node commands on CentOS.

  ```
  mpirun --allow-run-as-root -np 192 -N 96 -x PATH=$PATH -x LD_LIBRARY_PATH=
  $LD_LIBRARY_PATH -machinefile machinefile --mca btl ^openib ./lmp_mpi -in in.lj >> ./
  test_TwoNodes.log
  ```

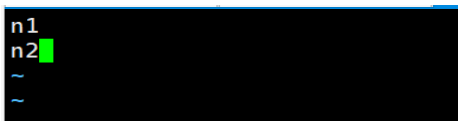
  Add the host names (for example, **n1** and **n2**) of the two specified compute nodes to the **machinefile** file, as shown in the following figure:

- Run dual-node commands on openEuler.

  ```
  mpirun --allow-run-as-root -np 192 -N 96 -machinefile machinefile -x PATH=$PATH -x
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH -mca pml ucx -mca btl ^vader,tcp,openib,uct --bind-to
  core --rank-by core ./lmp_mpi -in in.lj >> ./test_TwoNodes.log
  ```

  Add the host names (for example, **n1** and **n2**) of the two specified compute nodes to the **machinefile** file, as shown in the following figure:

 Copyright © Huawei Technologies Co., Ltd.

Output the result to the **test_TwoNodes.log** file in the current directory and check the value of **Performance** (unit: timesteps/s). A larger value indicates higher performance.

The following is an example of the test result.

```
Performance: 1605508.300 tau/day, 3716.454 timesteps/s
91.0% CPU use with 192 MPI tasks x no OpenMP threads
MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.01035    | 0.01174    | 0.013347   |   0.6 | 43.63
Neigh   | 0.0013662  | 0.0014931  | 0.0016205  |   0.1 |  5.55
Comm    | 0.01112    | 0.012935   | 0.014469   |   0.6 | 48.07
Output  | 7.3471e-05 | 0.00010074 | 0.00017455 |   0.0 |  0.37
Modify  | 0.00023517 | 0.00032949 | 0.00040742 |   0.0 |  1.22
Other   |            | 0.0003095  |            |       |  1.15
```

Table 3-145 Parameter description

Parameter	Description
-np	Total number of running MPI processes.
-N	Number of processes running on each server.
-machinefile	Name of the node to be used.

NOTE

- The dual-node test cases run in the shared directory. If the **PATH** and **LD_LIBRARY_PATH** environment variables have been configured in **3.28.4 Configuring the Compilation Environment**, do not need to configure them again.
- If hyper-threading is not enabled, the **np** value must be less than or equal to the number of nodes multiplied by the number of CPU cores on each node.
- **n1** and **n2** are the host names.

----End

3.29 NWChem 6.8.1 Porting Guide (openEuler 21.03)

3.29.1 Introduction

NWChem is computational chemistry software running on high-performance parallel supercomputers and common workstation clusters. NWChem uses standard quantum mechanics to describe electron wave functions or densities, to calculate the properties of molecules and periodic systems, and to simulate classical molecular dynamics and free energy.

For more information, visit the **official NWChem website**.

Programming language: Python

Brief description: scalable computational chemistry tool.

Open source license: ECL 2.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

NWChem 6.8.1

3.29.2 Environment Requirements

Hardware Requirements

Table 3-146 lists the hardware requirements.

Table 3-146 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-147 lists the software requirements.

Table 3-147 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://www.hikunpeng.com/en/developer/devkit/compiler?data=bisheng
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi
NWChem	v6.8.1	https://codeload.github.com/NWChemgit/NWChem/tar.gz/6.8.1-release
GlobalArray	5.6.5	https://github.com/edoapra/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz
Test cases	siosi5.nw	Provided by the software.

OS Requirements

Table 3-148 lists the OS requirements.

Table 3-148 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

3.29.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NWChem software porting.

Table 3-149 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	**/path/to/hypermpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
2	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
3	**/path/to/NWCHEM**	Installation path of NWChem.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.29.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-150 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hypermpi/hmpi/include:$INCLUDE
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

3.29.5 Obtaining the Source Code

Procedure

Step 1 Download the NWChem installation package.

https://codeload.github.com/NWChemgit/NWChem/tar.gz/6.8.1-release

Step 2 Use SFTP to upload the NWChem installation package to the */path/to/NWCHEM* directory on the server.

----End

3.29.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 3 Decompress the NWChem installation package.

```
cd /path/to/NWCHEM
tar -zxvf nwchem-6.8.1-release.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd nwchem-6.8.1-release
```

Step 5 Decompress the GlobalArray installation package to the **src/tools** directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd src/tools
wget https://github.com/edoapra/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz
tar -zxvf ga-5.6.5.tar.gz
```

Step 6 Perform the configuration.

```
export NWCHEM_TOP=/path/to/NWCHEM/nwchem-6.8.1-release
export NWCHEM_TARGET=LINUX64
export NWCHEM_MODULES=all
export USE_MPI=y
export USE_INTERNALBLAS=y
```

Step 7 Perform the compilation and installation.

```
cd /path/to/NWCHEM/nwchem-6.8.1-release/src
make nwchem_config
make FC=flang >& make.log
```

Step 8 Add the environment variables.

```
export PATH=/path/to/NWCHEM/nwchem-6.8.1-release/bin/LINUX64:$PATH
```

----End

3.29.7 Running and Verification

NOTE

- Determine the number of running processes based on the node configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the nodes.
  - Enable SSH password-free login between the nodes.
  - Disable the firewall and ensure that clients and servers are not disconnected.
- NWChem provides multiple test cases. Select one that is suitable for your servers. This section uses **siosi5.nw** as an example.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test case directory.

```
cd /path/to/NWCHEM/nwchem-6.8.1-release/web/benchmarks/dft
```

Step 3 Create a **hostfile** file.

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   Node1
   Node2
   ```

   NOTE

   Node1 and **Node2** are the names of the nodes where the code is executed. Change them based on your requirements and ensure that the nodes can ping each other. Enable SSH password-free login between the nodes.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Perform the computing.

- Single node
  ```
  mpirun --allow-run-as-root -N 8 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH -mca btl ^openib nwchem siosi5.nw -v 2>&1 | tee NWChem_examples.log
  ```

 Copyright © Huawei Technologies Co., Ltd.

The following figure shows the command output.

- Dual nodes
  mpirun --allow-run-as-root -N *8* **-x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH -mca btl ^openib --hostfile** *hostfile* **nwchem siosi5.nw -v 2>&1 | tee NWChem_examples.log**

  The following figure shows the command output.

  NOTE

  - **hostfile** indicates the list of node names to be used. In this example, two nodes are used.
  - **8** indicates the number of processes running on each node. You can change the value based on the node configuration.
  - Check the value of **Total times cpu** (unit: s) in the **NWChem_examples.log** file. A smaller value indicates better performance.

----End

3.30 VASP 5.4.4 Porting Guide (openEuler 21.03)

3.30.1 Introduction

Vienna Ab-initio Simulation Package (VASP) is a software package developed by the Hafner team of the University of Vienna for electronic structure calculation and quantum mechanics-molecular dynamics simulation. VASP is one of the most popular commercial software used in material simulation and computational physical science study.

For more information about VASP, visit the **VASP official website**.

Programming language: Fortran

Brief description: electronic structure calculation and quantum mechanics-molecular dynamics simulation.

Recommended Software Version

VASP 5.4.4

3.30.2 Environment Requirements

Hardware Requirements

Table 3-151 lists the hardware requirements.

Table 3-151 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-152 lists the software requirements.

Table 3-152 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
VASP	5.4.4	**https://www.vasp.at/**
FFTW	3.3.8	**http://www.fftw.org/download.html**
OpenBLAS	0.3.6	**https://github.com/xianyi/OpenBLAS/tree/v0.3.6**
ScaLAPACK	2.0.2	**http://www.netlib.org/scalapack/**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
VTST	179	http://theory.cm.utexas.edu/code/vtstcode-179.tgz
Wannier90	1.2	http://www.wannier.org/code/wannier90-1.2.tar.gz
Test cases	vasp.Hg	-

OS Requirements

Table 3-153 lists the OS requirements.

Table 3-153 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

3.30.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VASP software porting.

Table 3-154 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/hypermpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/OPENBLAS**	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	**/path/to/SCALAPACK**	Installation path of ScaLAPACK.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
5	**/path/to/FFTW**	Installation path of FFTW.	
6	**/path/to/VASP**	Installation path of VASP	
7	**/path/to/CASE**	Test path of VASP.	

3.30.4 Configuring the Compilation Environment

3.30.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-155 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing OpenBLAS	See **3.30.4.2 Installing OpenBLAS**.
4	Installing ScaLAPACK	See **3.30.4.3 Installing ScaLAPACK**.
5	Installing FFTW	See **3.30.4.4 Installing FFTW**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hypermpi/hmpi/include:$INCLUDE
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

3.30.4.2 Installing OpenBLAS

Before configuring the compilation environment, install the BiSheng Compiler and Hyper MPI, and import the BiSheng Compiler and Hyper MPI environment variables. If the BiSheng Compiler and Hyper MPI have been installed in the environment, directly import their environment variables.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
unzip OpenBLAS-0.3.6.zip
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Perform the configuration.

```
export CC=clang
export CXX=clang++
export FC=flang
```

Step 5 Perform the compilation and installation.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.30.4.3 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ScaLAPACK installation package.

```
tar -xvf scalapack-2.0.2.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd scalapack-2.0.2
```

Step 4 Create an **SLmake.inc** file.

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Edit the **SLmake.inc** file.

1. Open the test case file.
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the insert mode.

   Add the following content to line 18:

   ```
   $(LIBS) += -fuse-ld=lld
   ```

   Modify the content in lines 58 and 59 in the **SLmake.inc** file.

   ```
   BLASLIB      =/path/to/OPENBLAS/lib/libopenblas.a
   LAPACKLIB    =/path/to/OPENBLAS/lib/libopenblas.a
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
```

Step 7 Copy the generated link library to the installation directory.

```
cp libscalapack.a /path/to/SCALAPACK
```

----End

3.30.4.4 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

3.30.5 Obtaining the Source Code

Procedure

Step 1 Download the VASP, VTSTcode, Wannier90 1.2, and VSAPsol installation packages.

https://www.vasp.at/

http://theory.cm.utexas.edu/code/vtstcode-179.tgz

http://www.wannier.org/code/wannier90-1.2.tar.gz

https://github.com/henniggroup/VASPsol/archive/refs/tags/V1.0.tar.gz

Step 2 Use an SFTP tool to upload the VASP installation package to the */path/to/VASP* directory on the server.

----End

3.30.6 Compilation and Installation

Basic Installation Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the VASP installation package.

```
cd /path/to/VASP
tar -xvf vasp.5.4.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd vasp.5.4.4
```

Step 4 If the environment configuration has been loaded, skip this step. Configure the compiler, Hyper MPI, and precompilation environment based on the actual installation paths. For example:

```
export INCLUDE=/path/to/hypermpi/hmpi/include
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:/path/to/SCALAPACK:/path/to/FFTW/lib:
$LD_LIBRARY_PATH
```

NOTE

Set the environment variables of the BiSheng Compiler and Hyper MPI before you configure the compilation environment.

Step 5 Create and edit the **makefile.include** file.

1. Create a **makefile.include** file.
   ```
   vi makefile.include
   ```
2. Press **i** to enter the insert mode and add the following content to the **makefile.include** file:
   ```
   ##Precompiler options
   CPP_OPTIONS= -DHOST=\"LinuxGNU\" \
                -DMPI -DMPI_BLOCK=8000 \
                -Duse_collective \
                -DscaLAPACK \
   ```

```
            -DCACHE_SIZE=5000 \
            -Davoidalloc \
            -Duse_bse_te \
            -Dtbdyn \
            -Duse_shmem
CPP        = flang -E -P -C -w $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)
FC         = mpif90
FCL        = mpif90
FREE       = -ffree-form -ffree-line-length-none
FFLAGS     = -w
OFLAG      = -O1 -ffp-contract=fast  -ffpe-trap=invalid,zero,overflow
OFLAG_IN   = $(OFLAG)
DEBUG      = -O0
LLIBS      = -L/path/to/SCALAPACK -lscalapack -L/path/to/OPENBLAS/lib -lopenblas
FFTW       ?= /path/to/FFTW
LLIBS      += -L$(FFTW)/lib -lfftw3
INCS       = -I$(FFTW)/include
OBJECTS    = fftmpiw.o fftmpi_map.o  fftw3d.o  fft3dlib.o
OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o
# For what used to be vasp.5.lib
CPP_LIB    = $(CPP)
FC_LIB     = $(FC)
CC_LIB     = clang
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
FREE_LIB   = $(FREE)
OBJECTS_LIB= linpack_double.o getshmem.o
# For the parser library
CXX_PARS   = clang++
LIBS       += parser
LLIBS      += -Lparser -lparser -lstdc++
# Normally no need to change this
SRCDIR     = ../../src
BINDIR     = ../../bin
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make std
```

To generate **vasp_gam** and **vasp_ncl**, run the following command:

```
make all
```

----End

(Optional) Extended Installation Procedure

Install the VTST transition state tool.

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the VTSTcode installation package to the directory where VASP is installed.

Step 3 Decompress the VTSTcode installation package.

```
cd /path/to/VASP
tar -xvf vtstcode-179.tgz
```

Step 4 Back up the **chain.F** file in the **src** directory of the VASP installation package.

```
cp vasp.5.4.4/src/chain.F vasp.5.4.4/src/chain.F_bak
```

Step 5 Copy files of the **vtstcode-179** directory to the **src** directory.

```
cp vtstcode-179/* vasp.5.4.4/src/
```

Step 6 Modify the **src/main.F** file.

 Copyright © Huawei Technologies Co., Ltd.

1. Open the test case file.
   vi vasp.5.4.4/src/main.F
2. Press **i** to enter the insert mode and change the statements in lines 3146 and 3147 in the **src/main.F** file to the following:
   CALL CHAIN_FORCE(T_INFO%NIONS,DYN%POSION,TOTEN,TIFOR, & TSIF,LATT_CUR%A,LATT_CUR%B,IO%IU6)

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the compilation configuration file.

1. Open the test case file.
   vi vasp.5.4.4/src/.objects
2. Press **i** to enter the insert mode and add the following content between line 71 and line 72:
   bfgs.o dynmat.o instanton.o lbfgs.o sd.o cg.o dimer.o bbm.o \
   fire.o lanczos.o neb.o qm.o opt.o \

   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation and installation.

make std

To generate **vasp_gam** and **vasp_ncl**, run the following command:

make all

----End

(Optional) Extended Installation Procedure

Installing the VASPsol Tool

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download and decompress the VASPsol installation package.

cd */path/to/VASP*
wget https://github.com/henniggroup/VASPsol/archive/refs/tags/V1.0.tar.gz
tar -xvf V1.0.tar.gz

Step 3 Copy files of the **VASPsol** directory to the **src** directory.

cp VASPsol-1.0/src/solvation.F vasp.5.4.4/src/

Step 4 Modify the **makefile.include** file in step **Step 5** of **Basic Installation Procedure**.

1. Open the **makefile.include** file.
   vi makefile.include
2. Press **i** to enter the insert mode and add **-Dsol_compat** to **CPP_OPTIONS**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the compilation and installation.

make std

 Copyright © Huawei Technologies Co., Ltd.

To generate **vasp_gam** and **vasp_ncl**, run the following command:

```
make all
```

----End

Procedure for Installing the Wannier90 Interface

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the **wannier90-1.2** installation package in the directory where VASP is installed.

```
cd /path/to/VASP
tar -xvf wannier90-1.2.tar.gz
```

Step 3 Replace **make.sys** with **make.sys.gfort**.

```
cd wannier90-1.2
cp config/make.sys.gfort make.sys
```

Step 4 Modify the **make.sys** file.

1. Open the test case file.
   ```
   vi make.sys
   ```
2. Press **i** to enter the insert mode and modify the content in line 4.
   ```
   F90 = flang
   ```
   Add the following content to the end of the **make.sys** file.
   ```
   LIBS = -L/path/to/SCALAPACK -L/path/to/OPENBLAS/lib -lscalapack -lopenblas
   ```
   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the installation.

```
make wannier lib test
```

The **libwannnier.a** function library is generated, which needs to be added to the VASP installation and compilation.

Step 6 Modify the **makefile.include** file.

```
cd /path/to/VASP/vasp.5.4.4
vi makefile.include
```

Step 7 Press **i** to enter the insert mode and modify the **makefile.include** file.

Modify line 10 as follows:

```
-Duse_shmem -Dtbdyn -DVASP2WANNIER90
```

Add the following content between line 18 and line 19.

```
WANNIER90 = /path/to/VASP/wannier90-1.2/libwannier.a
```

Modify line 20: Add Wannier interface parameters to LLIBS.

```
LLIBS = $(WANNIER90) -L/path/to/SCALAPACK -lscalapack -L/path/to/OPENBLAS/lib -lopenblas
```

NOTE

After performing **Step 6**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

 Copyright © Huawei Technologies Co., Ltd.

Step 8 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Perform the compilation.

```
make std
```

----End

3.30.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a working directory and go to it.

```
mkdir -p /path/to/CASE
cd /path/to/CASE
```

Step 3 Prepare the test case file **vasp-small.gz** and copy it to the **/path/to/CASE** directory.

Step 4 Deploy the test case.

```
cd /path/to/CASE
tar -vxf vasp-small.gz
cd vasp.Hg
mv IN-long INCAR
```

Step 5 Set the environment variables.

```
export PATH=/path/to/VASP/vasp.5.4.4/bin:$PATH
```

Step 6 Create a **host** file.

```
vi hostfile
```

Step 7 Press **i** to enter the insert mode and add the following content:

```
node1
node2
```

NOTE

node1 and **node2** are the names of the nodes where the code is executed. Change them based on your requirements and ensure that the nodes can ping each other. Enable SSH password-free login between the nodes.

Step 8 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

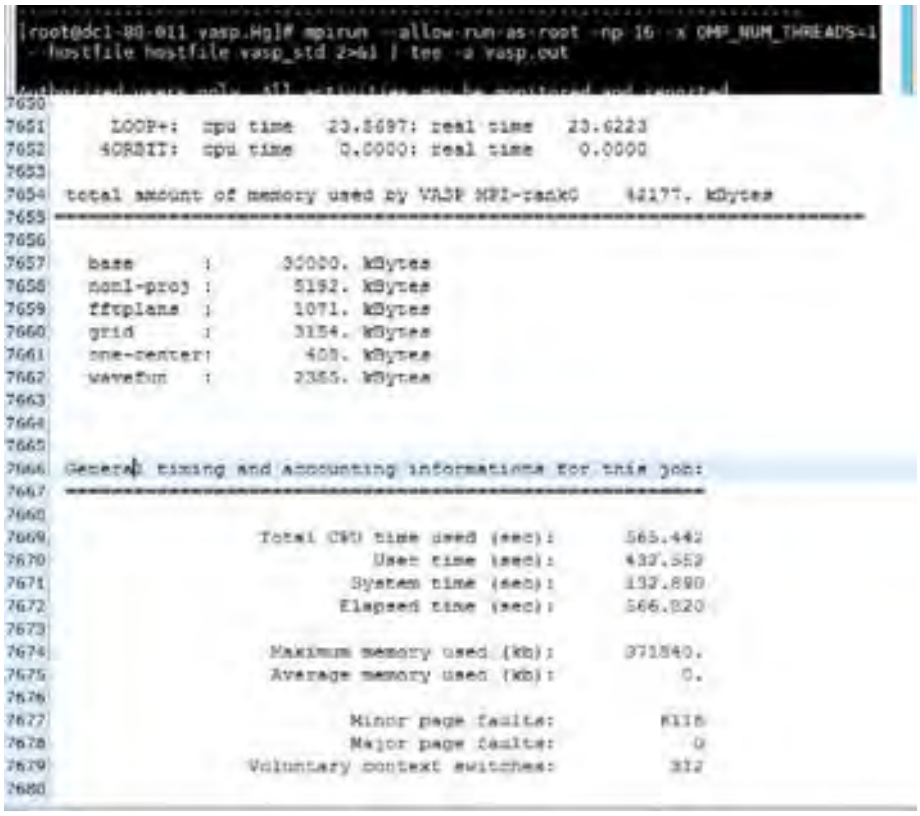
Step 9 Start the VASP program.

- Multiple nodes

  ```
  mpirun --allow-run-as-root -np 16 -N 8 -x OMP_NUM_THREADS=1 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH --hostfile hostfile vasp_std 2>&1 | tee -a vasp.out
  ```

  If the information shown in the following figure is displayed at the end of the **OUTCAR** log generated by VASP, the VASP program stops properly.

 Copyright © Huawei Technologies Co., Ltd.



- Single node
  mpirun --allow-run-as-root -np *16* **-x OMP_NUM_THREADS=1 vasp_std 2>&1 | tee -a vasp.out**

  If the information shown in the following figure is displayed at the end of the **OUTCAR** log generated by VASP, the VASP program stops properly.

NOTE

- The **-np** parameter indicates the total number of processes. The **-N** parameter indicates the number of processes running on each node. Here, two nodes are used as an example. Change the setting based on the actual host configuration. Alternatively, you do not need to specify the two parameters and the maximum number of processes will be used.
- The **-x OMP_NUM_THREADS** parameter indicates the number of threads to be used.
- Check the value of **real time** (unit: second) in the **OUTCAR** log. A smaller value indicates higher performance.

----End

3.31 Wannier90 3.1.0 Porting Guide (openEuler 21.03)

3.31.1 Introduction

Wannier90 is a computer package written in Fortran90. It is used to obtain maximally-localised Wannier functions in order to calculate bandstructures, Fermi surfaces, dielectric properties, sparse Hamiltonians and many things besides.

Many electronic structure codes have an interface to Wannier90, including Quantum ESPRESSO, Abinit, VASP, Siesta, Wien2k, Fleur, OpenMX, and GPAW. There are several post-processing codes that are able to use the output of Wannier90 for further analysis and calculation.

For more information, visit the **official Wannier90 website**.

Programming language: Fortran

Brief description: Wannier90 is a computer package, written in Fortran90, used to obtain maximally-localised Wannier functions.

Open source license: GPL 2.0

Recommended Software Version

Wannier90 3.1.0

3.31.2 Environment Requirements

Hardware Requirements

Table 3-156 lists the hardware requirements.

Table 3-156 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-157 lists the software requirements.

Table 3-157 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://repo.huaweicloud.com/hmpi/HMPI_1.0.2_aarch64.tar.gz**
Wannier90	v3.1.0	**https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0**
OpenBLAS	v0.3.6	**https://github.com/xianyi/OpenBLAS/tree/v0.3.6**

OS Requirements

Table 3-158 lists the OS requirements.

Table 3-158 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://repo.openeuler.org/openEuler-21.03/**
Kernel	4.14.0-115	Included in the OS image

3.31.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Wannier90 software porting.

Table 3-159 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
2	**/path/to/hypermpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	**/path/to/WANNIER90**	Installation path of Wannier90.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	**/path/to/OPENBLAS**	Installation path of OpenBLAS.	

3.31.4 Configuring the Compilation Environment

3.31.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-160 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing OpenBLAS dependencies	See **3.31.4.2 Installing OpenBLAS Dependencies**.
4	Installing Gnuplot	See **3.31.4.3 Installing Gnuplot**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

 Copyright © Huawei Technologies Co., Ltd.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hypermpi/hmpi/include
export PATH=/path/to/hypermpi/hmpi/bin:/path/to/hypermpi/ucx/bin:$PATH
export OPAL_PREFIX=/path/to/hypermpi/hmpi/
export LD_LIBRARY_PATH=/path/to/hypermpi/hmpi/lib:/path/to/hypermpi/ucx/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

3.31.4.2 Installing OpenBLAS Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the OpenBLAS installation directory.

```
cd /path/to/OPENBLAS
```

Step 3 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 5 Perform the configuration.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 6 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

Step 7 Perform the installation.

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

3.31.4.3 Installing Gnuplot

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

```
yum install gnuplot
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Set the Gnuplot environment variables.

```
export GNUTERM=dumb
export GNUPLOT=/usr
export PATH=/usr/bin:$PATH
export MANPATH=/usr/share/man/man1:$MANPATH
```

----End

3.31.5 Obtaining the Source Code

Procedure

Step 1 Download the Wannier90 installation package.

https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0

Step 2 Use an SFTP tool to upload the Wannier90 installation package to the */path/to/WANNIER90* directory on the server.

----End

3.31.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Wannier90 installation package directory.

```
cd /path/to/WANNIER90
```

Step 3 Decompress the Wannier90 installation package.

```
tar -xvf wannier90-3.1.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd wannier90-3.1.0
```

Step 5 Build Wannier90.

```
cp ./config/make.inc.gfort ./make.inc
```

Step 6 Modify the **make.inc** file.

1. Open the **make.inc** file.
   ```
   vi make.inc
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   - Modify line 4 as follows:
     ```
     F90=flang
     ```
   - Uncomment lines 7 and 8, and modify line 8 as follows:
     ```
     MPIF90 = mpif90
     ```
   - Uncomment line 21 and modify it as follows:
     ```
     LIBDIR=/path/to/OPENBLAS/lib
     ```
   - Modify line 37 as follows:
     ```
     LIBS = -L$(LIBDIR) -lopenblas
     ```

     NOTE

     After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Compile and install Wannier90.

```
make
make install
```

NOTE

After the compilation and installation are complete, two executable files **postw90.x** and **wannier90.x** are generated in the **wannier90-3.1.0** directory.

----End

3.31.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/WANNIER90/wannier90-3.1.0/examples/example03
```

Step 3 Perform the test.

- Single node

  ```
  time -p mpirun -np 8 --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x silicon
  ```

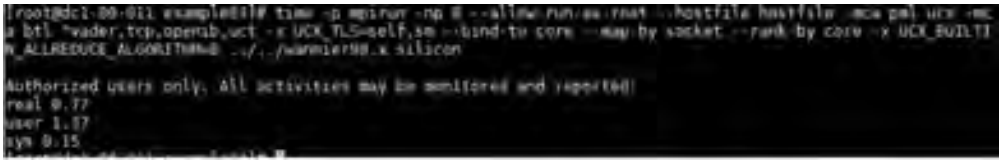
- Multiple nodes

  ```
  time -p mpirun -np 16 -N 8 -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH --allow-run-as-root --hostfile hostfile -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x silicon
  ```

  NOTE

  - In the preceding command, **-np** indicates the total number of processes and **-N** indicates the number of processes running on each node. The previous command uses dual nodes as an example. Modify the command based on the actual node configuration.
  - For the multi-node configuration, **--hostfile** is the list of node names. Create a host file in the **example03** directory and write the host names of the nodes into the file.
  - **-x UCX_TLS=self,sm** is subject to the network in the environment.
  - Ensure that each node has the same software paths and environment variables. The required software includes all the preceding software that needs to be installed.
  - Ensure that each node has the same test case directory.
  - Ensure that between the nodes, password-free login is enabled and the firewall is disabled.

  After the command is successfully executed, a **silicon.wout** file is generated. The following figure shows an example of the command output.



 Copyright © Huawei Technologies Co., Ltd.

Step 4 Modify the **silicon.win** file.

1. Open the **silicon.win** file.
   ```
   vi silicon.win
   ```
2. Press **i** to go to the edit mode. Add the following content below line 2:
   ```
   restart         = plot
   bands_plot      = true
   ```

   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the test command of **Step 3** again. After the command is successfully executed, the **silicon_band.gnu** and **silicon_band.dat** files are generated in the current directory. The operation of displaying graphs in this step is only a supplement to the program running, and does not affect the execution or performance of the program.

```
time -p mpirun -np 8 --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x
UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x silicon
```

NOTE

- **-np** indicates the total number of processes that are used. Set it based on the actual node configuration.
- **-x UCX_TLS=self,sm** is subject to the network in the environment.

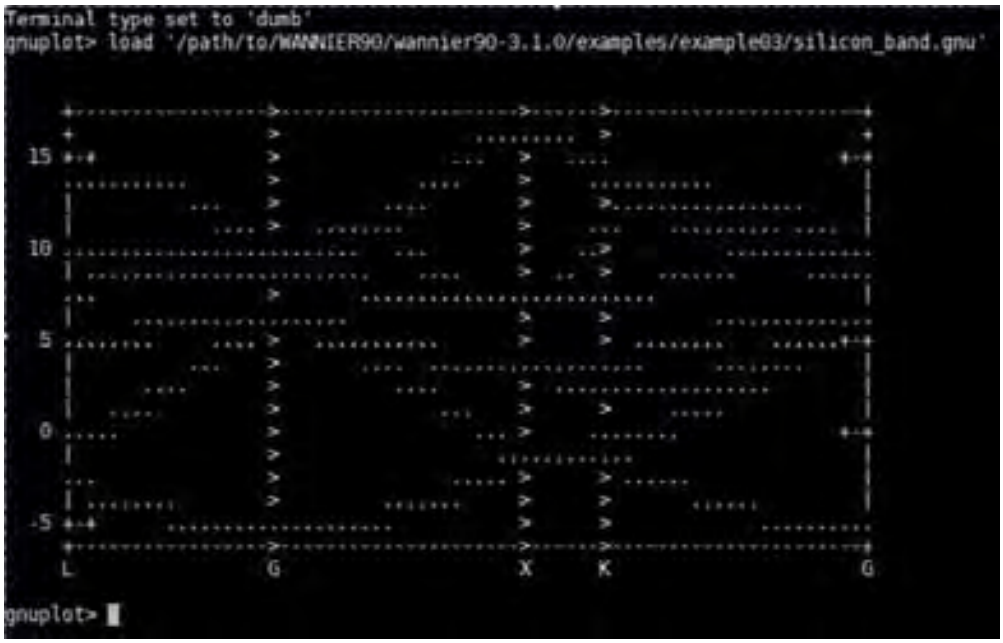
Step 6 Start the Gnuplot graphing utility to draw a graph.

```
gnuplot
```

The following information is displayed:

```
load '/path/to/WANNIER90/wannier90-3.1.0/examples/example03/silicon_band.gnu'
```

The following graph is displayed:



----End

3.32 QE 6.4.1 Porting Guide (openEuler 21.03)

3.32.1 Introduction

Quantum ESPRESSO (QE) is a software suite for ab initio quantum chemistry methods for electronic-structure calculation and materials modeling, distributed for free and as free software free under the GNU General Public License. It is based on density functional theory (DFT), plane wave (PW) basis sets and pseudopotentials (including norm conservation and ultrasoft). QE, written mainly in Fortran-90 with some parts in C or in Fortran-77, was built out of the merging and re-engineering of different independently-developed core packages, plus a set of packages designed to be inter-operable with the core components. The basic packages include Pwscf, which solves the self-consistent Kohn and Sham equations and obtains for a periodic solid, CP, which carries out Car-Parrinello molecular dynamics, and PostProc, which allows data analysis and plotting. Regarding the additional packages, it is noteworthy to point out atomic for the pseudopotential generation, PHonon package, which implements density-functional perturbation theory (DFPT) for the calculation of second- and third-order derivatives of the energy with respect to atomic displacements and NEB.

For more information, visit the **official QE website**.

Programming language: Fortran

Brief description: A software suite of ab initio quantum chemistry methods for electronic structure calculation and materials modeling.

Open source license: GPL 2.0

Recommended Software Version

QE 6.4.1

 Copyright © Huawei Technologies Co., Ltd.

3.32.2 Environment Requirements

Hardware Requirements

Table 3-161 lists the hardware requirements.

Table 3-161 Hardware requirements

Item	Description
CPU	Kunpeng 920 7260 processor

Software Requirements

Table 3-162 lists the software requirements.

Table 3-162 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.3	https://github.com/benjycui/bisheng/archive/refs/tags/bisheng@1.3.3.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
QE	6.4.1	https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.6.tar.gz
ScaLAPACK	2.0.2	http://www.netlib.org/scalapack/scalapack-2.0.1.tgz
Test cases	AUSURF112	https://github.com/QEF/benchmarks/tree/master/AUSURF112

OS Requirements

Table 3-163 lists the OS requirements.

Table 3-163 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.19	https://gitee.com/openeuler/kernel/tree/kernel-4.19/

 Copyright © Huawei Technologies Co., Ltd.

3.32.3 Paths for Software Porting

This chapter lists software installation paths involved in the QE software porting.

Table 3-164 Paths for software porting

No.	Software Installation Path	Usage	Description
1	**/path/to/bisheng**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	**/path/to/hmpi**	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
2	**/path/to/OpenBLAS**	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	**/path/to/SCALAPACK**	Installation path of ScaLAPACK.	
4	**/path/to/QE**	Installation path of QE.	
5	**/path/to/CASE**	Test path of QE.	

3.32.4 Configuring the Compilation Environment

3.32.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-165 Configuration process

No.	Configuration Item	Description
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing OpenBLAS	See **3.32.4.2 Installing OpenBLAS**.
4	Installing ScaLAPACK	See **3.32.4.3 Installing ScaLAPACK**.

3.32.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Set the environment variables.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 5 Compile and install OpenBLAS.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Set the OpenBLAS environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.32.4.3 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ScaLAPACK installation package.

```
tar xf scalapack-2.0.2.tgz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the directory generated after the decompression.

```
cd scalapack-2.0.2
```

Step 4 Create an **SLmake.inc** file.

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Edit the **SLmake.inc** file.

1. Open the **SLmake.inc** file.
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the insert mode and edit lines 58 and 59 of the file.
   ```
   BLASLIB = /path/to/OpenBLAS/lib/libopenblas.a
   LAPACKLIB = /path/to/OpenBLAS/lib/liblopenblas.a
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Compile and install the software and copy the compiled file.

```
make
cp libscalapack.a /path/to/SCALAPACK/
```

Step 7 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/SCALAPACK:$LD_LIBRARY_PATH
```

----End

3.32.5 Obtaining the Source Code

Procedure

Step 1 Download the QE installation package.

https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz

Step 2 Use SFTP to upload the QE installation package to the */path/to/QE/* directory on the server.

----End

3.32.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the QE installation package directory.

```
cd /path/to/QE
```

Step 3 Decompress the QE installation package.

```
tar -xvf qe-6.4.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd q-e-qe-6.4.1/
```

Step 5 Configure the dependency package environment.

```
export BLAS_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export LAPACK_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export SCALAPACK_LIBS="-L/path/to/SCALAPACK -lscalapack"
export DEFS="__PGI"
```

Step 6 Perform configuration.

```
./configure --prefix=/opt/software/install/QE --enable-openmp ARCH=arm F77=flang F90=flang
FC=flang CC=clang CFLAGS="-O3 -g -D__PGI -mcpu=native" FFLAGS="-O3 -g -D__PGI -mcpu=native"
LDFLAGS="$LDFLAGS" MPIF90=mpifort MPIF77=mpifort CC=mpicc
```

Step 7 Perform the compilation.

```
make -j 2 pwall
make install
```

Step 8 Declare the environment variables.

```
export PATH=/path/to/QE/bin:$PATH
```

----End

3.32.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create the working directory.

```
mkdir -p path/to/CASE
```

Step 3 Download the test case and use SFTP to upload the test case to the */path/to/CASE* directory on the server.

https://github.com/QEF/benchmarks/tree/master

Step 4 Decompress the test case package and copy the required files to the */path/to/CASE* directory.

```
unzip benchmarks-master.zip
cp benchmarks-master/AUSURF112/* ./
```

Step 5 Create a **hostfile** file.

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   n1
   n2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 On *node 2*, set the environment variables.

1. Open the file.
   ```
   vi /root/.bashrc
   ```
2. Press **i** to enter the insert mode, and add the following content to the end of the **.bashrc** file:
   ```
   export PATH=/path/to/bisheng/bin:/path/to/hmpi/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/bisheng/lib:/path/to/hmpi/lib:$LD_LIBRARY_PATH
   export PATH=/path/to/QE/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 On *node 1*, start the QE program.

 Copyright © Huawei Technologies Co., Ltd.

```
mpirun --allow-run-as-root -np 192 -N 96 -x OMP_NUM_THREADS=1 -x PATH=$PATH -x
LD_LIBRARY_PATH=$LD_LIBRARY_PATH -hostfile hostfile --mca btl ^openib pw.x -input ./ausurf.in
2>&1 | tee -a qe.log
```

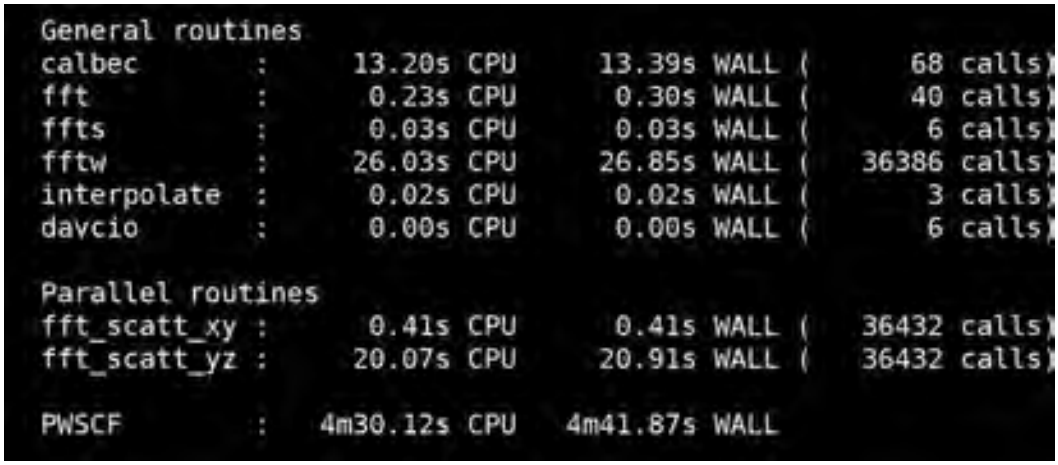
Table 3-166 Parameter description

Option	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
-x OMP_NUM_THREADS=1	Number of threads in each process. Set the parameter to **1**.
-hostfile	Name of the node to be used.

If the information shown in **Figure 3-16** is displayed at the end of the **qe.log** file generated by QE, the QE program ends properly.

Check the value (in the unit of **s**) of **WALL** in the **qe.log** file. A higher value indicates lower performance. **Figure 3-16** shows the output result.

Figure 3-16 Example result



----End

3.33 MOOSE 1.0.0 Porting Guide (openEuler 21.03)

3.33.1 Introduction

Multiphysics Object Oriented Simulation Environment (MOOSE) is an object-oriented finite element framework for the development of tightly coupled multiphysics solvers. MOOSE makes use of the PETSc non-linear solver package and libMesh to provide the finite element discretization.

 Copyright © Huawei Technologies Co., Ltd.

MOOSE features high parallelism (greater than 100,000 cores), complete multiphysics, and the use of both continuous and discrete Galerkin methods.

For more information, visit the **official MOOSE website**.

Programming languages: C/C++

One sentence description: a piece of object-oriented finite element multiphysics environment simulation software.

Open source license: GPL 2.0

Recommended Software Version

MOOSE 1.0.0

3.33.2 Environment Requirements

Hardware Requirements

Table 3-167 lists the hardware requirements.

Table 3-167 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-168 lists the software requirements.

Table 3-168 Software requirements

Item	Version	Download URL
MOOSE	1.0.0	**https://github.com/idaholab/moose/releases/tag/v1.0.0**
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**
Hyper MPI	1.1.0	**https://www.hikunpeng.com/en/developer/hpc/hypermpi**
CMake	3.21.2	**https://github.com/Kitware/CMake/releases/download/v3.21.2/cmake-3.21.2.tar.gz**
Python	3.8.2	**https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz**
PETSC	3.13.2	**https://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-3.13.2.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 3-169 lists the OS requirements.

Table 3-169 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

3.33.3 Paths for Software Porting

This chapter describes the software installation paths involved in the MOOSE software porting.

Table 3-170 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/compiler	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/hypermpi	Installation path of Hyper MPI.	See **Hyper MPI 1.1.0 Operation Guide**.
3	/path/to/MOOSE	Installation path of MOOSE.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/CMAKE	Installation path of CMake.	
5	/path/to/PYTHON	Installation path of Python.	

3.33.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

3.33.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-171 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing CMake	See **3.33.4.2 Installing CMake**.
4	Installing Python	See **3.33.4.3 Installing Python**.

3.33.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the CMake installation directory.

```
cd /path/to/CMAKE
```

Step 3 Decompress the CMake installation package.

```
tar -xvf cmake-3.21.2-linux-aarch64.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cmake-3.21.2-linux-aarch64
```

Step 5 Import the environment variables.

```
export PATH=/path/to/CMAKE/cmake-3.21.2-linux-aarch64/bin:$PATH
```

Step 6 Check whether CMake is installed.

```
cmake -version
```

If the following information is displayed, the installation is successful:

```
[root@dc1-80-011 petsc]# cmake -version
cmake version 3.21.2

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.33.4.3 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Python installation package.

```
cd /path/to/PYTHON
tar -xvf Python-3.8.2.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd Python-3.8.2
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/PYTHON
```

Step 5 Perform the compilation and installation.

```
make
make install
```

Step 6 Back up Python.

```
mv /usr/bin/python /usr/bin/python-bak
```

Step 7 Create a soft link for Python.

```
ln -s /path/to/PYTHON/bin/python3.8 /usr/bin/python
```

----End

3.33.5 Obtaining the Source Code

Procedure

Step 1 Configure git.

```
git config --global http."sslVerify" false
```

Step 2 Download the MOOSE source code.

```
git clone https://github.com/idaholab/moose.git
```

----End

3.33.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export CC=mpicc
export CXX=mpicxx
export F90=mpif90
export F77=mpif77
export FC=mpif90
```

Step 3 Go to the MOOSE directory.

```
cd /path/to/MOOSE/moose
```

Step 4 Obtain PETSc 3.13.2.

```
wget https://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-3.13.2.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Compile PETSc 3.13.2.

```
tar -xvf petsc-3.13.2.tar.gz
mv petsc-3.13.2.tar.gz petsc
cd petsc
export PETSC_DIR=/path/to/MOOSE/moose/petsc/
export PETSC_ARCH=arch-moose
./configure --download-hypre=1 --with-debugging=no --with-shared-libraries=1 --
download[1]fblaslapack=1 --download-metis=1 --download-ptscotch=1 --download-parmetis=1 --
download-superlu_dist=1 --download-mumps=1 --download-strumpack=0 --download[1]scalapack=1
--download-slepc=1 --with-mpi-dir=/path/to/hypermpi/hmpi --with-openmp=0 -- with-cxx-dialect=C+
+11 --with-fortran-bindings=0 --with-sowing=0 --with-64-bit-indices
```

NOTE

--with-mpi-dir is the MPI path. Downloading some of the packages requires Internet connection. If the packages cannot be automatically downloaded, you can manually download them. Change **1** to the path for storing the packages. For example, change **--download-hypre=1** to **--download-hypre=/path/to/MOOSE/package/v2.20.0.tar.gz**.

```
export OMPI_ALLOW_RUN_AS_ROOT=1
export OMPI_ALLOW_RUN_AS_ROOT_CONFIRM=1
make all
make check
```

If the following information is displayed, the installation is successful:

```
Running check examples to verify correct installation
Using PETSC_DIR=/path/to/MOOSE/moose/petsc and PETSC_ARCH=arch-moose
C/C++ example src/snes/tutorials/ex19 run successfully with 1 MPI process
C/C++ example src/snes/tutorials/ex19 run successfully with 2 MPI processes
C/C++ example src/snes/tutorials/ex19 run successfully with hypre
C/C++ example src/snes/tutorials/ex19 run successfully with mumps
C/C++ example src/snes/tutorials/ex19 run successfully with superlu_dist
Completed test examples
```

Step 6 Compile and install libmesh.

```
cd /path/to/MOOSE/moose/scripts
./update_and_rebuild_libmesh.sh
```

Step 7 Install MOOSE.

```
cd /path/to/MOOSE/moose/test
make
```

NOTE

If you run the **make -j** command, an error may be reported.

----End

3.33.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/MOOSE/moose/test/tests/preconditioners/hmg
```

Step 3 Perform the test.

- Single node

  ```
  time -p mpirun --allow-run-as-root -n 8 /path/to/MOOSE/moose/test/moose_test-opt -i
  diffusion_hmg.i Mesh/dmg/dim=3 Mesh/dmg/nz=100 Outputs/file_base=diffusion_hmg_3d_out -
  log_view --error --error-unused --error[1]override --no-gdb-backtrac
  ```

- Multiple nodes
  time -p mpirun --allow-run-as-root -n *16* **--hostfile hostfile** */path/to/MOOSE*/**moose/test/moose_test-opt -i diffusion_hmg.i Mesh/dmg/dim=3 Mesh/dmg/nz=100 Outputs/file_base=diffusion_hmg_3d_out -log_view --error --error[1]unused --error-override --no-gdb-backtrace**

  NOTE

  The **-np** parameter indicates the total number of processes. Set this parameter based on the server configuration. For the multi-node configuration, **--hostfile** is the list of node names. Create a **hostfile** file in the current directory and write the host names of the nodes into the file.

----End

3.34 SIESTA 4.0.2 Porting Guide (openEuler 21.03)

3.34.1 Introduction

SIESTA is both a method and its computer program implementation, to perform efficient electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids.

For more information, visit the **official SIESTA website**.

Programming languages: C/C++

Brief description: a piece of material calculation simulation software.

Recommended Software Version

SIESTA 4.0.2

 Copyright © Huawei Technologies Co., Ltd.

3.34.2 Environment Requirements

Hardware Requirements

Table 3-172 lists the hardware requirements.

Table 3-172 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-173 lists the software requirements.

Table 3-173 Software requirements

Item	Version	Download URL
SIESTA	4.0.2	https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz
ScaLAPACK	2.0.2	http://www.netlib.org/scalapack/scalapack-2.0.2.tgz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.6.tar.gz
GCC for openEuler	9.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2020.12-aarch64-linux.tar.gz
Hyper MPI	1.1.0	https://www.hikunpeng.com/en/developer/hpc/hypermpi

OS Requirements

Table 3-174 lists the OS requirements.

Table 3-174 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://repo.openeuler.org/openEuler-21.03/
Kernel	4.14.0-115	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

3.34.3 Paths for Software Porting

This chapter describes the software installation paths involved in the SIESTA software porting.

Table 3-175 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/hypermpi	Installation path of Hyper MPI.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/SCALAPACK	Installation path of ScaLAPACK.	
3	/path/to/SIESTA	Installation path of SIESTA.	
4	/opt/aarch64/compiler	Installation path of GCC for openEuler.	
5	/path/to/OPENBLAS	Installation path of OpenBLAS.	

3.34.4 Configuring the Compilation Environment

3.34.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-176 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing Hyper MPI	See **Hyper MPI 1.1.0 Operation Guide**.
3	Installing ScaLAPACK	See **3.34.4.2 Installing ScaLAPACK**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
4	Installing OpenBLAS	See **3.34.4.3 Installing OpenBLAS**.

3.34.4.2 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the ScaLAPACK installation package.

```
tar -xvf scalapack-2.0.2.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd scalapack-2.0.2
```

Step 4 Create an **SLmake.inc** file, and then compile and install it.

```
cp SLmake.inc.example SLmake.inc
make
```

Step 5 Copy the generated link library to the installation directory.

```
cp libscalapack.a /path/to/SCALAPACK
```

----End

3.34.4.3 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the OpenBLAS installation directory.

```
cd /path/to/OPENBLAS
```

Step 3 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 5 Perform the configuration.

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 6 Perform the compilation and installation.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 7 Load the environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.34.5 Obtaining the Source Code

Procedure

Step 1 Download the SIESTA installation package **siesta-4.0.2.tar.gz**.

https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz

Step 2 Use an SFTP tool to upload the SIESTA installation package to the */path/to/SIESTA* directory on the server.

----End

3.34.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain and decompress SIESTA to the planned path.

```
wget https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz
tar -xvf siesta-4.0.2.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd siesta-4.0.2
```

Step 4 Create the directory.

```
mkdir kpgcc_hmpi_compiler
```

Step 5 Go to the directory and execute the script.

```
cd kpgcc_hmpi_compiler
bash ../Src/obj_setup.sh
```

Step 6 Perform the configuration.

```
../Src/configure CC=mpicc CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' FC=mpifort FCFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' --enable-mpi --build=arm-linux --prefix=/path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler
```

Step 7 Modify the **arch.make** file.

1. Open the **arch.make** file.
   ```
   vi arch.make
   ```
2. Press **i** to enter the edit mode and modify lines 34, 35, 36, 38, 39, and 45.
   ```
   OPENBLAS_LIBS=/path/to/OPENBLAS/lib/libopenblas.a
   #BLAS_LIBS=-lblas
   #LAPACK_LIBS=-llapack
   SCALAPACK_LIBS=/path/to/SCALAPACK/libscalapack.a
   MATH_LIBS= -Wl,-rpath=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm
   LIBS=$(SCALAPACK_LIBS) $(BLACS_LIBS) $(OPENBLAS_LIBS) $(NETCDF_LIBS) $(MATH_LIBS)
   ```

   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Replace the **config.guess** and **config.sub** scripts in the **Fox** directory with the content provided in the URLs.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/SIESTA/siesta-4.0.2/Src/FoX/config
```

In the **Fox** directory, copy the content provided in the URL to replace the content of the **config.guess** and **config.sub** scripts.

http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess

http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub

Step 9 Perform the compilation and installation.

```
make
```

Step 10 Set the environment variables.

```
export PATH=/path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler/:$PATH
```

----End

3.34.7 Running and Verification

NOTE

- Determine the number of running processes based on the server configuration.
- If multiple nodes are running, configure the network environment:
  - In the **/etc/hosts** file, add the IP addresses and host names of the servers.
  - Enable password-free SSH login between the servers.
  - Disable the firewall and ensure that clients and servers are not disconnected.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Copy the **Tests** directory of the source package to the test case directory.

```
cd /path/to/SIESTA
cp -r ./siesta-4.0.2/Tests /path/to/TESTCASE
```

Step 3 Go to the test case directory.

```
cd /path/to/TESTCASE/Tests/h2o
```

Step 4 Configure the execution file.

1. Open the execution file.
   ```
   vi ../test.mk
   ```
2. Press **i** to enter the insert mode and modify the content in line 10.
   - Single node
     ```
     SIESTA= time -p mpirun --allow-run-as-root -np 8 /path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler/siesta
     ```
   - Multiple nodes
     ```
     SIESTA= time -p mpirun --allow-run-as-root -x PATH=$ (PATH) -x LD_LIBRARY_PATH=$ (LD_LIBRARY_PATH) --hostfile /home/hostfile -np 16 -N 8 /path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler/siesta
     ```

Table 3-177 Parameter description

Parameter	Description
-np	Total number of processes used in the test.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
-N	Number of processes running on each server.
--hostfile	Node name.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Create a **host** file.

1. Create a **host** file.
```
cd /home
vi hostfile
```
2. Press **i** to enter the insert mode and add the following content:
```
node1
node2
```

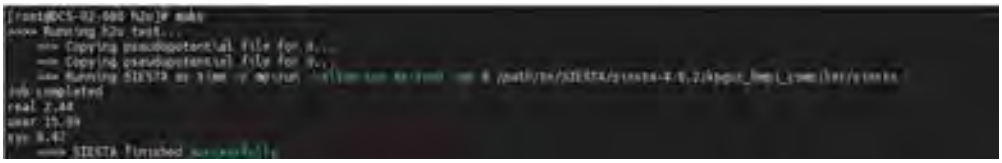
NOTE

node1 and **node2** are the host names of the servers.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the SIESTA test case.
```
cd /path/to/TESTCASE/Tests/h2o
make
```

- Result for single node



- Result for multiple nodes

----End

3.35 UDUNITS 2.2.28 Porting Guide (openEuler 21.03)

3.35.1 Introduction

The UDUNITS-2 package provides support for units of physical quantities. Its three main components are: (1) udunits2lib, a C library for units of physical quantities; (2) udunits2prog, a utility for obtaining the definition of a unit and for converting numeric values between compatible units; and (3) an extensive database of units.

For more information, visit the official UDUNITS-2 website.

Programming languages: C/Fortran

Brief description: database.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: GPLv3

Recommended Software Version

UDUNITS 2.2.28

3.35.2 Environment Requirements

Hardware Requirements

Table 3-178 lists the hardware requirements.

Table 3-178 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-179 lists the software requirements.

Table 3-179 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
UDUNITS	2.2.28	https://gfd-dennou.org/arch/ucar/unidata/pub/udunits/udunits-2.2.28.tar.gz

OS Requirements

Table 3-180 lists the OS requirements.

Table 3-180 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

3.35.3 Paths for Software Porting

This chapter lists the software installation paths involved in the UDUNITS software porting.

Table 3-181 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/UDUNITS	Installation path of UDUNITS.	The installation paths listed in this table are only examples. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

3.35.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-182 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, you need to specify the compilation parameters.

CC=clang CXX=clang++ FC=flang

----End

3.35.5 Obtaining the Source Code

Procedure

Step 1 Download the UDUNITS installation package **udunits-2.2.28.tar.gzx**.

https://gfd-dennou.org/arch/ucar/unidata/pub/udunits/udunits-2.2.28.tar.gz

Step 2 Use an SFTP tool to upload the UDUNITS installation package to the */path/to/UDUNITS* directory on the server.

----End

3.35.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/UDUNITS
```

Step 3 Decompress the installation package.

```
tar -xzvf udunits-2.2.28.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd udunits-2.2.28
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/UDUNITS CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 6 Perform the compilation.

 Copyright © Huawei Technologies Co., Ltd.

```
make
make install
```

----End

3.35.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check the installation path. If the following file is generated, the installation is successful:

The path is */PATH/to/UDUNITS/uduntis/lib*.

```
[ZZX@node01 lib]$ ls
libudunits2.a  libudunits2.la  libudunits2.so  libudunits2.so.0  libudunits2.so.0.1.0
[ZZX@node01 lib]$ pwd
/home/ZZX/WORK/udunits/udunitd/lib
```

----End

3.36 ROOT 6.20 Porting Guide (openEuler 21.03)

3.36.1 Introduction

ROOT is a modular scientific software toolkit. It provides all the functions required for big data processing, statistical analysis, visualization, and storage.

For more information about ROOT, visit the **ROOT official website**.

Programing languages: C++/Python

Brief description: modular scientific software toolkit.

Open source license: GPL 3.0

Recommended Software Version

ROOT 6.20

3.36.2 Environment Requirements

Hardware Requirements

Table 3-183 lists the hardware requirements.

Table 3-183 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 3-184 lists the software requirements.

Table 3-184 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz
Hyper MPI	1.0.2	https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz
ROOT	6.20	https://root.cern.ch/content/release-62000
zstd	1.3.4	https://github.com/facebook/zstd/releases/tag/v1.3.4
CMake	3.17.5	https://cmake.org/download/
Test cases	Benchmarks.C	Provided by the software.

OS Requirements

Table 3-185 lists the OS requirements.

Table 3-185 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

3.36.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ROOT software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-186 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/ROOT	Installation path of ROOT.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/CMAKE	Installation path of CMake.	
5	/path/to/ZSTD	Installation path of zstd.	

3.36.4 Configuring the Compilation Environment

3.36.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-187 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing CMake	See **3.36.4.2 Installing CMake**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
4	Installing zstd	See **3.36.4.3 Installing zstd**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source  ~/.bashrc
```

----End

3.36.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the CMake installation package.

```
tar -xvf cmake-3.17.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd cmake-3.17.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/CMAKE CC=mpicc CXX=mpicxx
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Set the environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.36.4.3 Installing zstd

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the zstd installation package.

```
tar -zxvf zstd-1.3.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd zstd-1.3.4
```

Step 4 Perform the compilation and installation.

```
make CC=mpicc CXX=mpicxx FC=mpif90
make install
```

Step 5 Set the zstd environment variables.

```
export LD_LIBRARY_PATH=/path/to/ZSTD/zstd-1.3.4/lib:$LD_LIBRARY_PATH
```

----End

3.36.5 Obtaining the Source Code

Procedure

Step 1 Download the ROOT installation package.

https://root.cern.ch/content/release-62000

Step 2 Use an SFTP tool to upload the ROOT installation package to the */path/to/ROOT* directory on the server.

----End

3.36.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the following directory:

```
cd /path/to/ROOT
```

Step 3 Decompress the ROOT installation package and go to the installation directory.

```
tar -xzvf root_v6.20.00.source.tar.gz
cd root-6.20.00
```

Step 4 Create and access the directory for compiling the ROOT software.

```
mkdir obj
cd obj
```

Step 5 Ensure that the server is connected to the Internet and compile and install the software.

```
cmake ./ -DCMAKE_INSTALL_PREFIX=/path/to/ROOT -DENABLE_PERL=FALSE
make
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Add the environment variables.

```
source ../bin/thisroot.sh
```

----End

3.36.7 Running and Verification

Procedure

Step 1 Log in to the server using PuTTY.

Step 2 Go to the test case directory.

```
cd /path/to/ROOT/root-6.20.00/obj/tutorials/legacy
```

Step 3 Start ROOT.

```
root
```

Step 4 Perform a benchmark test.

```
.x benchmarks.C
```

Check the value of **ROOTMARKS** in the command output. A larger value indicates better single-core performance.

The command output in the following figure is displayed:

```
---------------ROOT 6.20/00 benchmarks summary (in ROOTMARKS)-----
   For comparison, a Pentium IV 2.4Ghz is benchmarked at 600 ROOTMARKS
hsimple     = 2353.99 RealMARKS,  = 2100.00 CpuMARKS
hsum        = 2985.91 RealMARKS,  = 2250.00 CpuMARKS
fillrandom  =  439.55 RealMARKS,  =  200.00 CpuMARKS
fit1        =  377.89 RealMARKS,  =  300.00 CpuMARKS
tornado     = 25100.56 RealMARKS,  =     inf CpuMARKS
na49        = 69332.84 RealMARKS,  = 83400.00 CpuMARKS
geometry    = 4004.34 RealMARKS,  = 5400.00 CpuMARKS
na49view    = 3743.80 RealMARKS,  = 1800.00 CpuMARKS
ntuple1     = 1236.29 RealMARKS,  = 1246.15 CpuMARKS

**************************************************
* Your machine is estimated at 5253.07 ROOTMARKS   *
**************************************************
root [1]
```

----End

3.37 LAMMPS 10 Feb 2021 Porting Guide (CentOS 8.2)

3.37.1 Introduction

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a classical molecular dynamics simulation code designed to run efficiently on parallel computers. LAMMPS uses MPI to implement multi-machine parallel computing. Its latest version supports GPU computing based on CUDA and OpenCL. It is an open source code, distributed freely under the terms of the GNU Public License (GPL).

For more information, visit the **official LAMMPS website**.

Programming language: C++

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a classical molecular dynamics simulation code designed to run efficiently on parallel computers.

Open source license: GPL 2.0

Recommended Software Version

LAMMPS 10Feb21

3.37.2 Environment Requirements

Hardware Requirements

Table 3-188 lists the hardware requirements.

Table 3-188 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-189 lists the software requirements.

Table 3-189 Software requirements

Item	Version	Download Address
LAMMPS	10Feb21	https://lammps.sandia.gov/tars/
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
Test case	in.lj	Included in the **bench** directory of the LAMMPS source package.

OS Requirements

Table 3-190 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-190 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

3.37.3 Paths for Software Porting

This chapter lists software installation paths involved in the LAMMPS software porting.

Table 3-191 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FFTW	Installation path of FFTW.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/LAMMPS	Installation path of LAMMPS	
4	/path/to/CASE	Path of LAMMPS test cases.	
5	/path/to/ompi	Installation path of the Hyper MPI.	
6	/path/to/ucx		

3.37.4 Configuring the Compilation Environment

3.37.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-192 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing FFTW	See **3.37.4.2 Installing FFTW**.
3	Disabling the Nouveau driver	See **3.37.4.3 Disabling the Nouveau Driver**.
4	Installing NVIDIA CUDA	See **3.37.4.4 Installing NVIDIA CUDA**.
5	Installing the BiSheng Compiler	See **3.37.4.5 Installing the BiSheng Compiler**.
6	Installing Hyper MPI	See **3.37.4.6 Installing Hyper MPI**.

3.37.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Perform the compilation and installation.

```
make -j 96
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.37.4.3 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

3.37.4.4 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3** to **Step 5**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 5 Perform the verification.

```
nvcc --version
```

 Copyright © Huawei Technologies Co., Ltd.

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

3.37.4.5 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the BiSheng Compiler installation package.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 3 Create a BiSheng Compiler installation directory (**/opt/compiler** is used as an example).

```
mkdir -p /opt/compiler
```

Step 4 Copy the BiSheng Compiler installation package to the installation directory.

```
cp -r bisheng-compiler-1.3.3-aarch64-linux.tar.gz /opt/compiler
```

Step 5 Go to the installation package directory.

```
cd /opt/compiler
```

Step 6 Decompress the installation package.

```
tar -zxvf bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 7 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 8 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

3.37.4.6 Installing Hyper MPI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the HUCX and XUCG source packages.

```
wget -c https://github.com/kunpengcompute/hucx/archive/refs/heads/huawei.zip -O hucx-huawei.zip
wget -c https://github.com/kunpengcompute/xucg/archive/refs/heads/huawei.zip -O xucg-huawei.zip
```

Step 3 Decompress the HUCX and XUCG source packages.

```
unzip hucx-huawei.zip
unzip xucg-huawei.zip
```

Step 4 Copy the content in the XUCG source package to the **src/ucg** directory of the HUCX source package.

```
cp -r xucg-huawei/* hucx-huawei/src/ucg/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Go to the **hucx-huawei** directory.

```
cd hucx-huawei
```

Step 6 Set the environment variables.

```
export SEC_CFLAGS="-Wno-error=deprecated-declarations -DHAVE___CLEAR_CACHE=1 -Wno-error=asm-operand-widths -Wno-error=implicit-int-float-conversion -Wno-error=format -Wno-error=tautological-pointer-compare"
```

Step 7 Create an HUCX installation directory.

```
mkdir -p /path/to/ucx
```

Step 8 Install HUCX.

```
./autogen.sh
./contrib/configure-opt --prefix=/path/to/ucx CFLAGS="$SEC_CFLAGS" CXXFLAGS="$SEC_CFLAGS"
make -j32
make install
```

Step 9 Obtain the Hyper MPI source package.

```
wget -c https://github.com/kunpengcompute/hmpi/archive/refs/heads/huawei.zip -O hmpi-huawei.zip
```

Step 10 Decompress the Hyper MPI source package.

```
unzip hmpi-huawei.zip
```

Step 11 Go to the **hmpi-huawei** directory.

```
cd hmpi-huawei
```

Step 12 Install Hyper MPI.

```
./autogen.pl
./configure --prefix=/path/to/ompi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=/path/to/ucx
make -j16
make install
```

Step 13 Set the environment variables.

```
export PATH=/path/to/ompi/bin:/path/to/ucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/ompi/lib:/path/to/ucx/lib:$LD_LIBRARY_PATH
export OPAL_PREFIX=/path/to/ompi/
export INCLUDE=/path/to/ompi:$INCLUDE
```

----End

3.37.5 Obtaining the Source Code

Procedure

Step 1 Download the LAMMPS installation package.

https://lammps.sandia.gov/tars/

Step 2 Use an SFTP tool to upload the LAMMPS installation package to the */path/to/LAMMPS* directory on the server.

----End

3.37.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the main program installation directory.

```
cd /path/to/LAMMPS
```

Step 3 Decompress the installation package.

```
tar -xvf lammps-10Feb2021.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd lammps-10Feb21
```

Step 5 Modify the **Makefile.linux** configuration file of the GPU module.

1. Open the **Makefile.linux** file.

   ```
   cd lib/gpu
   vi Makefile.linux
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   ```
   CUDA_HOME = /usr/local/cuda-11.4
   NVCC=nvcc
   CUDA_ARCH=-arch=sm_80  # Graphics card model. The current environment uses A100 (Ampere architecture).
   CUDA_PRECISION=-D_DOUBLE_DOUBLE  # calculation precision.
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Compile the GPU module of LAMMPS.

```
make -f Makefile.linux -j 16
```

Step 7 View the GPU information.

```
./nvc_get_devices
```

If the following information is displayed, the compilation is successful:

```
Found 1 platform(s).
Using platform: NVIDIA Corporation NVIDIA CUDA Driver
CUDA Driver Version:                           11.40
Device 0: "NVIDIA A100-PCIE-40GB"
Type of device:                                GPU
Compute capability:                            8
Double precision support:                      Yes
Total amount of global memory:                 39.5861 GB
Number of compute units/multiprocessors:       108
Number of cores:                               20736
Total amount of constant memory:               65536 bytes
Total amount of local/shared memory per block: 49152 bytes
Total number of registers available per block: 65536
Warp size:                                     32
Maximum number of threads per block:           1024
Maximum group size (# of threads per block)    1024 x 1024 x 64
Maximum item sizes (# threads for each dim)    2147483647 x 65535 x 65535
Maximum memory pitch:                          2147483647 bytes
Texture alignment:                             512 bytes
Clock rate:                                    1.41 GHz
Run time limit on kernels:                     No
Integrated:                                    No
Support host page-locked memory mapping:       Yes
Compute mode:                                  Default
Concurrent kernel execution:                   Yes
Device has ECC support enabled:                Yes
```

Step 8 Go to the **src** directory.

```
cd ../../src
```

Step 9 Modify the **MAKE/OPTIONS/Makefile.g++_openmpi** file.

1. Open the **MAKE/OPTIONS/Makefile.g++_openmpi** file.

   ```
   vi MAKE/OPTIONS/Makefile.g++_openmpi
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the edit mode and modify the content in lines 54, 55, and 56.

```
FFT_INC =    -DFFT_FFTW -I/path/to/FFTW/include
FFT_PATH =  -L/path/to/FFTW/lib
FFT_LIB =    -lfftw3
```

NOTE

After performing **Step 9.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 10 Perform the compilation.

```
make yes-std
make no-lib
make yes-gpu
make -j 96 g++_openmpi
```

----End

3.37.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create the working directory.

```
mkdir -p /path/to/CASE
```

Step 3 Go to the working directory, and copy the test cases and binary files to the working directory.

```
cd /path/to/CASE
cp /path/to/LAMMPS/lammps-10Feb21/bench/in.lj ./
cp /path/to/LAMMPS/lammps-10Feb21/src/lmp_g++_openmpi ./
```

Step 4 Perform the test.

```
mpirun --allow-run-as-root -np 8 --mca btl ^openib ./lmp_g++_openmpi -sf gpu -in in.lj
```

If the following information is displayed, the running is successful:

Check the value (in the unit of **timesteps/s**) of **Performance** in the **log.lammps** file. A higher value indicates higher performance.

```
Performance: 231663.111 tau/day, 536.257 timesteps/s
99.9% CPU use with 8 MPI tasks x no OpenMP threads
MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.061975   | 0.18125    | 0.27592    |  15.7 | 47.70
Neigh   | 1.06e-06   | 2.1069e-06 | 3.64e-06   |   0.0 |  0.00
Comm    | 0.09717    | 0.19305    | 0.3114     |  15.1 | 50.81
Output  | 5.789e-05  | 0.0014428  | 0.0028119  |   2.2 |  0.38
Modify  | 0.0033437  | 0.0034793  | 0.0035951  |   0.1 |  0.92
Other   |            | 0.0007299  |            |       |  0.19
```

----End

3.38 QE 6.8 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

3.38.1 Introduction

Quantum ESPRESSO (QE) is a suite for first-principles electronic-structure calculations and materials modeling, distributed for free and as free software under the GNU General Public License. It is based on density-functional theory, plane wave basis sets, and pseudopotentials (both norm-conserving and ultrasoft). Quantum ESPRESSO, written mainly in Fortran-90 with some parts in C or in Fortran-77, was built out of the merging and re-engineering of different independently-developed core packages, plus a set of packages designed to be inter-operable with the core components. The basic packages include Pwscf, which solves the self-consistent Kohn and Sham equations and obtains for a periodic solid, CP, which carries out Car-Parrinello molecular dynamics, and PostProc, which allows data analysis and plotting. Regarding the additional packages, it is noteworthy to point out atomic for the pseudopotential generation, PHonon package, which implements density-functional perturbation theory (DFPT) for the calculation of second- and third-order derivatives of the energy with respect to atomic displacements and NEB.

For more information about LAMMPS, visit the **QE official website**.

Programming languages: C/Fortran

Brief description: scalable computational chemistry tools

Open source license: GPL 2.0

Recommended Software Version

QE 6.8

3.38.2 Environment Requirements

Hardware Requirements

Table 3-193 lists the hardware requirements.

Table 3-193 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla V100

Software Requirements

Table 3-194 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-194 Software requirements

Item	Version	Download Address
QUANTUM ESPRESSO	6.8	https://github.com/QEF/q-e/archive/refs/tags/qe-6.8.tar.gz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.6.tar.gz
Hyper MPI	1.1.1	https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
NVIDIA HPC SDK	21.9	https://developer.download.nvidia.com/hpc-sdk/21.9/nvhpc_2021_219_Linux_aarch64_cuda_11.4.tar.gz

OS Requirements

Table 3-195 lists the OS requirements.

Table 3-195 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

3.38.3 Paths for Software Porting

This chapter lists software installation paths involved in the QE software porting.

Table 3-196 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/qe	Installation path of QE.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/usr/local/cuda-11.4	Installation path of NVIDIA CUDA.	
4	/path/to/openblas	Installation path of OpenBLAS.	
5	/opt/nvidia/hpc_sdk	Installation path of the NVIDIA HPC SDK.	
6	/path/to/CASE	Path of QE test cases.	

3.38.4 Configuring the Compilation Environment

3.38.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-197 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Disabling the Nouveau driver	See **3.38.4.2 Disabling the Nouveau Driver**.
3	Installing NVIDIA CUDA	See **3.38.4.3 Installing NVIDIA CUDA**.
4	Installing Hyper MPI	See **3.38.4.4 Installing Hyper MPI**.
5	Installing the NVIDIA HPC SDK	See **3.38.4.5 Installing the NVIDIA HPC SDK**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
6	Installing OpenBLAS	See **3.38.4.6 Installing OpenBLAS**.

3.38.4.2 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

3.38.4.3 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3** to **Step 5**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Install NVIDIA CUDA.

 Copyright © Huawei Technologies Co., Ltd.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 5 Perform the verification.

```
nvcc --version
```

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

3.38.4.4 Installing Hyper MPI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the HUCX and XUCG source packages.

```
wget -c https://github.com/kunpengcompute/hucx/archive/refs/heads/huawei.zip -O hucx-huawei.zip
wget -c https://github.com/kunpengcompute/xucg/archive/refs/heads/huawei.zip -O xucg-huawei.zip
```

Step 3 Decompress the HUCX and XUCG source packages.

```
unzip hucx-huawei.zip
unzip xucg-huawei.zip
```

Step 4 Copy the content in the XUCG source package to the **src/ucg** directory of the HUCX source package.

```
cp -r xucg-huawei/* hucx-huawei/src/ucg/
```

Step 5 Go to the **hucx-huawei** directory.

```
cd hucx-huawei
```

Step 6 Set the environment variables.

```
export SEC_CFLAGS="-Wno-error=deprecated-declarations -DHAVE___CLEAR_CACHE=1 -Wno-
error=asm-operand-widths -Wno-error=implicit-int-float-conversion -Wno-error=format -Wno-
error=tautological-pointer-compare"
```

Step 7 Create an HUCX installation directory.

```
mkdir -p /path/to/ucx
```

Step 8 Install HUCX.

```
./autogen.sh
./contrib/configure-opt --prefix=/path/to/ucx CFLAGS="$SEC_CFLAGS" CXXFLAGS="$SEC_CFLAGS"
make -j32
make install
```

Step 9 Obtain the Hyper MPI source package.

```
wget -c https://github.com/kunpengcompute/hmpi/archive/refs/heads/huawei.zip -O hmpi-huawei.zip
```

Step 10 Decompress the Hyper MPI source package.

```
unzip hmpi-huawei.zip
```

Step 11 Go to the **hmpi-huawei** directory.

```
cd hmpi-huawei
```

Step 12 Install Hyper MPI.

```
./autogen.pl
./configure --prefix=/path/to/ompi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=/path/to/ucx
make -j16
make install
```

Step 13 Set the environment variables.

```
export PATH=/path/to/ompi/bin:/path/to/ucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/ompi/lib:/path/to/ucx/lib:$LD_LIBRARY_PATH
export OPAL_PREFIX=/path/to/ompi/
export INCLUDE=/path/to/ompi:$INCLUDE
```

----End

3.38.4.5 Installing the NVIDIA HPC SDK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the NVHPC software package.

```
wget https://developer.download.nvidia.com/hpc-sdk/21.9/nvhpc_2021_219_Linux_aarch64_cuda_11.4.tar.gz
```

Step 3 Create an installation directory.

```
mkdir -p /opt/compiler
```

Step 4 Copy the NVHPC software package to the installation directory.

```
cp –r nvhpc_2021_219_Linux_aarch64_cuda_11.4.tar.gz /opt/compiler
```

Step 5 Go to the installation directory.

```
cd /opt/compiler
```

Step 6 Decompress the software package.

```
tar -zxvf nvhpc_2021_219_Linux_aarch64_cuda_11.4.tar.gz
```

Step 7 Go to the directory generated after the decompression.

```
cd nvhpc_2021_219_Linux_aarch64_cuda_11.4/
```

Step 8 Start the installation.

```
./install
```

NOTE

Press **Enter** to confirm the installation location. The default installation location is **/opt/nvidia/hpc_sdk**. Then press **1** to select single-node installation. The installation program automatically finds the previously installed CUDA Toolkit. After the installation is complete, press **Enter**.

Step 9 Set the NVIDIA HPC SDK environment variables.

```
module load /opt/nvidia/hpc_sdk/modulefiles/nvhpc/21.9
```

Step 10 Perform the verification.

```
pgcc --version
```

If the command output contains "PGI Compilers and Tools", the installation is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.38.4.6 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -zxvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Compile and install OpenBLAS.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 5 Set the OpenBLAS environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.38.5 Obtaining the Source Code

Procedure

Step 1 Download the QE installation package.

https://github.com/QEF/q-e/releases/tag/qe-6.8

Step 2 Use an SFTP tool to upload the QE installation package to the */path/to/qe* directory on the server.

----End

3.38.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/qe
```

Step 3 Decompress the installation package.

```
tar -zxvf q-e-qe-6.8.tar
```

Step 4 Go to the directory generated after the decompression.

```
cd q-e-qe-6.8/
```

Step 5 Perform the compilation.

```
./configure --with-cuda=yes --with-cuda-runtime=11.4 --with-cuda-cc=80  --enable-openmp --with-scalapack=no
```

NOTE

Modify **--with-cuda-cc=80** based on the graphics card architecture of the current host. In this example, the graphics card is V100.

```
make -j32 pwall
```

 Copyright © Huawei Technologies Co., Ltd.

- If the compilation host is not connected to the Internet, download the **devicexlib-master.tar.gz** package and upload it to the compilation host. After the compilation is complete, run the **make pp -j32** command.

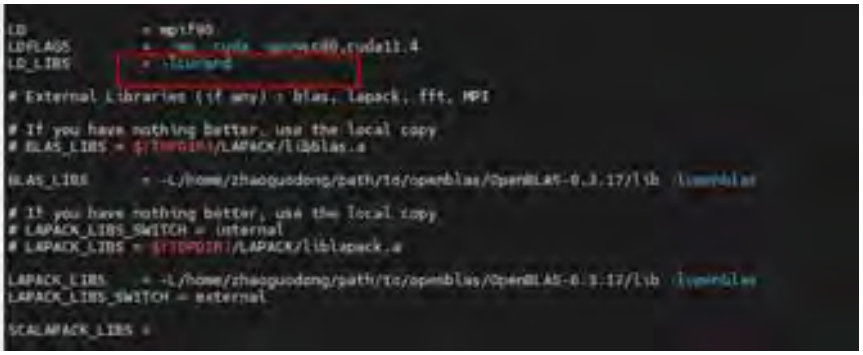
```
mv devicexlib-master.tar.gz /path/to/qe-gpu-6.7/external devicexlib.tar.gz
tar xzf devicexlib.tar.gz --strip-components=1
export F90FLAGS="-FAST -mCACHE_align -Mpreprocess -Mlarge_arrays"
./configure FC=pgf90 CC=pgcc --with-cuda=no --with-cuda-cc= --with-cuda-runtime= --disable-parallel --enable-cuda-env-check=no
make pp -j32
```

- If error information similar to the following is displayed, the link library configuration is incorrect. You need to modify the **make.inc** file.

a. Open the **make.inc** file.

```
vi make.inc
```

b. Press **i** to enter the edit mode and add **LD_LIBS = -lcurand** to the following location:



c. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Check whether an executable file is generated.

```
ll /path/to/qe-gpu-6.7/bin/pw.x
```

If the command output contains the **pw.x** file information, the executable file has been generated.

Step 7 Set the environment variables.

```
export PATH=/path/to/qe/bin/:$PATH
```

----End

3.38.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/CASE
```

Step 3 Download the test cases and use an SFTP tool to upload the test cases to the */path/to/CASE* directory on the server.

```
wget https://github.com/QEF/benchmarks/archive/refs/tags/bench0.0.tar.gz
```

Step 4 Decompress the test case package and copy the required files to the */path/to/CASE* directory.

```
tar xvf benchmarks-bench0.0.tar.gz
cp benchmarks-master/AUSURF112/* ./
```

Step 5 Start the QE program.

```
mpirun --allow-run-as-root -np 2 -x OMP_NUM_THREADS=1 --mca btl ^openib pw.x -input ./ausurf.in 2>&1 | tee -a qe.log
```

NOTE

-np indicates the total number of processes used in the test. The number **2** indicates that the test machine has two graphics cards.

-x OMP_NUM_THREADS=1 indicates that each process has one thread.

Check the value (in the unit of **s**) of **WALL** in the **qe.log** file. A higher value indicates lower performance. The following figure shows the output result.

```
General routines
calbec       :      2.33s CPU      2.34s WALL (      89 calls)
fft          :      4.23s CPU      4.31s WALL (      40 calls)
                                   0.40s GPU  (       3 calls)
ffts         :      0.25s CPU      0.25s WALL (       6 calls)
fftw         :     17.03s CPU     18.82s WALL (    2432 calls)
                                  17.52s GPU  (    2432 calls)
interpolate  :      0.46s CPU      0.47s WALL (       3 calls)
davcio       :      0.00s CPU      0.19s WALL (       6 calls)

Parallel routines

PWSCF        :   1m 8.84s CPU   1m13.45s WALL
```

----End

3.39 ABINIT 8.10.3 Porting Guide (openEuler 21.03)

3.39.1 Introduction

ABINIT is a software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials. Starting from the quantum equations of density functional theory, you can build up to advanced applications with perturbation theories based on DFT, and many-body Green's functions (GW and DMFT). ABINIT can calculate molecules, nanostructures and solids with any

 Copyright © Huawei Technologies Co., Ltd.

chemical composition, and comes with several complete and robust tables of atomic potentials.

For more information, visit the **official ABINIT website**.

Programming languages: C/Fortran 90

Brief description: A software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials.

Open source license: GPL 3.0

Recommended Software Version

ABINIT 8.10.3

3.39.2 Environment Requirements

Hardware Requirements

Table 3-198 lists the hardware requirements.

Table 3-198 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-199 lists the software requirements.

Table 3-199 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz**
ABINIT	8.10.3	**https://www.abinit.org/packages**
PNETCDF	1.9.0	**https://parallel-netcdf.github.io/wiki/Download.html**
NETCDF-C	4.7.0	**https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0**
NETCDF-Fortran	4.4.5	**https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
AtomPAW	4.0.10	**https://www.abinit.org/fallbacks**
BigDFT	1.7.1.25	**https://www.abinit.org/fallbacks**
LibXC	3.0.0	**https://www.abinit.org/fallbacks**
Wannier90	2.0.1.1	**https://www.abinit.org/fallbacks**
HDF5	1.10.1	**https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz**
OpenBLAS	0.6.3	**https://github.com/xianyi/OpenBLAS/tree/v0.3.6**
Test cases	tbasepar_1.files	Provided by the software.

OS Requirements

Table 3-200 lists the OS requirements.

Table 3-200 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://www.openeuler.org/zh/download**
Kernel	4.17	Included in the OS image

3.39.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ABINIT software porting.

Table 3-201 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/HDF5	Installation path of HDF5.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/NETCDF	Installation path of NetCDF-C.	
		Installation path of NetCDF-Fortran.	
5	/path/to/ABINIT	Installation path of ABINIT.	
6	/path/to/PNETCDF	Installation path of PnetCDF.	
7	/path/to/OPENBLAS	Installation path of OpenBLAS.	

3.39.4 Configuring the Compilation Environment

3.39.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-202 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
3	Installing HDF5	See **3.39.4.2 Installing HDF5**.
4	Installing NetCDF-C	See **3.39.4.3 Installing NetCDF-C**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
5	Installing NetCDF-Fortran	See **3.39.4.4 Installing NetCDF-Fortran**.
6	Installing PnetCDF	See **3.39.4.5 Installing PnetCDF**.
7	Installing OpenBLAS	See **3.39.4.6 Installing OpenBLAS**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source  ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

3.39.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> If the server has multiple MPIs, **CC**, **FC**, and **CXX** in the preceding command can be represented using absolute paths.

Step 5 Modify the **libtool** file in the current folder.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify line 11835 in the file.
   ```
   wl="-Wl,"
   ```

   > **NOTE**
   >
   > After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Set the HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

3.39.4.3 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L/path/to/HDF5/lib" CPPFLAGS="-I/path/to/HDF5/
include" CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the installation.

```
make
make install
```

Step 6 Set the NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.39.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd netcdf-fortran-4.4.5
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/NETCDF --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort F90=mpifort CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" CXXFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers" FCFLAGS="-O3 -fPIC" FFLAGS="-O3 -fPIC"
```

Step 5 Modify the **libtool** file in the current directory.

1. Open the **libtool** file.
   ```
   vi libtool
   ```
2. Press **i** to enter the insert mode and modify all the items of **wl=""** in the file.
   ```
   wl="-Wl,"
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make
make install
```

----End

3.39.4.5 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd parallel-netcdf-1.9.0
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/PNETCDF --enable-shared CC=mpicc --disable-dap FC=mpif90 CXX=mpicxx
```

Step 5 Perform the compilation and installation.

```
make
make install
```

Step 6 Set the PnetCDF environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

3.39.4.6 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the OpenBLAS installation package.

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 4 Set the environment variables.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 5 Compile and install OpenBLAS.

```
make
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Set the OpenBLAS environment variables.

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.39.5 Obtaining the Source Code

Procedure

Step 1 Download the ABINIT installation package.

https://www.abinit.org/

Step 2 Use an SFTP tool to upload the ABINIT installation package to the */path/to/ABINIT* directory on the server.

----End

3.39.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables.

```
export MPI_HOME=/path/to/HMPI
export CPPFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I
$MPI_HOME/include"
export CFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I
$MPI_HOME/include "
export CXXFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path-to/NETCDF/include -I
$MPI_HOME/include "
```

 Copyright © Huawei Technologies Co., Ltd.

```
export FCFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export FFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS=" -L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -L/path/to/PNETCDF/lib"
```

Step 3 Create an **extral** directory for storing third-party math libraries.

```
cd /path/to/ABINIT
mkdir extral
```

Step 4 Download the AtomPAW, BigDFT, LibXC, and Wannier90 installation packages.

https://www.abinit.org/fallbacks

- AtomPAW: **atompaw-4.0.1.0.tar.gz**
- BigDFT: **bigdft-1.7.1.25.tar.gz**
- LibXC: **libxc-3.0.0.tar.gz**
- Wannier90: **wannier90-2.0.1.1.tar.gz**

Step 5 Use an SFTP tool to upload the downloaded installation packages to the **extral** directory.

Step 6 Decompress the installation packages.

```
cd extral
tar -xzvf atompaw-4.0.1.0.tar.gz
tar -xzvf bigdft-1.7.1.25.tar.gz
tar -xzvf libxc-3.0.0.tar.gz
tar -xzvf wannier90-2.0.1.1.tar.gz
```

Step 7 Return to */path/to/ABINIT* and decompress the ABINIT installation package.

```
cd /path/to/ABINIT
tar -zxvf abinit-8.10.3.tar.gz
```

Step 8 Go to the directory generated after the decompression.

```
cd abinit-8.10.3
```

Step 9 Create and access the directory for compiling the ABINIT software.

```
mkdir build
cd build
```

Step 10 Perform the configuration.

```
../configure FC=mpif90 --with-mpi-prefix=/path/to/HMPI/ --with-dft-flavor="libxc+atompaw" --with-trio-flavor="netcdf" --with-tardir=/path/to/ABINIT/extral --with-netcdf-incs="-I/path/to/NETCDF/include" --withnetcdf-libs="-L/path/to/NETCDF/lib -lnetcdf -L/path/to/NETCDF/lib -lnetcdff" FCFLAGS="-g -O2 -ffree-line-length-none" --with-linalg-libs="-L/path/to/OPENBLAS/ -lopenblas" CFLAGS="-O2"
```

Step 11 Modify the configuration file.

```
sed -i -e "s/shiftr/rshift/g" src/78_effpot/m_random_xoroshiro128plus.F90
sed -i -e "s/shiftl/lshift/g" src/78_effpot/m_random_xoroshiro128plus.F90
```

Step 12 Go to the **ABINIT** directory.

```
cd /path/to/ABINIT/abinit-8.10.3
```

Step 13 Modify the configuration file.

```
sed -i 's/#define HAVE_FC_INT_QUAD 1/#undef HAVE_FC_INT_QUAD/' config.h
sed -i 's/#define HAVE_FC_LONG_LINES 1/#undef HAVE_FC_LONG_LINES/' config.h
sed -i 's/#define atomic_mass_units/at_mass_units/' src/77_ddb/m_phonons.F90
sed -i 's/#define number_of_atom_species/n_atom_species/' src/77_ddb/m_phonons.F90
sed -i 's/alpha = 0.2/!alpha = 0.2/' src/78_eph/m_phgamma.F90
```

Step 14 Modify the **src/56_io_mpi/m_hdr.F90** file.

1. Open the **src/56_io_mpi/m_hdr.F90** file.
   ```
   vi src/56_io_mpi/m_hdr.F90
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the edit mode and change **integer,public,parameter :: HDR_LATEST_HEADFORM = HDR_KNOWN_HEADFORMS(size(HDR_KNOWN_HEADFORMS))** to the following content:
```
integer,private,parameter :: HDR_KNOWN_HEADFORMS(1) = [123]
integer,public,parameter :: size_hdr_known_headforms = size(HDR_KNOWN_HEADFORMS)
integer,public,parameter :: HDR_LATEST_HEADFORM =
HDR_KNOWN_HEADFORMS(size_hdr_known_headforms)
```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 15 Modify the **src/62_iowfdenpot/m_io_screening.F90** file.

1. Open the **src/62_iowfdenpot/m_io_screening.F90** file.
```
vi src/62_iowfdenpot/m_io_screening.F90
```
2. Press **i** to enter the edit mode and change **integer,private,parameter :: HSCR_KNOWN_HEADFORMS(1) = [80]** to the following content:
```
integer,private,parameter :: HSCR_KNOWN_HEADFORMS(1) = [123]
integer,private,parameter :: size_hscr_known_headforms = size(HSCR_KNOWN_HEADFORMS)
integer,public,parameter :: HSCR_LATEST_HEADFORM =
HSCR_KNOWN_HEADFORMS(size_hscr_known_headforms)
```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 16 Perform the compilation and installation.
```
FC=mpif90 CC=mpicc CXX=mpicxx make multi multi_nprocs=32
```

NOTE

The **multi_nprocs** parameter indicates the number of cores for concurrent compilation. You can set this parameter based on the number of cores and application requirements. You are advised not to perform concurrent compilation for the initial compilation.

When you run compilation commands, the "unable to open MODULE file *.mod" error may be reported due to the path problem. Resolve this problem as follows:

Copy the **.mod** file to the **incs** directory, and then perform the compilation and installation. The **.mod** file is continuously generated during the compilation. Therefore, the **.mod** file needs to be copied for multiple times. If you do not want to perform the copy operation for multiple times, create a script. For example:

1. Create a script.
```
vi cp.sh
```
2. Press **i** to enter the edit mode and edit the file as follows:
```
#!/bin/bash

for i in `find -name *.mod`
do
        cp $i src/incs
done
```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Execute the script.
```
./cp.sh
```

Step 17 Add the environment variables.
```
export ABI_TESTS=/path/to/ABINIT/abinit-8.10.3/tests/
export ABI_TUTORIAL=$ABI_TESTS/tutorial/
export ABI_TUTORESPFN=$ABI_TESTS/tutorespfn/
export ABI_TUTOPARAL=$ABI_TESTS/tutoparal/
export ABI_TUTOPLUGS=$ABI_TESTS/tutoplugs/
export ABI_PSPDIR=$ABI_TESTS/Psps_for_tests/
export PATH=/path/to/ABINIT/abinit-8.10.3/build/src/98_main/:$PATH
```

----End

3.39.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test case directory.

```
cd /path/to/ABINIT/abinit-8.10.3/tests/tutorial/Input
mkdir Work_paral
```

Step 3 Go to the test case directory.

```
cd Work_paral
```

Step 4 Copy the test case file.

```
cp ../tbasepar_1.files .
cp ../tbasepar_1.in .
```

Step 5 Start the serial execution.

```
abinit < tbasepar_1.files > log 2> err
```

Step 6 Create a folder.

```
mkdir tmp
```

Step 7 Add a test case file.

1. Create a test case file.
   ```
   vi tbase1_x.files
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   tbasepar_1.in
   tbasepar_1.out
   tbasepar_1i
   tbasepar_1o
   ./tmp/tbasepar_1
   ../../../Psps_for_tests/HGH/82pb.4.hgh
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the computing.

```
mpirun --allow-run-as-root -np 32 ../../../../build/src/98_main/abinit < tbasepar_1.files
>&tbasepar_1.log &
```

Step 9 If the test must be performed on more than one node, edit the **hostfile** file.

1. Open the **hostfile** file.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and edit the file as follows:
   ```
   Node1
   Node2
   ```

   NOTE

   Node1 and **Node2** are the node names. Change them based on your requirements and ensure that the nodes can ping each other.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 If the test must be performed on more than one node, perform the computing on the nodes:

```
mpirun --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH --hostfile
hostfile -np 32  -N 16../../../../build/src/98 main/abinit < tbasepar_1.files >&tbasepar_1.log &
```

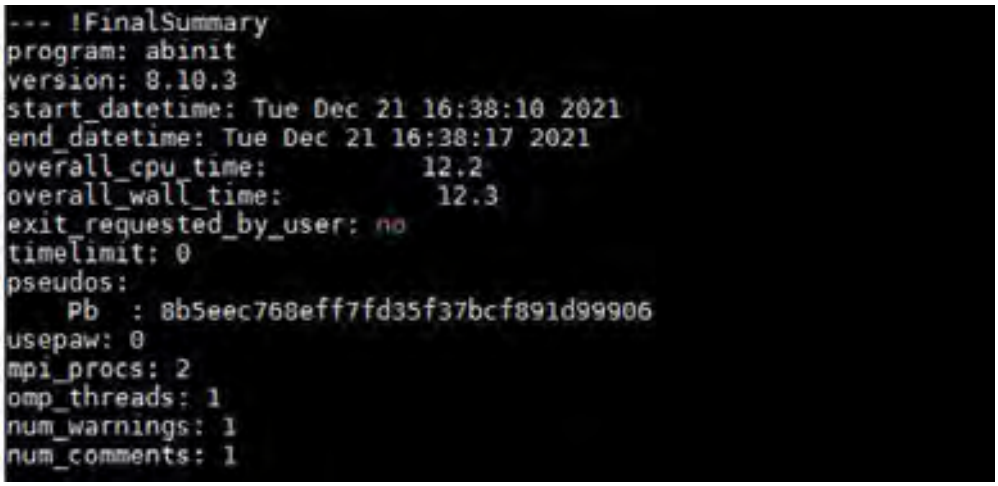
 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
--hostfile	Name of the node to be used.

Step 11 View the **tbasepar_1.log** file to obtain the time.

```
vi tbasepar_1.log
```

Information similar to the following is displayed:



----End

3.39.8 Troubleshooting

3.39.8.1 After a Dependency Package Is Configured, an Error Message Is Displayed Indicating that the Dynamic Link Library Cannot Be Found

Symptom

After a dependency package is configured, an error message is displayed indicating that the dynamic link library cannot be found.

Key Process and Cause Analysis

The dynamic link library is not loaded in the environment variables.

Conclusion and Solution

Check the environment variables. The NetCDF path does not have the corresponding library. The NetCDF-C path is separated from the NetCDF-Fortran

path. As a result, the NetCDF-Fortran library function cannot be found. Therefore, you need to **reload the environment variables**. If the program to be compiled uses Fortran and C at the same time, place the two dependency packages in the same directory.

3.40 AmberTools 21 Porting Guide (openEuler 21.03)

3.40.1 Introduction

AmberTools consists of several independently developed packages that work well by themselves, and with Amber itself. The suite can also be used to carry out complete molecular dynamics simulations (using NAB or mdgx), with either explicit water or generalized Born solvent models.

For more information about AmberTools, visit the **AmberTools official website**.

Programming languages: C/C++/Fortran

Brief description: A suite of programs for complete molecular dynamics simulations.

Open source license: GPL 3.0

Recommended Software Version

AmberTools 21

3.40.2 Environment Requirements

Hardware Requirements

Table 3-203 lists the hardware requirements.

Table 3-203 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-204 lists the software requirements.

Table 3-204 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Hyper MPI	1.1.1	https://www.hikunpeng.com/en/developer/hpc/hypermpi
AmberTools	21	http://ambermd.org/GetAmber.php#amber
Test cases	nab	Provided by the software.

OS Requirements

Table 3-205 lists the OS requirements.

Table 3-205 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

3.40.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AmberTools software porting.

Table 3-206 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See section "Installing Hyper MPI Using Source Code" in **Hyper MPI User Guide**.
3	/path/to/AMBER	Installation path of AmberTools.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

3.40.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-207 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See section "Installing Hyper MPI Using Source Code" in **Hyper MPI User Guide**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source  ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

3.40.5 Obtaining the Source Code

Procedure

Step 1 Download the AmberTools installation package.

http://ambermd.org/GetAmber.php#amber

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the AmberTools installation package to the */path/to/AMBER* directory on the server.

----End

3.40.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies using Yum.

```
yum install cmake* flex* bison* bison* -y
```

Step 3 Go to the */path/to/AMBER* directory.

```
cd /path/to/AMBER
```

Step 4 Decompress the installation package.

```
tar -xvf AmberTools21.tar.bz2
```

Step 5 Go to the directory generated after the decompression.

```
cd amber20_src
```

Step 6 Modify the compiler option.

1. Open the compiler.
   ```
   vim amber20_src/cmake/AmberCompilerConfig.cmake
   ```
2. Press **i** to enter the edit mode and change **gfortran** in line 136 to **flang**.

   NOTE

   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Go to the **build** directory of AMBER.

```
cd amber20_src/build
```

Step 8 Modify the configuration items in the **run_cmake** file.

1. Open the **run_cmake** file.
   ```
   vi run_cmake
   ```
2. Press **i** to enter the edit mode and modify the file as follows:

```
#  Assume this is Linux:
 CFLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110 -Wl,-z,muldefs -fuse-ld=lld" \
CXXFLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110 -Wno-c++11-narrowing -Wl,-z,muldefs -fuse-ld=lld" \
FCFLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110 -I/path/to/Hmpi111Bisheng210/hmpi/include -L/path/to/Hmpi111Bisheng210/hmpi/lib -fuse-ld=lld" \
cmake $AMBER_PREFIX/amber20_src \
 -DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber20 \
 -DCOMPILER=CLANG  \
 -DMPI=TRUE -DCUDA=FALSE -DINSTALL_TESTS=TRUE \
 -DDOWNLOAD_MINICONDA=FALSE -DMINICONDA_USE_PY3=FLASE \
 -DCMAKE_VERBOSE_MAKEFILE=ON \
 -DCMAKE_C_FLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110 -Wl,-z,muldefs -fuse-ld=lld" \
  -DCMAKE_CXX_FLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110 -Wno-c++11-narrowing -Wl,-z,muldefs -fuse-ld=lld" \
  -DCMAKE_Fortran_FLAGS="-O3 -mllvm -force-customized-pipeline -march=armv8.2-a -mcpu=tsv110
```

```
-I/path/to/Hmpi111Bisheng210/hmpi/include -L/path/to/Hmpi111Bisheng210/hmpi/lib -fuse-ld=lld" \
  2>&1 | tee  cmake.log
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Modify the **se_etimer.F** configuration file.

1. Open the **se_etimer.F** configuration file.
   vi ../../amber20_src/AmberTools/src/sebomd/se_etimer.F
2. Press **i** to enter the edit mode, deregister **tnow = 0.01*mclock()** in line 28, and add the following content:
   call cpu_time(tnow)

   NOTE

   After performing **Step 9.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Compile AmberTools.

./run_make

Step 11 Perform the installation.

make install

Step 12 Set the environment variables.

source */path/to/AMBER*/**amber20/amber.sh**

----End

3.40.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Amber21 test case directory.

cd */path/to/AMBER*/**amber21/AmberTools/benchmarks/nab**

Step 3 Modify the **bench_amber** file.

1. Open the **bench_amber** file.
   vi bench_amber
2. Press **i** to enter the edit mode and modify the file as follows:

```
#!/bin/sh
   sander=sander.MPI
   cat<<EOF > mdin
         benchmark a short md
   &cntrl
         igb=1, cut=20.0, rgbmax=20.0,
         tempi=50.0, temp0=100.0, tautp=0.4, ntt=1,
         ntb=0, nstlim=1000, ntpr=10,
         ntx=1, irest=0,
         ntc=2, ntf=2, tol=0.0000001,
   /
EOF
$DO_PARALLEL $sander -O -i mdin -p $1.top -c $1.mc.x -o $1.md.o
/bin/rm mdin restrt mdinfo
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Perform the computing.

 Copyright © Huawei Technologies Co., Ltd.

```
nab bench_md.nab
time ./bench_md gcn4dna
```

The following is an example of the output result.

```
|1st derivative timing summary:
|              constraints       0.000
|                    bonds       0.007
|                   angles       0.076
|                 torsions       0.089
|                 pairlist       2.544
|                  nonbond       0.036
|                gen. Born      34.360
|       Poisson Boltzmann       0.000
|                  3D-RISM       0.000
|                    Other       0.004
|------------------------
|                    Total      37.116

real    0m37.368s
user    0m37.327s
sys     0m0.012s
```

----End

3.40.8 Troubleshooting

3.40.8.1 Error "mclock, has not been explicitly declared" Reported When Compiling AmberTools

Symptom

The error "mclock, has not been explicitly declared" is reported when compiling AmberTools.

Key Process and Cause Analysis

None

Conclusion and Solution

Modify the **se_etimer.F** file.

1. Open the **se_etimer.F** file.
   ```
   vi /path/to/Ambertools/amber20_src/AmberTools/src/sebomd/se_etimer.F
   ```
2. Press **i** to enter the edit mode, deregister **tnow = 0.01*mclock()** in line 28, and add the following content:
   ```
   call cpu_time(tnow)
   ```

   NOTE

   After performing 1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

3.41 NAMD 2.13 Porting Guide (openEuler 21.03)

3.41.1 Introduction

NAMD is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems. NAMD uses the popular molecular graphics program VMD for simulation setup and trajectory analysis, but is also file-compatible with AMBER, CHARMM, and X-PLOR.

For more information about NAMD, visit the **NAMD official website**.

Programming language: C

Brief description: a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems.

Open source license: custom open source license

Recommended Software Version

NAMD 2.13

3.41.2 Environment Requirements

Hardware Requirements

Table 3-208 lists the hardware requirements.

Table 3-208 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-209 lists the software requirements.

Table 3-209 Software requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://www.hikunpeng.com/developer/boostkit/hpc/hypermpi**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
NAMD	2.13	https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD
Charm	6.9.0	https://charm.cs.illinois.edu/distrib/
TCL	8.5.9	http://www.ks.uiuc.edu/Research/namd/libraries/
FFTW	3.3.8	http://www.fftw.org/download.html
Test cases	Apoa1	https://www.ks.uiuc.edu/Research/namd/utilities/apoa1.tar.gz

OS Requirements

Table 3-210 lists the OS requirements.

Table 3-210 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	https://www.openeuler.org/zh/download
Kernel	4.17	Included in the OS image

3.41.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NAMD software porting.

Table 3-211 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.
3	/path/to/NAMD	Installation path of NAMD.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/TCL	Installation path of TCL.	actual paths planned during the installation process.
5	/path/to/CASE	Path of NAMD test cases.	
6	/path/to/FFTW	Installation path of FFTW.	

3.41.4 Configuring the Compilation Environment

3.41.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-212 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.
4	Installing FFTW	See **3.41.4.2 Installing FFTW**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

```
CC=clang CXX=clang++ FC=flang
```

----End

3.41.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration. Add an absolute path if an error is reported.

```
export CC=`which clang`
export CXX=`which clang++`
export FC=`which flang`
```

Step 5 Execute the **bootstrap.sh** script.

```
./bootstrap.sh
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/FFTW --enable-threads --enable-float
```

Step 7 Perform the compilation and installation.

```
make
make install
```

Step 8 Load the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.41.5 Obtaining the Source Code

Procedure

Step 1 Download the NAMD installation package.

https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD

Step 2 Use an SFTP tool to upload the NAMD installation package to the */path/to/NAMD* directory on the server.

----End

3.41.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/NAMD
```

Step 3 Create and go to the main program installation directory.

```
mkdir NAMD-2.13
cd NAMD-2.13
```

Step 4 Create a **build** directory.

```
mkdir build install
```

Step 5 Perform the configuration.

```
export BUILD_DIR=/path/to/NAMD/NAMD-2.13/build
export INSTALL_DIR=/path/to/NAMD/NAMD-2.13/install
```

Step 6 Set the environment variables of the compiler. If the setting fails, use the absolute path.

```
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
```

Step 7 Go to the **build** directory.

```
cd /path/to/NAMD/NAMD-2.13/build
```

Step 8 Upload the Charm package to the **build** directory.

Step 9 Install Charm.

```
tar -xvf charm-6.9.0.tar.gz
cd charm-6.9.0
mkdir ${INSTALL_DIR}/charm690/
cp src/arch/mpi-linux-x86_64 src/arch/mpi-linux-arm8 -rf
grep -rl 'm64' src/arch/mpi-linux-arm8 | xargs sed -i 's/\-m64//g'
./build charm++ mpi-linux-arm8 smp --with-production --destination=/path/to/NAMD/NAMD-2.13/
install/charm690/mpi-linux-arm8
```

Step 10 Go to the **build** directory.

```
cd /path/to/NAMD/NAMD-2.13/build
```

Step 11 Decompress the NAMD installation package.

```
tar -zxvf NAMD_2.13_Source.tar.gz
```

Step 12 Go to the directory generated after the decompression.

```
cd NAMD_2.13_Source
```

Step 13 Rename **Linux-ARM64.fftw3**.

```
cp arch/Linux-x86_64.fftw3 arch/Linux-ARM64.fftw3
```

Step 14 Go to the **NAMD-2.13** directory.

```
cd /path/to/NAMD/NAMD-2.13
```

Step 15 Use an SFTP tool to upload the tcl package to the current directory.

Step 16 Decompress the tcl file.

```
tar -xzvf tcl8.5.9-linux-arm64-threaded.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 17 Modify the **Make.charm** file.

1. Open the **Make.charm** file.
   cd */path/to/NAMD*/**NAMD-2.13/build/NAMD_2.13_Source**
   vi Make.charm
2. Press **i** to enter the edit mode and modify the file as follows:
   CHARMBASE = */path/to/NAMD*/NAMD-2.13/install/charm690
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 18 Modify the **Linux-ARM64-g++.arch** file.

1. Open the **Linux-ARM64-g++.arch** file.
   cd */path/to/NAMD*/**NAMD-2.13/build/NAMD_2.13_Source**
   vi arch/Linux-ARM64-g++.arch
2. Press **i** to enter the edit mode and modify the file as follows:

```
NAMD_ARCH = Linux-ARM64
  CHARMARCH = mpi-linux-arm8
  #FLOATOPTS = -O2 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer -mcpu=cortex-a9 -mtune=cortex-a9
  FLOATOPTS = -O2 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer
  CXX = clang++
  CXXOPTS = $(FLOATOPTS)
  CXXNOALIASOPTS = $(FLOATOPTS) -fno-strict-aliasing
  CC = clang
  COPTS = $(FLOATOPTS)
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 19 Configure NAMD.

./config Linux-ARM64-g++ --with-fftw3 --fftw-prefix */path/to/FFTW* **--with-tcl --tcl-prefix** */path/to/NAMD*/**NAMD-2.13/tcl8.5.9-linux-arm64-threaded --charm-arch mpi-linux-arm8**

Step 20 Go to the **Linux-ARM64-g++** directory.

cd Linux-ARM64-g++

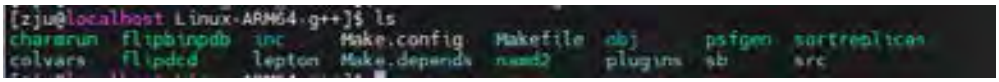
Step 21 Compile and install NAMD.

make

Step 22 Check whether an executable file is generated.

ls */path/to/NAMD*/**NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++/namd2**

If the **namd2** file exists, the compilation is successful.



----End

3.41.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variables of the NAMD executable file.

export PATH=*/path/to/NAMD*/**NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++/:$PATH**

Step 3 Go to the test case directory.

cd */path/to/CASE*

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Decompress the test case file.

```
tar -xf apoa1.tar.gz
```

Step 5 Modify the test case file.

1. Open the test case file.
   ```
   vi apoa1/apoa1.namd
   ```
2. Press **i** to enter the insert mode and add the following content to the end of the **apoa1/apoa1.namd** file:
   ```
   FFTWEstimate yes
   FFTWUseWisdom no
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the test.

```
cd apoa1
mpirun --allow-run-as-root --mca btl ^openib -np 1 namd2 +ppn 32 apoa1.namd +setcpuaffinity 2>&1 |tee -a namd.log
```

Step 7 If the test must be performed on more than one node, create a **hostfile** file.

1. Create **hostfile**.
   ```
   vi hostfile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   Node1
   Node2
   ```

   NOTE

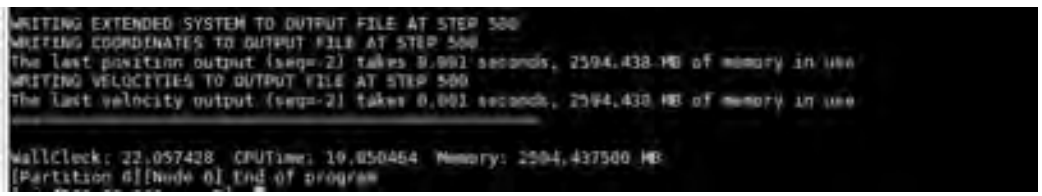
   Node1 and **Node2** are the node names. Change them based on your requirements and ensure that the nodes can ping each other.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 If the test must be performed on more than one node, run the following command to perform the test:

```
time mpirun --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH --hostfile hostfile --mca btl ^openib -np 2 -N 1 namd2 +ppn 32
apoa1.namd +setcpuaffinity 2>&1 |tee -a namd.log
```

Parameter	Description
-np	Total number of processes used in the test.
-N	Number of processes running on each server.
--hostfile	Name of the node to be used.

The following figure shows the test result:



----End

3.42 miniGhost 1.0 Porting Guide (openEuler 21.03)

3.42.1 Introduction

miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code. A broad range of scientific computation involves the use of difference stencils. In a parallel computing environment, this computation is typically implemented by decomposing the spacial domain, inducing a "halo exchange" of process-owned boundary data. This approach adheres to the Bulk Synchronous Parallel (BSP) model. Because commonly available architectures provide strong inter-node bandwidth relative to latency costs, many codes "bulk up" these messages by aggregating data into a message as a means of reducing the number of messages.

For more information on miniGhost, visit the **miniGhost official website**.

Programming language: Fortran

Brief description: miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code.

Open source license: LGPL 3.0

Recommended Software Version

miniGhost v1.0

3.42.2 Environment Requirements

Hardware Requirements

Table 3-213 lists the hardware requirements.

Table 3-213 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-214 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-214 Software Requirements

Item	Version	Download URL
BiSheng Compiler	1.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.1-aarch64-linux.tar.gz**
Hyper MPI	1.0.2	**https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.0.2-huawei.tar.gz**
miniGhost	v1.0	**https://github.com/Mantevo/miniGhost**

OS Requirements

Table 3-215 lists the OS requirements.

Table 3-215 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://www.openeuler.org/zh/download**
Kernel	4.17	Included in the OS image.

3.42.3 Paths for Software Porting

This chapter lists the software installation paths involved in the miniGhost software porting.

Table 3-216 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/BISHENG	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.
2	/path/to/HMPI	Installation path of the Hyper MPI.	See **Hyper MPI 1.0.2 Operation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/MINIGHOST	Installation path of miniGhost.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.42.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-217 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
2	Installing Hyper MPI	See **Hyper MPI 1.0.2 Operation Guide**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the insert mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/path/to/BISHENG/bin:/path/to/HMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/BISHENG/lib:/path/to/HMPI/lib:$LD_LIBRARY_PATH
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

When installing the BiSheng Compiler, specify the compilation parameter.

CC=clang CXX=clang++ FC=flang

----End

3.42.5 Obtaining the Source Code

Procedure

Step 1 Download the miniGhost installation package.

https://github.com/Mantevo/miniGhost

Step 2 Use an SFTP tool to upload the miniGhost installation package to the */path/to/MINIGHOST* directory on the server.

----End

3.42.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
unzip miniGhost-master.zip
```

Step 3 Go to the directory generated after the decompression.

```
cd /path/to/MINIGHOST/miniGhost-master/ref
```

Step 4 Create a **Makefile** file.

```
mv Makefile Makefile.bak
cp makefile.mpi.gnu Makefile
```

Step 5 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode and modify the file as follows:
   ```
   PROTOCOL = -D_MG_MPI
   MPI_LOC = /path/TO/OPENMPI
   MPI_INCLUDE = -I$(MPI_LOC)/include
   # State precision: -D_REAL4 and/or -D_REAL8.
   REAL_PRECISION = -D_MG_REAL8
   # Integer precision: -D_INT4 and/or -D_INT8.
   INT_PRECISION = -D_MG_INT4
   # Compilers
   FC=$(MPI_LOC)/bin/mpif90
   CC=$(MPI_LOC)/bin/mpicc
   CFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # C main calling Fortran subroutine:
   CFLAGS += -Df2c_ -fsigned-char
   CFLAGS += $(MPI_INCLUDE)
   FFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # Optimization
   OPT_F =-O3
   FFLAGS += $(OPT_F)
   #FFLAGS += $(MPI_INCLUDE)
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
# Free-form Fortran source code:
FFLAGS += -ffree-form -ffree-line-length-none
# Array bounds checking: (expensive!)
#FFLAGS += -fbounds-check
# Compile to include checkpointing capability.
FFLAGS += -D_MG_CHECKPT
NCPATH=
LNCDF=-L$(NCPATH)/lib
LD=$(CC)
LDFLAGS=$(CFLAGS) $(FFLAGS) -L/path/to/HMPI/lib -lmpi_mpifh
#LDFLAGS=$(CFLAGS) $(FFLAGS)
#LDFLAGS=$(FFLAGS)
LIBS=
include make_targets
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation.

```
make
```

----End

3.42.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Start miniGhost.x.

```
mpirun -mca btl ^openib -x LD_LIBRARY_PATH --allow-run-as-root -np 96 ./miniGhost.x
```

The following figure shows the test result in the **result-yaml** file:

----End

3.42.8 Troubleshooting

3.42.8.1 Error Reported When Running make

Symptom

An error is reported when running make.

```
undefined reference to `f90io_src_info03a'
undefined reference to symbol '__ps_powi_1'
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)
```

Key Process and Cause Analysis

None

Conclusion and Solution

Add the following content to the **LIBS=** line of the **Makefile** file in **Step 5**.

```
LIBS=-L/path/to/BISHENG/lib -lflang -lpgmath
```

3.43 MiniFE 2.2.0 Porting Guide (CentOS 7.6)

3.43.1 Introduction

MiniFE is a proxy application for unstructured implicit finite element code. It is similar to HPCCG and pHPCCG but provides a much more complete vertical covering of the steps in this class of applications.

For more information, see **official MiniFE website**.

Programming language: C++

Brief description: a proxy application for unstructured implicit finite element code

Open source license: LGPL-3.0

Recommended Software Version

MiniFE 2.2.0

3.43.2 Environment Requirements

Hardware Requirements

Table 3-218 lists the hardware requirements.

Table 3-218 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-219 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-219 Software requirements

Item	Version	Download Address
MiniFE	2.2.0	https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz

OS Requirements

Table 3-220 lists the OS requirements.

Table 3-220 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.43.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MiniFE software porting.

Table 3-221 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/MINIFE	Installation path of MiniFE	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.43.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-222 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.43.5 Obtaining the Source Code

Procedure

Step 1 Download the MiniFE installation package **miniFE-2.2.0.zip**.

https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz

Step 2 Use an SFTP tool to upload the MiniFE installation package to the */path/to/MINIFE* directory on the server.

----End

3.43.6 Compiling and Installing MiniFE

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

```
tar -xvf miniFE-2.2.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/miniFE/miniFE-2.2.0/ref/src
```

Step 4 Run the following command to perform compilation:

```
make
```

----End

3.43.7 Running and Verifying MiniFE

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to run **miniFE.x** to generate a YAML file:

```
mpirun --allow-run-as-root -n 96 ./miniFE.x nx=300 verify_solution=1
```

NOTE

nx indicates the size of each problem dimension.

 Copyright © Huawei Technologies Co., Ltd.

Check the values of **Total CG Mflops: 34377.9** and **Total Program Time: 29.567** in the generated YAML file.

```
Iterations: 200
Final Resid Norm: 0.00247174
WAXPY Time: 1.36617
WAXPY Flops: 4.8762e+10
WAXPY Mflops: 35692.6
DOT Time: 1.10191
DOT Flops: 2.16e+10
DOT Mflops: 19602.3
MATVEC Time: 8.12439
MATVEC Flops: 2.94036e+11
MATVEC Mflops: 36191.7
Total:
Total CG Time: 10.5998
Total CG Flops: 3.64398e+11
Total CG Mflops: 34377.9
Time per iteration: 0.0529988
Global All-RSS (kB): 15019072
Global Max-RSS (kB): 166592
Total Program Time: 29.567
```

----End

3.44 miniGhost v1.0 Porting Guide (CentOS 7.6)

3.44.1 Introduction

miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code. A broad range of scientific computation involves the use of difference stencils. In a parallel computing environment, this computation is typically implemented by decomposing the spacial domain, inducing a "halo exchange" of process-owned boundary data. This approach adheres to the Bulk Synchronous Parallel (BSP) model. Because commonly available architectures provide strong inter-node bandwidth relative to latency costs, many codes 'bulk up' these messages by aggregating data into a message as a means of reducing the number of messages.

For more information, visit the **official miniGhost website**.

Programming language: Fortran

Brief description: miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code.

Open source license: LGPL-3.0

Recommended Software Version

miniGhost v1.0

3.44.2 Environment Requirements

Hardware Requirements

Table 3-223 lists the hardware requirements.

Table 3-223 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-224 lists the software requirements.

Table 3-224 Software requirements

Item	Version	Download Address
miniGhost	v1.0	https://github.com/Mantevo/miniGhost

OS Requirements

Table 3-225 lists the OS requirements.

Table 3-225 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

3.44.3 Paths for Software Porting

This chapter lists the software installation paths involved in the miniGhost software porting.

Table 3-226 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/MINIGHOST	Installation path of miniGhost	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.44.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-227 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.44.5 Obtaining the Source Code

Procedure

Step 1 Download the miniGhost installation package **miniGhost-master.zip**.

https://github.com/Mantevo/miniGhost

Step 2 Use an SFTP tool to upload the miniGhost installation package to the */path/to/MINIGHOST* directory on the server.

----End

3.44.6 Compiling and Installing miniGhost

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the miniGhost installation package:

```
unzip miniGhost-master.zip
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/MINIGHOST/miniGhost-master/ref
```

Step 4 Run the following commands to create a **Makeflie** file:

```
mv Makefile Makefile.bak
cp makefile.mpi.gnu Makefile
```

Step 5 Run the following command to modify the **Makeflie** file:

1. Open **Makefile**.

   ```
   vi Makeflie
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   ```
   PROTOCOL = -D_MG_MPI
   MPI_LOC = /path/TO/OPENMPI
   MPI_INCLUDE = -I$(MPI_LOC)/include
   # State precision: -D_REAL4 and/or -D_REAL8.
   REAL_PRECISION = -D_MG_REAL8
   # Integer precision: -D_INT4 and/or -D_INT8.
   INT_PRECISION = -D_MG_INT4
   # Compilers
   FC=$(MPI_LOC)/bin/mpif90
   CC=$(MPI_LOC)/bin/mpicc
   CFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # C main calling Fortran subroutine:
   CFLAGS += -Df2c_ -fsigned-char
   CFLAGS += $(MPI_INCLUDE)
   FFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # Optimization
   OPT_F =-O3
   FFLAGS += $(OPT_F)
   #FFLAGS += $(MPI_INCLUDE)
   # Free-form Fortran source code:
   FFLAGS += -ffree-form -ffree-line-length-none
   # Array bounds checking: (expensive!)
   #FFLAGS += -fbounds-check
   # Compile to include checkpointing capability.
   FFLAGS += -D_MG_CHECKPT
   NCPATH=
   LNCDF=-L$(NCPATH)/lib
   LD=$(CC)
   LDFLAGS=$(CFLAGS) $(FFLAGS)  -L/path/to/GNU/gcc9.1/lib64/ -lgfortran -L/path/to/OPENMPI/
   openmpi-4.0.1-gcc9.1/lib -lmpi_mpifh
   #LDFLAGS=$(CFLAGS) $(FFLAGS)
   #LDFLAGS=$(FFLAGS)
   LIBS=
   include make_targets
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to perform compilation:

```
make
```

NOTE

Parallel compilation (by using the **make -j** command) is not supported. Run the **make** command to perform compilation.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.44.7 Running and Verifying miniGhost

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to start miniGhost.x:

```
mpirun -mca btl ^openib -x LD_LIBRARY_PATH --allow-run-as-root -np 96 ./miniGhost.x
```

View the value of each item in the generated **results.yaml** file.

Figure 3-17 shows the command output.

Figure 3-17 An example result

```
GFLOPS_Total:                   1.991E+01
GFLOPS_per_process:             1.556E-01
GFLOPS_stencil_Total:           1.991E+01
GFLOPS_stencil_per_process:     1.556E-01
FLOPS_Total:                    3.200E+11
FLOPS_Sums:                     2.560E+11
FLOPS_Divide:                   6.400E+10
Number_spikes:                  1
```

----End

3.45 ELK 8.4.30 Porting Guide (Kylin V10)

3.45.1 Introduction

ELK is an all-electron full-potential linearised augmented-planewave (FP-LAPW) code. Designed to be as developer friendly as possible so that new developments in the field of density functional theory (DFT) can be added quickly and reliably.

For more information, visit the **official ELK website**.

Programming language: Fortran

Brief description: An all-electron full-potential linearised augmented-planewave (FP-LAPW) code.

Open source license: custom open source license

Recommended Software Version

ELK 8.4.30

3.45.2 Environment Requirements

Hardware Requirements

Table 3-228 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-228 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-229 lists the software requirements.

Table 3-229 Software requirements

Item	Version	Download URL
ELK	8.4.30	https://nchc.dl.sourceforge.net/project/elk/elk-8.4.30.tgz

OS Requirements

Table 3-230 lists the OS requirements.

Table 3-230 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.45.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ELK software porting.

Table 3-231 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/elk	Installation path of ELK	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.45.4 Configuring the Compilation Environment

3.45.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-232 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the dependencies	See **3.45.4.2 Installing the Dependencies**.

3.45.4.2 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install blas-devel lapack-devel fftw-devel
```

----End

3.45.5 Obtaining the Source Code

Procedure

Step 1 Download the ELK installation package.

https://nchc.dl.sourceforge.net/project/elk/elk-8.4.30.tgz

Step 2 Use an SFTP tool to upload the ELK installation package to the */path/to/elk* directory on the server.

----End

3.45.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/elk
```

Step 3 Decompress the installation package.

```
tar xvf elk-8.4.30.tgz
```

Step 4 Go to the directory generated after the decompression.

```
cd elk-8.4.30
```

Step 5 Install ELK.

```
./setup
```

Type **2** and press **Enter**.

```
Choose compiler:

  1. Intel Fortran (ifort) with OpenMP
  2. GNU Fortran (gfortran) with OpenMP
  3. Portland Group Fortran (pgf90) with OpenMP
  4. G95 (g95)
  5. NAG Fortran (nagfor)
  6. IBM Fortran (xlf90_r) with OpenMP

 20. Intel Fortran profiling (debug only)
 21. GNU Fortran code check (debug only)
 22. G95 code check (debug only)

  o. Other       x. Exit

2
```

Step 6 Modify the **make.inc** file.

1. Open **make.inc**.
   ```
   vi make.inc
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   - Add **-lfftw3** to the end of lines 4 and 6.

 Copyright © Huawei Technologies Co., Ltd.

```
1
2 MAKE = make
3 F90 = gfortran
4 F90_OPTS = -O3 -ffast-math -funroll-loops -fopenmp -lfftw3
5 F77 = gfortran
6 F77_OPTS = -O3 -ffast-math -funroll-loops -fopenmp -lfftw3
7 AR = ar
```

- Comment out line 55.
- Uncomment line 61.

```
55 #SRC_FFT = zfftifc.f90
56 # To use a different FFT library, copy the relevant library or include files
57 # the elk/src directory, add any required compiler flags, uncomment the
58 # appropriate lines below and run 'make clean' followed by 'make'.
59 #
60 # FFTW, note that you can link the library with the -lfftw3 compiler flag
61 SRC_FFT = zfftifc_fftw.f90
62 #
63 # MKL, the file mkl_dfti.f90 should be copied to the elk/src directory
64 #SRC_FFT = mkl_dfti.f90 zfftifc_mkl.f90
65 #
```

NOTE

After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the compilation and installation.

```
make all
```

Step 8 Load the environment variable.

```
export PATH=/path/to/elk/elk-8.4.30/src:$PATH
```

----End

3.45.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify ELK.

```
elk --version
```

Information similar to the following is displayed:

```
[root@localhost elk-8.4.30]# elk --version
Elk code version 8.4.30 started
Error(readinput): error opening elk.in
```

Step 3 Execute the 31 test cases provided by the software.

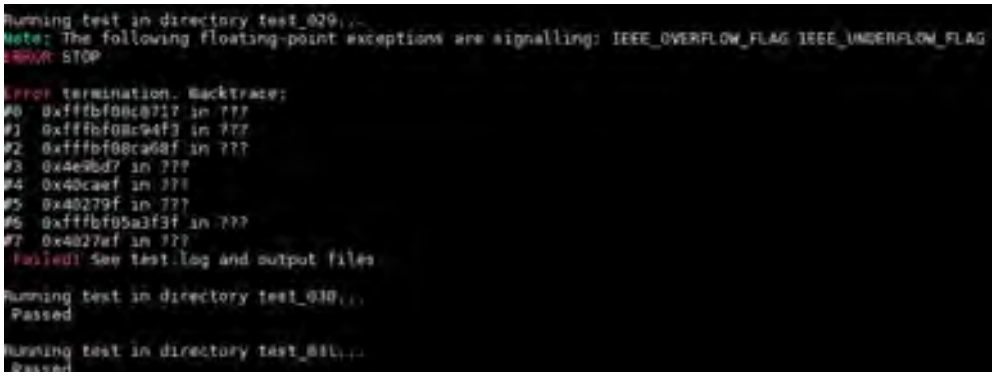
```
cd /path/to/elk/elk-8.4.30/
make test
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

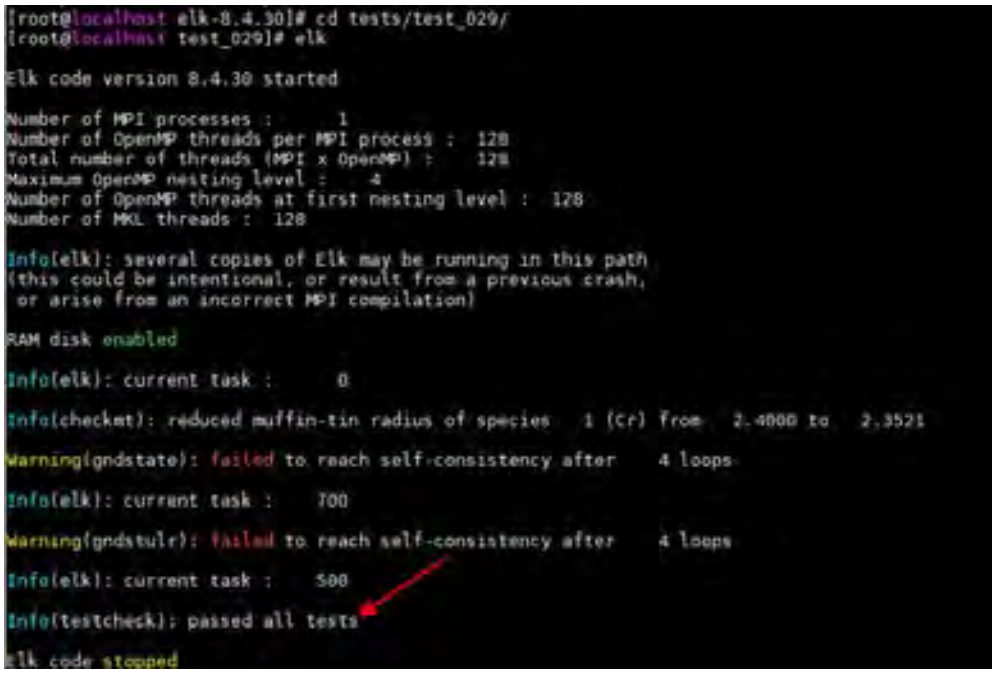
```
[root@localhost elk-8.4.30]# make test
cd tests; ./test.sh

Running test in directory test_001...
 Passed

Running test in directory test_002...
 Passed

Running test in directory test_003...
 Passed

Running test in directory test_004...
 Passed

Running test in directory test_005...
 Passed

Running test in directory test_006...
 Passed

Running test in directory test_007...
 Passed

Running test in directory test_008...
 Passed

Running test in directory test_009...
 Passed
```

The 29th test fails, as shown in the following figure:



Go to the 29th **test** file and run the ELK test again.

```
cd tests/test_029
elk
```



----End

3.46 MOPAC 22.0.4 Porting Guide (Kylin V10)

3.46.1 Introduction

Molecular Orbital PACkage (MOPAC) is a semiempirical quantum chemistry program based on Dewar and Thiel's NDDO approximation.

For more information, visit the **official MOPAC website**.

Programming language: Fortran

Brief description: A semiempirical quantum chemistry program.

Open source license: custom open source license

Recommended Software Version

MOPAC 22.0.4

3.46.2 Environment Requirements

Hardware Requirements

Table 3-233 lists the hardware requirements.

Table 3-233 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-234 lists the software requirements.

Table 3-234 Software Requirements

Item	Version	Download URL
MOPAC	22.0.4	https://github.com/openmopac/mopac/archive/refs/tags/v22.0.4.tar.gz

OS Requirements

Table 3-235 lists the OS requirements.

Table 3-235 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.46.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MOPAC software porting.

Table 3-236 Paths for Software Porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/mopac	Installation path of MOPAC	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.46.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-237 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

3.46.5 Obtaining the Source Code

Procedure

Step 1 Download the MOPAC installation package.

https://github.com/openmopac/mopac/archive/refs/tags/v22.0.4.tar.gz

Step 2 Use an SFTP tool to upload the MOPAC installation package to the */path/to/mopac* directory on the server.

----End

3.46.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/mopac
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the installation package.

```
tar xvf v22.0.4.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd mopac-22.0.4
```

Step 5 Create a **build** directory and go to the directory.

```
mkdir build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/mopac
make
make install
```

----End

3.46.7 Running and Verification

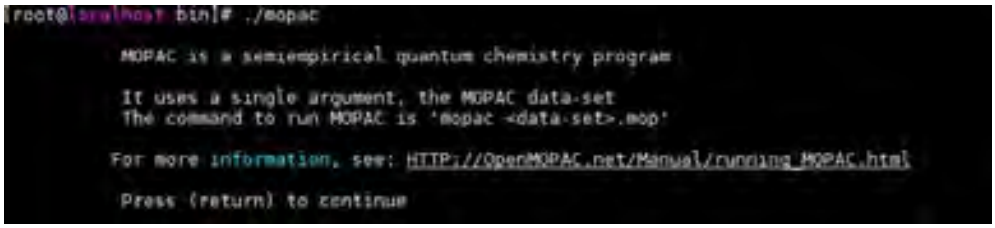
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify MOPAC.

```
export LD_LIBRARY_PATH=/path/to/mopac/lib64:$LD_LIBRARY_PATH
cd /path/to/mopac
./mopac
```

Information similar to that in the following figure is displayed:



----End

3.47 QMCPACK 3.14.0 Porting Guide (Kylin V10)

3.47.1 Introduction

The main repository for QMCPACK is an open-source production level many-body ab initio Quantum Monte Carlo code for computing the electronic structure of atoms, molecules, and solids.

For more information, visit the **official QMCPACK website**.

Programming language: C++/Python

Brief description: used to compute the electronic structure of atoms, molecules, and solids.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

QMCPACK 3.14.0

3.47.2 Environment Requirements

Hardware Requirements

Table 3-238 lists the hardware requirements.

Table 3-238 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-239 lists the software requirements.

Table 3-239 Software Requirements

Item	Version	Download URL
QMCPACK	3.14.0	https://github.com/QMCPACK/qmcpack/archive/refs/tags/v3.14.0.tar.gz

OS Requirements

Table 3-240 lists the OS requirements.

Table 3-240 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.47.3 Paths for Software Porting

This chapter lists the software installation paths involved in the QMCPACK software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-241 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/qmcpack	Installation path of QMCPACK	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.47.4 Configuring the Compilation Environment

3.47.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-242 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.
3	Installing Hyper MPI	See "Compiling the Hyper MPI Software Package Using GCC" in the "Installing Hyper MPI Using Source Code" section of the **Hyper MPI User Guide**
4	Installing the dependencies	See **3.47.4.2 Installing the Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

3.47.4.2 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install blas-devel lapack-devel fftw-devel boost-devel hdf5-devel
```

----End

3.47.5 Obtaining the Source Code

Procedure

Step 1 Download the QMCPACK installation package.

https://github.com/QMCPACK/qmcpack/archive/refs/tags/v3.14.0.tar.gz

Step 2 Use an SFTP tool to upload the QMCPACK installation package to the */path/to/qmcpack* directory on the server.

----End

3.47.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/qmcpack
```

Step 3 Decompress the installation package.

```
tar xvf v3.14.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd qmcpack-3.14.0
```

Step 5 Create a **build** directory and go to the directory.

```
mkdir build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/qmcpack -DCMAKE_C_COMPILER=mpicc -
DCMAKE_CXX_COMPILER=mpicxx
make -j
make install
```

----End

3.47.7 Running and Verification

Procedure

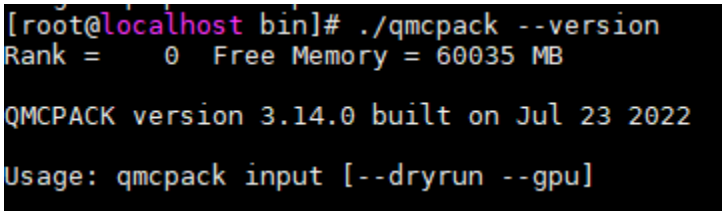
Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Verify QMCPACK.

```
cd /path/to/qmcpack/bin
./qmcpack --version
```

Information similar to that in the following figure is displayed:



----End

3.48 OpenCV 4.5.5 Porting Guide (openEuler 20.03)

3.48.1 Introduction

OpenCV is a cross-platform computer vision library distributed under the open-source BSD license, which can run on Linux, Windows, Android, and Mac OS. It is lightweight and efficient. It consists of a series of C functions and a few C++ classes. It provides interfaces for Python, Ruby, and MATLAB, implementing many common algorithms for image processing and computer vision.

For more information, visit the **OpenCV official website**.

Programming language: C/C++

Brief description: A cross-platform computer vision library.

Open source license: BSD

Recommended Software Version

OpenCV 4.5.5

3.48.2 Environment Requirements

Hardware Requirements

Table 3-243 lists the hardware requirements.

Table 3-243 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-244 Software requirements

Item	Version	Download URL
OpenCV	4.5.5	**https://github.com/opencv/opencv/archive/refs/tags/4.5.5.tar.gz**

OS Requirements

Table 3-245 lists the OS requirements.

Table 3-245 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	**https://repo.openeuler.org/openEuler-20.03-LTS-SP1/**
Kernel	4.19	Included in the OS image

3.48.3 Paths for Software Porting

This chapter lists the software installation paths involved in the OpenCV software porting.

Table 3-246 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/opencv	Installation path of OpenCV	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

3.48.4 Configuring the Compilation Environment

Configuration Process

Table 3-247 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

3.48.5 Obtaining the Source Code

Procedure

Step 1 Download the OpenCV installation package **opencv-4.5.5.tar.gz**.

https://github.com/opencv/opencv/archive/refs/tags/4.5.5.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/opencv* directory on the server.

----End

3.48.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the OpenCV dependency libraries and tools.

```
yum install git gcc-c++ cmake -y
```

Step 3 Go to the source package directory and decompress the source package.

```
cd /path/to/opencv
tar xvf 4.5.5.tar.gz
cd opencv-4.5.5
```

Step 4 Create a build directory.

```
mkdir build
cd build/
```

Step 5 Perform the compilation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/opencv/opencv
make -j8 && make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.48.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 View the library and header files generated during OpenCV installation.

ll */path/to/opencv*/**opencv/lib**

If the following information is displayed, the installation is successful:

----End

3.49 Phono3py 2.3.2 Porting Guide (Kylin V10)

3.49.1 Introduction

Phono3py calculates phonon-phonon interaction and related properties using the supercell approach. For example, the following physical values are obtained:

- Lattice thermal conductivity by relaxation time approximation and direct-solution of phonon Boltzmann equation and the Wigner transport equation
- Cumulative lattice thermal conductivity and related properties
- Imaginary and real parts of self energy (Phonon lifetime/linewidth)
- Joint density of states (JDOS) and weighted-JDOS
- Spectral function
- Built-in interfaces for VASP, QE (pw), CRYSTAL, TURBOMOLE, and Abinit (see Interfaces to calculators)

For more information, visit the **Phono3py official website**.

Brief description: Calculates phonon-phonon interaction and related properties using the supercell approach.

Programming language: Python

Open source license: BSD

Recommended Software Version

Phono3py 2.3.2

3.49.2 Environment Requirements

Hardware Requirements

Table 3-248 lists the hardware requirements.

Table 3-248 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 3-249 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz
Python	3.7.5	https://www.python.org/downloads/
Phono3py	2.3.2	https://github.com/phonopy/phono3py/archive/refs/tags/v2.3.2.tar.gz

OS Requirements

Table 3-250 lists the OS requirements.

Table 3-250 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

3.49.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Phono3py software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-251 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/PHONO3PY	Installation path of Phono3py	

3.49.4 Configuring the Compilation Environment

3.49.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-252 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.49.4.2 Installing OpenBLAS**.

3.49.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the OpenBLAS installation package.

```
cd /path/to/OPENBLAS
wget https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.18.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the software package.

```
tar -zxvf v0.3.18.tar.gz
```

Step 4 Go to the OpenBLAS installation directory.

```
cd OpenBLAS-0.3.18
```

Step 5 Compile and install the OpenBLAS source code.

```
make -j
make PREFIX=/path/to/OPENBLAS/openblas install
```

----End

3.49.5 Obtaining the Source Code

Procedure

Step 1 Download the Phono3py installation package **v2.3.2.tar.gz**.

https://github.com/phonopy/phono3py/archive/refs/tags/v2.3.2.tar.gz

Step 2 Use an SFTP tool to upload the Phono3py installation package to the */path/to/PHONO3PY* directory on the server.

----End

3.49.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 (Optional) Configure the virtual environment if needed.

```
pip install virtualenv
virtualenv phono3py
source phono3py/bin/activate
```

Step 3 Go to the source package directory and decompress the source package.

```
cd /path/to/PHONO3PY && tar -zxf v2.3.2.tar.gz
```

Step 4 Install the Python dependencies.

```
pip3 install numpy
pip3 install scipy
pip3 install phonopy
```

Step 5 Add configuration files.

```
cd phono3py-2.3.2/
echo "[phono3py]" > site.cfg
echo "extra_compile_args = -fopenmp" >> site.cfg
echo "extra_link_args = -lgomp" >> site.cfg
```

Step 6 Import the OpenBLAS header file configuration.

```
export C_INCLUDE_PATH=/path/to/OPENBLAS/openblas/include
export CPLUS_INCLUDE_PATH=/path/to/OPENBLAS/openblas/include
```

Step 7 Install Phono3py.

```
python3 setup.py install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.49.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
PHONO3PY.py
```

If the following information is displayed, the installation is successful:

----End

3.50 QDK 2.3.12 Porting Guide (Kylin V10)

3.50.1 Introduction

QDK is used to build QPKG files/applications for QNAP Turbo NAS. A QPKG file makes it easy for anyone to install and remove packages. It also gives a package maintainer almost total control on how the package is installed on the NAS. The major design goal of QDK is to make it easy for the package maintainer to create simple QPKG files and at the same time also support more advanced packages. QDK started out as a simple modification of the first official release of the QPKG SDK, but now supersedes it. It includes many new features like architecture check at installation, support for digital signatures, different compression algorithms, a comprehensive option to check that other required QPKG packages are installed (or that conflicting packages are not installed), and a powerful build script.

For more information, visit the **QDK page at GitHub**.

Programming language: Shell

Brief description: molecular simulation code.

Open source license: GPL

Recommended Software Version

QDK 2.3.12

3.50.2 Environment Requirements

Hardware Requirements

Table 3-253 lists the hardware requirements.

Table 3-253 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 3-254 Software requirements

Item	Version	Download URL
QDK	2.3.12	https://github.com/qnap-dev/QDK/archive/refs/tags/v2.3.12.tar.gz

OS Requirements

Table 3-255 lists the OS requirements.

Table 3-255 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

3.50.3 Paths for Software Porting

This chapter lists the software installation paths involved in the QDK software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-256 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/QDK	Installation path of QDK	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.50.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-257 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.

3.50.5 Obtaining the Source Code

Procedure

Step 1 Download the QDK installation package **v2.3.12.tar.gz**.

https://github.com/qnap-dev/QDK/archive/refs/tags/v2.3.12.tar.gz

Step 2 Use an SFTP tool to upload the QDK installation package to the */path/to/QDK* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.50.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory and decompress the source package.

```
cd /path/to/QDK && tar xvf v2.3.12.tar.gz
```

Step 3 Modify the installation script.

1. Open the **./InstallToUbuntu.sh** file.
   ```
   vim ./InstallToUbuntu.sh
   ```
2. Press **i** to enter the insert mode, comment out lines 9 and 10, and add line 11.

   Before the modification:

   ```
   9    apt-get update
   10   apt-get install -y build-essential wget bsdmainutils curl python openssl rsync
   ```

   After the modification:

   ```
   9    #apt-get update
   10   #apt-get install -y build-essential wget bsdmainutils curl python openssl rsync
   11   yum install -y  wget  curl python openssl rsync
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Compile and install QDK.

```
./InstallToUbuntu.sh
./InstallToUbuntu.sh install
```

----End

3.50.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the QDK directory.

```
cd /usr/share/QDK/bin
```

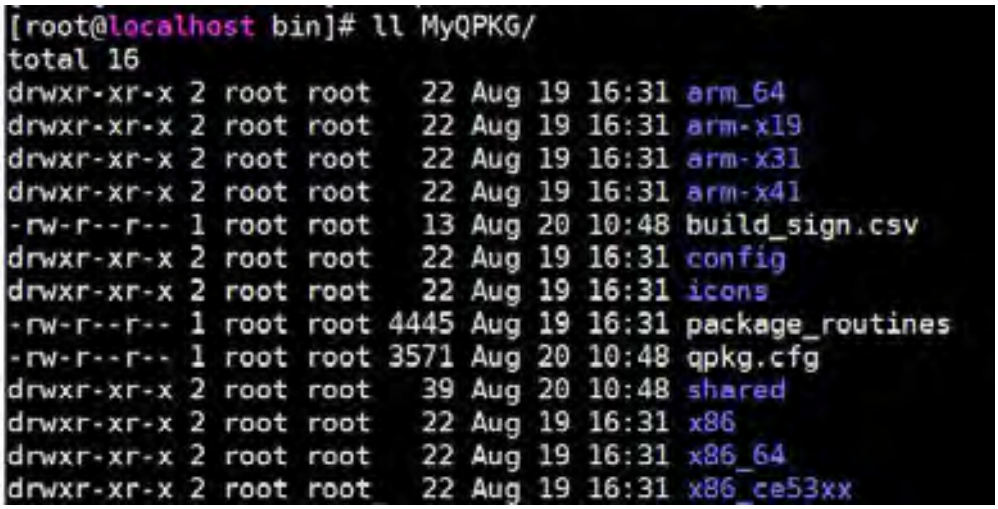
Step 3 Create a QPKG environment.

```
./qbuild --create-env MyQPKG
ll MyQPKG/
```

Figure shows an example of the command output.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-18 An example result



----End

3.51 Towhee 8.2.3 Porting Guide (Kylin V10)

3.51.1 Introduction

Towhee is a Monte Carlo molecular simulation code originally designed for the prediction of fluid phase equilibria using atom-based force fields and the Gibbs ensemble with particular attention paid to algorithms addressing molecule conformation sampling. The code has subsequently been extended to several ensembles, many different force fields, and solid (or at least porous) phases.

For more information, visit the **Towhee official website**.

Programming language: Fortran

Brief description: molecular simulation code.

Open source license: GPLv2

Recommended Software Version

Towhee 8.2.3

3.51.2 Environment Requirements

Hardware Requirements

Table 3-258 lists the hardware requirements.

Table 3-258 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 3-259 Software requirements

Item	Version	Download URL
Towhee	8.2.3	https://sourceforge.net/projects/towhee/files/towhee/towhee-8.2.3.tar.gz

OS Requirements

Table 3-260 lists the OS requirements.

Table 3-260 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

3.51.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Towhee software porting.

Table 3-261 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/TOWHEE	Installation path of Towhee	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.51.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-262 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.

3.51.5 Obtaining the Source Code

Procedure

Step 1 Download the Towhee installation package **towhee-8.2.3.tar.gz**.

https://sourceforge.net/projects/towhee/files/towhee/towhee-8.2.3.tar.gz

Step 2 Use an SFTP tool to upload the Towhee installation package to the */path/to/TOWHEE* directory on the server.

----End

3.51.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the source package directory and decompress the source package.

```
cd /path/to/TOWHEE && tar -zxf towhee-8.2.3.tar.gz
```

Step 3 Compile and install Towhee.

```
cd towhee-8.2.3
./configure
cd Source/
make towhee
```

----End

3.51.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Generate the directory.

```
ln -s /path/to/TOWHEE/towhee-8.2.3 /towheebase
```

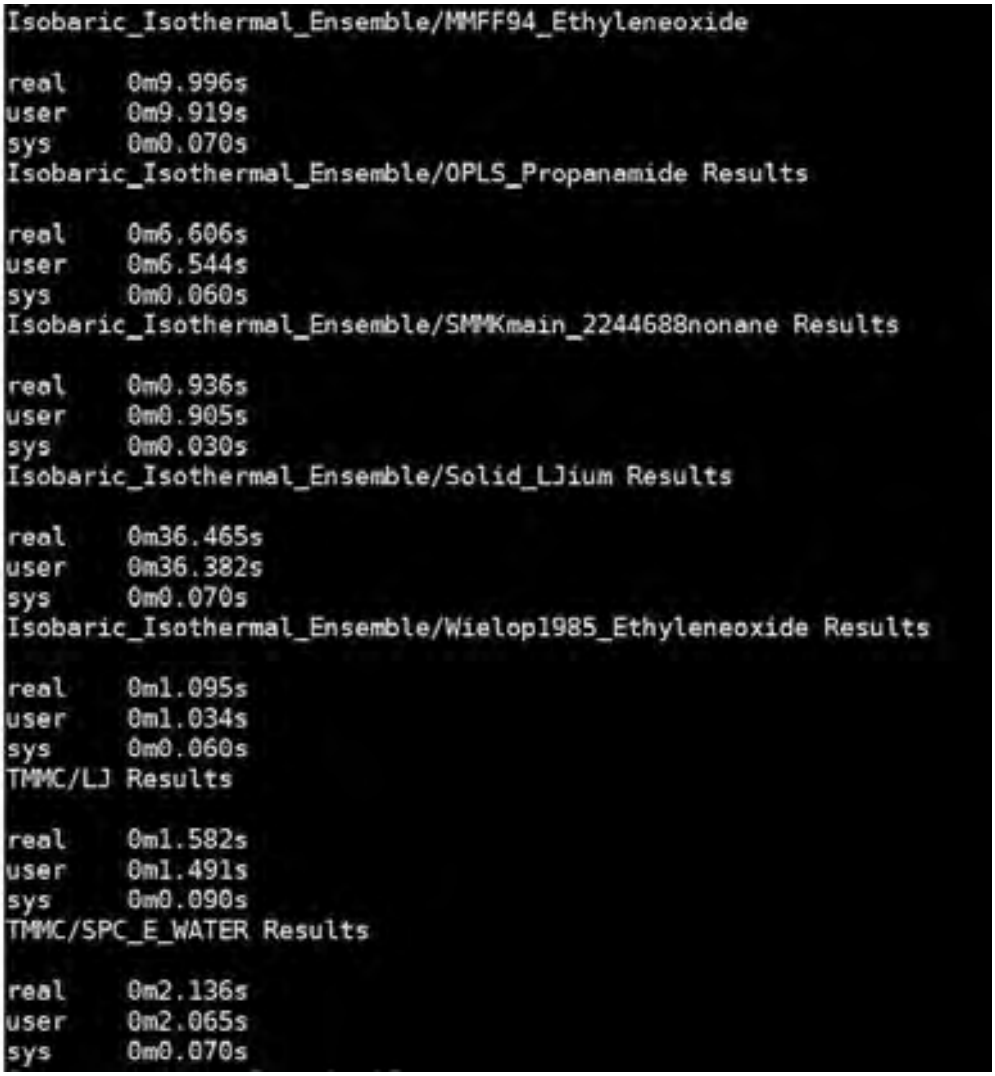
Step 3 Execute the test case.

```
cd /path/to/TOWHEE/towhee-8.2.3/Examples
./run_serial_test
```

Figure shows an example of the command output.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-19 An example result



----End

3.52 CosmoMC June2021 Porting Guide (Kylin V10)

3.52.1 Introduction

CosmoMC is a Fortran 2008 Markov-Chain Monte-Carlo (MCMC) engine for exploring cosmological parameter space, together with Fortran and python code for analysing Monte-Carlo samples and importance sampling (plus a suite of scripts for building grids of runs, plotting and presenting results).

For more information, visit the **CosmoMC page at GitHub**.

Programming languages: Fortran/Python

Brief description: An engine for exploring cosmological parameter space.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

CosmoMC June2021

3.52.2 Environment Requirements

Hardware Requirements

Table 3-263 lists the hardware requirements.

Table 3-263 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 3-264 Software requirements

Item	Version	Download URL
CosmoMC	June2021	https://github.com/cmbant/CosmoMC/archive/refs/tags/June2021.tar.gz

OS Requirements

Table 3-265 lists the OS requirements.

Table 3-265 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.52.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CosmoMC software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-266 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/CosmoMC	Installation path of CosmoMC	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.52.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-267 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

3.52.5 Obtaining the Source Code

Obtain the source code.

```
cd /path/to/CosmoMC
git clone --recursive https://github.com/cmbant/CosmoMC.git
```

3.52.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the source code directory.

```
cd /path/to/CosmoMC/CosmoMC
```

Step 3 Perform the compilation and installation.

```
make
```

A CosmoMC executable file is generated in the current directory.

```
[root@localhost CosmoMC]# ls cosmomc
cosmomc
```

Step 4 Load the environment variable.

```
export PATH=/path/to/CosmoMC/CosmoMC:$PATH
```

----End

3.52.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether CosmoMC is running properly.

```
cd /path/to/CosmoMC/CosmoMC
mpirun --allow-run-as-root -np 32 ./cosmomc test.ini
```

NOTE

The number **32** indicates the running thread. Replace it with the actual value.

Information similar to the following is displayed:

```
     0.463      0.927   BAO: MGS
     2.783      5.567   BAO: DR12BAO
   286.154    572.308   WL: DES = DES_1YR_final
Test likelihoods done, total logLike, chi-eq =    1184.772   2369.543
Expected likelihoods,  total logLike, chi-eq =    1184.788   2369.576
 ...OK, delta =   -1.6285457220419630E-002
 Likelihood calculation time (seconds)= 8.6081
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
```

----End

3.53 ParaFEM 5.0.3 Porting Guide (Kylin V10)

3.53.1 Introduction

ParaFEM is an open source parallel finite element analysis library that is documented by the text book "Programming the Finite Element Method."

For more information, visit the **ParaFEM page at GitHub**.

Programming language: Fortran

Brief description: An open source parallel finite element analysis library.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: custom open source license

Recommended Software Version

ParaFEM 5.0.3

3.53.2 Environment Requirements

Hardware Requirements

Table 3-268 lists the hardware requirements.

Table 3-268 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 3-269 Software requirements

Item	Version	Download URL
ParaFEM	5.0.3	https://github.com/ParaFEM/ParaFEM/releases/download/5.0.3/parafem.5.0.3.tar.gz

OS Requirements

Table 3-270 lists the OS requirements.

Table 3-270 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.53.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ParaFEM software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-271 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/ ParaFEM	Installation path of ParaFEM	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.53.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-272 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.

3.53.5 Obtaining the Source Code

Procedure

Step 1 Download the ParaFEM installation package **parafem.5.0.3.tar.gz**.

 Copyright © Huawei Technologies Co., Ltd.

https://github.com/ParaFEM/ParaFEM/releases/download/5.0.3/parafem.5.0.3.tar.gz

Step 2 Use an SFTP tool to upload the ParaFEM installation package to the */path/to/ParaFEM* directory on the server.

----End

3.53.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/ParaFEM
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf parafem.5.0.3.tar.gz
cd parafem
```

Step 4 Perform the compilation and installation.

```
./make-parafem MACHINE=linuxdesktop
```

If the following information is displayed, the installation is successful:

```
********************************************************************************
Build ERRORS encountered:
  None
********************************************************************************

ParaFEM Build Complete
```

Step 5 Load the environment variable.

```
export PATH=/path/to/ParaFEM/parafem/bin:$PATH
```

----End

3.53.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify ParaFEM.

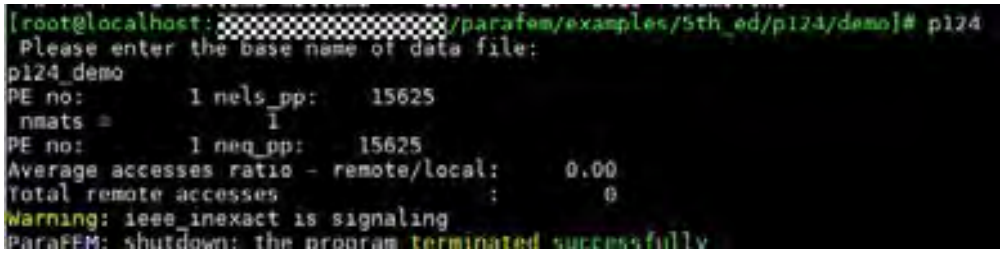
```
cd /path/to/ParaFEM/parafem/examples5/th_ed/p121/demo
p121
```

Type **p121_demo** and press **Enter**. The following figure shows the command output:

```
[root@localhost:/path/to/parafem/examples/5th_ed/p121/demo]# p121
 Please enter the base name of data file:
p121_demo
PE no:        1 nels_pp:     8000
PE no:        1 neq_pp:     98360
Average accesses ratio - remote/local:     0.00
Total remote accesses                :        0
Warning: ieee_inexact is signaling
ParaFEM: shutdown: the program terminated successfully
```

```
cd /path/to/ParaFEM/parafem/examples5/th_ed/p124/demo
p124
```

Type **p124_demo** and press **Enter**. The following figure shows the command output:



----End

3.54 VASPKIT 0.52 Porting Guide (Kylin V10)

3.54.1 Introduction

VASPKIT aims at providing a powerful and user-friendly interface to perform high throughput analysis of various material properties from the raw calculated data using the widely-used VASP code. The program can be conveniently run under either the interactive user interface or command line mode.

For more information, visit the **VASPKIT web page**.

Programming language: Fortran

Brief description: Performs high throughput analysis of various material properties from the raw calculated data using the widely-used VASP code.

Open source license: custom open source license

Recommended Software Version

VASPKIT 0.52

3.54.2 Environment Requirements

Hardware Requirements

Table 3-273 lists the hardware requirements.

Table 3-273 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 3-274 Software requirements

Item	Version	Download URL
vaspkit	0.52	https://sourceforge.net/projects/vaspkit/files/Src/vaspkit.0.52.tar.gz

OS Requirements

Table 3-275 lists the OS requirements.

Table 3-275 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.54.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VASPKIT software porting.

Table 3-276 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/vaspkit	Installation path of VASPKIT	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

3.54.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-277 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

3.54.5 Obtaining the Source Code

Procedure

Step 1 Download the VASPKIT installation package **vaspkit.0.52.tar.gz**.

https://sourceforge.net/projects/vaspkit/files/Src/vaspkit.0.52.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/vaspkit* directory on the server.

----End

3.54.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/vaspkit
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf vaspkit.0.52.tar.gz
cd vaspkit.0.52
```

Step 4 Modify the compilation file.

1. Open **make.inc**.
   ```
   vim make.inc
   ```
2. Press **i** to enter the insert mode. Change the content in line 6 to **FC=flang** and the content in line 12 to **CC=clang**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the compilation.

 Copyright © Huawei Technologies Co., Ltd.

```
cd src
make
```

----End

3.54.7 Running and Verification

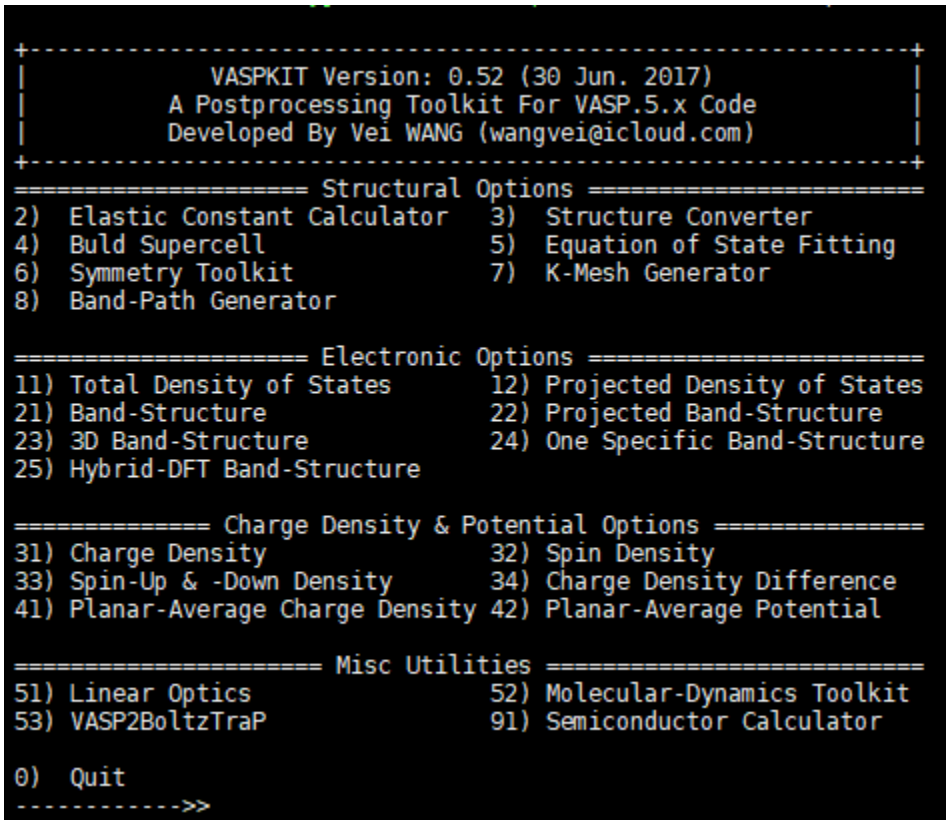
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify VASPKIT.

```
cd /path/to/vaspkit/vaspkit-0.52/src
./vaspkit
```

Information similar to the following is displayed:



----End

3.55 MNE-Python Porting Guide (openEuler 20.03)

3.55.1 Introduction

MNE-Python software is an open-source Python package for exploring, visualizing, and analyzing human neurophysiological data such as MEG, EEG, sEEG, ECoG, and more. It includes modules for data input/output, preprocessing, visualization, source estimation, time-frequency analysis, connectivity analysis, machine learning, and statistics.

For more information, visit the **MNE-Python page at GitHub**.

Programming language: Python

Brief description: A Python package for exploring, visualizing, and analyzing human neurophysiological data.

Open source license: BSD

Recommended Software Version

N/A

3.55.2 Environment Requirements

Hardware Requirements

Table 3-278 lists the hardware requirements.

Table 3-278 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 3-279 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz**
Python	3.7.5	**https://www.python.org/downloads/**

OS Requirements

Table 3-280 lists the OS requirements.

Table 3-280 OS requirements

Item	Version	Download URL
openEuler	20.03	**https://www.openeuler.org/en/download/**

 Copyright © Huawei Technologies Co., Ltd.

3.55.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MNE-Python software porting.

Table 3-281 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

3.55.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-282 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

3.55.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install MNE-Python.

```
pip install mne
```

Information similar to the following is displayed:

```
Installing collected packages: mne
Successfully installed mne-0.23.4
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.55.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the test case.

```
git clone https://github.com/mne-tools/mne-python.git
```

Step 3 Go to the test case directory.

```
cd mne-python/examples/datasets
```

Step 4 Execute the test case.

```
mkdir /root/mne_data
python hf_sef_data.py
```

If the error message "No module named'xxx'" is displayed, run the **pip install xxx** command to install the dependencies.

If the following information is displayed, the execution is successful:

----End

3.56 MiniDFT 1.1.1 Porting Guide (CentOS 7.6)

3.56.1 Introduction

MiniDFT is a plane-wave denstity functional theory (DFT) mini-app for modeling materials. It is extracted from Quantum Espresso (QE).

For more information, visit the **official MiniDFT website**.

Programming language: Fortran

Brief description: a mini-app for modeling materials.

Open source license: GPL

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

MiniDFT 1.1.1

3.56.2 Environment Requirements

Hardware Requirements

Table 3-283 lists the hardware requirements.

Table 3-283 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 3-284 lists the software requirements.

Table 3-284 Software Requirements

Item	Version	How to Obtain
MiniDFT	1.1.1	https://github.com/NERSC/MiniDFT/releases
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/tree/v0.3.6
FFTW	3.3.8	http://www.fftw.org/download.html
ScaLAPACK	2.1.0	http://www.netlib.org/scalapack/
Test case	small.in	Provided by the software.

OS Requirements

Table 3-285 lists the OS requirements.

Table 3-285 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

3.56.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MiniDFT software porting.

Table 3-286 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/FFTW	Installation path of FFTW	
4	/path/to/SCALAPACK	Installation path of ScaLAPACK	
5	/path/to/MINIDFT	Installation path of MiniDFT	

3.56.4 Configuring the Compilation Environment

3.56.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-287 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **3.56.4.2 Installing OpenBLAS**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing FFTW	See **3.56.4.3 Installing FFTW**.
4	Installing ScaLAPACK	See **3.56.4.4 Installing ScaLAPACK**.

3.56.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install BLAS and LAPACK:

```
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

NOTE

If you use a non-default compiler to install the OpenBLAS, download the OpenBLAS and use the specified compiler to install it. Specify the OpenBLAS path for compilation and installation in **Compiling and Installing**.

Step 3 Run the following command to decompress the OpneBLAS installation package:

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.6
```

Step 5 Run the following commands to perform the configuration:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 6 Run the following commands to perform compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS/install
```

----End

3.56.4.3 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd fftw-3.3.8
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW CC=gcc FC=gfortran F77=gfortran --enable-openmp --enable-threads --enable-threads --enable-shared=yes
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End

3.56.4.4 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar -xvf scalapack-2.1.0.tgz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd scalapack-2.1.0
```

Step 4 Run the following command to copy and paste the file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Modify the **SLmake.inc** file.

1. Open **SLmake.inc**.
   ```
   vi SLmake.inc
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   BLASLIB     = -L/path/to/OPENBLAS/lib -lopenblas
   LAPACKLIB  = -L/path/to/OPENBLAS/lib -lopenblas
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform compilation:

```
make
```

Step 7 Run the following command to copy the generated link library to the installation directory:

```
cp libscalapack.a /path/to/SCALAPACK
```

Step 8 Run the following commands to add the environment variables:

```
export LD_LIBRARY_PATH=/path/to/SCALAPACK:$LD_LIBRARY_PATH
```

----End

3.56.5 Obtaining the Source Code

Procedure

Step 1 Download the MiniDFT installation package **MiniDFT-MiniDFT-1.1.1.tar.gz**.

https://github.com/NERSC/MiniDFT/releases

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the MiniDFT installation package to the */path/to/MINIDFT* directory on the server.

----End

3.56.6 Compiling and Installing MiniDFT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the MiniDFT installation package:

```
tar -xzvf MiniDFT-MiniDFT-1.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd MiniDFT-MiniDFT-1.1.1/src
```

Step 4 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   FFTW_INCL = -I/path/to/FFTW/include
   FFTW_LIBS   = /path/to/SCALAPACK/libscalapack.a -Wl,--start-group /path/to/FFTW /lib/libfftw3.a /path/to/FFTW/lib/libfftw3_threads.a /path/to/OPENBLAS/lib/libopenblas.a -Wl,--end-group

   OPENBLAS_INC = -I/path/to/OPENBLAS/include
   OPENBLAS_LIB = /path/to/OPENBLAS/lib

   SCALAPACK_LIBS = /path/to/SCALAPACK/libscalapack.a

   CC = mpicc

   CFLAGS = -D__FFTW3 -D_para11e1 -D_SCALAPACK

   FC = mpifort
   FFLAGS = -cpp -no-pie

   LD = mpifort

   DFLAGS    += -D__OPENMP

   FFLAGS    += -fopenmp
   LDFLAGS   += -fopenmp

   include Makefile.base
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following commands to perform compilation and installation:

```
make
```

----End

3.56.7 Running and Verifying MiniDFT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the test case directory:

```
cd /path/to/MINIDFT/MiniDFT-MiniDFT-1.1.1/benchmark
```

Step 3 Run the following command to configure the threads:

```
export OMP_NUM_THREADS=1
```

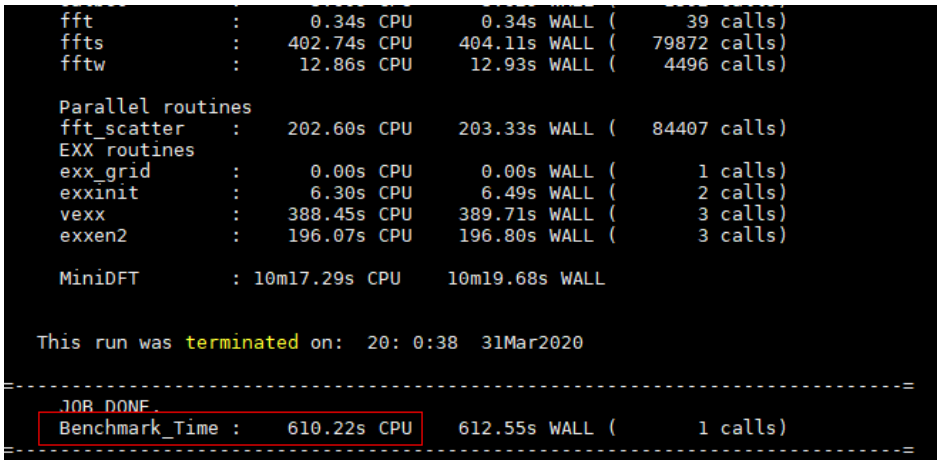
Step 4 Run the following command to perform the test:

```
mpirun -np 96 --allow-run-as-root -mca btl ^openib ../src/mini_dft -in small.in -nbgrp 4 > small.log
```

In the **small.log** file, check the value of **Benchmark_Time** (in seconds). A smaller value indicates higher performance.

Figure 3-20 shows the command output.

Figure 3-20 Result example



----End

3.56.8 Troubleshooting

An Error Is Reported During MiniDFT Compilation

Symptom

An error message "pwscf.f90:(.text+0x20): undefined reference to `dfftw_init_threads" is displayed during MiniDFT compilation.

Possible causes

FFTW does not have the threads module.

Procedure

Recompile FFTW and add the **--enable-threads** parameter during compilation. For details, see **Step 4**.

Problem 2: An Error is Reported When the MiniDFT Is Run

Symptom

When **large.in** is run, "mpirun noticed that process rank 84 with PID 0 on node XA320V2-48 exited on signal 11 (Segmentation fault)" is reported.

Possible causes

The software is memory consuming, causing shortage of memory resources.

Procedure

Increase the memory capacity or use a small **small.in** computing instance.

3.56.9 More Information

MiniDFT official installation guide:

https://github.com/NERSC/MiniDFT

3.57 dugksFoam 1.0 Porting Guide (Kylin V10)

3.57.1 Introduction

dugksFoam is an OpenFOAM solver based on the Boltzmann model equation of the discrete unified gas dynamics scheme. dugksFoam discretizes the governing equation in both physical space and velocity space. It solves the partial differential equations of the discrete velocity distribution functions in a finite volume framework. In dugksFoam, the fluxes of distribution functions are constructed from the local characteristic solution of the governing equation itself.

For more information, visit the **dugksFoam page at GitHub**.

Programming language: C/C++

Brief description: An OpenFOAM solver for Boltzmann model equation using discrete unified gas kinetic scheme.

Open source license: MIT License

Recommended Software Version

1.0

3.57.2 Environment Requirements

Hardware Requirements

Table 3-288 lists the hardware requirements.

Table 3-288 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-289 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-289 Software requirements

Item	Version	Download URL
dugksFoam	1.0	https://github.com/zhulianhua/dugksFoam/archive/refs/heads/master.zip

OS Requirements

Table 3-290 lists the OS requirements.

Table 3-290 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.57.3 Paths for Software Porting

This chapter lists the software installation paths involved in the dugksFoam software porting.

Table 3-291 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	$FOAM_RUN/../applications/dugksFoam	Installation path of dugksFoam.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/OpenFOAM	Installation path of OpenFOAM.	

3.57.4 Configuring the Compilation Environment

3.57.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-292 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing HMPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
3	Installing OpenFOAM	dugksFoam depends on OpenFOAM. Install OpenFOAM 2.4.0 or later. See **3.57.4.2 Installing OpenFOAM v2206**.

3.57.4.2 Installing OpenFOAM v2206

Procedure

Step 1 Download the source package and decompress it.

```
cd /path/to/OpenFOAM
wget https://dl.openfoam.com/source/v2206/OpenFOAM-v2206.tgz
wget https://dl.openfoam.com/source/v2206/ThirdParty-v2206.tgz
tar -xvf OpenFOAM-v2206.tgz
tar -xvf ThirdParty.v2206.tgz
```

Step 2 Set the environment variables.

```
source OpenFOAM-v2206/etc/bashrc
```

NOTE

The following warning is displayed. Ignore it.

No comletions for /path/to/OpenFOAM/OpenFOAM-v2206/platorms/linuxARM64gGccDPInt320pt/bin/

[ignore if OpenFOAM is not yet compiled]

Step 3 Check the system environment.

```
foamSystemCheck
```

NOTE

If "System check: PASS, Can continue to OpenFOAM installation" is displayed, you can continue to install OpenFOAM.

Step 4 Perform the compilation and installation.

```
./OpenFOAM-v2206/Allwmake -j -s -q -l
```

NOTE

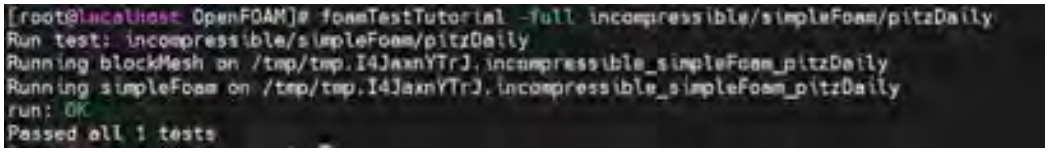
This will use all kernels (-j), reduced outputs (-s, -silent), queues (-q, -queue), and logs (-l, -log) to record the final output into a file (for example, **log.linux64GccDPInt32Opt**) for later checks.

Step 5 Perform the verification.

```
foamTestTutorial -full incompressible/simpleFoam/pitzDaily
```

 Copyright © Huawei Technologies Co., Ltd.

If the following information is displayed, the verification is successful.



----End

3.57.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create an installation directory.

```
mkdir -p $FOAM_RUN/../applications
cd $FOAM_RUN/../applications
```

Step 3 Download the source code.

```
git clone https://github.com/zhulianhua/dugksFoam.git
```

----End

3.57.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source code directory.

```
cd $FOAM_RUN/../applications/dugksFoam
```

Step 3 Modify the configuration file.

```
vi inc/FoamDefs/v2206/foam_defs.h
```

For example,

Before modification:

```
#define FOMA_MAJOR v2206
```

After modification:

```
#define FOMA_MAJOR 10
```

Before modification:

```
#define FOMA_MINOR
```

After modification:

```
#define FOMA_MINOR 0
```

NOTE

In this example, **10** indicates the version number of the OpenFOAM community distribution released with OpenFOAM v2206. dugksFoam uses this version number as the benchmark.

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Perform the compilation and installation.

```
./Allwmake -j 64
```

----End

3.57.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
which dugksFoam
```

If the following information is displayed, the verification is successful.

```
[root@localhost src]# which dugksFoam
/root/OpenFOAM/root-v2206/platforms/linuxARM64GccDPInt32Opt/bin/dugksFoam
```

----End

3.57.8 Change History

Table 3-293

Date	Description
2023-04-28	This issue is the first official release.

3.58 DUNS 4.7.1 Porting Guide (Kylin V10)

3.58.1 Introduction

The Diagonalized Upwind Navier-Stokes (DUNS) code is a 2D/3D, structured, multi-block, multi-species, reacting, steady/unsteady, Navier Stokes fluid dynamics code with q-omega turbulence model. It currently uses a diagonalized ADI procedure with upwind differences.

For more information, visit the **DUNS website**.

Programming language: Fortran

One-sentence description: Fluid dynamics code.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

4.7.1

 Copyright © Huawei Technologies Co., Ltd.

3.58.2 Environment Requirements

Hardware Requirements

Table 3-294 lists the hardware requirements.

Table 3-294 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 3-295 lists the OS requirements.

Table 3-295 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 3-296 lists the software requirements.

Table 3-296 Software requirements

Item	Version	Download URL
DUNS	4.7.1	https://sourceforge.net/projects/duns/files/duns/duns-2.7.1/duns-2.7.1-src.tar.bz2
GCC for openEuler	9.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz

3.58.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DUNS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-297 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/duns	Installation path of DUNS.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.58.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server is connected to the Internet.

Configuration Process

Table 3-298 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.

3.58.5 Obtaining the Source Code

Procedure

Step 1 Download the DUNS installation package.

https://sourceforge.net/projects/duns/files/duns/duns-2.7.1/duns-2.7.1-src.tar.bz2

Step 2 Use an SFTP tool to upload the ncview installation package to the */path/to/duns* directory on the server.

----End

3.58.6 Compilation and Installation

Procedure

Step 1 Go to the installation package directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/duns
```

Step 2 Decompress the installation package.

```
tar xvf duns-2.7.1-src.tar.bz2
cd duns-2.7.1
```

Step 3 Create an installation directory for future use.

```
export DUNSARCH=linux
export DUNSPATH27=`pwd`
export PATH=$DUNSPATH27/util:$PATH
```

Step 4 Modify the related file.

```
sed -i '10s/g77/gfortran/' makesystem/Makesystem_linux
sed -i "28s#.*#SYSLIBS   = -L/path/to/gcc/9.3.1/lib64 -Wl,-Bdynamic -lgcc_s -lm -lgfortran#"
makesystem/Makesystem_linux
```

NOTE

Replace **/path/to/gcc/9.3.1/lib64** in the preceding example command with the actual lib64 directory where GCC is installed.

```
sed -i '29s#.*#GRAFLIBS  = -lmbgrafic -lgrafic -L/usr/lib -Wl,-Bdynamic -lX11#' makesystem/
Makesystem_linux
sed -i 's/getline/getline1/g' `grep getline -rl --include="*" src/*`
```

Step 5 Perform the compilation and installation.

```
cd lib
make
make install
```

Step 6 Perform a 3D program.

```
cd ..
dunssource 3d all
cd duns-3d-all
line=`expr $(grep -n -m 1 "^c .*" assemble.f |awk -F: '{print $1}') - 1`
sed -i "1,${line}d" *.f *.c include/*
make -j
```

Step 7 Check whether **duns.e** and **dunsplot.e** are generated.

```
ls
```

If the following information is displayed, the generation is successful.

----End

3.58.7 Running and Verification

Procedure

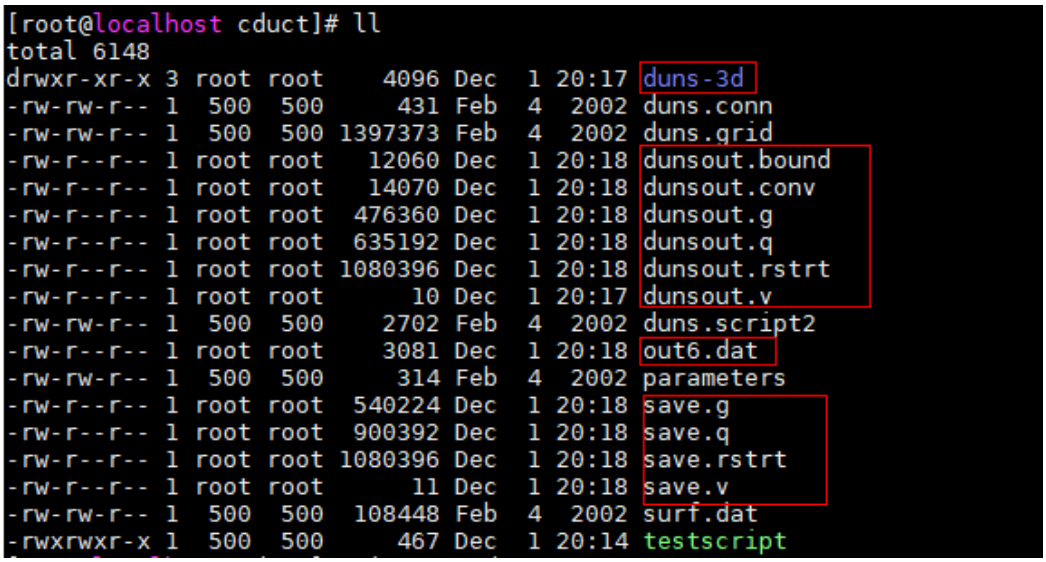
Step 1 Download the test cases.

```
cd /path/to/duns
wget https://sourceforge.net/projects/duns/files/duns/duns-2.7.1/duns-2.7.1-test.tar.bz2
tar xf duns-2.7.1-test.tar.bz2
cd duns-2.7.1/test/serial/cduct
```

Step 2 Run the test.

```
sed -i "/cd duns.*/a\       sed -i '1,42d' *.f include/*" testscript
./testscript run
```

The following information is displayed. After the execution is complete, related files such as **out6.dat** are generated in the current directory.



----End

3.58.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

3.59 CalculiX 2.19.0 Porting Guide (Kylin V10)

3.59.1 Introduction

CalculiX is a package designed to solve field problems. The method used is the finite element method. With CalculiX Finite Element Models can be built, calculated, and post-processed. The pre- and post-processor is an interactive 3D-tool using the openGL API. The solver is able to do linear and non-linear calculations. Static, dynamic, and thermal solutions are available. Both programs can be used independently. Because the solver makes use of the abaqus input format it is possible to use commercial pre-processors as well. In turn the pre-processor is able to write mesh related data for nastran, abaqus, ansys, code-aster and for the free-cfd codes dolfyn, duns, ISAAC, and OpenFOAM. A simple step reader is included. In addition external CAD interfaces are available. The program is designed to run on Unix platforms like Linux and Irix computers but also on MS-Windows.

For more information, visit the **CalculiX page at GitHub**.

Programming language: C

Brief description: With CalculiX Finite Element Models can be built, calculated, and post-processed.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Version

2.19.0

3.59.2 Environment Requirements

Hardware Requirements

Table 3-299 lists the hardware requirements.

Table 3-299 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 3-300 lists the OS requirements.

Table 3-300 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 3-301 lists the software requirements.

Table 3-301 Software requirements

Item	Version	Download URL
CalculiX	2.19.0	https://github.com/precice/calculix-adapter/archive/refs/heads/master.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
HMPI	1.2.1	https://gitee.com/kunpengcompute/hmpi/archive/refs/tags/v1.2.1-huawei.tar.gz

3.59.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CalculiX software porting.

Table 3-302 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/calculix	Installation path of CalculiX.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.59.4 Configuring the Compilation Environment

3.59.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 3-303 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing HMPI	See **Installing HMPI**.
3	Installing SPOOLES	See **3.59.4.2 Installing SPOOLES**.
4	Installing ARPACK	See **3.59.4.3 Installing ARPACK**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
5	Installing yaml-cpp	See **3.59.4.4 Installing yaml-cpp**.
6	Installing Eigen3	See **3.59.4.5 Installing Eigen3**.
7	Installing preCICE	See **3.59.4.6 Installing preCICE**.

3.59.4.2 Installing SPOOLES

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to
mkdir SPOOLES.2.2
wget http://www.netlib.org/linalg/spooles/spooles.2.2.tgz
tar xvf spooles.2.2.tgz -C SPOOLES.2.2
cd SPOOLES.2.2
```

Step 3 Modify the configuration file.

```
sed -i '14c\ CC = clang' Make.inc
sed -i '15c\#CC = /usr/lang-4.0/bin/cc' Make.inc
sed -i '9c\      draw.c \' Tree/src/makeGlobalLib
```

Step 4 Perform the installation.

```
make lib
```

----End

3.59.4.3 Installing ARPACK

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://www.caam.rice.edu/software/ARPACK/SRC/arpack96.tar.gz
wget https://www.caam.rice.edu/software/ARPACK/SRC/patch.tar.gz
tar xvf arpack96.tar.gz
tar xvf patch.tar.gz
cd ARPACK
```

Step 2 Modify the configuration file.

```
sed -i "28c\home =/path/to/ARPACK" ARmake.inc
sed -i '35c\PLAT = INTEL' ARmake.inc
sed -i '104c\FC = flang' ARmake.inc
sed -i '105c\#FFLAGS = -0 -cg89' ARmake.inc
sed -i '115c\MAKE = /usr/bin/make' ARmake.inc
sed -i '120c\SHELL = /usr/bin/sh' ARmake.inc
sed -i '24c\*      EXTERNAL    ETIME' UTIL/second.f
```

Step 3 Perform the installation.

```
make lib
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.59.4.4 Installing yaml-cpp

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://github.com/jbeder/yaml-cpp/archive/yaml-cpp-0.6.2.zip
unzip yaml-cpp-0.6.2.zip
cd yaml-cpp-yaml-cpp-0.6.2
```

Step 2 Create a build directory.

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/yaml-cpp-install -DBUILD_SHARED_LIBS=ON
```

Step 3 Perform the installation.

```
make
make install
```

Step 4 Set the environment variable.

```
export LD_LIBRARY_PATH=/path/to/yaml-cpp-install/lib:$LD_LIBRARY_PATH
```

----End

3.59.4.5 Installing Eigen3

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget http://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz
tar xvf eigen-3.4.0.tar.gz
cd eigen-3.4.0
```

Step 2 Create a build directory.

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/eigen3-install
```

Step 3 Perform the installation.

```
make
make install
```

----End

3.59.4.6 Installing preCICE

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://github.com/precice/precice/archive/v2.3.0.tar.gz -O precice-2.3.0.tar.gz
tar xvf precice-2.3.0.tar.gz
cd precice-2.3.0
```

Step 2 Create a build directory.

```
mkdir build
cd build
```

 Copyright © Huawei Technologies Co., Ltd.

```
export Eigen3_ROOT=/path/to/eigen3-install
cmake .. -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/path/to/precice-install -
DPRECICE_PETScMapping=OFF -DPRECICE_PythonActions=OFF
```

CAUTION

If **Could NOT find Boost** is displayed when you run the **cmake** command, run the **yum install boost boost-devel** command and run the preceding command again.

Step 3 Perform the installation.

```
make
make install
```

Step 4 Set the environment variable.

```
export LD_LIBRARY_PATH=/path/to/precice-install/lib64:$LD_LIBRARY_PATH
```

----End

3.59.5 Obtaining the Source Code

Procedure

Step 1 Download the CalculiX and CCX installation packages.

https://github.com/precice/calculix-adapter/archive/refs/heads/master.tar.gz

http://www.dhondt.de/ccx_2.19.src.tar.bz2

Step 2 Use an SFTP tool to upload the CalculiX and CCX installation packages to the */path/to/calculix* directory on the server.

----End

3.59.6 Compilation and Installation

Procedure

Step 1 Go to the installation directory of the main program.

```
cd /path/to/calculix
```

Step 2 Decompress the installation packages.

```
tar xvf master.tar.gz
tar xvf ccx_2.19.src.tar.bz2
```

Step 3 Load the environment variables.

```
export PKG_CONFIG_PATH=/path/to/precice-install/lib64/pkgconfig:$ PKG_CONFIG_PATH
export CPATH=/path/to/precice-install/include:$CPATH
```

Step 4 Modify the configuration file.

```
cd calculix-adapter-master
sed -i "6c\CCX = /path/to/calculix/CalculiX/ccx_2.19/src" Makefile
sed -i "10c\SPOOLES_INCLUDE = -I/path/to/SPOOLES.2.2" Makefile
sed -i "12c\SPOOLES_LIBS = /path/to/SPOOLES.2.2/spooles.a" Makefile
sed -i "15c\ARPACK_INCLUDE = -I/path/to/ARPACK" Makefile
sed -i "17c\ARPACK_LIBS = /path/to/ARPACK/libarpack_INTEL.a" Makefile
sed -i "20c\YAML_INCLUDE = -I/path/to/yaml-cpp-yaml-cpp-0.6.2/include" Makefile
sed -i "22c\YAML_LIBS = -L/path/to/yaml-cpp-yaml-cpp-0.6.2/build -lyaml-cpp" Makefile
```

```
sed -i '55c\CFLAGS = -Wall -O3 -fopenmp $(INCLUDES) -DARCH="Linux" -DSPOOLES -DARPACK -
DMATRIXSTORAGE' Makefile
sed -i '60c\CC = clang' Makefile
sed -i '62c\CC = clang' Makefile
sed -i '67c\FC = flang' Makefile
```

Step 5 Perform the compilation and installation.

```
make
```

----End

3.59.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/calculix/calculix-adapter-master/bin
./ccx_preCICE --version
```

Command output:

```
************************************************************
CalculiX Version 2.19, Copyright(C) 1998-20621 Guido Dhondt
CalculiX comes with ABSOLUTELY NO WARRANTY. This is free
software, and you are welcome to redistribute it under
certain conditions, see gpl.html
************************************************************
```

3.59.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

3.60 BerkeleyGW 3.0.1 Porting Guide (Kylin V10)

3.60.1 Introduction

BerkeleyGW is a many-body perturbation theory code for excited states, using the GW method and the GW plus Bethe-Salpeter equation (GW-BSE) method to solve respectively for quasiparticle excitations and optical properties of materials. BerkeleyGW is suitable for 3D, 2D, 1D, and molecular systems; applicable to insulating, metallic, and semi-metallic systems; massively parallelized with MPI, OpenMP, SIMD; and recently ported to GPUs, reaching 86x speedup compared to the CPU implementation.

For more information, visit the **BerkeleyGW official website**.

Programming language: C/Fortran

Brief description: A many-body perturbation theory code for excited states.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

3.0.1

3.60.2 Environment Requirements

Hardware Requirements

Table 3-304 lists the hardware requirements.

Table 3-304 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 3-305 lists the OS requirements.

Table 3-305 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 3-306 lists the software requirements.

Table 3-306 Software requirements

Item	Version	Download URL
BerkeleyGW	3.0.1	https://berkeleygw.org/download
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz
HMPI	1.2.1	https://gitee.com/kunpengcompute/hmpi/archive/refs/tags/v1.2.1-huawei.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

3.60.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BerkeleyGW software porting.

Table 3-307 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/BerkeleyGW	Installation path of BerkeleyGW.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.60.4 Configuring the Compilation Environment

3.60.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 3-308 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing HMPI	See **Installing HMPI**.
3	Installing LAPACK	See **3.60.4.2 Installing LAPACK**.
3	Installing FFTW	See **3.60.4.3 Installing FFTW**.
4	Installing ScaLAPACK	See **3.60.4.4 Installing ScaLAPACK**.

 Copyright © Huawei Technologies Co., Ltd.

3.60.4.2 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install lapack lapack-devel blas blas-devel
```

----End

3.60.4.3 Installing FFTW

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget http://www.fftw.org/fftw-3.3.8.tar.gz
tar xf fftw-3.3.8.tar.gz
```

Step 2 Go to the source package directory and perform the installation.

```
cd fftw-3.3.8
CC=mpicc CXX=mpicxx FC=mpif90 ./configure --prefix=/path/to/fftw
make
make install
```

----End

3.60.4.4 Installing ScaLAPACK

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget http://www.netlib.org/scalapack/scalapack-2.1.0.tgz
tar xf scalapack-2.1.0.tgz
```

Step 2 Go to the source package directory and perform the installation.

```
cd scalapack-2.1.0
mkdir build
cd build
CC=mpicc CXX=mpicxx FC=mpif90 cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/scalapack
make
make install
```

----End

3.60.5 Obtaining the Source Code

Procedure

Step 1 Download the BerkeleyGW installation package.

https://berkeleygw.org/download

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the BerkeleyGW installation package to the */path/to/BerkeleyGW* directory on the server.

----End

3.60.6 Compilation and Installation

Procedure

Step 1 Decompress the source package.

```
cd /path/to/BerkeleyGW
tar xf BerkeleyGW-3.0.1.tar.gz
```

Step 2 Go to the source code directory and create a configuration file.

```
cd BerkeleyGW-3.0.1
vim arch.mk
```

Add the following content, save the file, and exit.

```
COMPFLAG  = -DGNU
PARAFLAG  = -DMPI
MATHFLAG  = -DUSESCALAPACK -DUSEFFTW3

FCPP    = cpp -C -nostdinc
F90free = /path/to/hmpi/bin/mpif90 -ffree-form -ffree-line-length-none -Wall -pedantic-errors  -fopenmp

LINK    = /path/to/hmpi/bin/mpif90 -fuse-ld=lld
FOPTS   = -O3

FNOOPTS = $(FOPTS)
MOD_OPT = -J
INCFLAG = -I

C_PARAFLAG  = -DPARA
CC_COMP  = /path/to/hmpi/bin/mpiCC -Wall -pedantic-errors
C_COMP  = /path/to/hmpi/bin/mpicc -Wall -pedantic-errors
C_LINK  = /path/to/hmpi/bin/mpiCC -no-flang-libs -fuse-ld=lld
C_OPTS  = -O3
C_DEBUGFLAG =

REMOVE  = /bin/rm -f

FFTWLIB      = /path/to/fftw/lib/libfftw3.a
FFTWINCLUDE  = /path/to/fftw/include

LAPACKLIB    = /usr/lib64/liblapack.so.3 /usr/lib64/libblas.so.3
SCALAPACKLIB = /path/to/scalapack/lib/libscalapack.a $(LAPACKLIB)

TESTSCRIPT = make check-parallel
```

Step 3 Install BerkeleyGW.

```
make all-flavors
make install INSTDIR=/path/to/BerkeleyGW
```

Step 4 Set the environment variable.

```
export PATH=/path/to/BerkeleyGW/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

3.60.7 Running and Verification

Procedure

Step 1 Go to the test case directory.

```
cd /path/to/BerkeleyGWBerkeleyGW-3.0.1/testsuite/CUBE
```

Step 2 Decompress **homo.cube.tgz**.

```
tar xf homo.cube.tgz
```

Step 3 Perform a test.

```
time mpirun --allow-run-as-root -np 64 surface.x surface.inp
```

Check the **real** value (unit: s) in the command output. A smaller value indicates higher performance.

```
 the minimal bounding box for the isosurface

 xmin = -4.495638479 xmax = 4.495638479 dx = 8.991276957 bohr
 ymin = -3.233418444 ymax = 3.233418444 dy = 6.466836888 bohr
 zmin = -3.206370622 zmax = 3.206370622 dz = 6.412741244 bohr


 the minimal bounding box for the isosurface

 xmin = -4.495638479 xmax = 4.495638479 dx = 8.991276957 bohr
 ymin = -3.233418444 ymax = 3.233418444 dy = 6.466836888 bohr
 zmin = -3.206370622 zmax = 3.206370622 dz = 6.412741244 bohr


real    0m0.514s
user    0m10.472s
sys     0m1.532s
```

----End

3.60.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

3.61 Amber 20 Porting Guide (CentOS 8.2)

3.61.1 Introduction

Molecular dynamics (MD) simulation is the most widely used method for calculating a large complex system. Due to rapid development of molecular simulation since 1970, many force fields applicable to biochemical molecular systems, polymers, metals, and non-metallic materials have been systematically established. This greatly improves the capability and accuracy of computing complex system structures and some thermodynamic and spectral properties. MD simulation is a computational method developed by applying these force fields and according to Newton's mechanics of motion.

 Copyright © Huawei Technologies Co., Ltd.

Amber is one of the widely used molecular dynamics simulation software. After more than ten years of development, the software is mature and supports GPU acceleration, enhanced sampling, free energy prediction, and QM/MM. It is widely used in the simulation of proteins, polypeptides, polysaccharides, nucleic acids, phospholipids, and other biological molecules.

For more information, visit the **Amber official website**.

Programming language: C

Brief description: One of the widely used molecular dynamics simulation software.

Open source license: custom open source license

Recommended Software Version

Amber 20

3.61.2 Environment Requirements

Hardware Requirements

Table 3-309 lists the hardware requirements.

Table 3-309 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 3-310 lists the software requirements.

Table 3-310 Software requirements

Item	Version	Download URL
Amber	20	**http://Ambermd.org/index.php**
AmberTools	21	**http://Ambermd.org/index.php**
BiSheng Compiler	1.3.3	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz**
HMPI	1.2.0	**https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html**
CMake	3.8.1	**https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Anaconda3	2021.05	https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
Test case	Amber20_Benchmark_Suite	http://Ambermd.org/Amber20_Benchmark_Suite.tar.gz

OS Requirements

Table 3-311 lists the OS requirements.

Table 3-311 OS requirements

Item	Version	Download URL
CentOS	8.2	https://www.centos.org/download/
Kernel	4.18.0-193.el8.aarch64	Included in the OS image

3.61.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Amber software porting.

Table 3-312 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bisheng-1.3.3	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/HMPI	Installation path of HMPI.	
4	/path/to/NVIDIA-CUDA	Installation path of NVIDIA CUDA.	
5	/path/to/Anaconda3	Installation path of Anaconda3.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
7	/path/to/CMAKE	Installation path of CMake.	
8	/path/to/AMBER	Installation path of Amber.	
9	/path/to/CASE	Path for storing Amber test cases.	

3.61.4 Configuring the Compilation Environment

3.61.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-313 Configuration process

No.	Configuration Item	Remarks
1	Disabling the Nouveau driver	See **3.61.4.2 Disabling the Nouveau Driver**.
2	Installing NVIDIA CUDA	See **3.61.4.3 Installing NVIDIA CUDA**.
3	Installing the BiSheng Compiler	See **3.61.4.4 Installing the BiSheng Compiler**.
4	Installing HMPI	See **3.61.4.5 Installing HMPI**.
5	Installing Anaconda3	See **3.61.4.6 Installing Anaconda3**.
6	Installing CMake	See **3.61.4.7 Installing CMake**.

3.61.4.2 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Disable the Nouveau driver.

```
vim /etc/modprobe.d/disable-nouveau.conf
```

 Copyright © Huawei Technologies Co., Ltd.

Add the following two lines at the beginning of the file:

```
blacklist nouveau
options nouveau modeset=0
```

Step 3 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 4 Restart the server.

```
reboot
```

----End

3.61.4.3 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled (no command output is displayed).

```
lsmod |grep nouveau
```

Step 3 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo sh cuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.2/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.2/lib64:$LD_LIBRARY_PATH
```

Step 5 View the version information.

```
nvcc -v
```

----End

3.61.4.4 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-
compiler-1.3.3-aarch64-linux.tar.gz
```

Step 3 Create an installation directory.

```
mkdir -p /opt/compiler
```

Step 4 Copy the downloaded installation package to the installation directory.

```
cp -r bisheng-compiler-1.3.3-aarch64-linux.tar.gz /opt/compiler
```

Step 5 Go to the installation directory and decompress the installation package.

```
cd /opt/compiler
tar -zxvf bisheng-compiler-1.3.3-aarch64-linux.tar.gz
```

Step 6 Set the environment variables of the BiSheng Compiler.

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 7 View the BiSheng Compiler version.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

3.61.4.5 Installing HMPI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Download HUCX and XUCG.

```
wget -c https://github.com/kunpengcompute/hucx/archive/refs/heads/huawei.zip -O hucx-huawei.zip
wget -c https://github.com/kunpengcompute/xucg/archive/refs/heads/huawei.zip -O xucg-huawei.zip
```

Step 2 Decompress the installation packages.

```
unzip hucx-huawei.zip
unzip xucg-huawei.zip
```

Step 3 Copy the content in the XUCG source package to the **src/ucg** directory of the HUCX source package.

```
cp -r xucg-huawei/*  hucx-huawei/src/ucg/
cd hucx-huawei
```

Step 4 Perform the compilation and installation.

```
export SEC_CFLAGS="-Wno-error=deprecated-declarations -DHAVE___CLEAR_CACHE=1 -Wno-error=asm-operand-widths -Wno-error=implicit-int-float-conversion -Wno-error=format"
mkdir -p /path/to/ucx/
./autogen.sh
./contrib/configure-opt --prefix=/path/to/ucx CFLAGS="$SEC_CFLAGS" CXXFLAGS="$SEC_CFLAGS"
make -j32
make install
```

Step 5 Download HMPI.

```
wget -c https://github.com/kunpengcompute/hmpi/archive/refs/heads/huawei.zip -O hmpi-huawei.zip
```

Step 6 Decompress the installation package.

```
unzip hmpi-huawei.zip
cd hmpi-huawei/
```

Step 7 Perform the compilation and installation.

```
./autogen.pl
./configure --prefix=/path/to/install/ompi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=/path/to/install/ucx
make -j16
make install
```

Step 8 Set the environment variables.

```
export PATH=/path/to/install/ompi/bin:/path/to/install/ucx/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/ompi/lib:/path/to/install/ucx/lib:$LD_LIBRARY_PATH
export OPAL_PREFIX=/path/to/install/ompi/
export INCLUDE=/path/to/install/ompi:$INCLUDE
```

----End

3.61.4.6 Installing Anaconda3

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Anaconda3.

```
bash Anaconda3-2021.05-Linux-aarch64.sh
```

Step 3 Perform the installation as instructed.

Step 4 Set the Anaconda3 environment variable.

```
source ~/.bashrc
```

----End

3.61.4.7 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the CMake installation package.

```
wget https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz
```

Step 3 Decompress the installation package.

```
tar -zxvf cmake-3.8.1.tar.gz
cd cmake-3.8.1
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/CMAKE
make -j64
make install
```

Step 5 Set the environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

3.61.5 Obtaining the Source Code

Procedure

Step 1 Download the Amber installation packages **Amber20.tar.bz2** and **AmberTools21.tar.bz2**.

http://ambermd.org/index.php

Step 2 Use an SFTP tool to upload the AmberTools installation packages to the */path/to/AMBER* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

3.61.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/AMBER
```

Step 3 Decompress the installation packages.

```
tar -jxvf Amber20.tar.bz2
tar -jxvf AmberTools21.tar.bz2
```

Step 4 Go to the directory generated after the decompression.

```
cd amber20_src/build
```

Step 5 Modify the configuration file.

```
vim run_cmake
```

Modify lines 42 and 43, specifically, set **-DMPI=TRUE -DCUDA=TRUE -DDOWNLOAD_MINICONDA=FALSE**.

```
37 #  Assume this is Linux:
38
39   cmake $AMBER_PREFIX/amber20_src \
40     -DCMAKE_INSTALL_PREFIX=/opt/tools/amber20 \
41     -DCOMPILER=GNU   \
42     -DMPI=TRUE -DCUDA=TRUE -DINSTALL_TESTS=TRUE \
43     -DDOWNLOAD_MINICONDA=FALSE -DMINICONDA_USE_PY3=TRUE \
44     2>&1 | tee  cmake.log
```

Step 6 Perform the compilation and installation.

```
./run_cmake
make -j40 install
```

Step 7 Check whether an executable file is generated.

```
ll /path/to/install/AMBER/bin/pmemd
```

Step 8 Set the environment variable.

```
source /path/to/install/AMBER/amber.sh
```

----End

3.61.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Switch to the test directory.

```
cd /path/to/CASE
```

Step 3 Decompress the test case file.

```
tar -zxvf  Amber20_Benchmark_Suite.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd PME/Cellulose_production_NPT_4fs/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the Amber tests.

```
cp ../Coordinates/Cellulose.inpcrd inpcrd
cp ../Topologies/Cellulose.prmtop prmtop
mpirun -np 96 $AMBERHOME/exe/pmemd -O -i mdin.CPU -o mdout.CPU -p prmtop -c inpcrd
$AMBERHOME/exe/pmemd.cuda -O -i mdin.GPU -o mdout.GPU -p prmtop -c inpcrd
```

View the value of **ns/day** under **Performance** in the **mdout.CPU** file. A higher value indicates higher performance.

The following is an example of the test result:

```
Final Performance Info:
|     -----------------------------------------------------
|     Average timings for last      10 steps:
|     Elapsed(s) =      40.80 Per Step(ms) =    4079.94
|         ns/day =       0.08   seconds/ns = 1019985.72
|
|     Average timings for all steps:
|     Elapsed(s) =     783.10 Per Step(ms) =    3915.48
|         ns/day =       0.09   seconds/ns =  978871.00
```

----End

3.61.8 Change History

Date	Description
2021-08-21	This issue is the first official release.

3.62 Galaxy 22.01 Porting Guide (Kylin V10)

3.62.1 Introduction

Galaxy is a web-based data analysis platform. It is easy to use and completely free. It offers over 8,000 analysis tools. It is widely applied in bioinformatics, chemistry, ecology, climate science, and more.

For more information, visit the **Galaxy page at GitHub**.

Programming language: Python

Brief description: A web-based data analysis platform.

Open source license: Academic Free License (AFL) v.3.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

22.01

 Copyright © Huawei Technologies Co., Ltd.

3.62.2 Environment Requirements

Hardware Requirements

Table 3-314 lists the hardware requirements.

Table 3-314 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-315 lists the software requirements.

Table 3-315 Software requirements

Item	Version	Download URL
Galaxy	22.01	https://github.com/galaxyproject/galaxy/archive/refs/heads/master.zip

OS Requirements

Table 3-316 lists the OS requirements.

Table 3-316 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.62.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Galaxy software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-317 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/GALAXY	Installation path of Galaxy.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.62.4 Configuring the Compilation Environment

3.62.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 3-318 Configuration process

No.	Configuration Item	Remarks
1	Installing Python 3.7.6	See **3.62.4.2 Installing Python 3.7.6**.
2	Installing virtualenv	See **3.62.4.3 Installing virtualenv**.

3.62.4.2 Installing Python 3.7.6

Procedure

Step 1 Download the Python source package.

```
cd /path/to/GALAXY
wget https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tar.xz
```

Step 2 Decompress the installation package and perform the installation.

```
tar -xvf Python-3.7.6.tar.xz
cd python-3.7.6
./configure --prefix=/path/to/GALAXY/python3.7.6
make -j && make -install
```

Step 3 Add the environment variable.

```
vi ~/.bashrc
```

Step 4 Add the following content to the end of the file:

```
export PATH=/path/to/GALAXY/python3.7.6/bin:$PATH
```

Step 5 Exit the insert mode. Run the following command to save the file and exit.

```
wq!
```

Step 6 Enable this file.

```
source ~/.bashrc
```

Step 7 Perform the verification.

```
python -V
```

----End

3.62.4.3 Installing virtualenv

Procedure

Step 1 Install virtualenv.

```
pip install virtualenv
```

NOTE

If either of the following errors occurs, retry the installation using another mirror source.

ERROR: Could not find a version that satisfies the requirement virtualenv (from versions:)

ERROR: No matching distribution found for virtualenv

Step 2 (Optional) Install virtualenv.

```
pip install virtualenv -i Mirror source address
```

----End

3.62.5 Obtaining the Source Code

Procedure

Step 1 Download the Galaxy source code.

https://github.com/galaxyproject/galaxy/archive/refs/heads/master.zip

Step 2 Use an SFTP tool to upload the Galaxy installation package to the */path/to/GALAXY* directory on the server.

----End

3.62.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/GALAXY/
unzip master.zip
mv galaxy-master/ galaxy_install/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the following directory and enable the Python virtual environment.

```
cd galaxy_install
python -m virtualenv .env/
source .env/bin/activate
```

Step 4 Set the mirror source.

```
pip config set global.index-url Mirror source address
```

Step 5 Perform the installation.

```
sh run.sh
```

----End

3.62.7 Running and Verification

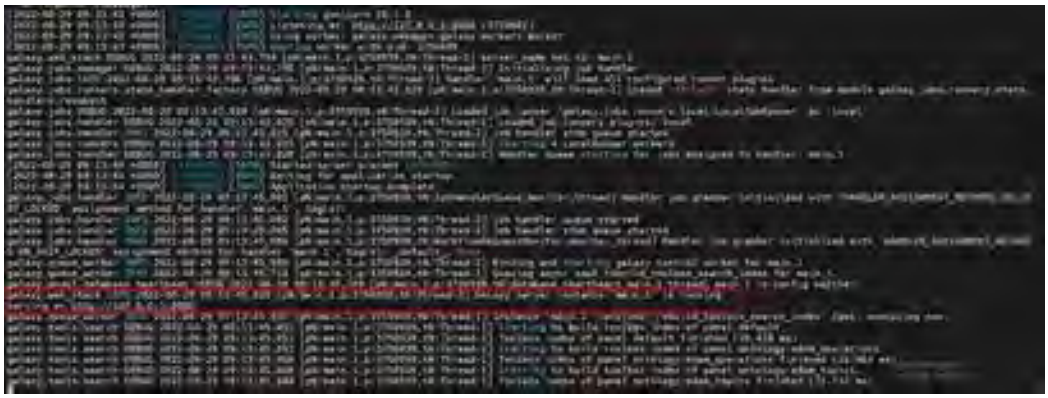
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/GALAXY/galaxy_install
sh run.sh
```

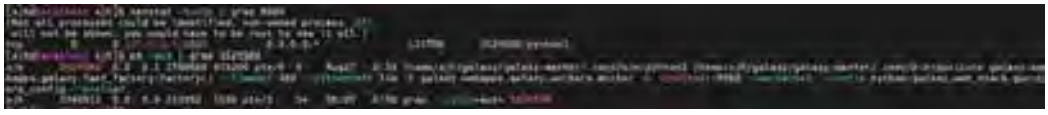
If the following information is displayed, the program is running properly.



Step 3 (Optional) Check whether the port is in use.

```
netstat -tunlp | grep 8080
ps -aux | grep pid (In this command, pid indicates the PID of the process that occupies port 8080.)
```

If the following information is displayed, the program is running properly.



----End

3.63 NWChem 6.8.1 Porting Guide (Kylin V10)

3.63.1 Introduction

NWChem is computational chemistry software running on high-performance parallel supercomputers and common workstation clusters. NWChem uses

standard quantum mechanics to describe electron wave functions or densities, to calculate the properties of molecules and periodic systems, and to simulate classical molecular dynamics and free energy.

For more information, visit the **NWChem page at GitHub**.

Programming language: C++/Fortran

Brief description: A scalable computational chemistry tool.

Open source license: ECL 2.0

Recommended Software Version

NWChem 6.8.1

3.63.2 Environment Requirements

Hardware Requirements

Table 3-319 lists the hardware requirements.

Table 3-319 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 3-320 lists the software requirements.

Table 3-320 Software requirements

Item	Version	Download URL
NWChem	6.8.1	**https://codeload.github.com/nwchemgit/nwchem/tar.gz/refs/tags/6.8.1-release**
Kunpeng GCC	9.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz**
HMPI	1.1.0	**https://mirrors.huaweicloud.com/hmpi/Hyper-MPI_1.1.0_openEuler-aarch64.tar.gz**
Nvidia A100 driver	470.57.02	Included in the CUDA toolkit
NVIDIA CUDA	11.4.1	**https://developer.download.nvidia.com/compute/cuda/11.4.1/local_installers/cuda_11.4.1_470.57.02_linux_sbsa.run**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 3-321 lists the OS requirements.

Table 3-321 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

3.63.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NWChem software porting.

Table 3-322 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/compiler	Installation path of the Kunpeng GCC.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/opt/HMPI	Installation path of HMPI.	
3	/usr/local/cuda-11.4	Installation path of NVIDIA CUDA.	
4	/path/to/nwchem	Installation path of NWChem.	
5	/path/to/testcase	Path for storing NWChem test cases.	

3.63.4 Configuring the Compilation Environment

3.63.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 3-323 Configuration process

No.	Configuration Item	Remarks
1	Disabling the Nouveau driver	See **3.63.4.2 Disabling the Nouveau Driver**.
2	Installing NVIDIA CUDA	See **3.63.4.3 Installing NVIDIA CUDA (Including the GPU Driver)**.
3	Installing the BiSheng Compiler	See "Installing the BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
4	Installing HMPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.

3.63.4.2 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Disable the Nouveau driver.

```
vim /etc/modprobe.d/disable-nouveau.conf
```

Add the following two lines at the beginning of the file:

```
blacklist nouveau
options nouveau modeset=0
```

Step 3 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 4 Restart the server.

```
reboot
```

----End

3.63.4.3 Installing NVIDIA CUDA (Including the GPU Driver)

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled (no command output is displayed).

```
lsmod |grep nouveau
```

Step 3 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.1/local_installers/
cuda_11.4.1_470.57.02_linux_sbsa.run
sh cuda_11.4.1_470.57.02_linux_sbsa.run
```

 Copyright © Huawei Technologies Co., Ltd.

Install all components as prompted.

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 5 Check the NVCC version.

```
nvcc -V
```

----End

3.63.5 Obtaining the Source Code

Procedure

Step 1 Download the **nwchem-6.8.1-release.tar.gz** installation package.

https://codeload.github.com/nwchemgit/nwchem/tar.gz/refs/tags/6.8.1-release

Step 2 Downloading the dependency package **ga-5.6.5.tar.gz**.

https://github.com/GlobalArrays/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz

Step 3 Use an SFTP tool to upload the installation package to the */path/to/nwchem* directory on the server.

----End

3.63.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/nwchem
```

Step 3 Decompress the installation package.

```
tar -xvf nwchem-6.8.1-release.tar.gz
```

Step 4 Modify the configuration file.

```
cp ga-5.6.5.tar.gz nwchem-6.8.1-release/src/tools
```

Step 5 Go to the directory generated after the decompression.

```
cd nwchem-6.8.1-release/src
```

Step 6 Set the environment variables.

```
export NWCHEM_TOP=/path/to/nwchem/nwchem-6.8.1-release
export LARGE_FILES=TRUE
export NWCHEM_TARGET=LINUX64
export NWCHEM_MODULES=all
export ARMCI_NETWORK=MPI-PR
export USE_MPI=y
export USE_MPIF=y
export USE_MPIF4=y
export USE_INTERNALBLAS=y
export TCE_CUDA=y
export CUDA_LIBS="-L/usr/local/cuda/lib64 -lcudart -lcublas"
export CUDA_FLAGS="-arch sm_80"
```

 Copyright © Huawei Technologies Co., Ltd.

```
export CUDA_INCLUDE="-I. -I/usr/local/cuda/include"
export CUDA=nvcc
```

Step 7 Perform the compilation and installation.

```
make nwchem_config
make
```

Step 8 Check whether an executable file is generated.

```
ls -l /path/to/nwchem/nwchem-6.8.1-release/bin/LINUX64/nwchem
```

If the following information is displayed, the installation is successful.

```
[root@2280node26 nwchem-6.8.1-release]# ll bin/LINUX64/nwchem
-rwxr-xr-x 1 root root 78234104 Jan 26 14:45 bin/LINUX64/nwchem
```

Step 9 Set the environment variables.

```
export PATH=/path/to/nwchem/nwchem-6.8.1-release/bin/LINUX64:$PATH
```

----End

3.63.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Switch to the test directory.

```
cd /path/to/testcase
```

Step 3 Copy the test cases to the test directory.

```
cp test1.nw /path/to/testcase
```

Step 4 Run the NWChem test.

```
mpirun -np 48 nwchem test1.nw
```

There is a main process when NWChem is running. The main process does not participate in parallel computing. Therefore, the number of actually calculated cores is the number of **np** minus 1. The value of **np** must be at least 2.

After the running is complete, the total running duration is printed. The following figure shows an example of the test result:

 Copyright © Huawei Technologies Co., Ltd.

```
 ------------------------------------------------------------------
 Iterations converged
 CCSD correlation energy / hartree =        -0.848328021034623
 CCSD total energy / hartree       =      -231.574950857314974

 Singles contributions

 Doubles contributions
 CCSD(T)
 Using CUDA CCSD(T) code
Using   1 device per node

 CCSD[T]  correction energy / hartree =        -0.040845613835246
 CCSD[T] correlation energy / hartree =        -0.889173634869869
 CCSD[T] total energy / hartree       =      -231.615796471150219
 CCSD(T)  correction energy / hartree =        -0.040170097797498
 CCSD(T) correlation energy / hartree =        -0.888498118832120
 CCSD(T) total energy / hartree       =      -231.615120955112474
 Cpu & wall time / sec            11.8           25.5

 Parallel integral file used    2074 records with       0 large values


 Task  times  cpu:       60.2s     wall:       77.9s
```

----End

3.63.8 Comparison Test

3.63.8.1 Performance Test

Note:

(1) Only the TCE CCSD(T) feature of NWChem supports GPU computing.

(2) This test case comes from the Internet. Only the last CCSD computing part uses GPUs, and other computing parts use CPUs. The computing effect is poor. **Table 3-324** shows the comparison result.

Table 3-324 Effect comparison

Test Command	Taishan 200 2280 (2 x 7260)	G5500 (2 x 6248R)
CPU	114 seconds	83.8 seconds
CPU + GPU	121 seconds	77.9 seconds

 Copyright © Huawei Technologies Co., Ltd.

3.63.9 Change History

Table 3-325

Date	Description
2021-07-27	This issue is the first official release.

3.64 T-Flows 0.9 Porting Guide (Kylin V10)

3.64.1 Introduction

T-Flows is a computational fluid dynamics (CFD) program for simulation of turbulent, single and multiphase flows. Numerical method is based on collocated finite volume method on unstructured arbitrary grids and turbulence models include a range of Reynolds-averaged Navier-Stokes (RANS) models, large eddy simulations (LES), as well as hybrid RANS-LES approach. It is written in Fortran 90 and executed in parallel using the Message Passing Interface (MPI).

For more information, visit the **T-Flows page at GitHub**.

Programming language: Fortran

Brief description: A CFD program.

Open source license: MIT License

Recommended Software Version

v0.9

3.64.2 Environment Requirements

Hardware Requirements

Table 3-326 lists the hardware requirements.

Table 3-326 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 3-327 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-327 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**
Hyper MPI	1.1.1	**https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz**
metis	5.1.0	**https://github.com/scivision/METIS/archive/refs/tags/v5.1.0.1.tar.gz**
T-Flows	0.9	**https://github.com/DelNov/T-Flows/archive/refs/tags/v0.9.tar.gz**

OS Requirements

Table 3-328 lists the OS requirements.

Table 3-328 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

3.64.3 Paths for Software Porting

This chapter lists the software installation paths involved in the T-Flows software porting.

Table 3-329 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/T-FLOWS	Installation path of T-Flows.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/metis	Installation path of metis.	

 Copyright © Huawei Technologies Co., Ltd.

3.64.4 Configuring the Compilation Environment

3.64.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-330 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See "Installing the BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
2	Installing HMPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
3	Installing metis	See **3.64.4.2 Installing metis**.

3.64.4.2 Installing metis

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y cmake
```

Step 3 Decompress the installation package.

```
tar zxvf v5.1.0.1.tar.gz
```

Step 4 Switch to the metis source code directory.

```
cd METIS-5.1.0.1
```

Step 5 Compile and install metis.

```
cmake .. -DCMAKE_INSTALL_PREFIX=../install
make
```

----End

3.64.5 Obtaining the Source Code

Procedure

Step 1 Download the T-Flows source package **v0.9.tar.gz**.

https://github.com/DelNov/T-Flows/archive/refs/tags/v0.9.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the T-Flows source package to the */path/to/T-FLOWS* directory on the server.

----End

3.64.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/T-FLOWS
```

Step 3 Decompress the source package.

```
tar -xvf v0.9.tar.gz
```

Step 4 Copy the dependency files.

```
cp /path/to/metis/METIS-5.1.0.1/install/lib/libmetis.a /path/to/T-FLOWS/T-Flows-0.9/Sources/Libraries/Metis_5.1.0_Linux_64/
```

Step 5 Modify **Makefile**.

Modify **Makefile** in each folder of */path/to/T-FLOWS/***T-Flows-0.9/Sources/**.

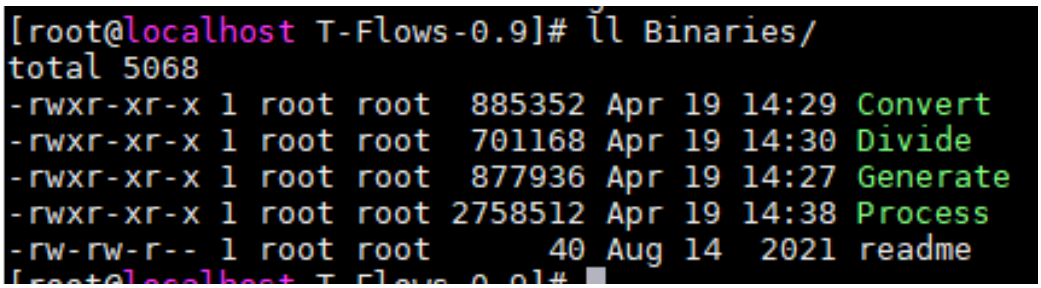
Change **FC = gfortran** to **FC = flang**.

Step 6 Compile T-Flows.

```
cd /path/to/T-FLOWS/T-Flows-0.9/Sources/Generate
make
cd /path/to/T-FLOWS/T-Flows-0.9/Sources/Convert
make
cd /path/to/T-FLOWS/T-Flows-0.9/Sources/Divide
make
cd /path/to/T-FLOWS/T-Flows-0.9/Sources/Process
make
```

Step 7 After the compilation is complete, an executable file is generated in **T-Flows-0.9/Binaries**.

If the following information is displayed, the installation is successful.



----End

 Copyright © Huawei Technologies Co., Ltd.

3.64.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Load the software directory.

```
export PATH=/path/to/T-FLOWS/T-Flows-0.9/Binaries:$PATH
```

Step 3 Switch to the test file directory of the source code.

```
cd /path/to/T-FLOWS/T-Flows-0.9/Tests/Rans/Backstep_Re_28000
```

Step 4 Perform the verification.

```
Generate
```

Enter **backstep**.

Enter **1**.

Enter **x**.

Enter **xy**.

 Copyright © Huawei Technologies Co., Ltd.

```
 # Grid currently has the following boundary conditions:
 #-------------------------------------------------------
 #  1. ADIABATIC_WALL
 #  2. HEATED_WALL
 #  3. INLET_FACE
 #  4. OUTLET_FACE
 #-------------------------------------------------------
 #=============================================================
 # Calculating distance from the walls
 #-------------------------------------------------------------
 # Type ordinal number(s) of wall or wall_flux boundary condition(s)
 # from the boundary condition list (see above) separated by space.
 # Cells' centers distances to the nearest wall will be calculated
 # for the listed wall boundary(s).
 #
 # This is needed for RANS and HYBRID turbulence models as well as
 # for proper initialization with potential pressure-like field.
 #
 # Type skip to skip this and set wall distance to one everywhere.
 #-------------------------------------------------------------
1
 # Distance to the wall calculated !
 # Creating the file: backstep.cfn
 # Creating the file: backstep.dim
 # Creating the file: backstep.vtu
 # Creating the file: backstep.faces.vtu
 # Creating the file: backstep.shadows.vtu
 #=========================================
 # Creating 1d file with the node
 # coordinates in non-homogeneous directions
 #-----------------------------------------
 # Insert non-homogeneous direction
 # (x, y, z, rx, ry, rz or skip)
 # -----------------------------------------
x
 # Creating the file: backstep.1d
 #===============================
 # Looking for homogeneous plane
 #-------------------------------
 # Insert homogeneous direction
 # (xy, yz, zx or skip)
 #-------------------------------
xy
 # Creating the file: backstep.2d
 #=========================================================
 #
```

The following information is displayed:

```
# Grid BACKSTEP statistics
#
#------------------------------------------------------
# Bounding box:
#   X range: -1.444E-01  to:  2.280E+00
#   Y range:  0.000E+00  to:  1.000E-01
#   Z range:  0.000E+00  to:  1.900E-01
#- - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Number of faces:             116120
# Number of shadows:             9600
#
# Face shape breakdown:
#   quadrilaterals:    116120
#- - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Number of bnd. cells:       1840
# Number of inside cells:    38400
#
# Cell shape breakdown:
#   quadrilaterals:      1840
#   hexahedra:          38400
#------------------------------------------------------
[root@localhost Backstep_Re_28000]#
```

----End

3.64.8 Change History

Table 3-331

Date	Description
2022-04-28	This issue is the first official release.

3.65 yambo 5.1.1 Porting Guide (openEuler 20.03)

3.65.1 Introduction

Yambo implements Many-Body Perturbation Theory (MBPT) methods (such as GW and BSE) and Time-Dependent Density Functional Theory (TDDFT), which allows for accurate prediction of fundamental properties as band gaps of semiconductors, band alignments, defect quasi-particle energies, optics and out-of-equilibrium properties of materials.

For more information, visit the **yambo page at GitHub**.

Programming language: Fortran

Brief description: An MBPT software package.

Open source license: GPL 2.0

NOTE

When using open source software, comply with the applicable licenses.

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

5.1.1

3.65.2 Environment Requirements

Hardware Requirements

Table 3-332 lists the hardware requirements.

Table 3-332 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 3-333 lists the OS requirements.

Table 3-333 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 3-334 lists the software requirements.

Table 3-334 Software requirements

Item	Version	Download URL
yambo	5.1.1	https://github.com/yambo-code/yambo/archive/refs/tags/5.1.1.tar.gz

3.65.3 Paths for Software Porting

This chapter lists the software installation paths involved in the yambo software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 3-335 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/yambo	Installation path of yambo.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

3.65.4 Configuring the Compilation Environment

3.65.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 3-336 Configuration process

No.	Configuration Item	Remarks
1	Installing dependencies	See **3.65.4.2 Installing Dependencies**.

3.65.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies using the **yum** command.

```
yum install gcc gcc-c++ libstdc++-devel autoconf automake
```

----End

3.65.5 Obtaining the Source Code

Procedure

Step 1 Download the yambo installation package.

https://github.com/yambo-code/yambo/archive/refs/tags/5.1.1.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the yambo installation package to the */path/to/yambo* directory on the server.

----End

3.65.6 Compilation and Installation

Procedure

Step 1 Go to the following directory:

```
cd /path/to/yambo
tar -xvf 5.1.1.tar.gz
cd yambo-5.1.1
```

Step 2 Modify the configuration file.

```
sed -i '6530c\ i?86*linux* | ia64*linux* | *x86*64* | *86*cygwin | aarch64*linux* ) ' ./configure
```

Step 3 Create a build file.

```
./configure
```

Step 4 Perform the compilation.

```
make core
```

----End

3.65.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/yambo/yambo-5.1.1/bin
./yambo -version
```

Command output:

```
This is yambo - Serial+HDF5_IO - Ver. 5.1.0 Revision 21422 Hash fde6e2a07
```

3.65.8 Change History

Date	Description
2023-04-26	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

4 Life Science

4.1 BLAST 2.9.0 Porting Guide (CentOS 7.6)
4.2 BWA 0.7.17 Porting Guide (CentOS 7.6)
4.3 CNVnator 0.4.1 Porting Guide (CentOS 7.6)
4.4 Cufflinks 2.2.1 Porting Guide (CentOS 7.6)
4.5 GATK 4.0.0.0 Porting Guide (CentOS 7.6)
4.6 HISAT2 2.1.0 Porting Guide (CentOS 7.6)
4.7 STAR 2.7.1a Porting Guide (CentOS 7.6)
4.8 TopHat2 2.1.1 Porting Guide (CentOS 7.6)
4.9 Bowtie2 2.4.1 Porting Guide (CentOS 7.6)
4.10 FastQC 0.11.9 Porting Guide (CentOS 7.6)
4.11 Beagle 5.1 Porting Guide (CentOS 7.6)
4.12 Bowtie 1.2.3 Porting Guide (CentOS 7.6)
4.13 pblat 2.1 Porting Guide (CentOS 7.6)
4.14 SnpEff V4_3 Porting Guide (CentOS 7.6)
4.15 Velvet 1.2.10 Porting Guide (CentOS 7.6)
4.16 XHMM Porting Guide (CentOS 7.6)
4.17 pysam 0.15.4 Porting Guide (CentOS 7.6)
4.18 Abyss 2.2.4 Porting Guide (CentOS 7.6)
4.19 Chaste 2019.1 Porting Guide (CentOS 7.6)
4.20 COPASI 4.27.217 Porting Guide (CentOS 7.6)
4.21 FASTA 36.3.8 Porting Guide (CentOS 7.6)
4.22 GMAP 2015-9-21 Porting Guide (CentOS 7.6)
4.23 VarScan 2.4.2 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.24 Minimap2 Porting Guide (CentOS 7.6)
4.25 bedtools 2.29.2 Porting Guide (CentOS 7.6)
4.26 BEAST2 2.6.3 Porting Guide (CentOS 7.6)
4.27 bcftools 1.10.2 Porting Guide (CentOS 7.6)
4.28 Picard 2.23.3 Porting Guide (CentOS 7.6)
4.29 SOAPdenovo r241 Porting Guide (CentOS 7.6)
4.30 Blastz 2004-12-27 Porting Guide (CentOS 7.6)
4.31 CAFE 5.0b2 Porting Guide (CentOS 7.6)
4.32 fastp Porting Guide (CentOS 7.6)
4.33 GCE 1.0.2 Porting Guide (CentOS 7.6)
4.34 GEMMA 0.98.1 Porting Guide (CentOS 7.6)
4.35 kmergenie 1.7051 Porting Guide (CentOS 7.6)
4.36 LASTZ 1.04.03 Porting Guide (CentOS 7.6)
4.37 MAFFT Porting Guide (CentOS 7.6)
4.38 megahit 1.2.9 Porting Guide (CentOS 7.6)
4.39 PASA 2.4.1 Porting Guide (CentOS 7.6)
4.40 Augustus 3.3.3 Porting Guide (CentOS 7.6)
4.41 Delly 0.8.5 Porting Guide (CentOS 7.6)
4.42 FastTree 2.1.11 Porting Guide (CentOS 7.6)
4.43 Jellyfish 2.2.10 Porting Guide (CentOS 7.6)
4.44 PLINK 1.9 Porting Guide (CentOS 7.6)
4.45 SNAP 2013-11-29 Porting Guide (CentOS 7.6)
4.46 kmersGWAS 0.2 Porting Guide (CentOS 7.6)
4.47 PAML 4.9j Porting Guide (CentOS 7.6)
4.48 Diamond 2.0.4 Porting Guide (CentOS 7.6)
4.49 HISAT 0.1.6 Porting Guide (CentOS 7.6)
4.50 MACS2 2.1.4 Porting Guide (CentOS 7.6)
4.51 Trinity 2.11.0 Porting Guide (CentOS 7.6)
4.52 LDBlockshow 1.36 Porting Guide (CentOS 7.6)
4.53 bioBakery Porting Guide (CentOS 8.2)
4.54 Bioconda Porting Guide (CentOS 8.2)
4.55 Boost 1.72 Porting Guide (openEuler 20.03)
4.56 Bustools 0.41.0 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

4.57 Circos 0.69.9 Porting Guide (CentOS 8.2)
4.58 cisTEM 1.0 Porting Guide (CentOS 7.6)
4.59 EMAN2 2.91 Porting Guide (CentOS 8.2)
4.60 EMBOSS 6.5.7 Porting Guide (CentOS 7.6)
4.61 HumanN 3.0.1 Porting Guide (CentOS 8.2)
4.62 kallisto 0.46.2 Porting Guide (CentOS 8.2)
4.63 kb-python 0.26.4 Porting Guide (CentOS 8.2)
4.64 KneadData 0.7.7 Porting Guide (CentOS 8.2)
4.65 Kraken2 Porting Guide (CentOS 8.2)
4.66 MetaPhlAn 3.0.13 Porting Guide (CentOS 8.2)
4.67 muparser v2.3.2 Porting Guide (openEuler 20.03)
4.68 Open CASCADE Technology 7.5.0 Porting Guide (openEuler 20.03)
4.69 PYCXX 6.2.8 Porting Guide (openEuler 20.03)
4.70 Python 3.8.2 Porting Guide (openEuler 20.03)
4.71 Qt 5.13.0 Porting Guide (openEuler 20.03)
4.72 RSeQC 4.0.0 Porting Guide (CentOS 8.2)
4.73 Sniffles 1.0.12 Porting Guide (CentOS 7.6)
4.74 SPIDER 24.08 Porting Guide (CentOS 7.6)
4.75 VEP 104.3 Porting Guide (CentOS 7.6)
4.76 VTK 9.0.1 Porting Guide (openEuler 20.03)
4.77 BWA 0.7.17 Porting Guide (openEuler 21.03)
4.78 ABSOLUTE 1.0.6 Porting Guide (CentOS 7.6)
4.79 GDS3D 1.9 Porting Guide (CentOS 7.6)
4.80 HOMER 4.11 Porting Guide (CentOS 8.2)
4.81 mCaller 1.0.3 Porting Guide (CentOS 8.2)
4.82 Muscle 5.0.1428 Porting Guide (CentOS 7.9)
4.83 Nanopolish 0.13.2 Porting Guide (CentOS 8.2)
4.84 Octave 6.4.0 Porting Guide (CentOS 8.2)
4.85 PanPhlAn 3.1 Porting Guide (openEuler 20.03)
4.86 PhyloPhlAn 3.0.60 Porting Guide (CentOS 8.2)
4.87 PyClone 0.13.1 Porting Guide (CentOS 8.2)
4.88 pyem v0.5 Porting Guide (CentOS 8.2)
4.89 PyTorch 1.9 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

4.90 SURVIVOR 1.0.6 Porting Guide (CentOS 8.2)
4.91 RELION 3.1.2 Porting Guide (CentOS 8.2)
4.92 VMD 1.9.4 Porting Guide (CentOS 8.2)
4.93 DBG2OLC Porting Guide (openEuler 20.03)
4.94 MAKER 2.31.11 Porting Guide (openEuler 20.03)
4.95 Platanus 1.2.4 Porting Guide (openEuler 20.03)
4.96 TASSEL 5.0 Porting Guide (openEuler 20.03)
4.97 MWASTools 1.20.0 Porting Guide (Kylin V10)
4.98 sRAP 1.26.0 Porting Guide (Kylin V10)
4.99 BRAKER 2.1.6 Porting Guide (Kylin V10)
4.100 Cactus 2.1.0 Porting Guide (Kylin V10)
4.101 CAFE 4.2.1 Porting Guide (Kylin V10)
4.102 Clustal X 2.1 Porting Guide (Kylin V10)
4.103 CNVfilteR 1.10.0 Porting Guide (Kylin V10)
4.104 DIAMOND 2.0.15 Porting Guide (Kylin V10)
4.105 Exonerate 2.4.0 Porting Guide (Kylin V10)
4.106 FreeBayes 1.3.6 Porting Guide (Kylin V10)
4.107 GMAP 2021-12-17 Porting Guide (Kylin V10)
4.108 GUIDEseq 1.26.0 Porting Guide (Kylin V10)
4.109 HiCUP v0.8.3 Porting Guide (Kylin V10)
4.110 Jellyfish 2.3.0 Porting Guide (Kylin V10)
4.111 KOBAS 3.0.3 Porting Guide (Kylin V10)
4.112 LTR_Finder Porting Guide (Kylin V10)
4.113 LTR_retriever 2.9.0 Porting Guide (Kylin V10)
4.114 MAFFT 7.490 Porting Guide (Kylin V10)
4.115 Meryl 1.3 Porting Guide (Kylin V10)
4.116 Minigraph 0.19 Porting Guide (Kylin V10)
4.117 MISA 25082020 Porting Guide (Kylin V10)
4.118 MODELTEST 0.1.7 Porting Guide (Kylin V10)
4.119 NINJA 1.2.2 Porting Guide (Kylin V10)
4.120 NOVOPlasty 4.3.1 Porting Guide (Kylin V10)
4.121 OrthoMCL 2.0.9 Porting Guide (Kylin V10)
4.122 PAL2NAL 14 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

4.123 PASA r20130907 Porting Guide (Kylin V10)
4.124 Picard 2.27.3 Porting Guide (Kylin V10)
4.125 ProtTest 3.4.2 Porting Guide (Kylin V10)
4.126 Purge_Dups 1.2.6 Porting Guide (Kylin V10)
4.127 Ragout 2.3 Porting Guide (Kylin V10)
4.128 RepeatMasker 4.1.2 Porting Guide (Kylin V10)
4.129 RepeatModeler 2.0.1 Porting Guide (Kylin V10)
4.130 RepeatScout 1.0.6 Porting Guide (Kylin V10)
4.131 RMBlast 2.10.0 Porting Guide (Kylin V10)
4.132 RSEM 1.3.3 Porting Guide (Kylin V10)
4.133 CANU 1.8 Porting Guide (CentOS 7.6)
4.134 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6)
4.135 CPAT 3.0.4 Porting Guide (Kylin V10)
4.136 MrBayes 3.2.7a Porting Guide (Kylin V10)
4.137 MSMC 1.1.0 Porting Guide (Kylin V10)
4.138 Prodigal 2.6.3 Porting Guide (Kylin V10)
4.139 Vaa3D 2.2.0.0 Porting Guide (openEuler 20.03)
4.140 WGCNA 1.70 Porting Guide (Kylin V10)
4.141 AutoDock 4.2.6 Porting Guide (Kylin V10)
4.142 bioconductor-manta 1.32.0 Porting Guide (Kylin V10)
4.143 CellSys 0.1.0 Porting Guide (Kylin V10)
4.144 cutadapt 4.1 Porting Guide (Kylin V10)
4.145 deepTools 3.5.1 Porting Guide (Kylin V10)
4.146 DESeq2 1.36.0 Porting Guide (Kylin V10)
4.147 featurecounts 2.0.3 Porting Guide (Kylin V10)
4.148 ggplot2 3.3.6 Porting Guide (Kylin V10)
4.149 MultiQC 1.12 Porting Guide (Kylin V10)
4.150 Open Babel 3.1.1 Porting Guide (Kylin V10)
4.151 PBSuite 15.8.24 Porting Guide (Kylin V10)
4.152 psipred 4.0 Porting Guide (Kylin V10)
4.153 SNPhylo 20180901 Porting Guide (Kylin V10)
4.154 TreeBeST 1.9.2 Porting Guide (Kylin V10)
4.155 BLASR Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

4.156 BreakDancer Porting Guide (Kylin V10)
4.157 Mugsy 1.2.3 Porting Guide (Kylin V10)
4.158 AlphaFold Porting Guide (CentOS 8.2)
4.159 Glimmer v3.02 Porting Guide (Kylin V10)
4.160 HaploView 4.1 Porting Guide (Kylin V10)
4.161 HH-suite v3.3.0 Porting Guide (Kylin V10)
4.162 art-nextgen-simulation-tools 20160605 Porting Guide (Kylin V10)
4.163 bioconductor-appreci8r 1.14.0 Porting Guide (Kylin V10)
4.164 ANGSD 0.938 Porting Guide (Kylin V10)
4.165 bioconductor-decontam 1.16.0 Porting Guide (Kylin V10)
4.166 bioconductor-clippda 1.46.0 Porting Guide (Kylin V10)
4.167 bioconductor-bubbletree 2.26.0 Porting Guide (Kylin V10)
4.168 AMAS 1.2 Porting Guide (Kylin V10)
4.169 bioconductor-ebcoexpress 1.42.0 Porting Guide (Kylin V10)
4.170 athena_meta 1.3 Porting Guide (Kylin V10)
4.171 bioconductor-GARS 1.16.0 Porting Guide (Kylin V10)
4.172 CD-HIT 4.8.1 Porting Guide (Kylin V10)
4.173 bioconductor-grohmm 1.30.1 Porting Guide (Kylin V10)
4.174 bioconductor-MAGeCKFlute 2.0.0 Porting Guide (Kylin V10)
4.175 bioconductor-intansv 1.36.0 Porting Guide (Kylin V10)
4.176 bioconductor-multihiccompare 1.14.0 Porting Guide (Kylin V10)
4.177 ClustalW 2.1 Porting Guide (Kylin V10)
4.178 HMMER 3.3.2 Porting Guide (Kylin V10)
4.179 IDBA-UD 1.0 Porting Guide (Kylin V10)
4.180 InterProScan 5.53-87.0 Porting Guide (Kylin V10)
4.181 MEME 5.4.0 Porting Guide (Kylin V10)
4.182 MetaVelvet 1.2.01 Porting Guide (Kylin V10)
4.183 mothur 1.48 Porting Guide (Kylin V10)
4.184 PHYLIP 3.697 Porting Guide (Kylin V10)
4.185 Primer3 2.4.0 Porting Guide (Kylin V10)
4.186 Sambamba 0.8.2 Porting Guide (openEuler 20.03)
4.187 SOAPnuke 2.1.7 Porting Guide (Kylin V10)
4.188 SOAPsnp Master Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

4.189 trimmomatic 0.33.0 Porting Guide (Kylin V10)

4.190 tRNAscan-SE 2.0.9 Porting Guide (Kylin V10)

4.191 vcfSampleCompare 2.013 Porting Guide (Kylin V10)

4.192 VelvetOptimiser 2.2.6 Porting Guide (Kylin V10)

4.193 wtdbg 1.1.006 Porting Guide (Kylin V10)

4.194 wtdbg2 2.5 Porting Guide (Kylin V10)

4.195 xpressplot 0.2.5 Porting Guide (Kylin V10)

4.1 BLAST 2.9.0 Porting Guide (CentOS 7.6)

4.1.1 Introduction

Basic Local Alignment Search Tool (BLAST) is an analysis tool used for similarity comparison in the protein database or DNA database. The BLAST program can quickly find the similarity between the object sequence and that recorded in public databases. The score in the BLAST result is a statistical description of similarity.

BLAST uses a local algorithm to obtain sequences that are similar in two sequences.

BLAST compares one or more sequences (in any form) To one or more nucleic acid or protein sequence libraries. BLAST can also find sequences with matched gaps.

Based on the method published by Altschul et al. on J. Mol. Biol (J. Mol. Biol. 215: 403-410 (1990)), BLAST checks/analyzes the homology between queried sequences and those recorded in sequence databases. Compared with the initial BLAST, BLAST 2.0 provided by NCBI adopts gap penalty in sequence alignment. BLAST can process any number of sequences, including protein sequences and nucleic acid sequences. Multiple databases of the same type can be selected, that is, either all protein databases or all nucleic acid databases. The queried sequence and the invoked database may be a combination of any form. You can analyze a nucleic acid sequence or protein sequence with a protein or nucleic acid database.

Software packages such as GCG and EMBOSS contain five types of BLAST:

- BLASTP is used to query protein sequences in protein databases. It compares each queried sequence with each sequence recorded in databases one by one.
- BLASTX is used to query nucleic acid sequences in protein databases. First, it translates the nucleic acid sequence into a protein sequence (translates one nucleic acid sequence into six possible proteins), and then performs a one-to-one protein sequence comparison on each of the protein sequences.
- BLASTN is used to query nucleic acid sequences in nucleic acid databases. It compares each known sequence in databases with each queried nucleic acid sequence one by one.
- TBLASTN is used to query protein sequences in nucleic acid databases. In contrast to BLASTX, it translates nucleic acid sequences in databases into protein sequences, and then compares them with the queried sequences.
- TBLASTX is used to query nucleic acid sequences in nucleic acid databases. It translates both the nucleic acid sequences in databases and the queried

 Copyright © Huawei Technologies Co., Ltd.

nucleic acid sequences into proteins (each nucleic acid sequence produces six possible protein sequences), which produces 36 comparison arrays each time.

For more information, visit the **official BLAST website**.

Programming language: C++

Brief description: An analysis tool used for similarity comparison in the protein database or DNA database.

Open source license: PUBLIC DOMAIN

Recommended Software Version

BLAST 2.9.0

4.1.2 Environment Requirements

Hardware Requirements

Table 4-1 lists the hardware requirements.

Table 4-1 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-2 lists the software requirements.

Table 4-2 Software Requirements

Item	Version	Download Address
BLAST	2.9.0	**http://mirror.ufs.ac.za/blast/executables/blast%2B/2.9.0/ncbi-blast%2B-2.9.0-src.tar.gz**
LMDB	0.9.24	**https://github.com/LMDB/lmdb/archive/refs/tags/LMDB_0.9.24.tar.gz**
Test case 1	protein_query.fastas	**https://gitlab.com/arm-hpc/packages/uploads/1acce57b88cc76f6c89a087e449456b5/protein_query.fasta**
Test case 2	SWISSPROT	**http://mirror.ufs.ac.za/blast/db/FASTA/swissprot.gz**

OS Requirements

Table 4-3 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-3 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

4.1.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BLAST software porting.

Table 4-4 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/LMDB	Installation path of LMDB	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/BLAST	Installation path of BLAST	
4	/path/to/CASE	Path for storing the test cases	

4.1.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-5 Configuration process

No.	Configuration Item	Description
1	Configure the basic environment.	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Description
2	Installing LMDB	Install tools by following the instructions in **LMDB**.

4.1.5 Obtaining the Source Code

Procedure

Step 1 Download the BLAST installation package **ncbi-blast-2.9.0+-src.tar.gz**.

https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.9.0/

Step 2 Use an SFTP tool to upload the BLAST installation package to the *path/to/BLAST* directory on the server.

----End

4.1.6 Compiling and Installing BLAST

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress and configure the BLAST path:

```
tar xzf ncbi-blast-2.9.0+-src.tar.gz
cd ncbi-blast-2.9.0+-src
export BLAST_TOP=`pwd`
```

Step 3 Run the following command to configure the compiler:

```
export CC=gcc
export CXX=g++
```

Step 4 Use an SFTP tool to upload **LMDB_0.9.24.tar.gz** to the */path/to/BLAST***/ncbi-blast-2.9.0+-src** directory.

Step 5 Run the following command to decompress and configure the LMDB path:

```
cd $BLAST_TOP
tar -xvf LMDB_0.9.24.tar.gz
cd lmdb-LMDB_0.9.24
export LMDB_PATH=$BLAST_TOP/lmdb-LMDB_0.9.24
```

Step 6 Run the following commands to compile and install BLAST:

```
cd $BLAST_TOP/c++
./configure
cd ReleaseMT/build
make all_r -j40
```

Step 7 After the installation is successful, the executable file is generated in the **$BLAST_TOP/c++/ReleaseMT/bin** directory. You can run the following command to view the file:

```
cd $BLAST_TOP/c++/ReleaseMT/bin
ls blastp
blastp
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.1.7 Running and Verifying BLAST

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use an SFTP tool to upload the test case to the */path/to/CASE* directory.

Step 3 Run the following commands to decompress and rename Swissprot Proteins data:

```
cd /path/to/CASE
gzip -d swissprot.gz
mv swissprot swissprot.fa
```

Step 4 Run the following commands to save all files to the **data** directory:

```
mkdir data
mv swissprot.fa data/
mv protein_query.fasta data/
```

Step 5 Run the following commands to compile the test case data.

```
export PATH=$BLAST_TOP/c++/ReleaseMT/bin:$PATH
makeblastdb -in data/swissprot.fa -dbtype prot
```

After the commands are executed, the following four files are generated in the **data** directory. You can run the **ls** command to view the files.

- swissprot.fa
- swissprot.fa.phr
- swissprot.fa.pin
- swissprot.fa.psq

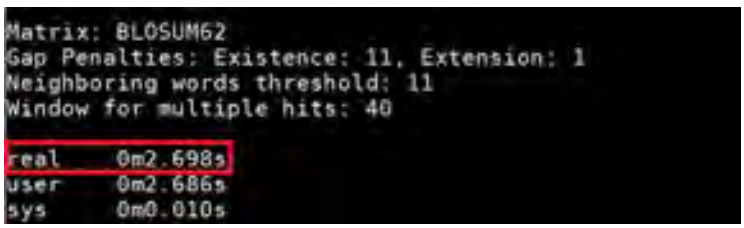
Step 6 Run the following command to run the test:

```
{ time blastp -query data/protein_query.fasta -db data/swissprot.fa ; } 2>&1 |tee
output_protein_alignments.txt
```

View the **real** value in the last **time** command output in the log. The value indicates *x* minutes *y* seconds. The lower the value is, the better the performance is.

Figure 4-1 shows the command output.

Figure 4-1 Test example



----End

4.1.8 More Information

Official Arm website:

 Copyright © Huawei Technologies Co., Ltd.

https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/resources/porting-and-tuning/building-blast-with-arm-compiler/single-page

4.2 BWA 0.7.17 Porting Guide (CentOS 7.6)

4.2.1 Introduction

Burrows-Wheeler Aligner (BWA) is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW, and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100 bp. The rest two are for longer sequences ranged from 70 bp to 1 Mbp. BWA-MEM and BWA-SW share similar features, such as long-read support and split alignment. However, BWA-MEM is generally recommended for high-quality queries as it is faster and more accurate. Compared with BWA-backtrack, BWA-MEM has better performance in 70–100 bp Illumina reads.

For more information, visit the **official BWA website**.

Programming language: C

Brief description: gene comparison

Open source license: GPL 3.0

Recommended Software Version

BWA 0.7.17

4.2.2 Environment Requirements

Hardware Requirements

Table 4-6 lists the hardware requirements.

Table 4-6 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-7 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-7 Software requirements

Item	Version	Download Address
BWA	0.7.17	**https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2**
Test file 1	ERR1044518_1.fastq	**https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_1.fastq.gz**
Test file 2	ERR1044518_2.fastq	**https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_2.fastq.gz**
Test file 3	hs38DH.fasta	**https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz** (Download the package, decompress it using **gzip -d**, and rename it **hs38DH.fasta**.)

OS Requirements

Table 4-8 lists the OS requirements.

Table 4-8 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.2.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BWA software porting.

Table 4-9 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/BWA	Installation path of BWA	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing the BWA test files	

4.2.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-10 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

4.2.5 Obtaining the Source Code

Procedure

Step 1 Download the BWA installation package **bwa-0.7.17.tar.bz2**.

https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2

Step 2 Use an SFTP tool to upload the BWA installation package to the */path/to/BWA* directory on the server.

----End

4.2.6 Compiling and Installing BWA

Prerequisites

You have obtained **SSE2NEON.h** from **https://github.com/jratcliff63367/sse2neon**.

 Copyright © Huawei Technologies Co., Ltd.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Go to the BWA installation package directory.

```
cd /path/to/BWA
```

Step 3 Decompress the BWA installation package.

```
tar -xvf bwa-0.7.17.tar.bz2
```

Step 4 Go to the directory generated after the decompression.

```
cd bwa-0.7.17
```

Step 5 Copy **sse2neon.h** to the compiler installation directory.

```
cp SSE2NEON.h /path/to/GNU/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include
```

Step 6 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify line 14 of the file.
   ```
   INCLUDES= -I/path/to/GNU/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include
   ```

   NOTE

   Change the **include** directory to the **include** directory of the GNU compiler.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the **ksw.c** file.

1. Open **ksw.c**.
   ```
   vi ksw.c
   ```
2. Press **i** to enter the insert mode and modify line 29 of the file.
   ```
   include <SSE2NEON.h>
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation.

```
make
```

Step 9 Load the BWA environment.

```
export PATH=/path/to/BWA/bwa-0.7.17:$PATH
```

----End

4.2.7 Running and Verifying BWA

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Create a working directory for BWA.

```
mkdir -p /path/to/CASE
```

Step 3 Go to the test file directory.

```
cd /path/to/CASE
```

Step 4 Obtain the test cases.

```
wget https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_1.fastq.gz
wget https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_2.fastq.gz
```

 Copyright © Huawei Technologies Co., Ltd.

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz -O hs38DH.fasta
```

Step 5 Decompress the test files.

```
gzip -d ERR1044518_1.fastq.gz
gzip -d ERR1044518_2.fastq.gz
```

Step 6 Create an index file.

```
bwa index -a bwtsw hs38DH.fasta
```

Step 7 Run the test.

```
bwa mem -t 128 hs38DH.fasta ERR1044518_1.fastq ERR1044518_2.fastq > bwa.sam 2 > bwa_log
```

NOTE

The number **128** indicates the number of used cores. You can change the number according to the server configuration.

Check the value of **Real time** (unit: s) in the **bwa_log** log. A smaller value indicates higher performance.

```
M::mem_process_seqs] Processed 3874074 reads in 2658.093 CPU sec, 22.437 real sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa mem -M -t 128 hs38DH.fasta ERR1044518_1.fastq ERR1044518_2.fastq
[main] Real time: 955.857 sec; CPU: 107304.320 sec
```

----End

4.3 CNVnator 0.4.1 Porting Guide (CentOS 7.6)

4.3.1 Introduction

CNVnator is a tool used for copy number variation (CNV) discovery and genotyping from read-depth (RD) analysis of personal genome sequencing in the field of gene sequencing. CNVnator is based on combining the established mean-shift approach with additional refinements (multiple-bandwidth partitioning and GC correction) to broaden the range of discovered CNVs. Furthermore, CNVnator is complementary in a straightforward way to split-read and read-pair approaches: It misses CNVs created by retrotransposable elements, but more than half of the validated CNVs that it identifies are not detected by split-read or read-pair.

For more information about CNVnator, visit the **official CNVnator website**.

Programming language: C++

Brief description: a tool used for copy number variation (CNV) discovery and genotyping from read-depth (RD) analysis of personal genome sequencing

Open source license: MIT

Recommended Software Version

CNVnator v0.4.1

 Copyright © Huawei Technologies Co., Ltd.

4.3.2 Environment Requirements

Hardware Requirements

Table 4-11 lists the hardware requirements.

Table 4-11 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-12 lists the software requirements.

Table 4-12 Software requirements

Item	Version	Download Address
CNVnator	0.4.1	https://github.com/abyzovlab/CNVnator
CERN ROOT	6.18	https://root.cern/download/root_v6.18.00.source.tar.gz
CMake	3.9.2	https://cmake.org/files/v3.3/cmake-3.3.2.tar.gz
Samtools	1.10	http://www.htslib.org/download/
Test case file	-	-

OS Requirements

Table 4-13 lists the OS requirements.

Table 4-13 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.3.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CNVnator software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-14 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/SAMTOOLS	Installation path of Samtools	
4	/path/to/SOURCE	Installation path of the ROOT tool package	
5	/path/to/CNVNATOR	Installation path of CNVnator	
6	/path/to/CASE	Path for storing the CNVnator test cases	

4.3.4 Configuring the Compilation Environment

4.3.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-15 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing the pre-installation packages and dependencies	See **4.3.4.2 Installing the Pre-installation Packages and Dependencies**.
3	Installing CMake	See **4.3.4.3 Installing CMake**.
4	Installing Samtools	See **4.3.4.4 Installing Samtools**.
5	Installing the ROOT tool package	See **4.3.4.5 Installing the ROOT Tool Package**.

4.3.4.2 Installing the Pre-installation Packages and Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install the pre-installation packages and dependencies:

```
yum install -y libcurl-devel.aarch64 cmake binutils \
libX11-devel libXpm-devel libXft-devel libXext-devel \
openssl-devel pcre-devel mesa-libGL-devel \
mesa-libGLU-devel glew-devel ftgl-devel mysql-devel \
fftw-devel cfitsio-devel graphviz-devel \
avahi-compat-libdns_sd-devel libldap-dev python-devel \
libxml2-devel gsl-static
```

----End

4.3.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the CMake installation package:

```
tar -xvf cmake-3.15.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd cmake-3.15.2
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 5 Run the following commands to perform compilation and installation:

 Copyright © Huawei Technologies Co., Ltd.

```
make
make install
```

Step 6 Run the following command to set the CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.3.4.4 Installing Samtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Samtools installation package:

```
tar -xvf samtools-1.10.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd samtools-1.10
```

Step 4 Run the following command to perform installation:

```
yum install -y bzip2-devel bzip2-libs bzip2
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/SAMTOOLS
```

Step 6 Run the following commands to perform compilation and installation:

```
make all all-htslib
make install install-htslib
```

Step 7 Run the following commands to set the Samtools environment variables:

```
export PATH=/path/to/SAMTOOLS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SAMTOOLS/lib:$LD_LIBRARY_PATH
```

----End

4.3.4.5 Installing the ROOT Tool Package

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the ROOT tool package to the **/path/to/SOURCE** directory.

Step 3 Run the following commands to decompress the installation package **root_v6.18.00.source.tar.gz**:

```
cd /path/to/SOURCE
tar -xvf root_v6.18.00.source.tar.gz
```

Step 4 Run the following commands to create a **root_build** directory in the */path/to/SOURCE* directory. ROOT cannot be compiled in the source code.

```
mkdir root_build
cd root_build
cmake ../root-6.18.00
```

Step 5 If the environment is not connected to the Internet, you need to manually download the required installation packages to the corresponding directory. The following installation packages are contained in the ROOT tool package.

 Copyright © Huawei Technologies Co., Ltd.

- Save the **davix-embedded-0.6.7.tar.gz** installation package to the **root_build/builtins/davix/DAVIX-prefix/src** directory.
- Save the **vdt-0.4.2.tar.gz** installation package to the **root_build/VDT-prefix/src** directory.
- Save the **tbb2019_U1.tar.gz** installation package to the **root_build/TBBefix/src** directory.

Step 6 Run the following command to modify the **CMakeCache.txt** file in **<root_build>**.

1. Open **CMakeCache.txt**.
   ```
   vi CMakeCache.txt
   ```
2. Press **i** to enter the insert mode and set clad installation to **OFF** in the **CMakeCache.txt** file.
   ```
   clad BOLL=OFF
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to perform compilation:

```
make -j32
```

Step 8 Run the following commands to load environment variables:

```
export PATH=/path/to/SOURCE/root_build/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SOURCE/root_build/lib:$LD_LIBRARY_PATH
```

----End

4.3.5 Obtaining the Source Code

Procedure

Step 1 Download the CNVnator installation package **CNVnator_v0.4.1.zip**.

https://github.com/abyzovlab/CNVnator

Step 2 Use an SFTP tool to upload the CNVnator installation package to the */path/to/CNVNATOR* directory on the server.

----End

4.3.6 Compiling and Installing CNVnator

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to go to the CNVnator installation directory and create the dependency link:

```
cd /path/to/CNVNATOR
unzip CNVnator_v0.4.1.zip
cd CNVnator_v0.4.1/src
ln -s /path/to/SAMTOOLS/samtools-1.10 samtools
ln -s /path/to/SOURCE/root_build root_build
```

Step 3 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and add the following content to the beginning of line 4 in the **Makefile** file:
   ```
   ROOTSYS=root_build
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to perform compilation:

```
make LIBS="-lcrypto"
```

Step 5 Run the following commands to load environment variables:

```
export PATH=/path/to/CNVNATOR/CNVnator_v0.4.1/src:$PATH
```

----End

4.3.7 Running and Verifying CNVnator

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a working directory and go to it:

```
mkdir -p /path/to/CASE
cd /path/to/CASE
```

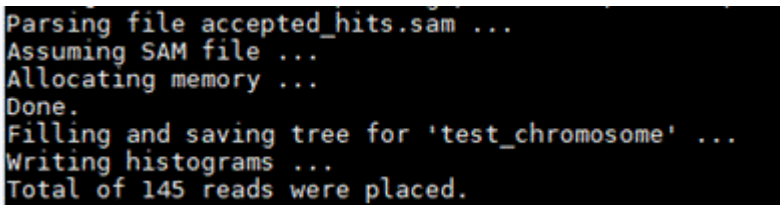
Step 3 Download the test case file and use an SFTP tool to upload it to the */path/to/CASE* directory.

Step 4 Run the following commands to perform the test:

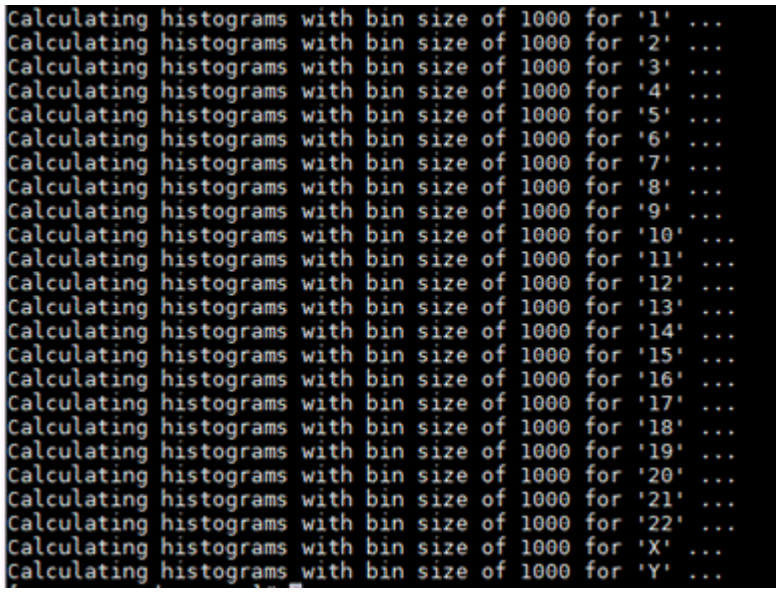
```
{ time cnvnator -root B17NC.root -tree B17NC.bam ; } 2>&1 |tee -a Mapping.log
{ time cnvnator -root B17NC.root -his 1000 -fasta chr14.GRCm38.primary_assembly.genome.fa.gz ; } 2>&1 |tee -a Histogram.log
{ time cnvnator -root B17NC.root -stat 1000 ; } 2>&1 |tee -a Statistics.log
```

If the command output shown in **Figure 4-2** is displayed, the calculation ends properly.

Figure 4-2 Test example



 Copyright © Huawei Technologies Co., Ltd.



----End

4.3.8 Troubleshooting

Problem 1: "std::unique_ptr" Is Reported During CMake Compilation

Symptom

"std::unique_ptr" is reported during CMake compilation, and the error message is "CMake Error at CMakeLists.txt:92 (message):

The C++ compiler does not support C++11 (e.g. std::unique_ptr)."

Possible Cause

This error is related to the system time setting.

Procedure

- After the system time is set correctly, decompress the source code package again for compilation.
- Run the following command to modify the **CmakeCache.txt** file:
  a. Open **CmakeCache.txt**.
  ```
  vi CmakeCache.txt
  ```
  b. Press **i** to enter the insert mode and modify line 362 of the file.
  ```
  CMake_HAVE_CXX_UNIQUE_PTR:INTERNAL=ON
  ```
  c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
  d. Perform compilation and installation.
  ```
  make
  ```

Problem 2: "CMake Error at TBB-stamp" Is Reported During ROOT Compilation

Symptom

 Copyright © Huawei Technologies Co., Ltd.

"CMake Error at TBB-stamp" is reported during ROOT compilation. The error information is similar to the following:

CMake Error at TBB-stamp/TBB-download-RelWithDebInfo.cmake:49 (message):

Command failed: 1

'/storage/softwares/TaiShan/CNVnator/CMAKE/bin/cmake' '-Dmake=' '-Dconfig=' '-P' '/storage/softwares/TaiShan/CNVnator/ROOT-6.18/TBB-prefix/src/TBB-stamp/TBB-download-RelWithDebInfo-impl.cmake'

Possible Cause

Such a failure to find the file is caused by the cluster's failure to connect to the network.

Procedure

Save the downloaded installation packages to the folder generated after CMake is installed. For details, see **4.3.4.3 Installing CMake**.

Problem 3: "[cnvnator] Error 1" Is Reported During CNVnator Compilation

Symptom

"[cnvnator] Error 1" is reported during CNVnator compilation. The error information is similar to the following:

/usr/bin/ld: samtools/htslib-1.9/libhts.a(hfile_s3.o): undefined reference to symbol 'HMAC@@libcrypto.so.10'

//usr/lib64/libcrypto.so.10: error adding symbols: DSO missing from command line

collect2: error: ld returned 1 exit status

make: *** [cnvnator] Error 1

Possible Cause

An error occurs during compilation and linking.

Procedure

Run the following **make** command:

```
make LIBS="-lcrypto"
```

4.3.9 More Information

CNVnator page on GitHub:

https://github.com/abyzovlab/CNVnator/blob/master/README.md

ROOT installation guide:

https://root.cern.ch/building-root#quick-start

4.4 Cufflinks 2.2.1 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.4.1 Introduction

Cufflinks is a program in the gene sequencing field that is mainly used to calculate the gene expression volume and search for differentially expressed genes. Based on the comparison result of TopHat, the program calculates the FPKM value of isoform (of each gene) based on or not based on the GTF comment file of the reference genome, and provides the comment result in **trascripts.gtf** (assemble transcription groups). Cufflinks contains several main executable programs, such as Cufflinks, Cuffmerge, Cuffcompare, and Cuffdiff.

For more information, visit the **official Cufflinks website**.

Programming language: C++

Brief description: a program used to calculate the gene expression volume and search for differentially expressed genes.

Open source license: Boost Software License

Recommended Software Version

Cufflinks 2.2.1

4.4.2 Environment Requirements

Hardware Requirements

Table 4-16 lists the hardware requirements.

Table 4-16 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-17 lists the software requirements.

Table 4-17 Software requirements

Item	Version	How to Obtain
Cufflinks	2.2.1	**https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master**
Boost	1_55_0	**http://downloads.sourceforge.net/project/boost/boost/1.55.0/boost_1_55_0.tar.bz2**
Eigen	3.3.7	**https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Samtools	0.1.19	https://codeload.github.com/samtools/samtools/tar.gz/0.1.19
htslib	1.9	https://github.com/samtools/htslib/releases/download/1.9/htslib-1.9.tar.bz2

OS Requirements

Table 4-18 lists the OS requirements.

Table 4-18 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Cufflinks software porting.

Table 4-19 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BOOST	Installation path of Boost	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/SAMTOOL	Installation path of SAMtools	
4	/path/to/HTSLIB	Installation path of HTSlib	
5	/path/to/EIGEN	Installation path of Eigen	
6	/path/to/CUFFLINKS	Installation path of Cufflinks	

 Copyright © Huawei Technologies Co., Ltd.

4.4.4 Configuring the Compilation Environment

4.4.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-20 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See the following sections of "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**: • Configuring the Local Yum Source • Installing GMP • Installing MPFR • Installing MPC • Installing the InfiniBand NIC Driver • Installing Open MPI
2	Installing Boost	See **4.4.4.2 Installing Boost**.
3	Installing Samtools	See **4.4.4.3 Installing Samtools**.
4	Installing HTSlib	See **4.4.4.4 Installing HTSlib**.
5	Installing Eigen	See **4.4.4.5 Installing Eigen**.

4.4.4.2 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

 Copyright © Huawei Technologies Co., Ltd.

NOTE

Cufflinks 2.2.1 requires that the Boost version be in the range of 1.47 to 1.55.

```
yum install -y xz-devel.aarch64 ncurses-devel.aarch64 bzip2-devel.aarch64 libicu.aarch64 libicu-devel.aarch64
```

Step 3 Run the following commands to add the GNU environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 4 Run the following command to decompress the installation package:

```
tar -xvf boost_1_55_0.tar.bz2
```

Step 5 Run the following command to switch to the directory generated after the package is decompressed:

```
cd boost_1_55_0
```

Step 6 Run the following command to run the **bootstrap.sh** script:

```
./bootstrap.sh
```

Step 7 Run the following command to compile and install Boost:

```
./b2 --prefix=/path/to/BOOST install
```

Step 8 Run the following command to check whether library files are generated in the **lib** directory of Boost:

```
ls /path/to/BOOST/lib
```

Step 9 Run the following command to add Boost environment variables:

```
export LD_LIBRARY_PATH=/path/to/BOOST/lib:$LD_LIBRARY_PATH
```

----End

4.4.4.3 Installing Samtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the SAMtools installation directory:

```
cd /path/to/SAMTOOL
```

Step 3 Run the following command to decompress the SAMtools installation package:

NOTE

Install SAMtools and HTSlib separately. Do not use SAMtools that contains HTSlib. Otherwise, errors may occur during compilation.

```
tar -xvf samtools-0.1.19.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd samtools-0.1.19
```

Step 5 Run the following command to obtain the GNU absolute directory:

```
which gcc
```

Step 6 Modify the **Makefile** file.

 Copyright © Huawei Technologies Co., Ltd.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   CC    = gcc
   ```

   After the modification:

   ```
   CC    = /path/to/GNU/bin/gcc
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following commands to perform compilation and installation:

```
make
```

Step 8 Run the following commands to copy the generated binary files to the default path of the root directory so that Cufflinks can directly identify the SAMtools and HTSlib files:

```
mkdir /usr/local/include/bam
cp /path/to/SAMTOOL/samtools-0.1.19/libbam.a /usr/local/lib
cp /path/to/SAMTOOL/samtools-0.1.19/*.h /usr/local/include/bam/
cp /path/to/SAMTOOL/samtools-0.1.19/samtools /usr/bin/
```

----End

4.4.4.4 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the HTSlib installation directory:

```
cd /path/to/HTSLIB
```

Step 3 Run the following command to decompress the HTSlib installation package:

```
tar -xvf htslib-1.9.tar.bz2
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd htslib-1.9
```

Step 5 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   CC    = gcc
   ```

   After the modification:

   ```
   CC    = /path/to/GNU/bin/gcc
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to perform compilation and installation:

```
make
```

Step 7 Run the following command to copy the generated file to the default path of the root directory so that Cufflinks can directly identify the HTSlib files:

 Copyright © Huawei Technologies Co., Ltd.

```
cp /path/to/HTSLIB/htslib-1.9/libhts.* /usr/local/lib
```

----End

4.4.4.5 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the Eigen installation directory:

```
cd /path/to/EIGEN
```

Step 3 Run the following command to decompress the Eigen installation package:

```
tar -xvf eigen-3.3.7.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd eigen-3.3.7
```

Step 5 Run the following command to create an **include** directory:

```
mkdir include
```

Step 6 Run the following command to copy the **Eigne** directory to the **include** directory:

```
cp Eigen include -rf
```

Step 7 Run the following command to obtain the Eigen absolute directory:

```
pwd
```

----End

4.4.5 Obtaining the Source Code

Procedure

Step 1 Download the Cufflinks installation package **cufflinks-master.zip**.

https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master

Step 2 Use an SFTP tool to upload the Cufflinks installation package to the */path/to/CUFFLINKS* directory on the server.

----End

4.4.6 Compiling and Installing Cufflinks

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the Cufflinks installation directory:

```
cd /path/to/CUFFLINKS
```

Step 3 Run the following command to decompress the Cufflinks installation package:

```
unzip cufflinks-master.zip
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd cufflinks-master
```

Step 5 Run the following command to generate a **configure** file:

```
./autogen.sh --install
```

Step 6 Run the following command perform configuration:

```
./configure --with-boost=/path/to/BOOST --with-eigen=/path/to/EIGEN/eigen-3.3.7 --prefix=/path/to/CUFFLINKS/build
```

Step 7 Run the following commands to perform compilation and installation:

```
make -j 16 install
```

Step 8 Run the following commands to add the Cufflinks environment variables:

```
export PATH=/path/to/CUFFLINKS/build/bin:$PATH
```

----End

4.4.7 Running and Verifying Cufflinks

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the test file path of the source code and execute the file:

```
cd /path/to/CUFFLINKS/cufflinks-master/external_tests/tiny_tests/2x75
```

Step 3 Run the following command to convert the test case format:

```
samtools view -bS accepted_hits.sam > accepted_hits.bam
```

Step 4 Run the following command to run the test case:

```
cufflinks accepted_hits.bam
```

If the following information is displayed, the Cufflinks process ends properly:

```
Warning: Using default Gaussian distribution due to insufficient paired-end reads in open ranges.  It is recommended that correct parameters (--frag-len-mean and --frag-dev) be provided.
> Map Properties:
>       Normalized Map Mass: 90.00
>       Raw Map Mass: 90.00
>       Fragment Length Distribution: Truncated Gaussian (default)
>                     Default Mean: 200
>                  Default Std Dev: 80
[19:58:21] Assembling transcripts and estimating abundances.
> Processed 1 loci.                       [*************************] 100%
```

----End

4.4.8 Troubleshooting

Problem 1: Error Message "undefined reference to lzma" Is Reported During Cufflinks Compilation

Symptom

An error message "undefined reference to'lzma_stream_buffer_bound'" is reported when Cufflinks is compiled.

 Copyright © Huawei Technologies Co., Ltd.

Possible Causes

The LZMA dependency package is not installed.

Procedure

Run the following command to install **xz-devel.aarch64**:

```
yum install xz-devel.aarch64
```

Problem 2: Error Message "could not detect bam" Is Displayed During Cufflinks Compilation

Symptom

An error message "error:we cand not detect bam"is reported when Cufflinks is compiled.

Possible Causes

- The **configure** file is incorrectly configured.
- **<bam/version.hpp>** is used to determine the BAM version in the **configure** file.

Procedure

- If the **configure** file is incorrectly configured:

  Run the following commands to modify the **configure** file:

  a. Open **configure**.
  ```
  vi configure
  ```
  b. Press **i** to enter the insert mode and change **no** to **yes** in line 5380 of the file.
  ```
  succeeded=yes
  ```
  c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
- If **<bam/version.hpp>** is used to determine the BAM version in the **configure** file (actually **version.h** is the version file of the BAM):

  Run the following commands to modify the **configure** file:

  a. Open **configure**.
  ```
  vi configure
  ```
  b. Press **i** to enter the insert mode and change **bam/version.hpp** to **bam/version.h**.
  ```
  bam/version.h
  ```
  c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Problem 3: Error Message "could not find htslib" Is Displayed During Cufflinks Compilation

Symptom

An error message "could not find htslib"is reported when Cufflinks is compiled.

Possible Causes

The built-in HTSlib of SAMtools is used, which is not supported in the current version.

 Copyright © Huawei Technologies Co., Ltd.

Procedure

Install HTSlib independently and copy the generated **libhts.*** file to the **lib** directory.

Run the following commands to install HTSlib:

```
tar -xvf htslib-1.9.tar.bz2
cd htslib-1.9
make
cp libhts.* /usr/local/lib
```

Problem 4: Error Message "GHash.hh:91:44: error" Is Displayed During Cufflinks Compilation

Symptom

An error is reported when Cufflinks is compiled. The error information is as follows:

```
GHash.hh:91:44: error: type/value mismatch at argument 1 in template parameter list for 'template<class
_Tp> struct std::hash'
91 |     while (pos<fCapacity && hash[pos].hash<0) pos++;
|                                          ^
```

Possible Causes

The Cufflinks source code has a bug, which has been fixed in the latest master release.

Procedure

Download the latest Cufflinks installation package.

https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master

Problem 5: Error Message "error while loading shared libraries" Is Displayed During the Test

Symptom

An error message similar to "error while loading shared libraries: libhts.so.2: cannot open shared object file: No such file or directory" is displayed during the test.

Possible Causes

The HTSlib environment variables have not been added.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to specify the HTSlib address:

```
export LD_LIBRARY_PATH=/path/to/HTSLIB/htslib-1.9:$LD_LIBRARY_PATH
```

----End

4.4.9 More Information

Official user guide:

 Copyright © Huawei Technologies Co., Ltd.

http://cole-trapnell-lab.github.io/cufflinks/getting_started/

4.5 GATK 4.0.0.0 Porting Guide (CentOS 7.6)

4.5.1 Introduction

Genome Analysis Toolkit (GATK), as the name implies, is a toolkit for analyzing genomes. Its main feature is to find variants and genotypes for variant calling from sequencing data, including SNP and INDEL.

GATK includes the following tools:

- Diagnostics and quality control tools
- Sequence data processing tools
- Variant discovery tools
- Variant evaluation tools
- Variant manipulation tools
- Commenting modules
- Reads filtering
- Resource file decoding tools
- Reference sequence utilities

GATK4 is the latest GATK version released by Broad Institute in January 2018. Compared with the previous version, this version has major changes, including newly developed processes and the latest tools based on machine learning algorithms. GATK4 greatly optimizes the performance, computing speed, interface flexibility, and scalability. The end-to-end process can run locally or on the cloud. In addition, it is developed based on Spark and supports cluster deployment.

For more information, visit the **official GATK website**.

Programming language: Java

Brief description: A toolkit for analyzing genomes.

Open source license: BSD 3-Clause "New" or "Revised" License

Recommended Software Version

GATK 4.0.0.0

4.5.2 Environment Requirements

Hardware Requirements

Table 4-21 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-21 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-22 lists the software requirements.

Table 4-22 Software requirements

Item	Version	Download Address
GATK	4.0.0.0	https://github.com/broadinstitute/gatk/releases/download/4.0.0.0/gatk-4.0.0.0.zip
Samtools	0.1.9	https://master.dl.sourceforge.net/project/samtools/samtools/0.1.9/samtools-0.1.9.tar.bz2
BLAT	35.1	https://codeload.github.com/djhshih/blat/tar.gz/v35.1
BWA	0.7.17	https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2/download
Hadoop	3.1.2	https://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2.tar.gz
Spark	2.4.4	https://archive.apache.org/dist/spark/
Test case 1	SRR742200_1.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_1.fastq.gz
Test case 2	SRR742200_2.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_2.fastq.gz
Test case 3	human_g1k_v37.fasta	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
Test case 4	dbsnp132_20101103.vcf	ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/dbsnp132_20101103.vcf.gz

OS Requirements

Table 4-23 lists the OS requirements.

Table 4-23 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

4.5.3 Paths for Software Porting

This chapter lists software installation paths involved in the GATK software porting.

Table 4-24 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BLAT	Installation path of BLAT	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/BMA	Installation path of BMA	
4	/path/to/SAMTOOLS	Installation path of Samtools	
5	/path/to/GATK	Installation path of GATK	

4.5.4 Configuring the Compilation Environment

4.5.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-25 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing dependencies	See **4.5.4.2 Installing Dependencies**.
3	Installing BLAT	See **4.5.4.3 Installing BLAT**.
4	Installing BWA	See **4.5.4.4 Installing BWA**.
5	Installing Samtools	See **4.5.4.5 Installing Samtools**.

4.5.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependencies:

```
yum install -y java-1.8.0-openjdk-devel.aarch64 ncurses-devel.aarch64 bzip2-devel.aarch64
```

----End

4.5.4.3 Installing BLAT

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependencies:

```
yum install libpng-devel -y
```

Step 3 Run the following command to decompress the BLAT installation package:

```
tar zxvf blat-35.1.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd blat-35.1/
```

Step 5 Run the following command to declare the installation environment:

```
export MACHTYPE=aarch64
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following command to perform the compilation and installation:

```
make
```

Step 7 Run the following command to declare the BLAT environment variables:

```
export PATH=/path/to/BLAT/blat-35.1/bin:$PATH
```

----End

4.5.4.4 Installing BWA

Prerequisites

You have downloaded the **sse2neon.h** file from **https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h**.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the BWA installation package:

```
tar -xvf bwa-0.7.17.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd bwa-0.7.17
```

Step 4 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify lines 1 and 14 of the file.
   ```
   CC=             /path/to/GNU/bin/gcc
   INCLUDES=       -I/path/to/GNU/gcc/include
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the **ksw.c** file.

1. Open **ksw.c**.
   ```
   vi ksw.c
   ```
2. Press **i** to enter the insert mode and modify lines 29 and 30 of the file.
   ```
   /*#include <emmintrin.h>*/
    #include <sse2neon.h>
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to copy the Neon instruction conversion code to a specific directory of the compiler:

```
cp sse2neon.h /path/to/GNU/gcc/include
```

Step 7 Run the following command to perform the compilation:

```
make
```

Step 8 Run the following command to set the BWA environment variables:

```
export PATH=/path/to/BMA/bwa-0.7.17:$PATH
```

Step 9 Run the following command to check whether the installation is successful:

```
bwa
```

If information similar to the following is displayed, the installation is successful:

 Copyright © Huawei Technologies Co., Ltd.

```
bwa <command> [options]
```

----End

4.5.4.5 Installing Samtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Samtools installation package:

```
tar -xvf samtools-0.1.9.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd samtools-0.1.9
```

Step 4 Run the following command to install the dependencies:

```
yum install xz-devel-5.2.2-1.el7.aarch64 bzip2-devel-1.0.6-13.el7.aarch64 ncurses-devel.aarch64 -y
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/SAMTOOLS
```

Step 6 Run the following commands to perform the compilation:

```
make
make install
```

Step 7 Run the following command to set the Samtools environment variables:

```
export PATH=/path/to/SAMTOOLS/bin:$PATH
```

Step 8 Run the following command to check whether the installation is successful:

```
samtools
```

If information similar to the following is displayed, the installation is successful:

```
Program: samtools (Tools for alignments in the SAM format)
Version: 0.1.9 (r183)

Usage:   samtools <command> [options]
```

----End

4.5.5 Obtaining the Source Code

Procedure

Step 1 Download the GATK installation package **gatk-4.0.0.0.zip**.

https://github.com/broadinstitute/gatk/releases/download/4.0.0.0/gatk-4.0.0.0.zip

Step 2 Use an SFTP tool to upload the GATK installation package to the */path/to/GATK* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.5.6 Compiling and Installing GATK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install GATK4:

```
unzip gatk-4.0.0.0.zip
export PATH=/path/to/GATK/gatk-4.0.0.0:$PATH
gatk -help
```

Step 3 Install and deploy the Hadoop and Spark services and then start them. For details, see "Compiling and Installing Hadoop and Spark" in **Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)**.

----End

4.5.7 Running and Verifying GATK

4.5.7.1 Running GATK4 in Spark Mode

Prerequisites

Prepare data files. Two types of data files are required for the GATK test: genome data files (in FASTA format) and original sequencing data files (FASTQ or BAM format).

This test case uses the following data files:

- Genome data files (in FASTA format)

  The following is the human genome information.

  human_g1k_v37.fasta

  The following two VCF files are from the 1000 Genomes Project and the Mills Project, which record the population InDel areas detected in those projects.

  - Mills_and_1000G_gold_standard.indels.hg19.sites.vcf.gz
  - 1000G_phase1.indels.hg19.sites.vcf.gz

  The following document is a collection of almost all public population variants currently available.

  dbsnp132_20101103.vcf

  The variation dataset selects one to three types of data based on the sequencing purpose.

- Original sequencing data (FASTQ or BAM format)
  - SRR742200_1.fastq
  - SRR742200_2.fastq

  a. Use PuTTY to log in to the server as the root user.

  b. Run the following commands to create an **input** folder:

  ```
  mkdir projectDir
  cd projectDir
  mkdir input
  ```

 Copyright © Huawei Technologies Co., Ltd.

c. Run the following commands to decompress the case files:

```
gzip -d SRR742200_1.fastq.gz
gzip -d SRR742200_1.fastq.gz
gzip -d human_g1k_v37.fasta.gz
gzip -d dbsnp132_20101103.vcf.gz
```

d. Run the following command to generate a 2-bit file using BLAT:

```
faToTwoBit human_g1k_v37.fasta human_g1k_v37.fasta.2bit
```

e. Run the following command to save all data files to the **input** folder:

```
cp human_g1k_v37.fasta human_g1k_v37.fasta.2bit SRR742200_1.fastq SRR742200_2.fastq
dbsnp132_20101103.vcf input
```

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create an index for the **fasta** file:

```
bwa index -a bwtsw human_g1k_v37.fasta
```

The command output is as follows:

```
[bwa_index] Pack FASTA... 27.89 sec
[bwa_index] Construct BWT for the packed sequence...
[BWTIncCreate] textLength=6203609478, availableWord=448508744
[BWTIncConstructFromPacked] 10 iterations done. 99999990 characters processed.
[BWTIncConstructFromPacked] 20 iterations done. 199999990 characters processed.
[BWTIncConstructFromPacked] 30 iterations done. 299999990 characters processed.
[BWTIncConstructFromPacked] 40 iterations done. 399999990 characters processed.
[BWTIncConstructFromPacked] 50 iterations done. 499999990 characters processed.
[BWTIncConstructFromPacked] 60 iterations done. 599999990 characters processed.
[BWTIncConstructFromPacked] 70 iterations done. 699999990 characters processed.
[BWTIncConstructFromPacked] 80 iterations done. 799999990 characters processed.
[BWTIncConstructFromPacked] 90 iterations done. 899999990 characters processed.
<The following output is omitted.>
```

Step 3 Run the following command to view the generated index files:

```
ls
```

NOTE

After **Step 2** and **Step 3** are complete, you will see 5 files in the current path: **human_g1k_v37.fasta.amb**, **human_g1k_v37.fasta.ann**, **human_g1k_v37.fasta.bwt**, **human_g1k_v37.fasta.pac**, and **human_g1k_v37.fasta.sa**. The generated index file is applicable to all pipelines and needs to be built only once. It takes about 1.5 hours to build the index file. The generated index file will be used in subsequent sequencing steps.

Step 4 Run the following commands to compare the **fastq** file:

1. Generate a dictionary.

```
gatk CreateSequenceDictionary -R human_g1k_v37.fasta -O human_g1k_v37.dict
```

If the information in the red box shown in the following figure is displayed, the execution is complete.

The generated **dict** file is used for genome comparison.

2. Perform the comparison.

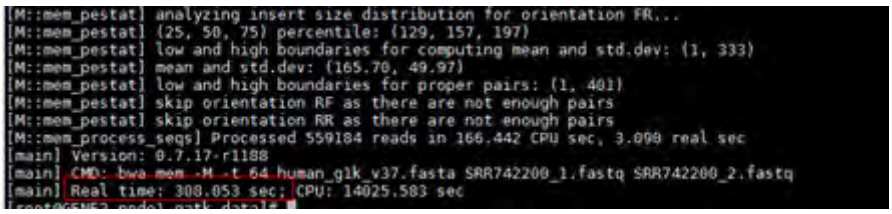
```
bwa mem -M -t 96 human_g1k_v37.fasta SRR742200_1.fastq SRR742200_2.fastq > SRR7.sam
```

 Copyright © Huawei Technologies Co., Ltd.

Table 4-26 Parameters

Parameter	Description
-t	The parameter is the number of threads, which is generally the same as the number of server cores.

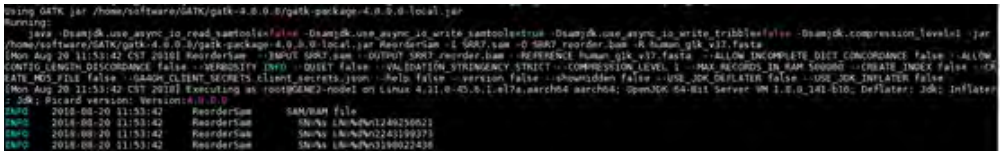
If the information in the red box shown in the following figure is displayed, the execution is complete.



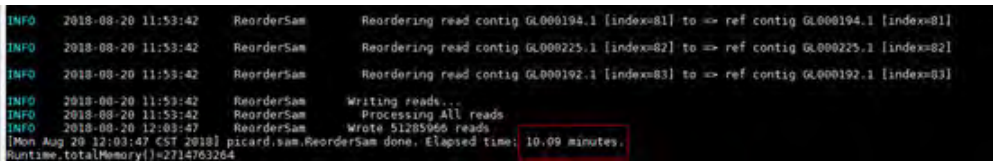
Step 5 Run the following command to re-sort the gene sequence:

By default, the **sam** files generated by BWA are sorted by dictionary mode. You need to convert the files for sorting by genome type, which is completed by the Picard tool integrated in GATK4.

```
gatk ReorderSam -I SRR7.sam -O SRR7_reorder.bam -R human_g1k_v37.fasta
```



If the information in the red box shown in the following figure is displayed, the execution is complete.



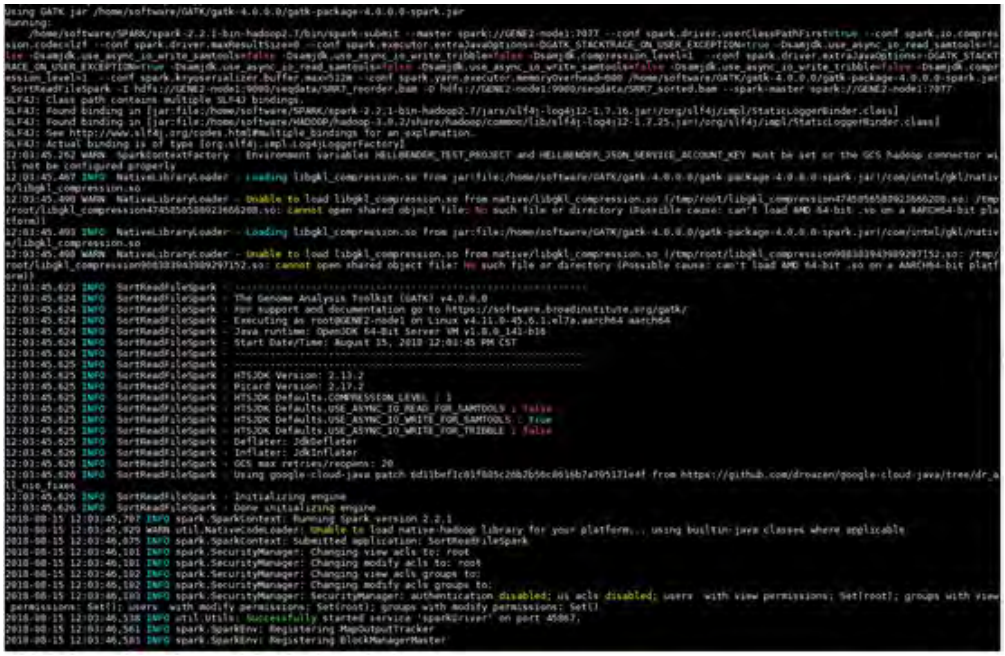
Step 6 Sort gene sequences by coordinate in descending order.

1. Spark runs in the HDFS file system and is invisible in the Linux system. Run the following command to transfer the **SRR7_reorder.bam** file to the HDFS file system:
   ```
   hadoop fs -mkdir /seqdata
   hadoop fs -put SRR7_reorder.bam /seqdata
   ```
2. Run the following commands to check whether the **sparklog** directory has been manually created:
   ```
   hadoop fs –mkdir /sparklog
   hadoop fs –ls /
   ```
3. Run the following commands to perform the sorting:
   ```
   gatk SortReadFileSpark -I hdfs://Hostname:9000/seqdata/SRR7_reorder.bam -O hdfs://Hostname:9000/seqdata/SRR7_sorted.bam -- --spark-runner SPARK --spark-master spark://Hostname:7077
   ```

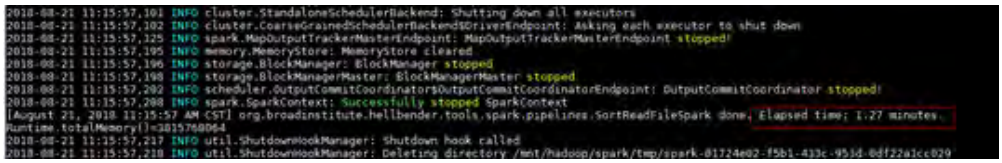
 Copyright © Huawei Technologies Co., Ltd.

NOTE

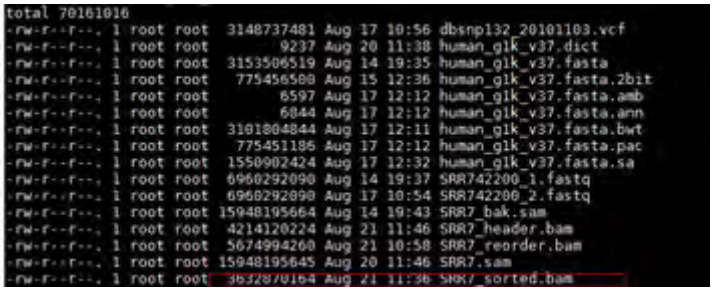
Change the host names based on the master nodes of Hadoop and Spark. For details, see **Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)**.



If the information in the red box shown in the following figure is displayed, the execution is complete.



4. Run the following command to copy the generated file to the data directory:
   ```
   hadoop fs -get /seqdata/SRR7_sorted.bam ./
   ```
5. Run the following command to view the data directory:
   ```
   ll
   ```
   If the information in the red box shown in the following figure is displayed, the execution is complete.



Step 7 Run the following command to add a header for the gene sequence:

GATK 2.0 and later versions do not support headless file detection. This step can be performed by using **-r** during BWA comparison or by using the AddOrReplaceReadGroups tool in GATK4.

```
gatk AddOrReplaceReadGroups -I SRR7_sorted.bam -O SRR7_header.bam -LB lib1 -PL illumina -PU unit1 -SM 20
```

 Copyright © Huawei Technologies Co., Ltd.

If the information in the red box shown in the following figure is displayed, the execution is complete.

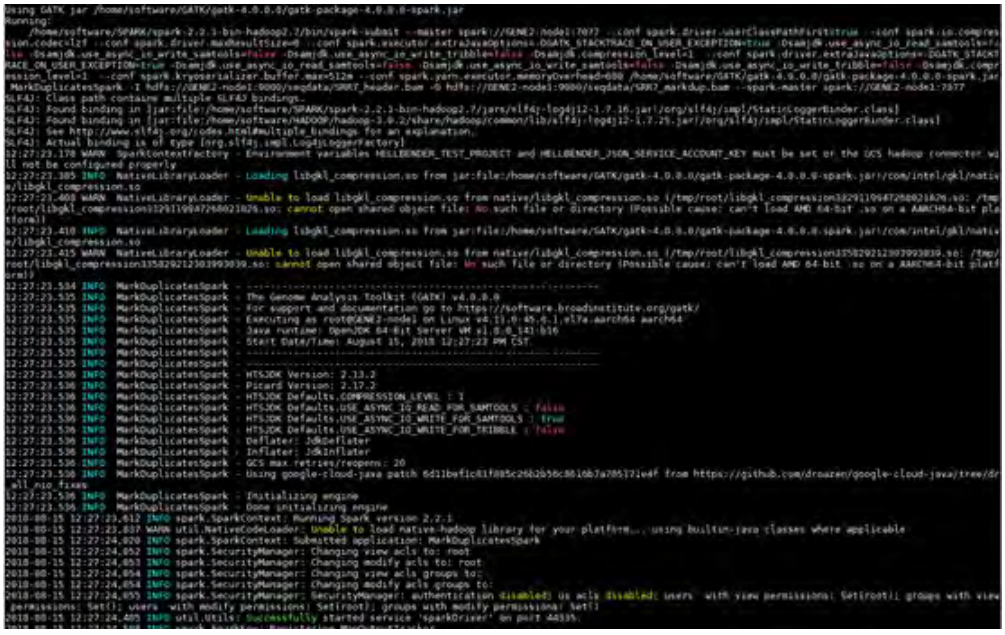
Step 8 Deduplicate gene sequences.

1. Run the following commands to upload the head file to HDFS:
   hadoop fs -put SRR7_header.bam /seqdata
2. Run the following command to query HDFS:
   hadoop fs –ls /seqdata

3. Run the following command to perform the deduplication.
   gatk MarkDuplicatesSpark -I hdfs://Hostname:9000/seqdata/SRR7_header.bam -O hdfs://Hostname:9000/seqdata/SRR7_markdup.bam -- --spark-runner SPARK --spark-master spark://Hostname:7077

   NOTE

   Change the host names based on the master nodes of Hadoop and Spark. For details, see **Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)**.



If the information in the red box shown in the following figure is displayed, the execution is complete.

Step 9 BQSR performs the recalibration.

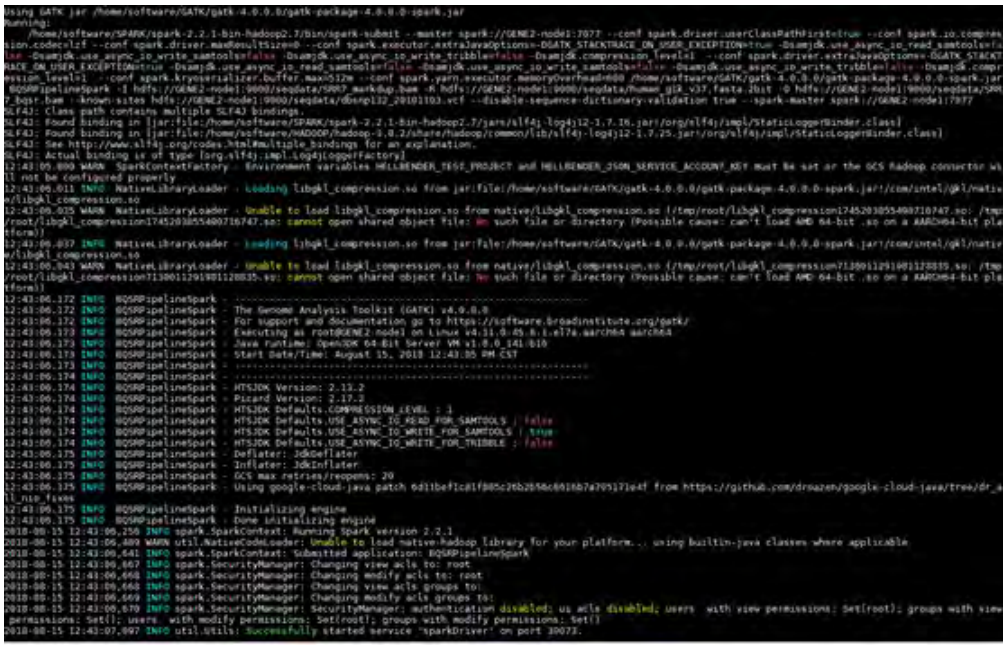
1. Run the following commands to upload the original file to HDFS:
   hadoop fs –put dbsnp132_20101103.vcf /seqdata
   hadoop fs –put human_g1k_v37.fasta.2bit /seqdata
   hadoop fs –ls /seqdata

   If the information in the red box shown in the following figure is displayed, the execution is complete.

2. Perform base quality score recalibration (BQSR).
   gatk BQSRPipelineSpark -I hdfs://Hostname:9000/seqdata/SRR7_markdup.bam -R hdfs://Hostname:9000/seqdata/human_g1k_v37.fasta.2bit -O hdfs://Hostname:9000/seqdata/SRR7_bqsr.bam --known-sites hdfs://Hostname:9000/seqdata/dbsnp132_20101103.vcf --disable-sequence-dictionary-validation true -- --spark-runner SPARK --spark-master spark://Hostname:7077

   NOTE

   Change the host names based on the master nodes of Hadoop and Spark. For details, see **Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)**.



   If the information in the red box shown in the following figure is displayed, the execution is complete.

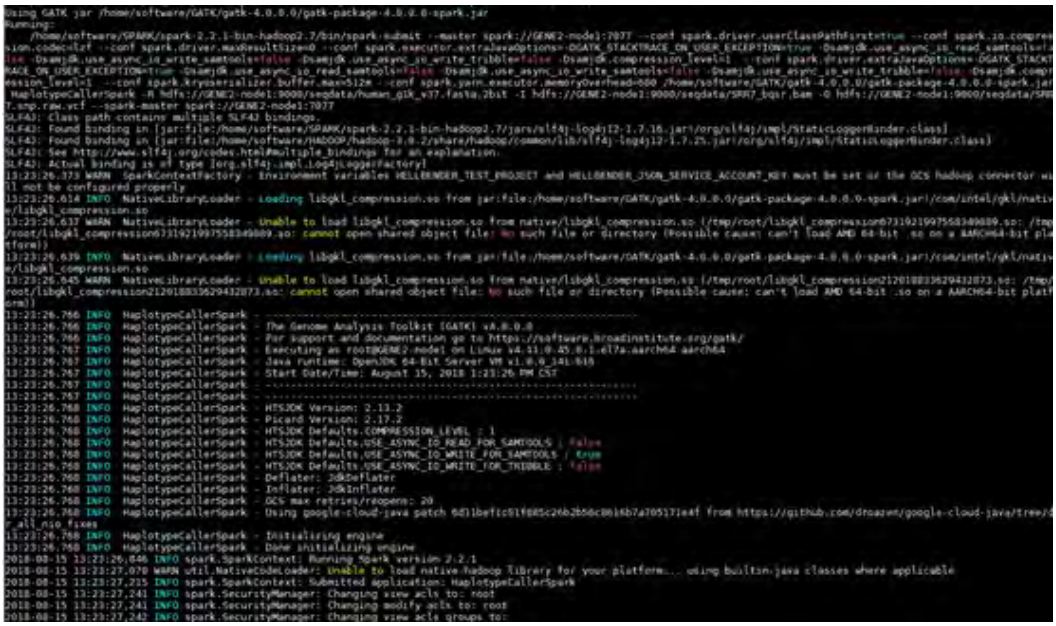
 Copyright © Huawei Technologies Co., Ltd.

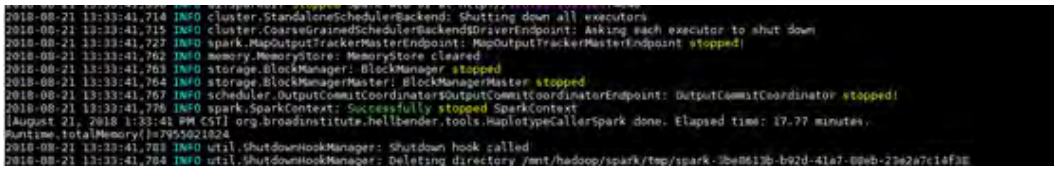
Step 10 Run the following command to detect the HaplortypeCaller variation process and generate the variant **vcf** file:

```
gatk HaplotypeCallerSpark -R hdfs://Hostname:9000/seqdata/human_g1k_v37.fasta.2bit -I hdfs://Hostname:9000/seqdata/SRR7_bqsr.bam -O hdfs://Hostname:9000/seqdata/SRR7.snp.raw.vcf -- --spark-runner SPARK --spark-master spark://Hostname:7077
```

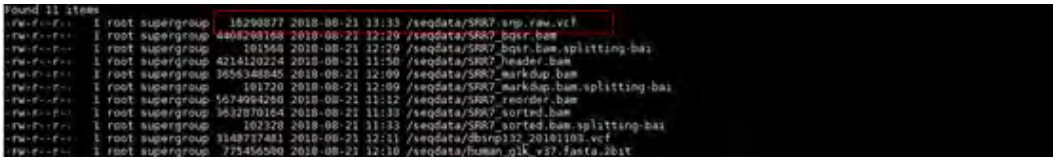
NOTE

Change the host names based on the master nodes of Hadoop and Spark. For details, see **Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)**.





The generated data is as follows:



----End

4.5.7.2 Running GATK4 in Non-Spark Mode

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create an index for the FASTA file:

```
bwa index -a bwtsw human_g1k_v37.fasta
```

Step 3 Run the following commands to compare the **fastq** file:

```
gatk CreateSequenceDictionary -R human_g1k_v37.fasta -O human_g1k_v37.dict
bwa mem -M -t 64 human_g1k_v37.fasta SRR742200_1.fastq SRR742200_2.fastq > SRR7.sam
```

Step 4 Run the following command to re-sort the gene sequence:

```
gatk ReorderSam -I SRR7.sam -O SRR7_reorder.bam -R human_g1k_v37.fasta
```

Step 5 Run the following command to sort gene sequences by coordinate in descending order:

```
gatk SortSam -I SRR7_reorder.bam -O SRR7_sorted.bam --SORT_ORDER coordinate
```

Step 6 Run the following command to add a head for the gene sequence:

```
gatk AddOrReplaceReadGroups -I SRR7_sorted.bam -O SRR7_addhead.bam -LB lib1 -PL illumina -PU unit1 -SM 20
```

Step 7 Run the following command to deduplicate the gene sequences:

```
gatk MarkDuplicates -I SRR7_addhead.bam -M test.metric -O SRR7_markdup.bam
```

Step 8 Run the following command to perform the recalibration:

1. Re-sort the FASTA reference files to generate the **fai** file:
   ```
   samtools faidx human_g1k_v37.fasta > human_g1k_v37.fai
   ```
2. Add an index to the **vcf** file:
   ```
   gatk IndexFeatureFile -F dbsnp132_20101103.vcf
   gatk BaseRecalibrator -I SRR7_markdup.bam --known-sites dbsnp132_20101103.vcf -O SRR7_bqsr.bam -R human_g1k_v37.fasta
   gatk ApplyBQSR -bqsr SRR7_bqsr.bam -I SRR7_markdup.bam -O SRR7_aybqsr.bam
   ```

Step 9 Run the following command to check the HaplortypeCaller variation process:

```
gatk HaplotypeCaller -I SRR7_aybqsr.bam -O SRR7_raw.vcf -R human_g1k_v37.fasta
```

----End

4.5.8 Troubleshooting

An Error Is Reported When GATK Is Run

Symptom:

An error message similar to the following is displayed when running GATK: "Unable to load libgkl_compression.so from native/libgkl_compression.so (/tmp/libgkl_compression8107837509941713683.so: /tmp/libgkl_compression8107837509941713683.so: cannot open shared object file: No such file or directory (Possible cause: can't load AMD 64-bit .so on an AARCH64-bit platform))".

Possible Cause

GKL is an acceleration library developed based on the Intel CPU, and needs to be ported to the Kunpeng platform.

Procedure

Ignore this error if performance comparison is not involved.

4.5.9 More Resources

GATK user guide:

https://software.broadinstitute.org/gatk/documentation/

 Copyright © Huawei Technologies Co., Ltd.

Gene file database:

https://hgdownload.soe.ucsc.edu/downloads.html

4.6 HISAT2 2.1.0 Porting Guide (CentOS 7.6)

4.6.1 Introduction

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (whole-genome, transcriptome, and exome sequencing data) against the general human population (as well as against a single reference genome). Based on an extension of BWT for a graph, a graph FM index (GFM) has been designed and implemented. In addition to using one global GFM index that represents general population, HISAT2 uses a large set of small GFM indexes that collectively cover the whole genome (each index representing a genomic region of 56 kbp, with 55,000 indexes needed to cover human population).

For more information, visit the **official HISAT2 website**.

Programming language: C++

Brief description: fast and sensitive alignment program.

Open source license: GPL 3.0

Recommended Software Version

HISAT2 2.1.0

4.6.2 Environment Requirements

Hardware Requirements

Table 4-27 lists the hardware requirements.

Table 4-27 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-28 lists the software requirements.

Table 4-28 Software requirements

Item	Version	Download Address
HISAT2	2.1.0	**https://github.com/infphilo/hisat2**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
sse2neon.h	master	https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h
bdgp6.tar.gz	NA	ftp://ftp.ccb.jhu.edu/pub/infphilo/hisat2/data/bdgp6.tar.gz
GTF file	Homo_sapiens.GRCh38.86.chr.gtf.gz	ftp://ftp.ensembl.org/pub/release-86/gtf/homo_sapiens/Homo_sapiens.GRCh38.86.chr.gtf.gz
FASTQ file	R1.fq.gz	http://opengene.org/data/R1.fq.gz
FASTQ file	R2.fq.gz	http://opengene.org/data/R2.fq.gz
FASTA file	Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz	https://ftp.ensembl.org/pub/release-84/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz

OS Requirements

Table 4-29 lists the OS requirements.

Table 4-29 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.6.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HISAT2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-30 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HISAT2	Installation path of HISAT2	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Planned path for storing the test cases of HISAT2	

4.6.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-31 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

4.6.5 Obtaining the Source Code

Procedure

Step 1 Download the HISAT2 installation package **hisat2-2.1.0.tar.gz**.

https://github.com/infphilo/hisat2

Step 2 Use an SFTP tool to upload the HISAT2 installation package to the */path/to/HISAT2* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.6.6 Compiling and Installing HISAT2

Prerequisites

You have downloaded the **sse2neon.h** file from **https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h**.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Copy the downloaded **sse2neon.h** file to the */path/to/GNU***/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include** directory.

Step 3 Run the following commands to decompress the HISAT2 installation package:

```
cd /path/to/HISAT2
tar -xvf hisat2-2.1.0.tar.gz
cd hisat2-2.1.0
```

Step 4 Run the following commands to set environment variables:

```
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

Step 5 Perform the following steps to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to go to the insert mode, and modify the **BITS=64 aarch64** branches of the file by inserting the following content in bold between line 140 and line 141:

   Currently, **BITS=64** supports only x86 and does not support AArch64. The AArch64 branch needs to be added.

   Before the modification:
   ```
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ```
   After the modification:
   ```
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ifeq (aarch64,$(shell uname -m))
   BITS=64
   endif
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the following steps to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to go to the insert mode, and modify the **-m64** and **-msse2** AArch64 branches of the file by inserting the following content in bold between line 159 and line 160.

   The compilation options **-m64** and **-msse2** in **Makefile** support only x86 and does not support AArch64. The AArch64 branch needs to be added.

   Before the modification (line 149 to line 156 in the source file):

 Copyright © Huawei Technologies Co., Ltd.

```
ifeq (32,$(BITS))
	BITS_FLAG = -m32
endif

ifeq (64,$(BITS))
	BITS_FLAG = -m64
endif
SSE_FLAG=-msse2
```

After the modification:

```
ifeq (32,$(BITS))
	BITS_FLAG = -m32
endif

ifeq (64,$(BITS))
	BITS_FLAG = -m64
endif
SSE_FLAG=-msse2

ifeq (aarch64,$(shell uname -m))
	BITS_FLAG =
	SSE_FLAG =
endif
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Perform the following steps to modify the **Makefile** file:

1. Open **Makefile**.

```
vi Makefile
```

2. Press **i** to go to the insert mode and modify the **FLAGS** parameters in the file.

Before the modification:

```
EXTRA_FLAGS += -DPOPCNT_CAPABILITY
DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG)
DEBUG_DEFS     = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\""
RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3
RELEASE_DEFS   = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

After the modification:

```
EXTRA_FLAGS +=
DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG) -funroll-loops -std=c++98
DEBUG_DEFS     = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\""
RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3 -std=c++98 -Xlinker --allow-
multiple-definition
RELEASE_DEFS   = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the following steps to modify the **aligner_sw.h** and **sse_util.h** header files:

1. Upload the **sse2neon.h** file to the **third_party** directory.
2. Open **aligner_sw.h**.

```
vi aligner_sw.h
```

3. Press **i** to go to the insert mode. Replace **#include <emmintrin.h>** with **#include <sse2neon.h>**.

Before the modification:

```
#include <emmintrin.h>
```

After the modification:

```
#include <sse2neon.h>
```

 Copyright © Huawei Technologies Co., Ltd.

4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
5. Modify the **sse_util.h** file. For details, see **Step 8.2** to **Step 8.4**.

Step 9 Run the following command to compile and install HISAT2:

```
make
```

----End

4.6.7 Running and Verifying HISAT2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the */path/to/CASE* directory:

```
cd /path/to/CASE
```

Step 3 Run the following command to set the environment variables for the HISAT executable file:

```
export PATH=/path/to/HISAT2/hisat2-2.1.0/:$PATH
```

Step 4 Run the following command to decompress **bdgp6.tar.gz**, which is listed in **Table 4-28**:

```
cd /path/to/CASE
tar -xvf bdgp6.tar.gz
cd bdgp6
```

Step 5 Run the following commands to modify the **make_bdgp6.sh** file:

1. Open **make_bdgp6.sh**.
   ```
   vi make_bdgp6.sh
   ```
2. Press **i** to enter the insert mode and comment out the following line:
   ```
   # ENSEMBL_BASE=ftp://xxxxxxxx
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Use an SFTP tool to copy the **Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz** file to the */path/to/CASE/***bdgp6** directory.

Step 7 Run the following commands to decompress the files:

```
gzip -d Homo_sapiens.GRCh38.86.chr.gtf.gz
gzip -d R1.fq.gz
gzip -d R2.fq.gz
```

Step 8 Run the following command to run the **make_bdgp6.sh** file:

```
./make_bdgp6.sh
```

Step 9 Run the following commands to create the transcriptome.

```
extract_exons.py Homo_sapiens.GRCh38.86.chr.gtf > genome.exon
extract_splice_sites.py Homo_sapiens.GRCh38.86.chr.gtf > genome.ss
```

Step 10 Run the following command to index the genome and transcriptome:

```
hisat2-build -p96 genome.fa --ss genome.ss --exon genome.exon genome_tran
```

Step 11 Execute the following command to run HISAT2:

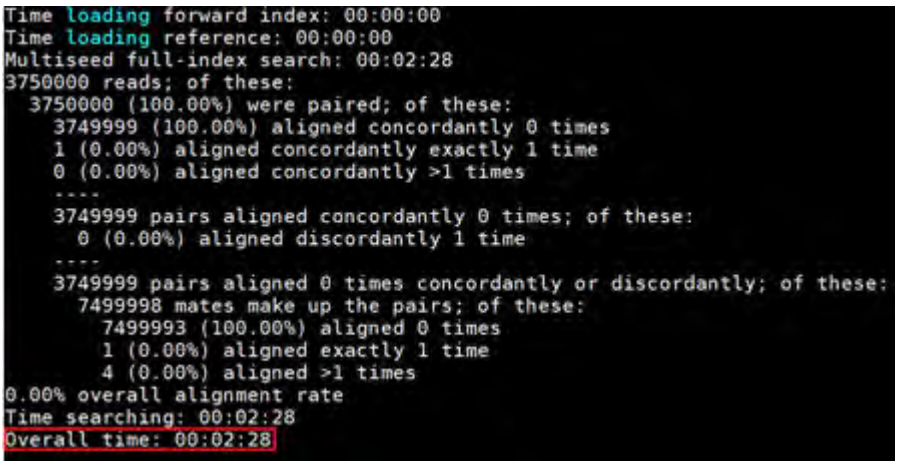
```
hisat2 -t -p 96 -x genome_tran -1 R1.fq -2 R2.fq -S Test.sam
```

Check the value of **Overall time** (the format is [hh]:[mm]:[ss]) in the log. A smaller value indicates better performance.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-3 shows an example of the test result.

Figure 4-3 Test example



----End

4.6.8 More Information

daehwankimlab website:

http://daehwankimlab.github.io/hisat2/manual/

4.7 STAR 2.7.1a Porting Guide (CentOS 7.6)

4.7.1 Introduction

STAR is an ultrafast universal RNA-seq aligner.

Motivation

Accurate alignment of high-throughput RNA-seq data is a challenging and yet unsolved problem because of the non-contiguous transcript structure, relatively short read lengths and constantly increasing throughput of the sequencing technologies. Currently available RNA-seq aligners suffer from high mapping error rates, low mapping speed, read length limitation and mapping biases.

Results

To align a large (> 8 billion reads) ENCODE Transcriptome RNA-seq dataset, the Spliced Transcripts Alignment to a Reference (STAR) software based on a previously undescribed RNA-seq alignment algorithm uses sequential maximum mappable seed search in uncompressed suffix arrays followed by seed clustering and stitching procedure. STAR outperforms other aligners by aligning to the human genome 550 million 2 × 76 bp paired-end reads per hour on a modest 12-core server, while at the same time improving alignment sensitivity and precision. In addition to unbiased de novo detection of canonical junctions, STAR can discover non-canonical splices and chimeric (fusion) transcripts, and is also capable of mapping full-length RNA sequences. Using Roche 454 sequencing of reverse transcription polymerase chain reaction amplicons, experiments validated

 Copyright © Huawei Technologies Co., Ltd.

1960 novel intergenic splice junctions with an 80% to 90% success rate, corroborating the high precision of the STAR mapping strategy.

Availability and Implementation

STAR is implemented as a standalone C++ code. STAR is free open source software distributed under GPLv3 license and can be downloaded from http://code.google.com/p/rna-star/.

For more information, visit the **official STAR website**.

Programming language: C++

Brief description: an RNA-seq aligner

Open source license: GPL 3.0

Recommended Software Version

STAR 2.7.1a

4.7.2 Environment Requirements

Hardware Requirements

Table 4-32 lists the hardware requirements.

Table 4-32 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-33 lists the software requirements.

Table 4-33 Software requirements

Item	Version	Download Address
STAR	2.7.1a	**https://github.com/alexdobin/STAR**
Test case file 1	Homo_sapiens.GRCh38.dna.chromosome.2.fa.gz	**ftp://ftp.ensembl.org/pub/release-97/fasta/homo_sapiens/dna/ftp://ftp.ensembl.org/pub/release-97/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.chromosome.2.fa.gz**
Test case file 2	Homo_sapiens.GRCh38.86.chr.gtf.gz	**ftp://ftp.ensembl.org/pub/release-86/gtf/homo_sapiens/Homo_sapiens.GRCh38.86.chr.gtf.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
Test case file 3	TG_r1.fastq.gz TG_r2.fastq.gz	http://tools.genxpro.net:9000/omiras/testdata/

OS Requirements

Table 4-34 lists the OS requirements.

Table 4-34 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.7.3 Paths for Software Porting

This chapter lists the software installation paths involved in the STAR software porting.

Table 4-35 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/STAR	Installation path of STAR	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing the STAR test cases	

4.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-36 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.7.5 Obtaining the Source Code

Procedure

Step 1 Download the STAR installation package **STAR-2.7.1a.tar.gz**.

https://github.com/alexdobin/STAR

Step 2 Use an SFTP tool to upload the STAR installation package to the */path/to/STAR* directory on the server.

----End

4.7.6 Compiling and Installing STAR

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the system dependency **zlib-devel**:

```
yum install zlib-devel -y
```

NOTE

The Yum source is a local source of the CentOS 7.6 image. You can modify the files in the **/etc/yum.repos.d** directory as required.

Step 3 Run the following commands to decompress the STAR installation package:

```
tar -xvf STAR-2.7.1a.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd STAR-2.7.1a
export BLAST_TOP=`pwd`
```

Step 5 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi source/Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   LDFLAGSextra ?= -flto -march=native -mcpu=tsv110
   CXXFLAGSextra ?= -flto -march=native -mcpu=tsv110
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to compile and install STAR:

```
cd source
make STAR
```

Step 7 Run the following command to check whether an executable file is generated:

```
-rwxr-xr-x 1 root root 2018464 Aug  9 13:29 STAR
```

Step 8 Run the following command to set STAR environment variables:

```
export PATH=/path/to/STAR/STAR-2.7.1a/source/:$PATH
```

----End

4.7.7 Running and Verifying STAR

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the test cases are stored:

```
cd /path/to/CASE
```

Step 3 Run the following commands to decompress the test case packages:

```
gzip -d Homo_sapiens.GRCh38.dna.chromosome.2.fa.gz
gzip -d Homo_sapiens.GRCh38.86.chr.gtf.gz
```

Step 4 Run the following commands to create a genome index:

```
mkdir chr1_hg38_index
{ time STAR --runThreadN 96 --genomeSAindexNbases 12 --runMode genomeGenerate --genomeDir chr1_hg38_index --genomeFastaFiles Homo_sapiens.GRCh38.dna.chromosome.2.fa --sjdbGTFfile Homo_sapiens.GRCh38.86.chr.gtf --sjdbOverhang 99 ; } 2>&1 |tee genome-index.log
```

Step 5 Run the following command to compare genomes:

```
{ time STAR --runMode alignReads --outSAMtype BAM Unsorted --readFilesCommand zcat --genomeDir chr1_hg38_index/ --outFileNamePrefix Homo_sapiens.GRCh38 --readFilesIn TG_r1.fastq.gz TG_r2.fastq.gz ; } 2>&1 |tee Mapping.log
```

Step 6 After the genome comparison is complete, the following files and directories are generated:

- Homo_sapiens.GRCh38Aligned.out.bam
- Homo_sapiens.GRCh38Log.final.out
- Homo_sapiens.GRCh38Log.out
- Homo_sapiens.GRCh38Log.progress.out
- Homo_sapiens.GRCh38SJ.out.tab
- Homo_sapiens.GRCh38_STARtmp

Step 7 When the genome index is created and genome comparison is performed, the software does not automatically generate logs about the running time or performance. You need to run the **time** command to redirect the entire running time to the log file.

View the **real** value (*x* minutes *y* seconds) in the last **time** command output in the log. A smaller value indicates higher performance.

Information similar to the following is displayed.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@XA320V2-19 CASE]# cat Mapping.log
Jan 16 20:42:41 ..... started STAR run
Jan 16 20:42:41 ..... loading genome
Jan 16 20:42:42 ..... started mapping
Jan 16 20:42:53 ..... finished mapping
Jan 16 20:42:53 ..... finished successfully
real    0m11.581s
user    0m10.703s
sys     0m0.579s
```

----End

4.7.8 More Resources

HBCTraining website:

https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

4.8 TopHat2 2.1.1 Porting Guide (CentOS 7.6)

4.8.1 Introduction

TopHat2 is a program that quickly splices and maps RNA-Seq data. It uses the ultra-fast high-throughput short-read comparison program to compare RNA-Seq information to the mammalian size genome, then the mapping results are analyzed to identify the splice points between exons. TopHat is the result of a joint development of the University of Maryland Center for Bioinformation and Computer Biology, the Departments of Mathematics and Molecular and Cell Biology of the University of California, Berkeley, and the Harvard Department of Stem Cell and Regenerative Biology.

For more information, visit the **official TopHat2 website**.

Programming language: C

Brief description: a program that quickly splices and maps RNA-Seq data

Open source license: Boost Software License 1.0

Recommended Software Version

TopHat2 2.1.1

4.8.2 Environment Requirements

Hardware Requirements

Table 4-37 lists the hardware requirements.

Table 4-37 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-38 lists the software requirements.

Table 4-38 Software requirements

Item	Version	Download Address
TopHat2	2.1.1	https://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz
Bowtie2	2.3.5.1	https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1/bowtie2-2.3.5.1-source.zip
SIMDe	master	https://github.com/nemequ/simde
Boost	1.70.0	https://github.com/boostorg/boost/archive/refs/tags/boost-1.70.0.tar.gz
Test case	N/A	http://ccb.jhu.edu/software/tophat/downloads/test_data.tar.gz

OS Requirements

Table 4-39 lists the OS requirements.

Table 4-39 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.8.3 Paths for Software Porting

This chapter lists the software installation paths involved in the TopHat2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-40 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BOWTIE2	Installation path of Bowtie2	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/BOOST	Installation path of Boost	
4	/path/to/CASE	Path for storing the test case	
5	/path/to/TOPHAT2	Installation path of TopHat2	

4.8.4 Configuring the Compilation Environment

4.8.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-41 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Bowtie2	See **4.8.4.2 Installing Bowtie2**.
3	Installing Boost	See **4.8.4.3 Installing Boost**.

 Copyright © Huawei Technologies Co., Ltd.

4.8.4.2 Installing Bowtie2

The Bowtie2 or Bowtie commands are required for TopHat data test in the PATH environment. Therefore, you need to install Bowtie2 first.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the Bowtie2 installation directory:

```
cd /path/to/BOWTIE2
```

Step 3 Run the following command to decompress the Bowtie2 source code package:

```
unzip bowtie2-2.3.5.1-source.zip
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd bowtie2-2.3.5.1
```

Step 5 Run the following command to decompress SIMDe:

```
unzip simde-master.zip
```

Step 6 Run the following command to copy the SIMDe files to Bowtie2:

```
cp -a simde-master/simde bowtie2-2.3.5.1/third_party/simde/
```

Step 7 Run the following commands to check whether the **bowtie2-2.3.5.1/third_party/simde/simde/x86/sse2.h** file exists:

```
cd /path/to/BOWTIE2/bowtie2-2.3.5.1/third_party/simde/simde/x86
-rw-r--r-- 1 root root 99060 Aug  5 04:12 sse2.h
```

Step 8 Run the following command to switch to the **bowtie2-2.3.5.1** directory:

```
cd /path/to/BOWTIE2/bowtie2-2.3.5.1
```

Step 9 Run the following command to perform the compilation:

```
make NO_TBB=1 POPCNT_CAPABILITY=0 all
```

Step 10 Run the following command to load environment variables:

```
export PATH=/path/to/BOWTIE2/bowtie2-2.3.5.1:$PATH
```

----End

4.8.4.3 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Boost installation package:

```
tar xvf boost-1.70.0.tar.bz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd boost-boost-1.70.0
```

Step 4 Because TopHat requires only the thread and system libraries of Boost, Bootstrap only needs to contain the two libraries. Run the following command to specify the Boost installation path:

```
./bootstrap.sh --with-libraries=system,thread --prefix=/path/to/BOOST --with-toolset=gcc
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following command to perform the compilation and installation:

```
./b2 -d+2 -q -j 32 threading=multi link=shared variant=release --prefix=/path/to/BOOST install
```

Step 6 Run the following command to load environment variables:

```
export LD_LIBRARY_PATH=/path/to/BOOST/lib:$LD_LIBRARY_PATH
```

----End

4.8.5 Obtaining the Source Code

Procedure

Step 1 Download the installation package **tophat-2.1.1.tar.gz**.

https://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz

Step 2 Use an SFTP tool to upload the TopHat2 installation package to the */path/to/TOPHAT2* directory on the server.

----End

4.8.6 Compiling and Installing TopHat2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the **TOPHAT2** directory:

```
cd /path/to/TOPHAT2
```

Step 3 Run the following command to decompress the TopHat2 installation package:

```
tar xvf tophat-2.1.1.tar.gz
```

Step 4 Prepare a **Patch** file and copy the following content to the **tophat.patch** file in the same directory as the **tophat-2.1.1** directory.

1. Create a **tophat.patch** file.

   vi tophat.patch

2. Press **i** to enter the insert mode and add the following content:

```
diff -uBw -aur tophat-2.1.1/src/juncs_db.cpp tophat-2.1.1-patched/src/juncs_db.cpp
--- tophat-2.1.1/src/juncs_db.cpp   2016-02-14 19:21:17.354079000 +0100
+++ tophat-2.1.1-patched/src/juncs_db.cpp   2016-06-02 11:26:34.097425159 +0200
@@ -338,7 +338,7 @@
            uint32_t left_coord = atoi(scan_left_coord);
            uint32_t right_coord = atoi(scan_right_coord);
            bool antisense = *orientation == '-';
-           junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left_coord, right_coord,
antisense), JunctionStats()));
+           junctions.insert(make_pair(Junction(ref_id, left_coord, right_coord, antisense),
JunctionStats()));
        }
    }

diff -uBw -aur tophat-2.1.1/src/tophat_reports.cpp tophat-2.1.1-patched/src/tophat_reports.cpp
--- tophat-2.1.1/src/tophat_reports.cpp 2016-02-23 22:20:44.320710000 +0100
+++ tophat-2.1.1-patched/src/tophat_reports.cpp 2016-06-02 11:26:22.057239478 +0200
@@ -2705,7 +2705,7 @@
                junction_stat.gtf_match = true;
                junction_stat.accepted = true;
```

 Copyright © Huawei Technologies Co., Ltd.

```
-               gtf_junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left_coord,
right_coord, antisense), junction_stat));
+               gtf_junctions.insert(make_pair(Junction(ref_id, left_coord, right_coord, antisense),
junction_stat));
            }
        }
        fprintf(stderr, "Loaded %d GFF junctions from %s.\n", (int)(gtf_junctions.size()),
gtf_juncs.c_str());
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following command to install the patch file:

```
patch -Np0 -i tophat.patch
```

Step 6 Run the following command to switch to the directory generated after the package is decompressed:

```
cd tophat-2.1.1
```

Step 7 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/TOPHAT2 --with-boost=/path/to/BOOST
```

Step 8 Run the following commands to perform the compilation and installation:

```
make -j 1
make install
```

----End

4.8.7 Running and Verifying TopHat2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to set TopHat2 environment variables:

```
export PATH=/path/to/TOPHAT2/bin:$PATH
```

Step 3 Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/CASE
```

Step 4 Run the following command to decompress the test case:

```
tar xvf test_data.tar.gz
```

Step 5 Run the following command to switch to the directory where the test case is stored:

```
cd test_data
```

Step 6 Run the following command to run the TopHat2 test:

```
tophat2 -r 20 test_ref reads_1.fq reads_2.fq
```

If "Run complete" is displayed, the test case is executed properly. If "FAILED" is displayed, an error occurs.

View the **real time** value (in seconds) in the last **time** command output in the log. A smaller value indicates higher performance.

The following is an example of the output result.

```
[2020-01-17 15:30:37] Beginning TopHat run (v2.1.1)
-----------------------------------------------
[2020-01-17 15:30:37] Checking for Bowtie
                  Bowtie version:     2.3.5.1
```

 Copyright © Huawei Technologies Co., Ltd.

```
[2020-01-17 15:30:37] Checking for Bowtie index files (genome)..
        Found both Bowtie1 and Bowtie2 indexes.
[2020-01-17 15:30:37] Checking for reference FASTA file
[2020-01-17 15:30:37] Generating SAM header for test_ref
[2020-01-17 15:30:37] Preparing reads
         left reads: min. length=75, max. length=75, 100 kept reads (0 discarded)
        right reads: min. length=75, max. length=75, 100 kept reads (0 discarded)
[2020-01-17 15:30:37] Mapping left_kept_reads to genome test_ref with Bowtie2
[2020-01-17 15:30:38] Mapping left_kept_reads_seg1 to genome test_ref with Bowtie2 (1/3)
[2020-01-17 15:30:38] Mapping left_kept_reads_seg2 to genome test_ref with Bowtie2 (2/3)
[2020-01-17 15:30:38] Mapping left_kept_reads_seg3 to genome test_ref with Bowtie2 (3/3)
[2020-01-17 15:30:38] Mapping right_kept_reads to genome test_ref with Bowtie2
[2020-01-17 15:30:38] Mapping right_kept_reads_seg1 to genome test_ref with Bowtie2 (1/3)
[2020-01-17 15:30:38] Mapping right_kept_reads_seg2 to genome test_ref with Bowtie2 (2/3)
[2020-01-17 15:30:38] Mapping right_kept_reads_seg3 to genome test_ref with Bowtie2 (3/3)
[2020-01-17 15:30:38] Searching for junctions via segment mapping
[2020-01-17 15:30:38] Retrieving sequences for splices
[2020-01-17 15:30:38] Indexing splices
Building a SMALL index
[2020-01-17 15:30:39] Mapping left_kept_reads_seg1 to genome segment_juncs with Bowtie2 (1/3)
[2020-01-17 15:30:39] Mapping left_kept_reads_seg2 to genome segment_juncs with Bowtie2 (2/3)
[2020-01-17 15:30:39] Mapping left_kept_reads_seg3 to genome segment_juncs with Bowtie2 (3/3)
[2020-01-17 15:30:39] Joining segment hits
[2020-01-17 15:30:39] Mapping right_kept_reads_seg1 to genome segment_juncs with Bowtie2 (1/3)
[2020-01-17 15:30:39] Mapping right_kept_reads_seg2 to genome segment_juncs with Bowtie2 (2/3)
[2020-01-17 15:30:39] Mapping right_kept_reads_seg3 to genome segment_juncs with Bowtie2 (3/3)
[2020-01-17 15:30:39] Joining segment hits
[2020-01-17 15:30:40] Reporting output tracks
-----------------------------------------------
[2020-01-17 15:30:40] A summary of the alignment counts can be found in ./tophat_out/align_summary.txt
[2020-01-17 15:30:40] Run complete: 00:00:02 elapsed
```

----End

4.8.8 More Information

Patch file:

https://aur.archlinux.org/cgit/aur.git/tree/make_pair.patch?h=tophat

4.9 Bowtie2 2.4.1 Porting Guide (CentOS 7.6)

4.9.1 Introduction

Bowtie2 is a tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100 or 1000 of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes.

It is usually the first step in aligning the genomic (including identifying variation calling, ChIP-seq, RNA-seq, BS-seq) pipelines.

It can handle very long readings (for example, 10s or 100s of kilobytes), but it optimizes the length and error pattern of readings generated by recent sequencers, such as Illumina HiSeq 2000, Roche 454, and Ion Torrent instruments.

Bowtie2 indexes the genome with an FM Index (based on Burrows-Wheeler Transform or BWT) to keep its memory footprint small. For the human genome, the memory footprint is about 3.2 GB. Bowtie2 supports gapped, local, and end-to-end alignment modes. Multiple processors can be used at the same time to greatly improve the alignment speed.

 Copyright © Huawei Technologies Co., Ltd.

For more information, visit the **official Bowtie2 website**.

Programming language: C

Brief description: A tool for aligning sequencing reads to long reference sequences

Open source license: GPL 3.0

Recommended Software Version

Bowtie2 2.4.1

4.9.2 Environment Requirements

Hardware Requirements

Table 4-42 lists the hardware requirements.

Table 4-42 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-43 lists the software requirements.

Table 4-43 Software requirements

Item	Version	Download Address
Bowtie2	2.4.1	**https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip**
Python 3	3.7.0	**https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz**
SIMDe	master	**https://github.com/nemequ/simde**
sse2neno	master	**https://github.com/jratcliff63367/sse2neon/archive/master.zip**
Test case	chromFa.tar.gz	**http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz**

OS Requirements

Table 4-44 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-44 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.9.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Bowtie2 software porting.

Table 4-45 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BOWTIE2	Installation path of Bowtie2	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PYTHON3	Installation path of Python 3	
4	/path/to/SIMDE	Installation path of SIMDe	
5	/path/to/SSE2NENO	Installation path of sse2neon	

4.9.4 Configuring the Compilation Environment

4.9.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-46 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Python 3	See **4.9.4.2 Installing Python 3**.

4.9.4.2 Installing Python 3

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Python 3 installation package:

```
tar -xvf Python-3.7.0.tgz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd Python-3.7.0
```

Step 4 Run the following commands to compile and install Python 3:

```
./configure --prefix=/path/to/PYTHON3
make -j16
make install
```

Step 5 Run the following commands to set the Python 3 environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 6 Run the following command to check whether the Python 3 environment has taken effect:

```
python3 -V
```

----End

4.9.5 Obtaining the Source Code

Procedure

Step 1 Download the Bowtie2 installation package **bowtie2-2.4.1-source.zip**.

https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip

Step 2 Use an SFTP tool to upload the Bowtie2 installation package to the */path/to/BOWTIE2* directory on the server.

----End

4.9.6 Compiling and Installing bowtie 2

4.9.6.1 Installing Bowtie2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install **zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64** using Yum:

```
yum install zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64
```

Step 3 Run the following command to decompress the installation package:

```
unzip bowtie2-2.4.1-source.zip
```

Step 4 Run the following commands to switch to the directory generated after the package is decompressed:

```
cd bowtie2-2.4.1
mkdir -p third_party/simde/simde/x86/
```

Step 5 Run the following commands to perform the configuration:

```
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

Step 6 Run the following command to decompress SIMDe:

```
unzip simde-master.zip
```

Step 7 Run the following command to copy the SIMDe files to Bowtie2:

```
cp -a /path/to/SIMDE/simde-master/simde /path/to/BOWTIE2/bowtie2-2.4.1/third_party/simde/
```

Step 8 Run the following command to decompress the sse2neno installation package:

```
unzip master.zip
```

Step 9 Run the following command to switch to the directory generated after the package is decompressed:

```
cd sse2neon-master
```

Step 10 Run the following command to copy the **sse2neon.h** file to Bowtie2:

```
cp -a SSE2NEON.h /path/to/BOWTIE2/bowtie2-2.4.1/third_party/simde/simde/x86/
```

Step 11 Run the following commands to perform the compilation:

```
cd ../bowtie2-2.4.1
make all
```

Step 12 Run the following command to set environment variables:

```
export PATH=/path/to/BOWTIE2/bowtie2-2.4.1:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.9.6.2 Creating a Bowtie2 Index

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory generated after the package is decompressed:

```
tar -zxvf chromFa.tar.gz
```

Step 3 Run the following command to perform the redirection:

```
cat *.fa > mm10.fa
```

Step 4 Run the following command to create an index:

```
bowtie2-build mm10.fa mm10
```

----End

4.9.7 Running and Verifying Bowtie2

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to run Bowtie2 and obtain the **SAM** file:

```
bowtie2 -p 6 -3 5 --local -x mm10 -1 bowtie2-2.4.1/example/reads/reads_1.fq -2 bowtie2-2.4.1/example/reads/reads_2.fq -S example.sam
```

The following is an example of the test result.

```
10000 reads; of these:
10000 (100.00%) were paired; of these:
10000 (100.00%) aligned concordantly 0 times
0 (0.00%) aligned concordantly exactly 1 time
0 (0.00%) aligned concordantly >1 times
0.00% overall alignment rate
```

----End

4.9.8 More Information

Bowtie2 Installation Guide (in Chinese only):

https://blog.csdn.net/u011262253/article/details/79833969

Download **chromFa.tar.gz**.

http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz

4.10 FastQC 0.11.9 Porting Guide (CentOS 7.6)

4.10.1 Introduction

FastQC is a quality control tool for high-throughput sequence data. It aims to provide a simple method to check the quality of original sequence data from high-throughput sequencing pipes.

 Copyright © Huawei Technologies Co., Ltd.

For more information, visit the **official FastQC website**.

Programming language: C/Fortran 90

Brief description: a quality control tool for gene sequence data

Open source license: GPL 3.0

Recommended Software Version

FastQC 0.11.9

4.10.2 Environment Requirements

Hardware Requirements

Table 4-47 lists the hardware requirements.

Table 4-47 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-48 lists the software requirements.

Table 4-48 Software requirements

Item	Version	Download Address
FastQC	0.11.9	**https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip**
OpenJDK	1.8.0	Provided by the system image
Test case	B17NC_R1	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz**

OS Requirements

Table 4-49 lists the OS requirements.

Table 4-49 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.10.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FastQC software porting.

Table 4-50 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FASTQC	Installation path of FastQC	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.10.4 Configuring the Compilation Environment

4.10.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-51 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenJDK	See **4.10.4.2 Installing OpenJDK**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing the GUI	See **4.10.4.3 Installing the GUI**.

4.10.4.2 Installing OpenJDK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to check whether OpenJDK is installed. If it is installed, skip **Step 3**.

```
java -version
```

NOTE

FastQC requires that the OpenJDK version be later than 1.6.0. Otherwise, compatibility issues may occur.

Step 3 Run the following command to install OpenJDK:

```
yum install java* -y
```

----End

4.10.4.3 Installing the GUI

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the X11 GUI:

```
yum -y groupinstall "Server with GUI"
```

----End

4.10.5 Obtaining the Source Code

Procedure

Step 1 Download the FastQC installation package **fastqc_v0.11.9.zip**.

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip

Step 2 Use an SFTP tool to upload the FastQC installation package to the */path/to/FASTQC* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.10.6 Compiling and Installing FastQC

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/FASTQC
```

Step 3 Run the following command to copy the installation package to the main program installation directory:

```
cp fastqc_v0.11.9.zip /path/to/FASTQC
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/FASTQC
```

Step 5 Run the following command to decompress the FastQC installation package:

```
unzip fastqc_v0.11.9.zip
```

Step 6 Run the following command to switch to the directory generated after the package is decompressed:

```
cd FastQC
```

Step 7 Run the following command to configure the address displayed on the GUI:

```
export DISPLAY=IP address of the server that can be connected to the PC:0.0
```

Step 8 Run the following command to check whether the **fastqc** script is available:

```
chmod +755 fastqc
./fastqc
```

Figure 4-4 shows an example of the test result:

Figure 4-4 Result example

NOTE

On a non-Windows 7 OS, skip **Step 7** and **Step 8**.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.10.7 Running and Verifying FastQC

Procedure (Using CLI)

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a test directory:

```
mkdir -p /path/to/FASTQC/test
```

Step 3 Run the following commands to copy the test case to the test directory and switch to the test directory:

```
cp B17NC_R1.fastq.gz /path/to/FASTQC/test
cd /path/to/FASTQC/test
```

Step 4 Run the following command to decompress the test case:

```
gzip -d B17NC_R1.fastq.gz
```

Step 5 Run the following command to run the test:

```
/path/to/FASTQC/FastQC/fastqc -o ./ ./B17NC_R1.fastq
```

Step 6 After the execution is complete, the **B17NC_R1_fastqc.html** and **B17NC_R1_fastqc.zip** files are generated. Run the following command to check whether the files exist:

```
ll
```

Use a browser to open the **html** file and view the result.

----End

Procedure (Using GUI)

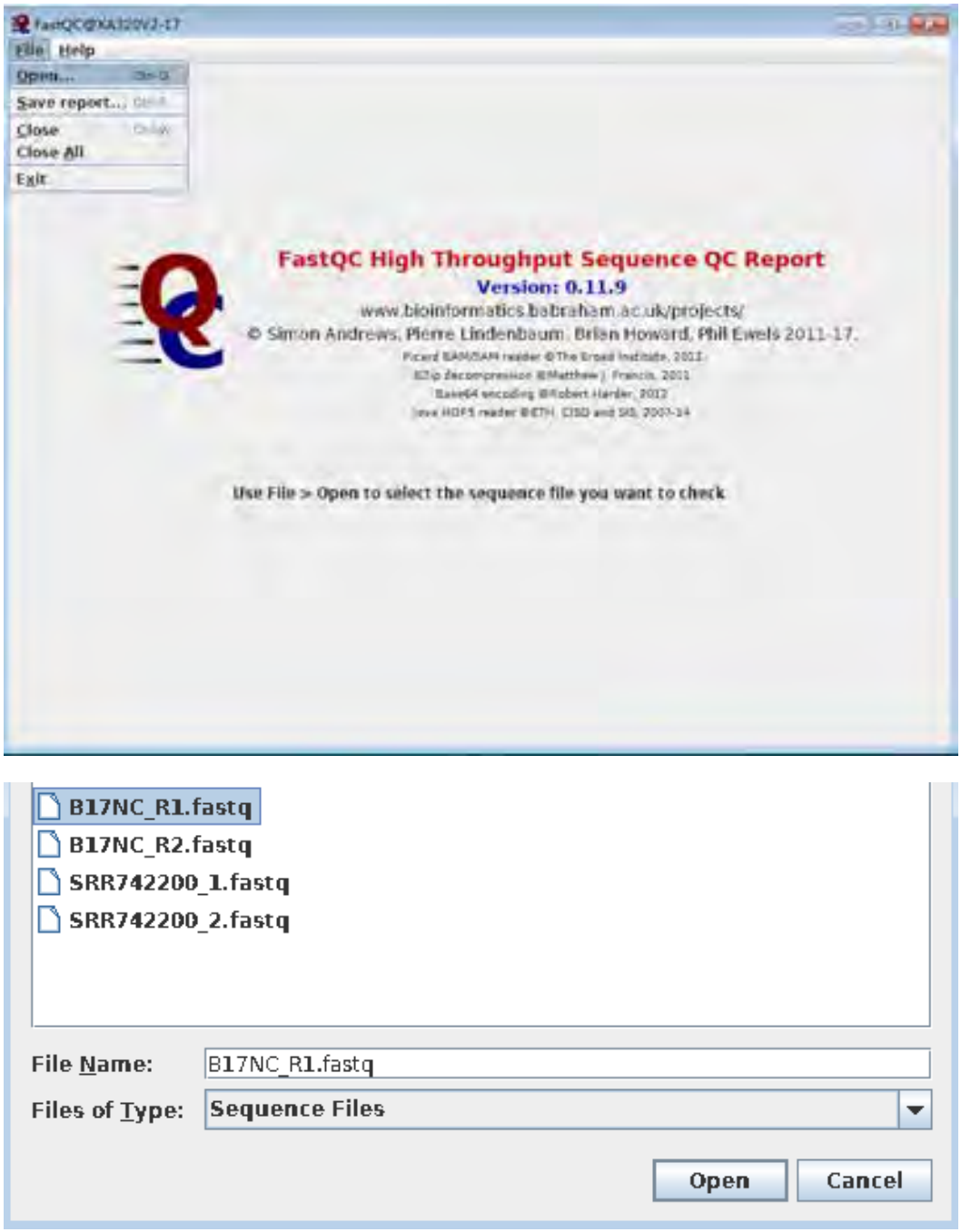
Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to log in to the GUI:

```
/path/to/FASTQC/FastQC/fastqc
```

Step 3 Click **File** > **Open** in the upper left corner and select a case.

 Copyright © Huawei Technologies Co., Ltd.



Step 4 Select a test case and click **Open**. If the following page is displayed, the test is running.

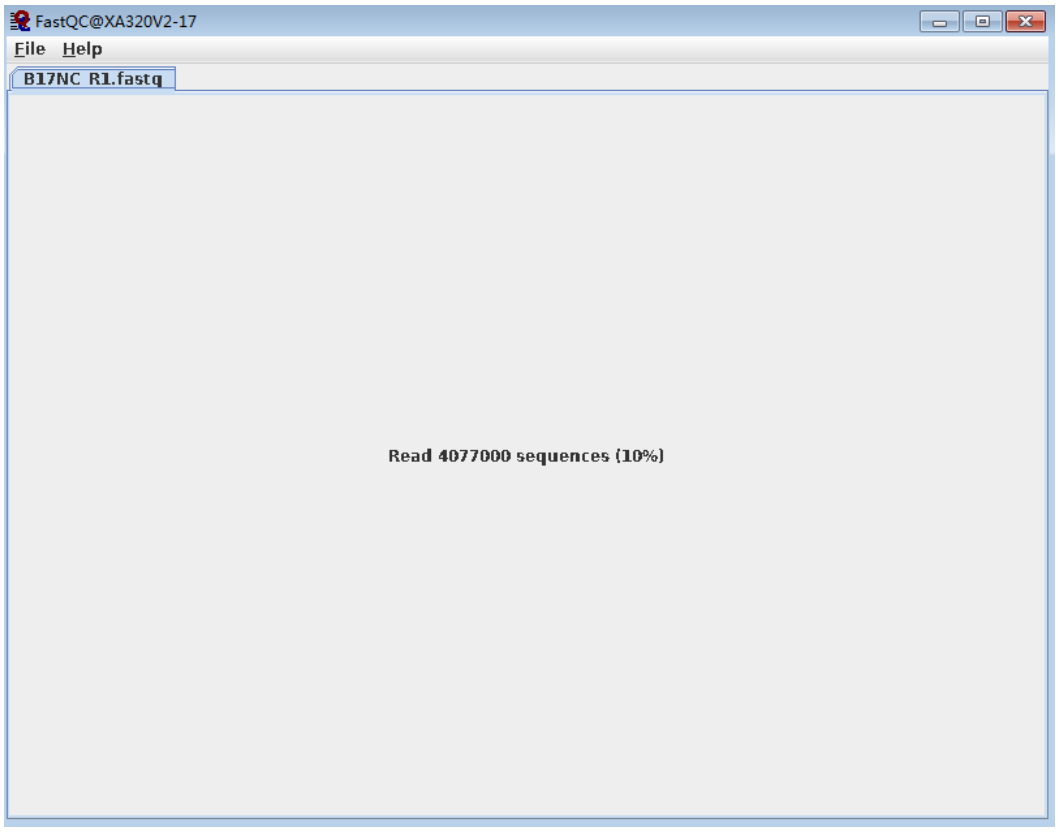
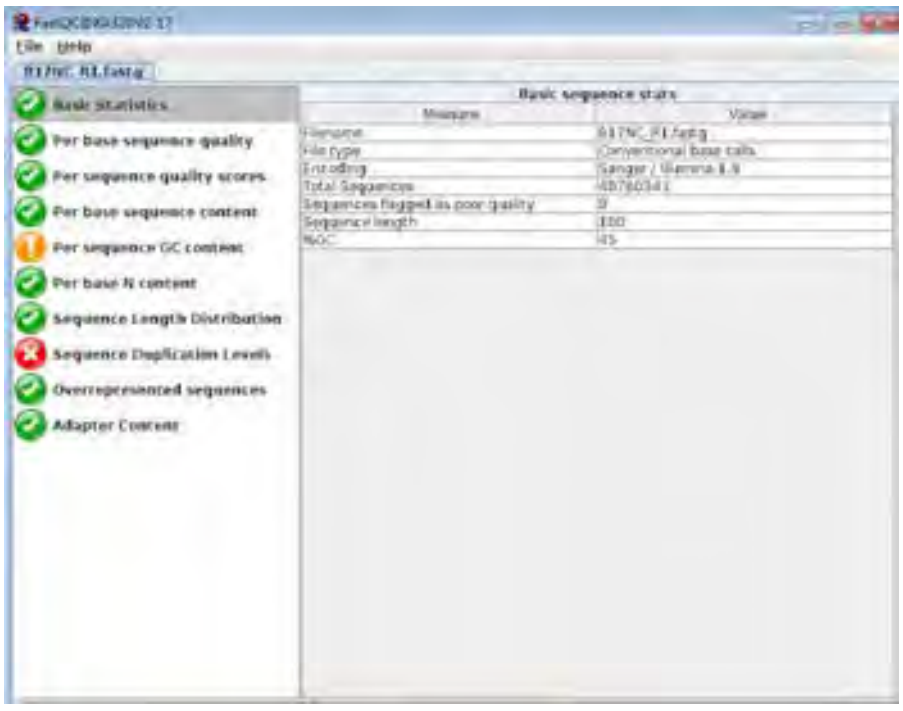


Figure 4-5 shows an example of the test result.

Figure 4-5 Result example



----End

 Copyright © Huawei Technologies Co., Ltd.

4.10.8 More Resources

FastQC installation guide:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/INSTALL.txt

4.11 Beagle 5.1 Porting Guide (CentOS 7.6)

4.11.1 Introduction

Beagle is one of the common software for genotype imputation. The latest version is 5.1, which greatly improves the accuracy and running speed. Compared with other software of the same type, Beagle has two obvious advantages: low memory consumption and fast running speed.

For more information, visit the **official Beagle website**.

Programming language: Java

Brief description: genotype imputation software

Open source license: GPL 3.0

Recommended Software Version

Beagle 5.1

4.11.2 Environment Requirements

Hardware Requirements

Table 4-52 lists the hardware requirements.

Table 4-52 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-53 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-53 Software requirements

Item	Version	Download Address	Remarks
Beagle	5.1	**https://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar**	Beagle tool package
bref3.27Apr20.b81.jar	-	**http://faculty.washington.edu/browning/beagle/bref3.27Apr20.b81.jar**	Tool package for converting the VCF format to the Bref3 format
unbref3.27Apr20.b81.jar	-	**http://faculty.washington.edu/browning/beagle/unbref3.27Apr20.b81.jar**	Tool package for converting the Bref3 format to the VCF format
1000 Genomes Project data	-	**http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz**	Test case file

OS Requirements

Table 4-54 lists the OS requirements.

Table 4-54 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.11.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Beagle software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-55 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
4	/path/to/BEAGLE	Installation path of Beagle	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.11.4 Configuring the Compilation Environment

4.11.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-56 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the Java environment	See **4.11.4.2 Setting Up the Java Environment**.

 Copyright © Huawei Technologies Co., Ltd.

4.11.4.2 Setting Up the Java Environment

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the Java package:

```
yum install java-1.8.0-openjdk.aarch64
```

Step 3 Run the following command to check whether Java has been installed:

```
java -version
```

If the following information is displayed, Java has been installed successfully:

```
openjdk version "1.8.0_212"
```

----End

4.11.5 Obtaining the Source Code

Procedure

Step 1 Download the Beagle package **beagle.27Apr20.b81.jar**.

https://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar

Step 2 Download the tool package **bref3.27Apr20.b81.jar** for converting VCF files to Bref3 files.

http://faculty.washington.edu/browning/beagle/bref3.27Apr20.b81.jar

Step 3 Download the tool package **unbref3.27Apr20.b81.jar** for converting Bref3 files to VCF files.

http://faculty.washington.edu/browning/beagle/unbref3.27Apr20.b81.jar

Step 4 Download the test file **test.27Apr20.b81.vcf.gz**.

http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz

Step 5 Use an SFTP tool to upload the downloaded installation package to the */path/to/BEAGLE* directory on the server.

----End

4.11.6 Running and Verifying Beagle

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the */path/to/BEAGLE* directory:

cd */path/to/BEAGLE*

Step 3 Run the following commands to create a script file and edit it:

1. Create a **test.sh** script file.
   ```
   vi test.sh
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and add the following content to the file:

```
#!/bin/sh
echo
echo "*** Creating test files: ref.27Apr20.b81.vcf.gz target.27Apr20.b81.vcf.gz ***"
echo
zcat test.27Apr20.b81.vcf.gz | cut -f1-190 | tr '/' '|' | gzip > ref.27Apr20.b81.vcf.gz
zcat test.27Apr20.b81.vcf.gz | cut -f1-9,191-200 | gzip > target.27Apr20.b81.vcf.gz

echo
echo "*** Running test analysis with \"gt=\" argument ***"
echo
java -jar beagle.27Apr20.b81.jar gt=test.27Apr20.b81.vcf.gz out=out.gt

echo
echo "*** Running test analysis with \"ref=\" and \"gt=\" arguments ***"
echo
java -jar beagle.27Apr20.b81.jar ref=ref.27Apr20.b81.vcf.gz gt=target.27Apr20.b81.vcf.gz out=out.ref

echo
echo "*** Making \"bref3\" file ***"
echo
java -jar bref3.27Apr20.b81.jar ref.27Apr20.b81.vcf.gz > ref.27Apr20.b81.bref3

echo
echo "*** Running test analysis with \"bref3\" file ***"
echo
java -jar beagle.27Apr20.b81.jar ref=ref.27Apr20.b81.bref3 gt=target.27Apr20.b81.vcf.gz out=out.bref3
```

NOTE

java -Xmx *[GB]* **g -jar beagle.jar** *[arguments]*

- In the preceding command, **[GB]** indicates the upper limit of the memory pool and **[arguments]** indicates the arguments separated by spaces.
- **[arguments]** must include the two parameters:
  - **gt = [file]** : specifies a VCF file that contains genotype data for the study sample.
  - **out = [character string]** : prefix of the output file name.
- For details about more parameters, see the official document at **http://faculty.washington.edu/browning/beagle/beagle_5.1_08Nov19.pdf**.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to grant the execute permission on the script:

```
chmod a+x test.sh
```

Step 5 Run the following command to execute the script:

```
./test.sh
```

The following is an example of the command output. The command output is also saved to the **.log** file.

```
Cumulative Statistics:

Study markers:          1,356

Haplotype phasing time:       1 second
Total time:                   1 second

End time: 10:33 AM CST on 09 May 2020
beagle.27Apr20.b81.jar finished
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.12 Bowtie 1.2.3 Porting Guide (CentOS 7.6)

4.12.1 Introduction

Bowtie is an ultrafast, memory-efficient short read aligner geared toward quickly aligning large sets of short DNA sequences (reads) to large genomes.

Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: for the human genome, the index is typically about 2.2 GB (for unpaired alignment) or 2.9 GB (for paired-end or colorspace alignment). Multiple processors can be used simultaneously to achieve greater alignment speed.

For more information, visit the **official Bowtie website**.

Programming language: C

Brief description: ultrafast, memory-efficient short read aligner, used in the next generation of sequencers for short DNA sequences

Open source license: GPL 3.0

Recommended Software Version

Bowtie 1.2.3

4.12.2 Environment Requirements

Hardware Requirements

Table 4-57 lists the hardware requirements.

Table 4-57 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-58 lists the software requirements.

Table 4-58 Software requirements

Item	Version	How to Obtain
Bowtie	1.2.3	**https://codeload.github.com/BenLangmead/bowtie/zip/v1.2.3**
simde	master	**https://codeload.github.com/nemequ/simde/zip/master**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
sse2neno	master	https://codeload.github.com/jratcliff63367/sse2neon/zip/master
chromFa	hg19	https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
Test case 1	SRR742200_1.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_1.fastq.gz
Test case 2	SRR742200_2.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_2.fastq.gz

OS Requirements

Table 4-59 lists the OS requirements.

Table 4-59 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.12.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Bowtie software porting.

Table 4-60 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BOWTIE	Installation path of Bowtie	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/SIMDE	Installation path of SIMDE	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/SSE2NENO	Installation path of sse2neno	
5	/path/to/CHROMFA	Installation path of chromFa	

4.12.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-61 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.12.5 Obtaining the Source Code

Procedure

Step 1 Download the following software packages and case files.

- Download the Bowtie installation package **bowtie-1.2.3.zip**.
  https://github.com/BenLangmead/bowtie/archive/v1.2.3.zip
- Download the simde-master installation package **simde-master.zip**.
  https://codeload.github.com/nemequ/simde/zip/master
- Download the sse2neon-master installation package **sse2neon-master.zip**.
  https://codeload.github.com/jratcliff63367/sse2neon/zip/master
- Download the human genome package **chromFa.tar.gz**.
  https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
- Download the **SRR742200_1.fastq** test case file **SRR742200_1.fastq.gz** from:
  ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_1.fastq.gz

 Copyright © Huawei Technologies Co., Ltd.

- Download the **SRR742200_2.fastq** test case file **SRR742200_2.fastq.gz** from:
  ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_2.fastq.gz

Step 2 Upload the software packages and case files to the server using an SFTP tool.

- Upload the Bowtie installation package to the */path/to/BOWTIE* directory on the server.
- Upload the simde-master installation package to the */path/to/SIMDE* directory on the server.
- Upload the sse2neon-master installation package to the */path/to/SSE2NENO* directory on the server.
- Upload the **chromFa.tar.gz** installation package to the */path/to/CHROMFA* directory on the server.
- Upload the **SRR742200_1.fastq.gz** file to the */path/to/CHROMFA/man_hg19/* directory on the server.
- Upload the **SRR742200_2.fastq.gz** file to the */path/to/CHROMFA/man_hg19/* directory on the server.

----End

4.12.6 Compiling and Installing Bowtie

4.12.6.1 Installing Bowtie

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install dependencies:

```
yum install zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64
```

Step 3 Run the following command to decompress the installation package:

```
unzip bowtie-1.2.3.zip
```

Step 4 Run the following command to switch to the directory that contains decompressed files:

```
cd /path/to/BOWTIE/bowtie-1.2.3
```

Step 5 Run the following commands to modify the configuration file **Makefile**:

1. Open **Makefile**.

   ```
   vi Makefile
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   To support AArch64, add the following information in bold to the end of about line 162:

   ```
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ifeq (aarch64,$(shell uname -m))
   BITS=64
   endif
   ```

   Modify the following content in about line 187:

 Copyright © Huawei Technologies Co., Ltd.

```
DEBUG_FLAGS = -O0 -g3
RELEASE_FLAGS = -O3
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

Add the following content in bold after the commented lines to add the AArch64 branch:

```
M64_FLAGS := -m64
ifeq (aarch64,$(shell uname -m))
M64_FLAG :=
endif
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to modify the **alphabet.cpp** file:

1. Open **alphabet.cpp**.
   vi */path/to/BOWTIE/***bowtie-1.2.3/alphabet.cpp**
2. Press **i** to enter the insert mode and modify line 276 of the file.

   Before the modification:

   ```
   char mask2iupac[16] = {...}
   ```

   After the modification:

   ```
   signed char mask2iupac[16] = {...}
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to create a folder:

```
mkdir -p ./third_party/simde/simde/x86
```

Step 8 Run the following command to perform the configuration:

```
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

Step 9 Run the following command to decompress the SIMDE installation package:

```
unzip simde-master.zip
```

Step 10 Run the following command to copy the **simde** directory in **simde-master** to **bowtie-1.2.3/third_party/simde/**:

```
cp -a  simde-master/simde  /path/to/BOWTIE/bowtie-1.2.3/third_party/simde/
```

Step 11 Run the following command to decompress the sse2neno installation package:

```
unzip sse2neon-master.zip
```

Step 12 Run the following command to copy the **SSE2NEON.h** file to **bowtie-1.2.3/third_party/simde/simde/x86/**:

```
cp -a sse2neon-master/SSE2NEON.h  /path/to/BOWTIE/bowtie-1.2.3/third_party/simde/simde/x86/
```

Step 13 Run the following command to perform the compilation:

```
cd /path/to/BOWTIE/bowtie-1.2.3
make
```

Step 14 Run the following command to perform the installation:

```
make install
```

Step 15 Run the following command to set environment variables:

export PATH=*/path/to/BOWTIE***/bowtie-1.2.3:$PATH**

----End

 Copyright © Huawei Technologies Co., Ltd.

4.12.6.2 Creating a Bowtie Index

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the **CHROMFA** directory:

```
cd /path/to/CHROMFA
```

Step 3 Run the following command to decompress the chromFa installation package:

```
tar -xzvf chromFa.tar.gz
```

Step 4 Run the following command to perform the redirection:

```
cat *.fa > man.fa
```

Step 5 Run the following command to create an index:

```
bowtie-build --threads 128 man.fa man_index
```

Step 6 Run the following commands to create a **man_hg19** directory and save the generated **man_index*.ebwt** file to this directory:

```
mkdir man_hg19
mv  *.ebwt  man_hg19
```

----End

4.12.7 Running and Verifying Bowtie

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the **man_hg19** directory:

```
cd /path/to/CHROMFA/man_hg19/
```

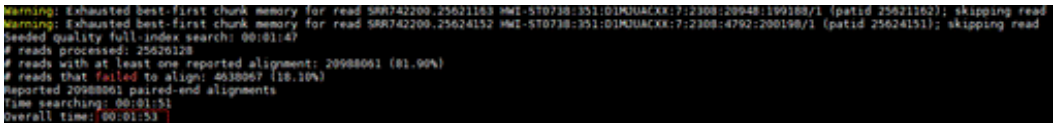
Step 3 Run the following commands to decompress the case files:

```
gzip -d SRR742200_1.fastq.gz
gzip -d SRR742200_2.fastq.gz
```

Step 4 Run the following command to run Bowtie to obtain the **.sam** file (the corresponding biological sequence text **xxx.fastq** is required):

```
bowtie -S -t -q -p 128 man_index -1 ./SRR742200_1.fastq -2 ./SRR742200_2.fastq SRR742200_out.sam
```

The following is an example of the result:



----End

4.12.8 Troubleshooting

Failure 1: An Error Occurs During the Bowtie Installation

Symptom

When running the **make** command to perform compilation during Bowtie installation, an error occurs. The error message is "narrowing conversion of '-1' from 'int' to 'char' inside { } [-Wnarrowing]".

Possible Causes

There is a narrowing conversion error.

Procedure

Run the following commands to modify the **alphabet.cpp** file:

1. Open **alphabet.cpp**.
   vi */path/to/BOWTIE/***bowtie-1.2.3/alphabet.cpp**
2. Press **i** to enter the insert mode and modify line 276 of the file.

   Before the modification:
   ```
   char mask2iupac[16] = {...}
   ```
   After the modification:
   ```
   signed char mask2iupac[16] = {...}
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

4.12.9 More Information

Bowtie installation guide:

https://bbs.huaweicloud.com/forum/thread-47080-1-1.html?

Download **chromFa.tar.gz**:

https:/hgdownload.soe.ucsc.edu/downloads.html#mm9pairwise-primate

4.13 pblat 2.1 Porting Guide (CentOS 7.6)

4.13.1 Introduction

For DNA sequences, BLAST-like alignment tool (blat) is used to design and find sequences of 95% and greater similarity of length 40 bases or more. For protein sequences, blat is used to design and find sequences of 80% and greater similarity of length 20 amino acids or more. pblat is a multi-threaded blat algorithm that fully utilizes the advantages of multi-core processors and greatly reduces the running time with the same memory consumption.

For more information, visit the **official pblat website**.

Programming language: C

Brief description: A BLAST-like alignment tool that supports multiple threads.

Recommended Software Version

pblat 2.1

 Copyright © Huawei Technologies Co., Ltd.

4.13.2 Environment Requirements

Hardware Requirements

Table 4-62 lists the hardware requirements.

Table 4-62 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-63 lists the software requirements.

Table 4-63 Software requirements

Item	Version	Download Address
pblat	2.1	https://codeload.github.com/icebert/pblat/tar.gz/2.1
Test case	hg19.fa	https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/open_database/hg19/gatk_bundle/hg19.fa
	NONCODEv5_human.fa	http://www.noncode.org/datadownload/NONCODEv5_human.fa.gz

OS Requirements

Table 4-64 lists the OS requirements.

Table 4-64 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.13.3 Paths for Software Porting

This chapter lists the software installation paths involved in the pblat software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-65 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PBLAT	Installation path of pblat	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-66 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.13.5 Obtaining the Source Code

Procedure

Step 1 Download the pblat installation package **pblat-2.1.tar.gz**.

https://codeload.github.com/icebert/pblat/tar.gz/2.1

Step 2 Use an SFTP tool to upload the pblat installation package to the */path/to/PBLAT* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.13.6 Compiling and Installing pblat

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the pblat installation package:

```
tar -zxvf pblat-2.1.tar.gz
```

Step 3 Run the following command to switch to the directory that contains decompressed files:

```
cd pblat-2.1/
```

Step 4 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   MACHTYPE=aarch64
   CC=gcc
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following commands to modify the **htmshell.c** file:

1. Open **htmshell.c**.
   ```
   vi lib/htmshell.c
   ```
2. Run the following command to locate line 714:
   ```
   :714
   ```
3. Press **i** to enter the insert mode and modify line 714 of the file.
   ```
   if (format != NULL)
   ```
4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to compile pblat:

```
make -j16
```

Step 7 Run the following command to move the executable program of pblat to the planned path:

```
cp pblat /path/to/PBLAT
```

Step 8 Run the following command to add environment variables:

```
export PATH=/path/to/PBLAT:$PATH
```

----End

4.13.7 Running and Verifying pblat

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create a test file directory and copy the test files:

```
mkdir /path/to/PBLAT/test
cp hg19.fa /path/to/PBLAT/test
cp NONCODEv5_human.fa.gz /path/to/PBLAT/test
```

Step 3 Run the following command to go to the test case directory:

```
cd /path/to/PBLAT/test
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to process the test case files:

```
gzip -d NONCODEv5_human.fa.gz
cat NONCODEv5_human.fa | grep -v "^#" | head -799996 > test-799996.fa
```

Step 5 Run the following command to perform a test:

```
{ time pblat -threads=96 hg19.fa test-799996.fa out.psl ; } 2>&1 |tee pblat-kunpeng.log
```

Check the **real** value (in seconds) in the **pblat-kunpeng.log** file. A smaller value indicates higher performance.

Figure 4-6 is an example of the output result.

Figure 4-6 Result example

```
Searched 251927 bases in 260 sequences
Searched 246170 bases in 260 sequences
Searched 314326 bases in 260 sequences
Searched 267271 bases in 260 sequences
Searched 265877 bases in 260 sequences
Searched 352678 bases in 260 sequences
Searched 374791 bases in 260 sequences
Searched 277221 bases in 260 sequences
Searched 304065 bases in 260 sequences
Searched 382588 bases in 260 sequences
Searched 405694 bases in 260 sequences
real 1539.22
user 42670.01
sys 10.29
```

----End

4.13.8 Troubleshooting

Problem 1: An Error Is Reported When the make Command Is Executed

Symptom

An error message " error: invalid operands to binary != (have'va_list '{aka' __va_list '} and' void * ')" is displayed when the make command is executed.

Possible Cause

The source code programming language is not standard.

Procedure

Modify line 714 in the **htmshell.c** file by following the instructions provided in Step 5.

4.13.9 More Resources

pblat installation and running guide:

https://github.com/icebert/pblat

4.14 SnpEff V4_3 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.14.1 Introduction

SnpEff is a variant annotation and effect prediction tool. It annotates and predicts the effects of genetic variants (such as amino acid changes).

For more information, visit the **official SnpEff website**.

Programming language: Java

Brief description: A variant annotation and effect prediction tool

Recommended Software Version

SnpEff V4.3

4.14.2 Environment Requirements

Hardware Requirements

Table 4-67 lists the hardware requirements.

Table 4-67 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-68 lists the software requirements.

Table 4-68 Software requirements

Item	Version	Download Address
SnpEff	V4.3	**https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip**
GRCh 37.75 database	37.75	**https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip**

OS Requirements

Table 4-69 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-69 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.14.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SnpEff software porting.

Table 4-70 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/SNPEFF	Installation path of SnpEff	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.14.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-71 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.14.5 Obtaining the Source Code

Procedure

Step 1 Download the SnpEff installation package **snpEff_v4_3_core.zip**.

https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip

Step 2 Download the GRCh37.75 installation package **snpEff_v4_3_GRCh37.75.zip**.

https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip

Step 3 Use an SFTP tool to upload the SnpEff and GRCh37.75 installation packages to the */path/to/SNPEFF* directory on the server.

----End

4.14.6 Compiling and Installing SnpEff

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the Java environment:

```
yum install java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Run the following command to switch to the SnpEff directory:

```
cd /path/to/SNPEFF
```

Step 4 Run the following command to decompress the installation package:

```
unzip snpEff_v4_3_core.zip
```

Step 5 Run the following command to copy the **GRCh37.75** file:

```
cp snpEff_v4_3_GRCh37.75.zip /path/to/SNPEFF/snpEff
```

Step 6 Run the following command to switch to the SnpEff directory:

```
cd /path/to/SNPEFF/snpEff
```

Step 7 Run the following command to decompress the installation package:

```
unzip snpEff_v4_3_GRCh37.75.zip
```

----End

4.14.7 Running and Verifying SnpEff

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to list the databases (genomes) supported by SnpEff:

```
java -jar snpEff.jar databases
```

Step 3 Run the following command to switch to the SnpEff directory:

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/SNPEFF/snpEff
```

Step 4 Run the following command to export the text that includes comments:

```
java -jar snpEff.jar -v GRCh37.75 /path/to/SNPEFF/snpEff/examples/test.chr22.vcf > test.chr22.out.vcf
```

After the execution is complete, three files are generated: **snpEff_genes.txt**, **snpEff_summary.html**, and **test.chr22.out.vcf**.

The following is an output example:

```
00:00:59        Creating summary file: snpEff_summary.html
00:00:59        Creating genes file: snpEff_genes.txt
00:00:59        done.
00:00:59        Logging
00:01:00        Checking for updates...
[root@           snpEff]# ll
total 62768
drwxr-xr-x 5 root root       52 May 25 09:58 data
drwxr-xr-x 2 root root     4096 May 25 18:16 examples
drwxr-xr-x 3 root root      275 Jul  3  2016 galaxy
drwxr-xr-x 3 root root     4096 Jul  3  2016 scripts
-rw-r--r-- 1 root root  6138511 Jul  3  2016 snpEff.config
-rw-r--r-- 1 root root   401190 May 26 09:12 snpEff_genes.txt
-rw-r--r-- 1 root root 21209595 Jul  3  2016 snpEff.jar
-rw-r--r-- 1 root root   323525 May 26 09:12 snpEff_summary.html
-rw-r--r-- 1 root root 18756269 Jul  3  2016 SnpSift.jar
-rw-r--r-- 1 root root 17427440 May 26 09:12 test.chr22.out.vcf
```

----End

4.14.8 More Resources

SnpEff installation guide

http://snpeff.sourceforge.net/download.html#install

More databases:

https://sourceforge.net/projects/snpeff/files/databases/

4.15 Velvet 1.2.10 Porting Guide (CentOS 7.6)

4.15.1 Introduction

Velvet is a de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454, developed by Daniel Zerbino and Ewan Birney at the European Bioinformatics Institute (EMBL-EBI). Data in multiple libraries can be used at the same time.

For more information, visit the **official Velvet website**.

Programming language: C

Brief description: A common genome assembly software, which is used to process second-generation sequencing data and has good assembly integrity.

Open source license: GPL 3.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

Velvet 1.2.10

4.15.2 Environment Requirements

Hardware Requirements

Table 4-72 lists the hardware requirements.

Table 4-72 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-73 lists the software requirements.

Table 4-73 Software requirements

Item	Version	Download Address
Velvet	1.2.10	https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10

OS Requirements

Table 4-74 lists the OS requirements.

Table 4-74 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.15.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Velvet software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-75 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/VELVET	Installation path of Velvet	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.15.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-76 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.15.5 Obtaining the Source Code

Procedure

Step 1 Download the Velvet installation package **v1.2.10.tar.gz**.

https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10

Step 2 Use an SFTP tool to upload the Velvet installation package to the */path/to/Velvet* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.15.6 Compiling and Installing Velvet

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/VELVET
```

Step 3 Run the following command to copy the installation package to the main program installation directory:

```
cp velvet_1.2.10.tgz /path/to/VELVET
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/VELVET
```

Step 5 Run the following command to decompress the installation package:

```
tar -xvf velvet-1.2.10.tar.gz
```

Step 6 Run the following command to go to the directory generated after the decompression:

```
cd velvet-1.2.10
```

Step 7 Run the following commands to set the number of threads and the thread limit:

```
export OMP_NUM_THREADS=96
export OMP_THREAD_LIMIT=95
```

NOTE

- **OMP_NUM_THREADS**: specifies the number of threads, which is equal to the number of CPU cores.
- **OMP_THREAD_LIMIT**: specifies the thread limit, which is equal to the number of CPU cores minus 1.

Step 8 Run the following command to modify the **Makefile** file:

```
sed -ri 's/\-m64/\-mabi=lp64/g' Makefile
```

Step 9 Run the following command to compile and install the software:

```
make 'GATEGORIES=10' 'MAXKMERLENGTH=57' 'LONGSEQUENCES=1' 'OPENMP=1' 'BUNDLEDZLIB=1'
```

After the compilation is complete, two executable files are generated: **velveth** and **velvetg**.

----End

4.15.7 Running and Verifying Velvet

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the test directory:

```
cd /path/to/VELVET/velvet-1.2.10
```

Step 3 Run the following command to run the executable file **velveth** to prepare data:

```
./velveth output/ 31 -shortPaired -fastq.gz tests/read1.fq.gz -shortPaired -fastq.gz tests/read2.fq.gz
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

- **output**: specifies the output directory.
- **shortPaired**: specifies the sequencing type.
- **Fastaq**: specifies the sequence format. Different parameters are used to specify the format of the input file. **-fasta** corresponds to the FASTA format. **-fastq** corresponds to the FASTQ format. **-fastq.gz** corresponds to the fastq.gz format. **-fasta.gz** corresponds to the fasta.gz format. **-sam**: corresponds to the sam format. **-bam** corresponds to the bam format.

The following is an example of the command output:

```
[0.000000] Reading FastQ file tests/read1.fq.gz;
[0.083994] 25000 sequences found
[0.084004] Done
[0.084266] Reading FastQ file tests/read2.fq.gz;
[0.162083] 25000 sequences found
[0.162091] Done
[0.364858] Reading read set file output//Sequences;
[0.376048] 50000 sequences found
[0.427913] Done
[0.427929] 50000 sequences in total.
[0.428268] Writing into roadmap file output//Roadmaps...
[0.468365] Inputting sequences...
[0.468486] Inputting sequence 0 / 50000
[1.182309]  === Sequences loaded in 0.713954 s
[1.303762] Done inputting sequences
[1.303770] Destroying splay table
[1.305914] Splay table destroyed
```

Step 4 Run the following command to run the executable file **velvetg** to assemble the genome:

```
./velvetg output/ -min_contig_lgth 100
```

NOTE

- **output**: specifies the output directory of **Step 3**.
- **min_contig_lgth**: specifies the minimum length of the Contig field. A Contig field whose length is less than the minimum length will be deleted and will not be displayed in the final result. After the running is complete, the **contigs.fa** file in the output directory is the final assembly result.

The following is an example of the command output:

```
[1.258599] Concatenation over!
[1.259034] Writing contigs into output//contigs.fa...
[1.292427] Writing into stats file output//stats.txt...
[1.327635] Writing into graph file output//LastGraph...
Final graph has 987 nodes and n50 of 199, max 2546, total 110172, using 0/50000 reads
```

----End

4.15.8 More Resources

Information on GitHub:

https://github.com/dzerbino/velvet/tree/master

4.16 XHMM Porting Guide (CentOS 7.6)

4.16.1 Introduction

XHMM uses principal component analysis (PCA) normalization and a hidden Markov model (HMM) to detect and genotype copy number variation (CNV) from normalized read-depth data from targeted sequencing experiments. XHMM was explicitly designed to be used with targeted exome sequencing at high coverage (at least 60x - 100x) using Illumina HiSeq (or similar) sequencing of at least ~50 samples.

For more information, visit the **official XHMM website**.

Programming language: C++

Brief description: tool for predicting copy number variation (CNV).

Open source license: GPL v3

4.16.2 Environment Requirements

Hardware Requirements

Table 4-77 lists the hardware requirements.

Table 4-77 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-78 lists the software requirements.

Table 4-78 Software requirements

Item	Version	Download Address
XHMM	Latest XHMM development version	**https://bitbucket.org/statgen/xhmm/get/master.zip**
LAPACK	v3.8.0	**https://github.com/Reference-LAPACK/lapack/archive/v3.8.0.tar.gz**
OpenBLAS	v0.3.9	**https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz**
Test case	POST_GATK_DATA	**https://statgen.bitbucket.io/xhmm/POST_GATK_DATA.zip**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-79 lists the OS requirements.

Table 4-79 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.16.3 Paths for Software Porting

This chapter lists the software installation paths involved in the XHMM software porting.

Table 4-80 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/OPENBLAS	Installation path of OpenBLAS	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/LAPACK	Installation path of LAPACK	
4	/path/to/XHMM	Installation path of XHMM	
5	/path/to/TEST	Installation path of the XHMM test case	

4.16.4 Configuring the Compilation Environment

4.16.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-81 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing OpenBLAS	See **4.16.4.2 Installing OpenBLAS**.
3	Installing LAPACK	See **4.16.4.3 Installing LAPACK**.

4.16.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

```
tar xzvf OpenBLAS-0.3.9.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd OpenBLAS-0.3.9/
```

Step 4 Run the following commands to set environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform the installation and compilation:

```
make -j 96
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following commands to set the OpenBLAS environment variables:

```
export LIBRARY_PATH=/path/to/OPENBLAS/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

4.16.4.3 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

```
tar xzvf lapack-3.8.0.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd lapack-3.8.0/
```

Step 4 Run the following command to configure the compilation file:

```
cp make.inc.example make.inc
```

Step 5 Run the following commands to perform the installation and compilation:

```
make lapacklib -j96
mkdir /path/to/LAPACK
```

Step 6 Run the following command to set the LAPACK environment variables:

```
export LIBRARY_PATH=/path/to/LAPACK:$LIBRARY_PATH
```

----End

4.16.5 Obtaining the Source Code

Procedure

Step 1 Download the following installation packages:

- Download the XHMM source code package **statgen-xhmm-998f7c405974.zip**.

  https://bitbucket.org/statgen/xhmm/get/master.zip
- Download the LAPACK source code package **lapack-3.8.0.tar.gz**.

  https://github.com/Reference-LAPACK/lapack/archive/v3.8.0.tar.gz
- Download the OpenBLAS source code package **OpenBLAS-0.3.9.tar.gz**.

  https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz
- Download the XHMM test case **POST_GATK_DATA.zip**.

  https://statgen.bitbucket.io/xhmm/POST_GATK_DATA.zip

Step 2 Use an SFTP tool to upload the following installation packages to the server.

- Upload the XHMM source code package and XHMM test cases to the */path/to/XHMM* directory on the server.
- Upload the LAPACK source code package to the */path/to/LAPACK* directory on the server.
- Upload the OpenBLAS source code package to the */path/to/OPENBLAS* directory on the server.

----End

4.16.6 Compiling and Installing XHMM

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

```
unzip statgen-xhmm-*.zip
```

Step 3 Run the following command to switch to the directory generated after decompression:

 Copyright © Huawei Technologies Co., Ltd.

```
cd statgen-xhmm-*
```

Step 4 Run the following command to edit the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify line 89:

   Before the modification:
   ```
   LAPACK_LIBS=lapack
   ```
   After the modification:
   ```
   LAPACK_LIBS=openblas lapack
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following commands to perform the compilation and installation:

```
make -j 96
mkdir /path/to/XHMM
cp build/execs/xhmm params.txt /path/to/XHMM
```

Step 6 Run the following command to add the XHMM installation path to the **PATH** environment variable:

```
export PATH=/path/to/XHMM:$PATH
```

----End

4.16.7 Running and Verifying XHMM

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Download the test case package **POST_GATK_DATA.zip**.

Step 3 Use an SFTP tool to upload the test case package to the */path/to/TEST* directory on the server.

Step 4 Run the following commands to decompress the test case package:

```
cd /path/to/TEST
unzip POST_GATK_DATA.zip
```

Step 5 Run the following commands to merge multiple samples (at the same point) output by GATK depths using XHMM:

```
xhmm --mergeGATKdepths -o ./DATA.RD.txt \
--GATKdepths group1.DATA.sample_interval_summary \
--GATKdepths group2.DATA.sample_interval_summary \
--GATKdepths group3.DATA.sample_interval_summary
```

The execution result is as follows:

```
%%%
```

```
Reading GATK depth file group1.DATA.sample_interval_summary
Reading GATK depth file group2.DATA.sample_interval_summary
Reading GATK depth file group3.DATA.sample_interval_summary
Writing GATK read-depth matrix of 30 samples by 300 targets to ./DATA.RD.txt

Total CPU time for processing this job: 6 seconds
[user time: 6
system time: 0
children user time: 0
children system time: 0]
Total time for processing this job: 0 seconds
```

Generated file:

```
DATA.RD.txt
```

----End

4.17 pysam 0.15.4 Porting Guide (CentOS 7.6)

4.17.1 Introduction

Pysam is a Python module used to read and operate files in SAM, BAM, VCF, and BCF formats. It is a lightweight wrapper of the htslib C-API and provides support for SAMtools, BCFtools, and tabix.

For more information, visit the **official Pysam website**.

Programming language: Python/Cython

Open source license: MIT License

Brief description: A Python module used to read and operate SAM/BAM/VCF/BCF files.

Recommended Software Version

Pysam v0.15.4

4.17.2 Environment Requirements

Hardware Requirements

Table 4-82 lists the hardware requirements.

Table 4-82 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-83 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-83 Software requirements

Item	Version	Download Address
Pysam	v0.15.4	https://github.com/pysam-developers/pysam/archive/v0.15.4.tar.gz
Python	3.8.2	https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tar.xz
Cython	0.29.19	https://files.pythonhosted.org/packages/79/36/69246177114d0b6cb7bd4f9aef177b434c0f4a767e05201b373e8c8d7092/Cython-0.29.19.tar.gz

OS Requirements

Table 4-84 lists the OS requirements.

Table 4-84 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.17.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Pysam software porting.

Table 4-85 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PYSAM	Installation path of Pysam	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PYTHON	Installation path of Python	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/CYTHON	Installation path of Cython	

4.17.4 Configuring the Compilation Environment

4.17.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-86 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the Python interpreter	See **4.1**.
3	Installing the Cython package	See **4.2**.

4.17.4.2 Installing the Python Interpreter

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xf Python-3.8.2.tar.xz
```

Step 3 Go to the directory generated after the decompression.

```
cd Python-3.8.2
```

Step 4 Create a **Makefile** file.

```
./configure --prefix=/path/to/PYTHON
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the compilation and installation.

```
make -j 96
make install
```

Step 6 Set the Python environment variables.

```
export PATH=/path/to/PYTHON/bin:$PATH
```

----End

4.17.4.3 Installing the Cython Package

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xzvf Cython-0.29.19.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd Cython-0.29.19/
```

Step 4 Compile Cython.

```
python3 setup.py build
```

Step 5 Install Cython.

```
python3 setup.py install
```

----End

4.17.5 Obtaining the Source Code

Procedure

Step 1 Download the Pysam source code package **pysam-0.15.4.tar.gz**.

https://github.com/pysam-developers/pysam/archive/v0.15.4.tar.gz

Step 2 Use an SFTP tool to upload the Pysam source code package to the */path/to/PYSAM* directory on the server.

----End

4.17.6 Compiling and Installing Pysam

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xzvf pysam-0.15.4.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pysam-0.15.4/
```

Step 4 Compile Pysam.

```
python3 setup.py build
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Install Pysam.

```
python3 setup.py install
```

----End

4.17.7 Running and Verifying Pysam

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **pysam-0.15.4/tests/pysam_data/** folder in the Pysam source code directory and locate **example_aligned_pairs.sam**.

```
cd tests/pysam_data/
```

Step 3 Run Pysam.

```
python3
import pysam
samfile = pysam.AlignmentFile("example_aligned_pairs.sam")
samfile.count()
```

In the command output, **52** indicates that the Pysam has successfully parsed the **sam** file.

```
Python 3.8.2 (default, May 23 2020, 16:05:37)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pysam
>>> samfile = pysam.AlignmentFile("example_aligned_pairs.sam")
>>> samfile.count()
52
```

----End

4.17.8 Troubleshooting

Problem 1: Error Reported During Python 3.8.2 Installation and Compilation

Symptom

During the installation and compilation of Python 3.8.2, the error message "ModuleNotFoundError: No module named '_ctypes'" is displayed.

Possible Causes

CentOS 7 does not have the development link library software package of the external function library (libffi).

Procedure

Before installing and compiling Python 3.8.2, run the **yum install libffi-devel -y** command to install the libffi-devel package.

Problem 2: Error Reported During Configuration

Symptom

During the configuration process, the error message "configure: error: liblzma development files not found" is displayed.

 Copyright © Huawei Technologies Co., Ltd.

Possible Causes

The dependency package **xz-devel** is missing.

Procedure

Before installing and compiling Python 3.8.2, run the **yum install xz-devel -y** command to install the **xz-devel** package.

4.17.9 More Information

https://pysam.readthedocs.io/en/latest/api.html#

4.18 Abyss 2.2.4 Porting Guide (CentOS 7.6)

4.18.1 Introduction

Abyss is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes. Mainstream NGS genome assembly software divides sequences into k-mers and then obtains assembled sequences based on the de Bruijn Graph algorithm. When the program is running, the k-mer strings are stored in the memory. Therefore, the memory of the computer must be large enough. Abyss is based on the Bloom filter. Its advantage is that it can perform parallel operations and run multiple stitching tasks at the same time. Therefore, Abyss processes more genes than NGS.

For more information, visit the **official Abyss website**.

Programming language: C

Brief description: a gene sequence assembler.

Open source license: GPL

Recommended Software Version

Abyss 2.2.4

4.18.2 Environment Requirements

Hardware Requirements

Table 4-87 lists the hardware requirements.

Table 4-87 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 4-88 Software requirements

Item	Version	Download Address
Abyss	2.2.4	https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz
Boost	1.56.0	http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2
test-data	Test case	https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz

OS Requirements

Table 4-89 lists the OS requirements.

Table 4-89 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.18.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Abyss software porting.

Table 4-90 Paths for software porting

No.	Software Installation Path	Usage	Description
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/ABYSS	Installation path of Abyss	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the
3	/path/to/BOOST	Installation path of Boost	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Description
3	/path/to/TESTCASE	Installation path of Abyss	actual paths planned during the installation process.

4.18.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-91 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.18.5 Obtaining the Source Code

Procedure

Step 1 Download the Abyss installation package **abyss-2.2.4.tar.gz**.

https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz

Step 2 Download the dependency package **boost_1_56_0.tar.bz2**.

http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2

Step 3 Download the test case package **test-data.tar.gz**.

https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz

Step 4 Use an SFTP tool to upload the Abyss installation package to the */path/to/ABYSS* directory on the server.

Step 5 Use an SFTP tool to upload the Boost dependency package to the */path/to/BOOST* directory on the server.

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Use an SFTP tool to upload the Abyss test case package to the */path/to/TESTCASE* directory on the server.

----End

4.18.6 Compiling and Installing Abyss

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the installation package:

```
tar -zxvf abyss-2.2.4.tar.gz
```

Step 3 Run the following commands to decompress the dependency package:

```
cd /path/to/BOOST
tar jxf boost_1_56_0.tar.bz2
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd /path/to/ABYSS/abyss-2.2.4
```

Step 5 Run the following command to generate a **configure** file:

```
./autogen.sh
```

Step 6 Run the following command to add the installation path of the currently running MPI to the temporary environment variables:

```
export CURRENT_MPI_PATH=`which mpirun | sed 's/\/bin\/mpirun//g'`
```

Step 7 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/ABYSS --with-boost=/path/to/BOOST/boost_1_56_0 --without-sparsehash --with-mpi=$CURRENT_MPI_PATH
```

Step 8 Run the following commands to perform the installation and compilation:

```
make -j 32 AM_CXXFLAGS=-Wall
make install AM_CXXFLAGS=-Wall
```

Step 9 Run the following commands to add the environment variables:

```
export PATH=/path/to/ABYSS/bin:$PATH
```

----End

4.18.7 Running and Verifying Abyss

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the test case package:

```
tar xvf test-data.tar.gz
```

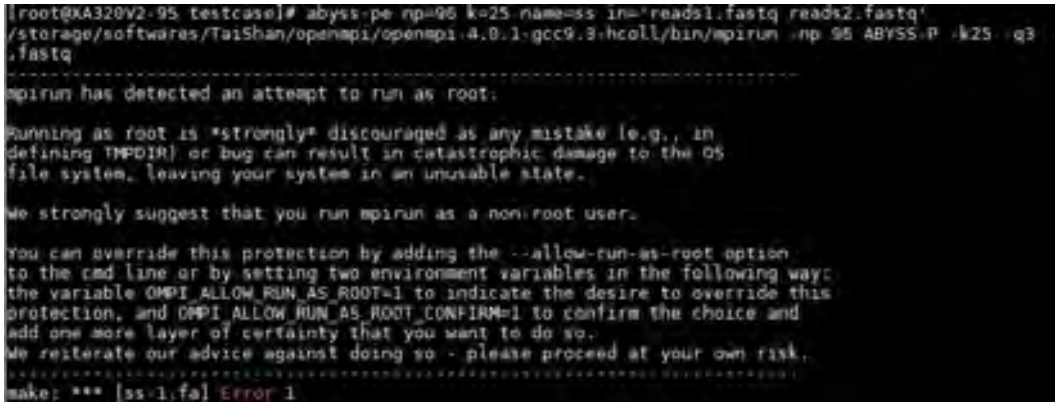
Step 3 Run the following commands to switch the directory and perform the assembling:

```
cd /path/to/TESTCASE
abyss-pe k=25 name=test in='reads1.fastq reads2.fastq'
```

Step 4 After the preceding steps are performed, only a single process can be executed. If multiple processes are executed, an error is reported, as shown in the following

 Copyright © Huawei Technologies Co., Ltd.

figure. To execute multiple processes concurrently, perform the following steps to modify the file.



1. Run the following command to switch to the Abyss installation path and find the **abyss-pe** file:
   cd */path/to/ABYSS/***bin***
2. Run the following command to modify the **abyss-pe** file:
   a. Open **abyss-pe**.
      vi abyss-pe
   b. Press **i** to enter the insert mode and add **--allow-run-as-root -mca coll_hcoll_enable 0** to lines 543 and 551 of the **abyss-pe** file.
      ```
      $(gtime) $(mpirun) --allow-run-as-root -mca coll_hcoll_enable 0 -np $(np) abyss-paired-dbg-mpi $(abyssopt) $(ABYSS_OPTIONS) -o $*-1.fa $(in) $(se)
      $(gtime) $(mpirun) --allow-run-as-root -mca coll_hcoll_enable 0 -np $(np) ABYSS-P $(abyssopt) $(ABYSS_OPTIONS) -o $@ $(in) $(se)
      ```
   c. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
3. Run the following command to switch the directory and perform the assembling:
   cd */path/to/TESTCASE/*

   NOTE

   The **name** parameter indicates the prefix of the generated file. After the execution is complete, many files are generated, among which **test-contigs.fa** and **test-scaffolds.fa** correspond to the **contig** and **scaffold** results, respectively.

   The following command output is displayed:

```
n       n:1000  L50     min     N80     N50     N20     E-size  max     sum     name
756     19      6       1315    1752    3094    4947    3141    6122    47778   n=10 s=1000
757     20      7       1315    1700    2214    4947    2991    6122    47778   n=10 s=2000
759     22      9       1315    1700    2205    3191    2385    4066    47789   n=10 s=5000
759     22      9       1315    1700    2205    3191    2385    4066    47789   n=10 s=10000

Best scaffold N50 is 3094 at n=10 s=1000.

n       n:1000  L50     min     N80     N50     N20     E-size  max     sum     name
756     19      6       1315    1752    3094    4947    3141    6122    47778   n=10 s=1000
PathConsensus  --dot -k25  -p0.9  -s test-7.fa -g test-7.dot -o test-7.path test-6.fa test-6.dot
test-6.path
Ambiguous paths: 3
Merged:          0
No paths:        3
Too many paths:  0
Too complex:     0
Dissimilar:      0
cat test-6.fa test-7.fa \
      |MergeContigs   -k25 -o test-8.fa - test-7.dot test-7.path
```

 Copyright © Huawei Technologies Co., Ltd.

```
The minimum coverage of single-end contigs is 2.28.
The minimum coverage of merged contigs is 3.83248.
Consider increasing the coverage threshold parameter, c, to 3.83248.
ln -sf test-8.fa test-scaffolds.fa
PathOverlap --overlap   -k25 --dot test-7.dot test-7.path >test-8.dot
ln -sf test-8.dot test-scaffolds.dot
abyss-fac   test-unitigs.fa test-contigs.fa test-scaffolds.fa |tee test-stats.tab
n       n:500   L50     min     N80     N50     N20     E-size  max     sum     name
```

----End

4.19 Chaste 2019.1 Porting Guide (CentOS 7.6)

4.19.1 Introduction

Chaste (Cancer, Heart and Soft Tissue Environment) is a general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology. Current functionality includes tissue and cell level electrophysiology, discrete tissue modeling, and soft tissue modeling. The package is being developed by a team mainly based in the Computational Biology Group at the Department of Computer Science, University of Oxford, and development draws on expertise from software engineering, high performance computing, mathematical modeling, and scientific computing.

Programming language: C++

Brief description: general purpose simulation package

Open source license: 3-Clause BSD

Recommended Software Version

Chaste 2019.1

4.19.2 Environment Requirements

Hardware Requirements

Table 4-92 lists the hardware requirements.

Table 4-92 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-93 Software requirements

Item	Version	Download Address
Chaste	2019.1	https://github.com/Chaste/Chaste/archive/release_2019.1.tar.gz
PETSc	3.6.2	http://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-lite-3.6.2.tar.gz
Boost	1.58.0	http://downloads.sourceforge.net/project/boost/boost/1.58.0/boost_1_58_0.tar.gz
HDF5	1.8.16	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.16/src/hdf5-1.8.16.tar.bz2
Python	2.5	https://www.python.org/ftp/python/2.5/Python-2.5.tgz
Amara	1.2.0.2	https://files.pythonhosted.org/packages/af/0e/e81dfe4b4fb24023207d86e4d4ff418b0e83927f2155ed3552b0e3196846/Amara-1.2.0.2.tar.gz
4Suite-XML	1.0.2	https://files.pythonhosted.org/packages/0e/ae/3c5b4fffb12be7c3a80c99475853349e1cf8477f99051921ea06fbf5e3b9/4Suite-XML-1.0.2.tar.gz
lxml	3.2.1	https://files.pythonhosted.org/packages/a8/17/942c2f167cade0f387b2077299865b6e1d26dca75e1587e12df408b67d9a/lxml-3.2.1.tar.gz
RDFLib	2.4.2	https://files.pythonhosted.org/packages/04/06/3904f93f05a4734e4daec069aba9d590f53db899a6a81e8e79ffe6602466/rdflib-2.4.2.tar.gz
python-dateutil	1.5	Contained in CentOS 7.6
XSD	4.0.0-25	https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/x/xsd-4.0.0-25.el7.aarch64.rpm
Xerces	3.1.1-10	Contained in CentOS 7.6
VTK	6.2.0	http://www.vtk.org/files/release/6.2/VTK-6.2.0.tar.gz
METIS	5.1.0	http://ftp.mcs.anl.gov/pub/petsc/externalpackages/metis-5.1.0-p1.tar.gz
ParMETIS	4.0.3	http://ftp.mcs.anl.gov/pub/petsc/externalpackages/parmetis-4.0.3-p2.tar.gz
f2cblaslapack	3.4.2	http://ftp.mcs.anl.gov/pub/petsc/externalpackages/f2cblaslapack-3.4.2.q1.tar.gz
SUNDIALS	2.5.0	http://ftp.mcs.anl.gov/pub/petsc/externalpackages/sundials-2.5.0p1.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
Setuptools	0.6c8	https://files.pythonhosted.org/packages/29/17/f98a2cb39bb5b40357fd54fba1dd105e224a447a91867ac5c4ef6f8f0191/setuptools-0.6c8.tar.gz
OpenBLAS	0.3.6	https://github.com/xianyi/OpenBLAS/tree/v0.3.6
Test cases	-	https://chaste.cs.ox.ac.uk/chaste/tutorials/release_3.3/UserTutorials/CardiacExecutable/Propagation3d/Propagation3d.tgz

OS Requirements

Table 4-94 lists the OS requirements.

Table 4-94 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.19.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Chaste software porting.

Table 4-95 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CHASTE	Installation path of Chaste.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PETSC	Installation path of PETSc.	
4	path/to/BOOST	Installation path of Boost.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	path/to/PYTHON	Installation path of Python.	
6	path/to/AMARA	Installation path of Amara.	
7	path/to/LXML	Installation path of lxml.	
8	path/to/RDFLIB	Installation path of RDFLib.	
9	path/to/VTK	Installation path of VTK.	
10	path/to/XSD	Installation path of XSD.	
11	/path/to/OPENBLAS	Installation path of OpenBLAS.	

4.19.4 Configuring the Compilation Environment

4.19.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-96 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Xerces and python-dateutil	See **4.19.4.2 Installing Xerces and python-dateutil**.
3	Installing PETSc	See **4.19.4.3 Installing PETSc**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
4	Installing Boost	See **4.19.4.4 Installing Boost**.
5	Installing Python	See **4.19.4.5 Installing Python**.
6	Installing Amara	See **4.19.4.6 Installing Amara**.
7	Installing XSD	See **4.19.4.7 Installing XSD**.
8	Installing RDFLib	See **4.19.4.8 Installing RDFLib**.
9	Installing VTK	See **4.19.4.9 Installing VTK**.
10	Installing lxml	See **4.19.4.10 Installing lxml**.
11	Installing OpenBLAS	See **4.19.4.11 Installing OpenBLAS**.

4.19.4.2 Installing Xerces and python-dateutil

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Xerces and python-dateutil using Yum:

```
yum install xerces-c.aarch64 xerces-c-devel.aarch64 python-lxml.aarch64 python-dateutil.noarch -y
```

----End

4.19.4.3 Installing PETSc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the METIS, ParMETIS, f2cblaslapack, and SUNDIALS installation packages to the */path/to/PETSC* directory.

Step 3 Decompress the PETSc installation package.

```
cd /path/to/PETSC
tar -xvf petsc-lite-3.6.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd petsc-3.6.2
```

Step 5 Load environment variables.

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the configuration.

```
./configure --with-make-np=4 --with-fc=0 --with-x=false --with-ssl=false --download-f2cblaslapack=/path/to/PETSC/f2cblaslapack-3.4.2.q1.tar.gz --download-parmetis=/path/to/PETSC/parmetis-4.0.3-p2.tar.gz --download-sundials=/path/to/PETSC/sundials-2.5.0p1.tar.gz --with-shared-libraries --with-mpi-dir=$MPI_DIR --download-metis=/path/to/PETSC/metis-5.1.0-p1.tar.gz --download-hdf5=/path/to/PETSC/hdf5-1.8.16.tar.bz2
```

NOTE

- In the previous command, the **--download** parameter indicates that all the other software will be also installed.
- The **--download** parameter can be in a format similar to **--download-hdf5=1** or a format with a specified path. In the former format, the software will be downloaded and installed from the Internet; in the latter format, the software will be installed from the specified path.
- If the **problem** described in **4.19.8 Troubleshooting** is reported when you perform step **Step 6**, refer to the solution described in **Problem 1: An Error Is Reported During PETSc Configuration**. After the fault is rectified, perform step **Step 6** again.

Step 7 Perform the installation.

```
make
```

Step 8 Set the environment variables.

```
export PETSC_DIR=/path/to/PETSC/petsc-3.6.2
export PETSC_ARCH=linux-gnu
export SUNDIALS_ROOT=$PETSC_DIR/$PETSC_ARCH
export HDF5_ROOT=$PETSC_DIR/$PETSC_ARCH
```

----End

4.19.4.4 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the Boost source package to the */path/to/BOOST* directory.

Step 3 Decompress the Boost installation package.

```
cd /path/to/BOOST
tar -xvf boost_1_58_0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd boost_1_58_0
```

Step 5 Modify the Boost source code.

```
sed -ri 's/\-m64/\-mabi=lp64/g' `grep -Rl '\-m64'`
```

Step 6 Perform the configuration.

```
./bootstrap.sh --prefix=/path/to/BOOST/boost_1_58_0/build --with-libraries=system,filesystem,serialization,program_options --with-toolset=gcc
```

Step 7 Perform the compilation and installation.

```
./b2 install
```

Step 8 Set the Boost environment variable.

```
export LD_LIBRARY_PATH=/pat/to/BOOST/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.19.4.5 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the Python source package to the *path/to/PYTHON* directory.

Step 3 Decompress the Python installation package.

```
cd /path/to/PYTHON
tar -zxvf Python-2.5.tgz
```

Step 4 Go to the directory generated after the decompression.

```
cd Python-2.5
```

Step 5 Perform the configuration.

```
SVNVERSION=not-found ./configure --prefix=path/to/PYTHON
```

Step 6 Perform the compilation and installation.

```
make -j 16
make install
```

Step 7 Set the Python environment variable.

```
export PATH=path/to/PYTHON/bin:$PATH
```

----End

4.19.4.6 Installing Amara

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the 4Suite-XML and Amara source packages to the */path/to/AMARA* directory.

Step 3 Install the dependency package **4Suite-XML-1.0.2**.

```
cd /path/to/AMARA
tar -xvf 4Suite-XML-1.0.2.tar.gz
cd 4Suite-XML-1.0.2
python setup.py install
```

Step 4 Decompress the Amara installation package.

```
cd /path/to/AMARA
tar -zxvf Amara-1.2.0.2.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd Amara-1.2.0.2
```

Step 6 Perform the installation.

```
python setup.py install
```

NOTE

- The Amara version must be 1.2.0.2, because 2.* versions are incompatible with Chaste.
- Amara 1.2.0.2 must be installed using Python 2.5.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.19.4.7 Installing XSD

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the XSD software package to the */path/to/XSD* directory.

Step 3 Install XSD.

```
cd path/to/XSD
```

NOTE

- If the Internet Yum source is accessible, you can use the **yum install** command to install XSD.
- If an error message is displayed indicating that the dependency is missing when you download and install the RPM package, download and install the dependency.

----End

4.19.4.8 Installing RDFLib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the RDFLib and setuptools source packages to the */path/to/RDFLIB* directory.

Step 3 Install the setuptools source package.

```
cd /path/to/RDFLIB
unzip setuptools-0.6c8.zip
cd setuptools-0.6c8
python setup.py install
```

Step 4 Decompress the RDFLib source package.

```
cd path/to/RDFLIB
tar -zxvf rdflib-2.4.2.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd rdflib-2.4.2
```

Step 6 Perform the installation.

```
yum install libxml2-devel libxslt-devel -y
python setup.py install
```

----End

4.19.4.9 Installing VTK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the VTK source package to the */path/to/VTK* directory.

Step 3 Decompress the VTK installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd path/to/VTK
tar -xvf VTK-6.2.0.tar.gz
cd VTK-6.2.0
```

Step 4 Compile and install VTK.

```
cmake -DCMAKE_INSTALL_PREFIX=path/to/VTK
make -j 16
make install
```

Step 5 Set the VTK environment variable.

```
export PATH=/path/to/VTK/bin:$PATH
```

----End

4.19.4.10 Installing lxml

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the lxml source package to the *path/to/LXML* directory.

Step 3 Decompress the lxml source package.

```
cd /path/to/LXML
tar -xvf lxml-3.2.1.tar.gz
```

Step 4 Install lxml.

```
cd lxml-3.2.1
python setup,py install
```

----End

4.19.4.11 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the OpenBLAS source package to the *path/to/OPENBLAS* directory.

Step 3 Install BLAS and LAPACK using Yum.

```
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

Step 4 Decompress the OpenBLAS source package.

```
cd /path/to/OPENBLAS
tar -xvf OpenBLAS-0.3.6.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd OpenBLAS-0.3.6
```

Step 6 Perform the compilation and installation.

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.19.5 Obtaining the Source Code

Procedure

Step 1 Download the **Chaste-release_2019.1.tar.gz** source package.

https://github.com/Chaste/Chaste/archive/release_2019.1.tar.gz

Step 2 Use an SFTP tool to upload the Chaste source package to the */path/to/CHASTE* directory on the server.

----End

4.19.6 Compiling and Installing Chaste

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source code package.

```
cd /path/to/CHASTE
tar -xvf Chaste-release_2019.1.tar.gz
```

Step 3 Modify the **translators.py** file.

1. Open **translators.py**.

   ```
   vim /path/to/CHASTE/Chaste-release_2019.1/python/pycml/translators.py
   ```

2. Press **i** to enter the insert mode and modify the file.

   Uncomment line 1830 and comment out line 1831.

   Uncomment line 1838 and comment out line 1839.

   Before the modification:

   ```
   1830                #self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1831                self.writeln(error_template.format(self.var_display_name(var)))
   1832                self.close_block(False)
   1833            for var in high_range_vars:
   1834                if using_cvode:
   1835                    additional_tolerance_adjustment = ' + tol'
   1836                self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
   additional_tolerance_adjustment, ')')
   1837                self.open_block()
   1838                #self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1839                self.writeln(error_template.format(self.var_display_name(var)))
   ```

   After the modification:

   ```
   1830                self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1831                #self.writeln(error_template.format(self.var_display_name(var)))
   1832                self.close_block(False)
   1833            for var in high_range_vars:
   1834                if using_cvode:
   1835                    additional_tolerance_adjustment = ' + tol'
   1836                self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
   additional_tolerance_adjustment, ')')
   1837                self.open_block()
   1838                self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1839                #self.writeln(error_template.format(self.var_display_name(var)))
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Compile and install Chaste.

```
cd /path/to/CHASTE
mkdir chaste_build
cd chaste_build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/CHASTE/chaste_build /path/to/CHASTE/Chaste-release_2019.1 -DBoost_INCLUDE_DIR=/path/to/CHASTE/CHASTE_LIBS/boost_1_58_0/build/include
make -j48 Continuous
make install
export PATH=/path/to/CHASTE/chaste_build/apps:$PATH
```

----End

4.19.7 Running and Verifying Chaste

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **CHASTE** directory.

```
cd /path/to/CHASTE
```

Step 3 Execute the test cases in serial.

```
tar -xvf Propagation3d.tgz
cd Propagation3d
Chaste ChasteParameters.xml
```

After the execution is complete, the following information is displayed and files are generated in the **/tmp/root/testoutput/ChasteResults** directory.

```
      InMesh          Init         AssSys           Ode         Comms        AssRhs        NeuBCs
DirBCs           Ksp         Output   DataConversion       PostProc         User1         User2
User3         Total
  24.748 ( 17%)    15.570 ( 10%)    76.167 ( 51%)    23.206 ( 15%)     0.032 (  0%)     0.605 (  0%)     0.000
(  0%)     0.000 (  0%)     6.753 (  5%)     0.118 (  0%)     8.987 (  6%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)   149.822 (100%)  (seconds)
```

NOTE

progress_status.txt: estimates the time required for the simulation.

3dResults.h5: provides the simulation output in HDF5 format.

Output: the folder that stores the output in Meshalyzer format.

cmgui_output: the folder that stores the output in the CMGUI format.

vtk_output: the folder that stores the output in Paraview (VTK) format.

Step 4 Execute the test cases in parallel.

```
mpirun -np 96 --mca btl ^openib --allow-run-as-root Chaste ChasteParameters.xml
```

After the execution is complete, the following information is displayed and files are generated in the **/tmp/root/testoutput/ChasteResults** directory.

```
Proc      InMesh          Init         AssSys           Ode         Comms        AssRhs
NeuBCs         DirBCs           Ksp         Output   DataConversion       PostProc         User1
User2         User3         Total
  0:    5.108 ( 21%)     0.676 (  3%)     0.881 (  4%)     0.355 (  1%)     0.061 (  0%)     0.015 (  0%)     0.000
(  0%)     0.000 (  0%)     0.319 (  1%)     0.181 (  1%)    16.723 ( 68%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)    24.743 (100%)  (seconds)
  1:    5.195 ( 21%)     0.759 (  3%)     1.147 (  5%)     0.356 (  1%)     0.046 (  0%)     0.015 (  0%)     0.000
(  0%)     0.000 (  0%)     0.319 (  1%)     0.181 (  1%)    16.719 ( 68%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)    24.742 (100%)  (seconds)
  2:    5.109 ( 21%)     0.676 (  3%)     0.867 (  4%)     0.355 (  1%)     0.061 (  0%)     0.014 (  0%)     0.000
(  0%)     0.000 (  0%)     0.319 (  1%)     0.181 (  1%)    16.718 ( 68%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)    24.739 (100%)  (seconds)
```

 Copyright © Huawei Technologies Co., Ltd.

```
  3:    5.129 ( 21%)     0.692 (  3%)     0.953 (  4%)     0.355 (  1%)     0.059 (  0%)     0.014 (  0%)     0.000
(  0%)     0.000 (  0%)     0.319 (  1%)     0.181 (  1%)    16.719 ( 68%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)    24.743 (100%)   (seconds)
  4:    5.104 ( 21%)     0.672 (  3%)     0.857 (  3%)     0.356 (  1%)     0.061 (  0%)     0.015 (  0%)     0.000
(  0%)     0.000 (  0%)     0.319 (  1%)     0.181 (  1%)    16.719 ( 68%)     0.000 (  0%)     0.000 (  0%)
0.000 (  0%)     0.000 (  0%)    24.739 (100%)   (seconds)
```

----End

4.19.8 Troubleshooting

Problem 1: An Error Is Reported During PETSc Configuration

Symptom

After the **./configure** command is run, the following error information is displayed:

```
. This script, last modified 2008-01-23, has failed to recognize
the operating system you are using. It is advised that you
download the most up to date version of the config scripts from
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD
and
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD
```

Possible Causes

The **config.guess** and **config.sub** certificate files have expired.

Procedure

Step 1 Search for the **config.guess** and **config.sub** files.

```
find ./ -name config.guess
find ./ -name config.sub
```

Step 2 Replace the content in the files with the content at the links.

config.guess:

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD

config.sub:

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD

----End

Problem 2: An Error Is Reported During Boost Installation

Symptom

After the **./b2 install** command is run, the error message "gcc: error: unrecognized command line option '-m64'" is displayed.

Possible Causes

-m64 is an x86 64-bit application compilation option. The length of **int** is set to 32 bits, and the lengths of **long** and **pointer** are set to 64 bits to generate code for the AMD x86 64-bit architecture. However, ARM64 does not support this setting.

 Copyright © Huawei Technologies Co., Ltd.

Procedure

Modify the Boost source code and set the compilation option for ARM64 to **-mabi=lp64**.

```
sed -ri 's/\-m64/\-mabi=lp64/g' `grep -Rl '\-m64'`
```

Problem 3: An Error Is Reported During Python Configuration

Symptom

After the **./configure** command is run, the error message "gcc: error: directory": No such file or directory" is displayed.

Possible Causes

A bug exists in early Python versions.

Procedure

Run the following command:

```
SVNVERSION=not-found ./configure
```

Problem 4: An Error Is Reported During Chaste Installation

Symptom

After the **make -j48 Continuous** command is run, the error message "AttributeError: 'str' object has no attribute 'format'" is displayed.

Possible Causes

Format is a new method for formatting strings in Python 2.6.

Procedure

Modify the **translators.py** file.

1. Open **translators.py**.
   ```
   vim /path/to/CHASTE/Chaste-release_2019.1/python/pycml/translators.py
   ```
2. Press **i** to enter the insert mode and modify the file.

   Uncomment line 1830 and comment out line 1831.

   Uncomment line 1838 and comment out line 1839.

   Before the modification:

   ```
   1830                #self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1831                self.writeln(error_template.format(self.var_display_name(var)))
   1832                self.close_block(False)
   1833            for var in high_range_vars:
   1834                if using_cvode:
   1835                    additional_tolerance_adjustment = ' + tol'
   1836                self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
   additional_tolerance_adjustment, ')')
   1837                self.open_block()
   1838                #self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   1839                self.writeln(error_template.format(self.var_display_name(var)))
   ```

   After the modification:

   ```
   1830                self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ',
   self.code_name(var) , ' << std::endl << std::flush;')
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
1831                #self.writeln(error_template.format(self.var_display_name(var)))
1832                self.close_block(False)
1833            for var in high_range_vars:
1834                if using_cvode:
1835                    additional_tolerance_adjustment = ' + tol'
1836                self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
additional_tolerance_adjustment, ')')
1837                self.open_block()
1838                self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ',
self.code_name(var) , ' << std::endl << std::flush;')
1839                #self.writeln(error_template.format(self.var_display_name(var)))
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

4.19.9 More Resources

Chaste official website:

http://www.cs.ox.ac.uk/chaste/

4.20 COPASI 4.27.217 Porting Guide (CentOS 7.6)

4.20.1 Introduction

COPASI is a software application for simulation and analysis of biochemical networks and their dynamics. COPASI is a stand-alone program that supports models in the SBML standard and can simulate their behavior using ODEs or Gillespie's stochastic simulation algorithm; arbitrary discrete events can be included in such simulations.

For more information, visit the **official COPASI website**.

Programming language: C++

Brief description: a biochemical system simulator

Open source license: Artistic License 2.0

Recommended Software Version

COPASI 4.27.217

4.20.2 Environment Requirements

Hardware Requirements

Table 4-97 lists the hardware requirements.

Table 4-97 Hardware requirements

Item	Description
CPU	Kunpeng 920

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table Software Requirements lists the software requirements.

Table 4-98 Software requirements

Item	Version	Download Address
COPASI	4.27.217	https://codeload.github.com/copasi/COPASI/tar.gz/Build-217
copasi-dependencies	4.26.213	https://codeload.github.com/copasi/copasi-dependencies/tar.gz/v4.26.213

OS Requirements

Table 4-99 lists the OS requirements.

Table 4-99 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.20.3 Paths for Software Porting

This chapter lists the software installation paths involved in the COPASI software porting.

Table 4-100 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/COPASI	Installation path of COPASI.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.20.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.20.5 Obtaining the Source Code

Procedure

Step 1 Download the COPASI installation package **COPASI-Build-217.tar.gz**.

https://codeload.github.com/copasi/COPASI/tar.gz/Build-217

Step 2 Download the dependency package **copasi-dependencies-4.26.213.tar.gz** from its official website.

https://codeload.github.com/copasi/copasi-dependencies/tar.gz/v4.26.213

Step 3 Use an SFTP tool to upload the downloaded installation packages to the */path/to/COPASI* directory on the server.

----End

4.20.6 Compiling and Installing COPASI

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependencies:

```
yum install qt.aarch64 qt-devel.aarch64 -y
```

Step 3 Run the following commands to set environment variables:

```
export PATH=/path/to/cmake/bin:$PATH
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following commands to decompress the official dependency package:

```
cd /path/to/COPASI
tar -xvf copasi-dependencies-4.26.213.tar.gz
```

Step 5 Run the following commands to compile and install the dependencies:

```
cd copasi-dependencies-4.26.213
./createLinux.sh
```

Step 6 Run the following command to copy the files in the **lib64** directory to the **lib** directory:

```
cp -r ./bin/lib64/* ./bin/lib
```

Step 7 Run the following command to decompress the COPASI installation package:

```
cd /path/to/COPASI
tar -xvf COPASI-Build-217.tar.gz
```

Step 8 Run the following commands to compile and install COPASI:

```
cd COPASI-Build-217/
mkdir build
cd build
cmake -DBUILD_GUI=OFF -DCMAKE_INSTALL_PREFIX=/path/to/COPASI/COPASI-Build-217/build -
DCOPASI_DEPENDENCY_DIR=/path/to/COPASI/copasi-dependencies-4.26.213/bin ../
make install
```

Step 9 Run the following command to add the environment variable:

```
export PATH=/path/to/COPASI/COPASI-Build-217/build/bin:$PATH
```

----End

4.20.7 Running and Verifying COPASI

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the test case folder provided by the software:

```
cd /path/to/COPASI/COPASI-Build-217/TestSuite/events
```

Step 3 Run the following command to run COPASI:

```
CopasiSE --nologo EventTest10.cps
```

After the execution is complete, an **EventTest10.1.txt** file is generated. Some content in the file is as follows:

```
Time    [dClk-mRNA]     [PDP1_n]        [VRI_n] [dCLK_c]        [dCLK_n]        [PER/TIM]       [PER/TIM-
dCLK_n]         [vri-mRNA]      [VRI_c] [pdp1-mRNA]     [PDP1_c]        [per/tim-mRNA]  Values[T2]
Values[periodPDP1_n]         Values[peaktimePDP1_n]  Values[periodPDP1_n2]   Values[entryn]
   2 0      1       1       1       1       1       1       1       1       1       1       1       289     0       0       0       3
   3 0.2    1.060865        1.292758        1.14419 1.09696 4.367713e-07    0.2857835       1.473923
0.8788726       1.342719        0.9681366       1.247499        0.9871192       289.2198        0       0       0       3
   4 0.4    1.068151        1.648851        1.341789        1.183658        2.294009e-07    0.5653802
1.089928        0.7703788       1.552953        0.9366577       1.473933        0.9741363       289.439 0
0       0       3
   5 0.6    1.031855        2.059697        1.548262        1.245526        1.586807e-07    0.839073
0.8118105       0.6734762       1.660906        0.9055907       1.679842        0.9610636       289.657 0
0       0       3
   6 0.8    0.9729359       2.516842        1.734552        1.276873        1.212695e-07    1.10751
0.6115929       0.5871731       1.69151 0.8749563       1.865798        0.9479062       289.8732        0
0       0       3
   7 1      0.9055821       3.011998        1.883421        1.279214        9.706993e-08    1.371431
0.467994        0.5105355       1.665151        0.8447758       2.032401        0.9346702       290.087 0
0       0       3
```

```
  8 1.2     0.8373431       3.537083        1.986414        1.257421        7.963505e-08    1.631489
0.3650866       0.4426853       1.598318        0.8150713       2.180288        0.9213625       290.2979
0       0       0       3
  9 1.4     0.772229        4.084265        2.041389        1.217347        6.629547e-08    1.888166
0.2911314       0.382799        1.504186        0.7858652       2.310128        0.9079909       290.5051
0       0       0       3
 10 1.6     0.7126733       4.646009        2.050526        1.164757        5.573753e-08    2.14175
0.2376022       0.3301069       1.393124        0.75718 2.422625        0.8945639       290.7083        0
0       0       3
 11 1.8     0.6606023       5.215126        2.018753        1.104975        4.723108e-08    2.392337
0.1983997       0.283891        1.273154        0.7290379       2.518516        0.8810909       290.9068
0       0       0       3
```

----End

4.20.8 Troubleshooting

Problem 1: An Error Is Reported After the make install Progress Reaches 100%

Symptom

After the **make install** progress reaches 100%, an error message "make[2]: *** No rule to make target `*/path/to/COPASI/*copasi-dependencies-master/bin/lib/libexpat.a', needed by `copasi/CopasiSE/CopasiSE'. Stop." is displayed.

Possible Causes

- The **libexpat.a** file does not exist in the corresponding directory.
- When the script is used to install the dependency, the file is generated in the **lib64** directory at the same level.

Procedure

Step 1 Run the following command to switch to the **copasi-dependencies-4.26.213** directory:

```
cd /path/to/COPASI/copasi-dependencies-4.26.213
```

Step 2 Run the following command to copy the files in the **lib64** directory to the **lib** directory:

```
cp -r ./bin/lib64* ./bin/lib
```

----End

4.21 FASTA 36.3.8 Porting Guide (CentOS 7.6)

4.21.1 Introduction

The FASTA programs are a comprehensive set of similarity searching and alignment programs for searching protein and DNA sequence databases. Like the BLAST programs BLASTP and BLASTN, the fasta program itself uses a rapid heuristic strategy for finding similar regions in protein and DNA sequences. But in addition to heuristic similarity searching, the FASTA package provides programs for rigorous local (ssearch) and global (ggsearch) similarity searching, as well as a program for finding non-overlapping sequence similarities (lalign).

For more information, visit **FASTA**.

 Copyright © Huawei Technologies Co., Ltd.

Programming language: C

Brief description: A comprehensive set of similarity searching and alignment programs for searching protein and DNA sequence databases.

Open source license: Apache 2.0

Recommended Software Version

FASTA 36.3.8

4.21.2 Environment Requirements

Hardware Requirements

Table 4-101 lists the hardware requirements.

Table 4-101 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table Software requirements lists the software requirements.

Table 4-102 Software requirements

Item	Version	Download Address
FASTA	36.3.8	https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz
sse2neon	master	https://github.com/DLTcollab/sse2neon.git

OS Requirements

Table 4-103 lists the OS requirements.

Table 4-103 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.21.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FASTA software porting.

Table 4-104 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FASTA	Installation path of FASTA.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.21.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-105 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.21.5 Obtaining the Source Code

Procedure

Step 1 Download the FASTA installation package **fasta36-36.3.8h_04-May-2020.tar.gz**.

https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the FASTA installation package to the */path/to/FASTA* directory on the server.

----End

4.21.6 Compiling and Installing FASTA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar -xvf fasta36-36.3.8h_04-May-2020.tar.gz
unzip sse2neon-master.zip
```

Step 3 Go to the directory generated after the decompression.

```
cp sse2neon-master/sse2neon.h /path/to/FASTA/fasta36-36.3.8h_04-May-2020/src/
cd /path/to/FASTA/fasta36-36.3.8h_04-May-2020/make
```

Step 4 Modify the **Makefile.linux64_sse2** file.

1. Open **Makefile.linux64_sse2**.
   ```
   vim Makefile.linux64_sse2
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   13 SHELL=/bin/bash
   14
   15 CC = gcc -g -O2 -march=armv8.2-a
   16 LIB_DB=
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Go to the **src** directory.

```
cd ../src
```

Step 6 Modify the **smith_waterman_sse2.c** file.

1. Open **smith_waterman_sse2.c**.
   ```
   vim smith_waterman_sse2.c
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   23 #include "sse2neon.h"
   24 #endif
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the **global_sse2.c** file.

1. Open **global_sse2.c**.
   ```
   vim global_sse2.c
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   21 #else
   22 #include "sse2neon.h"
   23 #endif
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Modify the **glocal_sse2.c** file.

1. Open **glocal_sse2.c**.
   ```
   vim glocal_sse2.c
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   ```
   21 #else
   22 #include "sse2neon.h"
   23 #endif
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Perform the compilation and installation.

```
make -f ../make/Makefile.linux64_sse2
```

Step 10 Go to the **data** directory.

```
cd /path/to/FASTA/fasta36-36.3.8h_04-May-2020/data
```

----End

4.21.7 Running and Verifying FASTA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run FASTA.

```
export PATH=/path/to/FASTA/fasta36-36.3.8h_04-May-2020/bin:$PATH
mpirun --allow-run-as-root -np 96 ../bin/fasta36 -q ../seq/titin_hum.seq ../seq/prot_test.lseg
```

The following is an example of the output result:

```
82027 residues in 3 query   sequences
1674 residues in 12 library sequences
 Tcomplib [36.3.8h May, 2020] (128 proc in memory [0G])
 start: Fri May 22 09:28:42 2020 done: Fri May 22 09:29:32 2020
 Total Scan time: 51.260 Total Display time:  0.080

Function used was FASTA [36.3.8h May, 2020]
```

----End

4.22 GMAP 2015-9-21 Porting Guide (CentOS 7.6)

4.22.1 Introduction

GMAP is a genomic mapping and alignment program for mRNA and EST sequences.

Recommended Software Version

GMAP 2015-09-21

4.22.2 Environment Requirements

Hardware Requirements

Table 4-106 lists the hardware requirements.

Table 4-106 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table Software Requirements lists the software requirements.

Table 4-107 Software requirements

Item	Version	Download Address
GMAP	2015-09-21	http://research-pub.gene.com/gmap/src/gmap-gsnap-2015-09-21.tar.gz
Test case 1	human_g1k_v37.fasta	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

OS Requirements

Table 4-108 lists the OS requirements.

Table 4-108 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.22.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GMAP software porting.

Table 4-109 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/GMAP	Installation path of GMAP.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/TESTCASE	Installation path of the test file.	

4.22.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-110 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.22.5 Obtaining the Source Code

Procedure

Step 1 Download the GMAP installation package **gmap-gsnap-2015-09-21.tar.gz**.

http://research-pub.gene.com/gmap/src/gmap-gsnap-2015-09-21.tar.gz

Step 2 Download the **human_g1k_v37.fasta** file.

http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

Step 3 Use an SFTP tool to upload the GMAP installation package to the */path/to/GMAP* directory on the server.

Step 4 Use an SFTP tool to upload the test case file to the *path/to/TESTCASE* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.22.6 Compiling and Installing GMAP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar -zxvf gmap-gsnap-2015-09-21.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd /path/to/GMAP/gmap-2015-09-21
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/GMAP/Install --build=arm-linux
```

Step 5 Perform the compilation and installation.

```
make -j 32
make install
```

Step 6 Add the environment variable.

```
export PATH=/path/to/GMAP/Install/bin:$PATH
```

----End

4.22.7 Running and Verifying GMAP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 View the GMAP version.

```
gmap --verion
```

If information similar to the following is displayed, GMAP has been installed:

Step 3 Decompress the test case file.

```
cd path/to/TESTCASE
gzip -d human_g1k_v37.fasta.gz
```

Step 4 Create a reference sequence index.

```
mkdir human_index_home
gmap_build -D /path/to/TESTCASE/human_index_home -d human_g1k_v37 human_g1k_v37.fasta
```

NOTE

- **-D**: index path.
- **-d**: index file.

Step 5 Perform the comparison.

 Copyright © Huawei Technologies Co., Ltd.

```
{ time -p gmap -t 96 -D /path/to/TESTCASE/human_index_home -d human_g1k_v37 -f gff3_gene
human_g1k_v37.fasta; } 2>&1 |tee -a mapping_human.gff3
```

NOTE

-f: The output file is in gff3_gene format.

If the **mapping_human.gff3** file is generated, the execution is successful.

----End

4.23 VarScan 2.4.2 Porting Guide (CentOS 7.6)

4.23.1 Introduction

VarScan is a software tool developed at the Genome Institute at Washington University to detect variants in NGS data. VarScan employs a robust heuristic/statistic approach to call variants that meet desired thresholds for read depth, base quality, variant allele frequency, and statistical significance.

For more information, visit the **official VarScan website**.

Programming language: Java

Brief description: A software tool used to detect variants in NGS data.

Recommended Software Version

VarScan 2.4.2

4.23.2 Environment Requirements

Hardware Requirements

Table 4-111 lists the hardware requirements.

Table 4-111 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table Software requirements lists the software requirements.

Table 4-112 Software requirements

Item	Version	Download Address
VarScan	2.4.2	**https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2**
Samtools	1.10	**https://codeload.github.com/samtools/samtools/tar.gz/1.10**
Test case	hg19.fa	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/open_database/hg19/gatk_bundle/hg19.fa**
	bwa.sam	For BWA result files, see **BWA 0.7.17 Porting Guide (CentOS 7.6)**.

OS Requirements

Table 4-113 lists the OS requirements.

Table 4-113 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.23.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VarScan software porting.

Table 4-114 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/SAMTOOLS	Installation path of Samtools.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/VARSCAN	Installation path of VarScan.	

4.23.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-115 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.23.5 Obtaining the Source Code

Procedure

Step 1 Download the **samtools-1.10.tar.gz** package.

https://codeload.github.com/samtools/samtools/tar.gz/1.10

Step 2 Download the VarScan installation package **varscan-2.4.2.tar.gz**.

https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2

Step 3 Use an SFTP tool to upload the Samtools installation package to the */path/to/SAMTOOLS* directory and the VarScan installation package to the */path/to/VARSCAN* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.23.6 Compiling and Installing VarScan

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Go to the **SAMTOOLS** directory.

```
cd /path/to/SAMTOOLS
```

Step 4 Decompress the Samtools installation package.

```
tar -xvf samtools-1.10.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd samtools-1.10
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/SAMTOOLS/samtools-1.10-build
```

Step 7 Perform the compilation and installation.

```
make -j16
make install
```

Step 8 Add the Samtools environment variable.

```
export PATH=/path/to/SAMTOOLS/samtools-1.10-build/bin:$PATH
```

Step 9 Go to the VarScan directory.

```
cd /path/to/VARSCAN
```

Step 10 Decompress **varscan-2.4.2.tar.gz**.

```
tar -xvf varscan-2.4.2.tar.gz
```

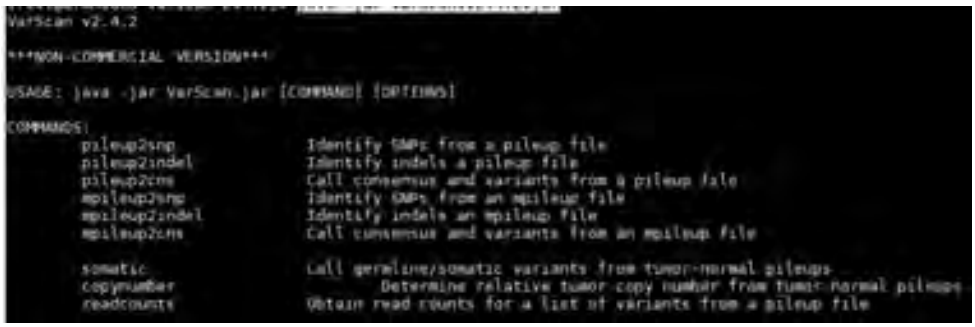
Step 11 Go to the directory generated after the decompression.

```
cd varscan-2.4.2
```

Step 12 Test the VarScan environment.

```
java -jar VarScan.v2.4.2.jar
```

If the following information is displayed, the environment is normal.



----End

 Copyright © Huawei Technologies Co., Ltd.

4.23.7 Running and Verifying VarScan

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Copy the test case files and go to the test directory.

```
cp hg19.fa /path/to/VARSCAN/varscan-2.4.2
cp bwa.sam /path/to/VARSCAN/varscan-2.4.2
cd /path/to/VARSCAN/varscan-2.4.2
```

Step 3 Use Samtools to convert the format and sort the data.

```
samtools view -bS bwa.sam > bwa.bam
samtools sort -@96 -o bwa.sort.bam bwa.bam
```

Step 4 Use Samtools to process the test case files.

```
samtools mpileup -f hg19.fa bwa.sort.bam > myData.pileup
```

Step 5 Execute the test case.

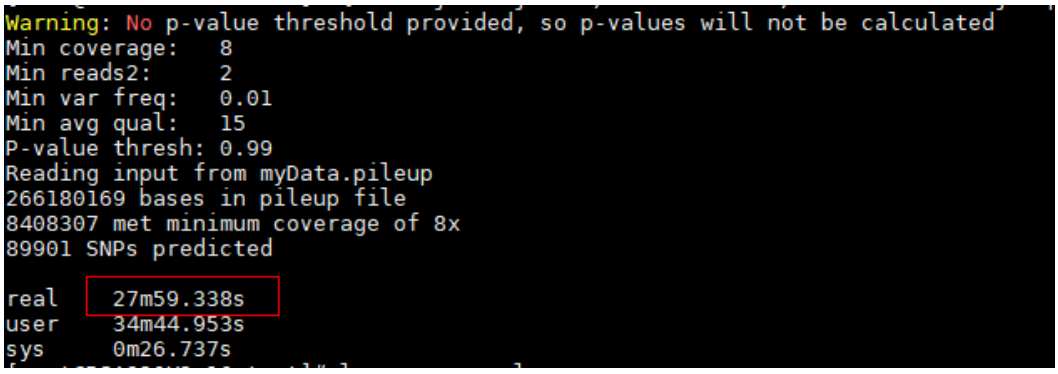
```
{ time java -jar VarScan.v2.4.2.jar pileup2snp myData.pileup > VarScan.vcf ; } 2>&1 |tee varscan.log
```

After the execution is complete, the **VarScan.vcf** and **varscan.log** files are generated in the current directory. **VarScan.vcf** is the case comparison result file and **varscan.log** is the run log file.

Check the value of **real** (in seconds) in the **varscan.log** file. A smaller value indicates higher performance.

Figure 4-7 is an example of the output result.

Figure 4-7 An example result



----End

4.24 Minimap2 Porting Guide (CentOS 7.6)

4.24.1 Introduction

Minimap2 is a versatile sequence alignment program that aligns DNA or mRNA sequences against a large reference database.

Programming language: C/C++

Brief description: versatile sequence alignment program

 Copyright © Huawei Technologies Co., Ltd.

Open source license: MIT

Recommended Software Version

Obtain the latest version using Git.

4.24.2 Environment Requirements

Hardware Requirements

Table 4-116 lists the hardware requirements.

Table 4-116 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table Software requirements lists the software requirements.

Table 4-117 Software requirements

Item	Version	How to Obtain
Minimap2	master	https://github.com/lh3/minimap2/archive/master.zip

OS Requirements

Table 4-118 lists the OS requirements.

Table 4-118 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.24.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Minimap2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-119 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/MINIMAP2	Installation path of Minimap2.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.24.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-120 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.24.5 Obtaining the Source Code

Procedure

Step 1 Download the Minimap2 installation package **minimap2-master.zip**.

https://github.com/lh3/minimap2/archive/master.zip

Step 2 Use an SFTP tool to upload the Minimap2 installation package to the */path/to/MINIMAP2* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.24.6 Compiling and Installing Minimap2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the Minimap2 dependency libraries and tools.

```
yum install wget git gcc-c++ zlib -y
```

Step 3 Decompress the installation package.

```
cd /path/to/MINIMAP2
unzip minimap2-master.zip
```

Step 4 Compile Minimap2. Kunpeng is compatible with the Armv8 64-bit instruction set. Perform the compilation as follows:

```
cd /path/to/MINIMAP2/minimap2-master
make arm_neon=1 aarch64=1
```

Step 5 Check the Minimap2 version.

```
cd /path/to/MINIMAP2/minimap2-master
./minimap2 --v
```

If the following information is displayed, the compilation is successful:

```
2.17-r974-dirty
```

----End

4.24.7 Running and Verifying Minimap2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the working directory.

```
cd /path/to/MINIMAP2/minimap2-master
```

Step 3 Execute the test case.

```
./minimap2 -a test/MT-human.fa test/MT-orang.fa > test.sam
```

If the following information is displayed, the execution is successful:

```
[M::mm_idx_gen::0.002*2.34] collected minimizers
[M::mm_idx_gen::0.006*2.70] sorted minimizers
[M::main::0.006*2.69] loaded/built the index for 1 target sequence(s)
[M::mm_mapopt_update::0.007*2.51] mid_occ = 2
[M::mm_idx_stat] kmer size: 15; skip: 10; is_hpc: 0; #seq: 1
[M::mm_idx_stat::0.007*2.40] distinct minimizers: 3111 (100.00% are singletons); average occurrences:
1.000; average spacing: 5.326; total length: 16569
[M::worker_pipeline::0.032*1.32] mapped 1 sequences
[M::main] Version: 2.17-r974-dirty
[M::main] CMD: ./minimap2 -a test/MT-human.fa test/MT-orang.fa
[M::main] Real time: 0.035 sec; CPU: 0.045 sec; Peak RSS: 0.005 GB
```

----End

4.25 bedtools 2.29.2 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.25.1 Introduction

bedtools is a powerful set of utilities for genomics analysis tasks. For example, bedtools allows one to intersect, merge, count, complement, and shuffle genomic intervals from multiple files in widely-used genomic file formats such as BAM, BED, GFF/GTF, VCF. While each individual tool is designed to do a relatively simple task (e.g., intersect two interval files), quite sophisticated analyses can be conducted by combining multiple bedtools operations on the Unix command line.

For more information, visit the **official bedtools website** .

Programming language: C/C++

Brief description: a powerful set of utilities for genomics analysis tasks

Open source license: GNU

Recommended Software Version

bedtools 2.29.2

4.25.2 Environment Requirements

Hardware Requirements

Table 4-121 lists the hardware requirements.

Table 4-121 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-122 lists the software requirements.

Table 4-122 Software requirements

Item	Version	Download Address
bedtools	2.29.2	https://github.com/arq5x/bedtools2/releases/download/v2.29.2/bedtools-2.29.2.tar.gz
Test case	aluY.chr1.bed	Test case provided by the software.
	gerp.chr1.bed	Provided by the software.

OS Requirements

Table 4-123 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-123 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.25.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bedtools software porting.

Table 4-124 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BEDTOOLS	Installation path of bedtools.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/TESTCASE	Path for storing test cases.	

4.25.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-125 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.25.5 Obtaining the Source Code

Procedure

Step 1 Download the **bedtools-2.29.2.tar.gz** installation package.

https://github.com/arq5x/bedtools2/releases/download/v2.29.2/bedtools-2.29.2.tar.gz

Step 2 Use an SFTP tool to upload the bedtools installation package to the */path/to/BEDTOOLS* directory on the server.

----End

4.25.6 Compiling and Installing bedtools

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install zlib-devel bzip2-devel xz-devel -y
```

Step 3 Decompress the installation package.

```
tar -zxvf bedtools-2.29.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd bedtools2
```

Step 5 Perform the compilation and installation.

```
make
```

Step 6 Add the environment variable.

```
export PATH=/path/to/BEDTOOLS/bedtools2/bin:$PATH
```

----End

4.25.7 Running and Verifying bedtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Copy the test cases in the **data** directory to the **TESTCASE** directory.

```
cd /path/to/BEDTOOLS/bedtools2/data
cp gerp.chr1.bed.gz aluY.chr1.bed.gz /path/to/TESTCASE
```

Step 3 Decompress the test case file.

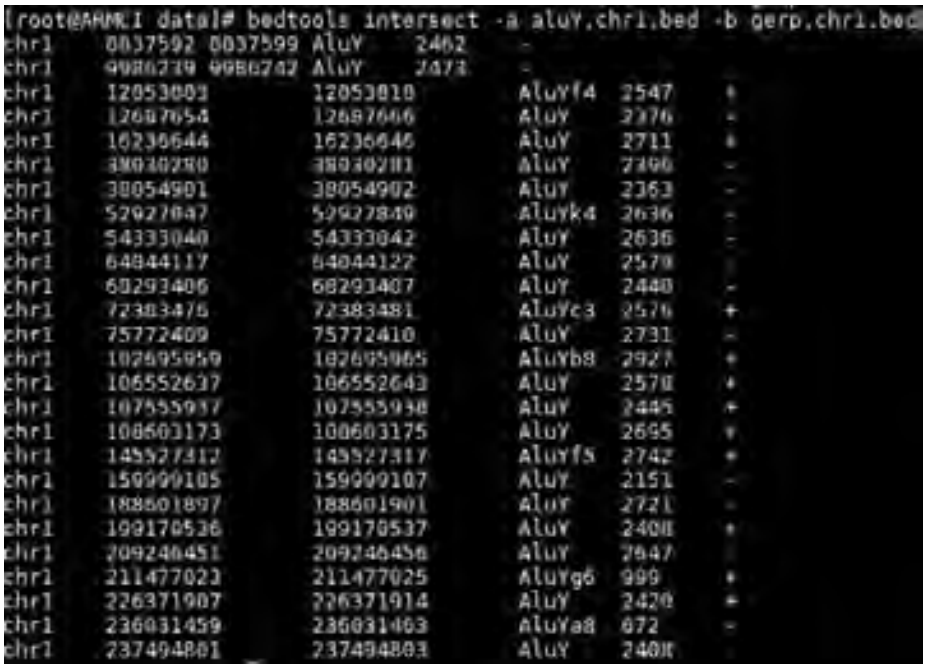
```
cd /path/to/TESTCASE
gzip -d gerp.chr1.bed.gz
gzip -d aluY.chr1.bed.gz
```

Step 4 Generate the overlap of the two intersecting files.

```
bedtools intersect -a aluY.chr1.bed -b gerp.chr1.bed
```

If information similar to the following is displayed, the execution is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.26 BEAST2 2.6.3 Porting Guide (CentOS 7.6)

4.26.1 Introduction

BEAST 2 is a cross-platform program for Bayesian phylogenetic analysis of molecular sequences. It estimates rooted, time-measured phylogenies using strict or relaxed molecular clock models. It can be used as a method of reconstructing phylogenies but is also a framework for testing evolutionary hypotheses without conditioning on a single tree topology. BEAST 2 uses Markov chain Monte Carlo (MCMC) to average over tree space, so that each tree is weighted proportional to its posterior probability. BEAST 2 includes a graphical user-interface for setting up standard analyses and a suit of programs for analysing the results.

For more information, visit the **official BEAST 2 website**.

Programming language: Java

Open source license: GPL

Recommended Software Version

BEAST v2.6.3

4.26.2 Environment Requirements

Hardware Requirements

Table 4-126 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-126 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-127 lists the software requirements.

Table 4-127 Software requirements

Item	Version	Download Address
BEAST 2	V2.6.3	https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz
Test case	Primates.nex	Provided by the software.

OS Requirements

Table 4-128 lists the OS requirements.

Table 4-128 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.26.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BEAST 2 software porting.

Table 4-129 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/BEAST	Installation path of BEAST 2.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.26.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-130 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.26.5 Obtaining the Source Code

Procedure

Step 1 Download the **BEAST.v2.6.3.Linux.tgz** installation package.

https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz

Step 2 Use an SFTP tool to upload the BEAST 2 installation package to the */path/to/BEAST* directory on the server.

----End

4.26.6 Compiling and Installing BEAST 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Install the Java package.

```
yum install java-1.8.0-openjdk.aarch64
```

Step 3 Decompress the BEAST 2 installation package.

```
tar -zxvf BEAST.v2.6.3.Linux.tgz
```

Step 4 Add the environment variable.

```
export PATH=/path/to/BEAST/beast/bin:$PATH
```

----End

4.26.7 Running and Verifying BEAST 2

The **Primates.nex** dataset is provided in the **nexus** directory of the BEAST 2 installation package. You can use BEAUti to set parameters and create an XML file. After the generated XML file is imported, run the **beast** command to generate the log file (*.log), tree file (*.trees), and XML status file (*.state).

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

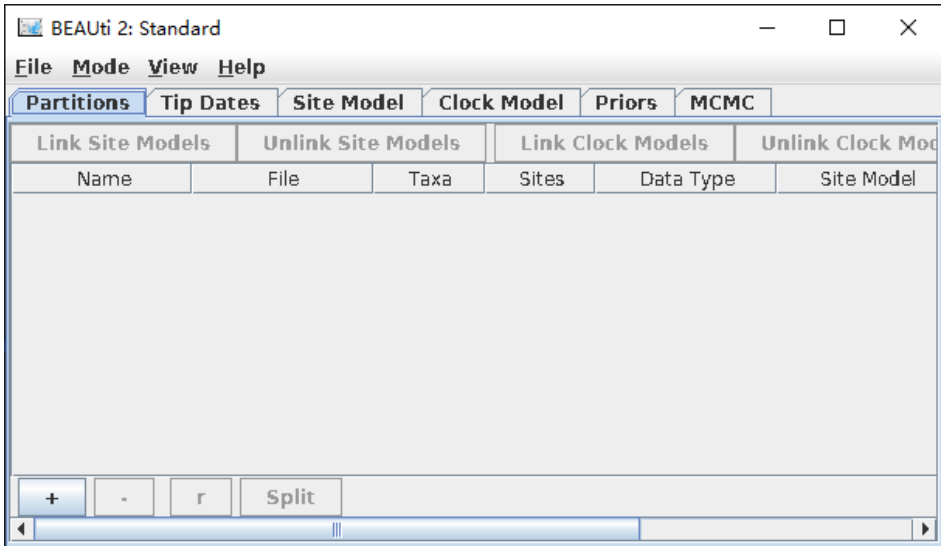
Step 2 Go to the **nexus** directory.

```
cd /path/to/BEAST/beast/examples/nexus
```

Step 3 Run the following command. The **BEAUti 2: Standard** window is displayed, as shown in **Figure 4-8**.

```
beauti
```

Figure 4-8 BEAUti 2: Standard

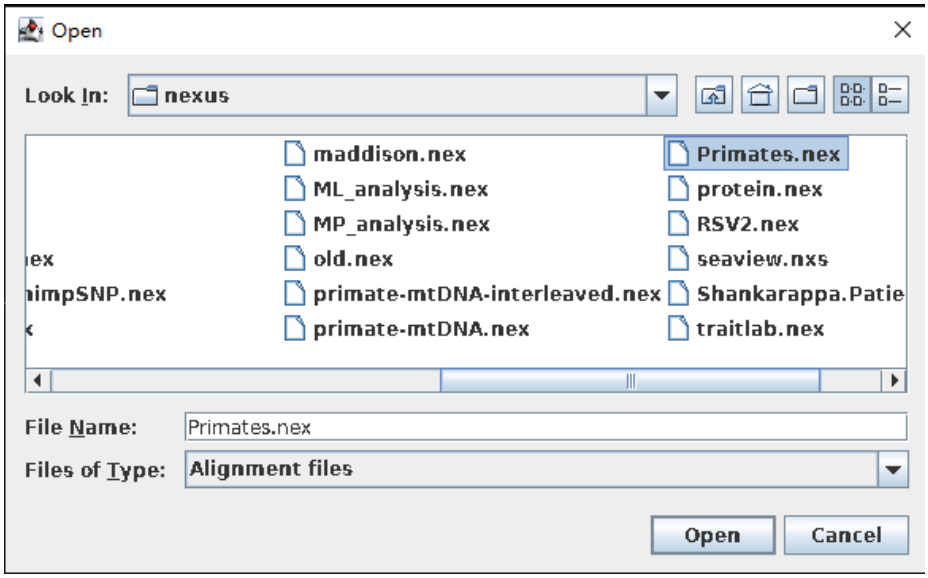


Step 4 Import a sequence.

Choose **File > Import Alignment**. The **Open** dialog box is displayed. Select the **Primates.nex** sequence file in the **nexus** directory, as shown in **Figure 4-9**.

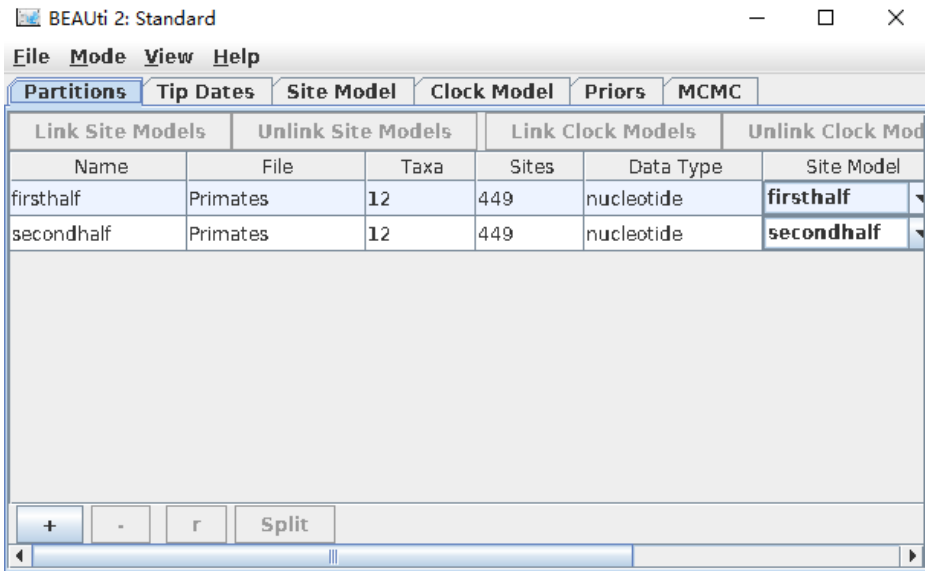
 Copyright © Huawei Technologies Co., Ltd.

Figure 4-9 Primates.nex sequence file



Step 5 Link or unlink partition models (Click **Open** and the following dialog box is displayed).

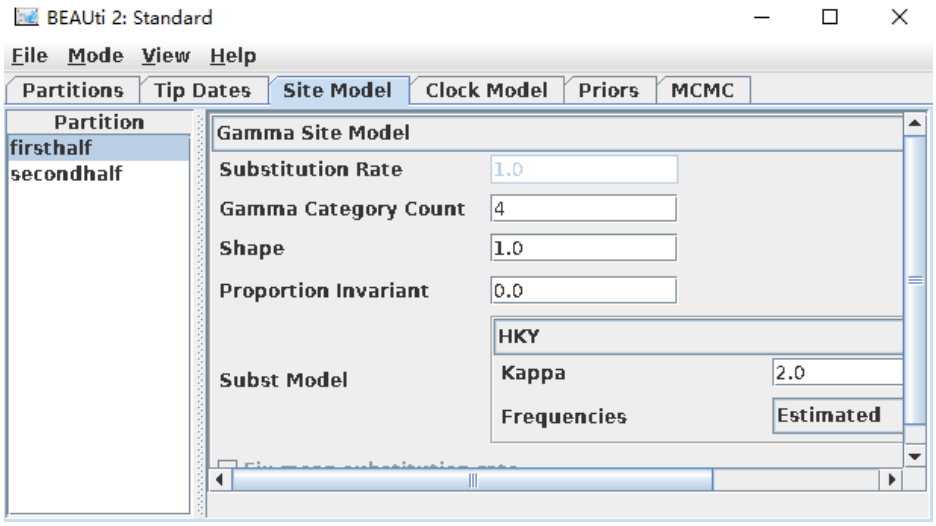
Generally, aligned gene sequence information is imported during analysis. However, in some cases, partitions may also exist. If joint analysis is required, link partition models. Otherwise, unlink them.



Step 6 Set the substitution model.

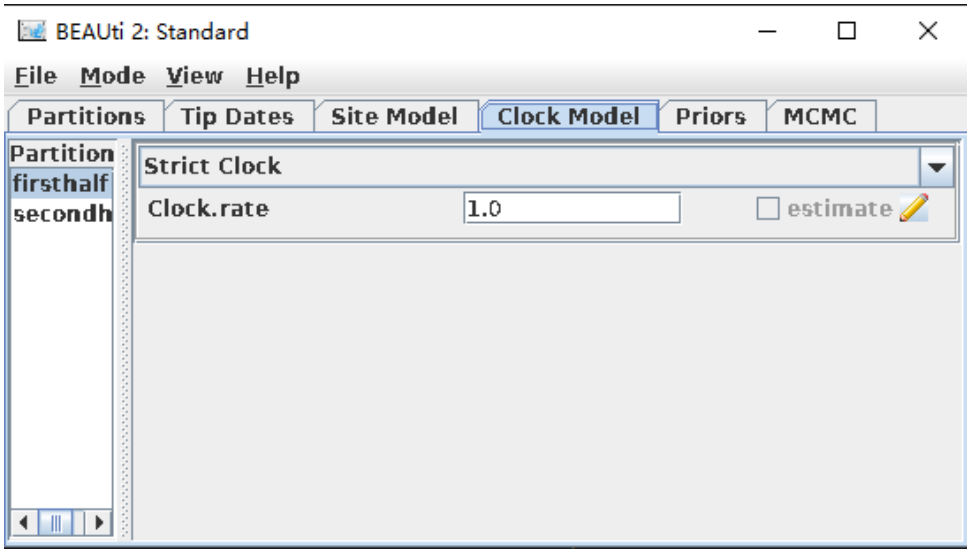
For the nucleotide substitution model, you can select **GTR**, **HKY**, **JC69**, or **TN93**. **Gamma Category Count** can be set to a value ranging from 4 to 8. **HKY** and **4** are selected here as an example. Other parameters such as **Substitution Rate**, **Shape** (the shape parameter of gamma distribution), and **Frequencies** are estimated in the analysis.

 Copyright © Huawei Technologies Co., Ltd.



Step 7 Set the molecular clock.

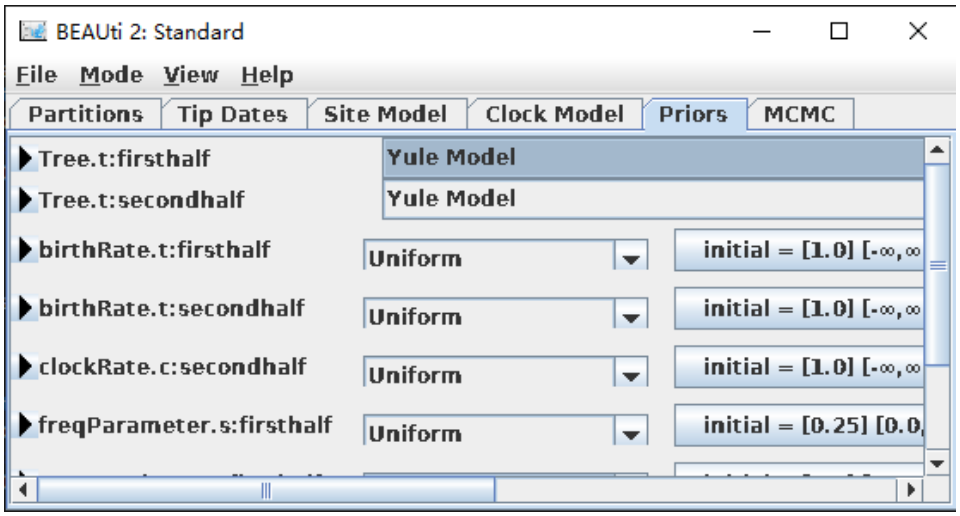
Click the **Clock Model** tab. Parameters include **Strict Clock**, **Relaxed Clock Exponential**, and **Relaxed Clock Log Normal**. If **Strict Clock** is selected, the rate difference between molecules in the model is not considered.



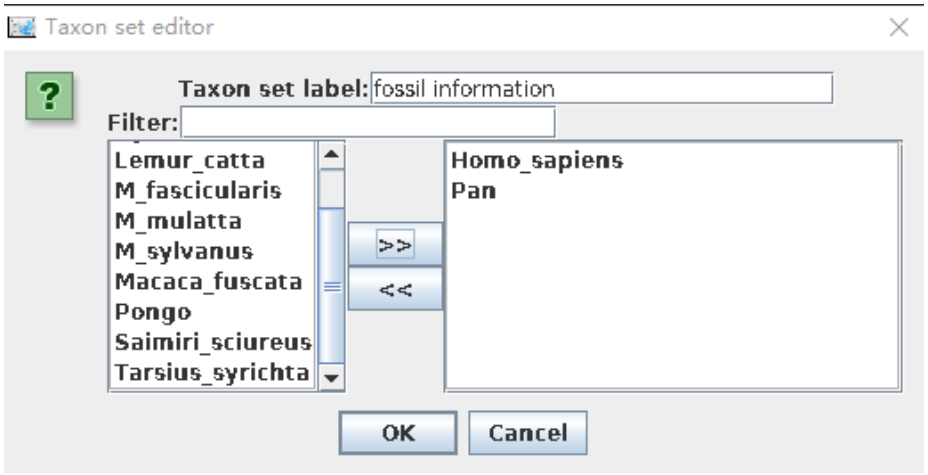
Step 8 Set verification information.

1. Click **Priors** and set **Tree.t:firsthalf** and **Tree.t:secondhalf** to **Yule Model**. Retain the default settings for other parameters.

 Copyright © Huawei Technologies Co., Ltd.

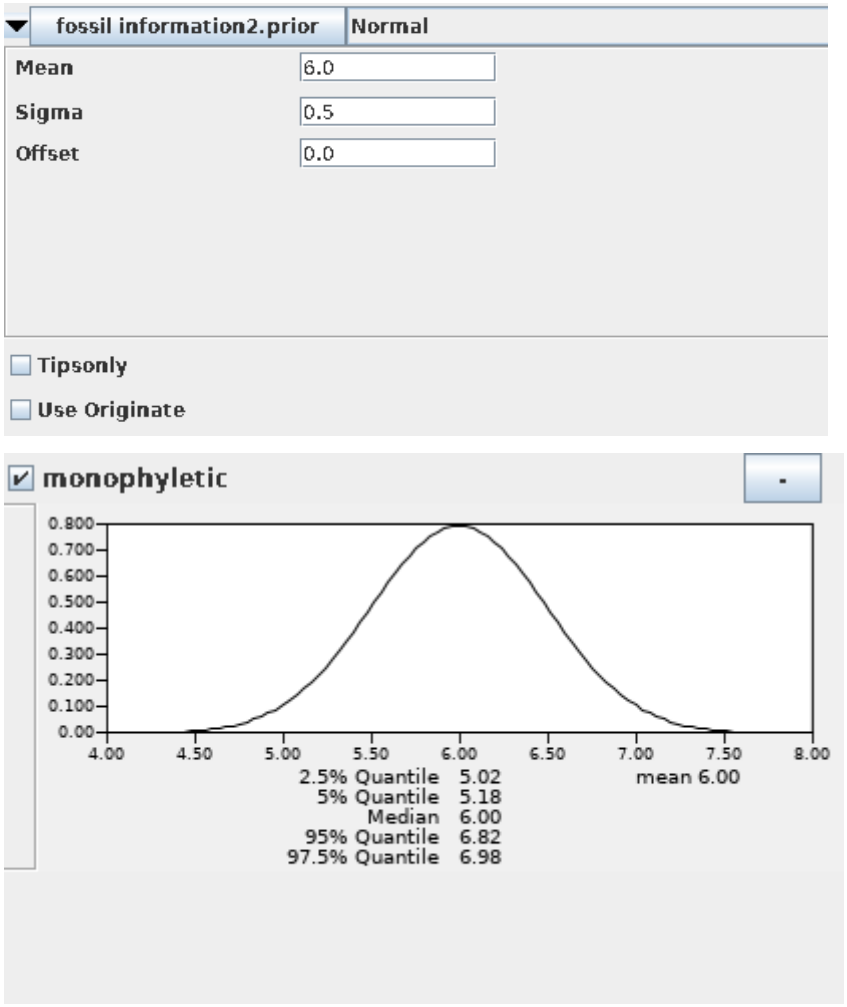


2. Click **+Add Prior** at the bottom of the **Priors** area. In the window that is displayed, select **Tree.t:secondhalf**. In **Taxon set editor**, define a name, for example, **fossil information**. Select two species (for example, **Homo_sapiens** and **Pan**) whose divergence time needs to be calibrated and move them to the right border.

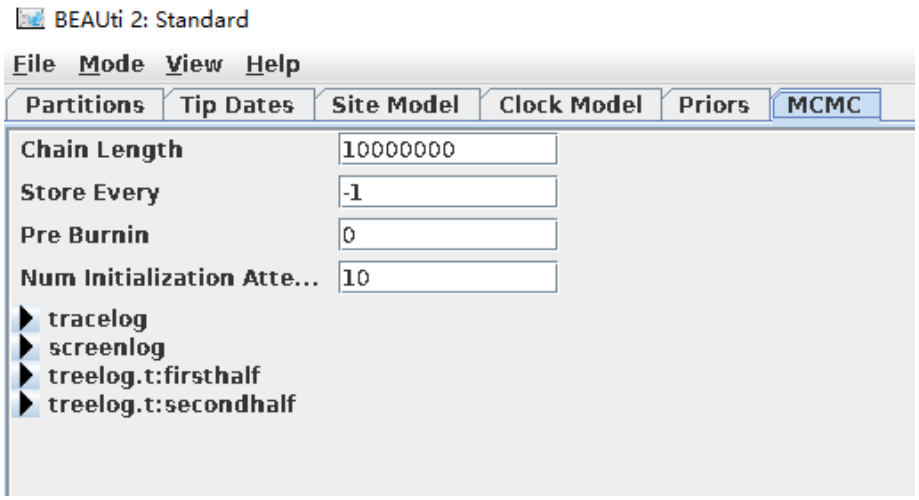


Select **Normal** in the monophyletic drop-down list to verify the distribution information. Set the divergence time. The divergence of Homo sapiens and Pan is between 5 and 7 My. Therefore, set the center point of the normal distribution to 6 My and the standard deviation to 0.5 My, and select **monophyletic**.

 Copyright © Huawei Technologies Co., Ltd.



Step 9 Click **MCMC** to set the MCMC options.



Chain Length: 10,000,000 (by default)

tracelog: The chain is sampled once every 1000 times.

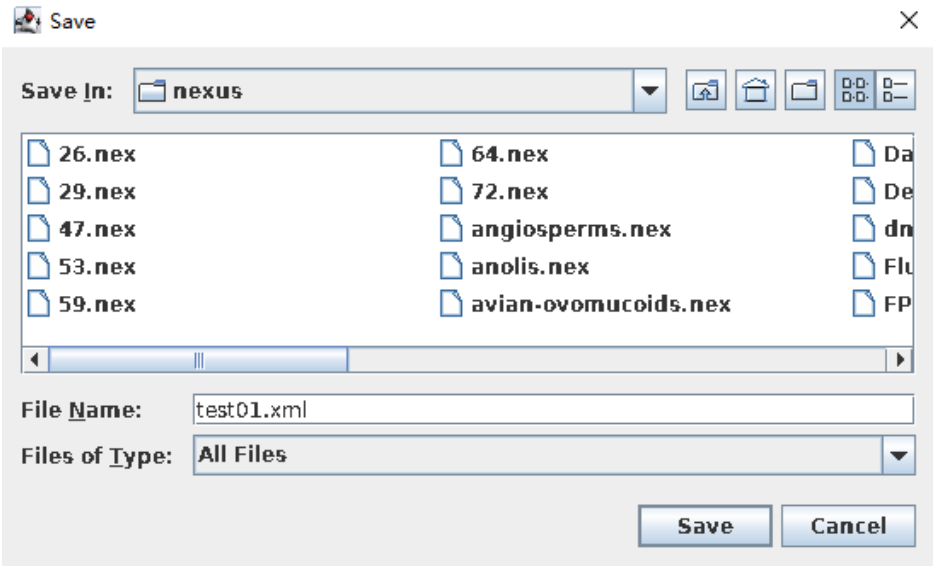
screenlog: The log is displayed on the screen every 1000 times.

treelog.t:tree: The topology is printed every 1000 times.

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Save the XML file.

Choose **File** > **Save As** and define an output file name to generate an XML file.



Step 11 Perform a BEAST test.

```
beast -threads 128 test01.xml
```

Check the value of **Total calculation time** (in seconds) in the **Primates.log** file. A smaller value indicates higher performance.

The following is an example of the output result:

```
    Tuning: The value of the operator's tuning parameter, or '-' if the operator ca
  #accept: The total number of times a proposal by this operator has been accepted
  #reject: The total number of times a proposal by this operator has been rejected
    Pr(m): The probability this operator is chosen in a step of the MCMC (i.e. the
Pr(acc|m): The acceptance probability (#accept as a fraction of the total proposal


Total calculation time: 3542.974 seconds
End likelihood: 1654.1860932605232
```

----End

4.26.8 More Resources

BEAST documentation: **http://beast.community/**

4.27 bcftools 1.10.2 Porting Guide (CentOS 7.6)

4.27.1 Introduction

BCFtools is a tool provided by Samtools developers to manipulate files in the variant call format (VCF). It can process VCF files as well as their binary counterparts BCF files. BCFtools consists of many commands for analyzing genotypic variants, manipulating VCF files, and creating indexes for VCF/BCF files.

Programming language: C

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a tool for manipulating VCF files

Open source license: MIT/Expat

Recommended Software Version

BCFtools 1.10.2

4.27.2 Environment Requirements

Hardware Requirements

Table 4-131 lists the hardware requirements.

Table 4-131 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-132 lists the software requirements.

Table 4-132 Software requirements

Item	Version	How to Obtain
bcftools	1.10.2	https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2

OS Requirements

Table 4-133 lists the OS requirements.

Table 4-133 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.27.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BCFtools software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-134 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BCFTOOLS	Installation path of BCFtools.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.27.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-135 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.27.5 Obtaining the Source Code

Procedure

Step 1 Download the **BCFtools-1.10.2.tar.bz2** installation package.

https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2

Step 2 Use an SFTP tool to upload the BCFtools installation package to the */path/to/BCFTOOLS* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.27.6 Compiling and Installing BCFtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install zlib-devel bzip2 bzip2-devel xz-devel libcurl-devel -y
```

Step 3 Decompress the installation package.

```
tar jxvf bcftools-1.10.2.tar.bz2
```

Step 4 Go to the directory generated after the decompression.

```
cd bcftools-1.10.2
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/BCFTOOLS/Install CFLAGS="-O3 -march=armv8.2-a"
```

Step 6 Perform the compilation and installation.

```
make -j 32
make install
```

Step 7 Add the environment variable.

```
export PATH=/path/to/BCFTOOLS/Install/bin:$PATH
```

----End

4.27.7 Running and Verifying BCFtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check the BCFtools version.

Information similar to **Figure 4-10** is displayed.

Figure 4-10 Version information

```
[root@ARMCI bcftools-1.10.2]# bcftools

Program: bcftools (Tools for variant calling and manipulating VCFs and BCFs)
Version: 1.10.2 (using htslib 1.10.2)
```

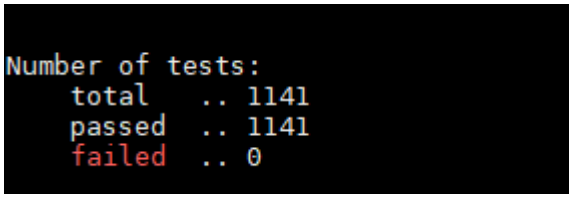
Step 3 Perform a test.

```
cd bcftools-1.10.2
make test
```

If the following information is displayed, the BCFtools test is successful, as shown in **Figure 4-11**.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-11 Successful BCFtools test



----End

4.28 Picard 2.23.3 Porting Guide (CentOS 7.6)

4.28.1 Introduction

Picard is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF.

For more information, visit the **official Picard website**.

Programming language: Java

Brief description: toolkit used to process HTS data and formats

Open source license: MIT

Recommended Software Version

Picard 2.23.3

4.28.2 Environment Requirements

Hardware Requirements

Table 4-136 lists the hardware requirements.

Table 4-136 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-137 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-137 Software requirements

Item	Version	Download Address
Picard	2.23.3	**https://github.com/broadinstitute/picard/releases/download/2.23.3/picard.jar**
Samtools	0.1.9	**https://codeload.github.com/samtools/samtools/tar.gz/0.1.9**
Test case	hg19.fa	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/open_database/hg19/gatk_bundle/hg19.fa**
	B17NC_R1.fq.gz	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz**
	B17NC_R2fq.gz	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz**
	hs38DH.fasta	**https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz**

OS Requirements

Table 4-138 lists the OS requirements.

Table 4-138 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.28.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Picard software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-139 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PICARD	Installation path of Picard.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/SAMTOOLS	Installation path of Samtools.	
4	/path/to/TESTCASE	Installation path of test cases.	
5	/path/to/BWA	Installation path of BWA.	See **BWA 0.7.17 Porting Guide (CentOS 7.6)**.

4.28.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-140 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.28.5 Obtaining the Source Code

Procedure

Step 1 Download the **samtools-0.1.9.tar.bz2** package.

https://codeload.github.com/samtools/samtools/tar.gz/0.1.9

Step 2 Download the **picard.jar** installation package.

https://github.com/broadinstitute/picard/releases/download/2.23.3/picard.jar

Step 3 Use an SFTP tool to upload the Samtools package to the */path/to/SAMTOOLS* directory and the Picard installation package to the */path/to/PICARD* directory on the server.

----End

4.28.6 Running and Installing Picard

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install dependencies.

```
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Go to the Samtools directory.

```
cd /path/to/SAMTOOLS
```

Step 4 Decompress the Samtools package.

```
tar -xvf samtools-0.1.9.tar.gz
```

Step 5 Go to the directory generated after decompression.

```
cd samtools-0.1.9
```

Step 6 Obtain the absolute path of GNU.

```
which gcc
```

Step 7 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   CC    = gcc
   ```

   After the modification:

   ```
   CC    = /path/to/GNU/bin/gcc
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation and installation.

```
make
```

Step 9 Add the Samtools environment variable.

```
export PATH=/path/to/SAMTOOLS/samtools-0.1.9:$PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Go to the Picard directory.

```
cd /path/to/PICARD
```

Step 11 Test the Picard environment.

```
java -jar picard.jar
```

If the following information is displayed, the environment is normal.

----End

4.28.7 Running and Verifying Picard

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Copy the test case files and go to the test directory.

```
cp B17NC_R1.fq.gz /path/to/TESTCASE
cp B17NC_R2.fq.gz /path/to/TESTCASE
cp GCA_000001405.15_GRCh38_full_analysis_set.fna.gz /path/to/TESTCASE
cd /path/to/TESTCASE
```

Step 3 Decompress the test files.

```
gzip -d B17NC_R1.fastq.gz
gzip -d B17NC_R2.fastq.gz
gzip -d GCA_000001405.15_GRCh38_full_analysis_set.fna.gz
```

Step 4 Rename the file to **hs38DH.fasta**.

```
mv GCA_000001405.15_GRCh38_full_analysis_set.fna hs38DH.fasta
```

Step 5 Add the BWA environment variable.

```
export PATH=/path/to/BWA/bwa-0.7.17:$PATH
export PLATFORM="Illumina"
export RG_ID="L1"
export LIB="pt2"
export SAMPLE=B17NC
```

Step 6 Process the test case files.

```
bwa index -a bwtsw hg19.fa
bwa mem -t 96 -o B17NC.sam -M -R "@RG\tID:$RG_ID\tPL:$PLATFORM\tSM:$SAMPLE\tCN:$LIB"
hg19.fa B17NC_R1.fastq B17NC_R2.fastq
```

Step 7 Use Samtools to convert the format and sort the data.

```
samtools view -bS B17NC.sam > B17NC.bam
samtools sort $SAMPLE.bam $SAMPLE.sort
```

 Copyright © Huawei Technologies Co., Ltd.

Step 8 Execute the test case.

```
{ time java -jar picard.jar CollectAlignmentSummaryMetrics R=hs38DH.fasta I=B17NC.sort.bam
O=output.txt; } 2>&1 |tee picard.log
```

After the execution is complete, the **output.txt** and **picard.log** files are generated in the current directory. The **output.txt** file contains the output and the **picard.log** file contains the run logs.

Check the value of **Elapsed time** (in minutes) in the **picard.log** file. A smaller value indicates higher performance.

Figure 4-12 is an example of the output result.

Figure 4-12 An example result

----End

4.28.8 Troubleshooting

Problem 1: An error is reported when the Picard command is run.

Symptom

An error message similar to the following is displayed when Picard is run: "Unable to load libgkl_compression.so from native/libgkl_compression.so (/tmp/libgkl_compression8107837509941713683.so: /tmp/libgkl_compression8107837509941713683.so: cannot open shared object file: No such file or directory (Possible cause: can't load AMD 64-bit .so on a AARCH64-bit platform))".

Possible Cause

Genomics Kernel Library (GKL) is an acceleration library developed based on the Intel CPU, and needs to be ported to the Kunpeng platform.

Solution

Ignore this error if performance comparison is not involved.

4.29 SOAPdenovo r241 Porting Guide (CentOS 7.6)

4.29.1 Introduction

SOAPdenovo is a novel short-read assembly method that can build a de novo draft assembly for the human-sized genomes. The software is specially designed to assemble Illumina GA short reads. The new version reduces memory consumption during graph creation, resolves repeats in contig steps, increases the coverage and

 Copyright © Huawei Technologies Co., Ltd.

length of scaffold assembly, and improves gap closure. Therefore, SOAPdenovo is more suitable for large genome assembly.

For more information, visit the **official SOAPdenovo website**.

Programming language: C/C++

Brief description: software for genome assembly

Recommended Software Version

SOAPdenovo r241

4.29.2 Environment Requirements

Hardware Requirements

Table 4-141 lists the hardware requirements.

Table 4-141 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-142 lists the software requirements.

Table 4-142 Software requirements

Item	Version	Download Address
SOAPdenovo	r241	**https://codeload.github.com/aquaskyline/SOAPdenovo2/tar.gz/r241**
Test case	B17NC_R1.fq.gz	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz**
	B17NC_R2.fq.gz	**https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz**

OS Requirements

Table 4-143 lists the OS requirements.

Table 4-143 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

4.29.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SOAPdenovo software porting.

Table 4-144 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/SOAPDENOVO	Installation path of SOAPdenovo.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.29.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-145 Configuration process

No.	Configuration Item	Description
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.29.5 Obtaining the Source Code

Procedure

Step 1 Download the **SOAPdenovo2-r241.tar.gz** installation package.

https://codeload.github.com/aquaskyline/SOAPdenovo2/tar.gz/r241

Step 2 Use an SFTP tool to upload the SOAPdenovo installation package to the */path/to/SOAPDENOVO* directory on the server.

----End

4.29.6 Compiling and Installing SOAPdenovo

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the SOAPdenovo directory.

```
cd /path/to/SOAPDENOVO
```

Step 3 Decompress **SOAPdenovo2-r241.tar.gz**.

```
tar -xvf SOAPdenovo2-r241.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd SOAPdenovo2-r241
```

Step 5 Modify the **standardPregraph/hashFunction.c** file.

1. Open **standardPregraph/hashFunction.c**.

   ```
   vi standardPregraph/hashFunction.c
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   static uint32_t cpuid ( uint32_t functionInput )
   {
   uint32_t eax;
   uint32_t ebx;
   uint32_t ecx;
   uint32_t edx;
   #ifdef __PIC__
   asm ( "pushl %%ebx\n\t" /* save %ebx */
   "cpuid\n\t"
   "movl %%ebx, %[ebx]\n\t" /* save what cpuid just put in %ebx */
   "popl %%ebx" : "=a" ( eax ), [ebx] "=r" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput )
   : "cc" );
   #else
   asm ( "cpuid" : "=a" ( eax ), "=b" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput ) );
   #endif
   return ecx;
   }
   ```

   After the modification:

   ```
   static uint32_t cpuid ( uint32_t functionInput )
   {
   uint32_t eax;
   uint32_t ebx;
   uint32_t ecx;
   uint32_t edx;

   #ifdef __aarch64__
   ```

```
ecx = 0x0;
#else
#ifdef __PIC__
asm ( "pushl %%ebx\n\t" /* save %ebx */
"cpuid\n\t"
"movl %%ebx, %[ebx]\n\t" /* save what cpuid just put in %ebx */
"popl %%ebx" : "=a" ( eax ), [ebx] "=r" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput )
: "cc" );
#else
asm ( "cpuid" : "=a" ( eax ), "=b" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput ) );
#endif
#endif
return ecx;
}
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation.

```
make
```

Step 7 Test the SOAPdenovo environment.

```
./SOAPdenovo-63mer
```

If the following information is displayed, the environment is normal.

```
Version 2.04: released on July 13th, 2012
Compile Aug 12 2020     11:12:50

Usage: SOAPdenovo <command> [option]
    pregraph        construct kmer-graph
    sparse_pregraph construct sparse kmer-graph
    contig          eliminate errors and output contigs
    map             map reads to contigs
    scaff           construct scaffolds
    all             do pregraph-contig-map-scaff in turn
```

----End

4.29.7 Running and Verifying SOAPdenovo

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Copy the test case files and go to the test directory.

```
cp B17NC_R1.fq.gz /path/to/SOAPDENOVO/SOAPdenovo2-r241
cp B17NC_R2.fq.gz /path/to/SOAPDENOVO/SOAPdenovo2-r241
cd /path/to/SOAPDENOVO/SOAPdenovo2-r241
```

Step 3 Decompress the case files.

```
gzip -d B17NC_R1.fastq.gz
gzip -d B17NC_R2.fastq.gz
```

Step 4 Clear the cache.

```
echo 3 > /proc/sys/vm/drop_caches
```

Step 5 Create a configuration file.

1. Create a configuration file.
   ```
   vi config_file
   ```
2. Press **i** to enter the insert mode and add the following content:

```
max_rd_len=100
[LIB]
avg_ins=200
reverse_seq=0
asm_flags=3
rd_len_cutoff=100
rank=1
q1=B17NC_R1.fastq
q2=B17NC_R2.fastq
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

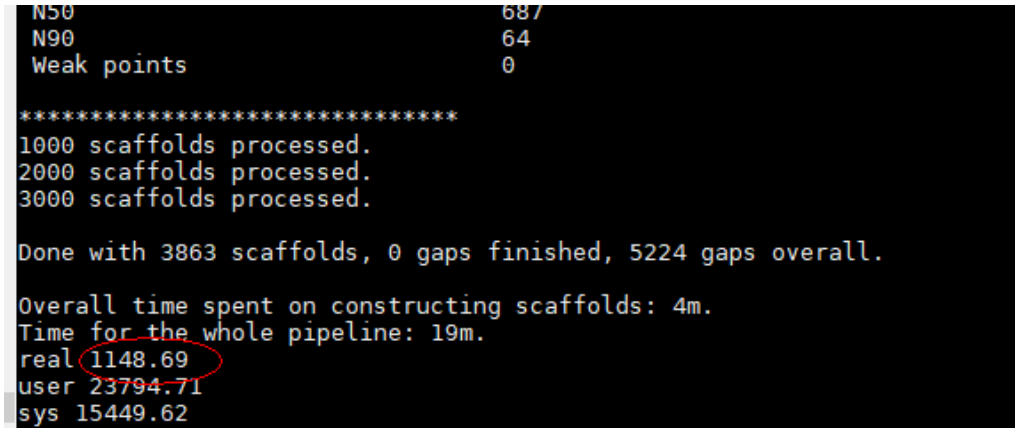
Step 6 Execute the test case.

```
{ time -p ./SOAPdenovo-63mer all -s config_file -K 63 -p 96 -R -o graph_prefix; } 2>&1 |tee SOAPdenovo.log
```

Check the value of **real** (in seconds) in the **SOAPdenovo.log** file. A smaller value indicates higher performance.

Figure 4-13 is an example of the output result.

Figure 4-13 An example result



----End

4.30 Blastz 2004-12-27 Porting Guide (CentOS 7.6)

4.30.1 Introduction

Blastz is a nucleotide local alignment program developed by Webb Miller's group. It takes FASTA or nib sequences as input and produces output in its own format, lav. It has some notable algorithmic improvements over traditional blast.

For more information about Blastz, visit the **Blastz official website**.

Programming language: C

Brief description: a nucleotide local alignment program

Open-source license: GPL

Recommended Version

Blastz 2004-12-27.

 Copyright © Huawei Technologies Co., Ltd.

4.30.2 Environment Requirements

Hardware Requirements

Table 4-146 lists the hardware requirements.

Table 4-146 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-147 lists the software requirements.

Table 4-147 Software requirements

Item	Version	Download Link
Blastz	2004-12-27	http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz
Test case	hg38.chr4.fa	http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz
	galGal3.chr4.fa	http://hgdownload.soe.ucsc.edu/goldenPath/galGal3/chromosomes/chr4.fa.gz

OS Requirements

Table 4-148 lists the OS requirements.

Table 4-148 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

4.30.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Blastz software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-149 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	*/path/to/BLASTZ*	Installation path of Blastz.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	*/path/to/CASE*	Path for storing the test case.	

4.30.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-150 Configuration process

No.	Configuration Item	Remarks
1	Setting Up the Basic Environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.30.5 Obtaining Source Code

Procedure

Step 1 Download the Blastz installation package **blastz-2004-12-27.deprecated.tar.gz** from the following link:

http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz

Step 2 Run the following commands to download the **hg38.chr4.fa** and **galGal3.chr4.fa** test cases:

wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz -O hg38.chr4.fa.gz

 Copyright © Huawei Technologies Co., Ltd.

wget http://hgdownload.soe.ucsc.edu/goldenPath/galGal3/chromosomes/chr4.fa.gz -O galGal3. chr4.fa.gz

Step 3 Use an SFTP tool to upload the downloaded source code package to the ***/path/to/BLASTZ*** directory on the server.

Step 4 Use an SFTP tool to upload the test cases to the ***/path/to/CASE*** directory on the server.

----End

4.30.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the Blastz directory:

cd */path/to/BLASTZ*

Step 3 Run the following command to decompress the Blastz package:

tar -xvf blastz-2004-12-27.deprecated.tar.gz

Step 4 Run the following command to switch to the directory generated after decompression:

cd blastz-source

Step 5 Run the following command to modify the compilation parameters:

sed -i 's/CFLAGS= -O/CFLAGS= -O3 -march=armv8.2-a -mtune=tsv110 -flto/g' Makefile

Step 6 Run the following command to perform compilation and installation:

CC=`which gcc` make

Step 7 Run the following command to add the Blastz environment variables:

export PATH=*/path/to/BLASTZ***/blastz-source:$PATH**

Step 8 Run the following command to test the Blastz environment:

blastz --help

If information similar to the following is displayed, the environment is normal:

 Copyright © Huawei Technologies Co., Ltd.

```
blastz.v7 command options (e.g., "blastz seq1 seq2 B=0 C=2"):
Default values are given in parentheses.
  m(80M) bytes of space for trace-back information
  v(0) 0: quiet; 1: verbose progress reports to stderr
  B(2) 0: single strand; >0: both strands
  C(0) 0: no chaining; 1: just output chain; 2: chain and extend;
       3: just output HSPs
  E(30) gap-extension penalty.
  G(0) diagonal chaining penalty.
  H(0) interpolate between alignments at threshold K = argument.
  K(3000) threshold for MSPs
  L(K) threshold for gapped alignments
  M(0) mask any base in seq1 hit this many times; 0 = no dynamic masking
  O(400) gap-open penalty.
  P(1) 0: entropy not used; 1: entropy used; >1 entropy with feedback.
  Q load the scoring matrix from a file.
  R(0) antidiagonal chaining penalty.
  T(1) 0: W-bp words;  1: 12of19;  2: 12of19 without transitions.
                       3: 14of22;  4: 14of22 without transitions.
  W(8) word size (unused unless T=0)
  X(10*(A-to-A match score)) X-drop parameter for ungapped extension.
  Y(O+300E) X-drop parameter for gapped extension.
  Z(1) increment between successive words in sequence 1.
```

----End

4.30.7 Running and Verifying Blastz

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the **CASE** directory:

cd */path/to/CASE*

Step 3 Run the following commands to decompress the test case packages:

gzip -d hg38.chr4.fa.gz

gzip -d galGal3.chr4.fa.gz

Step 4 Run the following command to run the test cases:

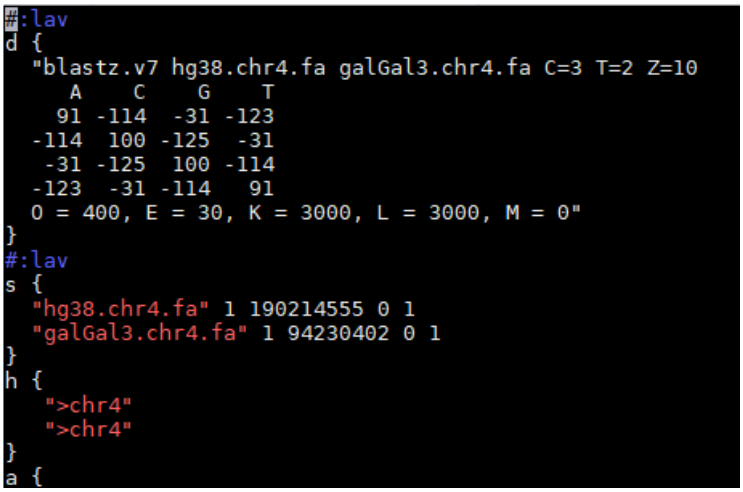
time blastz hg38.chr4.fa galGal3.chr4.fa C=3 T=2 Z=10 > hg38_c4-vs-galGal3_c4-kp7265.out

View the **hg38_c4-vs-galGal3_c4-kp7265.out** information.

Figure 4-14 shows an example of the result.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-14 An example result



----End

4.31 CAFE 5.0b2 Porting Guide (CentOS 7.6)

4.31.1 Introduction

The purpose of CAFE is to analyze changes in gene family size in a way that accounts for phylogenetic history and provides a statistical foundation for evolutionary inferences. The program uses a birth and death process to model gene gain and loss across a user-specified phylogenetic tree. The distribution of family sizes generated under this model can provide a basis for assessing the significance of the observed family size differences among taxa.

For more information about CAFE, visit the **CAFE official website**.

Programming language: C++

Brief description: CAFE analyzes changes in gene family size and provides a statistical foundation for evolutionary inferences.

Open-source license: ELC-2.0

Recommended Version

CAFE 5.0b2.

4.31.2 Environment Requirements

Hardware Requirements

Table 4-151 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-151 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-152 lists the software requirements.

Table 4-152 Software requirements

Item	Version	Download Link
CAFE	5.0b2	https://github.com/hahnlab/CAFExp/archive/v5.0b2.tar.gz

OS Requirements

Table 4-153 lists the OS requirements.

Table 4-153 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

4.31.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CAFE software porting.

Table 4-154 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	*/path/to/CAFE*	Installation path of CAFE.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.31.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-155 Configuration process

No.	Configuration Item	Remarks
1	Setting Up the Basic Environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.31.5 Obtaining Source Code

Procedure

Step 1 Download the CAFE source code package **CAFExp-5.0b2.tar.gz** from the following link:

https://github.com/hahnlab/CAFExp/archive/v5.0b2.tar.gz

Step 2 Use an SFTP tool to upload the downloaded software package to the ***/path/to/CAFE*** directory on the server.

----End

4.31.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to decompress the CAFE installation package:

tar -xvf CAFExp-5.0b2.tar.gz

Step 3 Run the following command to switch to the directory generated after decompression:

cd CAFExp-5.0b2

Step 4 Run the following command to generate the configuration file:

autoconf

Step 5 Run the following command to perform compilation and installation:

./configure

Step 6 Run the following command to modify the **Makefile** file:

1. Run the following command to modify the **Makefile** file:

   vi Makefile

2. Press **i** to go to the edit mode and add the following content to line 7 of the file:

   ```
   -march=armv8.2-a -mtune=tsv110 -flto
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to perform compilation:

make -j 16

Step 8 Run the following command to set environment variables:

export PATH=*/path/to/CAFE***/CAFExp-5.0b2/bin:$PATH**

----End

4.31.7 Running and Verifying CAFE

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory where the test case is stored:

cd */path/to/CAFE***/CAFExp-5.0b2/examples**

Step 3 Run the following commands to run the case:

../bin/cafexp -t mammals_tree.txt -i mammal_gene_families.txt -o singlelambda

Step 4 After the commands are executed, the following information is displayed:

```
Completed 24 iterations
Time: 0H 0M 7S
Best match is: 0.0018174666100378
Final -lnL: 164769.22040255
47 values were attempted (0% rejected)

Inferring processes for Base model
Score (-lnL): 164769.22040255
```

 Copyright © Huawei Technologies Co., Ltd.

```
Computing pvalues...done!
Starting reconstruction processes for Base model
Done!
```

----End

4.31.8 More Resources

To obtain more resources, visit CAFE at GitHub: **https://hahnlab.github.io/CAFE/**

4.32 fastp Porting Guide (CentOS 7.6)

4.32.1 Introduction

fastp is a tool designed to provide fast all-in-one preprocessing for FastQ files. It is developed in C++ and supports multiple threads to provide high performance.

For more information, visit the **fastp page at GitHub**.

Programming languages: C/C++

Brief description: a fast all-in-one preprocessing for FastQ preprocessor

Open source license: MIT license

Recommended Software Version

fastp 0.20.1

4.32.2 Environment Requirements

Hardware Requirements

Table 4-156 lists the hardware requirements.

Table 4-156 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-157 lists the software requirements.

Table 4-157 Software requirements

Item	Version	Download Link
fastp	0.20.1	**https://github.com/OpenGene/fastp/archive/v0.20.1.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Link
Test case 1	ERR1044518_1.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_1.fastq.gz
Test case 2	ERR1044518_2.fastq	ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_2.fastq.gz

OS Requirements

Table 4-158 lists the OS requirements.

Table 4-158 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.32.3 Paths for Software Porting

This chapter lists the software installation paths involved in the fastp software porting.

Table 4-159 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FASTP	Installation path of fastp.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.32.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-160 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.32.5 Obtaining Source Code

Procedure

Step 1 Download the fastp source code package **fastp-0.20.1.tar.gz** from the following link:

https://github.com/OpenGene/fastp/archive/v0.20.1.tar.gz

Step 2 Use an SFTP tool to upload the source code package to the */path/to/FASTP* directory on the server.

----End

4.32.6 Compiling and Installing fastp

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the fastp installation package:

```
tar -xvf fastp-0.20.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd fastp-0.20.1
```

Step 4 Run the following command to perform the compilation and installation:

```
make -j 16
```

Step 5 Run the following command to set the environment variable:

```
export PATH=/path/to/FASTP/fastp-0.20.1:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.32.7 Running and Verifying fastp

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the directory where the test case is stored:

```
cd /path/to/FASTP/testdata
```

Step 3 Run the following commands to decompress the case files:

```
gzip -d ERR1044518_1.fastq.gz
gzip -d ERR1044518_2.fastq.gz
```

Step 4 Run the following commands to run the test case:

```
{ time fastp -i ERR1044518_1.fastq -I ERR1044518_2.fastq -w 16 ; } 2>&1 |tee fastp.log
```

After the command is executed, the following information is displayed:

```
Duplication rate: 0.581538%
Insert size peak (evaluated by paired-end reads): 169
JSON report: fastp.json
HTML report: fastp.html
```

NOTE

fastp supports multi-thread running. **-w** specifies the number of threads, a maximum of 16 allowed.

After the command is executed, the **fast.json** and **fast.html** reports are generated. You can use the **-j** and **-h** parameters to specify the paths and names of the two files, respectively.

To view more parameter information, run the **fastp --help** command.

----End

4.32.8 More Resources

To obtain more resources, visit fastp at GitHub: **https://github.com/OpenGene/fastp**.

4.33 GCE 1.0.2 Porting Guide (CentOS 7.6)

4.33.1 Introduction

Genomic charactor estimator (GCE) is a bayes model based method to estimate the genome size, genomic repeat content, and the heterozygsis rate of the sequencing sample. The estimated result can be used to design the sequencing strategy.

For more information, visit the **GCE page at GitHub**.

Programming languages: Roff+Perl

Brief description: a bayes model based method to estimate genome characteristics

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

GCE 1.0.2

4.33.2 Environment Requirements

Hardware Requirements

Table 4-161 lists the hardware requirements.

Table 4-161 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 4-162 Software requirements

Item	Version	Download URL
CMake	3.17.5	https://cmake.org/download/
GCE	1.0.2	https://github.com/fanagislab/GCE/archive/master.tar.gz

OS Requirements

Table 4-163 lists the OS requirements.

Table 4-163 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.33.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GCE software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-164 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CMAKE	Installation path of CMake.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/GCE	Installation path of GCE.	

4.33.4 Configuring the Compilation Environment

4.33.4.1 Configuration Description

Prerequisites

Servers must be connected to the Internet. Otherwise, you need to download the installation packages to the local PC and use an SFTP tool to upload the installation packages to the corresponding directories on the servers.

Configuration Process

Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing CMake	See **4.33.4.2 Installing CMake**.

4.33.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to install dependencies:

```
yum install boost boost-devel openssl-devel
```

Step 3 Run the following command to decompress the CMake installation package:

```
tar -zxf cmake-3.17.5.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd cmake-3.17.5
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/CMAKE
```

Step 6 Run the following command to perform the compilation and installation:

```
make -j&& make install
```

Step 7 Run the following command to set the CMake environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.33.5 Obtaining Source Code

Procedure

Step 1 Download the GCE source code package from the following link:

https://github.com/fanagislab/GCE/archive/master.tar.gz

Step 2 Use an SFTP tool to upload the downloaded source code package to the */path/to/GCE* directory on the server.

----End

4.33.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the installation package and go to the installation directory:

```
cd /path/to/GCE
tar -zxf master.tar.gz&& cd GCE-master/gce-1.0.2/
```

Step 3 Run the following command to delete the compiled script:

```
rm -rf gce
```

Step 4 Run the following command to modify the configuration:

```
sed -i 's/g++/g++ -O3 -march=armv8.2-a -mtune=tsv110/g' Makefile
```

Step 5 Run the following command to perform the compilation and installation:

```
make
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.33.7 Running and Verifying GCE

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the installation directory of the executable file:

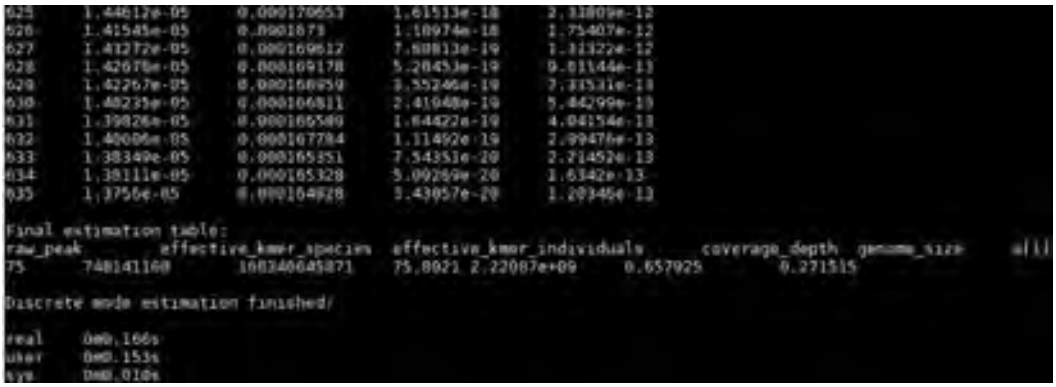
```
cd /home/GCE/gce-1.0.2
```

Step 3 Run the following command to run the test:

```
time ./gce -g 173854609857 -f test/Achatina_fulica/AF.kmer.freq.stat.2colum
```

Figure 4-15 shows the running result.

Figure 4-15 Running result



----End

4.33.8 More Resources

To obtain more resources, visit GCE at GitHub: **https://github.com/fanagislab/GCE**.

4.34 GEMMA 0.98.1 Porting Guide (CentOS 7.6)

4.34.1 Introduction

GEMMA is a software toolkit for fast application of linear mixed models (LMM) and related models to genome-wide association studies (GWAS) and other large-scale data sets.

For more information about GEMMA, visit the **GEMMA page at GitHub**.

Programming language: C++

Brief description: a software toolkit for fast application of linear mixed models (LMMs) and related models

Open-source license: GPL3.0

Recommended Version

GEMMA 0.98.1

4.34.2 Environment Requirements

Hardware Requirements

Table 4-165 lists the hardware requirements.

Table 4-165 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 4-166 Software requirements

Item	Version	Download Link
GEMMA	0.98.1	https://github.com/genetics-statistics/GEMMA/archive/0.98.1.tar.gz
cmake	3.17.3	https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz
OpenBLAS	0.3.9	https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz
eigen	3.3.8	https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz
gsl	2.5.0	https://codeload.github.com/ampl/gsl/tar.gz/v2.5.0

OS Requirements

Table 4-167 lists the OS requirements.

Table 4-167 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.34.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GEMMA software porting.

Table 4-168 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/GEMMA	Installation path of GEMMA.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CMAKE	Installation path of CMake.	
4	/path/to/OpenBLAS	Installation path of OpenBLAS.	
5	/path/to/EIGEN	Installation path of Eigen.	
6	/path/to/GSL	Installation path of GSL.	

4.34.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-169 Configuration process

No.	Configuration Item	Remarks
1	Setting Up the Basic Environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing CMake	For details, see **4.34.4.1 Installing CMake**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing OpenBLAS	For details, see **4.34.4.2 Installing OpenBLAS**.
4	Installing Eigen	For details, see **4.34.4.3 Installing Eigen**.
5	Installing GSL	For details, see **4.34.4.4 Installing GSL**.

4.34.4.1 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the CMake installation package:

tar zxvf cmake-3.17.3.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd cmake-3.17.3

Step 4 Run the following command to perform configuration:

./configure --prefix=*/path/to/CMAKE*

Step 5 Run the following commands to perform compilation and installation:

make

make install

Step 6 Run the following command to set CMake environment variables:

export PATH=*/path/to/CMAKE***/bin:$PATH**

----End

4.34.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

tar xzvf OpenBLAS-0.3.9.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd OpenBLAS-0.3.9/

Step 4 Run the following commands to set environment variables:

export CC=`which gcc`

 Copyright © Huawei Technologies Co., Ltd.

export CXX=`which g++`

export FC=`which gfortran`

Step 5 Run the following commands to perform compilation and installation:

make -j 32

make PREFIX=*/path/to/OPENBLAS* **install**

Step 6 Run the following commands to set OpenBLAS environment variables:

export LIBRARY_PATH=*/path/to/OPENBLAS***/lib:$LIBRARY_PATH**

export LD_LIBRARY_PATH=*/path/to/OPENBLAS***/lib:$LD_LIBRARY_PATH**

----End

4.34.4.3 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

tar xvf eigen-3.3.8.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd eigen-3.3.8

Step 4 Run the following commands to perform compilation and installation:

mkdir build

cd build

cmake -DCMAKE_INSTALL_PREFIX=*/path/to/EIGEN ../*

make install

----End

4.34.4.4 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the installation package:

tar xvf gsl-2.5.0.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd gsl-2.5.0

Step 4 Run the following commands to perform compilation and installation:

mkdir build

 Copyright © Huawei Technologies Co., Ltd.

cd build

../configure --prefix=*/path/to/GSL*

make -j 32

make install

Step 5 Run the following commands to set GSL environment variables:

export LIBRARY_PATH=*/path/to/GSL***/lib:$LIBRARY_PATH**

export LD_LIBRARY_PATH=*/path/to/GSL***/lib:$LD_LIBRARY_PATH**

export CPLUS_INCLUDE_PATH=*/path/to/GSL***/include:$CPLUS_INCLUDE_PATH**

export C_INCLUDE_PATH=*/path/to/GSL***/include:$C_INCLUDE_PATH**

----End

4.34.5 Obtaining Source Code

Procedure

Step 1 Download the GEMMA source code package **GEMMA-0.98.1.tar.gz** from the following link:

https://github.com/genetics-statistics/GEMMA/archive/0.98.1.tar.gz

Step 2 Use an SFTP tool to upload the source code package to the */path/to/GEMMA/* directory on the server.

----End

4.34.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install dependencies:

yum install blas-devel.aarch64 lapack-devel.aarch64 -y

Step 3 Run the following command to decompress the installation package:

tar -xvf GEMMA-0.98.1.tar.gz

Step 4 Run the following command to go to the directory generated after decompression:

cd GEMMA-0.98.1

Step 5 Run the following command to perform compilation and installation:

make EIGEN_INCLUDE_PATH=*/path/to/EIGEN* **/include/eigen3 WITH_OPENBLAS=1 GCC_FLAGS="-Wall" LIBS="***/path/to/GSL* **/lib/libgsl.a** */path/to/GSL* **/lib/libgslcblas.a -L***/path/to/OPENBLAS* **/lib -pthread -lopenblas -llapack -lblas -lz" OPENBLAS_INCLUDE_PATH=***/path/to/OPENBLAS* **/include -j 16**

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following command to set environment variables:

export PATH=*/path/to/GEMMA***/GEMMA-0.98.1/bin:$PATH**

----End

4.34.7 Running and Verifying GEMMA

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the installation directory of the executable file:

cd */path/to/GEMMA***/GEMMA-0.98.1/bin**

Step 3 Run the following command to run the test case:

./gemma -g ../example/mouse_hs1940.geno.txt.gz -p ../example/mouse_hs1940.pheno.txt -gk -o mouse_hs1940

After the command is run, the following information is displayed:

```
GEMMA 0.98.1 (2018-12-10) by Xiang Zhou and team (C) 2012-2018
Reading Files ...
## number of total individuals = 1940
## number of analyzed individuals = 1410
## number of covariates = 1
## number of phenotypes = 1
## number of total SNPs/var        =    12226
## number of analyzed SNPs         =    10768
Calculating Relatedness Matrix ...
================================================== 100%
**** INFO: Done.
```

----End

4.34.8 More Resources

To obtain more resources, visit GEMMA at GitHub: **https://github.com/genetics-statistics/GEMMA**.

4.35 kmergenie 1.7051 Porting Guide (CentOS 7.6)

4.35.1 Introduction

KmerGenie analyzes the k-mer and evaluates the genome. KmerGenie conducts the automatic analysis under multiple preset k-mers. In addition to regular k-mer frequency statistics, the size of the genome can be automatically calculated based on different k-mers, and an optimal k-mer for genome assembly can be evaluated as an alternative.

Brief description: a tool for k-mer analysis and genome size evaluation.

Programing language: C++/Python

Open source license: GPL

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

KmerGenie 1.7051

4.35.2 Environment Requirements

Hardware Requirements

Table 4-170 lists the hardware requirements.

Table 4-170 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table lists the software requirements.

Table 4-171 Software requirements

Item	Version	Download URL
kmergenie	1.7051	http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz
Python	2.7.5	Contained in the OS
R	3.3.1	https://mirrors.tuna.tsinghua.edu.cn/CRAN/src/base/R-3/R-3.3.1.tar.gz
Distribute	0.7.3	https://pypi.python.org/packages/source/d/distribute/distribute-0.7.3.zip --no-check-certificate
Test case	human_g1k_v37	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

OS Requirements

Table 4-172 lists the OS requirements.

Table 4-172 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.35.3 Paths for Software Porting

This chapter lists the software installation paths involved in the KmerGenie software porting.

Table 4-173 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/KMERGENIE	Installation path of KmerGenie.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/R	Installation path of R.	
5	/path/to/TESTCASE	Path for storing the test case.	

4.35.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-174 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.35.5 Obtaining the Source Code

Procedure

Step 1 Download the **kmergenie-1.7051.tar.gz** installation package.

http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz

Step 2 Download test case file **human_g1k_v37.fasta**.

http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

Step 3 Use an SFTP tool to upload the KmerGenie installation package to the */path/to/KMERGENIE* directory.

Step 4 Use an SFTP tool to upload the **human_g1k_v37.fasta** test case to the */path/to/TESTCASE* directory on the server.

----End

4.35.6 Compiling and Installing KmerGenie

4.35.6.1 Installing R

Procedure

Step 1 Run the following command to install dependencies:

```
yum install readline-devel libXt-devel libcurl-devel pcre-devel bzip2-devel -y
```

Step 2 Run the following command to decompress the R installation package:

```
tar zxvf R-3.3.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd R-3.3.1
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/R
```

Step 5 Run the following commands to perform the compilation:

```
make
make install
```

Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/R/bin:$PATH
```

Step 7 Run the following command to check whether R is successfully installed:

```
R
```

If information similar to the following is displayed, the installation is successful:

 Copyright © Huawei Technologies Co., Ltd.

```
[root@XA320V2-112 R-3.3.1]# R

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
```

----End

4.35.6.2 Installing Distribute

Procedure

Step 1 Run the following command to decompress the installation package:

```
unzip distribute-0.7.3.zip
```

Step 2 Run the following command to go to the directory generated after the decompression:

```
cd distribute-0.7.3
```

Step 3 Run the following command to install Distribute:

```
python setup.py install
```

If the following information is displayed, the installation is successful:

```
Installed /usr/lib/python2.7/site-packages/distribute-0.7.3-py2.7.egg
Processing dependencies for distribute==0.7.3
Searching for setuptools==0.9.8
Best match: setuptools 0.9.8
Adding setuptools 0.9.8 to easy-install.pth file
Installing easy_install script to /usr/bin
Installing easy_install-2.7 script to /usr/bin

Using /usr/lib/python2.7/site-packages
Finished processing dependencies for distribute==0.7.3
```

----End

4.35.6.3 Installing KmerGenie

Procedure

Step 1 Run the following command to decompress the installation package:

```
tar zxvf kmergenie-1.7051.tar.gz
```

Step 2 Run the following command to go to the directory generated after the decompression:

```
cd kmergenie-1.7051
```

Step 3 Run the following command to add compilation parameters:

1. Open **Makefile**.
   ```
   vi makefile
   ```

2. Press **i** to enter the insert mode and modify line 8 in the **makefile** file as follows:

   Before the modification:

   ```
   CXXFLAGS ?=-O4
   ```

   After the modification:

   ```
   CXXFLAGS ?=-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following commands to perform the compilation:

```
make
```

Step 5 Run the following command to set the environment variable:

```
export PATH=/path/to/KMERGENIE/kmergenie-1.7051:$PATH
```

----End

4.35.7 Running and Verifying KmerGenie

Procedure

Step 1 Run the following command to check the KmerGenie version:

```
kmergenie
```

The following version information is displayed:

```
[root@XA320V2-112 kmergenie-1.7051]# kmergenie
KmerGenie 1.7051

Usage:
    kmergenie <read_file> [options]

Options:
    --diploid    use the diploid model (default: haploid model)
    --one-pass   skip the second pass to estimate k at 2 bp resolution (default: two passes)
    -k <value>   largest k-mer size to consider (default: 121)
    -l <value>   smallest k-mer size to consider (default: 15)
    -s <value>   interval between consecutive kmer sizes (default: 10)
    -e <value>   k-mer sampling value (default: auto-detected to use ~200 MB memory/thread)
    -t <value>   number of threads (default: number of cores minus one)
    -o <prefix>  prefix of the output files (default: histograms)
    --debug      developer output of R scripts
    --orig-hist  legacy histogram estimation method (slower, less accurate)
```

Step 2 Run the following command to go to the directory where the test case is stored:

```
cd /path/to/TESTCASE
```

Step 3 Run the following command to decompress the test case file:

```
gzip -d human_g1k_v37.fasta.gz
```

Step 4 Run the following command to run the test case:

```
kmergenie -k 200 -t 96 human_g1k_v37.fasta
```

If the following information is displayed and multiple **.pdf** files are generated, the verification is successful:

 Copyright © Huawei Technologies Co., Ltd.

----End

4.36 LASTZ 1.04.03 Porting Guide (CentOS 7.6)

4.36.1 Introduction

LASTZ is a program for aligning DNA sequences. Originally designed to handle sequences the size of human chromosomes and from different species, it is also useful for sequences produced by Next-Generation Sequencing (NGS) technologies such as Roche 454.

For more information, visit the **official LASTZ website**.

Programing language: C/Python

Brief description: a program for aligning DNA sequences

Open source license: MIT

Recommended Software Version

LASTZ 1.04.03

4.36.2 Environment Requirements

Hardware Requirements

Table 4-175 lists the hardware requirements.

Table 4-175 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-176 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-176 Software requirements

Item	Version	Download URL
LASTZ	1.04.03	https://github.com/lastz/lastz/archive/1.04.03.tar.gz
Test case	hg38.chr4.fa	http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz
	galGal3.chr4.fa	http://hgdownload.soe.ucsc.edu/goldenPath/galGal3/chromosomes/chr4.fa.gz

OS Requirements

Table 4-177 lists the OS requirements.

Table 4-177 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.36.3 Paths for Software Porting

This chapter lists the software installation paths involved in the LASTZ software porting.

Table 4-178 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/LASTZ	Installation path of LASTZ.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing test cases.	

 Copyright © Huawei Technologies Co., Ltd.

4.36.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-179 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.36.5 Obtaining the Source Code

Procedure

Step 1 Download LASTZ installation package **lastz-1.04.03.tar.gz** and upload it to the */path/to/LASTZ* directory on the server.

https://codeload.github.com/lastz/lastz/tar.gz/1.04.03

Step 2 Use an SFTP tool to upload the LASTZ source code package to the */path/to/LASTZ* directory on the server.

Step 3 Run the following commands to download the **hg38.chr4.fa** and **galGal3.chr4.fa** test cases:

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz -O hg38. chr4.fa.gz
wget http://hgdownload.soe.ucsc.edu/goldenPath/galGal3/chromosomes/chr4.fa.gz -O galGal3. chr4.fa.gz
```

Step 4 Use an SFTP tool to upload the test cases **hg38.chr4.fa** and **galGal3.chr4.fa** to the **/path/to/CASE** directory on the server.

----End

4.36.6 Compiling and Installing LASTZ

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the LASTZ directory:

```
cd /path/to/LASTZ
```

Step 3 Run the following command to decompress the LASTZ installation package:

```
tar -xvf lastz-1.04.03.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd lastz-1.04.03
```

Step 5 Run the following commands to modify the configuration file:

1. Open **Makefile**.
   ```
   vim src/Makefile
   ```
2. Press **i** to enter the insert mode and change the values of **CC**, **definedForAll**, and **CFLAGS**.
   ```
   CC=/path/to/GNU/bin/gcc
   definedForAll = -Wall -Wextra -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE
   CFLAGS = -O3 -march=armv8.2-a -mtune=tsv110 -flto ${definedForAll} ${VERSION_FLAGS}
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to declare the **home** environment variable:

```
export HOME=/path/to/LASTZ
```

Step 7 Run the following commands to perform the compilation and installation:

```
make -j16
make install
```

Step 8 Run the following command to add the LASTZ environment variable:

```
export PATH=/path/to/LASTZ/lastz-distrib/bin:$PATH
```

Step 9 Run the following command to test the LASTZ environment:

```
lastz --help
```

The following figure shows that the environment is normal:

```
NOTE: the following list is not comprehensive.  The most up-to-date list is
      available at http://www.bx.psu.edu/~rsharris/lastz

  target[[start..end]]   spec/file containing target sequence (fasta, fastq,
                         nib, 2bit or hsx);  [start..end] defines a subrange of
                         the file
                         (use --help=files for more details)
  query[[start..end]]    spec/file containing query sequences;  if absent,
                         queries come from stdin (if needed)
  --self                 the target sequence is also the query
                         (this replaces the query file)
  --seed=match<length>   use a word with no gaps instead of a seed pattern
  --[no]transition[=2]   allow one or two transitions in a seed hit
                         (by default a transition is allowed)
  --step=<length>        set step length (default is 1)
  --strand=both          search both strands
  --strand=plus          search + strand only (matching strand of query spec)
  --strand=minus         search - strand only (opposite strand of query spec)
                         (by default both strands are searched)
  --ambiguous=n[,<penalty>] treat N as an ambiguous nucleotide
                         (by default N is treated as a sequence splicing
                          character)
  --ambiguous=iupac[,<penalty>] treat any ambiguous IUPAC-IUB character as a
                         completely ambiguous nucleotide
                         (by default any sequence file with B,D,H,K,M,R,S,V,W,Y
                          is rejected)
  --[no]gfextend         perform gap-free extension of seed hits to HSPs
                         (by default extension is performed)
```

----End

4.36.7 Running and Verifying LASTZ

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the **CASE** directory:

```
cd /path/to/CASE
```

Step 3 Run the following commands to decompress the test case packages:

```
gzip -d hg38.chr4.fa.gz
gzip -d galGal3.chr4.fa.gz
```

Step 4 Run the following command to run the test case:

```
time lastz hg38.chr4.fa galGal3.chr4.fa --notransition --step=20 --nogapped --format=maf > hg38_4_vs_galGal3_4-kp7265.maf
```

Step 5 Run the following command to view **hg38_4_vs_galGal3_4-kp7265.maf**:

```
vim hg38_4_vs_galGal3_4-kp7265.maf
```

Figure 4-16 is an example of the output result.

Figure 4-16 Result example

```
##maf version=1 scoring=lastz.v1.04.03
# lastz.v1.04.03 --notransition --step=20 --nogapped --format=maf
#
# hsp_threshold      = 3000
# gapped_threshold   = 3000
# x_drop             = 910
# y_drop             = 9400
# gap_open_penalty   = 400
# gap_extend_penalty = 30
#         A    C    G    T
#    A   91 -114  -31 -123
#    C -114  100 -125  -31
#    G  -31 -125  100 -114
#    T -123  -31 -114   91
a score=4178
s chr4 27586068 89 + 190214555 AAACACTCTACAGAGCTTGGGAATGGGATGACTGTCAGAA
s chr4  1196889 89 +  94230402 AAGCTCTTCAAAAAGCTCGGGACTGGGATGACTGGAAAGA
```

----End

4.36.8 Troubleshooting

Problem 1: An Error Is Reported During LASTZ Compilation

Symptom

An error is reported during LASTZ compilation. The error information is similar to "lastz.c:450:9: error: variable 'freeTargetRev' set but not used [-Werror=unused-but-set-variable]".

Possible Cause

During GCC compilation, if the **-Werror** parameter is configured in the **Makefile** file, all warnings are processed as errors.

Procedure

Delete the **-Werror** parameter from the **Makefile** file. For details, see **Step 5**.

 Copyright © Huawei Technologies Co., Ltd.

4.36.9 More Resources

To obtain more resources, visit the official LASTZ website:

https://github.com/lastz/lastz/tree/1.04.03

4.37 MAFFT Porting Guide (CentOS 7.6)

4.37.1 Introduction

MAFFT is a multiple sequence alignment program for UNIX-like operating systems. It offers a range of multiple alignment methods, L-INS-i (accurate; for alignment of <~200 sequences), FFT-NS-2 (fast; for alignment of <~30,000 sequences), etc.

For more information, visit the **official MAFFT website**.

Programming language: C

Brief description: a program used to create multiple sequence alignments of amino acid or nucleotide sequences based on the Fast Fourier Transform

Open source license: Apache License 2.0

Recommended Software Version

7.470 or later

4.37.2 Environment Requirements

Hardware Requirements

Table 4-180 lists the hardware requirements.

Table 4-180 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-181 lists the software requirements.

Table 4-181 Software requirements

Item	Version	Download URL
MAFFT	7.471-without-extensions	**https://mafft.cbrc.jp/alignment/software/mafft-7.471-without-extensions-src.tgz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
MAFFT	7.471-with-extensions	**https://mafft.cbrc.jp/alignment/software/mafft-7.471-with-extensions-src.tgz**
Test case	LSU rRNA	**https://mafft.cbrc.jp/alignment/software/ex1.txt**

OS Requirements

Table 4-182 lists the OS requirements.

Table 4-182 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

4.37.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MAFFT software porting.

Table 4-183 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/MAFFT	Installation path of MAFFT.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing test cases.	

 Copyright © Huawei Technologies Co., Ltd.

4.37.4 Obtaining the Source Code

Procedure

Step 1 Download the MAFFT installation packages: **mafft-7.471-with-extensions-src.tgz** and **mafft-7.471-without-extensions-src.tgz**.

Download addresses:

https://mafft.cbrc.jp/alignment/software/mafft-7.471-without-extensions-src.tgz

https://mafft.cbrc.jp/alignment/software/mafft-7.471-with-extensions-src.tgz

Step 2 Use an SFTP tool to upload the MAFFT installation packages to the */path/to/MAFFT* directory on the server.

----End

4.37.5 Compiling and Installing MAFFT

4.37.5.1 Basic Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/MAFFT
```

Step 3 Run the following command to copy the installation package to the main program installation directory:

```
cp mafft-7.471-without-extensions-src.tgz  /path/to/MAFFT
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/MAFFT
```

Step 5 Run the following command to decompress the installation package:

```
tar -xvf  mafft-7.471-without-extensions-src.tgz
```

Step 6 Run the following command to go to the directory generated after the decompression:

```
cd mafft-7.471-without-extensions/core
```

Step 7 Run the following commands to configure related files:

```
perl -p -i -e 's#PREFIX =.*#PREFIX = /PATH/to/MAFFT#' Makefile
sed -i 's/gcc/\`which gcc\`/g' Makefile
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110 -flto/g' Makefile
```

Step 8 Compile and install the main program of MAFFT.

1. Run the following commands to compile and install the main program:

```
make clean
make
make install
```

2. Run the following command to add the environment variable:

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/MAFFT/bin:$PATH
```

----End

4.37.5.2 (Optional) Installing MPI

Procedure

Step 1 Run the following command to go to the directory generated after the decompression:

```
cd mafft-7.471-without-extensions/MPI
```

Step 2 Run the following commands to configure related files:

```
perl -p -i -e 's#PREFIX =.*#PREFIX = /path/to/MAFFT #' Makefile
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110 -flto/g' Makefile
```

Step 3 Run the following commands to compile and install the MAFFT extensions:

```
make clean
make
make install
```

----End

4.37.5.3 (Optional) Installing MAFFT Extensions

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to copy the installation package to the main program installation directory:

```
cp mafft-7.471-with-extensions-src.tgz  /path/to/MAFFT
```

Step 3 Run the following command to decompress the installation package:

```
tar -xvf  mafft-7.471-with-extensions-src.tgz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd /path/to/MAFFT/mafft-7.471-with-extensions/extensions
```

Step 5 Run the following command to configure related files:

```
perl -p -i -e 's#PREFIX =.*#PREFIX = /PATH/to/MAFFT #' Makefile
```

Step 6 Run the following commands to compile and install the MAFFT extensions:

```
make clean
make
make install
```

----End

4.37.6 Running and Verifying MAFFT

Procedure

Step 1 Run the following command to go to the test directory:

```
cd /path/to/CASE
```

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to add the environment variable:

```
export PATH=/path/to/MAFFT/bin:$PATH
```

Step 3 Run the following command to check whether the software is correctly installed:

```
{ time MAFFT_N_THREADS_PER_PROCESS="1" MAFFT_MPIRUN="mpirun --allow-run-as-root -n 128" MAFFT_TMPDIR="/path/to/CASE" mafft --mpi --large --globalpair ex1.txt > kp_out; } 2>&1 | tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log
```

The following figure shows the execution result:

```
Progressive alignment ...
STEP    54 /58
Reallocating..done. *alloclen = 11410
STEP    58 /58
done.
tbfast (nuc) Version 7.471
alg=A, model=DNA200 (2), 1.53 (4.59), -0.00 (-0.00), noshift, amax=0.0
1 thread(s)


Strategy:
 G-large-INS-1 (Not tested.)
 ?

If unsure which option to use, try 'mafft --auto input > output'.
For more information, see 'mafft --help', 'mafft --man' and the mafft page.

The default gap scoring scheme has been changed in version 7.110 (2013 Oct).
It tends to insert more gaps into gap-rich regions than previous versions.
To disable this change, add the --leavegappyregion option.


real    0m38.018s
user    37m53.309s
sys     3m43.047s
```

----End

4.37.7 More Resources

To obtain more resources, visit the official website of MAFFT:

https://mafft.cbrc.jp/alignment/software/

4.38 megahit 1.2.9 Porting Guide (CentOS 7.6)

4.38.1 Introduction

MEGAHIT is an ultra-fast and memory-efficient Next-Generation Sequencing (NGS) assembler. It is optimized for metagenomes, but also works well on generic single genome assembly and single-cell assembly.

Programming language: C++/C/Python

Open source license: GPL 3.0

Recommended Software Version

MEGAHIT 1.2.9

 Copyright © Huawei Technologies Co., Ltd.

4.38.2 Environment Requirements

Hardware Requirements

Table 4-184 lists the hardware requirements.

Table 4-184 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-185 lists the software requirements.

Table 4-185 Software requirements

Item	Version	Download URL
MEGAHIT	1.2.9	https://codeload.github.com/voutcn/megahit/tar.gz/v1.2.9
Kunpeng Porting Advisor	2.3.0	https://www.hikunpeng.com/developer/devkit/porting-advisor/download?tab=0
sse2neon	1.5.0	https://codeload.github.com/DLTcollab/sse2neon/tar.gz/refs/tags/v1.5.0
Test case 1	SRR1976948	https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/SRR1976948.abundtrim.subset.pe.fq.gz
Test case 2	SRR1977249	https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/SRR1977249.abundtrim.subset.pe.fq.gz

OS Requirements

Table 4-186 lists the OS requirements.

Table 4-186 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	How to Obtain
Kernel	4.14.0-115	Included in the OS image.

4.38.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MEGAHIT software porting.

Table 4-187 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/MEGAHIT	Installation path of MEGAHIT.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PORT_ADVISOR	Installation path of the Kunpeng Porting Advisor.	
4	/path/to/SSE2NEON	Installation path of sse2neon.	

4.38.4 Configuring the Compilation Environment

4.38.4.1 Configuration Description

Prerequisites

Servers must be connected to the Internet. Otherwise, you need to download the installation packages to the local PC and use an SFTP tool to upload the installation packages to the corresponding directories on the servers.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-188 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the Kunpeng Porting Advisor	See "Installation" in **Kunpeng Porting Advisor**. The web mode is recommended.
3	Installing CMake	See **4.38.4.2 Installing CMake**.
4	Installing sse2neon	See **4.38.4.3 Installing sse2neon**.

4.38.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install dependencies, such as CMake:

```
yum install -y cmake gzip zlib bzip2
```

----End

4.38.4.3 Installing sse2neon

Procedure

Step 1 sse2neon exists as a header file. You can download and decompress it.

https://codeload.github.com/DLTcollab/sse2neon/tar.gz/refs/tags/v1.5.0

Step 2 Use an SFTP tool to upload the sse2neon source code package to the **/path/to/SSE2NEON** directory on the server.

Step 3 Run the following command to decompress the sse2neon source code package:

```
tar zxvf sse2neon-1.5.0.tar.gz
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.38.5 Obtaining the Source Code

Procedure

Step 1 Download the MEGAHIT source code package.

https://codeload.github.com/voutcn/megahit/tar.gz/v1.2.9

Step 2 Use the SFTP tool to upload the MEGAHIT source code package to the */path/to/MEGAHIT* directory on the server.

----End

4.38.6 Compiling and Installing MEGAHIT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the source code installation directory:

```
cd /path/to/MEGAHIT
```

Step 3 Run the following command to decompress the source code package:

```
tar -xvf v1.2.9.tar.gz
```

Step 4 Run the following command to switch to the folder of the decompressed files:

```
cd megahit-1.2.9
```

Step 5 Run the following command to modify the **CMakeLists.txt** file:

1. Open **CMakeLists.txt**.
   ```
   vi CMakeLists.txt
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:
   ```
   6    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
   ...
   76 set_target_properties(megahit_core PROPERTIES COMPILE_FLAGS "-mbmi2 -DUSE_BMI2 -
   mpopcnt")
   77 set_target_properties(megahit_core_popcnt PROPERTIES COMPILE_FLAGS "-mpopcnt")
   ```

   After the modification:
   ```
   6    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -march=armv8.2-a")
   ...
   76 set_target_properties(megahit_core PROPERTIES COMPILE_FLAGS "")
   77 set_target_properties(megahit_core_popcnt PROPERTIES COMPILE_FLAGS "")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following command to switch the directory:

```
cd ./src/utils/
```

Step 7 Run the following command to copy the header file of the Kunpeng Porting Advisor to the current directory:

```
cp /path/to/PORT_ADVISOR/portadv/tools/inline_asm/config/KunpengTrans.h ./
```

Step 8 Run the following command to modify **cpu_dispatch.h**:

```
vi cpu_dispatch.h
```

The new code content has been marked:

 Copyright © Huawei Technologies Co., Ltd.

```
#if defined(__aarch64__)      // New code
  #include "KunpengTrans.h"     // New code
#endif      // New code

#ifndef MEGAHIT_CPU_DISPATCH_H
#define MEGAHIT_CPU_DISPATCH_H

inline bool HasPopcnt() {
  unsigned eax, ebx, ecx, edx;
#ifdef _MSC_VER
  int cpuid[4];
  __cpuid(cpuid, 1);
  eax = cpuid[0], ebx = cpuid[1], ecx = cpuid[2], edx = cpuid[3];
#else
#if defined(__x86_64__)    // New code
  asm volatile("cpuid\n\t"
               : "=a"(eax), "=b"(ebx), "=c"(ecx), "=d"(edx)
               : "0"(1));
#elif defined(__aarch64__)    // New code
  {
    eax = 1;      // New code
    GetSupportedCPUID(&eax,&ebx,&ecx,&edx);     // New code
  }
#endif  // New code
#endif
  return ecx >> 23U & 1U;
}

inline bool HasBmi2() {
  unsigned eax, ebx, ecx, edx;
#ifdef _MSC_VER
  int cpuid[4];
  __cpuidex(cpuid, 7, 0);
  eax = cpuid[0], ebx = cpuid[1], ecx = cpuid[2], edx = cpuid[3];
#else
#if defined(__x86_64__)    // New code
  asm volatile("cpuid\n\t"
               : "=a"(eax), "=b"(ebx), "=c"(ecx), "=d"(edx)
               : "0"(7), "2"(0));
#elif defined(__aarch64__)        // New code
  {
    eax = 7;    // New code
    ecx = 0;    // New code
    GetSupportedCPUID(&eax,&ebx,&ecx,&edx);    // New code
  }
#endif    // New code
#endif
  return ebx >> 8U & 1U;
}

#endif  // MEGAHIT_CPU_DISPATCH_H
```

Step 9 Run the following command to switch the directory:

```
cd ../kmlib/
```

Step 10 Run the following command to modify the **kmrns.h** file and replace the header file:

```
vi kmrns.h
```

Before the modification:

```
#include <x86intrin.h>
```

After the modification:

```
#include <sse2neon.h>
```

Step 11 Run the following command to copy the **sse2neon.h** header file to the current directory:

```
cp /path/to/SSE2NEON/sse2neon-1.5.0/sse2neon.h ./
```

 Copyright © Huawei Technologies Co., Ltd.

Step 12 Run the following command to create a **build** folder and go to the folder:

```
mkdir /path/to/MEGAHIT/build && cd /path/to/MEGAHIT/build
```

Step 13 Run the following commands to perform the compilation and installation:

```
CC=`which gcc` CXX=`which g++` cmake -DCMAKE_INSTALL_PREFIX=/path/to/MEGAHIT -DCMAKE_BUILD_TYPE=Release ../
make -j 16
make install
```

Step 14 Run the following command to load the environment variable:

```
export PATH=/path/to/MEGAHIT/bin:$PATH
```

----End

4.38.7 Running and Verifying MEGAHIT

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the source code installation directory:

```
cd /path/to/MEGAHIT
```

Step 3 Run the following command to create a test case folder and go to the folder:

```
mkdir testcase && cd testcase
```

Step 4 Upload the downloaded test case file to the current directory.

Step 5 Run the following command to run the test case:

```
megahit -t 128 --12 SRR1976948.abundtrim.subset.pe.fq.gz SRR1977249.abundtrim.subset.pe.fq.gz > megahit.log
```

After the execution is complete, check the value of **Time elapsed** (unit: second) in the **megahit.log** file. A smaller value indicates better performance. The following is an example of the output result:

```
.
.
.
2020-11-19 16:21:34 - Merging to output final contigs
2020-11-19 16:21:34 - 3227 contigs, total 10298772 bp, min 200 bp, max 301573 bp, avg 3191 bp, N50 6898 bp
2020-11-19 16:21:35 - ALL DONE. Time elapsed: 126.348002 seconds
```

----End

4.38.8 More Resources

To obtain more resources, visit the MEGAHIT page of GitHub:

https://github.com/voutcn/megahit

4.39 PASA 2.4.1 Porting Guide (CentOS 7.6)

4.39.1 Introduction

Program to Assemble Spliced Alignments (PASA) is a eukaryotic genome annotation tool that exploits spliced alignments of expressed transcript sequences

 Copyright © Huawei Technologies Co., Ltd.

to automatically model gene structures and to maintain gene structure annotation consistent with the most recently available sequence data. PASA can also identify and classify all splicing variations supported by the transcript alignments.

For more information, visit the **PASA official website**.

Programming language: Perl/C++

Brief description: A eukaryotic genome annotation tool

Recommended Software Version

PASA 2.4.1

4.39.2 Environment Requirements

Hardware Requirements

Table 4-189 lists the hardware requirements.

Table 4-189 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-190 lists the software requirements.

Table 4-190 Software requirements

Item	Version	Download URL
PASA	2.4.1	**https://github.com/PASApipeline/PASApipeline/archive/pasa-v2.4.1.tar.gz**
URI	1.35	**http://search.cpan.org/CPAN/authors/id/G/GA/GAAS/URI-1.35.tar.gz**
blat	35.1	**https://codeload.github.com/djhshih/blat/tar.gz/v35.1**
gmap	2015-09-21	**http://research-pub.gene.com/gmap/src/gmap-gsnap-2015-09-21.tar.gz**
samtools	0.1.9	**https://codeload.github.com/samtools/samtools/tar.gz/0.1.9**
mariadb	5.5.60	Contained in the CentOS 7.6 image
Test case	genome_sample.fasta	Provided by the software

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
	all_transcripts.fasta.clean	
	all_transcripts.fasta	
	FL_accs.txt	

OS Requirements

Table 4-191 lists the OS requirements.

Table 4-191 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.39.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PASA software porting.

Table 4-192 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/BLAT	Installation path of Blat.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/URI	Installation path of URI.	
4	/path/to/PASA	Installation path of PASA.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	/path/to/SAMTOOLS	Installation path of Samtools.	
6	/path/to/GMAP	Installation path of GMAP.	See **GMAP 2015.9.21 Porting Guide (CentOS 7.6)**.

4.39.4 Configuring the Compilation Environment

4.39.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-193 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Perl	See **4.39.4.2 Installing Perl**.
3	Installing URI	See **4.39.4.3 Installing URI**.
4	Installing Blat	See **4.39.4.4 Installing Blat**.
5	Installing MariaDB	See **4.39.4.5 Installing MariaDB**.
6	Installing Samtools	See **4.39.4.6 Installing Samtools**.

4.39.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to install Perl:

```
yum install perl* -y
```

----End

4.39.4.3 Installing URI

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the URI installation package:

```
tar -xvf URI-1.35.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd URI-1.35
```

Step 4 Run the following commands to perform the compilation and installation:

```
perl Makefile.PL
make
make install
```

----End

4.39.4.4 Installing Blat

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install libpng-devel -y
```

Step 3 Run the following command to decompress the Blat installation package:

```
tar zxvf blat-35.1.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd blat-35.1/
```

Step 5 Run the following command to declare the installation environment:

```
export MACHTYPE=aarch64
```

Step 6 Run the following command to perform the compilation and installation:

```
make
```

Step 7 Run the following command to declare the Blat environment variable:

```
export PATH=/path/to/BLAT/blat-35.1/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.39.4.5 Installing MariaDB

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install MariaDB:

```
yum install mariadb* -y
```

Step 3 Run the following commands to modify the **Makefile** file:

1. Open **my.cnf**.
   ```
   vim /etc/my.cnf
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:

   ```
   [mysqld]
   ```

   After the modification:

   ```
   [mysqld]
   skip-grant-tables
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to start the database:

```
systemctl start mariadb.service
```

Step 5 Run the following command to access the database:

```
mysql -u root
```

Step 6 Run the following command to access MySQL:

```
use mysql;
```

Step 7 Run the following commands to modify the plugin attribute:

```
update user set authentication_string=PASSWORD(''), plugin='mysql_native_password' where
user='root';
flush privileges;
```

Step 8 Run the following commands to create a user with the read-only permission and a user with all permissions:

```
MariaDB [(none)]> grant select on *.* to 'test'@'%' identified by '123456';
MariaDB [(none)]> grant all on *.* to 'pasa'@'%' identified by '123456';
MariaDB [(none)]> flush privileges;
```

Step 9 After the creation is complete, enter **exit**.

Step 10 Run the following commands to check whether the users are successfully created:

```
mysql -upasa -p123456
```

The users are successfully installed if the following information is displayed:

```
[root@ARMCI sample_data]# mysql -upasa -p123456
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 5.5.65-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.39.4.6 Installing Samtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Run the following command to decompress the Samtools installation package:

```
tar -xvf samtools-0.1.9.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd samtools-0.1.9
```

Step 5 Run the following command to obtain the absolute path of GNU:

```
which gcc
```

Step 6 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

   Before the modification:
   ```
   CC       =gcc
   ```
   After the modification:
   ```
   CC       =path/to/GUN/bin/gcc
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to perform the compilation and installation:

```
make
```

Step 8 Run the following command to add the Samtools environment variable:

```
export PATH=/path/to/SAMTOOLS/samtools-0.1.9:$PATH
```

----End

4.39.5 Obtaining the Source Code

Procedure

Step 1 Download the PASA installation package **pasa-v2.4.1.tar.gz**.

https://github.com/PASApipeline/PASApipeline/archive/pasa-v2.4.1.tar.gz

Step 2 Use an SFTP tool to upload the PASA package to the */path/to/PASA* directory.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.39.6 Compiling and Installing PASA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PASA package:

```
tar -xvf pasa-v2.4.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd PASApipeline-pasa-v2.4.1
```

Step 4 Run the following command to perform the compilation and installation:

```
make -j16
```

Step 5 Run the following command to declare the environment variable:

```
export PATH=/path/to/PASA/PASApipeline-pasa-v2.4.1:$PATH
```

----End

4.39.7 Running and Verifying PASA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to add the GMAP environment variable:

```
export PATH=/path/to/GMAP/Install/bin:$PATH
```

Step 3 Run the following commands to modify the configuration file:

1. Open the configuration file.

   ```
   cp /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/pasa.CONFIG.template /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/conf.txt
   vim /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/conf.txt
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   ```
   MYSQLSERVER=localhost
   MYSQL_RO_USER=test
   MYSQL_RO_PASSWORD=123456
   MYSQL_RW_USER=pasa
   MYSQL_RW_PASSWORD=123456
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to go to the directory where the test case is stored:

```
cd /path/to/PASA/PASApipeline-pasa-v2.4.1/sample_data
```

Step 5 Run the following commands to modify the configuration file:

```
cp /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/pasa.alignAssembly.Template.txt alignAssembly.config
vi alignAssembly.config
DATABASE=sample_data
```

Enter **:wq!** to save the change and exit.

Step 6 Run the following commands to run the test case:

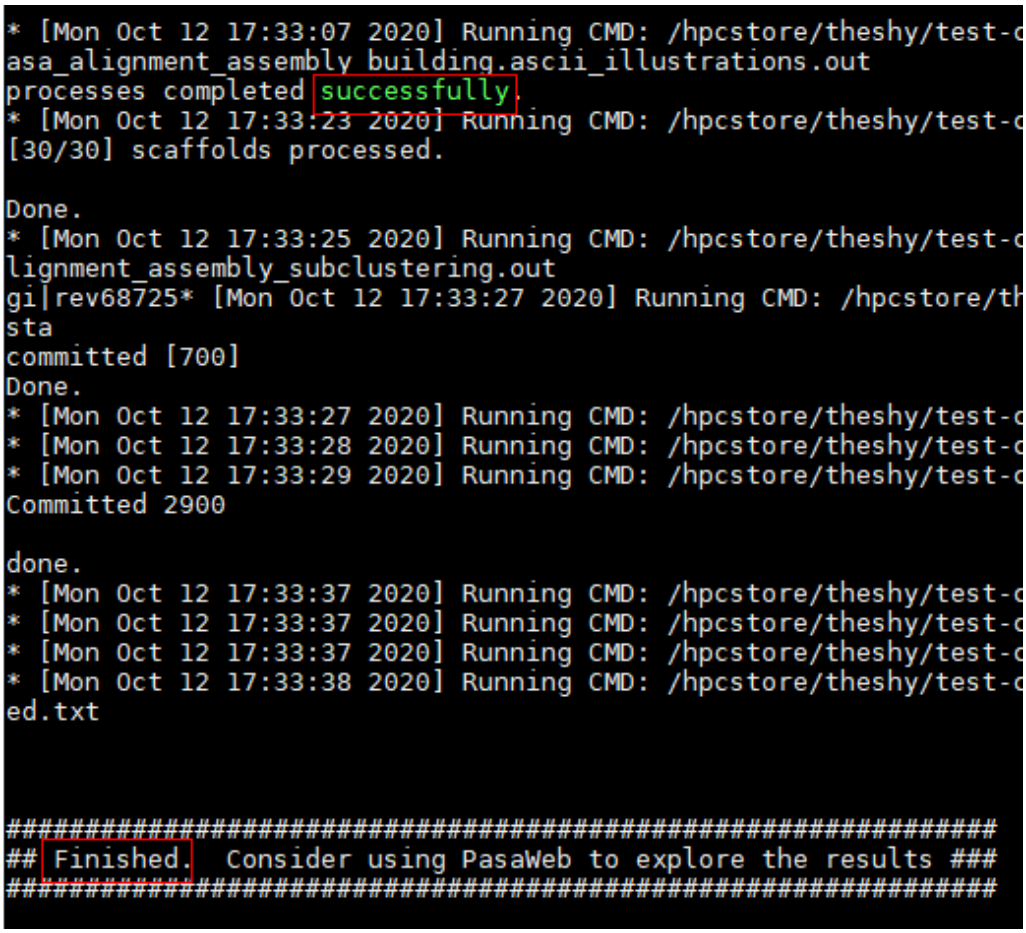
```
gzip -d genome_sample.fasta.gz
{ time Launch_PASA_pipeline.pl -c  alignAssembly.config -C -R -g genome_sample.fasta -t all_transcripts.fasta.clean -T -u all_transcripts.fasta -f FL_accs.txt --ALIGNERS blat,gmap --CPU 96; } 2>&1 |tee PASA.log
```

 Copyright © Huawei Technologies Co., Ltd.

View the **PASA.log** file. If **Finished** is recorded in the log file, the PASA process ends properly.

Figure 4-17 shows an example of the running result.

Figure 4-17 Result example



----End

4.39.8 Troubleshooting

Problem 1: An error is reported when MariaDB verifies the user.

Symptom

An error message similar to "ERROR 1045 (28000): Access denied for user 'test'@'localhost' (using password: YES)" is displayed when MariaDB verifies the user.

Possible Cause

The default plugin attribute is incorrect.

Procedure:

See **Step 7**.

Problem 2: An error is reported when MariaDB creates a database.

Symptom

MariaDB verifies the user successfully, but an error message similar to "ERROR 1045 (28000): Access denied for user 'pasa'@'localhost' (using password: YES)" during database creation.

Possible Cause

The permission of the **paas** user is incorrect.

Procedure:

Run the following commands: **mysql -upasa -p123456**
use mysql;
insert into mysql.user(Host,User,Password) values("localhost","pasa",password("123456"));

Problem 3: An error is reported during the PASA running.

Symptom

An error message similar to "Can't locate URI/Escape.pm in @INC" is displayed during the PASA running.

Possible Cause

The URI is missed in Perl.

Procedure

See **4.39.4.3 Installing URI**.

4.39.9 More Resources

PASA official website:

https://github.com/PASApipeline/PASApipeline/wiki

4.40 Augustus 3.3.3 Porting Guide (CentOS 7.6)

4.40.1 Introduction

AUGUSTUS is an open-source program that predicts genes in eukaryotic genomic sequences. It is the most accurate *ab initio* program whose accuracy is dependent on the quality of the training gene sets used for specific parameters of species.

For more information, please visit **http://bioinf.uni-greifswald.de/augustus/**.

Programming language: C++

Brief description: Gene prediction

Recommended Version

Augustus 3.3.3

 Copyright © Huawei Technologies Co., Ltd.

4.40.2 Environment Requirements

Hardware Requirements

Table 4-194 lists the hardware requirements.

Table 4-194 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-195 lists the software requirements.

Table 4-195 Software requirements

Item	Version	Download URL
cmake	3.8.1	https://cmake.org/download/
htslib	1.11	https://github.com/samtools/htslib/archive/1.11.tar.gz
bcftools	1.11	https://github.com/samtools/bcftools/archive/1.11.tar.gz
samtools	1.11	https://github.com/samtools/samtools/archive/1.11.tar.gz
bamtools	2.5.0	https://github.com/pezmaster31/bamtools/archive/v2.5.0.tar.gz
Augustus	3.3.3	https://github.com/Gaius-Augustus/Augustus/archive/v3.3.3.tar.gz
Test case	sag178	http://bioinf.uni-greifswald.de/augustus/datasets/sag178.gb.gz

OS Requirements

Table 4-196 lists the OS requirements.

Table 4-196 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.40.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the AUGUSTUS software porting.

Table 4-197 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	For details, see "Planning Data for Installation" in the **HPC Solution Basic Environment Setup Guide**.
2	/path/to/AUGUSTUS/htslib	Installation path of HTSlib.	The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/AUGUSTUS/bcftools	Installation path of BCFtools.	
4	/path/to/AUGUSTUS/samtools	Installation path of SAMtools.	
5	/path/to/AUGUSTUS/cmake	Installation path of CMake.	
6	/path/to/AUGUSTUS/bamtools	Installation path of Bamtools.	
7	/path/to/AUGUSTUS/augustus	Installation path of AUGUSTUS.	
8	/path/to/AUGUSTUS/CASE	Path for storing cases.	

4.40.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-198 Configuration process

No.	Configuration Item	Remarks
1	Set up the basic environment.	For details, see "Setting Up the Environment for the Single-Node Scenario" in the **HPC Solution Basic Environment Setup Guide**.
2	Install CMake.	For details, see **4.40.4.1 Installing CMake**.
3	Install HTSlib.	For details, see **4.40.4.2 Installing HTSlib**.
4	Install BCFtools.	For details, see **4.40.4.3 Installing BCFtools**.
5	Install SAMtools.	For details, see **4.40.4.4 Installing SAMtools**.
6	Install Bamtools.	For details, see **4.40.4.5 Installing Bamtools**.

4.40.4.1 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use the SFTP tool to upload the CMake installation package to the */path/to/CMAKE* directory.

Step 3 Run the following command to decompress the CMake installation package:

cd */path/to*/**AUGUSTUS && tar -zxf cmake-3.8.1.tar.gz**

Step 4 Run the following command to switch to the directory generated after decompression:

cd cmake-3.8.1

Step 5 Run the following command to perform configuration:

./configure --prefix=*/path/to*/**AUGUSTUS/cmake**

Step 6 Run the following commands to perform compilation and installation:

make

make install

Step 7 Run the following command to set CMake environment variables:

 Copyright © Huawei Technologies Co., Ltd.

export PATH=*/path/to/***cmake/bin:$PATH**

----End

4.40.4.2 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use the SFTP tool to upload the HTSlib installation package to the */path/to/AUGUSTUS* directory.

Step 3 Run the following command to install dependencies:

yum install -y libcurl libcurl-devel openssl openssl-devel zlib-devel bzip2 bzip2-devel xz-devel boost-devel

Step 4 Run the following command to decompress the package:

cd */path/to***/AUGUSTUS && tar -zxf htslib-1.11.tar.gz**

Step 5 Run the following command to rename and go to the installation directory:

mv htslib-1.11 htslib && cd htslib

Step 6 Run the following commands to perform configuration:

autoheader

autoconf

sed -i 's/\-O2/\-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "\-O2" ./`

CC=`which gcc` ./configure --prefix=*/path/to***/AUGUSTUS/htslib**

Step 7 Run the following commands to perform compilation and installation:

make -j 16

make install

----End

4.40.4.3 Installing BCFtools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use the SFTP tool to upload the BCFtools installation package to the */path/to/AUGUSTUS* directory.

Step 3 Run the following command to decompress the package:

cd */path/to/AUGUSTUS* **&& tar -zxf bcftools-1.11.tar.gz**

Step 4 Run the following command to rename and go to the installation directory:

mv bcftools-1.11 bcftools && cd bcftools

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to perform configuration:

autoheader

autoconf

sed -i 's/\-O2/\-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "\-O2" ./`

CC=`which gcc` ./configure --prefix=*/path/to/AUGUSTUS***/bcftools**

Step 6 Run the following commands to perform compilation and installation:

make -j 16

make install

----End

4.40.4.4 Installing SAMtools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install dependencies:

yum install -y ncurses ncurses-base ncurses-libs ncurses-devel

Step 3 Use the SFTP tool to upload the SAMtools installation package to the */path/to/AUGUSTUS* directory.

Step 4 Run the following command to decompress the package:

cd */path/to/AUGUSTUS* **&& tar -zxf samtools-1.11.tar.gz**

Step 5 Run the following command to rename and go to the installation directory:

mv samtools-1.11 samtools && cd samtools

Step 6 Run the following commands to perform configuration:

autoheader

autoconf

sed -i 's/\-O2/\-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "\-O2" ./`

CC=`which gcc` ./configure --prefix=*/path/to/AUGUSTUS***/samtools**

Step 7 Run the following commands to perform compilation and installation:

make -j 16

make install

----End

 Copyright © Huawei Technologies Co., Ltd.

4.40.4.5 Installing Bamtools

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use the SFTP tool to upload the Bamtools installation package to the */path/to/AUGUSTUS* directory.

Step 3 Run the following command to decompress the package:

cd */path/to/AUGUSTUS* **&& tar -zxf bamtools-2.5.0.tar.gz**

Step 4 Run the following command to rename and go to the installation directory:

mv bamtools-2.5.0 bamtools && cd bamtools

Step 5 Run the following commands to perform configuration:

mkdir build && cd build

cmake -DCMAKE_INSTALL_PREFIX=../ ..

Step 6 Run the following commands to perform compilation and installation:

make -j 16

make install

----End

4.40.5 Obtaining the Source Code

Procedure

Step 1 Download the Augustus installation package **Augustus-3.3.3.tar.gz** at

https://github.com/Gaius-Augustus/Augustus/archive/v3.3.3.tar.gz

Step 2 Use the SFTP tool to upload the AUGUSTUS installation package to the */path/to/AUGUSTUS* directory.

----End

4.40.6 Compiling and Installing AUGUSTUS

Procedure

Step 1 Run the following command to go to the installation directory and decompress the software package:

cd */path/to/AUGUSTUS* **&& tar -zxf Augustus-3.3.3.tar.gz && cd Augustus-3.3.3**

Step 2 Run the following commands to perform configuration:

export TOOLDIR=*/path/to/AUGUSTUS*

export LD_LIBRARY_PATH=*/path/to/AUGUSTUS***/bamtools/lib64:$LD_LIBRARY_PATH**

 Copyright © Huawei Technologies Co., Ltd.

export LIBRARY_PATH=*/path/to/AUGUSTUS***/bamtools/lib64:$LIBRARY_PATH**

sed -i 's/\/usr\/include\/bamtools/*\/path\/to\/AUGUSTUS***|/bamtools\/include\/bamtools/g' `grep -rl "/usr/include/bamtools" ./`**

Step 3 Run the following command to set the installation path:

sed -i 's/\/opt\/augustus/*\/path\/to\/AUGUSTUS***\/augustus_aarch64/g' Makefile**

Step 4 Run the following commands to perform compilation and installation:

CC=`which gcc` CXX=`which g++` CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' make -j

make install

----End

4.40.7 Running and Verifying AUGUSTUS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to set environment variables:

export PATH=*/path/to/AUGUSTUS***/augustus_aarch64-3.3.3/bin:$PATH**

Step 3 Run the following commands to run the test case:

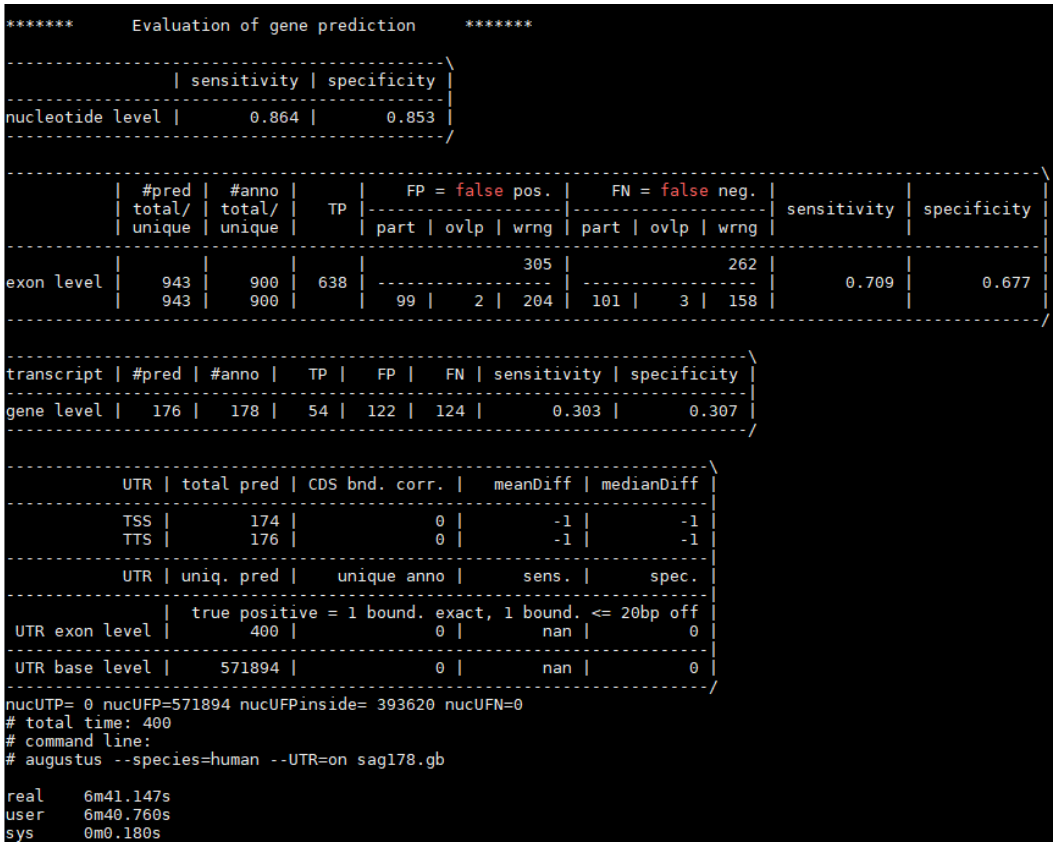
cd */path/to/AUGUSTUS***/CASE**

{ time augustus --species=human --UTR=on sag178.gb; } 2>&1 | tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log

Figure 4-18 shows an example of the running result.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-18 Result example



```
*******      Evaluation of gene prediction     *******

---------------------------------------\
               | sensitivity | specificity |
---------------------------------------|
nucleotide level |       0.864 |       0.853 |
---------------------------------------/

-------------------------------------------------------------------------------------------------\
          |  #pred |  #anno |      |    FP = false pos. |    FN = false neg. |             |             |
          | total/ | total/ |   TP |--------------------|--------------------| sensitivity | specificity |
          | unique | unique |      | part | ovlp | wrng | part | ovlp | wrng |             |             |
-------------------------------------------------------------------------------------------------|
          |        |        |      |                305 |                262 |             |             |
exon level |    943 |    900 |  638 | ------------------ | ------------------ |       0.709 |       0.677 |
          |    943 |    900 |      |   99 |    2 |  204 |  101 |    3 |  158 |             |             |
-------------------------------------------------------------------------------------------------/

-----------------------------------------------------------------------\
transcript | #pred | #anno |   TP |   FP |   FN | sensitivity | specificity |
-----------------------------------------------------------------------|
gene level |   176 |   178 |   54 |  122 |  124 |       0.303 |       0.307 |
-----------------------------------------------------------------------/

-----------------------------------------------------------------------\
           UTR | total pred | CDS bnd. corr. |   meanDiff | medianDiff |
-----------------------------------------------------------------------|
           TSS |        174 |              0 |         -1 |         -1 |
           TTS |        176 |              0 |         -1 |         -1 |
-----------------------------------------------------------------------|
           UTR | uniq. pred |    unique anno |      sens. |      spec. |
-----------------------------------------------------------------------|
               |  true positive = 1 bound. exact, 1 bound. <= 20bp off |
 UTR exon level |        400 |              0 |        nan |          0 |
-----------------------------------------------------------------------|
 UTR base level |     571894 |              0 |        nan |          0 |
-----------------------------------------------------------------------/
nucUTP= 0 nucUFP=571894 nucUFPinside= 393620 nucUFN=0
# total time: 400
# command line:
# augustus --species=human --UTR=on sag178.gb

real    6m41.147s
user    6m40.760s
sys     0m0.180s
```

----End

4.41 Delly 0.8.5 Porting Guide (CentOS 7.6)

4.41.1 Introduction

Delly is an integrated structural variant (SV) prediction method that can discover, genotype and visualize deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data. It uses paired-ends, split-reads, and read-depth to sensitively and accurately delineate genomic rearrangements throughout the genome.

For more information about Delly, visit the **Delly official website**.

Programming language: C++

Brief description: an integrated structural variant (SV) prediction method

Recommended Software Version

Delly 0.8.5

 Copyright © Huawei Technologies Co., Ltd.

4.41.2 Environment Requirements

Hardware Requirements

Table 4-199 lists the hardware requirements.

Table 4-199 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-200 lists the software requirements.

Table 4-200 Software requirements

Item	Version	Download Link
Boost	1.66	https://github.com/boostorg/boost/archive/refs/tags/boost-1.66.0.tar.gz
samtools	1.10	https://github.com/samtools/samtools/archive/1.10.tar.gz
Bcftools	1.10.2	https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2
BWA	0.7.17	https://sourceforge.net/projects/bio-bwa/files/
Delly	0.8.5	https://github.com/dellytools/delly/archive/v0.8.5.tar.gz
htslib	1.10.2	https://github.com/samtools/htslib/archive/1.10.2.tar.gz
Test cases	hg19.fa	https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/open_database/hg19/gatk_bundle/hg19.fa
	B17NC_R1.fq.gz	https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz

OS Requirements

Table 4-201 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-201 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.41.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Delly software porting.

Table 4-202 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/DELLY/delly-0.8.5	Installation path of Delly.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/DELLY/boost	Installation path of Boost.	
4	/path/to/DELLY/CASE	Path for storing test cases.	

4.41.4 Configuring the Compilation Environment

4.41.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-203 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing Boost	See **4.41.4.2 Installing Boost**.
3	Installing Samtools	See "Installing SAMtools" in **Augustus 3.3.3 Porting Guide (CentOS 7.6)**.
4	Installing bcftools	See **bcftools 1.10.2 Porting Guide (CentOS 7.6)**.
5	Installing BWA	See **BWA 0.7.17 Porting Guide (CentOS 7.6)**.

4.41.4.2 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Use an SFTP tool to upload the **boost_1_66_0.tar.gz** installation package to the */path/to/DELLY* directory on the server.

Step 3 Run the following command to decompress the installation package and go to the directory generated after the decompression:

```
cd /path/to/DELLY && tar -zxf boost_1_66_0.tar.gz && cd boost_1_66_0
```

Step 4 Run the following command to perform configuration:

```
./bootstrap.sh --prefix=/path/to/DELLY/boost
```

Step 5 Run the following command to perform compilation and installation:

```
./b2 -j16 && ./b2 install
```

----End

4.41.5 Obtaining Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Download the Delly installation package **v0.8.5.tar.gz** and the HTSlib installation package **1.10.2.tar.gz**.

Download links:

 Copyright © Huawei Technologies Co., Ltd.

Delly installation package: **https://github.com/dellytools/delly/archive/v0.8.5.tar.gz**

HTSlib installation package: **https://github.com/samtools/htslib/archive/1.10.2.tar.gz**

Step 3 Use an SFTP tool to upload the Delly and HTSlib installation packages to the */path/to/AUGUSTUS* directory on the server.

----End

4.41.6 Compilation and Installation

Procedure

Step 1 Run the following command to clear the environment:

```
yum erase `yum list installed |grep boost|awk {'print $1'}` -y
```

Step 2 Run the following command to install dependencies:

```
yum install -y libcurl libcurl-devel openssl openssl-devel zlib-devel bzip2 bzip2-devel xz-devel
```

Step 3 Run the following commands to set environment variables:

```
export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:/path/to/boost/include
export LIBRARY_PATH=$LIBRARY_PATH:/path/to/boost/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/boost/lib
```

Step 4 Run the following command to decompress the Delly and HTSlib installation packages:

```
cd /path/to/DELLY && tar -zxf v0.8.5.tar.gz && tar -zxf 1.10.2.tar.gz
```

Step 5 Run the following command to move the HTSlib source code to the **src/htslib** directory of the Delly source code:

```
mv htslib-1.10.2/* delly-0.8.5/src/htslib/
```

Step 6 Run the following commands to modify the **Makefile** file:

```
cd /path/to/DELLY/delly-0.8.5
sed -i 's/g++/`which g++`/g' Makefile
sed -i 's/\.\/configure/CC=`which gcc` CXX=`which g++` \.\/configure/g' Makefile
sed -i 's/O3/O3 -std=c++11 -march=armv8.2-a -mtune=tsv110/g' Makefile
```

Step 7 Run the following commands to perform compilation:

```
CC=`which gcc` CXX=`which g++` make PARALLEL=1 all
```

NOTE

PARALLEL=1 indicates that the multi-thread mode is enabled. If this parameter is removed, the multi-thread mode is enabled.

Step 8 Run the following command to load environment variables:

```
export PATH=/path/to/DELLY/delly-0.8.5/src:$PATH
```

----End

4.41.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to go to the test case directory:

```
cd /path/to/DELLY/CASE
```

Step 3 Run the following command to decompress the test case package:

```
gzip -d B17NC_R1.fq.gz
```

Step 4 Run the following command to create an index file:

```
bwa index -a bwtsw hg19.fa
```

Step 5 Run the following command to create a **.sam** file:

```
bwa mem -t 96 hg19.fa B17NC_R1.fastq > bwa.sam
```

Step 6 Run the following commands to create a **.bam** file and process it:

```
samtools view -bS bwa.sam > B17NC_R1.bam
samtools sort -@96 B17NC_R1.bam -o B17NC_R1.sort.bam
samtools index -@96 B17NC_R1.sort.bam B17NC_R1.sort.bam.bai
```

Step 7 Run the following script to copy the **.bam** file to simulate the running of multiple samples at the same time:

```
#!/bin/bash
for varible1 in {1..127}
do
cp B17NC_R1.sort.bam B17NC_R1${varible1}.sort.bam && cp B17NC_R1.sort.bam.bai B17NC_R1$
{varible1}.sort.bam.bai
done
```

Step 8 Run the following commands to run the test case:

```
cp /path/to/DELLY/delly-0.8.5/excludeTemplates/human.hg19.excl.tsv ./
{ time OMP_NUM_THREADS=128 delly call -x human.hg19.excl.tsv -o delly.bcf -g hg19.fa
B17NC_R1*.sort.bam; }  2>&1 | tee -a  kp920-7265.log
```

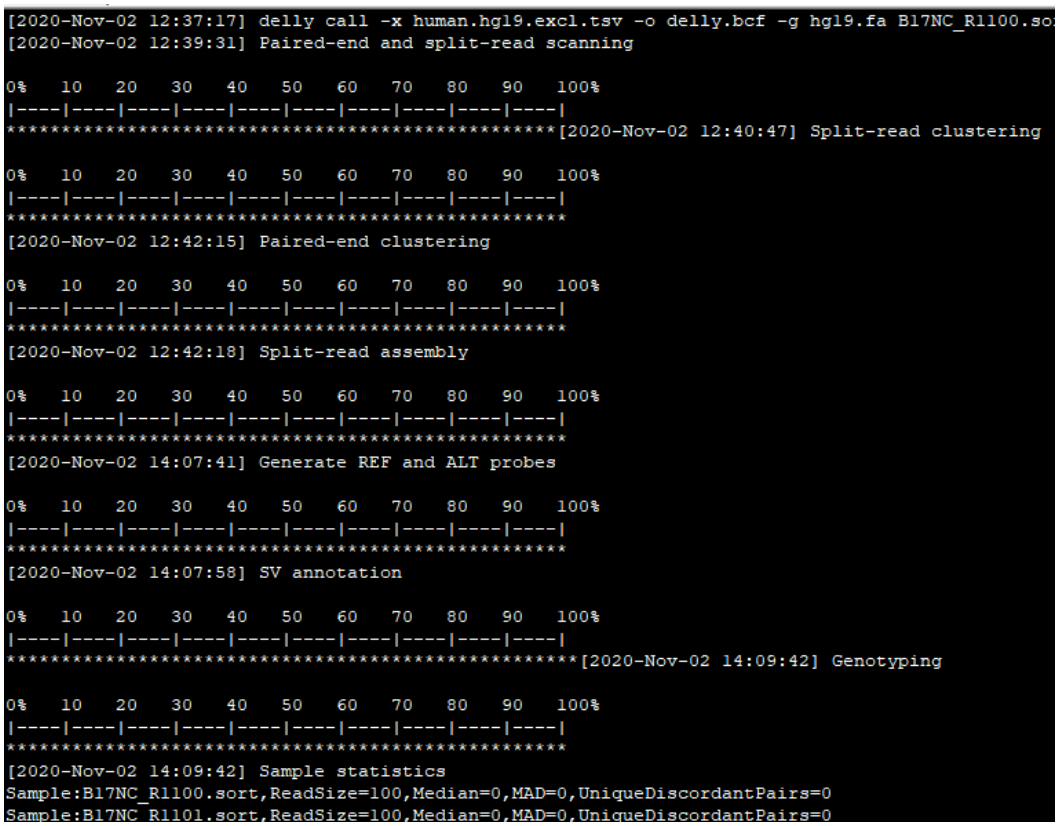
NOTE

Delly performs parallelization at the sample level. Therefore, **OMP_NUM_THREADS** must be equal to or less than the number of samples. In this example, there are 128 samples and **OMP_NUM_THREADS** is 128.

If the information shown in **Figure 4-19** is displayed, the running is successful.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-19 Running result

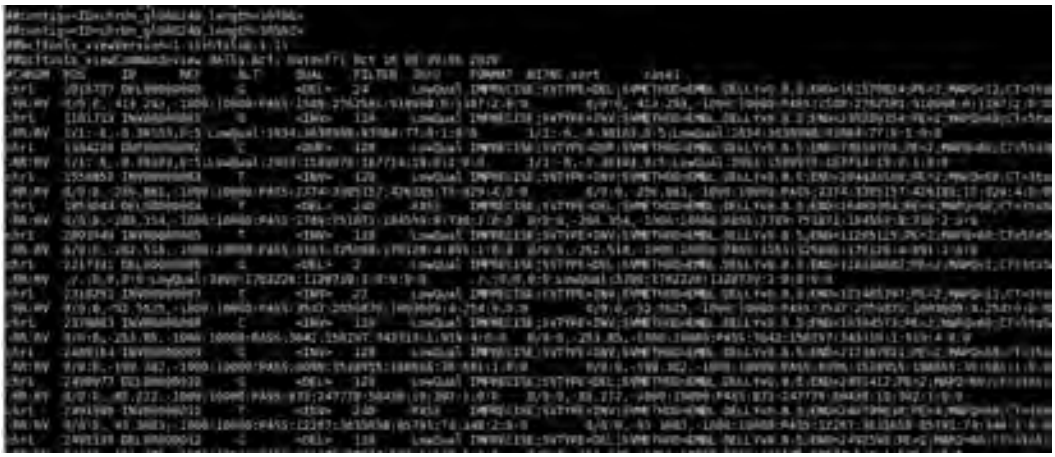


Step 9 Run the following command to convert the running result into the VCF format and view the command output:

```
bcftools view delly.bcf > delly.vcf && cat delly.vcf | more
```

Figure shows the command output.

Figure 4-20 Command output



----End

4.42 FastTree 2.1.11 Porting Guide (CentOS 7.6)

4.42.1 Introduction

FastTree infers approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences.

Programming language: C

Brief description: infers approximately-maximum-likelihood phylogenetic trees.

Recommended Version

FastTree-2.1.11

4.42.2 Environment Requirements

Hardware Requirements

Table 4-204 lists the hardware requirements.

Table 4-204 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-205 lists the software requirements.

Table 4-205 Software requirements

Item	Version	Download Link
FastTree	2.1.11	**http://meta.microbesonline.org/fasttree/FastTree.c**
Test case	all.masked	**http://morgannprice.org/16S/403K/all.masked.gz**

OS Requirements

Table 4-206 lists the OS requirements.

Table 4-206 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.42.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FastTree software porting.

Table 4-207 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/FastTree	Installation path of FastTree.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CASE	Path for storing test cases.	

4.42.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-208 Configuration process

No.	Configuration Item	Remarks
1	Setting Up the Basic Environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing GCC	For details, see **4.42.4.1 Installing GCC**.

 Copyright © Huawei Technologies Co., Ltd.

4.42.4.1 Installing GCC

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install GCC:

yum install -y gcc*

----End

4.42.5 Obtaining Source Code

Procedure

Step 1 Download the FastTree C code **FastTree.c** from the following link:

http://meta.microbesonline.org/fasttree/FastTree.c

Step 2 Use an SFTP tool to upload the downloaded installation package to the */path/to/FastTree* directory on the server.

----End

4.42.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the installation directory:

cd */path/to*/**FastTree**

Step 3 Run the following command to compile the concurrent versions:

gcc -DOPENMP -DUSE_DOUBLE -fopenmp -O3 -march=armv8.2-a -mtune=tsv110 -finline-functions -funroll-loops -Wall -o FastTreePar-2.1.11 FastTree.c -lm

Step 4 Run the following command to set the maximum number of concurrent threads to 128:

export OMP_NUM_THREADS=128

Step 5 Run the following commands to add the path to the **.bashrc** file:

echo "export PATH=*/path/to***/FastTree:$PATH" >> /root/.bashrc**

source /root/.bashrc

----End

 Copyright © Huawei Technologies Co., Ltd.

4.42.7 Running and Verifying FastTree

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the case directory and decompress the case:

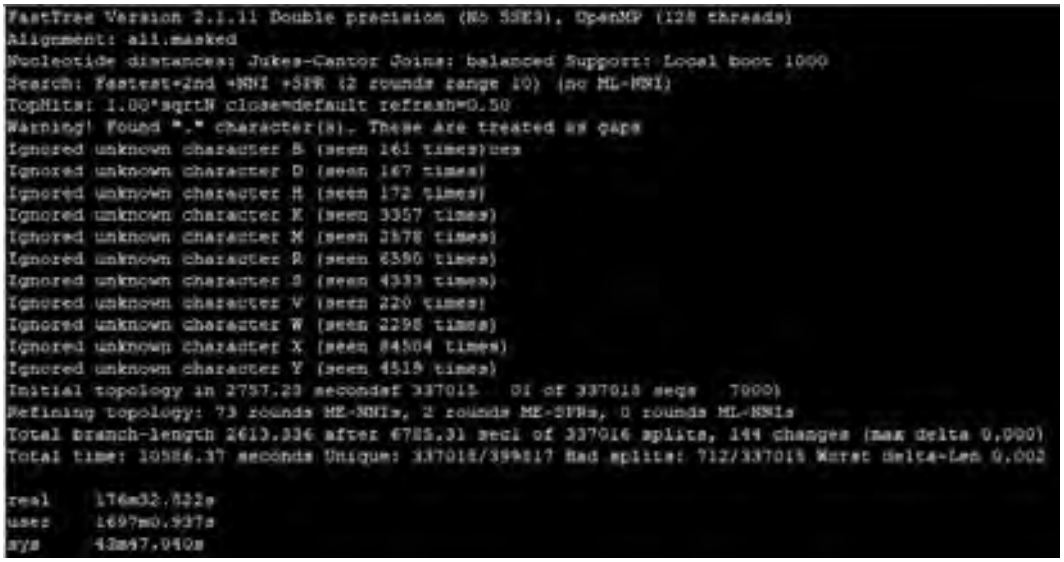
cd */home*/**CASE && gzip -d all.masked.gz**

Step 3 Run the following command to run the case:

time OMP_NUM_THREADS=128 FastTreePar-2.1.11 -fastest -noml -gtr -nt all.masked > `hostname`_`date '+%Y%m%d%H%M%S'`.masked.ft.rooted

Figure 4-21 shows the command output.

Figure 4-21 Command output



----End

4.42.8 More Resources

To obtain more resources, visit the official FastTree website:

http://www.microbesonline.org/fasttree/#BranchLen

4.43 Jellyfish 2.2.10 Porting Guide (CentOS 7.6)

4.43.1 Introduction

Jellyfish is a tool for fast, memory-efficient, and multithreaded counting of k-mers in DNA. It stores k-mer counts in hash tables.

For more information, visit the **official Jellyfish website**.

Programming language: C++

Brief description: k-mer counting and statistics

Open source license: GNU

Recommended Software Version

Jellyfish 2.2.10

4.43.2 Environment Requirements

Hardware Requirements

Table 4-209 lists the hardware requirements.

Table 4-209 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-210 lists the software requirements.

Table 4-210 Software requirements

Item	Version	Download URL
Jellyfish	2.2.10	https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz
Test case	human_g1k_v37	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

OS Requirements

Table 4-211 lists the OS requirements.

Table 4-211 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.43.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Jellyfish software porting.

Table 4-212 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/JELLYFISH	Installation path of Jellyfish.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/TESTCASE	Path for storing test cases.	

4.43.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-213 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.43.5 Obtaining the Source Code

Procedure

Step 1 Download Jellyfish installation package **jellyfish-2.2.10.tar.gz** from the following address:

 Copyright © Huawei Technologies Co., Ltd.

https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz

Step 2 Download the test case file **human_g1k_v37.fasta** from the following address:

http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

Step 3 Use an SFTP tool to upload the Jellyfish installation package to the */path/to/JELLYFISH* directory on the server.

Step 4 Use an SFTP tool to upload the **human_g1k_v37.fasta** test case to the */path/to/TESTCASE* directory on the server.

----End

4.43.6 Compiling and Installing Jellyfish

Procedure

Step 1 Run the following command to decompress the installation package:

```
tar zxvf jellyfish-2.2.10.tar.gz
```

Step 2 Run the following command to go to the directory generated after the decompression:

```
cd jellyfish-2.2.10
```

Step 3 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/JELLYFISH/Install CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110 -flto' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110 -flto'
```

Step 4 Run the following commands to perform the installation and compilation:

```
make -j 32
make install
```

Step 5 Run the following command to add the environment variable:

```
export PATH=/path/to/JELLYFISH/Install/bin:$PATH
```

----End

4.43.7 Running and Verifying Jellyfish

Procedure

Step 1 Run the following command to go to the directory where the test case is stored:

```
cd /path/to/TESTCASE
```

Step 2 Run the following command to decompress the test case file:

```
gzip -d human_g1k_v37.fasta.gz
```

Step 3 Run the following command to count k-mers:

```
jellyfish count -m 21 -s 100M -t 96 -o mer_counts -c 7 human_g1k_v37.fasta
```

If the **mer_counts** file is generated, the verification is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.44 PLINK 1.9 Porting Guide (CentOS 7.6)

4.44.1 Introduction

PLINK is a free, commonly used, open-source whole-genome association analysis toolset designed by Shaun Purcell. The software is designed flexibly to perform a wide range of basic, large-scale genetic analyses.

For more information about PLINK, visit **http://www.cog-genomics.org/plink2/**.

One-sentence description: whole-genome association analysis

Programming language: C/C++

Recommended Version

The recommended version is PLINK 1.9.

4.44.2 Environment Requirements

Hardware Requirements

Table 4-214 lists the hardware requirements.

Table 4-214 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-215 lists the software requirements.

Table 4-215 Software requirements

Item	Version	Download URL
PLINK	1.9	**https://github.com/chrchang/plink-ng/archive/master.tar.gz**
zlib	1.2.11	**https://udomain.dl.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz**
Test case	hapmap_r23a	**http://zzz.bwh.harvard.edu/plink/dist/hapmap_r23a.zip**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-216 lists the OS requirements.

Table 4-216 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.44.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the PLINK software porting.

Table 4-217 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in the *HPC Solution Basic Environment Setup Guide*.
2	/path/to/PLINK	Installation path of PLINK.	The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/TESTCASE	Path for storing test cases.	

4.44.4 Obtaining the Source Code

Procedure

Step 1 Download PLINK installation package **master.tar.gz** from the following address:

https://github.com/chrchang/plink-ng/archive/master.tar.gz

Step 2 Download zlib installation package **zlib-1.2.11.tar.gz** from the following address:

https://udomain.dl.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Use the SFTP tool to upload the PLINK and zlib installation packages to the */path/to/PLINK* directory on a server.

----End

4.44.5 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-218 Configuration process

No.	Configuration Item	Description
1	Basic environment setup	For details, see "Environment Setup in Cluster Scenarios" in the *HPC Solution Basic Environment Setup Guide*.

4.44.6 Compiling and Installing PLINK

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency software:

yum install blas-devel lapack-devel atlas-devel perl-Digest-SHA -y

Step 3 Run the following commands to create library file links:

ln -s /usr/lib64/atlas/libtatlas.so.3.10 /usr/lib64/atlas/libatlas.so

ln -s /usr/lib64/atlas/libsatlas.so.3.10 /usr/lib64/atlas/libcblas.so

Step 4 Run the following command to go to the planned PLINK installation directory and decompress the source code package:

cd */path/to***/PLINK && tar -zxf master.tar.gz**

Step 5 Run the following command to move the zlib source code package to the specified directory:

mv zlib-1.2.11.tar.gz plink-ng-master

Step 6 Run the following command to switch to the installation directory and modify the configuration file:

cd plink-ng-master/1.9 && sed -i '0,/curl/s/curl/# curl/g' plink_first_compile

Step 7 Run the following command to switch to the directory generated after decompression:

 Copyright © Huawei Technologies Co., Ltd.

./plink_first_compile

Step 8 Run the following command to configure the environment variable:

export PATH=*/path/to***/PLINK/plink-ng-master/1.9:$PATH**

----End

4.44.7 More Resources

To obtain more resources, visit the official website of PLINK:

http://zzz.bwh.harvard.edu/plink/ld.shtml

4.45 SNAP 2013-11-29 Porting Guide (CentOS 7.6)

4.45.1 Introduction

Semi-HMM-based Nucleic Acid Parser (SNAP) is a general purpose gene finding program suitable for both eukaryotic and prokaryotic genomes.

For more information, visit the **official SNAP website**.

Programming language: C/C++

Brief description: gene sequencing

Open source license: GNU

Recommended Software Version

SNAP-2013-11-29

4.45.2 Environment Requirements

Hardware Requirements

Table 4-219 lists the hardware requirements.

Table 4-219 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-220 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-220 Software requirements

Item	Version	Download URL
SNAP	2013-11-29	http://korflab.ucdavis.edu/Software/snap-2013-11-29.tar.gz
Cufflinks	2.2.1	https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master
Maker	3.01.03	http://www.yandell-lab.org/software/maker.html
Test case 1	hg38.gtf	http://genome.ucsc.edu/cgi-bin/hgTables
Test case 2	hs38DH.fasta	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz

OS Requirements

Table 4-221 lists the OS requirements.

Table 4-221 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.45.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SNAP software porting.

Table 4-222 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/CUFFLINKS	Installation path of Cufflinks.	See **Cufflinks 2.2.1 Porting Guide (CentOS 7.6)**.
3	/path/to/SNAP	Installation path of SNAP.	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
4	/path/to/MAKER	Installation path of Maker.	

4.45.4 Configuring the Compilation Environment

4.45.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-223 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Cufflinks	See **Cufflinks 2.2.1 Porting Guide (CentOS 7.6)**.
3	Installing Maker	See **4.45.4.2 Installing Maker**.

4.45.4.2 Installing Maker

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
tar -xvf maker-3.01.03.tgz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to set the environment variable:

```
export PATH=/path/to/MAKER/maker/src/bin:$PATH
```

----End

4.45.5 Obtaining the Source Code

Procedure

Step 1 Download SNAP installation package **snap-2013-11-29.tar.gz** from the following address:

http://korflab.ucdavis.edu/Software/snap-2013-11-29.tar.gz

Step 2 Use the SFTP tool to upload the SNAP installation package to the */path/to/SNAP* directory on the server.

----End

4.45.6 Compiling and Installing SNAP

Procedure

Step 1 Run the following command to decompress the installation package:

```
tar zxvf snap-2013-11-29.tar.gz
```

Step 2 Run the following command to go to the directory generated after the decompression:

```
cd snap
```

Step 3 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and delete the **-Werror** option from line 97 to line 101.

   Before the modification:

   ```
   make $(APP)  CC="gcc" CFLAGS="-O2 –Wall -Werror"
   make $(APP2) CC="gcc" CFLAGS="-O2 –Wall -Werror "
   make $(APP3) CC="gcc" CFLAGS="-O2 –Wall -Werror "
   make $(APP4) CC="gcc" CFLAGS="-O2 –Wall -Werror "
   make $(APP5) CC="gcc" CFLAGS="-O2 –Wall -Werror "
   ```

   After the modification:

   ```
   make $(APP)  CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP2) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP3) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP4) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP5) CC="gcc" CFLAGS="-O2 –Wall"
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the following command to go to the **Zoe** directory:

```
cd Zoe
```

Step 5 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.
   ```
   vi Makefile
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the insert mode and delete the **-Werror** option from line 94 to line 102.

   Before the modification:

   ```
   gcc:
       make $(APP)  CC="gcc" CFLAGS="-O2 -Wall -Werror"
   all:
       make $(APP)  CC="gcc" CFLAGS="-O2 –Wall -Werror "
       make $(APP2) CC="gcc" CFLAGS="-O2 –Wall -Werror "
       make $(APP3) CC="gcc" CFLAGS="-O2 –Wall -Werror "
       make $(APP4) CC="gcc" CFLAGS="-O2 –Wall -Werror "
       make $(APP5) CC="gcc" CFLAGS="-O2 –Wall -Werror "
   ```

   After the modification:

   ```
   gcc:
       make $(APP)  CC="gcc" CFLAGS="-O2 -Wall"
   all:
       make $(APP)  CC="gcc" CFLAGS="-O2 –Wall"
       make $(APP2) CC="gcc" CFLAGS="-O2 –Wall"
       make $(APP3) CC="gcc" CFLAGS="-O2 –Wall"
       make $(APP4) CC="gcc" CFLAGS="-O2 –Wall"
       make $(APP5) CC="gcc" CFLAGS="-O2 –Wall"
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to perform the installation and compilation:

```
cd /path/to/SNAP/snap
make
```

Step 7 Run the following command to add the environment variable:

```
export PATH=/path/to/SNAP/snap:$PATH
```

----End

4.45.7 Running and Verifying SNAP

Procedure

Step 1 Run the following command to declare the Cufflinks environment:

```
export PATH=/path/to/CUFFLINKS/build/bin:$PATH
```

Step 2 Run the following command to process the **gtf** test case:

```
gffread hg38.gtf -o- >hg38.gff3
```

Step 3 Run the following command to use Maker to process the **gff3** test case:

```
/path/to/MAKER/maker/src/bin/maker2zff hg38.gff3
```

The **genome.ann(ZFF)** and **genome.dna** files are generated in the current directory.

Step 4 Run the following commands to use **fathom** and **forge** to process the test cases:

```
fathom genome.ann genome.dna -categorize 1000
fathom uni.ann uni.dna -export 1000 -plus
forge export.ann export.dna
hmm-assembler.pl test . >test.hmm
```

Step 5 Run the following commands to decompress and rename the test cases:

```
gzip -d GCA_000001405.15_GRCh38_full_analysis_set.fna.gz
mv GCA_000001405.15_GRCh38_full_analysis_set.fna hs38DH.fasta
```

Step 6 Run the following command to perform a test:

```
{ time snap test.hmm hs38DH.fasta>snap.zff ; } 2>&1 |tee snap.log
```

 Copyright © Huawei Technologies Co., Ltd.

If the following information is displayed, the verification is successful:

```
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done
scoring....decoding.10.20.30.40.50.60.70.80.90.100 done

real    214m8.893s
user    211m44.431s
sys 0m31.469s
```

----End

4.45.8 More Resources

- **hg38.gtf** download address: **http://genome.ucsc.edu/cgi-bin/hgTables**
- Download method: Set **output format** and **output file** as shown in the following figure, and click **get output**.

4.46 kmersGWAS 0.2 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.46.1 Introduction

kmersGWAS is a library for running k-mers-based GWAS. Structural variability and polymorphism presence or non-presence are common in plant genomes.

For more information, visit the **official website**.

Programming languages: C++, Python, and R

Brief description: a library for running k-mers-based GWAS

Recommended Software Version

kmersGWAS 0.2-beta

4.46.2 Environment Requirements

Hardware Requirements

Table 4-224 lists the hardware requirements.

Table 4-224 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-225 lists the software requirements.

Table 4-225 Software requirements

Item	Version	Download URL
GEMMA	0.96	**https://github.com/genetics-statistics/GEMMA/archive/v0.96.tar.gz**
R	3.1.1	**http://cran.rstudio.com/src/base/R-3/R-3.1.1.tar.gz**
CMake	3.17.3	**https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz**
OpenBLAS	0.3.9	**https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz**
Eigen	3.3.8	**https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz**
GSL	2.5.0	**https://github.com/ampl/gsl/archive/v2.5.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
kmersGWAS	0.2-beta	https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz
CTPL	0.0.2	https://github.com/vit-vit/CTPL/archive/v.0.0.2.tar.gz
KMC	3.1.2rc1	https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz
cxxopts	2.2.1	https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz
Test case	1001G_kmers_table	https://zenodo.org/record/3596255/files/A_thaliana_1001G_kmers_table.tar.gz

OS Requirements

Table 4-226 lists the OS requirements.

Table 4-226 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

4.46.3 Paths for Software Porting

This chapter lists the software installation paths involved in the kmersGWAS software porting.

Table 4-227 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/R	Installation path of R.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with
3	/path/to/OPENBLAS	Installation path of OpenBLAS.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/EIGEN	Installation path of Eigen.	the actual paths planned during the installation process.
5	/path/to/GSL	Installation path of GSL.	
6	/path/to/GEMMA	Installation path of Gemma.	
7	/path/to/KMERSGWAS	Installation path of kmersGWAS.	
8	/path/to/CASE	Path for storing test cases.	

4.46.4 Configuring the Compilation Environment

4.46.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-228 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing R	See **4.46.4.2 Installing R**.
3	Installing CMake	See **4.46.4.3 Installing CMake**.
4	Installing OpenBLAS	See **4.46.4.4 Installing OpenBLAS**.
5	Installing Eigen	See **4.46.4.5 Installing Eigen**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
6	Installing GSL	See **4.46.4.6 Installing GSL**.
7	Installing Gemma	See **4.46.4.7 Installing Gemma**.

4.46.4.2 Installing R

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the R installation package to the */path/to/R* directory.

Step 3 Run the following command to decompress the installation package and go to the installation directory:

```
cd /path/to/R && tar -zxf R-3.1.1.tar.gz && cd R-3.1.1
```

Step 4 Run the following command to perform the configuration:

```
./configure --enable-R-shlib --enable-R-static-lib --with-libpng --with-jpeglib --prefix=/path/to/R
```

Step 5 Run the following command to perform the compilation and installation:

```
make all -j8 && make install
```

----End

4.46.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the CMake installation package to the */path/to/CMAKE* directory.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/CMAKE && tar -zxf cmake-3.17.3.tar.gz
```

Step 4 Run the following command to go to the installation directory:

```
cd cmake-3.17.3
```

Step 5 Run the following commands to perform the compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap --prefix=/path/to/CMAKE
gmake && gmake install
```

Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.46.4.4 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the OpenBLAS installation package to the */path/to/OPENBLAS* directory.

Step 3 Run the following command to install dependencies:

```
yum install blas blas-devel lapack lapack-devel -y
```

Step 4 Run the following command to decompress the installation package:

```
cd /path/to/OPENBLAS && tar -zxf OpenBLAS-0.3.9.tar.gz
```

Step 5 Run the following command to go to the directory generated after the decompression:

```
cd OpenBLAS-0.3.9/
```

Step 6 Run the following commands to set the environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 7 Run the following commands to perform the compilation and installation:

```
make -j 32
make PREFIX=/path/to/OPENBLAS install
```

----End

4.46.4.5 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Eigen installation package to the */path/to/EIGEN* directory.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/EIGEN && tar -zxf eigen-3.3.8.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd eigen-3.3.8
```

Step 5 Run the following commands to perform the compilation and installation:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/EIGEN ..
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.46.4.6 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the GSL installation package to the */path/to/GSL* directory.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/GSL && tar -zxf gsl-2.5.0.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd gsl-2.5.0
```

Step 5 Run the following commands to perform the compilation and installation:

```
mkdir build
cd build
../configure --prefix=/path/to/GSL
make -j 32
make install
```

----End

4.46.4.7 Installing Gemma

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Gemma installation package to the */path/to/GEMMA* directory.

Step 3 Run the following command to install dependencies:

```
yum install blas-devel lapack-devel -y
```

Step 4 Run the following command to decompress the installation package:

```
cd /path/to/GEMMA && tar -zxf GEMMA-0.96.tar.gz
```

Step 5 Run the following command to go to the directory generated after the decompression:

```
cd GEMMA-0.96
```

Step 6 Run the following command to modify compilation optimization parameters:

```
sed -i 's/\-O3/-O3 -march=armv8.2-a -mtune=tsv110/' Makefile
```

Step 7 Run the following command to replace the parameter **-m64** by **-mabi=lp64**:

```
sed -i 's/\-m64/\-mabi=lp64/g' Makefile
```

Step 8 Run the following commands to change the value of the compilation parameter **LIBS_LNX_S_LAPACK** and delete the compilation parameter **static**:

```
sed -i 's/LIBS_LNX_S_LAPACK = /# LIBS_LNX_S_LAPACK = /' Makefile
sed -i '/LIBS_LNX_S_LAPACK = /a LIBS_LNX_S_LAPACK = -lgfortran -lopenblas -llapack -lblas -lgfortran' Makefile
sed -i 's/CPPFLAGS += -static/#CPPFLAGS += -static/' Makefile
```

Step 9 Run the following commands to set the environment variables:

 Copyright © Huawei Technologies Co., Ltd.

```
export LIBRARY_PATH=/path/to/OPENBLAS/lib:/path/to/GSL/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:/path/to/GSL/lib:$LD_LIBRARY_PATH
export CPLUS_INCLUDE_PATH=/path/to/OpenBLAS/include:/path/to/EIGEN/include/eigen3:/path/to/GSL/include:$CPLUS_INCLUDE_PATH
export C_INCLUDE_PATH=/path/to/OpenBLAS/include:/path/to/EIGEN/include/eigen3:/path/to/GSL/include:$C_INCLUDE_PATH
```

Step 10 Run the following commands to perform the compilation and installation:

```
make -j
```

----End

4.46.5 Obtaining the Source Code

Procedure

Step 1 Download the **kmersGWAS**, **KMC**, **CTPL**, and **cxxopts** source code packages.

Download the **kmersGWAS** source code package at **https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz**

Download the KMC source code package at **https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz**

Download the CTPL source code package at **https://github.com/vit-vit/CTPL/archive/v.0.0.2.tar.gz**

Download the **cxxopts** source code package at **https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz**

Step 2 Use an SFTP tool to upload the installation packages to the **/path/to/KMERSGWAS** directory.

----End

4.46.6 Compiling and Installing kmersGWAS

Procedure

NOTE

When installing kmersGWAS on an x86 server, you can skip **Step 2**, **Step 8**, **Step 9**, and **Step 10**.

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to add the **sse2neon** dependency:

```
cd /path/to/KMERSGWAS
wget https://github.com/DLTcollab/sse2neon/archive/master.tar.gz --no-check-certificate -O sse2neon-master.tar.gz
tar -zxf sse2neon-master.tar.gz
export CPLUS_INCLUDE_PATH=/path/to/KMERSGWAS/sse2neon-master/:$CPLUS_INCLUDE_PATH
export C_INCLUDE_PATH=/path/to/KMERSGWAS/sse2neon-master/:$C_INCLUDE_PATH
```

Step 3 Run the following commands to decompress the installation package:

```
cd /path/to/KMERSGWAS
tar -zxf kmersGWAS-0.2-beta.tar.gz
```

Step 4 Run the following commands to copy the KMC package to kmersGWAS:

```
tar -zxf KMC-3.1.2rc1.tar.gz
cp -rf KMC-3.1.2rc1/* kmersGWAS-0.2-beta/include/KMC/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following commands to copy the CTPL package to kmersGWAS:

```
tar -zxf CTPL-v.0.0.2.tar.gz
cp -rf CTPL-v.0.0.2/* kmersGWAS-0.2-beta/include/CTPL/
```

Step 6 Run the following commands to copy the cxxopts package to kmersGWAS:

```
tar -zxf cxxopts-2.2.1.tar.gz
cp -rf cxxopts-2.2.1/* kmersGWAS-0.2-beta/include/cxxopts/
```

Step 7 Run the following command to go to the kmersGWAS installation directory:

```
cd kmersGWAS-0.2-beta
```

Step 8 Run the following command to delete the **-msse4.2** compilation option:

```
sed -i 's/-msse4.2//g' Makefile
```

Step 9 Run the following command to add compilation optimization parameters:

```
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110/g' Makefile
```

Step 10 Replace the **nmmintrin.h** reference file.

- Run the following commands to comment out **nmmintrin.h** and **smmintrin.h**:

```
sed -i 's/#include <nmmintrin.h>/\/\/#include <nmmintrin.h>/g;s/#include <smmintrin.h>/\/\/
#include <smmintrin.h>/g' src/kmers_multiple_databases.cpp
sed -i 's/#include <smmintrin.h>/\/\/#include <smmintrin.h>/g' src/snps_multiple_databases.cpp
```

- Run the following commands to import the header file **sse2neon.h**:

```
sed -i '/#include <nmmintrin.h>/ i\#include "sse2neon.h"' src/kmers_multiple_databases.cpp
sed -i '/#include <smmintrin.h>/ i\#include "sse2neon.h"' src/snps_multiple_databases.cpp
```

Step 11 Run the following command to perform the compilation and installation:

```
make -j
```

Step 12 Run the following commands to delete the original **gemma_0_96** and create a new soft link:

```
cd /path/to/KMERSGWAS/kmersGWAS-0.2-beta/external_programs
rm -rf gemma_0_96
ln -s /path/to/GEMMA/bin/gemma gemma_0_96
```

----End

4.46.7 Running and Verifying kmersGWAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to set the environment variables:

```
export PATH=/path/to/KMERSGWAS/kmersGWAS-0.2-beta/bin:$PATH
export PATH=/path/to/R/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following commands to go to the directory for storing test cases, decompress the test cases, and move them to the specified directory:

```
cd /path/to/CASE
tar -zxf A_thaliana_1001G_kmers_table.tar.gz
mv A_thaliana_1001G_kmers_table /path/to/KMERSGWAS/kmersGWAS-0.2-beta/examples/
flowering_time_arabidopsis/
```

Step 4 Run the following command to go to the directory of test cases:

```
cd /path/to/KMERSGWAS/kmersGWAS-0.2-beta/examples/flowering_time_arabidopsis/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Run the following command to run the test case:

```
{ time python2.7 ../../kmers_gwas.py --pheno FT10.pheno --kmers_table
A_thaliana_1001G_kmers_table/kmers_table -l 31 -p 128 --outdir run_GWAS_FT10_kp_`date '+%Y%m
%d%H%M%S'`; } 2>&1 | tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log
```

Figure 4-22 shows an example of the running result.

Figure 4-22 Result example

```
pve estimate =0.999907                                          0.00%
se(pve) =-nan
Reading SNPs  ==================================================100.00%
pve estimate =0.999907
se(pve) =-nan
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%
Reading SNPs  ==================================================100.00%

real    59m7.595s
user    5521m25.976s
sys     1m0.051s
```

----End

4.47 PAML 4.9j Porting Guide (CentOS 7.6)

4.47.1 Introduction

PAML is a package of programs for phylogenetic analyses of DNA or protein sequences using maximum likelihood.

For more information about PAML, visit the **official PAML website**.

Programming language: Python

One-sentence description: phylogenetic analyses of DNA or protein sequences

Recommended Version

The recommended version is PAML4.9j.

4.47.2 Environment Requirements

Hardware Requirements

Table 4-229 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-229 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-230 lists the software requirements.

Table 4-230 Software requirements

Item	Version	Download URL
PAML	4.9	http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz

OS Requirements

Table 4-231 lists the OS requirements.

Table 4-231 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.47.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the PAML software porting.

Table 4-232 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in the **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/PAML	Installation path of PAML.	The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.47.4 Obtaining the Source Code

Procedure

Step 1 Download PAML source code package **paml4.9j.tgz** from the following address:

http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz

Step 2 Use the SFTP tool to upload the source code package to the */path/to/PAML/* directory on the server.

----End

4.47.5 Compiling and Installing PAML

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the PMAL installation package:

tar xf paml4.9j.tgz

Step 3 Run the following command to switch to the directory generated after decompression:

cd paml4.9j

Step 4 Run the following commands to delete the Windows executable file (.exe file) from the **paml4.9j/bin/** folder:

cd */path/to/PAML/***paml4.9j/bin**

rm *.exe

Step 5 Run the following commands to perform compilation:

cd */path/to/PAML/***paml4.9j/src**

make -f Makefile

ls -lF

 Copyright © Huawei Technologies Co., Ltd.

rm *.o

Step 6 Run the following commands to copy the executable file in the **src** directory to the **bin** directory:

cp */path/to/PAML/***paml4.9j/src/baseml basemlg codeml pamp evolver yn00 chi2** */path/to/PAML/***paml4.9j/bin**

cd ..

ls -lF bin

If the following three files exist in the **bin** directory, the compilation is successful:

```
bin/baseml
bin/codeml
bin/evolver
```

----End

4.47.6 Running and Verifying PAML

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the installation directory of the executable file:

cd */path/to/PAML/***paml4.9j**

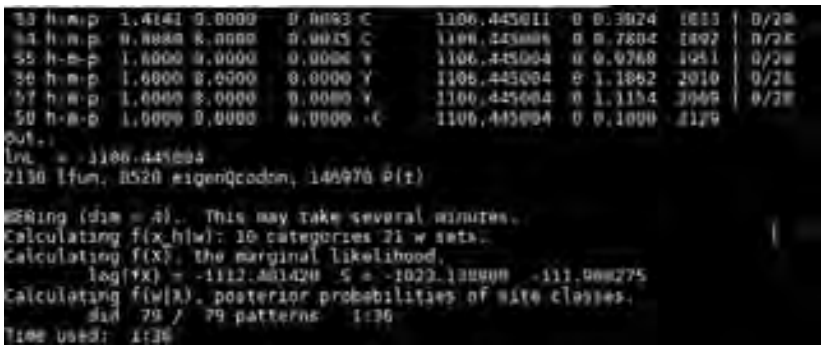
Step 3 Run the following commands to run the test case:

cp src/codeml */path/to/PAML/***paml4.9j/examples/HIVNSsites**

cd */path/to/PAML/***paml4.9j/examples/HIVNSsites**

./codeml

After the command is run, the following information is displayed:



----End

4.47.7 More Resources

To obtain more resources, visit the abacus page of PAML:

http://abacus.gene.ucl.ac.uk/software/paml.html

 Copyright © Huawei Technologies Co., Ltd.

4.48 Diamond 2.0.4 Porting Guide (CentOS 7.6)

4.48.1 Introduction

Diamond is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data. The key features are:

For more information about Diamond, visit the **official Diamond website**.

Programming language: C++

Brief description: a sequence aligner for protein and translated DNA searches

Open-source license: GPL3.0

Recommended Version

diamond 2.0.4

4.48.2 Environment Requirements

Hardware Requirements

Table 4-233 lists the hardware requirements.

Table 4-233 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 2-2 lists the software requirements.

Table 4-234 Software requirements

Item	Version	Download Link
diamond	2.0.4	**https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz**
cmake	3.17.3	**https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz**
Test case 1	swissprot.gz	**http://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/swissprot.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Link
Test case 2	human_g1k_v37.fasta.gz	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

OS Requirements

Table 4-235 lists the OS requirements.

Table 4-235 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.48.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Diamond software porting.

Table 4-236 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment.	For details, see "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.
2	*/path/to/DIAMOND*	Installation path of Diamond.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	*/path/to/CMAKE*	Installation path of CMake.	
4	*/path/to/TESTCASE*	Path for storing test cases.	

 Copyright © Huawei Technologies Co., Ltd.

4.48.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-237 Configuration process

No.	Configuration Item	Remarks
1	Setting Up the Basic Environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing CMake	For details, see **4.48.4.1 Installing CMake**.

4.48.4.1 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the CMake installation package:

tar zxvf cmake-3.17.3.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd cmake-3.17.3

Step 4 Run the following command to perform configuration:

./configure --prefix=*/path/to/CMAKE*

Step 5 Run the following commands to perform compilation and installation:

make

make install

Step 6 Run the following command to set CMake environment variables:

export PATH=*/path/to/CMAKE***/bin:$PATH**

----End

 Copyright © Huawei Technologies Co., Ltd.

4.48.5 Obtaining Source Code

Procedure

Step 1 Download the Diamond source code package **diamond-2.0.4.tar.gz** from the following link:

https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz

Step 2 Use an SFTP tool to upload the downloaded installation package to the */path/to/diamond* directory on the server.

----End

4.48.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Diamond installation package:

tar -xvf diamond-2.0.4.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd diamond-2.0.4

Step 4 Run the following commands to create a folder:

mkdir build

Step 5 Run the following commands to perform compilation and installation:

cd build

export CC=`which gcc`

export CXX=`which g++`

export FC=`which gfortran`

cmake -DCMAKE_INSTALL_PREFIX=*/path/to/DIAMOND* **../**

make -j 16

make install

Step 6 Run the following command to set environment variables:

export PATH=*/path/to/DIAMOND***/bin:$PATH**

----End

 Copyright © Huawei Technologies Co., Ltd.

4.48.7 Running and Verifying Diamond

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to go to the case directory and decompress the case:

cd */path/to/TESTCASE*

gzip -d human_g1k_v37.fasta.gz

gzip -d swissprot.gz

Step 3 Run the following command to run the case:

diamond blastp -p 128 -q human_g1k_v37.fasta -d swissprot -o out.tsv --very-sensitive

After the command is executed, the following information is displayed:

```
.......
Loading reference sequences...  [0s]
Deallocating buffers...  [0s]
Deallocating queries...  [0.002s]
Loading query sequences...  [0s]
Closing the input file...  [0s]
Closing the output file...  [0s]
Closing the database file...  [0.003s]
Deallocating taxonomy...  [0s]
Total time = 538.485s
Reported 0 pairwise alignments, 0 HSPs.
0 queries aligned.
```

----End

4.48.8 More Resources

For more information, visit the official Diamond website:

http://www.diamondsearch.org/

4.49 HISAT 0.1.6 Porting Guide (CentOS 7.6)

4.49.1 Introduction

HISAT is a fast and sensitive spliced alignment program for mapping RNA-seq reads. It includes a new indexing scheme based on the Burrows-Wheeler transform (BWT) and the FM index, called hierarchical indexing, that employs the two types of indexes: one global FM index representing the whole genome, and many separate local FM indexes for small regions collectively covering the genome. All these together, HISAT enables extremely fast and sensitive alignment of reads, in particular those spanning two exons or more.

For more information, visit the **official HISAT website**.

Programming language: C/C++

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a fast and sensitive spliced alignment program for mapping RNA-seq reads

Open source license: GPL

Recommended Software Version

HISAT 0.1.6

4.49.2 Environment Requirements

Hardware Requirements

Table 4-238 lists the hardware requirements.

Table 4-238 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-239 lists the software requirements.

Table 4-239 Software requirements

Item	Version	Download URL
HISAT	0.1.6	http://www.ccb.jhu.edu/software/hisat/downloads/hisat-0.1.6-beta-source.zip
Test case file 1	R1.fq	http://opengene.org/data/R1.fq.gz
Test case file 2	R2.fq	http://opengene.org/data/R2.fq.gz
Test case file 3	Drosophila_melanogaster.BDGP6.dna.toplevel.fa	ftp://ftp.ensembl.org/pub/release-84/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz

OS Requirements

Table 4-240 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-240 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.49.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HISAT software porting.

Table 4-241 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HISAT	Installation path of HISAT.	The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/TESTCASE	Path for storing the test cases.	

4.49.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-242 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.49.5 Obtaining Source Code

Procedure

Step 1 Download the HISAT installation package **hisat-0.1.6-beta-source.zip** from the following link:

http://www.ccb.jhu.edu/software/hisat/downloads/hisat-0.1.6-beta-source.zip

Step 2 Download the test case files **R1.fq.gz**, **R2.fq.gz**, and **Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz**.

Download link 1: **http://opengene.org/data/R1.fq.gz**

Download link 2: **http://opengene.org/data/R2.fq.gz**

Download link 3: **ftp://ftp.ensembl.org/pub/release-84/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz**

Step 3 Use an SFTP tool to upload the HISAT installation package to the */path/to/HISAT* directory on the server.

Step 4 Use an SFTP tool to upload the test cases to the */path/to/TESTCASE* directory on the server.

----End

4.49.6 Compilation and Installation

Prerequisites

You have obtained the **sse2neon.h** file, which is available at **https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h**.

Procedure

Step 1 Run the following command to decompress the installation package:

unzip hisat-0.1.6-beta-source.zip

Step 2 Run the following command to go to the directory generated after decompression:

cd hisat-0.1.6-beta

Step 3 Run the following command to set environment variables:

export NO_TBB=1

export POPCNT_CAPABILITY=0

Step 4 Run the following command to modify the **Makefile** file:

1. Open **Makefile**.

   vi Makefile

2. Press **i** to go to the edit mode. Modify the AArch64 branch of **BITS=64** in the **Makefile** file. Specifically, between line 143 and line 144, insert the following

 Copyright © Huawei Technologies Co., Ltd.

content in bold: (**BITS=64** supports x86 only and does not support AArch64. You need to add an AArch64 branch.)

Before the modification:

```
BITS=32
ifeq (x86_64,$(shell uname -m))
BITS=64
endif
```

After the modification:

```
BITS=32
ifeq (x86_64,$(shell uname -m))
BITS=64
endif
ifeq (aarch64,$(shell uname -m))
BITS=64
endif
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.

   vi Makefile

2. Press **i** to go to the insert mode. Modify the AArch64 branches of **-m64** and **-msse2** in the **Makefile** file. Specifically, between line 165 and line 168, insert the following content in bold: The compilation options **-m64** and **-msse2** in **Makefile** support x86 only and do not support AArch64. You need to add AArch64 branches.

   Before the modification: (lines 152 to 159 in the original file)

```
ifeq (32,$(BITS))
BITS_FLAG = -m32
endif
ifeq (64,$(BITS))
BITS_FLAG = -m64
endif
SSE_FLAG=-msse2
```

   After the modification:

```
ifeq (32,$(BITS))
        BITS_FLAG = -m32
endif
ifeq (64,$(BITS))
        BITS_FLAG = -m64
endif
SSE_FLAG=-msse2
ifeq (aarch64,$(shell uname -m))
        BITS_FLAG =
        SSE_FLAG =
endif
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to modify the **Makefile** file:

1. Open **Makefile**.

   vi Makefile

2. Press **i** to go to the insert mode and modify the **FLAGS** parameters in the **Makefile** file.

   Before the modification:

```
DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG)
DEBUG_DEFS     = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\""
RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3
```

 Copyright © Huawei Technologies Co., Ltd.

```
RELEASE_DEFS   = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

After the modification:

```
DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG) -funroll-loops -std=c++98
DEBUG_DEFS     = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\""
RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3 -std=c++98 -Xlinker --allow-
multiple-definition
RELEASE_DEFS   = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following command to copy **sse2neon.h** to the installation directory of the compiler.

cp sse2neon.h */path/to/GNU*/**include**

Step 8 Run the following commands to modify the **aligner_sw.h** and **sse_util.h** header files:

1. Open **aligner_sw.h**.

   vi aligner_sw.h

2. Press **i** to go to the insert mode. Modify the **aligner_sw.h** file by replacing **#include <emmintrin.h>** with **#include <sse2neon.h>**.

   Before the modification:

   ```
   #include <emmintrin.h>
   ```

   After the modification:

   ```
   #include <sse2neon.h>
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

NOTE

Modify the **sse_util.h** file in the same way.

Step 9 Run the following command to perform compilation and installation:

make

Step 10 Run the following command to add the environment variable:

export PATH=*/path/to/HISAT/***hisat-0.1.6-beta**:**$PATH**

----End

4.49.7 Running and Verification

Procedure

Step 1 Run the following command to go to the test case directory:

cd */path/to/TESTCASE*

Step 2 Run the following commands to decompress the test case file packages:

gzip -d Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz

gzip -d R1.fq.gz

 Copyright © Huawei Technologies Co., Ltd.

gzip -d R2.fq.gz

Step 3 Run the following command to rename the **fa** file:

mv Drosophila_melanogaster.BDGP6.dna.toplevel.fa genome.fa

Step 4 Run the following command to create a genome index:

hisat-build genome.fa genome

Step 5 Run the following command to compare gene sequences:

hisat -t -x genome -p 96 -1 R1.fq -2 R2.fq -S result.sam

If the following information is displayed, the verification is successful:

```
Time loading reference: 00:00:00
Time loading forward index: 00:00:00
Multiseed full-index search: 00:02:44
3750000 reads; of these:
  3750000 (100.00%) were paired; of these:
    3750000 (100.00%) aligned concordantly 0 times
    0 (0.00%) aligned concordantly exactly 1 time
    0 (0.00%) aligned concordantly >1 times
    ----
    3750000 pairs aligned concordantly 0 times; of these:
      0 (0.00%) aligned discordantly 1 time
    ----
    3750000 pairs aligned 0 times concordantly or discordantly; of these:
      7500000 mates make up the pairs; of these:
        7499999 (100.00%) aligned 0 times
        1 (0.00%) aligned exactly 1 time
        0 (0.00%) aligned >1 times
0.00% overall alignment rate
Time searching: 00:02:48
Overall time: 00:02:48
```

----End

4.50 MACS2 2.1.4 Porting Guide (CentOS 7.6)

4.50.1 Introduction

With the improvement of sequencing techniques, chromatin immunoprecipitation followed by high throughput sequencing (ChIP-Seq) is getting popular to study genome-wide protein-DNA interactions. To address the lack of powerful ChIP-Seq analysis method, we presented the Model-based Analysis of ChIP-Seq (MACS), for identifying transcript factor binding sites. MACS captures the influence of genome complexity to evaluate the significance of enriched ChIP regions and MACS improves the spatial resolution of binding sites through combining the information of both sequencing tag position and orientation. MACS can be easily used for ChIP-Seq data alone, or with a control sample with the increase of specificity.

Programming language: Python 2

Recommended Software Version

MACS2 2.1.4

 Copyright © Huawei Technologies Co., Ltd.

4.50.2 Environment Requirements

Hardware Requirements

Table 4-243 lists the hardware requirements.

Table 4-243 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-244 lists the software requirements.

Table 4-244 Software requirements

Item	Version	Download URL
MACS2	2.1.4	https://codeload.github.com/macs3-project/MACS/zip/v2.1.4
NumPy	3.17.3	https://files.pythonhosted.org/packages/b7/6f/24647f014eef9b67a24adfcbcd4f4928349b4a0f8393b3d7fe648d4d2de3/numpy-1.16.6.zip
Cython	0.29.21	https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz

OS Requirements

Table 4-245 lists the OS requirements.

Table 4-245 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.50.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MACS2 software porting.

Table 4-246 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/MACS2	Installation path of MACS2.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CYTHON	Installation path of Cython.	
4	/path/to/NUMPY	Installation path of NumPy.	

4.50.4 Configuring the Compilation Environment

4.50.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-247 Configuration process

No.	Configuration Item	Remarks
1	python-devel	See **4.50.4.2 Installing python-devel**.
2	Cython	See **4.50.4.3 Installing Cython**.
3	NumPy	See **4.50.4.4 Installing NumPy**.

 Copyright © Huawei Technologies Co., Ltd.

4.50.4.2 Installing python-devel

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the local image to install python-devel through Yum.

```
yum install python-devel
```

----End

4.50.4.3 Installing Cython

Procedure

Method 1

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the **CYTHON** directory:

```
cd /path/to/CYTHON
```

Step 3 Run the following command to install pip for Python 2:

```
pip install cython
```

Alternatively, run the following command:

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple cython
```

----End

Method 2

Step 1 Download the Cython file from the following address:

https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz

Step 2 Run the following command to install Cython:

```
python setup.py install
```

Step 3 Run the following command to load Cython using pip:

```
pip install Cython
```

----End

4.50.4.4 Installing NumPy

Step 1 Run the following command to obtain the NumPy source code:

```
wget  https://files.pythonhosted.org/packages/b7/6f/24647f014eef9b67a24adfcbcd4f4928349b4a0f8393b3d7fe648d4d2de3/numpy-1.16.6.zip
```

Step 2 Run the following command to decompress the installation package:

```
unzip numpy-1.16.6.zip
```

Step 3 Run the following command to go to the directory generated after the decompression:

 Copyright © Huawei Technologies Co., Ltd.

```
cd numpy-1.16.6
```

Step 4 Run the following command to install the program:

```
python setup.py install
```

----End

4.50.5 Obtaining the Source Code

Procedure

Step 1 Download MACS2 source package **MACS2-2.1.4.zip**.

You are advised to download the installation package from the web page of the PC on the Internet and copy it to the server.

Download address: **https://codeload.github.com/macs3-project/MACS/zip/v2.1.4**

Step 2 Use an SFTP tool to upload the downloaded installation package to the */path/to/MACS2* directory on the server.

----End

4.50.6 Compiling and Installing MACS2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the MACS2 installation package:

```
unzip MACS-2.1.4.zip
```

Step 3 Run the following command to go to the directory generated after the decompression:

```
cd MACS-2.1.4
```

Step 4 Run the following commands to start the installation:

```
python setup.py build
python setup.py install
```

Step 5 Run the following commands to set the environment variables and make them take effect:

```
vim /etc/profile
export PATH=/usr/bin:$PATH
export PYTHONPATH=/usr/lib64/python2.7/site-packages:$PYTHONPATH
source /etc/profile
```

----End

4.50.7 Running and Verifying MACS2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to check the version:

 Copyright © Huawei Technologies Co., Ltd.

```
macs2 --version
```

Step 3 Run the following commands to run the test case:

```
cd /path/to/MACS2/MACS-2.1.4/test
sh test.sh game
```

NOTE

Game is a mandatory identifier for running. For details, see **test.sh**.

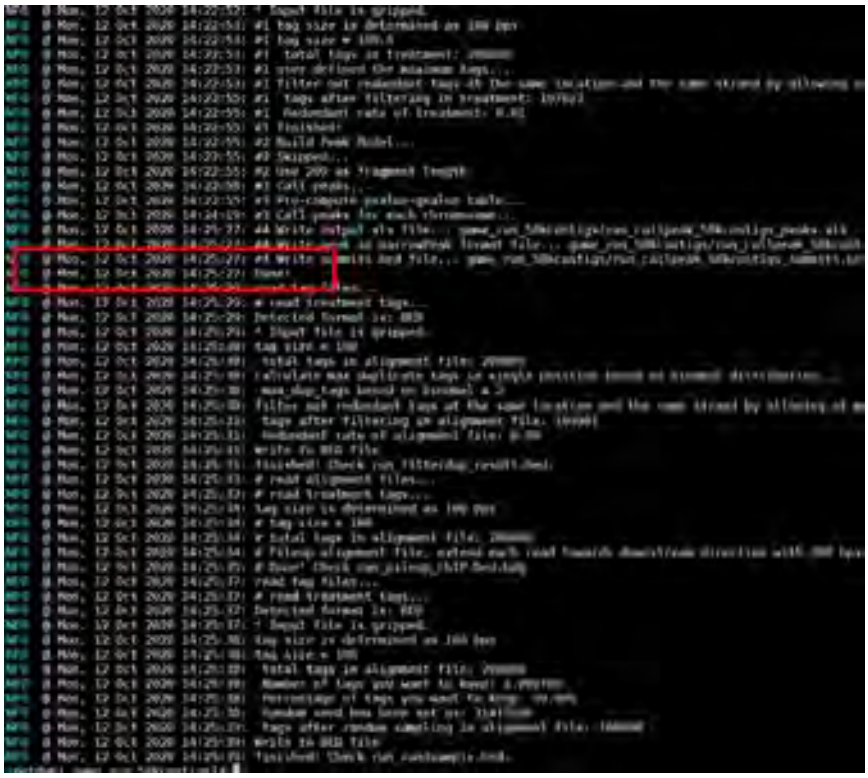
After the commands are executed, the following information is displayed:

```
Last step: search for errors or warnings in log files
game_run_50kcontigs/run_callpeak_50kcontigs.log
game_run_50kcontigs/run_filterdup_50kcontigs.log
game_run_50kcontigs/run_pileup_ChIP.bed.log
game_run_50kcontigs/run_randsample.log
game_run_bdgbroadcall/run_bdgbroadcall_w_prefix.log
game_run_bdgcmp/run_bdgcmp.log
game_run_bdgdiff/run_bdgdiff_w_o_file.log
game_run_bdgdiff/run_bdgdiff_w_prefix.log
game_run_bdgpeakcall/run_bdgpeakcall_w_prefix.log
game_run_callpeak_broad/run_callpeak_broad.log
game_run_callpeak_narrow_revert/run_callpeak_narrow_revert.log
game_run_callpeak_narrow/run_callpeak_narrow0.log
game_run_callpeak_narrow/run_callpeak_narrow1.log
game_run_callpeak_narrow/run_callpeak_narrow2.log
game_run_callpeak_narrow/run_callpeak_narrow3.log
game_run_callpeak_narrow/run_callpeak_narrow4.log
game_run_callpeak_narrow/run_callpeak_narrow5.log
game_run_callpeak_pe_broad/run_callpeak_bampe_broad.log
game_run_callpeak_pe_broad/run_callpeak_bedpe_broad.log
game_run_callpeak_pe_narrow/run_callpeak_bampe_narrow.log
game_run_callpeak_pe_narrow/run_callpeak_bedpe_narrow.log
game_run_callpeak_pe_narrow/run_callpeak_pe_narrow_onlychip.log
game_run_cmbreps/run_cmbreps_fisher.log
game_run_cmbreps/run_cmbreps_max.log
game_run_cmbreps/run_cmbreps_mean.log
game_run_filterdup/run_filterdup_d.log
game_run_filterdup/run_filterdup.log
game_run_filterdup/run_filterdup_pe_d.log
game_run_filterdup/run_filterdup_pe.log
game_run_pileup/run_pileup_ChIP.bed.log
game_run_pileup/run_pileup_ChIPPE.bampe.log
game_run_pileup/run_pileup_ChIPPE.bedpe.log
game_run_pileup/run_pileup_CTRL.bed.log
game_run_pileup/run_pileup_CTRLPE.bampe.log
game_run_pileup/run_pileup_CTRLPE.bedpe.log
game_run_predictd/run_predictd.log
game_run_randsample/run_randsample_bampe.log
game_run_randsample/run_randsample_bedpe.log
game_run_randsample/run_randsample.log
game_run_refinepeak/run_refinepeak_w_ofile.log
game_run_refinepeak/run_refinepeak_w_prefix.log
```

Step 4 Run the following commands to view logs in **game_run_50kcontigs**:

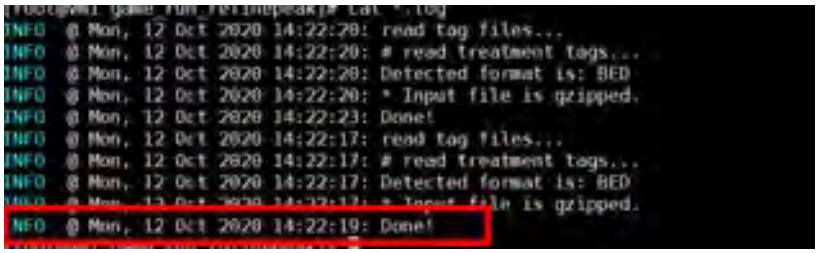
```
cd game_run_50kcontigs/
cat *.log
```

 Copyright © Huawei Technologies Co., Ltd.



Step 5 Run the following commands to view logs in **game_run_refinepeak**:

```
cd game_run_refinepeak/
cat *.log
```



----End

4.50.8 More Resources

To obtain more resources, visit the MACS2 page of GitHub:

https://github.com/macs3-project/MACS

4.51 Trinity 2.11.0 Porting Guide (CentOS 7.6)

4.51.1 Introduction

Trinity, developed at the Broad Institute and the Hebrew University of Jerusalem, represents a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data. Trinity combines three independent software modules: Inchworm, Chrysalis, and Butterfly, applied sequentially to process a large number of RNA-seq reads. Trinity divides the sequence data into many

independent de Bruijn graphs, each representing the transcriptional complexity at a given gene or locus, and then processes each graph independently to extract full-length splicing isoforms and to tease apart transcripts derived from paralogous genes.

For more information about Trinity, visit the **official Trinity website**.

Programming language: Perl

Brief description: Gene assembly

Recommended Software Version

Trinity 2.11.0

4.51.2 Environment Requirements

Hardware Requirements

Table 4-248 lists the hardware requirements.

Table 4-248 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-249 lists the software requirements.

Table 4-249 Software requirements

Item	Version	Download URL
CMake	3.17.3	**https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz**
Boost	1.66.0	**https://boostorg.jfrog.io/artifactory/main/release/1.66.0/source/boost_1_66_0.tar.gz**
Salmon	1.4.0	**https://github.com/COMBINE-lab/salmon/archive/v1.4.0.tar.gz**
Jellyfish	2.2.10	**https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz**
HTSlib	1.11	**https://github.com/samtools/htslib/archive/1.11.tar.gz**
Samtools	1.11	**https://github.com/samtools/samtools/archive/1.11.tar.gz**
Bowtie 2	2.3.5.1	**https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1/bowtie2-2.3.5.1-source.zip**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
sse2neno	master	https://github.com/jratcliff63367/sse2neon/archive/master.zip
SIMDe	v0.7.0-rc-1	https://github.com/simd-everywhere/simde/archive/v0.7.0-rc-1.tar.gz
Python	3.8	https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tgz
Cython	0.29.21	https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz
NumPy	1.19.4	https://files.pythonhosted.org/packages/c5/63/a48648ebc57711348420670bb074998f79828291f68aebfff1642be212ec/numpy-1.19.4.zip
Trinity	2.11.0	https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz
Inchworm	master	https://github.com/trinityrnaseq/Inchworm/archive/master.zip
Chrysalis	master	https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99fdfacda2.zip
seqtk-trinity	master	https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip
bamsifter	master	https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99bfd918da7d7739.zip
Test case 1	SRR3050898_1.fastq	http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_1.fastq.gz
Test case 2	SRR3050898_2.fastq	http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_2.fastq.gz

OS Requirements

Table 4-250 lists the OS requirements.

Table 4-250 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

4.51.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Trinity software porting.

Table 4-251 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/SALMON	Installation path of Salmon.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/JELLYFISH	Installation path of Jellyfish.	
4	/path/to/BOWTIE2	Installation path of Bowtie 2.	
5	/path/to/HTSLIB	Installation path of HTSlib.	
6	/path/to/SAMTOOLS	Installation path of Samtools.	
7	/path/to/PYTHON	Installation path of Python.	
8	/path/to/BOOST	Installation path of Boost.	
9	/path/to/TRINITY	Installation path of Trinity.	
10	/path/to/CMAKE	Installation path of CMake.	
11	/path/to/CASE	Path for storing test cases.	

4.51.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.51.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-252 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing the dependencies	See **4.51.4.2 Installing All Dependencies Using Yum**.
3	Installing Boost	See **4.51.4.3 Installing Boost**.
4	Installing CMake	See **4.51.4.4 Installing CMake**.
5	Installing Salmon	See **4.51.4.5 Installing Salmon**.
6	Installing HTSlib	See **4.51.4.6 Installing HTSlib**.
7	Installing Samtools	See **4.51.4.7 Installing Samtools**.
8	Install Jellyfish.	See **4.51.4.8 Installing Jellyfish**.
9	Installing Bowtie 2	See **4.51.4.9 Installing Bowtie 2**.
10	Installing Python	See **4.51.4.10 Installing Python**.
11	Installing Cython	See **4.51.4.11 Installing Cython**.
12	Installing NumPy	See **4.51.4.12 Installing NumPy**.

4.51.4.2 Installing All Dependencies Using Yum

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following **yum** command to install all the dependencies:

```
yum install -y java-1.8.0-openjdk-devel.aarch64 libcurl libcurl-devel openssl openssl-devel zlib-devel bzip2 bzip2-devel xz-devel boost-devel ncurses ncurses-base ncurses-libs ncurses-devel zlib zlib-devel zlib-static install gcc make zlib zlib-devel bzip2-devel ncurses-devel openssl openssl-devel xz lzma xz-devel readline readline-devel sqlite sqlite-devel gdbm gdbm-devel tk tk-devel libffi libffi-devel libcurl libcurl-devel openssl openssl-devel zlib-devel bzip2 bzip2-devel bzip2-libs xz-devel boost-devel atlas
```

----End

4.51.4.3 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Boost installation package to the */path/to/BOOST* directory on the server.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/BOOST && tar -zxf boost_1_66_0.tar.gz
```

Step 4 Run the following command to go to the installation directory:

```
cd boost_1_66_0
```

Step 5 Run the following commands to perform the compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap.sh --prefix=/path/to/BOOST
./b2 -j 32
./b2 install
```

----End

4.51.4.4 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the CMake installation package to the */path/to/CMAKE* directory on the server.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/CMAKE && tar -zxf cmake-3.17.3.tar.gz
```

Step 4 Run the following command to go to the installation directory:

```
cd cmake-3.17.3
```

Step 5 Run the following commands to perform the compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap --prefix=/path/to/CMAKE
gmake && gmake install
```

----End

4.51.4.5 Installing Salmon

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Salmon installation package to the */path/to/SALMON* directory on the server.

Step 3 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 4 Run the following command to decompress the installation package:

```
cd /path/to/SALMON && tar -zxf v1.4.0.tar.gz
```

Step 5 Run the following command to go to the installation directory:

```
cd salmon-1.4.0
```

Step 6 Run the following command to create a **build** directory and go to the directory:

```
mkdir build && cd build
```

Step 7 Run the following commands to perform the compilation and installation:

```
CC=`which gcc` CXX=`which g++` cmake -DNO_IPO=TRUE -DCMAKE_C_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_INSTALL_PREFIX=/path/to/SALMON -DBOOST_INCLUDEDIR=/path/to/BOOST/include -DBOOST_LIBRARYDIR=/path/to/BOOST/lib ..
make -j && make install
```

----End

4.51.4.6 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the HTSlib installation package to the */path/to/htslib* directory on the server.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/HTSLIB && tar -zxf htslib-1.11.tar.gz
```

Step 4 Run the following command to rename and go to the installation directory:

```
mv htslib-1.11 htslib && cd htslib
```

Step 5 Run the following commands to perform the configuration:

```
autoheader
autoconf
sed -i 's/\-O2/\-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "\-O2" ./`
CC=`which gcc` ./configure --prefix=/path/to/HTSLIB
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

----End

4.51.4.7 Installing Samtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Samtools installation package to the */path/to/SAMTOOLS* directory on the server.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/SAMTOOLS && tar -zxf samtools-1.11.tar.gz
```

Step 4 Run the following command to go to the installation directory:

```
cd samtools-1.11
```

Step 5 Run the following commands to perform the configuration:

```
autoheader
autoconf
sed -i 's/\-O2/\-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "\-O2" ./`
CC=`which gcc` ./configure --prefix=/path/to/SAMTOOLS --with-htslib=/path/to/HTSLIB
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j 16
make install
```

----End

4.51.4.8 Installing Jellyfish

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Jellyfish installation package to the */path/to/JELLYFISH* directory on the server.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/JELLYFISH && tar -zxf jellyfish-2.2.10.tar.gz
```

Step 4 Run the following command to go to the directory generated after the decompression:

```
cd jellyfish-2.2.10
```

Step 5 Run the following command to perform the configuration:

```
CC=`which gcc` CXX=`which g++` ./configure --prefix=/path/to/JELLYFISH CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110 -flto' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110 -flto'
```

Step 6 Run the following command to perform the installation and compilation:

```
make -j && make install
```

----End

4.51.4.9 Installing Bowtie 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the installation packages of Bowtie 2, SIMDe, and sse2neon to the */path/to/BOWTIE2* directory.

Step 3 Run the following commands to decompress the installation package and go to the directory generated after the decompression:

```
cd /path/to/BOWTIE2
unzip bowtie2-2.3.5.1-source.zip
tar -zxf v0.7.0-rc-1.tar.gz
unzip master.zip
cd bowtie2-2.3.5.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to copy the SIMDe files to Bowtie 2:

```
cp -a ../simde-0.7.0-rc-1/simde third_party/simde
```

Step 5 Run the following command to copy the **sse2neon.h** file to Bowtie 2:

```
cp -a ../sse2neon-master/SSE2NEON.h third_party/simde/simde/x86/
```

Step 6 Run the following command to perform the compilation:

```
CC=`which gcc` CXX=`which g++` CXXFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' make NO_TBB=1 POPCNT_CAPABILITY=0 all
```

----End

4.51.4.10 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Python installation package to the */path/to/PYTHON* directory on the server.

Step 3 Run the following commands to go to the installation directory and decompress the installation package:

```
cd /path/to/PYTHON
tar -zxf Python-3.8.0.tgz && cd Python-3.8.0
```

Step 4 Run the following command to configure the compilation information:

```
./configure --prefix=/path/to/PYTHON --enable-shared --enable-loadable-sqlite-extensions --enable-optimization
```

Step 5 Run the following command to perform the compilation and installation:

```
make -j && make install
```

----End

4.51.4.11 Installing Cython

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the Cython installation package to the */path/to/PYTHON* directory on the server.

Step 3 Run the following commands to set the environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following command to decompress the installation package and go to the installation directory:

```
tar -zxf Cython-0.29.21.tar.gz && cd Cython-0.29.21
```

Step 5 Run the following command to perform the compilation and installation:

```
python3 setup.py build && python3 setup.py install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.51.4.12 Installing NumPy

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use an SFTP tool to upload the NumPy installation package to the */path/to/PYTHON* directory on the server.

Step 3 Run the following commands to set the environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following command to decompress the installation package and go to the installation directory:

```
unzip numpy-1.19.4.zip && cd numpy-1.19.4
```

Step 5 Run the following command to perform the compilation and installation:

```
python3 setup.py build && python3 setup.py install
```

----End

4.51.5 Obtaining the Source Code

Procedure

Step 1 Download the installation packages of Trinity, Inchworm, Chrysalis, seqtk-trinity, bamsifter, and HTSlib.

Download Trinity at **https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz**

Download Inchworm at **https://github.com/trinityrnaseq/Inchworm/archive/master.zip**

Download Chrysalis at **https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99fdfacda2.zip**

Download seqtk-trinity at **https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip**

Download bamsifter at **https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99bfd918da7d7739.zip**

Download HTSlib at **https://github.com/samtools/htslib/archive/1.11.zip**

Step 2 Use an SFTP tool to upload the installation packages of Trinity, Inchworm, and Chrysalis to the */path/to/TRINITY* directory.

----End

4.51.6 Compiling and Installing Trinity

Procedure

Step 1 Run the following commands to go to the installation directory and decompress the installation packages:

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/TRINITY
tar -zxf v2.11.0.tar.gz
unzip master.zip
unzip 8ef13ed1fa618186ec3dc04522115c99fdfacda2.zip
unzip f034f247d5fa1fd9847c390d99bfd918da7d7739.zip
unzip f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip
unzip 1.11.zip
```

Step 2 Run the following commands to save the source code of Inchworm, Chrysalis, seqtk-trinity, and bamsifter to the corresponding directories in the Tinity source code:

```
cp -rf Chrysalis-8ef13ed1fa618186ec3dc04522115c99fdfacda2/* trinityrnaseq-2.11.0/Chrysalis
cp -rf Inchworm-master/* trinityrnaseq-2.11.0/Inchworm
cp -rf seqtk-trinity-f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c/* trinityrnaseq-2.11.0/trinity-plugins/seqtk-trinity
cp -rf bamsifter-f034f247d5fa1fd9847c390d99bfd918da7d7739/* trinityrnaseq-2.11.0/trinity-plugins/bamsifter
cp -rf htslib-1.11/* trinityrnaseq-2.11.0/trinity-plugins/bamsifter/htslib
```

Step 3 Run the following command to decompress **ParaFly-0.1.0.tar.gz**:

```
cd trinityrnaseq-2.11.0/trinity-plugins && tar -zxf ParaFly-0.1.0.tar.gz
```

Step 4 Run the following commands to replace the parameter **-m64** by **-mabi=lp64**:

```
cd /path/to/TRINITY/trinityrnaseq-2.11.0
sed -i '/tar -zxvf ${PARAFLY_CODE}.tar.gz/d' trinity-plugins/Makefile
sed -i 's/\-m64/\-mabi=lp64/g' `grep -rl "\-m64" ./`
```

Step 5 Run the following commands to add the compilation optimization parameters:

```
sed -i 's/cmake/cmake -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110"/g' ./Chrysalis/Makefile ./Inchworm/Makefile
sed -i 's/\-fopenmp/-O3 -march=armv8.2-a -mtune=tsv110 -fopenmp/g' ./trinity-plugins/Makefile
```

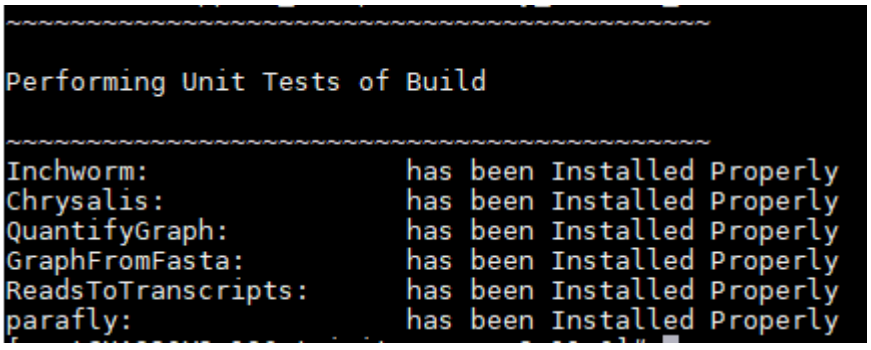
Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 7 Run the following command to perform the compilation and installation:

```
make -j
```

If the information shown in the following figure is displayed, the installation is successful:



----End

4.51.7 Running and Verifying Trinity

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following commands to set the environment variables:

```
export PATH=/path/to/SAMTOOLS/bin:$PATH
export PATH=/path/to/SALMON/bin:$PATH
export PATH=/path/to/JELLYFISH/bin:$PATH
export PATH=/path/to/BOWTIE2/bowtie2-2.3.5.1:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
export PATH=/path/to/PYTHON/bin:$PATH
export PATH=/path/to/TRINITY/trinityrnaseq-2.11.0:$PATH
export LD_LIBRARY_PATH=/path/to/HTSLIB/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to go to the test case directory and decompress the case file:

```
cd /path/to/CASE
gzip -d SRR3050898_1.fastq.gz && gzip -d SRR3050898_2.fastq.gz
```

Step 4 Run the following command to change the maximum size of the Java heap to 30 GB:

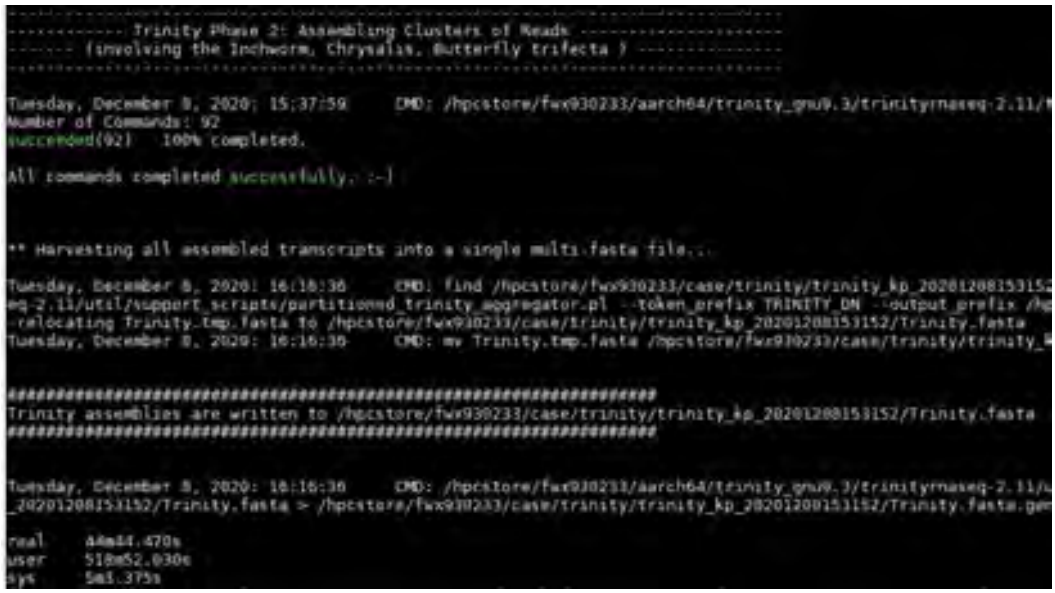
```
sed -i 's/bflyHeapSpaceMax = "10G"/bflyHeapSpaceMax = "30G"/g' /path/to/TRINITY/
trinityrnaseq-2.11.0/Trinity
```

Step 5 Run the following command to run the test case:

```
{ time Trinity --seqType fq --left SRR3050898_1.fastq --right SRR3050898_2.fastq --CPU 128 --
max_memory 200G --output trinity_kp_`date '+%Y%m%d%H%M%S'`; } 2>&1 | tee -a
`hostname`_`date '+%Y%m%d%H%M%S'`.log
```

Figure 4-23 shows an example of the running result.

Figure 4-23 Result example



----End

4.52 LDBlockshow 1.36 Porting Guide (CentOS 7.6)

4.52.1 Introduction

LDBlockShow is a fast and effective tool to generate linkage disequilibrium (LD) heatmap from VCF files. It is more time and memory saving than other current tools. LDBlockShow can generate the plots of LD heatmap and interested statistics

or annotation results simultaneously. In addition, it also supports subgroup analysis.

For more information, visit the **official LDBlockShow website**.

Programming language: Perl/HTML/C++/Shell

One-sentence description: a quick and effective tool for generating LD heat maps from VCF files

Open source license: MIT License

Recommended Software Version

LDBlockShow 1.36

4.52.2 Environment Requirements

Hardware Requirements

Table 4-253 lists the hardware requirements.

Table 4-253 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-254 lists the software requirements.

Table 4-254 Software requirements

Software	Version	Download Address
SVG	2.85	**https://www.cpan.org/modules/by-module/SVG/SVG-2.85.tar.gz**
LDBlockShow	1.36	**https://github.com/BGI-shenzhen/LDBlockShow/archive/v1.36.tar.gz**
PLINK	1.9	**https://github.com/chrchang/plink-ng/archive/master.tar.gz**
Test case	example	Built-in Example1/Example2/Example3/Example4

OS Requirements

Table 4-255 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-255 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

4.52.3 Paths for Software Porting

This chapter lists the software installation paths involved in the LDBlockShow software porting.

Table 4-256 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/LDBLOCKSHOW	Installation path of LDBlockshow.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/SVG	Installation path of SVG.	
4	/path/to/PLINK	Installation path of PLINK.	
5	/path/to/ZLIB	Installation path of zlib.	

4.52.4 Obtaining the Source Code

Procedure

Step 1 Run the following command to download the SVG installation package:

```
wget https://www.cpan.org/modules/by-module/SVG/SVG-2.85.tar.gz
```

Step 2 Run the following command to download the LDBlockShow installation package:

```
wget https://github.com/BGI-shenzhen/LDBlockShow/archive/v1.36.tar.gz --no-check-certificate
```

Step 3 Run the following command to download the PLINK installation package:

```
wget https://github.com/chrchang/plink-ng/archive/master.tar.gz --no-check-certificate
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run the following command to download the zlib installation package:

```
wget https://udomain.dl.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz
```

----End

4.52.5 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-257 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

4.52.6 Compiling and Installing LDBlockShow

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the base dependency:

```
yum install zlib perl-ExtUtils-CBuilder perl-ExtUtils-MakeMaker
```

Step 3 Run the following commands to configure the compilers:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 4 Run the following commands to install SVG:

```
tar -xzf SVG-2.85.tar.gz
cd SVG-2.85
perl Makefile.PL
make
make install
```

Step 5 Run the following commands to install LDBlockShow:

```
tar -zxvf v1.36.tar.gz
cd LDBlockShow-1.36
chmod +x configure
./configure
make
cp LDBlockShow ./bin/
cp: overwrite './bin/LDBlockShow'? y
```

Step 6 Run the following commands to install PLINK:

```
yum install blas-devel lapack-devel atlas-devel perl-Digest-SHA -y
ln -s /usr/lib64/atlas/libtatlas.so.3.10 /usr/lib64/atlas/libatlas.so
```

 Copyright © Huawei Technologies Co., Ltd.

```
ln -s /usr/lib64/atlas/libsatlas.so.3.10 /usr/lib64/atlas/libcblas.so
tar -zxf master.tar.gz
mv zlib-1.2.11.tar.gz plink-ng-master
cd plink-ng-master/1.9 && sed -i '0,/curl/s/curl/# curl/g' plink_first_compile
./plink_first_compile
cp plink /path/to/LDBLOCKSHOW/LDBlockShow-1.36/bin/
cp: overwrite '/path/to/LDBLOCKSHOW/LDBlockShow-1.36/bin/plink'? y
```

----End

4.52.7 Running and Verifying LDBlockShow

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to start LDBlockShow:

```
cd /path/to/LDBLOCKSHOW/LDBlockShow-1.36/example/Example1
sh run.sh
```

The following figure shows the command output:

```
Start Time :
Thu Dec 24 17:09:54 CST 2020
#Detected VCF File is phased file with '|', Read VCF in Phase mode
##Start Region Cal...    :chr11 24100000 24200000; In This Region TotalSNP Number is 7
find blocks...
Start draw... SVG info: SNPNumber :7 , SVG (width,height) = (455,385)
convert   SVG ---> PNG ...
ALL done
End Time :
Thu Dec 24 17:09:54 CST 2020
```

----End

4.53 bioBakery Porting Guide (CentOS 8.2)

4.53.1 Introduction

bioBakery workflows is a collection of workflows and tasks for executing common microbial community analyses using standardized, validated tools and parameters. Quality control and statistical summary reports are automatically generated for most data types, which include 16S amplicons, metagenomes, and metatranscriptomes. Workflows are run directly from the command line and tasks can be imported to create your own custom workflows. The workflows and tasks are built with **AnADAMA2** which allows for parallel task execution locally and in a grid compute environment. bioBakery workflows is a branch of Anaconda.

For more information about bioBakery, visit the **bioBakery official website**.

Programming language: Python

Brief description: bioBakery workflows is a collection of workflows and tasks for executing common microbial community analyses.

Open source license: MIT License

Recommended Software Version

bioBakery workflows 3.0.0a7

 Copyright © Huawei Technologies Co., Ltd.

4.53.2 Environment Requirements

Hardware Requirements

Table 4-258 lists the hardware requirements.

Table 4-258 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-259 lists the software requirements.

Table 4-259 Software requirements

Item	Version	Download Address
bioBakery workflows	3.0.0a7	**https://pypi.io/packages/source/b/biobakery_workflows/biobakery_workflows-3.0.0a7.tar.gz**
Anaconda	2021.11	**https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh**

OS Requirements

Table 4-260 lists the OS requirements.

Table 4-260 OS requirements

Item	Version	Download URL
CentOS	8.2	**https://vault.centos.org/8.2.2004/isos/aarch64/**

4.53.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bioBakery software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-261 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/root/anaconda3	Installation path of Anaconda.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/BIOBAKERY	Installation path of bioBakery.	

4.53.4 Configuring the Compilation Environment

Configure Process

Table 4-262 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Anaconda	See **4.53.4.1 Installing Anaconda**.

4.53.4.1 Installing Anaconda

Procedure

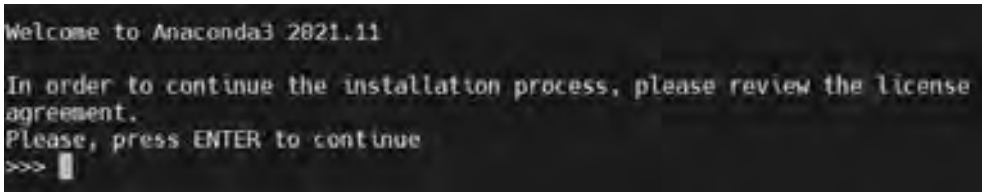
Step 1 Obtain the Anaconda installation package.

```
wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh
```

Step 2 Install Anaconda.

```
bash Anaconda3-2021.11-Linux-aarch64.sh
```

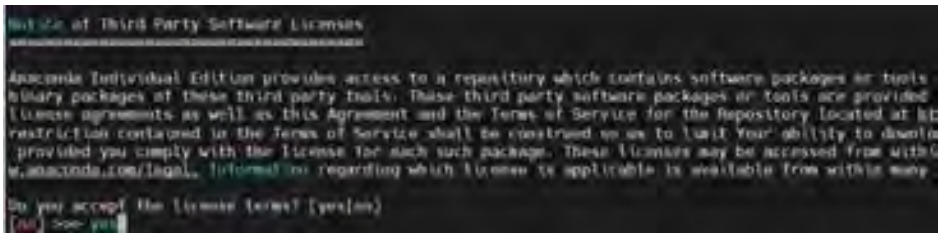
1. When the following information is displayed, press **Enter**.



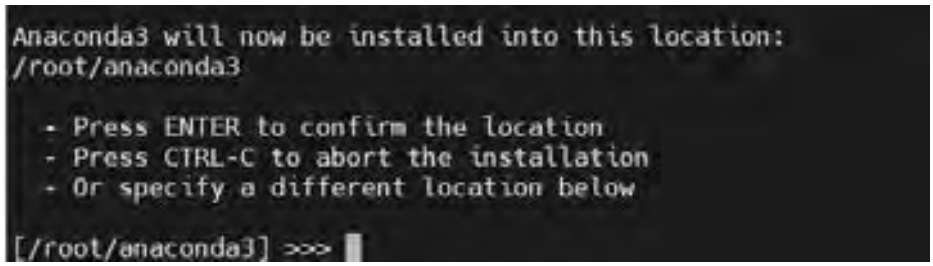
 Copyright © Huawei Technologies Co., Ltd.

2. When the following information is displayed, type **q**.

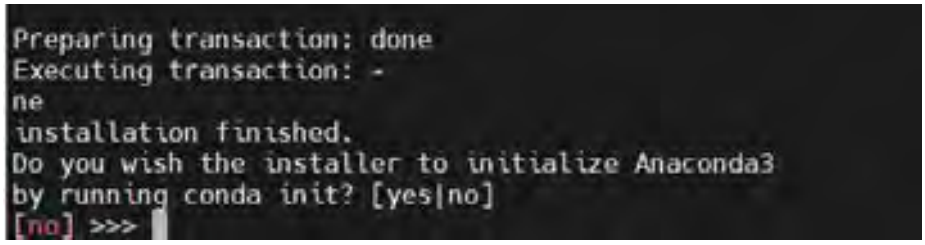
3. When the following information is displayed, type **yes** and press **Enter**.



4. On the screen shown in the following figure, type a desired Anaconda installation path. You can also press **Enter** to select the default installation path **/root/anaconda3**.

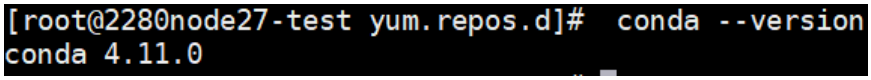


5. Press **Enter**. No initialization operations will be performed.



Step 3 Set the conda environment variables.

1. Set the environment variables.
   source */root/anaconda3*/**etc/profile.d/conda.sh**
2. Check whether conda has taken effect.
   conda -version



3. Initialize conda.
   ```
   conda init bash
   ```
4. Restart the shell. You can see that conda has entered the base environment.
   ```
   (base) [root@localhost ~]#
   ```

Step 4 Configure and update conda.

```
conda config --set auto_activate_base False
conda config --set auto_update_conda False
conda config --set channel_priority flexible
conda update --all -n base
```

If the update is required, type **y** and press **Enter**.

----End

4.53.5 Compilation and Installation

Procedure

Step 1 Add a local conda repository.

```
conda config --add channels file://root/anaconda3/conda-bld/linux-aarch64
```

NOTE

The default installation path is **/root/anaconda3**. You can replace it with the actual path.

Step 2 Create a bioBakery compilation and running environment.

```
conda create -n bioBakery
```

Step 3 Activate the bioBakery environment.

```
conda activate bioBakery
```

Step 4 Create a **biobakery_workflows** directory and a **recipe** directory.

```
mkdir -p biobakery_workflows/recipe
```

Step 5 Create a **conda_build_config.yaml** file.

1. Create a **conda_build_config.yaml** file.
   ```
   vi conda_build_config.yaml
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   r_base: '3.4'
   pin_run_as_build:
     python:
       max_pin: x.x
       min_pin: x.x
   python: '2.7'
   numpy: '1.11'
   perl: 5.26.0
   lua: '5'
   cpu_optimization_target: nocona
   target_platform: linux-aarch64
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Create a **meta.yaml** file.

1. Create a **meta.yaml** file.
```
vi meta.yaml
```
2. Press **i** to enter the edit mode and add the following content:
```
# This file created by conda-build 3.0.27
# meta.yaml template originally from:
# /n/home05/ljmciver/conda-biobakery/recipes/biobakery_workflows, last modified Fri Feb 19
18:18:45 2021
# ------------------------------------------------

package:
   name: biobakery_workflows
   version: 3.0.0a7
source:
   fn: biobakery_workflows-3.0.0a7.tar.gz
   sha256: 8819d6a110d01005d5c855a5f6a906b0a6b75e33d20b96471ae41c5ab354203f
   url: https://pypi.io/packages/source/b/biobakery_workflows/biobakery_workflows-3.0.0a7.tar.gz
build:
   entry_points:
      - biobakery_workflows = biobakery_workflows.biobakery_workflows:main
      - biobakery_workflows_databases = biobakery_workflows.biobakery_workflows_databases:main
   noarch: python
   number: '0'
   script: python setup.py install  --single-version-externally-managed --record=record.txt
   string: pyh0b62126_0
requirements:
   build:
      - libgcc-ng 11.2.0
      - openssl 1.1.1l
      - ncurses 6.2
      - python_abi 3.8
      - wheel 0.37.0
      - libgomp 11.2.0
      - setuptools 59.2.0
      - pip 21.3.1
      - sqlite 3.36.0
      - ld_impl_linux-aarch64 2.36.1
      - _libgcc_mutex 0.1
      - readline 8.1
      - _openmp_mutex 4.5
      - libffi 3.4.2
      - zlib 1.2.11
      - python 3.8.12
      - libstdcxx-ng 11.2.0
      - tk 8.6.11
      - certifi 2021.10.8
      - ca-certificates 2021.10.8
   run:
      - python 3.8.12
      - anadama2
test:
   commands:
      - biobakery_workflows --help
      - biobakery_workflows_databases --help
   imports:
      - biobakery_workflows
      - biobakery_workflows.data
      - biobakery_workflows.document_templates
      - biobakery_workflows.tasks
      - tests
about:
   description: bioBakery workflows is a collection of workflows and tasks for executing
      common microbial community analyses using standardized, validated tools and
      parameters. Quality control and statistical summary reports are automatically
      generated for most data types, which include 16S amplicons, metagenomes, and
      metatranscriptomes. Workflows are run directly from the command line and tasks
```

```
      can be imported to create your own custom workflows. The workflows and tasks
      are built with AnADAMA2 which allows for parallel task execution locally and
      in a grid compute environment.
  doc_url: http://huttenhower.sph.harvard.edu/biobakery_workflows
  home: http://huttenhower.sph.harvard.edu/biobakery_workflows
  license: MIT
  license_family: MIT
  summary: 'bioBakery workflows: A collection of meta-omic analysis workflows'
extra:
  copy_test_source_files: true
  final: true
  recipe-maintainers: biobakery-users@googlegroups.com
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 7 Create a **meta.yaml.template** file.

1. Create a **meta.yaml.template** file.

```
vi meta.yaml.template
```

2. Press **i** to enter the edit mode and add the following content:

```
{% set name = "biobakery_workflows" %}
{% set version = "3.0.0a7" %}
{% set hash_type = "sha256" %}
{% set hash_value = "8819d6a110d01005d5c855a5f6a906b0a6b75e33d20b96471ae41c5ab354203f"
%}

package:
  name: '{{ name|lower }}'
  version: '{{ version }}'

source:
  fn: '{{ name }}-{{ version }}.tar.gz'
  url: https://pypi.io/packages/source/{{ name[0] }}/{{ name }}/{{ name }}-{{ version }}.tar.gz
  '{{ hash_type }}': '{{ hash_value }}'

build:
  noarch: python
  number: 0
  script: python setup.py install  --single-version-externally-managed --record=record.txt

  entry_points:
    - biobakery_workflows = biobakery_workflows.biobakery_workflows:main
    - biobakery_workflows_databases = biobakery_workflows.biobakery_workflows_databases:main

requirements:
  build:
    - python
    - setuptools
  run:
    - python
    - anadama2
    - kneaddata
    - humann

test:
  imports:
    - biobakery_workflows
    - biobakery_workflows.data
    - biobakery_workflows.document_templates
    - biobakery_workflows.tasks
    - tests
  commands:
    - biobakery_workflows --help
    - biobakery_workflows_databases --help

about:
  home: http://huttenhower.sph.harvard.edu/biobakery_workflows
  license: MIT
  license_family: MIT
```

 Copyright © Huawei Technologies Co., Ltd.

```
  summary: 'bioBakery workflows: A collection of meta-omic analysis workflows'
  description: 'bioBakery workflows is a collection of workflows and tasks for executing common
microbial community analyses using standardized, validated tools and parameters. Quality control
and statistical summary reports are automatically generated for most data types, which include 16S
amplicons, metagenomes, and metatranscriptomes. Workflows are run directly from the command
line and tasks can be imported to create your own custom workflows. The workflows and tasks are
built with AnADAMA2 which allows for parallel task execution locally and in a grid compute
environment.'
  doc_url: http://huttenhower.sph.harvard.edu/biobakery_workflows

extra:
  recipe-maintainers: 'biobakery-users@googlegroups.com'
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 8 Exit from the **biobakery_workflows** and **recipe** directories.

```
cd ../..
```

Step 9 Compile biobakery_workflows.

```
conda build ./biobakery_workflows/recipe
```

NOTE

If the compilation fails, add the Tsinghua source and try again.

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free conda
config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge conda
config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda
```

Step 10 Install biobakery_workflows.

```
conda install biobakery_workflows
```

----End

4.53.6 Running and Verification

Procedure

Step 1 Activate the bioBakery environment.

```
conda activate biobakery
```

Step 2 View the version information.

```
biobakery_workflows --version
biobakery_workflows v3.0.0-alpha.7
```

----End

4.53.7 More Resources

Anaconda official library:

https://anaconda.org/

4.54 Bioconda Porting Guide (CentOS 8.2)

4.54.1 Introduction

Bioconda is a channel of Conda. Conda is an open-source software package manager and environment manager. It is used to install multiple versions of

software packages and their dependencies and easily switch between them. Bioconda is a channel for the Conda package manager specializing in bioinformatics software. It consists of a bundle of software which can be installed directly.

For more information about Bioconda, visit the **Bioconda official website**.

Programming languages: Python, Shell

Brief description: Bioconda is a channel of Conda.

Open source license: MIT license

Recommended Software Version

Anaconda3-2021.11

4.54.2 Environment Requirements

Hardware Requirements

Table 4-263 lists the hardware requirements.

Table 4-263 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-264 lists the software requirements.

Table 4-264 Software requirements

Item	Version	Download Address
Anaconda	2021.11	**https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh**

OS Requirements

Table 4-265 lists the OS requirements.

Table 4-265 OS requirements

Item	Version	Download URL
CentOS	8.2	**https://vault.centos.org/8.2.2004/isos/aarch64/**

 Copyright © Huawei Technologies Co., Ltd.

4.54.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Bioconda software porting.

Table 4-266 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/root/anaconda3	Installation path of Anaconda.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

4.54.4 Configuring the Compilation Environment

Configure Process

Table 4-267 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.54.5 Compilation and Installation

Procedure

Step 1 Obtain the Anaconda installation package.

```
wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh
```

Step 2 Install Anaconda.

```
bash Anaconda3-2021.11-Linux-aarch64.sh
```

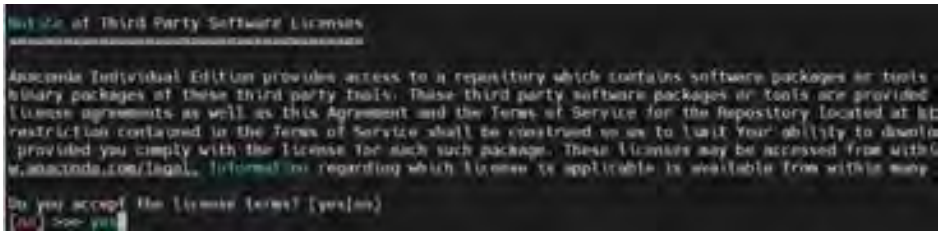
1. When the following information is displayed, press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

```
Welcome to Anaconda3 2021.11

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

2. When the following information is displayed, type **q**.

3. When the following information is displayed, type **yes** and press **Enter**.



4. On the screen shown in the following figure, type a desired Anaconda installation path. You can also press **Enter** to select the default installation path **/root/anaconda3**.

```
Anaconda3 will now be installed into this location:
/root/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/root/anaconda3] >>>
```

5. Press **Enter**. No initialization operations will be performed.

```
Preparing transaction: done
Executing transaction: -
ne
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>>
```

Step 3 Set the conda environment variables.

1. Set the environment variables.
   source */root/anaconda3*/**etc/profile.d/conda.sh**
2. Check whether conda has taken effect.
   conda -version

   ```
   [root@2280node27-test yum.repos.d]#  conda --version
   conda 4.11.0
   ```

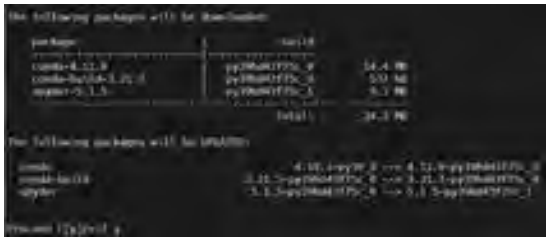
3. Initialize conda.
   conda init bash
4. Restart the shell. You can see that conda has entered the base environment.

   ```
   (base) [root@localhost ~]#
   ```

Step 4 Configure and update conda.

```
conda config --set auto_activate_base False
conda config --set auto_update_conda False
conda config --set channel_priority flexible
conda update --all -n base
```

If the update is required, type **y** and press **Enter**.



Step 5 Add a Bioconda source.

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda
conda config --set show_channel_urls yes
```

----End

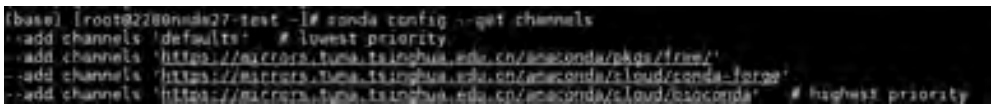
4.54.6 Running and Verification

Procedure

Step 1 Run the following command to view the Bioconda source:

```
conda config --get channels
```

If information similar to the following is displayed, Bioconda is successfully installed.



----End

4.54.7 More Resources

Anaconda official library:

 Copyright © Huawei Technologies Co., Ltd.

https://anaconda.org/

4.55 Boost 1.72 Porting Guide (openEuler 20.03)

4.55.1 Introduction

Boost is a set of C++ libraries that extend the functionality of the C++ Standard Library. Boost is a portable set of C++ libraries that provide source code. As a backup of the Standard Library, Boost is one of the development engines for the C ++ standardization process.

For more information about Boost, visit the **Boost official website**.

Programming language: C++

Brief description: Boost is a set of C++ libraries that extend the functionality of the C++ Standard Library.

Open source license: BSL-1.0 License

Recommended Software Version

Boost 1.72

4.55.2 Environment Requirements

Hardware Requirements

Table 4-268 lists the hardware requirements.

Table 4-268 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-269 lists the software requirements.

Table 4-269 Software requirements

Item	Version	Download Address
Boost	1.72	https://sourceforge.net/projects/boost/files/boost/1.72.0/boost_1_72_0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-270 lists the OS requirements.

Table 4-270 OS requirements

Item	Version	Download Address
openEuler	20.03	https://www.openeuler.org/en/download/
Kernel	4.19.90-2106.3.0.0095.oe1	https://www.openeuler.org/en/download/

4.55.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Boost software porting.

Table 4-271 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in User Guide (GCC for openEuler).
3	/path/to/BOOST	Installation path of Boost.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

4.55.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configure Process

Table 4-272 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See **User Guide (GCC for openEuler)**.

4.55.5 Obtaining the Source Code

Procedure

Step 1 Download the Boost1.72 installation package.

https://sourceforge.net/projects/boost/files/boost/1.72.0/boost_1_72_0.tar.gz

Step 2 Use an SFTP tool to upload the Boost1.72 installation package to the */path/to/BOOST* directory on the server.

----End

4.55.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Boost installation package.

```
cd /path/to/BOOST
tar -xvf  boost_1_72_0.tar.gz
```

Step 3 Go to the directory generated after decompression.

```
cd boost_1_72_0/
```

Step 4 Perform the compilation.

```
./bootstrap.sh
./b2
./b2 install --prefix=/path/to/BOOST/BOOST
```

----End

4.55.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Boost directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/BOOST/BOOST
```

Step 3 Query the version.

```
cat /path/to/BOOST/BOOST/include/boost/version.hpp
```

----End

4.56 Bustools 0.41.0 Porting Guide (CentOS 8.2)

4.56.1 Introduction

Bustools is a program for manipulating BUS files for single cell RNA-Seq datasets. It can be used to error correct barcodes, collapse UMIs, produce gene count or transcript compatibility count matrices, and is useful for many other tasks.

For more information about Bustools, visit the **Bustools official website**.

Programming languages: C/C++

Brief description: Bustools is a program for manipulating BUS files for single cell RNA-Seq datasets.

Open source license: GPLv3

Recommended Software Version

Bustools v0.41.0

4.56.2 Environment Requirements

Hardware Requirements

Table 4-273 lists the hardware requirements.

Table 4-273 Hardware

Item	Remarks
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 4-274 lists the software requirements.

Table 4-274 Software requirements

Item	Version	Download Address
Bustools	0.46.2	**https://github.com/BUStools/bustools**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
CMake	3.20.2	They are installed using Yum and do not need to be manually downloaded.
GCC	8.5.0	
Git	/	

OS Requirements

Table 4-275 lists the OS requirements.

Table 4-275 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.56.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Bustools software porting.

Table 4-276 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/usr/local/bin/bustools	Installation path of Bustools.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.56.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-277 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing GCC, CMake, and Git	See **4.56.4.1 Installing GCC, CMake, and Git**.

4.56.4.1 Installing GCC, CMake, and Git

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC, CMake, and Git.

```
yum install gcc-c++
yum install cmake
yum install git
```

Step 3 Perform the verification.

```
gcc -v
cmake --version
git --version
```

If the correct version is displayed, the installation is successful.

----End

4.56.5 Obtaining the Source Code

Procedure

Step 1 Download the Bustools installation package.

```
cd /usr/local/bin/
git clone https://github.com/BUStools/bustools.git
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.56.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

```
cd /usr/local/bin/bustools
mkdir build
cd build
cmake ..
make
make install
```

Step 3 Perform the verification.

```
bustools
```

----End

4.57 Circos 0.69.9 Porting Guide (CentOS 8.2)

4.57.1 Introduction

Circos is free software licensed under GPL. It is written in Perl, can be deployed on any operating system for which Perl is available (e.g. Windows, Mac OS X, Linux and other UNIX flavours), and produces bitmap (PNG) and vector (SVG) images using plain text configuration and input files. Circos was originally conceived for visualizing genomic data such as alignments and structural variation. Over time, support was added for 2D data tracks such as line, scatter, heatmap and histogram plots.

For more information about Circos, visit the **Circos official website**.

Programming language: Perl

Brief description: Circos is free software that can run on any operating system with a Perl interpreter, such as Windows, Mac OS X, Linux, and other UNIX-like systems.

Open source license: GPL

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

Circos0.69-9

4.57.2 Environment Requirements

Hardware Requirements

Table 4-278 lists the hardware requirements.

Table 4-278 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-279 lists the software requirements.

Table 4-279 Software requirements

Item	Version	Download Address
Perl	5.34	https://www.cpan.org/src/5.0/perl-5.34.0.tar.gz
xz	5.25	https://nchc.dl.sourceforge.net/project/lzmautils/xz-5.2.5.tar.gz
libgd	2.25	https://codeload.github.com/libgd/libgd/zip/refs/tags/gd-2.2.5

OS Requirements

Table 4-280 lists the OS requirements.

Table 4-280 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.57.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Circos software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-281 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	/path/to/perl/	Installation path of Perl.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/circos	Installation path of Circos	

4.57.4 Configuring the Compilation Environment

4.57.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-282 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Perl	See **4.57.4.2 Installing Perl**.
3	Installing libgd	See **4.57.4.3 Installing libgd**.
4	Installing GD	See **4.57.4.4 Installing GD**.

 Copyright © Huawei Technologies Co., Ltd.

4.57.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the **cd** command to go to the directory where the installation package is saved.

Step 3 Decompress the installation package.

```
tar -zxvf perl-5.34.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd perl-5.34.0
```

Step 5 Perform the compilation and installation.

```
./Configure -des -Dprefix=/path/to/perl/perl-5.34.0/ -Dusethreads -Uinstalluserbinperl -Dcc=gcc
make
make test
make install
```

Step 6 Update the environment variables.

```
export PERL5LIB=/path/to/perl/perl-5.34.0/lib
export PATH=/path/to/perl/perl-5.34.0/bin:$PATH
```

Step 7 Check whether the installation is successful.

```
perl -version
```

If the correct version information is displayed, the installation is successful.

----End

4.57.4.3 Installing libgd

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the **cd** command to go to the directory where the installation package is saved.

Step 3 Decompress the installation package.

```
unzip gd-2.2.5.zip
```

Step 4 Go to the directory generated after the decompression.

```
cd libgd-gd-2.2.5
```

Step 5 Perform the compilation and installation.

```
./bootstrap.sh
./configure --enable-shared
make
make install
```

Step 6 Check whether the installation is successful.

```
gdlib-config --version
```

If the correct version information is displayed, the installation is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.57.4.4 Installing GD

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

```
yum -y install perl-CPAN
```

Step 3 If you use CPAN for the initial installation, use the default settings.

```
cpan -i GD
```

----End

4.57.5 Obtaining the Source Code

Procedure

Step 1 Download the Circos installation package.

http://circos.ca/distribution/circos-0.69-9.tgz

Step 2 Use an SFTP tool to upload the Circos installation package to the */path/to/circos* directory on the server.

----End

4.57.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/circos
```

Step 3 Decompress the installation package.

```
tar -zxvf circos-0.69-9.tgz
```

Step 4 Install the required Perl modules.

```
cd circos-0.69-9/bin
```

```
[root@localhost bin]# ls
circos  circos.exe  compile.bat  compile.make  gddiag  list.modules  test.modules
[root@localhost bin]#
```

Step 5 List the Perl modules.

```
./list.modules
```

Step 6 Check whether the required Perl modules are correctly installed.

```
./test.modules
```

Step 7 Check the modules in sequence.

```
./circus -modules
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost bin]# ./circos -modules
ok       1.52 Carp
ok       0.45 Clone
ok       2.63 Config::General
ok       3.80 Cwd
ok      2.179 Data::Dumper
ok       2.58 Digest::MD5
ok       2.85 File::Basename
ok       3.80 File::Spec::Functions
ok     0.2311 File::Temp
ok       1.52 FindBin
ok       0.39 Font::TTF::Font
ok       2.73 GD
ok        0.2 GD::Polyline
ok       2.52 Getopt::Long
ok       1.46 IO::File
ok      0.430 List::MoreUtils
ok       1.55 List::Util
ok       0.01 Math::Bezier
ok   1.999818 Math::BigFloat
ok       0.07 Math::Round
ok       0.08 Math::VecStat
ok    1.03_01 Memoize
ok       1.97 POSIX
ok       1.30 Params::Validate
ok       2.01 Pod::Usage
ok       2.05 Readonly
ok 2017060201 Regexp::Common
ok       2.86 SVG
ok       1.19 Set::IntSpan
ok     1.6611 Statistics::Basic
ok       3.23 Storable
ok       1.23 Sys::Hostname
ok       2.04 Text::Balanced
ok       0.62 Text::Format
ok     1.9767 Time::HiRes
```

Step 8 Install the modules that are missing one by one. The following describes how to install Carp.

```
cpan install Carp
```

----End

4.57.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform a test case.

```
cd /path/to /circos/circos-0.69-9/example/
rm -rf *.svg *.png
../bin/circos  -conf  ./etc/circos.conf
```

On Windows, view the **.png** and **.svg** files generated in the current directory.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.58 cisTEM 1.0 Porting Guide (CentOS 7.6)

4.58.1 Introduction

cisTEM is user-friendly software to process cryo-EM images of macromolecular complexes and obtain high-resolution 3D reconstructions from them. It was originally developed by Tim Grant, Alexis Rohou and Nikolaus Grigorieff and comprises a number of tools to process image data including movies, micrographs and stacks of single-particle images, implementing a complete "pipeline" of processing steps to obtain high-resolution single-particle reconstructions.

cisTEM is distributed under the Janelia Research Campus Software License and can be downloaded here. We recommend downloading and using the pre-compiled binaries, rather than compiling the source code, for best performance. New users are encouraged to follow the tutorial, which provides a quick way to become familiar with the most important functions of cisTEM.

For more information about cisTEM, visit the **cisTEM official website**.

Programming language: C/C++

Brief description: cisTEM processes cryo-EM images of macromolecular complexes and obtain high-resolution 3D reconstructions from them.

Open source license: GPL to Janelia

Recommended Software Version

cisTEM 1.0

4.58.2 Environment Requirements

Hardware Requirements

Table 4-283 lists the hardware requirements.

Table 4-283 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-284 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-284 Software requirements

Item	Version	Download Address
cisTEM	1.0.0-beta	**https://cistem.org/**
GCC	7.3.0	**https://codeload.github.com/gcc-mirror/gcc/tar.gz/refs/tags/releases/gcc-7.3.0**
GMP	6.1.0	**http://gcc.gnu.org/pub/gcc/infrastructure/**
MPC	1.0.3	**http://gcc.gnu.org/pub/gcc/infrastructure/**
MPFR	3.1.4	**http://gcc.gnu.org/pub/gcc/infrastructure/**
FFTW	3.3.8	**http://www.fftw.org/fftw-3.3.8.tar.gz**
wxWidgets	3.0.2	**https://github.com/wxWidgets/wxWidgets/releases/download/v3.0.2/wxWidgets-3.0.2.tar.bz2**

OS Requirements

Table 4-285 lists the OS requirements.

Table 4-285 OS requirements

Item	Version	Download Address
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115.el7a.0.1	**https://www.centos.org/download/**

4.58.3 Paths for Software Porting

This chapter lists the software installation paths involved in the cisTEM software porting.

Table 4-286 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/FFTW	Installation path of FFTW.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/WXWIDGETS	Installation path of wxWidgets.	used in commands in this document with the actual paths planned during the installation process.
4	/path/to/CISTEM	Installation path of RELION.	

4.58.4 Configuring the Compilation Environment

4.58.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-287 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing FFTW	See **4.58.4.2 Installing FFTW**.
3	Installing wxWidgets	See **4.58.4.3 Installing wxWidgets**.

4.58.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

 Copyright © Huawei Technologies Co., Ltd.

```
./configure --prefix=/path/to/FFTW--enable-shared --enable-static --enable-fma --enable-neon --
enable-float
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

----End

4.58.4.3 Installing wxWidgets

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install wxWidgets.

```
yum install libtiff-devel gtk2-devel*
wget https://github.com/wxWidgets/wxWidgets/releases/download/v3.0.2/wxWidgets-3.0.2.tar.bz2
tar xvf wxWidgets-3.0.2.tar.bz2
cd wxWidgets-3.0.2
mkdir gtk_config
cd gtk_config
../configure --prefix=/path/to/WXWIDGETS --build=aarch64-unknown-linux-gnu --disable-share --
enable-unicode --with-libtiff=sys --with-gtk=2
make -j16
```

Step 3 Configure the environment variables.

```
export PATH=/path/to/WXWIDGETS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/WXWIDGETS/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/WXWIDGETS/lib:$ INCLUDE
```

Step 4 Perform the verification.

```
wx-config  --version
```

If the correct wxWidgets version is displayed, the installation is successful.

----End

4.58.5 Obtaining the Source Code

Procedure

Step 1 Download the cisTEM installation package.

https://cistem.org/

Step 2 Use an SFTP tool to upload the cisTEM installation package to the */path/to/CISTEM* directory on the server.

----End

4.58.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xvf cistem-1.0.0-beta-source-code.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the directory generated after the decompression.

```
cd cistem-1.0.0-beta
```

Step 4 Modify the source code.

1. Open the source code file.

   ```
   vi src/core/matrix.cpp
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   Change line 7

   ```
   #define _AL_SINCOS(x, s, c)  __asm__ ("fsincos" : "=t" (c), "=u" (s) : "0" (x))
   ```

   to:

   ```
   #define _AL_SINCOS(x, s, c)   s = sinf(x);  c = cosf(x);
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the source code.

1. Open the source code file.

   ```
   vi src/gui/AssetPickerComboPanel.h
   ```

2. Press **i** to enter the insert mode and modify the file as follows:

   Change line 49

   ```
   bool FillComboBox(long wanted_refinement_package, bool always_select_latest = false)
   {AssetComboBox->FillWithRefinements(wanted_refinement_package, always_select_latest);}
   ```

   to:

   ```
   bool FillComboBox(long wanted_refinement_package, bool always_select_latest = false) {return
   AssetComboBox->FillWithRefinements(wanted_refinement_package, always_select_latest);}
   ```

   Change line 65

   ```
   bool FillComboBox(bool include_all_images_group) {AssetComboBox-
   >FillWithImageGroups(include_all_images_group);}
   ```

   to:

   ```
   bool FillComboBox(bool include_all_images_group) {return AssetComboBox-
   >FillWithImageGroups(include_all_images_group);}
   ```

   Change line 73

   ```
   bool FillComboBox(bool include_all_movies_group) {AssetComboBox-
   >FillWithMovieGroups(include_all_movies_group);}
   ```

   to:

   ```
   bool FillComboBox(bool include_all_movies_group) {return AssetComboBox-
   >FillWithMovieGroups(include_all_movies_group);}
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Modify the source code.

1. Open the source code file.

   ```
   vi src/programs/projectx/projectx.cpp
   ```

2. Press **i** to enter the insert mode and add the following content:

   Change line 297

   ```
   return true;
   ```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the source code.

```
sed -i 's/\-03/\-00/' configure
```

Step 8 Perform the compilation and installation.

```
yum install libtiff xorg-x11-*
./configure --prefix=/path/to/CISTEM --with-fftw-dir=/path/to/FFTW
```

 Copyright © Huawei Technologies Co., Ltd.

```
make -j
make install
```

----End

4.58.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/CISTEM/bin
```

Step 3 Run **CISTEM**.

```
./cisTEM
```

If the following information is displayed, the running is successful.

----End

4.59 EMAN2 2.91 Porting Guide (CentOS 8.2)

4.59.1 Introduction

EMAN2 is a broadly based greyscale scientific image processing suite with a primary focus on processing data from transmission electron microscopes. EMAN2 can process large datasets (> 100,000 particles) efficiently.

For more information about EMAN2, visit the **EMAN2 official website**.

Programming language: Python

Brief description: EMAN2 is a broadly based greyscale scientific image processing suite with a primary focus on processing data from transmission electron microscopes.

 Copyright © Huawei Technologies Co., Ltd.

Open source licenses: GPL2 and BSD 3-Clause

Recommended Software Version

EMAN2 2.91

4.59.2 Environment Requirements

Hardware Requirements

Table 4-288 lists the hardware requirements.

Table 4-288 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-289 lists the software requirements.

Table 4-289 Software requirements

Item	Version	Download Address
EMAN2	2.91	https://github.com/cryoem/eman2.git
Anaconda	2021.05	https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh

OS Requirements

Table 4-290 lists the OS requirements.

Table 4-290 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.59.3 Paths for Software Porting

This chapter lists the software installation paths involved in the EMAN2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-291 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	Setting up the basic environment	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/root/anaconda3	Installation path of Anaconda.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/EMAN2	Installation path of EMAN2.	

4.59.4 Configuring the Compilation Environment

Configure Process

Table 4-292 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Anaconda	See **4.59.4.1 Installing Anaconda**.
3	Porting bsddb3	See **4.59.4.2 Porting bsddb3**.
4	Porting tensorflow2.4.1	See **4.59.4.3 Porting TensorFlow 2.4.1**.

4.59.4.1 Installing Anaconda

Procedure

Step 1 Obtain the Anaconda installation package.

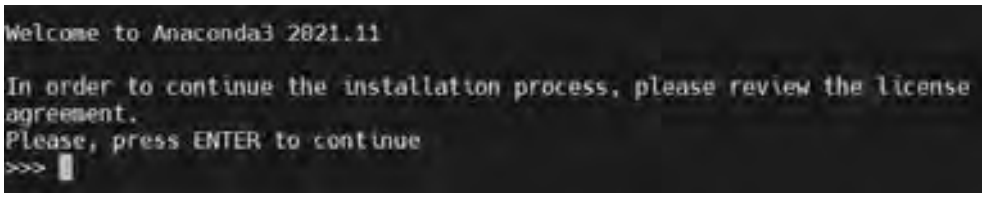
```
wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-aarch64.sh
```

Step 2 Install Anaconda.

```
bash Anaconda3-2021.11-Linux-aarch64.sh
```

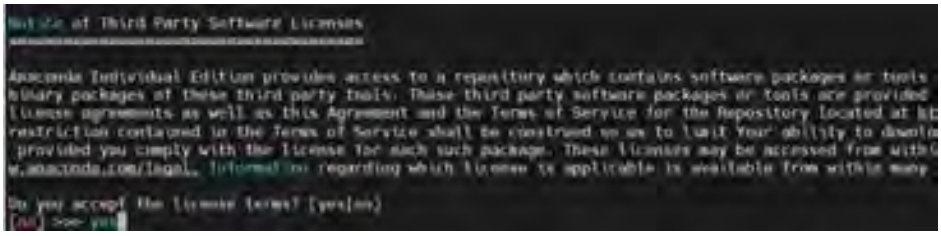
1. When the following information is displayed, press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

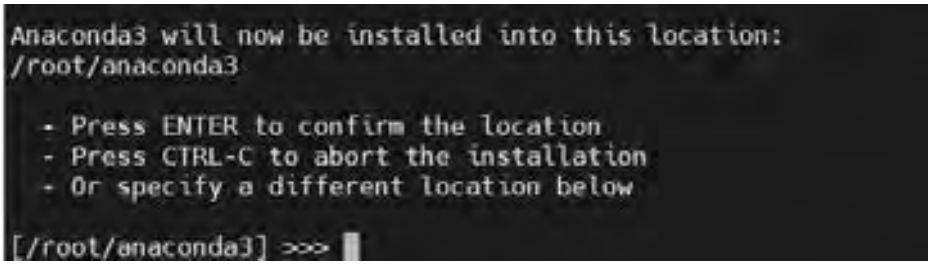


2. When the following information is displayed, type **q**.

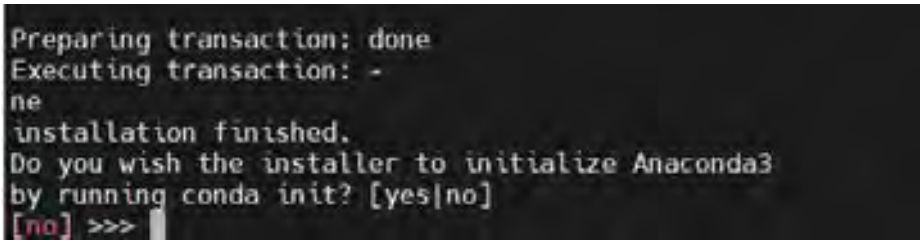
3. When the following information is displayed, type **yes** and press **Enter**.



4. On the screen shown in the following figure, type a desired Anaconda installation path. You can also press **Enter** to select the default installation path **/root/anaconda3**.



5. Press **Enter**. No initialization operations will be performed.



Step 3 Set the conda environment variables.

1. Set the environment variables.
   source */root/anaconda3*/**etc/profile.d/conda.sh**
2. Check whether conda has taken effect.
   conda -version

   ```
   [root@2280node27-test yum.repos.d]#  conda --version
   conda 4.11.0
   ```

3. Initialize conda.
   conda init bash
4. Restart the shell. You can see that conda has entered the base environment.

   ```
   (base) [root@localhost ~]#
   ```

Step 4 Configure and update conda.

```
conda config --set auto_activate_base False
conda config --set auto_update_conda False
conda config --set channel_priority flexible
conda update --all -n base
```

If the update is required, type **y** and press **Enter**.

----End

4.59.4.2 Porting bsddb3

Procedure

Step 1 Create a bsddb3 build environment.

```
conda create -n conda_build conda-build python=3.7.12 -c conda-forge
```

Step 2 Activate the build environment.

```
conda activate conda_build
```

Step 3 Upload the conda file.

1. Create a **bsddb3** directory.
   mkdir bsddb3
2. Use an SFTP tool to upload the **meta.yaml** and **build.sh** files to the **bsddb3** directory.
   - The content of **meta.yaml** is as follows:

     ```
     package:
     name: bsddb3
     version: 6.2.9
     source:
     git_url: https://github.com/wchangque/bsddb3.git
     build:
     number: '1'
     string: py37_0
     requirements:
     build:
     ```

 Copyright © Huawei Technologies Co., Ltd.

```
- _libgcc_mutex 0.1
- _openmp_mutex 4.5
- binutils_impl_linux-aarch64
- binutils_linux-aarch64 2.36
- gcc_impl_linux-aarch64 9.4.0
- gcc_linux-aarch64 9.4.0
- kernel-headers_linux-aarch64 4.18.0
- ld_impl_linux-aarch64 2.36.1
- libgcc-devel_linux-aarch64 9.4.0
- libgcc-ng 11.2.0
- libgomp 11.2.0
- libsanitizer 9.4.0
- libstdcxx-ng 11.2.0
- sysroot_linux-aarch64 2.17
host:
- _libgcc_mutex 0.1
- _openmp_mutex 4.5
- ca-certificates 2021.10.8
- ld_impl_linux-aarch64 2.36.1
- libdb 6.2.32
- libffi 3.4.2
- libgcc-ng 11.2.0
- libgomp 11.2.0
- libnsl 2.0.0
- libstdcxx-ng 11.2.0
- libzlib 1.2.11
- ncurses 6.2
- openssl 3.0.0
- pip 21.3.1
- python 3.7.12
- python_abi 3.7
- readline 8.1
- setuptools 58.5.3
- sqlite 3.36.0
- tk 8.6.11
- wheel 0.37.0
- xz 5.2.5
- zlib 1.2.11
run:
- libdb >=6.2.32,<6.3.0a0
- libgcc-ng >=9.4.0
- python >=3.7,<3.8.0a0
- python_abi 3.7.* *_cp37m
test:
imports:
- bsddb3
about:
description: 'This module provides a nearly complete wrapping of the Oracle/Sleepycat
C API for the Database Environment,
Database, Cursor, Log Cursor, Sequence and Transaction objects,
and each of these is exposed as a Python type in the bsddb3.db module.
The database objects can use various access methods: btree, hash, recno, and queue.
Complete support of Berkeley DB distributed transactions.
Complete support for Berkeley DB Replication Manager.
Complete support for Berkeley DB Base Replication. Support for RPC.
'
doc_url: https://www.jcea.es/programacion/pybsddb_doc/
home: https://www.jcea.es/programacion/pybsddb.htm
license: BSD-3-Clause
license_family: BSD
license_file: LICENSE.txt
summary: Python bindings for Oracle Berkeley DB
extra:
copy_test_source_files: true
final: true
recipe-maintainers:
- shadowwalkersb
```

- The content of **build.sh** is as follows:

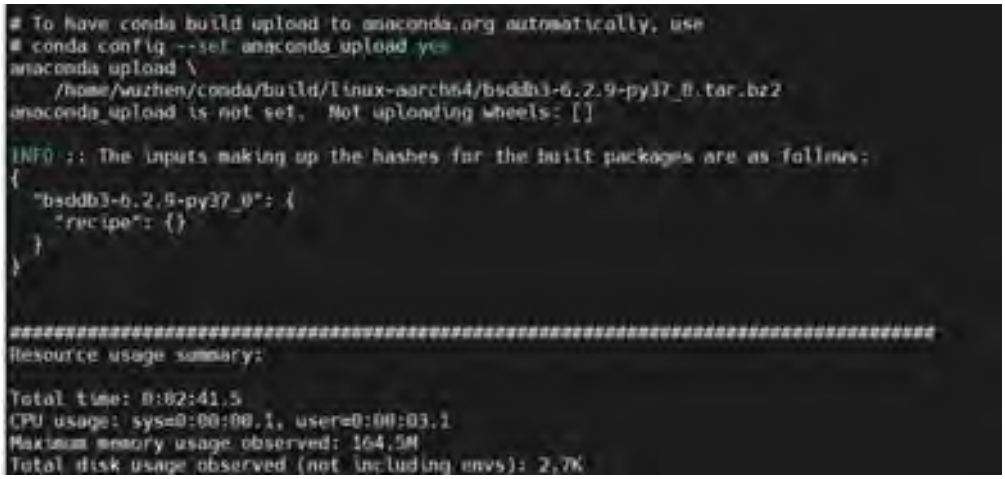
```
#!/usr/bin/env bash
export YES_I_HAVE_THE_RIGHT_TO_USE_THIS_BERKELEY_DB_VERSION=1
BERKELEYDB_DIR=$PREFIX $PYTHON -m pip install . --no-deps --ignore-installed
```

Step 4 Build the local conda package of bsddb3.

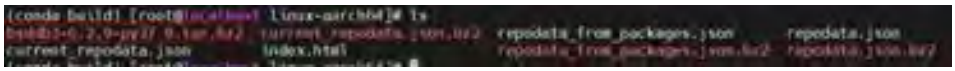
Create a build directory in the same directory as bsddb3.

```
mkdir build
export CONDA_BLD_PATH=build
conda build -c conda-forge bsddb3/
```

The build is successful if the following information is displayed:



The required package is stored in the **build/linux-aarch64/** directory.



----End

4.59.4.3 Porting TensorFlow 2.4.1

Procedure

Step 1 Create a TensorFlow build environment.

```
conda create -n bazel_build bazel=3.7 python=3.7.12  -c conda-forge
```

Step 2 Activate the build environment.

```
conda activate bazel_build
```

Step 3 Install the build dependencies.

```
conda install -c conda-forge keras-preprocessing=1.1.2 numpy=1.18.5
```

Step 4 Download the TensorFlow 2.4.1 source code.

```
git clone https://github.com/tensorflow/tensorflow.git -b v2.4.1
```

Step 5 Go to the source code directory.

```
cd tensoflow
```

Step 6 Configure the build environment.

```
python configure.py
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Compile the source code.

```
bazel build //tensorflow/tools/pip_package:build_pip_package
```

Due to network problems, many resources cannot be automatically obtained during bazel build. You need to download the required resources and change the URLs in the build file to the local paths. For details, see **4.59.7.1 Many Resources Fail to Be Automatically Obtained for Bazel Build When Porting TensorFlow 2.4.1**.

Step 8 Build the local conda package of TensorFlow.

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package /path/to/tensorflow
```

The built TensorFlow installation package is stored in the **/path/to/tensorflow** directory.

```
tensorflow-2.4.1-cp37-cp37m-linux_aarch64.whl
```

----End

4.59.5 Compilation and Installation

Procedure

Step 1 Install the graphics library.

```
yum install -y mesa-libGL mesa-libGL-devel mesa-libGLU mesa-libGLU-devel
```

Step 2 Create an EMAN2 compilation and running environment.

```
conda create -n eman2
```

Step 3 Activate the EMAN2 environment.

```
conda activate eman2
```

Step 4 Configure the EMAN2 environment.

```
conda install -c conda-forge _libgcc_mutex=0.1 _openmp_mutex=4.5 absl-py=1.0.0 aiohttp=3.8.1 aiosignal=1.2.0 alsa-lib=1.2.3 anyio=3.4.0 argcomplete=1.12.3 argon2-cffi=21.1.0 astor=0.8.1 astunparse=1.6.3 async-timeout=4.0.1 asynctest=0.13.0 attrs=21.2.0 babel=2.9.1 backcall=0.2.0 backports=1 backports.functools_lru_cache=1.6.4 bleach=4.1.0 blinker=1.4 boost=1.74.0 boost-cpp=1.74.0 brotli=1.0.9 brotli-bin=1.0.9 brotlipy=0.7.0 bzip2=1.0.8 c-ares=1.18.1 ca-certificates=2021.10.8 cachetools=4.2.4 certifi=2021.10.8 cffi=1.15.0 charset-normalizer=2.0.8 click=8.0.3 cmake=3.21.3 colorama=0.4.4 configparser=5.1.0 cryptography=35.0.0 cycler=0.11.0 dataclasses=0.8 dbus=1.13.6 debugpy=1.5.1 decorator=5.1.0 defusedxml=0.7.1 entrypoints=0.3
```

 Copyright © Huawei Technologies Co., Ltd.

```
expat=2.4.1 fftw=3.3.10 fontconfig=2.13.1 fonttools=4.28.2 freetype=2.10.4 frozenlist=1.2.0
future=0.18.2 gast=0.4.0 gettext=0.19.8.1 glib=2.70.1 glib-tools=2.70.1 google-auth=2.3.3 google-auth-
oauthlib=0.4.6 google-pasta=0.2.0 grpcio=1.42.0 gsl=2.6 gst-plugins-base=1.18.5 gstreamer=1.18.5
h5py=2.10.0 hdf5=1.10.6 icu=68.2 idna=3.1 importlib-metadata=4.8.2 importlib_metadata=4.8.2
importlib_resources=5.4.0 ipykernel=6.5.1  ipython_genutils=0.2.0 jbig=2.1 jedi=0.18.1 jinja2=3.0.3
joblib=1.1.0 jpeg=9d json5=0.9.5 jsonschema=4.2.1 jupyter_client=7.1.0 jupyter_core=4.9.1
jupyter_server=1.12.1 jupyterlab=3.2.4 jupyterlab_pygments=0.1.2 jupyterlab_server=2.8.2 keras-
preprocessing=1.1.2 kiwisolver=1.3.2 krb5=1.19.2 lcms2=2.12 ld_impl_linux-64=2.36.1 lerc=3
libblas=3.9.0 libbrotlicommon=1.0.9 libbrotlidec=1.0.9 libbrotlienc=1.0.9 libcblas=3.9.0 libclang=11.1.0
libcurl=7.80.0 libdb=6.2.32 libdeflate=1.8 libedit=3.1.20191231 libev=4.33 libevent=2.1.10 libffi=3.4.2
libgcc-ng=11.2.0 libgfortran-ng=11.2.0 libgfortran5=11.2.0 libglib=2.70.1 libgomp=11.2.0 libiconv=1.16
liblapack=3.9.0 libllvm11=11.1.0 libnghttp2=1.43.0 libnsl=2.0.0 libogg=1.3.4 libopenblas=0.3.18
libopus=1.3.1 libpng=1.6.37 libpq=13.5 libprotobuf=3.19.1 libsodium=1.0.18 libssh2=1.10.0 libstdcxx-
ng=11.2.0 libtiff=4.3.0 libuuid=2.32.1 libuv=1.42.0 libvorbis=1.3.7 libwebp-base=1.2.1 libxcb=1.13
libxkbcommon=1.0.3 libxml2=2.9.12 libzlib=1.2.11 lz4-c=1.9.3 markdown=3.3.6 markupsafe=2.0.1
matplotlib=3.5.0 matplotlib-base=3.5.0 matplotlib-inline=0.1.3 mistune=0.8.4 mpi=1 multidict=5.2.0
munkres=1.1.4 mysql-common=8.0.27 mysql-libs=8.0.27 nbclassic=0.3.4 nbclient=0.5.9 nbconvert=6.3.0
nbformat=5.1.3 ncurses=6.2 nest-asyncio=1.5.1 nodejs=16.12.0 nose=1.3.7 notebook=6.4.6 nspr=4.32
nss=3.72 numpy=1.18.5 oauthlib=3.1.1 olefile=0.46 openjpeg=2.4.0 openmpi=4.1.2 openssl=1.1.1l
opt_einsum=3.3.0 packaging=21.3 pandas=1.3.4  pandocfilters=1.5.0 parso=0.8.2 pcre=8.45
pexpect=4.8.0 pickleshare=0.7.5 pillow=8.4.0 pip=21.3.1 prometheus_client=0.12.0 prompt-
toolkit=3.0.22 protobuf=3.19.1 pthread-stubs=0.4 ptyprocess=0.7.0 pyasn1=0.4.8 pyasn1-modules=0.2.7
pycparser=2.21 pygments=2.10.0 pyjwt=2.3.0 pyopengl=3.1.5 pyopenssl=21.0.0 pyparsing=3.0.6
pyqt=5.12.3 pyqt-impl=5.12.3 pyqt5-sip=4.19.18 pyqtchart=5.12 pyqtwebengine=5.12.1
pyrsistent=0.18.0 pysocks=1.7.1 python=3.7.12 python-dateutil=2.8.2 python-flatbuffers=2
python_abi=3.7 pytz=2021.3 pyu2f=0.1.5 pyzmq=22.3.0 qt=5.12.9 readline=8.1 requests=2.26.0 requests-
oauthlib=1.3.0 rhash=1.4.1 rsa=4.8 scikit-learn=1.0.1 scipy=1.7.3 send2trash=1.8.0 setuptools=59.4.0
six=1.16.0 sniffio=1.2.0 sqlite=3.37.0 tensorboard=2.7.0 tensorboard-data-server=0.6.0 tensorboard-
plugin-wit=1.8.0 tensorflow-estimator=2.5.0 termcolor=1.1.0 terminado=0.12.1 testpath=0.5.0
threadpoolctl=3.0.0 tk=8.6.11 tornado=6.1 tqdm=4.62.3 traitlets=5.1.1 typing-extensions=4.0.0
typing_extensions=4.0.0 unicodedata2=13.0.0.post2 urllib3=1.26.7 wcwidth=0.2.5 webencodings=0.5.1
websocket-client=1.2.1 werkzeug=2.0.1 wheel=0.37.0 wrapt=1.13.3 xorg-libxau=1.0.9 xorg-
libxdmcp=1.1.3 xz=5.2.5 yarl=1.7.2 zeromq=4.3.4 zipp=3.6.0 zlib=1.2.11 zstd=1.5.0 ftgl pandoc
ipython=7.30
```

Step 5 Install bsddb3.

1. Copy the bsddb3 package generated in **4.59.4.2 Porting bsddb3** to the Anaconda package directory.
   ```
   cp bsddb3-6.2.9-py37_0.tar.bz2 /root/anaconda3/pkgs/
   ```
2. Install bsddb3.
   ```
   conda install --use-local /root/anaconda3/pkgs/bsddb3-6.2.9-py37_0.tar.bz2
   ```

Step 6 Install tensorlfow2.4.1.

1. Copy the TensorFlow package generated in **4.59.4.3 Porting TensorFlow 2.4.1** to the Anaconda package directory.
   ```
   cp tensorflow-2.4.1-cp37-cp37m-linux_aarch64.whl /root/anaconda3/pkgs/
   ```
2. Install tensorlfow2.4.1.
   ```
   pip install /root/anaconda3/pkgs/tensorflow-2.4.1-cp37-cp37m-linux_aarch64.whl --no-deps
   ```

Step 7 Download the EMAN2 source code.

```
git config --global http.sslVerify false
git clone https://github.com/cryoem/eman2.git
cd eman2
git fetch --all --prune
git checkout master
git pull
```

NOTICE

The EMAN2 source code must be obtained by running the **git clone** command. The released source code package has bugs.

 Copyright © Huawei Technologies Co., Ltd.

Step 8 Modify the **CmakeLists.txt** file.

```
sed -i "s/-m64/-mabi=lp64/g" CMakeLists.txt
```

Step 9 Compile EMAN2.

```
mkdir build && cd build
cmake .. -DENABLE_OPTIMIZE_MACHINE=ON
make -j`cat /proc/cpuinfo| grep "processor"| wc -l`
make install
```

----End

4.59.6 Running and Verification

Procedure

Step 1 Activate the EMAN2 environment.

```
conda activate eman2
```

Step 2 Run **e2version.py**.

```
/root/anaconda3/envs/eman2/bin/e2version.py
```

Information similar to the following is displayed:

Step 3 Run **e2speed.py**.

```
/root/anaconda3/envs/eman2/bin/e2speedtest.py
```

Information similar to the following is displayed:

Step 4 Run **e2display.py**.

```
/root/anaconda3/envs/eman2/bin/e2display.py
```

Information similar to the following is displayed:

NOTICE

The X11 must be installed on the server and forwarded through SSH to display images. Alternatively, log in to the server GUI and run the **e2display.py** script.

----End

4.59.7 Troubleshooting

4.59.7.1 Many Resources Fail to Be Automatically Obtained for Bazel Build When Porting TensorFlow 2.4.1

Symptom

Due to network problems, many resources cannot be automatically obtained during bazel build. You need to download the required resources and change the URLs in the build file to the local paths.

Key Process and Cause Analysis

A network exception occurs.

Conclusion and Solution

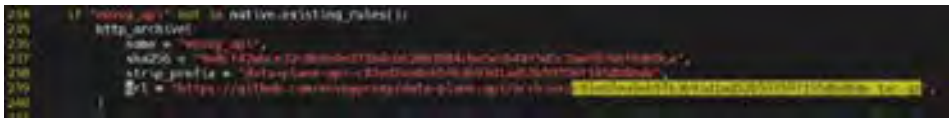
Step 1 Run the following command in the build directory and **/root/.cache/bazel/_bazel_root** directory:

```
grep -rn "c83ed7ea9eb5fb3b93d1ad52b59750f1958b8bde.tar.gz"
```

The following information is displayed:

Step 2 Find **url = ...** and open the file to the corresponding line (line 239 in this example):

vim *b9ee4e17aa65b8928460ddaabd4265de/external/com_github_grpc_grpc/bazel/grpc_deps.bzl*



Step 3 Create a local source path.

mkdir -p */path/to/tenflor_requires* && **cd** */path/to/tenflor_requires*

Step 4 Download the required dependencies based on the displayed URL.

wget https://github.com/envoyproxy/data-plane-api/archive/c83ed7ea9eb5fb3b93d1ad52b59750f1958b8bde.tar.gz --no-check-certificate

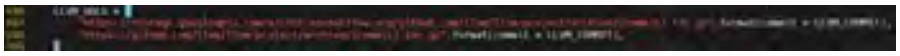
Step 5 Modify the preceding URL as follows:

```
name = "envoy_api",
sha256 = "9e8cf42abce32c9b0e9e271b0cb62803084cbe5e5b49f5d5c2aef0766f9d69ca",
strip_prefix = "data-plane-api-c83ed7ea9eb5fb3b93d1ad52b59750f1958b8bde",
#url = "https://github.com/envoyproxy/data-plane-api/archive/
c83ed7ea9eb5fb3b93d1ad52b59750f1958b8bde.tar.gz",
url="file:///path/to/tenflor_requires/c83ed7ea9eb5fb3b93d1ad52b59750f1958b8bde.tar.gz",
```

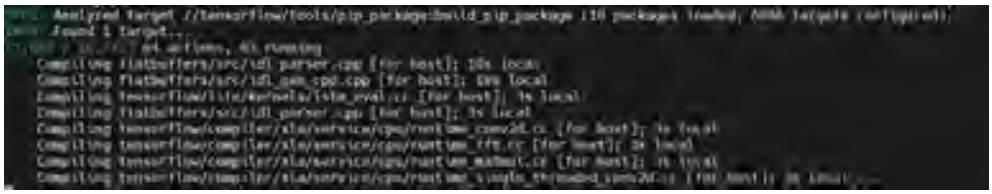
Step 6 Continue to run the build command. This problem may occur multiple times during subsequent builds. You can use the preceding method to rectify the fault.

NOTE

A variable may be used for the packet name. If the URL is not matched, delete the package name and run the **grep** command to match the URL.



If the following information is displayed, all dependencies have been obtained and the compilation is in progress.



----End

4.59.8 More Resources

TensorFlow source code:

https://github.com/tensorflow/tensorflow/tree/master

Building the TensorFlow source code:

https://www.tensorflow.org/install/source

Anaconda official library:

https://anaconda.org/

4.60 EMBOSS 6.5.7 Porting Guide (CentOS 7.6)

 Copyright © Huawei Technologies Co., Ltd.

4.60.1 Introduction

EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community. The software automatically copes with data in a variety of formats and even allows transparent retrieval of sequence data from the web. Also, as extensive libraries are provided with the package, it is a platform to allow other scientists to develop and release software in true open source spirit. EMBOSS also integrates a range of currently available packages and tools for sequence analysis into a seamless whole. EMBOSS breaks the historical trend towards commercial software packages.

For more information about EMBOSS, visit the **EMBOSS official website**.

Programming languages: C/C++

Brief description: EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community.

Open source licenses: GPL, LGPL

Recommended Software Version

EMBOSS v6.5.7

4.60.2 Environment Requirements

Hardware Requirements

Table 4-293 lists the hardware requirements.

Table 4-293 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-294 lists the software requirements.

Table 4-294 Software requirements

Item	Version	Download Address
EMBOSS	6.5.7	**ftp://emboss.open-bio.org/pub/EMBOSS/old/6.5.0/EMBOSS-6.5.7.tar.gz**
CMake	3.18.2	They are installed using Yum and do not need to be manually downloaded.
GCC/GCC-C++	8.5.0	

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
libX11, libX11-devel	1.6.8	

OS Requirements

Table 4-295 lists the OS requirements.

Table 4-295 OS requirements

Item	Version	Download Address
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.18.0-193.el8.aarch64	**https://www.centos.org/download/**

4.60.3 Paths for Software Porting

This chapter lists the software installation paths involved in the EMBOSS software porting.

Table 4-296 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/opt/EMBOSS	Installation path of EMBOSS.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

4.60.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configure Process

Table 4-297 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the dependencies	See **4.60.4.1 Installing the Dependencies**.

4.60.4.1 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y epel-release
yum install -y gcc
yum install -y gcc-c++
yum install -y cmake
yum install -y make
yum install -y libX11
yum install -y libX11-devel
```

Step 3 Perform the verification.

```
gcc --version
cmake --version
make --version
```

If the correct version is displayed, the installation is successful.

----End

4.60.5 Obtaining the Source Code

Procedure

Step 1 Download the source code package.

ftp://emboss.open-bio.org/pub/EMBOSS/old/6.5.0/EMBOSS-6.5.7.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */opt/EMBOSS* directory on the server.

----End

4.60.6 Compilation and Installation

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

 Copyright © Huawei Technologies Co., Ltd.

```
tar -zxvf EMBOSS-6.5.7.tar.gz
```

Step 3 Perform compilation and installation.

```
cd EMBOSS-6.5.7/
./configure --prefix=/opt/EMBOSS --build=aarch64-unknown-linux-gnu
make -j 8
make install
```

Step 4 Verify the result.

```
cd /opt/EMBOSS/bin
./embossversion
```

```
(eman2) [root@localhost bin]# ./embossversion
Report the current EMBOSS version number
6.5.7.0
(eman2) [root@localhost bin]#
```

----End

4.60.7 More Resources

EMBOSS installation guide:

https://blog.csdn.net/Cassiel60/article/details/88218706

4.61 HumanN 3.0.1 Porting Guide (CentOS 8.2)

4.61.1 Introduction

HMP Unified Metabolic Analysis Network (HUMAnN) is a method for efficiently and accurately profiling the abundance of microbial metabolic pathways and other molecular functions from metagenomic or metatranscriptomic sequencing data.

For more information, visit the **official HUMAnN website**.

Programming language: Python

Brief description: HUMAnN is a method for efficiently and accurately profiling the abundance of microbial metabolic pathways and other molecular functions from metagenomic or metatranscriptomic sequencing data.

Open source license: custom open source license

Recommended Software Version

HUMAnN 3.0.1

4.61.2 Environment Requirements

Hardware Requirements

Table 4-298 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-298 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-299 lists the software requirements.

Table 4-299 Software requirements

Item	Version	Download Address
Python	3.9.6	https://www.python.org/downloads/release/python-396/
MetaPhlAn	3.0.13	https://mirrors.aliyun.com/pypi/simple
diamond	2.0.4	https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz

OS Requirements

Table 4-300 lists the OS requirements.

Table 4-300 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.61.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HUMAnN software porting.

Table 4-301 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/python-3.9.6	Installation path of Python.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/Metaphlan	Installation path of MetaPhlAn.	

4.61.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-302 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing MetaPhlAn	See **MetaPhlAn 3.0.13 Porting Guide (CentOS 8.2)**.
3	Installing diamond	See **Diamond 2.0.4 Porting Guide (CentOS 7.6)**.

4.61.5 Compilation and Installation

4.61.5.1 Installing HUMAnN

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install HUMAnN.

```
pip3 install humann -i https://mirrors.aliyun.com/pypi/simple --trusted-host mirrors.aliyun.com
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.61.5.2 Installing Bowtie 2

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Delete the Bowtie 2 preinstalled by HUMAnN:

```
rm -rf /path/to/python-3.9.6/bin/bowtie2*
```

Step 3 Download the Bowtie 2 installation package.

http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.4.2/bowtie2-2.4.2-linux-aarch64.zip

Step 4 Decompress the installation package.

```
unzip bowtie2-2.4.2-linux-aarch64.zip
```

Step 5 Edit the configuration file.

1. Open the configuration file.
   ```
   vi /etc/profile
   ```
2. Press **i** to enter the insert mode and add the Bowtie 2 path.
   ```
   export PATH=/home/gcc-9.3.1-2020.12-aarch64-linux/bin:/usr/local/python-3.9.6/bin:/home/bowtie2-2.4.2-linux-aarch64:$PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Make the configuration take effect.

```
source /etc/profile
```

Step 7 Check whether Bowtie 2 is successfully installed.

```
bowtie2 -h
```

The installation is successful if the following information is displayed:

```
[root@ceph3 bowtie2-2.4.2-linux-aarch64]# bowtie2 -h
Bowtie 2 version 2.4.2 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)
Usage:
  bowtie2 [options]* -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r> | --interleaved <i> | -b <bam>} [-S <
sam>]

  <bt2-idx>  Index filename prefix (minus trailing .X.bt2).
             NOTE: Bowtie 1 and Bowtie 2 indexes are not compatible.
  <m1>       Files with #1 mates, paired with files in <m2>.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <m2>       Files with #2 mates, paired with files in <m1>.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <r>        Files with unpaired reads.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <i>        Files with interleaved paired-end FASTQ/FASTA reads
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <bam>      Files are unaligned BAM sorted by read name.
  <sam>      File for SAM output (default: stdout)

  <m1>, <m2>, <r> can be comma-separated lists (no whitespace) and can be
  specified many times.  E.g. '-U file1.fq,file2.fq -U file3.fq'.

Options (defaults in parentheses):

 Input:
  -q                 query input files are FASTQ .fq/.fastq (default)
  --tab5             query input files are TAB5 .tab5
  --tab6             query input files are TAB6 .tab6
  --qseq             query input files are in Illumina's qseq format
  -f                 query input files are (multi-)FASTA .fa/.mfa
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.61.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Query the HUMAnN version.

```
humann --version
humann v3.0.1
```

Step 3 Run the test case.

```
humann --input /path/to/python-3.9.6/lib/python3.9/site-packages/humann/tests/data/demo.fastq --output /opt/humann/
```

```
Output files will be written to: /home/ycl/tese_output

Running metaphlan ........

Found g__Bacteroides.s__Bacteroides_dorei : 57.96% of mapped reads
Found g__Bacteroides.s__Bacteroides_vulgatus : 42.04% of mapped reads

Total species selected from prescreen: 2

Selected species explain 100.00% of predicted community composition


Creating custom ChocoPhlAn database ........


Running bowtie2-build ........


Running bowtie2 ........

Total bugs from nucleotide alignment: 2
g__Bacteroides.s__Bacteroides_vulgatus: 1195 hits
g__Bacteroides.s__Bacteroides_dorei: 1260 hits

Total gene families from nucleotide alignment: 545

Unaligned reads after nucleotide alignment: 88.3095238095 %


Running diamond ........


Aligning to reference database: uniref90_demo_prots_v201901b.dmnd

Total bugs after translated alignment: 3
g__Bacteroides.s__Bacteroides_vulgatus: 1195 hits
g__Bacteroides.s__Bacteroides_dorei: 1260 hits
unclassified: 1017 hits

Total gene families after translated alignment: 588

Unaligned reads after translated alignment: 83.6000000000 %


Computing gene families ...

Computing pathways abundance and coverage ...

Output files created:
/home/ycl/tese_output/demo_genefamilies.tsv
/home/ycl/tese_output/demo_pathabundance.tsv
/home/ycl/tese_output/demo_pathcoverage.tsv
```

Step 4 View the running result.

 Copyright © Huawei Technologies Co., Ltd.

----End

4.62 kallisto 0.46.2 Porting Guide (CentOS 8.2)

4.62.1 Introduction

kallisto is a program for quantifying abundances of transcripts from RNA-Seq data, or more generally of target sequences using high-throughput sequencing reads. It is based on the novel idea of pseudoalignment for rapidly determining the compatibility of reads with targets, without the need for alignment. On benchmarks with standard RNA-Seq data, kallisto can quantify 30 million human bulk RNA-seq reads in less than 3 minutes on a Mac desktop computer using only the read sequences and a transcriptome index that itself takes than 10 minutes to build. Pseudoalignment of reads preserves the key information needed for quantification, and kallisto is therefore not only fast, but also comparably accurate to other existing quantification tools. In fact, because the pseudoalignment procedure is robust to errors in the reads, in many benchmarks kallisto significantly outperforms existing tools.

For more information about kallisto, visit the **kallisto official website**.

Programming languages: C/C++

Brief description: kallisto is a program for quantifying abundances of transcripts from RNA-Seq data, or more generally of target sequences using high-throughput sequencing reads.

Open source license: GPLv3

Recommended Software Version

kallisto 0.46.2

4.62.2 Environment Requirements

Hardware Requirements

Table 4-303 lists the hardware requirements.

Table 4-303 Hardware

Item	Remarks
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-304 lists the software requirements.

Table 4-304 Software requirements

Item	Version	Download Address
kallisto	0.46.2	https://github.com/pachterlab/kallisto
CMake	3.20.2	They are installed using Yum and do not need to be manually downloaded.
GCC	8.5.0	
Git	/	
HTSlib	/	https://github.com/samtools/htslib

OS Requirements

Table 4-305 lists the OS requirements.

Table 4-305 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.62.3 Paths for Software Porting

This chapter lists the software installation paths involved in the kallisto software porting.

Table 4-306 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/home/htslib	Installation path of HTSlib.	The installation paths provided here are only an example. Shared paths are recommended. All the

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/kallisto	Installation path of kallisto.	paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.62.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-307 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing GCC, CMake, and Git	See **4.62.4.1 Installing GCC, CMake, and Git**.
3	Installing HTSlib	See **4.62.4.2 Installing HTSlib**.

4.62.4.1 Installing GCC, CMake, and Git

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC, CMake, and Git.

```
yum install gcc-c++
yum install cmake
yum install git
```

Step 3 Perform the verification.

```
gcc -v
cmake --version
git --version
```

If the correct version is displayed, the installation is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.62.4.2 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install HTSlib.

```
yum install autoconf automake libtool
yum -y install xz-devel
cd /home
git clone https://github.com/samtools/htslib.git
cd htslib
autoreconf -i
./configure
make
make install
```

----End

4.62.5 Obtaining the Source Code

Procedure

Step 1 Download the kallisto installation package.

```
cd /usr/local/bin
git clone https://github.com/pachterlab/kallisto.git
```

----End

4.62.6 Installing kallisto

Procedure

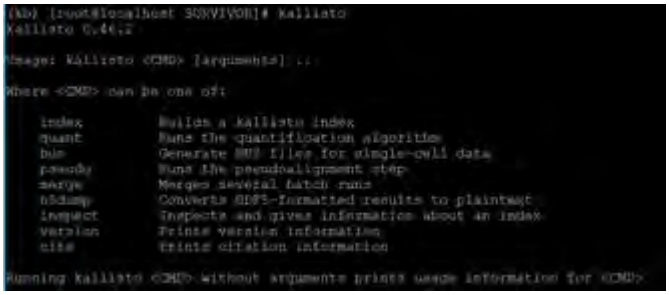
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install kallisto.

```
yum install libarchive
cd /path/to/kallisto
mkdir build
cd build
cmake ..
make
make install
```

Step 3 Perform the verification.

```
kallisto
```



----End

 Copyright © Huawei Technologies Co., Ltd.

4.63 kb-python 0.26.4 Porting Guide (CentOS 8.2)

4.63.1 Introduction

kb-python is a python package that wraps the kallisto | bustools single-cell RNA-seq workflow.

For more information about kb-python, visit the **kb-python official website**.

Programming language: Python

Brief description: kb-python is a python package that wraps the kallisto | bustools single-cell RNA-seq workflow.

Open source license: GPLv3

Recommended Software Version

kb-python 0.26.4

4.63.2 Environment Requirements

Hardware Requirements

Table 4-308 lists the hardware requirements.

Table 4-308 Hardware

Item	Remarks
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 4-309 lists the software requirements.

Table 4-309 Software requirements

Item	Version	Download Address
kb-python	0.26.4	**https://github.com/pachterlab/kb_python**
kallisto	0.46.2	**https://github.com/pachterlab/kallisto**
bustools	0.41.0	**https://github.com/BUStools/bustools**
CMake	3.20.2	They are installed using Yum and do not need to be manually downloaded.
GCC	8.5.0	

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
Git	/	
Python	3.8	https://www.python.org
Anaconda3	2021.05	https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh

OS Requirements

Table 4-310 lists the OS requirements.

Table 4-310 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.63.3 Paths for Software Porting

This chapter lists the software installation paths involved in the kb_python software porting.

Table 4-311 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/root/anaconda3	Installation path of Anaconda3.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/usr/local/bin/kallisto	Installation path of kallisto.	
4	/usr/local/bin/bustools	Installation path of bustools.	
5	/home/kb_python	Installation path of kb_python.	

 Copyright © Huawei Technologies Co., Ltd.

4.63.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-312 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing GCC, CMake, and Git	See **4.63.4.1 Installing GCC, CMake, and Git**.
3	Installing Anaconda3	See **4.63.4.2 Installing Anaconda3**.
4	Installing kallisto	See **kallisto 0.46.2 Porting Guide (CentOS 8.2)**
5	Installing Bustools	See **Bustools 0.41.0 Porting Guide (CentOS 8.2)**.

4.63.4.1 Installing GCC, CMake, and Git

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC, CMake, and Git.

```
yum install gcc-c++
yum install cmake
yum install git
```

Step 3 Perform the verification.

```
gcc -v
cmake --version
git --version
```

If the correct version is displayed, the installation is successful.

----End

4.63.4.2 Installing Anaconda3

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Anaconda installation package.

```
wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh
```

Step 3 Install Anaconda3.

```
bash Anaconda3-2021.05-Linux-aarch64.sh
```

1. Type **enter** to view the license.
2. Type **q** to continue.
3. Type **yes** to accept the license.
4. Type **enter** to confirm the default installation path **/root/anaconda3**.
5. Type **yes** to start the initialization.

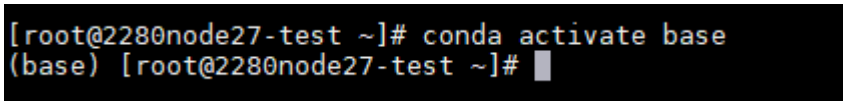
Step 4 Load the environment variables.

```
source ~/.bashrc
```

Step 5 Check whether Anaconda3 is successfully installed.

```
conda activate base
```

If the "base" information is displayed, Anaconda3 is successfully installed.



----End

4.63.5 Obtaining the Source Code

Procedure

Step 1 Download the kb_python installation package.

```
cd /home
git clone https://github.com/pachterlab/kb_python.git
```

----End

4.63.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create and enable the conda environment for kb_python.

```
conda activate base
conda create -n kb
conda activate kb
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels conda-forge
conda config --set show_channel_urls yes
conda update -n base -c defaults conda
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/ is optional. If the network speed is slow when conda is used to install the software, you can run this command to add a conda source from China.

Step 3 Install the Python environment.

```
conda install python=3.8
pip install Cython --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Step 4 Install PYSAM.

```
cd /home
git clone https://github.com/pysam-developers/pysam.git
cd pysam
yum install zlib-devel
yum install bzip2-devel
python setup.py install
```

Step 5 Install tables.

```
conda install h5py python-blosc
pip install tables
```

Step 6 Install kb-python.

```
cd /home/kb_python
pip install -r requirements.txt
pip install -r dev-requirements.txt
python setup.py install
```

Step 7 Verify kb-python.

```
kb info
```

If the following information is displayed, kb-python is successfully installed.

- For error message "OSError: [Errno 8] Exec format error: '/root/anaconda3/envs/kb/lib/python3.8/site-packages/kb_python-0.26.4 -py3.8.egg/kb_python/bins/linux/kallisto/kallisto'", run the following command:
  ```
  ln -sf /usr/local/bin/bustools /root/anaconda3/envs/kb/lib/python3.8/site-packages/kb_python-0.26.4-py3.8.egg/kb_python/bins/linux/bustools/bustools
  ```
- For error message "OSError: [Errno 8] Exec format error: '/root/anaconda3/envs/kb/lib/python3.8/site-packages/kb_python-0.26.4-py3.8.egg/kb_python/bins/linux/bustools/bustools'", run the following command:
  ```
  ln -sf /usr/local/bin/kallisto /root/anaconda3/envs/kb/lib/python3.8/site-packages/kb_python-0.26.4-py3.8.egg/kb_python/bins/linux/kallisto/kallsto
  ```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.63.7 More Resources

Kallisto official website:

https://www.kallistobus.tools/

4.64 KneadData 0.7.7 Porting Guide (CentOS 8.2)

4.64.1 Introduction

KneadData is a tool designed to perform quality control on metagenomic sequencing data, especially data from microbiome experiments. KneadData invokes Trimmomatic for its quality filtering/trimming, as well as Tandem Repeat Finder (TRF) and FastQC, and Bowtie2 to align your reads to your list of contaminant reference sequences.

For more information about KneadData, visit the **KneadData official website**.

Programming language: Python

Brief description: KneadData is a tool designed to perform quality control on metagenomic sequencing data.

Open source license: MIT

Recommended Software Version

KneadData 0.7.7

4.64.2 Environment Requirements

Hardware Requirements

Table 4-313 lists the hardware requirements.

Table 4-313 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-314 lists the software requirements.

Table 4-314 Software requirements

Item	Version	Download Address
Python	2.7	**https://www.python.org**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-315 lists the OS requirements.

Table 4-315 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.64.3 Paths for Software Porting

This chapter lists the software installation paths involved in the KneadData software porting.

Table 4-316 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/kneaddata	Installation path of KneadData.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/Bowtie2	Installation path of Bowtie2.	

4.64.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-317 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.64.5 Obtaining the Source Code

Procedure

Step 1 Download the KneadData installation package.

https://github.com/biobakery/kneaddata

Step 2 Use an SFTP tool to upload the KneadData installation package to the **/path/to/kneaddata** directory on the server.

Step 3 Download the Bowtie2 installation package.

```
wget http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.4.2/bowtie2-2.4.2-linux-aarch64.zip
```

Step 4 Use SFTP to upload the Bowtie 2 installation package to the */path/to/Bowtie2* directory on the server.

----End

4.64.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the KneadData installation package.

```
cd /path/to/kneaddata
unzip kneaddata-0.7.7-alpha.zip
```

Step 3 Go to the directory generated after the decompression.

```
cd kneaddata-0.7.7-alpha
```

Step 4 Edit the **setup.py** file.

1. Open the **setup.py** file.
   ```
   vi setup.py
   ```
2. As shown in the following figure, add **import shutil**, which will be used when coping the Bowtie2 file.

   ```
   import sys
   import os
   import shutil

   # required python versions (2.7+ or 3.0+)
   required_python_version_major = [2,3]
   required_python_version_minor = [7,0]
   ```

3. In the **def download_unpack_zip** module, add the following content:
   ```
   test1 = r'/path/to/Bowtie2/bowtie2-2.4.2-linux-aarch64.zip'
   shutil.copy(test1,download_file)
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
def download_unpack_zip(url,download_file_name,folder,software_name):
    """
    Download the url to the file and decompress into the folder
    """

    # Check for write permission to the target folder
    if not os.access(folder, os.W_OK):
        print("Warning: The directory is not writeable: "+
              folder + " . Please modify the permissions.")
    download_file=os.path.join(folder, download_file_name)
    download(url, download_file)
    test1 = r'/opt/cpl/kneaddata-master/bowtie2_download_folder/bowtie2-2.4.2-linux-aarch64.zip'
    shutil.copy(test1,download_file)
```

4. Search for **if not bowtie2_installed or replace_install**.

   Comment out the two lines containing **bowtie2-2.2.3-linux.x86_64.zip** and add the following content:

   ```
   bowtie2_file="bowtie2-2.4.2-linux-aarch64.zip"
   bowtie2_url="http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.4.2/bowtie2-2.4.2-linux-aarch64.zip"
   ```

   Comment out **bowtie2_folder="bowtie2-2.2.3"** and add the following content:

   ```
   bowtie2_folder="bowtie2-2.4.2-linux-aarch64"
   ```

```
    if not bowtie2_installed or replace_install:
        bowtie2_exe="bowtie2"
#       bowtie2_file="bowtie2-2.2.3-linux-x86_64.zip"
#       bowtie2_url="http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.3/bowtie2-2.2.3-linux-x86_64.zip"
        bowtie2_file="bowtie2-2.4.2-linux-aarch64.zip"
        bowtie2_url="http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.4.2/bowtie2-2.4.2-linux-aarch64.zip"
        # if this is a MAC OS, select a different binary download
        if mac_os:
            bowtie2_file="bowtie2-2.2.3-macos-x86_64.zip"
            bowtie2_url="http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.3/bowtie2-2.2.3-macos-x86_64.zip"

        bowtie2_folder="bowtie2-2.4.2-linux-aarch64"
#       bowtie2_folder="bowtie2-2.2.3"
```

Step 5 Start the installation.

```
python setup.py install
```

----End

4.64.7 Running and Verification

4.64.7.1 Verifying the Version

Step 1 Query the KneadData version.

```
kneaddata --version
```

----End

4.64.7.2 Verifying the Single-Node Running

Step 1 Run the test case.

```
kneaddata --input examples/demo.fastq --reference-db examples/demo_db --output kneaddata_demo_output
```

```
[root@master kneaddata-0.7.7-alpha]# kneaddata --input examples/demo.fastq --reference-db examples/demo_db --output kneaddata_demo_output
Initial number of reads ( /opt/cpl/kneaddata-0.7.7-alpha/examples/demo.fastq ): 1075
Running Trimmomatic ...
Total reads after trimming ( /opt/cpl/kneaddata-0.7.7-alpha/kneaddata_demo_output/demo_kneaddata.trimmed.fastq ): 1075
Decontaminating ...
Running bowtie2 ...
Total reads after removing those found in reference database ( /opt/cpl/kneaddata-0.7.7-alpha/kneaddata_demo_output/demo_kneaddata_demo_db_bowtie2_clean.fastq ): 1065
Total reads after merging results from multiple databases ( /opt/cpl/kneaddata-0.7.7-alpha/kneaddata_demo_output/demo_kneaddata.fastq ): 1065

Final output file created:
/opt/cpl/kneaddata-0.7.7-alpha/kneaddata_demo_output/demo_kneaddata.fastq

[root@master kneaddata-0.7.7-alpha]#
```

Activate Windows
Go to Settings to activate Windows.

Step 2 View the output file.

```
ls kneaddata_demo_output
```

----End

4.65 Kraken2 Porting Guide (CentOS 8.2)

4.65.1 Introduction

Kraken2 is a taxonomic classification system using exact k-mer matches to achieve high accuracy and fast classification speeds. It assigns a taxonomic identification to each sequencing read, by using the lowest common ancestor (LCA) of matching genomes in the database.

For more information about Kraken2, visit the **Kraken2 official website**.

Programming language: C++

Brief description: Kraken2 is a taxonomic classification system using exact k-mer matches to achieve high accuracy and fast classification speeds.

Open source license: MIT

Recommended Software Version

Kraken 2.1.2

4.65.2 Environment Requirements

Hardware Requirements

Table 4-318 lists the hardware requirements.

Table 4-318 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-319 lists the OS requirements.

Table 4-319 OS requirements

Item	Version	Download URL
CentOS	8.2	**https://vault.centos.org/8.2.2004/isos/aarch64/**

 Copyright © Huawei Technologies Co., Ltd.

4.65.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Kraken2 software porting.

Table 4-320 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/kraken2	Installation path of Kraken2.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.65.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-321 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.65.5 Obtaining the Source Code

Procedure

Step 1 Download the Kraken2 installation package.

https://github.com/DerrickWood/kraken2

Step 2 Use an SFTP tool to upload the Kraken2 installation package to the **/path/to/kraken2** directory on the server.

----End

4.65.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/kraken2
```

Step 3 Decompress the installation package.

```
unzip kraken2-master.zip
```

Step 4 Modify the source code.

```
vi src/Makefile
```

Add **-march=armv8-a -fsigned-char**:

```
CXX = g++
CXXFLAGS = -fopenmp -Wall -std=c++11 -O3 -march=armv8-a -fsigned-char
CXXFLAGS += -DLINEAR_PROBING
```

```
vi src/hyperloglogpuls.cc
```

Modify lines 46 to 50 as follows:

```
#ifdef WIN32
#include <intrin.h>
#define __builtin_clz( x ) _lzcnt( x )
#define __builtin_clzl( x ) _lzcnt64( x )
#elif defined(__aarch64__)
//  Suggestion:  Add code that adapts to the Kunpeng platdorm.
#endif
```

```
#ifdef WIN32
#include <intrin.h>
#define __builtin_clz(x) __lzcnt(x)
#define __builtin_clzl(x) __lzcnt64(x)
#elif defined(__aarch64__)
// Suggestion:  Add code that adapts to the Kunpeng platform.
#endif
```

Step 5 Perform the compilation and installation.

```
sh install_kraken2.sh /path/to/kraken2
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.65.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 View the help information.

/path/to/kraken2/**kraken2 -help**

```
[root@localhost kraken2-master]# /home/kraken2/kraken2 --help
Usage: kraken2 [options] <filename(s)>

Options:
  --db NAME               Name for Kraken 2 DB
                          (default: none)
  --threads NUM           Number of threads (default: 1)
  --quick                 Quick operation (use first hit or hits)
  --unclassified-out FILENAME
                          Print unclassified sequences to filename
  --classified-out FILENAME
                          Print classified sequences to filename
  --output FILENAME       Print output to filename (default: stdout); "-" will
                          suppress normal output
  --confidence FLOAT      Confidence score threshold (default: 0.0); must be
                          in [0, 1].
  --minimum-base-quality NUM
                          Minimum base quality used in classification (def: 0,
                          only effective with FASTQ input).
  --report FILENAME       Print a report with aggregrate counts/clade to file
  --use-mpa-style         With --report, format report output like Kraken 1's
                          kraken-mpa-report
  --report-zero-counts    With --report, report counts for ALL taxa, even if
                          counts are zero
  --report-minimizer-data With --report, report minimizer and distinct minimizer
                          count information in addition to normal Kraken report
  --memory-mapping        Avoids loading database into RAM
  --paired                The filenames provided have paired-end reads
  --use-names             Print scientific names instead of just taxids
  --gzip-compressed       Input files are compressed with gzip
  --bzip2-compressed      Input files are compressed with bzip2
  --minimum-hit-groups NUM
                          Minimum number of hit groups (overlapping k-mers
                          sharing the same minimizer) needed to make a call
                          (default: 2)
  --help                  Print this message
  --version               Print version information
```

----End

4.66 MetaPhlAn 3.0.13 Porting Guide (CentOS 8.2)

4.66.1 Introduction

MetaPhlAn is a computational tool for profiling the composition of microbial communities (Bacteria, Archaea and Eukaryotes) from metagenomic shotgun sequencing data (i.e. not 16S) with species-level. With the newly added StrainPhlAn module, it is now possible to perform accurate strain-level microbial profiling.

For more information about MetaPhlAn, visit the **MetaPhlAn official website**.

Programming language: Python

Brief description: MetaPhlAn is a computational tool for profiling the composition of microbial communities.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: custom open source license

Recommended Software Version

MetaPhlAn 3.0.13

4.66.2 Environment Requirements

Hardware Requirements

Table 4-322 lists the hardware requirements.

Table 4-322 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-323 lists the software requirements.

Table 4-323 Software requirements

Item	Version	Download Address
Python	3.9.6	https://www.python.org/downloads/release/python-396/
Samtools	1.10	https://github.com/samtools/samtools/releases/download/1.10/samtools-1.10.tar.bz2

OS Requirements

Table 4-324 lists the OS requirements.

Table 4-324 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.66.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MetaPhlAn software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-325 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/Python-3.9.6	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.66.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-326 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See **User Guide (GCC for openEuler)**.

4.66.5 Obtaining the Source Code

Procedure

Step 1 Download the Python installation package.

https://www.python.org/downloads/release/python-396/

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Python installation package to the **/path/to/Python-3.9.6** directory on the server.

----End

4.66.6 Compilation and Installation

4.66.6.1 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Python-3.9.6 installation package.

```
cd /path/to/Python-3.9.6
tar -xvf Python-3.9.6.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd Python-3.9.6
```

Step 4 Install the dependencies.

```
yum install gcc libffi* openssl libssl zlib* bzip2
```

Step 5 Perform the compilation and installation.

```
./configure --prefix=/path/to/Python-3.9.6 --enable-shared --enable-optimizations
make
make install
```

Step 6 Configure the variables.

1. Open the **/etc/profile** file.
   ```
   vi /etc/profile
   ```
2. Press **i** to enter the edit mode and add the following content:
   ```
   export PATH=/path/to/Python-3.9.6/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/Python-3.9.6/lib:$LD_LIBRARY_PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.
4. Reload the environment variables.
   ```
   source /etc/profile
   ```

Step 7 View the version information.

```
python3 --version
pip3 --version
```

If the correct version is displayed, the installation is successful.

```
[root@ceph3 home]# python3 --version
Python 3.9.6
[root@ceph3 home]# pip3 --version
pip 21.1.3 from /usr/local/lib/python3.9/site-packages/pip (python 3.9)
[root@ceph3 home]#
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.66.6.2 Installing Samtools

Procedure

Step 1 Download the Samtools installation package **samtools-1.10.tar.bz2**.

https://github.com/samtools/samtools/releases/download/1.10/samtools-1.10.tar.bz2

Step 2 Install the dependencies.

```
yum install -y bzip2 bzip2-devel ncurses ncurses-devel xz xz-devel libcurl libcurl-devel zlib zlib-devel
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar -jxvf samtools-1.10.tar.bz2
cd samtools-1.10
```

Step 4 Perform the compilation and installation.

```
./configure
make && make install
```

Step 5 View the version information.

```
samtools --version
```

If the correct version is displayed, the installation is successful.

```
[root@ceph3 home]# samtools --version
samtools 1.10
Using htslib 1.10
Copyright (C) 2019 Genome Research Ltd.
[root@ceph3 home]#
```

----End

4.66.6.3 Installing MetaPhlAn

Procedure

Step 1 Install MetaPhlAn.

```
pip3 install metaphlan -i https://mirrors.aliyun.com/pypi/simple --trusted-host mirrors.aliyun.com
```

Step 2 View the version information.

```
metaphlan --version
```

If the correct version is displayed, the installation is successful.

```
[root@ceph3 ~]# metaphlan --version
MetaPhlAn version 3.0.13 (27 Jul 2021)
[root@ceph3 ~]#
```

----End

4.66.7 Running and Verification

 Copyright © Huawei Technologies Co., Ltd.

4.66.7.1 Downloading the Input File

Step 1 Download the input file.

https://raw.githubusercontent.com/biobakery/biobakery/master/test_suite/biobakery_tests/data/metaphlan3/input/SRS014470-Tongue_dorsum.fasta.gz

----End

4.66.7.2 Installing Bowtie2

Procedure

Step 1 Download Bowtie2.

http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.4.2/bowtie2-2.4.2-linux-aarch64.zip

Step 2 Decompress the installation package.

```
unzip bowtie2-2.4.2-linux-aarch64.zip
```

Step 3 Edit the configuration file.

1. Open the configuration file.
   ```
   vi /etc/profile
   ```
2. Press **i** to enter the edit mode and add the Bowtie2 path.
   ```
   export PATH=/home/gcc-9.3.1-2020.12-aarch64-linux/bin:/usr/local/python-3.9.6/bin:/home/bowtie2-2.4.2-linux-aarch64:$PATH
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 4 Make the configuration take effect.

```
source /etc/profile
```

Step 5 Check whether the installation is successful.

```
bowtie2 -h
```

The installation is successful if the following information is displayed:

 Copyright © Huawei Technologies Co., Ltd.

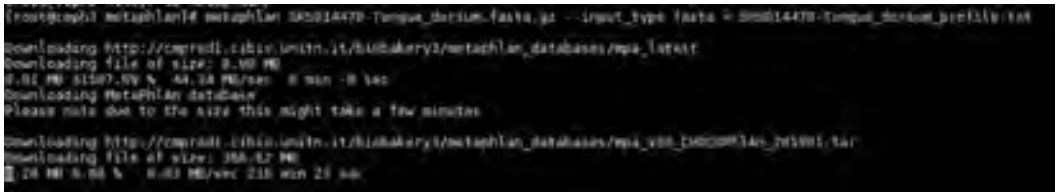
```
[root@ceph3 bowtie2-2.4.2-linux-aarch64]# bowtie2 -h
Bowtie 2 version 2.4.2 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)
Usage:
  bowtie2 [options]* -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r> | --interleaved <i> | -b <bam>} [-S <
sam>]

  <bt2-idx>  Index filename prefix (minus trailing .X.bt2).
             NOTE: Bowtie 1 and Bowtie 2 indexes are not compatible.
  <m1>       Files with #1 mates, paired with files in <m2>.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <m2>       Files with #2 mates, paired with files in <m1>.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <r>        Files with unpaired reads.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <i>        Files with interleaved paired-end FASTQ/FASTA reads
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
  <bam>      Files are unaligned BAM sorted by read name.
  <sam>      File for SAM output (default: stdout)

  <m1>, <m2>, <r> can be comma-separated lists (no whitespace) and can be
  specified many times.  E.g. '-U file1.fq,file2.fq -U file3.fq'.

Options (defaults in parentheses):

 Input:
  -q                 query input files are FASTQ .fq/.fastq (default)
  --tab5             query input files are TAB5 .tab5
  --tab6             query input files are TAB6 .tab6
  --qseq             query input files are in Illumina's qseq format
  -f                 query input files are (multi-)FASTA .fa/.mfa
```

----End

4.66.7.3 Viewing the Running Result

Step 1 Run the test case.

```
metaphlan SRS014470-Tongue_dorsum.fasta.gz --input_type fasta > SRS014470-Tongue_dorsum_profile.txt
```



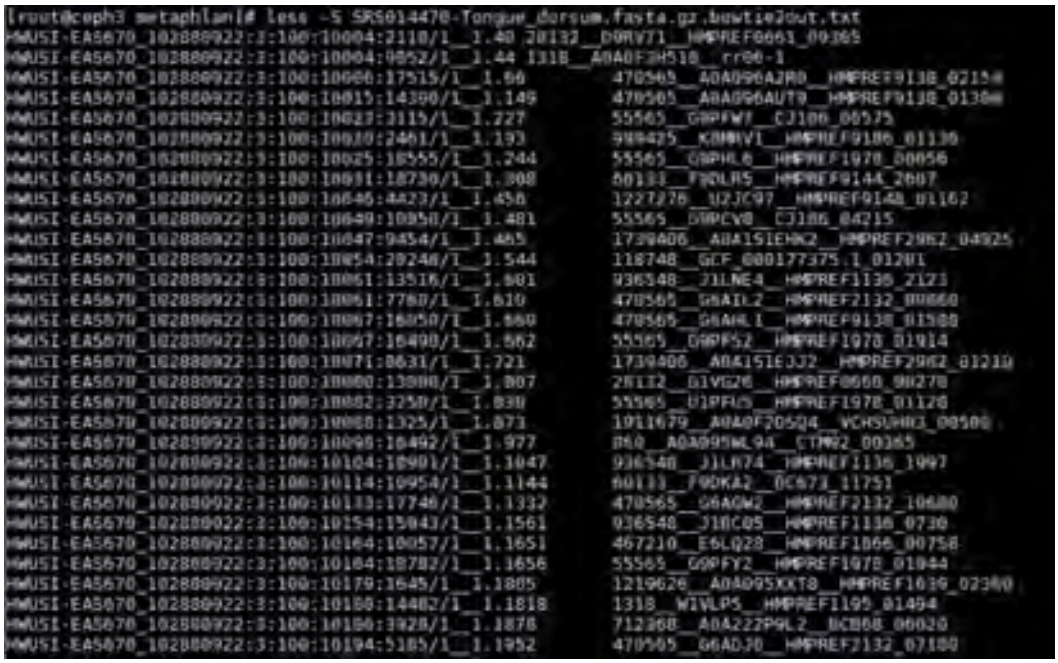
Step 2 View the output file.

```
ls -ltr
```

Step 3 View the running result.

```
less -S SRS014470-Tongue_dorsum.fasta.gz.bowtie2out.txt
```

 Copyright © Huawei Technologies Co., Ltd.



less -S SRS014470-Tongue_dorsum_profile.txt

----End

4.66.8 Troubleshooting

4.66.8.1 Python Installation Failure

Symptom

Python fails to be installed and the following error information is displayed:

```
Traceback (most recent call last):
  File "<frozen zipimport>", line 520, in _get_decompress_func
ModuleNotFoundError: No module named 'zlib'
```

Key Process and Cause Analysis

The zlib library is missing.

Conclusion and Solution

Install zlib.

```
yum install zlib*
```

4.66.8.2 SSL Exception Occurs During MetaPhlAn Installation

Symptom

When installing MetaPhlAn, a message is displayed indicating that the SSL module is unavailable.

Key Process and Cause Analysis

The SSL module is unavailable.

Conclusion and Solution

Step 1 Recompile Python and add the following compilation options after **configure**:

```
./configure --prefix=/path/to/Python-3.9.6  --enable-shared --enable-optimizations
```

Step 2 Perform the installation.

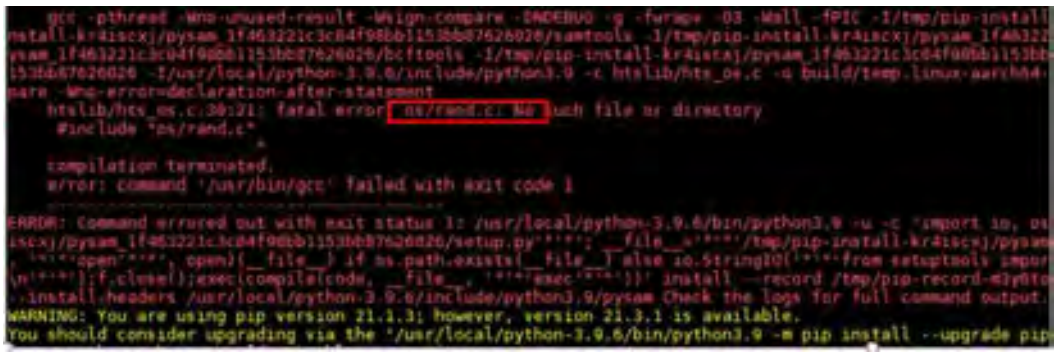
```
make
make install
```

----End

4.66.8.3 os/rand.c File Not Found When Installing MetaPhlAn

Symptom

During MetaPhlAn installation, an error message is displayed indicating that the **os/rand.c** file is missing.



 Copyright © Huawei Technologies Co., Ltd.

Key Process and Cause Analysis

The **os/rand.c** file is missing.

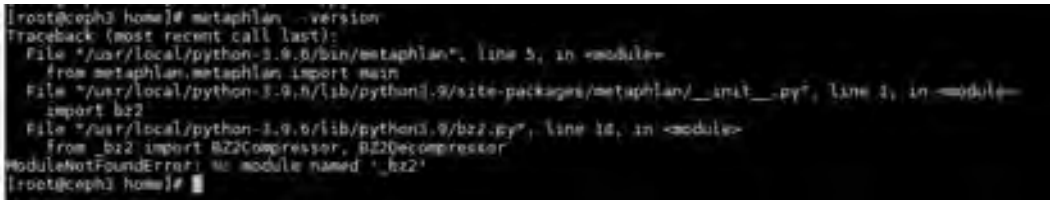
Conclusion and Solution

Install Samtools. For details, see **4.66.6.2 Installing Samtools**.

4.66.8.4 MetaPhlAn Version Verification Error

Symptom

When verifying the installed MetaPhlAn version, a message is displayed indicating that **_bz2** is not found.



Key Process and Cause Analysis

The dependencies are missing.

Conclusion and Solution

Step 1 Install bzip2 (see **4.66.6.1 Installing Python**) and then compile and install Python again.

Step 2 Reinstall MetaPhlAn. For details, see **4.66.6.3 Installing MetaPhlAn**.

----End

4.66.8.5 An Error Occurs When Running a Single Sample

Symptom

When running a single sample, a message is displayed indicating that **bowtie2-build** is missing.

 Copyright © Huawei Technologies Co., Ltd.

Key Process and Cause Analysis

The dependencies are missing.

Conclusion and Solution

Install Bowtie2. For details, see **4.66.7.2 Installing Bowtie2**.

4.67 muparser v2.3.2 Porting Guide (openEuler 20.03)

4.67.1 Introduction

muparser is an extensible high performance math expression parser library. It works by transforming a mathematical expression into bytecode and precalculating constant parts of the expression. For best performance muparser allows parallelized evaluation of expressions.

For more information about Kraken2, visit the **muparser official website**.

Programming language: C++

Brief description: muparser is an extensible high performance math expression parser library.

Open source license: BSD-2-Clause License

Recommended Software Version

muparser v2.3.2

4.67.2 Environment Requirements

Hardware Requirements

Table 4-327 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-327 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-328 lists the software requirements.

Table 4-328 Software requirements

Item	Version	Download Address
muparser	v2.3.2	https://github.com/beltoforion/muparser/archive/refs/tags/v2.3.2.tar.gz

OS Requirements

Table 4-329 lists the OS requirements.

Table 4-329 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	https://repo.openeuler.org/openEuler-20.03-LTS-SP1/
Kernel	4.19	Included in the OS image

4.67.3 Paths for Software Porting

This chapter lists the software installation paths involved in the muparser software porting.

Table 4-330 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Planned installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in User Guide (GCC for openEuler).

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/muparser	Installation path of muparser.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.67.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-331 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See **User Guide (GCC for openEuler)**.

4.67.5 Obtaining the Source Code

Procedure

Step 1 Download the muparser installation package.

https://github.com/beltoforion/muparser/archive/refs/tags/v2.3.2.tar.gz

Step 2 Use an SFTP tool to upload the muparser installation package to the **/path/to/muparser** directory on the server.

----End

4.67.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where the muparser installation package is stored and decompress the package.

```
cd /path/to/muparser
tar -zxvf v2.3.2.tar.gz
```

Step 3 Go to the installation directory.

```
cd muparser-2.3.2/
```

Step 4 Perform the compilation.

```
cmake . -DENABLE_SAMPLES=ON -DENABLE_OPENMP=OFF -DENABLE_WIDE_CHAR=OFF -
DBUILD_SHARED_LIBS=ON
make -j64
make install
ldconfig
```

----End

4.67.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the muparser directory.

```
cd /path/to/muparser/muparser-2.3.2
```

Step 3 Run the following command:

```
./example1
```

If the following information is displayed, muparser is successfully installed.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost muparser-2.3.2]# ./example1


  Version 2.3.2 (20200617; 64BIT; RELEASE; ASCII; OPENMP)
  (C) 2004 - 2020 Ingo Berg

------------------------------------------------------------
  Compiled with GCC Version 7.3.0
  IEEE 754 (IEC 559) is Available
  64-bit build
------------------------------------------------------------
Running unit tests:

testing name restriction enforcement...passed
testing syntax engine...passed
testing postfix operators...passed
testing infix operators...passed
testing variable/constant detection...passed
testing multiarg functions...passed
testing expression samples...passed
testing if-then-else operator...passed
testing member functions...passed
testing binary operators...passed
testing error codes...passed
testing string arguments...passed
testing bulkmode...passed
Test passed (569 expressions)
------------------------------------------------------------
Commands:

  list var     - list parser variables
  list exprvar - list expression variables
  list const   - list all numeric parser constants
  opt on       - enable optimizer (default)
  opt off      - disable optimizer
  locale de    - switch to german locale
  locale en    - switch to english locale
  locale reset - reset locale
  test bulk    - test bulk mode
  quit         - exits the parser

Constants:

  "_e"   2.71828182845904523536028747135266249775724709369995
  "_pi"  3.14159265358979323846264338327950288419716939937510
------------------------------------------------------------
Enter an expression or a command:
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.67.8 More Resources

Reference document:

https://github.com/beltoforion/muparser/blob/master/Install.txt

4.68 Open CASCADE Technology 7.5.0 Porting Guide (openEuler 20.03)

4.68.1 Introduction

Open CASCADE Technology (OCCT) is an open-source full-scale 3D geometry library. OCCT is widely used for the development of specialized programs dealing with the following engineering and mechanical domains:

3D modeling (CAD), manufacturing (CAM), numerical simulation (CAE), measurement equipment (CMM) and quality control (CAQ). OCCT provides six modules by organizing C++ library files. As a core part of the OCC, the visualization module is the implementation of the visualization technology.

For more information about OCC, visit the **OCCT official website**.

Programming language: C++

Brief description: OCCT is an object-oriented C++ class library designed for rapid production of sophisticated domain-specific CAD/CAM/CAE applications.

Open source license: LGPL-2.1 License

Recommended Software Version

Open CASCADE Technology 7.5.0

4.68.2 Environment Requirements

Hardware Requirements

Table 4-332 lists the hardware requirements.

Table 4-332 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-333 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-333 Software requirements

Item	Version	Download Address
Open CASCADE Technology	7.5.0	https://github.com/Open-Cascade-SAS/OCCT/archive/refs/tags/V7_5_0.tar.gz
FreeType	2.10.0	https://sourceforge.net/projects/freetype/files/freetype2/2.10.0/freetype-2.10.0.tar.gz
FreeImage	3.18.0	http://downloads.sourceforge.net/freeimage/FreeImage3180.zip

OS Requirements

Table 4-334 lists the OS requirements.

Table 4-334 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	https://repo.openeuler.org/openEuler-20.03-LTS-SP1/
Kernel	4.19	Included in the OS image

4.68.3 Paths for Software Porting

The following table lists the software installation paths involved in the Open CASCADE Technology software porting.

Table 4-335 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/FREETYPE	Installation path of FreeType.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.
2	/path/to/FREEIMGE	Installation path of FreeImage.	
4	/path/to/OCCT	Installation path of Open CASCADE Technology.	

 Copyright © Huawei Technologies Co., Ltd.

4.68.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-336 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Installing FreeType	See **4.68.4.1 Installing FreeType**.
4	Installing FreeImage	See **4.68.4.2 Installing FreeImage**.

4.68.4.1 Installing FreeType

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the FreeType installation package.

```
wget https://sourceforge.net/projects/freetype/files/freetype2/2.10.0/freetype-2.10.0.tar.gz
```

Step 3 Decompress the FreeType installation package.

```
tar -zxvf freetype-2.10.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd freetype-2.10.0
```

Step 5 Perform the configuration.

```
./configure  --prefix=/path/to/FREETYPE
```

Step 6 Perform the compilation and installation.

```
make -j64
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.68.4.2 Installing FreeImage

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the FreeImage installation package.

```
wget http://downloads.sourceforge.net/freeimage/FreeImage3180.zip
```

Step 3 Decompress the FreeImage installation package.

```
unzip FreeImage3180.zip
```

Step 4 Go to the directory generated after the decompression.

```
cd FreeImage
```

Step 5 Edit the **Makefile.gnu** configuration file.

```
sed -i '26s/privacy/privacy -Wno-narrowing/g' Makefile.gnu
```

Step 6 Configure the environment variables.

```
export CC=gcc CXX=g++ FC=gfortran
```

Step 7 Perform the compilation and installation.

```
make -j64
make DESTDIR=/path/to/FREEIMGE install
```

Step 8 Copy the **include** and **lib** directories to the parent directory.

```
cd /path/to/FREEIMGE/usr
cp -r /path/to/FREEIMGE/usr/* /path/to/FREEIMGE/
```

----End

4.68.5 Obtaining the Source Code

Procedure

Step 1 Download the Open CASCADE Technology installation package.

https://github.com/Open-Cascade-SAS/OCCT/archive/refs/tags/V7_5_0.tar.gz

Step 2 Use an SFTP tool to upload the Open CASCADE Technology installation package to the */path/to/OCCT* directory on the server.

----End

4.68.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Open CASCADE Technology installation package.

```
wget https://github.com/Open-Cascade-SAS/OCCT/archive/refs/tags/V7_5_0.tar.gz
```

Step 3 Decompress the Open CASCADE Technology installation package.

```
tar -zxvf V7_5_0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

 Copyright © Huawei Technologies Co., Ltd.

```
cd OCCT-7_5_0
```

Step 5 Create a **build** directory.

```
mkdir build
```

Step 6 Go to the **build** directory.

```
cd build
```

Step 7 Perform the configuration.

```
cmake -DINSTALL_DIR=/path/to/OCCT -DUSE_TCL=OFF -DBUILD_MODULE_Draw=OFF -
DUSE_FREETYPE=ON -D3RDPARTY_FREETYPE_DIR=/path/to/FREETYPE -DUSE_GL2PS=OFF -
DUSE_FREEIMAGE=ON -D3RDPARTY_FREEIMAGE_DIR=/path/to/FREEIMGE -DCMAKE_C_COMPILER=gcc -
DCMAKE_CXX_COMPILER=g++ ../
```

Step 8 Perform the compilation and installation.

```
make -j64
make install
```

----End

4.68.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory where Open CASCADE Technology is installed.

```
cd /path/to/OCCT
```

Step 3 Check for library files.

```
ls lib
```

- If the preceding information is displayed, the installation is successful.
- Otherwise, perform **4.68.4 Configuring the Compilation Environment** to **4.68.6 Compilation and Installation** to perform the installation again or contact Huawei technical support.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.68.8 More Resources

Reference document:

https://github.com/Open-Cascade-SAS/OCCT/tags

4.69 PYCXX 6.2.8 Porting Guide (openEuler 20.03)

4.69.1 Introduction

PyCXX is a set of classes to help create extensions of Python in the C++ language. The first part encapsulates the Python C API taking care of exceptions and ref counting. The second part supports the building of Python extension modules in C ++.

For more information about PYCXX, visit the **PYCXX official website**.

Programming language: Python

Brief description: PyCXX is a set of classes to help create extensions of Python in the C++ language.

Open source license: custom open source license

Recommended Software Version

PYCXX 6.2.8

4.69.2 Environment Requirements

Hardware Requirements

Table 4-337 lists the hardware requirements.

Table 4-337 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-338 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-338 Software requirements

Item	Version	Download Address
PYCXX	6.2.8	https://sourceforge.net/projects/cxx/files/CXX/PyCXX%20V6.2.8/pycxx-6.2.8.tar.gz

OS Requirements

Table 4-339 lists the OS requirements.

Table 4-339 OS requirements

Item	Version	Download URL
openEuler	20.03	https://www.openeuler.org/en/download/

4.69.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PYCXX software porting.

Table 4-340 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/PYCXX	Installation path of PYCXX.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.69.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configure Process

Table 4-341 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.69.5 Obtaining the Source Code

Procedure

Step 1 Download the PYCXX installation package.

https://sourceforge.net/projects/cxx/files/CXX/PyCXX%20V6.2.8/pycxx-6.2.8.tar.gz

Step 2 Use an SFTP tool to upload the PYCXX installation package to the */path/to/PYCXX* directory on the server.

----End

4.69.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the PYCXX installation directory.

```
cd /path/to/PYCXX
```

Step 3 Decompress the PYCXX installation package.

```
tar -zxvf pycxx-6.2.8.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd pycxx-6.2.8
```

Step 5 Perform the compilation and installation.

```
python setup.py install
python setup_makefile.py linux linux.mak
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.69.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the PYCXX installation directory.

```
cd /path/to/PYCXX
```

Step 3 Verify the installation.

```
make -f linux.mak clean test
```

```
PYTHONPATH=obj /usr/bin/python3 -W default Demo/Python3/test_pycxx_iter.py
5 IterT count(6)
6 IterT count(7)
7 IterT count(8)
refcount of it: 2
7
6
5
refcount of it: 2
```

- If the preceding information is displayed, the installation is successful.
- Otherwise, perform **4.69.4 Configuring the Compilation Environment** to **4.69.6 Compilation and Installation** to perform the installation again or contact Huawei technical support.

----End

4.69.8 More Resources

Reference document:

http://cxx.sourceforge.net/

4.70 Python 3.8.2 Porting Guide (openEuler 20.03)

4.70.1 Introduction

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python was invented by Guido van Rossum at the end of 1989, and the first version was released in 1991. Like Perl, Python complies with GNU General Public License (GPL).

For more information about Qt, visit the **Python official website**.

Programming language: Python

Brief description: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Open source license: GPL

Recommended Software Version

Python 3.8.2

 Copyright © Huawei Technologies Co., Ltd.

4.70.2 Environment Requirements

Hardware Requirements

Table 4-342 lists the hardware requirements.

Table 4-342 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-343 lists the software requirements.

Table 4-343 Software requirements

Item	Version	Download Address
Python	3.8.2	https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz

OS Requirements

Table 4-344 lists the OS requirements.

Table 4-344 OS requirements

Item	Version	Download Address
openEuler	20.03	https://www.openeuler.org/en/download/

4.70.3 Paths for Software Porting

The following table lists the software installation paths involved in the Python 3.8.2 software porting.

Table 4-345 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in User Guide (GCC for openEuler).

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/PYTHON	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.70.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-346 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.70.5 Obtaining the Source Code

Procedure

Step 1 Download the Python installation package.

https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz

Step 2 Use an SFTP tool to upload the Python-3.8.2 installation package to the */path/to/PYTHON* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.70.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the Python-3.8.2.tgz installation package.

```
cd /path/to/PYTHON
```

Step 3 Decompress the Python installation package.

```
tar -zxvf Python-3.8.2.tgz
```

Step 4 Go to the directory generated after the decompression.

```
cd Python-3.8.2
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/PYTHON --enable-shared --enable-loadable-sqlite-extensions --enable-optimizations
```

Step 6 Perform the compilation and installation.

```
make -j64 && make install
```

Step 7 Configure the environment variables.

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 8 Add a soft link.

```
rm -f /usr/bin/python3
rm -f /usr/bin/pip3
ln -s /path/to/PYTHON/bin/python3 /usr/bin/python3
ln -s /path/to/PYTHON/bin/pip3 /usr/bin/pip3
```

----End

4.70.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Configure the environment variables.

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 3 Query the Python version.

```
python3 -V
```

If the correct version is displayed, the installation is successful.

Step 4 View the pip version.

```
pip3 -V
```

If the Python3 version is displayed, the installation is successful.

Step 5 Start Python.

```
python3
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.70.8 Troubleshooting

4.70.8.1 Error Reported When Starting Python3

Symptom

When starting Python3, a message is displayed indicating that libpython3.8.so.1.0 is not found.

Key Process and Cause Analysis

The libpython3.8.so.1.0 library file is not loaded when starting Python3.

Conclusion and Solution

Copy libpython3.8.so.1.0 from the decompressed compilation directory to the library directory.

```
cp libpython3.8.so.1.0 /usr/local/lib64/
cp libpython3.8.so.1.0 /usr/lib64/
cp libpython3.8.so.1.0 /usr/lib/
```

4.70.8.2 Error Reported for Running the Yum Command After Python3 Is Installed

Symptom

After Python3 is installed, the message "ModuleNotFoundError: No module named 'dnf'" is displayed when you use Yum.

```
[root@localhost ~]# yum list
Traceback (most recent call last):
  File "/usr/bin/yum", line 57, in <module>
    from dnf.cli import main
ModuleNotFoundError: No module named 'dnf'
```

Key Process and Cause Analysis

openEuler has built-in Python 3.7, which uses the Yum from the dnf soft link and is closely related to libpython3.7m.so.1.0.

```
[root@localhost ~]# ll /usr/bin/yum
lrwxrwxrwx. 1 root root 5 Dec 21 23:10 /usr/bin/yum -> dnf-3
```

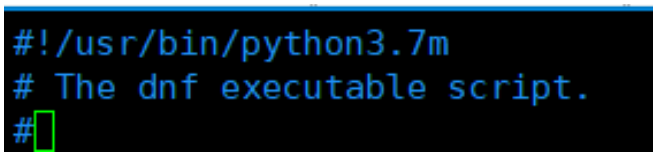
 Copyright © Huawei Technologies Co., Ltd.

Conclusion and Solution

Step 1 Open the **/usr/bin/yum** file.

```
vi /usr/bin/yum
```

Step 2 Press **i** to enter the edit mode. Change **#!/usr/bin/python3** at the beginning to **#!/usr/bin/python3.7m**.



Step 3 Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

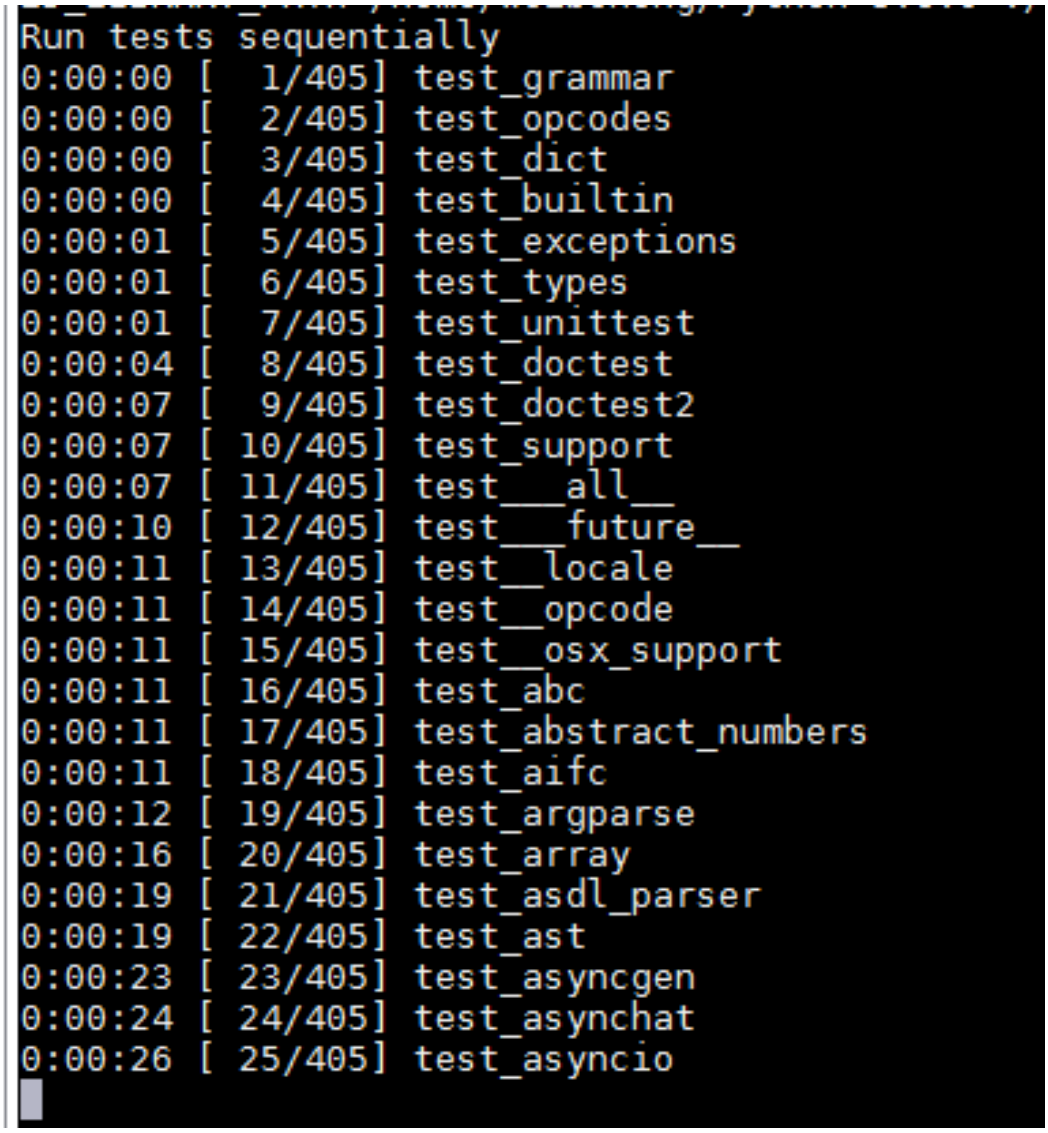
----End

4.70.8.3 System Suspended When Running the "Run tests sequebtially" Command

Symptom

When running **Run tests sequebtially**, the system is suspended at asyncio.

 Copyright © Huawei Technologies Co., Ltd.



Key Process and Cause Analysis

Compile and install Python 3 (for example, Python 3.6.0) of the built-in test set. As a result, the system is suspended at asyncio.

Conclusion and Solution

- Solution 1

  Remove the optimization parameters from the **configure** step so that the test set is not verified during compilation.

- Solution 2

  Remove the asynicio test script from the **Lib/test/** directory.

4.71 Qt 5.13.0 Porting Guide (openEuler 20.03)

4.71.1 Introduction

Qt is a cross-platform C++ graphical user interface (GUI) application development framework developed by Qt Company in 1991. It can be used to develop both GUI programs and non-GUI programs, such as console tools and servers. Qt is an object-oriented framework that uses special code generation extensions (called Meta Object Compiler, MOC) and macros. Qt is easy to extend and allows true component-based programming.

For more information about Qt, visit the **Qt official website**.

Programming language: C++

Brief description: Qt is a cross-platform C++ GUI application development framework.

Open source license: GPL

Recommended Software Version

Qt 5.13.0

4.71.2 Environment Requirements

Hardware Requirements

Table 4-347 lists the hardware requirements.

Table 4-347 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-348 lists the software requirements.

Table 4-348 Software requirements

Item	Version	Download Address
GCC for openEuler	10.3.0	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz**

OS Requirements

Table 4-349 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-349 OS requirements

Item	Version	Download Address
openEuler	20.03	**https://www.openeuler.org/en/download/**

4.71.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Qt software porting.

Table 4-350 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/QT	Installation path of Qt.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.71.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-351 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Setting up the GCC for openEuler compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.71.5 Obtaining the Source Code

Procedure

Step 1 Download the Qt installation package.

http://download.qt.io/archive/qt/5.13/5.13.0/single/qt-everywhere-src-5.13.0.tar.xz

Step 2 Use an SFTP tool to upload the Qt installation package to the */path/to/QT* directory on the server.

----End

4.71.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Qt installation directory.

```
cd /path/to/QT
```

Step 3 Decompress the Qt installation package.

```
tar -Jxvf qt-everywhere-src-5.13.0.tar.xz
```

Step 4 Go to the directory generated after the decompression.

```
cd qt-everywhere-src-5.13.0
```

Step 5 Perform the configuration.

```
./configure -qt-xcb
```

Step 6 Select an installation type.

1. Enter **o**, which indicates an open source version
2. Enter **yes**, which indicates that you accept the GPL license

 Copyright © Huawei Technologies Co., Ltd.

```
[root@33914d1b44a4 qt-everywhere-src-5.13.0]# ./configure -qt-xcb
+ cd qtbase
+ /root/qt-everywhere-src-5.13.0/qtbase/configure -top-level -qt-xcb
Creating qmake...
.....................................................................................Done.
Info: creating super cache file /root/qt-everywhere-src-5.13.0/.qmake.super
Info: creating cache file /root/qt-everywhere-src-5.13.0/.qmake.cache
Info: creating stash file /root/qt-everywhere-src-5.13.0/.qmake.stash

Selecting Qt Edition.

Type 'c' if you want to use the Commercial Edition.
Type 'o' if you want to use the Open Source Edition.

Which edition of Qt do you want to use? o

This is the Qt Open Source Edition.

You are licensed to use this software under the terms of
the GNU Lesser General Public License (LGPL) version 3
or the GNU General Public License (GPL) version 2.

Type 'L' to view the GNU Lesser General Public License version 3 (LGPLv3).
Type 'G' to view the GNU General Public License version 2 (GPLv2).
Type 'y' to accept this license offer.
Type 'n' to decline this license offer.

Do you accept the terms of either license? yes
```

Step 7 Perform the compilation and installation.

```
gmake –j64
gmake install
```

Step 8 Configure the environment variables.

```
export QTDIR=/usr/local/Qt-5.13.0
export PATH=$QTDIR/bin:$PATH
export MANPATH=$QTDIR/man:$MANPATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
```

NOTE

/usr/local is the default installation directory.

----End

4.71.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether GCC for openEuler has been loaded.

```
gcc -v
```

- If yes, go to **Step 4**.
- If no, perform **Step 3** to **Step 4**.

Step 3 Configure the environment variables.

```
export QTDIR=/usr/local/Qt-5.13.0
export PATH=$QTDIR/bin:$PATH
export MANPATH=$QTDIR/man:$MANPATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
```

Step 4 Verify the qmake version.

```
qmake –v
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost ~]# qmake -v
QMake version 3.1
Using Qt version 5.13.0 in /usr/local/Qt-5.13.0/lib
```

----End

4.71.7.1 Compiling and Generating helloworld

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a **helloworld** directory.

```
mkdir helloworld
```

Step 3 Go to the **helloworld** directory.

```
cd helloworld
```

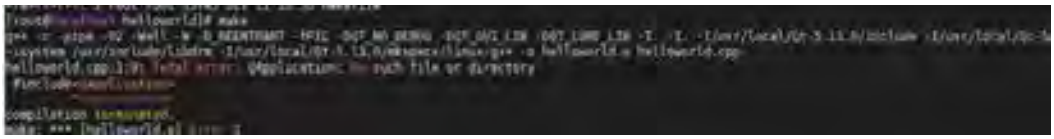
Step 4 Create a **helloworld.cpp** file and edit it.

1. Create and open the **helloworld.cpp** file.
   ```
   vi helloworld.cpp
   ```
2. Press **i** to enter insert mode and add the following information:
   ```
   #include<QApplication>
   #include<QLabel>
   int main(int argc,char *argv[])
    {
   QApplication app(argc,argv);
   QLabel *label=new QLabel("helloworld!");
   label->show();
   return app.exec();
    }
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the compilation.

```
qmake -project
qmake
make
```

If the following error information is displayed after running the **make** command, add **QT+=widgets** to the end of the **helloworld.pro** file.



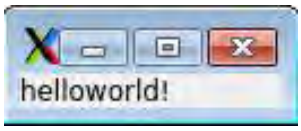
NOTE

helloworld.pro is the **a.pro** file generated by running the **qmake -project** command. **a** indicates the directory name.

Step 6 Run **helloworld**.

```
./helloworld
```

 Copyright © Huawei Technologies Co., Ltd.



----End

4.71.7.2 Compiling and Generating standarddialogs

Procedure

Step 1 Go to the **standarddialogs** directory.

```
cd /usr/local/Qt-5.13.0/examples/widgets/dialogs/standarddialogs
```

Step 2 Start the compilation.

```
qmake standarddialogs.pro
make -j64
```

Step 3 Run **standarddialogs**.

```
./standarddialogs
```

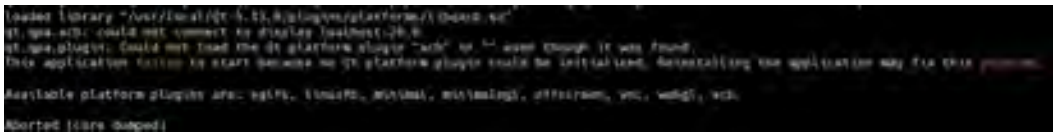
----End

4.71.8 Troubleshooting

4.71.8.1 Error Reported When Running the ./helloworld Command

Symptom

The following error information is displayed when running the **./helloworld** command:



Key Process and Cause Analysis

X11 forwarding is disabled by default in openEuler.

Conclusion and Solution

Step 1 Open the **/etc/ssh/sshd_config** file.

```
vi /etc/ssh/sshd_config
```

Step 2 Set the line number.

```
:set nu
```

Step 3 Press **i** to enter the insert mode. In line 100, delete **#** and change **no** to **yes**.

```
 97 #AllowAgentForwarding yes
 98 #AllowTcpForwarding yes
 99 #GatewayPorts no
100 X11Forwarding yes
101 #X11DisplayOffset 10
102 #X11UseLocalhost yes
103 #PermitTTY yes
104 PrintMotd no
105 #PrintLastLog yes
106 #TCPKeepAlive yes
107 #PermitUserEnvironment no
108 #Compression delayed
```

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Restart the sshd service.

```
systemctl restart sshd.service
```

Step 6 Open a new SSH connection.

----End

4.71.9 More Resources

Reference document:

https://bbs.huaweicloud.com/forum/thread-41064-1-1.html

4.72 RSeQC 4.0.0 Porting Guide (CentOS 8.2)

4.72.1 Introduction

RSeQC package provides a number of useful modules that can comprehensively evaluate high throughput sequence data especially RNA-seq data. Some basic modules quickly inspect sequence quality, nucleotide composition bias, PCR bias and GC bias, while RNA-seq specific modules evaluate sequencing saturation,

 Copyright © Huawei Technologies Co., Ltd.

mapped reads distribution, coverage uniformity, strand specificity, transcript level RNA integrity, and more.

For more information about RSeQC, visit the **RSeQC official website**.

Programming language: Python

Brief description: RSeQC package provides a number of useful modules that can comprehensively evaluate high throughput sequence data especially RNA-seq data.

Open source license: GPLv3

Recommended Software Version

RSeQC 4.0.0

4.72.2 Environment Requirements

Hardware Requirements

Table 4-352 lists the hardware requirements.

Table 4-352 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-353 lists the software requirements.

Table 4-353 Software requirements

Item	Version	Download Address
RseQC	4.0.0	**https://sourceforge.net/projects/rseqc/files/RSeQC-4.0.0.tar.gz**
Python	3.9.6	**https://www.python.org/downloads/release/python-396/**

OS Requirements

Table 4-354 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-354 OS requirements

Item	Version	Download Address
CentOS	8.2	**https://www.centos.org/download/**
Kernel	4.18.0-193.el8.aarch64	Included in the OS image

4.72.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Python software porting.

Table 4-355 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/PYTHON	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.72.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configure Process

Table 4-356 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Python	See **4.72.4.1 Installing Python**.

4.72.4.1 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Python installation package.

```
wget https://www.python.org/ftp/python/3.9.6/Python-3.9.6.tgz
```

Step 3 Decompress the package.

```
tar -xvf Python-3.9.6.tgz
```

Step 4 Install the dependencies.

```
yum install gcc libffi* openssl libssl zlib* bzip2
```

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/PYTHON  --enable-shared --enable-optimizations
```

Step 6 Perform the compilation and installation.

```
make
make install
```

Step 7 Configure the environment variables.

```
export PATH=/path/to/PYTHON:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON:$ LD_LIBRARY_PATH
```

Step 8 Verify the version.

```
python3 --version
pip3 --version
```

If a version number is displayed in the command output, the installation is successful.

----End

4.72.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Compile and install RSeQC.

```
pip3 install RSeQC
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

By default, it is installed in a subdirectory of Python.

----End

4.72.6 Running and Verification

Procedure

Step 1 Check for the 4.0.0 version information.

```
bam2fq.py --version
```

```
[root@localhost python3]# bam2fq.py --version
bam2fq.py 4.0.0
[root@localhost python3]#
```

- If the preceding information is displayed, the installation is successful.
- Otherwise, perform **4.72.4 Configuring the Compilation Environment** to **4.72.5 Compilation and Installation** to perform the installation again or contact Huawei technical support.

----End

4.73 Sniffles 1.0.12 Porting Guide (CentOS 7.6)

4.73.1 Introduction

Sniffles2 is a fast structural variant caller for long-read sequencing. It accurately detects SVs on germline, somatic and population-level for PacBio and Oxford Nanopore read data.

For more information about Sniffles, visit the **Sniffles official website**.

Programming languages: C/C++

Brief description: Sniffles2 is a fast structural variant caller for long-read sequencing.

Open source license: custom open source license

Recommended Software Version

Sniffles 1.0.12

4.73.2 Environment Requirements

Hardware Requirements

Table 4-357 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-357 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-358 lists the software requirements.

Table 4-358 Software requirements

Item	Version	Download Address
Sniffles	1.0.12	https://github.com/fritzsedlazeck/Sniffles/archive/master.tar.gz
CMake	3.18.2	They are installed using Yum and do not need to be manually downloaded.
GCC/GCC-C++	8.5.0	
libarchive	3.3.3	
zlib/zlib-devel/zlib-static	1.2.11	

OS Requirements

Table 4-359 lists the OS requirements.

Table 4-359 OS requirements

Item	Version	Download Address
CentOS	7.6	https://www.centos.org/download/
Kernel	4.18.0-193.el8.aarch64	Included in the OS image

4.73.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Sniffles software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-360 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/SNIFFLES	Installation path of Sniffles.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.73.4 Configuring the Compilation Environment

4.73.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-361 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Configuring the Yum source	See **4.2**.
3	Installing the dependencies	See **4.73.4.3 Installing the Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

4.73.4.2 Configuring the Yum Source

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Configure the Yum source.

1. Open the **CentOS-Base.repo** file.
   ```
   vi /etc/yum.repos.d/CentOS-Base.repo
   ```
2. Press **i** to enter the insert mode. Delete the existing content and add the following content:
   ```
   [BaseOS]
   name=CentOS-$releasever - Base
   mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=BaseOS&infra=
   $infra
   #baseurl=http://mirror.centos.org/$contentdir/$releasever/BaseOS/$basearch/os/
   gpgcheck=1
   enabled=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Clear the cache.

```
yum clean all
```

Step 4 Refresh the cache.

```
yum makecache
```

----End

4.73.4.3 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y epel-release
yum install -y gcc
yum install -y gcc-c++
yum install -y cmake
yum install -y make
yum install -y zlib*
yum install -y libarchive
```

Step 3 Verify the dependencies.

```
gcc --version
cmake --version
```

If a version number is displayed in the command output, the installation is successful.

----End

4.73.5 Obtaining the Source Code

Procedure

Step 1 Download the Sniffles installation package.

https://github.com/fritzsedlazeck/Sniffles/archive/master.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the CESM installation package to the **/path/to/SNIFFLES** directory on the server.

----End

4.73.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
wget https://github.com/fritzsedlazeck/Sniffles/archive/master.tar.gz -O Sniffles.tar.gz
```

Step 3 Decompress the installation package.

```
tar -zxvf Sniffles.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd Sniffles-master
```

Step 5 Create a **build** folder and switch to the folder.

```
mkdir build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake ../
make install
```

----End

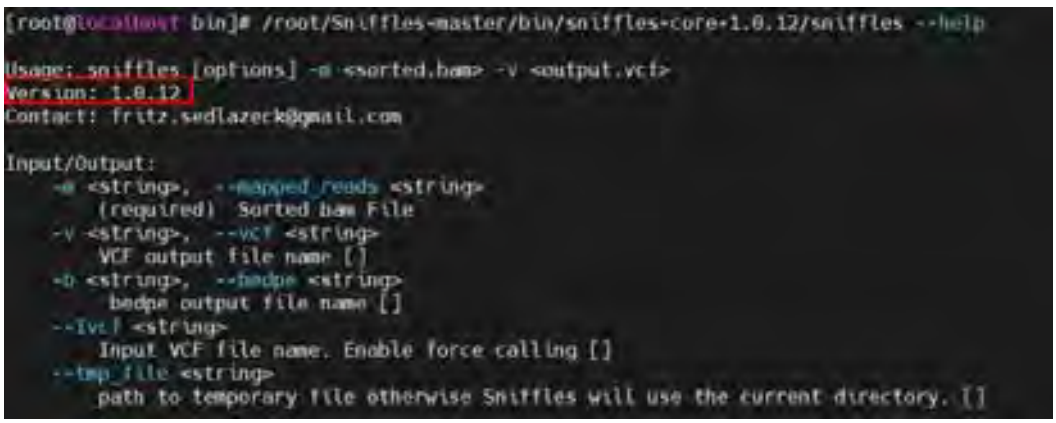
4.73.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check for the following version information:

```
find / -name "sniffles"
/root/Sniffles-master/bin/sniffles-core-1.0.12/sniffles –help
```



- If the preceding information is displayed, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.

- Otherwise, perform **4.73.4 Configuring the Compilation Environment** to **4.73.6 Compilation and Installation** to perform the installation again or contact Huawei technical support.

----End

4.73.8 More Resources

Sniffles installation guide:

https://www.bbsmax.com/A/RnJWOGVvJq/

4.74 SPIDER 24.08 Porting Guide (CentOS 7.6)

4.74.1 Introduction

SPIDER (System for Processing Image Data from Electron microscopy and Related fields) is an image processing system for electron microscopy, especially useful for single-particle reconstruction. SPIDER has been in use since 1978 and contains over 210,000 lines of Fortran code and 7,400 files. The latest release from July 2020 (which will probably be the final release) contains support for reading/writing MRC files. It is mainly used in averaging of single particle objects, multivariate statistical analysis, 3D reconstruction, electron tomography, and object segmentation from volumes.

This release consists of the spider and web modules, and contains all related source code (mostly released under the **GPLv2** license) and some precompiled executable files.

The SPIDER module is used for computing that does not require graphical interaction or visualization. The Web module is used for operations involving graphical interaction with data, for example, displaying an image file. It has a click-through interface using the Motif/X-Window system or Java (JWeb).

For more information about VEP, visit the **SPIDER official website**.

Programming language: Fortran

Brief description: SPIDER is an image processing system for electron microscopy, especially useful for single-particle reconstruction.

Open source license: GPLv2

Recommended Software Version

SPIDER 24.08

4.74.2 Environment Requirements

Hardware Requirements

Table 4-362 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-362 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-363 lists the software requirements.

Table 4-363 Software requirements

Item	Version	Download Address
SPIDER	24.08	https://github.com/spider-em/SPIDER/releases/download/24.08/spiderweb.24.08.tar.gz
GNU	4.8.5	The software is provided by the system and does not need to be downloaded.
FFTW	3.3.8	http://www.fftw.org/fftw-3.3.8.tar.gz

OS Requirements

Table 4-364 lists the OS requirements.

Table 4-364 OS requirements

Item	Version	Download URL
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image

4.74.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SPIDER software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-365 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/SPIDER	Installation path of SPIDER.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.74.4 Configuring the Compilation Environment

4.74.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-366 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing FFTW	See **4.74.4.2 Installing FFTW**.

4.74.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the FFTW installation package.

```
tar -zxvf fftw-3.3.8.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Go to the directory generated after the decompression.

```
cd fftw-3.3.8
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/FFTW--enable-shared --enable-static --enable-fma --enable-neon --enable-float
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Configure the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/FFTW/include:$INCLUDE
```

----End

4.74.5 Obtaining the Source Code

Procedure

Step 1 Download the SPIDER installation package.

https://github.com/spider-em/SPIDER/releases/download/24.08/spiderweb.24.08.tar.gz

Step 2 Use an SFTP tool to upload the SPIDER installation package to the */path/to/SPIDER* directory on the server.

----End

4.74.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the SPIDER installation directory.

```
cd /path/to/SPIDER
```

Step 3 Decompress the installation package.

```
tar -xvf spiderweb.24.08.tar.gz
```

Step 4 Go to the web module directory generated after the decompression.

```
cd web/src
```

Step 5 Install the dependencies.

```
yum install motif libXp libXt libX11 libSM libxcb libXau libXp-devel motif-devel
```

NOTE

If there is no output after you run the **echo $DISPLAY** command, open the shell window again.

Step 6 Compile the web module.

```
make -f Makefile_linux_dir64
```

 Copyright © Huawei Technologies Co., Ltd.

Step 7 Go to the SPIDER module directory generated after the decompression.

```
cd ../../spider/src
```

Step 8 Modify the configuration file. For details, see **Table 4-367**.

Table 4-367 Modifying the configuration file

File Name	Procedure

star2doc.f	1. Open the **star2doc.f** file.
	`vi star2doc.f`
	2. Set the line number.
	`:set nu`
	3. Press **i** to enter the insert mode. Modify line 358 as follows:
	`358 READ(CTOKEN(1:NCHAR),*) IDLIST`
	4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

pw3sr.f	1. Open the **pw3sr.f** file.
	`vi pw3sr.f`
	2. Set the line number.
	`:set nu`
	3. Press **i** to enter the insert mode. Modify lines 37 and 38 as follows:
	`37 INTEGER :: NX,NY,NZ`
	`38 REAL :: Q(NX+2-MOD(NX,2),NY,NZ)`
	4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

pw2sr.f	1. Open the **pw2sr.f** file.
	`vi pw2sr.f`
	2. Set the line number.
	`:set nu`
	3. Press **i** to enter the insert mode. Modify lines 241 and 242 as follows:
	`241 INTEGER :: NX,NY`
	`242 REAL :: Q(NX+2-MOD(NX,2), NY)`
	4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

pr3d.f	1. Open the **pr3d.f** file.
	`vi pr3d.f`
	2. Set the line number.
	`:set nu`
	3. Press **i** to enter the insert mode. Modify line 88 as follows:
	`88 WANTSQRTS = (FSCOP .eqv. .TRUE.)`
	4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

File Name	Procedure
Makefile_linux.gfort	1. Open the **Makefile_linux.gfort** file. **vi Makefile_linux.gfort** 2. Set the line number. **:set nu** 3. Press **i** to enter the insert mode. Modify line 39 as follows: 39 FFTWLIBDIR = path/to/fftw/lib 4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Compile the SPIDER module.

```
yum install gcc-gfortran.aarch64
make -f Makefile_linux.gfort
```

----End

4.74.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **/web/bin** directory.

cd */path/to/SPIDER*/**web/bin**

Step 3 Start the web module of SPIDER.

```
./web_linux_dir64_tmp
```

NOTE

If the preceding information is displayed, the web module is started successfully.

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Go to the **/spider/bin** directory.

```
cd /path/to/SPIDER/spider/bin
```

Step 5 Start the SPIDER module of SPIDER.

```
./spider_linux_gfort
```

Enter the following content as prompted:

```
.PROJECT/DATA EXTENSION: PRJ/DAT
.OPERATION: ((3-0.26)/4)**2
```

If the command output is **0.46922499**, the software is running properly.

----End

4.75 VEP 104.3 Porting Guide (CentOS 7.6)

4.75.1 Introduction

Variant Effect Predictor (VEP) is a freely available, open source tool for the annotation and filtering of genomic variants. VEP determines the effect of your variants (SNPs, insertions, deletions, CNVs, or structural variants) on genes, transcripts, and protein sequence, as well as regulatory regions.

For more information about VEP, visit the **VEP official website**.

Programming language: Perl

Brief description: VEP is a freely available, open source tool for the annotation and filtering of genomic variants.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

VEP 104.3

4.75.2 Environment Requirements

Hardware Requirements

Table 4-368 lists the hardware requirements.

Table 4-368 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-369 lists the software requirements.

Table 4-369 Software requirements

Item	Version	Download Address
VEP	104.3	https://github.com/Ensembl/ensembl-vep/archive/refs/tags/release/104.3.tar.gz
GNU	9.3.0	https://codeload.github.com/gcc-mirror/gcc/tar.gz/releases/gcc-9.3.0
GMP	6.1.0	http://gcc.gnu.org/pub/gcc/infrastructure/
MPC	3.1.4	http://gcc.gnu.org/pub/gcc/infrastructure/
MPFR	1.0.3	http://gcc.gnu.org/pub/gcc/infrastructure/
htslib	1.14	https://github.com/samtools/htslib/releases/download/1.14/htslib-1.14.tar.bz2
homo_sapines	GRCh38	http://ftp.ensembl.org/pub/release-104/variation/indexed_vep_cache/homo_sapiens_vep_104_GRCh38.tar.gz

OS Requirements

Table 4-370 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-370 OS requirements

Item	Version	Download URL
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image

4.75.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VEP software porting.

Table 4-371 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HTSLIB	Installation path of HTSlib.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/VEP	Installation path of VEP.	

4.75.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-372 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing HTSlib	See **4.75.4.1 Installing HTSlib**.

4.75.4.1 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HTSlib installation package.

```
tar -zxvf htslib-1.14.tar.bz2
```

Step 3 Go to the directory generated after the decompression.

```
cd htslib-1.14
```

Step 4 Perform the configuration.

```
./configure --prefix=/path/to/HTSLIB --disable-bz2
```

Step 5 Perform the compilation and installation.

```
make -j
make install
```

Step 6 Configure the environment variables.

```
export HTSLIB_DIR=/path/to/HTSLIB
export PATH=/path/to/HTSLIB/bin/:$PATH
export LD_LIBRARY=/path/to/HTSLIB/lib/:$LD_LIBRARY
```

----End

4.75.5 Obtaining the Source Code

Procedure

Step 1 Download the VEP installation package.

https://github.com/Ensembl/ensembl-vep/archive/refs/tags/release/104.3.tar.gz

Step 2 Use an SFTP tool to upload the VEP installation package to the */path/to/VEP* directory on the server.

----End

4.75.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar -zxvf 104.3.tar.gz
```

Step 3 Go to the directory generated after the decompression.

 Copyright © Huawei Technologies Co., Ltd.

```
cd ensembl-vep-release-104.3
```

Step 4 Install and use the cpanm tool to download the dependency package.

```
yum install -y perl perl-App-cpanminus.noarch perl-core perl-DBD-MySQL perl-Try-Tiny.noarch
cpanm Archive::Zip
cpanm DBI
cpanm Bio::Root::Version
cpanm Bio::DB::HTS::Tabix
cpanm HTTP::Tiny
```

Step 5 Perform the compilation and installation.

```
perl INSTALL.pl  --NO_HTSLIB
```

Select **n** for the three options in the command output.

```
Do you want to install any cache files (y/n)? n
Skipping cache installation

The VEP can use FASTA files to retrieve sequence data for HGVS notations and reference sequence checks.
FASTA files will be stored in /root/.vep
Do you want to install any FASTA files (y/n)? n
Skipping FASTA installation - Exiting

The VEP can use plugins to add functionality and data.
Plugins will be installed in /root/.vep/Plugins
Do you want to install any plugins (y/n)? n
Skipping plugin installation
```

Step 6 (Optional) Download the cache file.

```
cd /root/.vep
wget http://ftp.ensembl.org/pub/release-104/variation/indexed_vep_cache/
homo_sapiens_vep_104_GRCh38.tar.gz --no-check-certificate
tar homo_sapiens_vep_104_GRCh38.tar.gz
```

----End

4.75.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test directory and switch to the directory.

```
mkdir -p /path/to/ ensembl-vep-release-104.3
cd /path/to/ ensembl-vep-release-104.3
```

Step 3 Start VEP.

```
./vep -i examples/homo_sapiens_GRCh38.vcf -o output.txt –cache
```

NOTE

If **ensembldb.ensembl.org** cannot be connected, download the cache file (see **Step 6**) and use the offline mode.

```
./vep -i examples/homo_sapiens_GRCh38.vcf -o output.txt –cache
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@Malluma ensembl-vep-release-104.3]# ll
total 656
drwxr-xr-x. 45 root root   4096 Nov 27 14:51 Bio
drwxr-xr-x.  7 root root   4096 Nov 27 11:59 biodbhts
-rwxrwxr-x.  1 root root  16608 May 26  2021 convert_cache.pl
-rw-rw-r--.  1 root root    279 May 26  2021 cpanfile
drwxrwxr-x.  2 root root     32 May 26  2021 docker
drwxrwxr-x.  2 root root   4096 May 26  2021 examples
-rwxrwxr-x.  1 root root  16620 May 26  2021 filter_vep
-rwxrwxr-x.  1 root root   4880 May 26  2021 haplo
drwxr-xr-x.  8 root root   4096 Nov 27 11:59 htslib
-rw-r--r--.  1 root root      5 Dec  1 09:09 input.txt
-rwxrwxr-x.  1 root root  56607 May 26  2021 INSTALL.pl
-rw-rw-r--.  1 root root  11478 May 26  2021 LICENSE
drwxrwxr-x.  3 root root     25 May 26  2021 modules
-rw-r--r--.  1 root root 221455 Dec  1 09:40 output.txt
-rw-r--r--.  1 root root  11452 Dec  1 09:40 output.txt_summary.html
-rw-rw-r--.  1 root root  13941 May 26  2021 README.md
drwxrwxr-x.  4 root root   4096 May 26  2021 t
drwxrwxr-x.  2 root root    198 May 26  2021 travisci
drwxrwxr-x.  2 root root     78 May 26  2021 validator
-rw-r--r--.  1 root root 221455 Dec  1 09:35 variant_effect_output.txt
-rw-r--r--.  1 root root  11442 Dec  1 09:35 variant_effect_output.txt_summary.html
-rw-r--r--.  1 root root   1441 Nov 27 14:52 variant_effect_output.txt_warnings.txt
-rwxrwxr-x.  1 root root   5141 May 26  2021 variant_recoder
-rwxrwxr-x.  1 root root  14462 May 26  2021 vep
[root@Malluma ensembl-vep-release-104.3]# ll /root/
```

NOTE

If an **output.txt** file is generated, the execution is successful.

Step 4 View the **html** file.

----End

4.76 VTK 9.0.1 Porting Guide (openEuler 20.03)

4.76.1 Introduction

The Visualization Toolkit (VTK) is an open-source software system for image processing, 3D graphics, volume rendering and visualization. VTK includes many advanced algorithms (e.g., surface reconstruction, implicit modeling, decimation) and rendering techniques (e.g., hardware-accelerated volume rendering, LOD control). VTK is used by academicians for teaching and research and by many commercial firms who use VTK to build or extend products.

For more information, visit the **official VTK website**.

Programming language: C++

Brief description: Visualization Toolkit (VTK) is open source software for manipulating and displaying scientific data.

Open source license: BSD

Recommended Software Version

VTK 9.0.1

4.76.2 Environment Requirements

Hardware Requirements

Table 4-373 lists the hardware requirements.

Table 4-373 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-374 lists the software requirements.

Table 4-374 Software requirements

Item	Version	Download URL
GCC for openEuler	9.3	**https://www.hikunpeng.com/document/detail/en/kunpengdevps/compiler/ug-hgcc/kunpenghgcc_06_0001.html**
VTK	9.0.1	**https://www.vtk.org/files/release/9.0/VTK-9.0.1.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
CMake	3.16.5	https://cmake.org/files/v3.16/cmake-3.16.5.tar.gz

OS Requirements

Table 4-375 lists the OS requirements.

Table 4-375 OS requirements

Item	Version	Download URL
openEuler	20.03	https://docs.openeuler.org/en/docs/20.03_LTS/docs/Installation/installation-guideline.html

4.76.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VTK software porting.

Table 4-376 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler.	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/VTK	Installation path of VTK.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.76.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-377 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler compiler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.76.5 Obtaining the Source Code

Procedure

Step 1 Download the VTK installation package.

https://www.vtk.org/files/release/9.0/VTK-9.0.1.tar.gz

Step 2 Use an SFTP tool to upload the VTK installation package to the */path/to/VTK* directory on the server.

----End

4.76.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the VTK installation package.

```
cd /path/to/VTK
tar -zxvf VTK-9.0.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd VTK-9.0.1
```

Step 4 Perform the compilation.

```
mkdir build
cd build
cmake .. -D CMAKE_INSTALL_PREFIX=/usr/local/VTK-9.0.1
```

Step 5 Perform the installation.

```
make -j48
make install
```

Step 6 Check whether the installation is successful.

```
ls -l /usr/local
```

 Copyright © Huawei Technologies Co., Ltd.

If the command output contains the VTK version information, the installation is successful.

----End

4.76.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether GCC for openEuler has been loaded.

```
gcc -v
```

If the command output contains the GCC for openEuler version information, GCC for openEuler has been loaded.

Step 3 Running VTK requires invoking the library. You can use an example test to check whether VTK can be run. You can run the **git clone** command to download it.

```
git clone https://github.com/lorensen/VTKExamples.git
```

Step 4 Go to the following directory:

```
cd VTKExamples
cd build
```

Step 5 Perform the compilation.

```
cmake .. -DVTK_DIR:PATH=/usr/local/VTK-9.0.1 -DBUILD_TESTING:BOOL=ON
make
```

Step 6 Test the generated test case file.

```
ctest
```

```
-- VTKWikiExamples: PineRootConnectivity requires git lfs to be installed and will not be built
-- VTKWikiExamples: PineRootConnectivityA requires git lfs to be installed and will not be built
-- VTKWikiExamples: PineRootDecimation requires git lfs to be installed and will not be built
-- VTKWikiExamples: DecimateHawaii requires git lfs to be installed and will not be built
-- VTKWikiExamples: Not building Qt examples
-- VTKWikiExamples: git lfs is not installed. Some tests will not be created.
-- Configuring done
-- Generating done
-- Build files have been written to: /data/VTK/VTKExamples/build
```

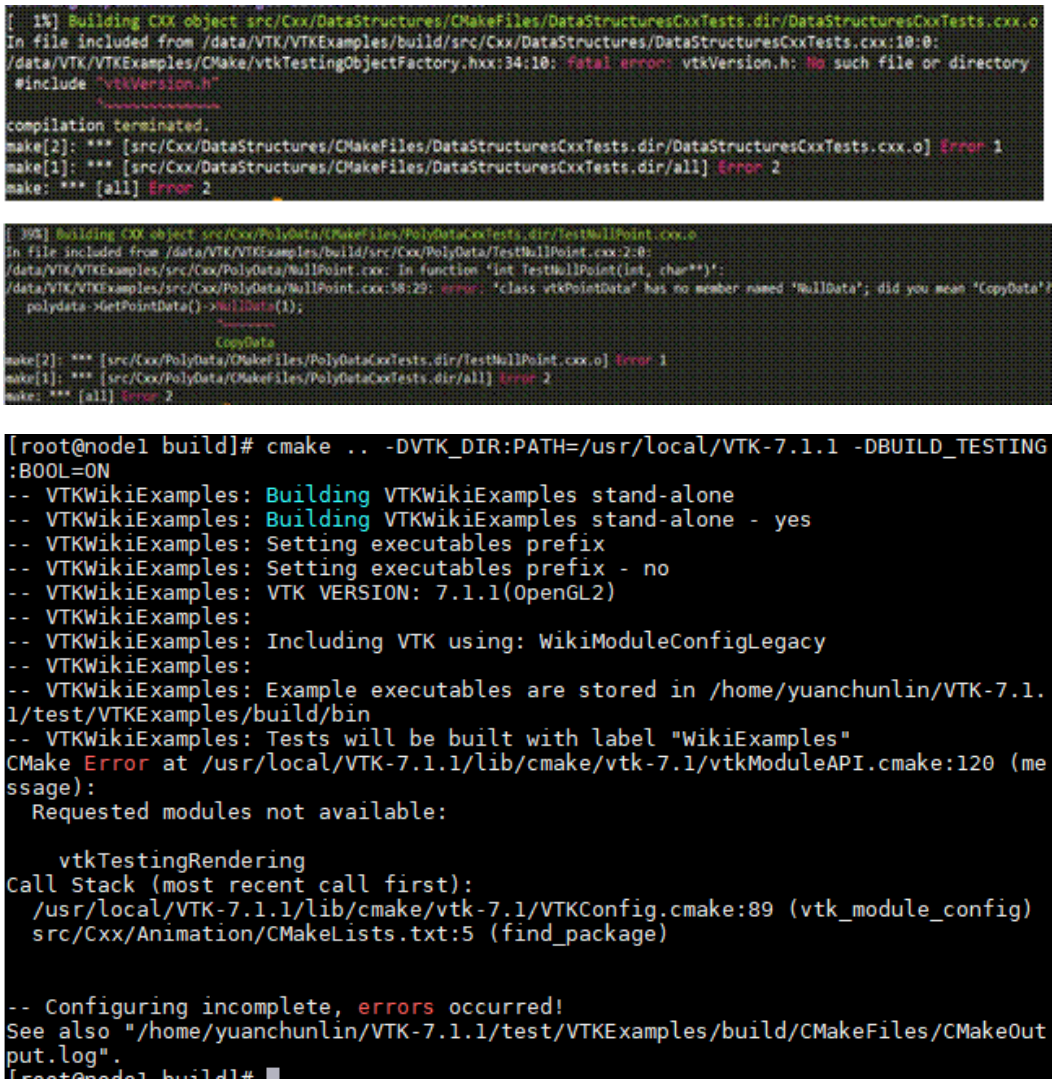
----End

4.76.8 Troubleshooting

4.76.8.1 Error Reported During Test Case Compilation

Symptom

The **vtkVersion.h: No such file or directory** error is reported during test case compilation.



Key Process and Cause Analysis

Some test cases are not available in version 9.0.1.

Conclusion and Solution

Mask those test cases as follows:

Modify **VTKExamples/CMakeLists.txt** to mask the following modules:

 Copyright © Huawei Technologies Co., Ltd.

```
add_subdirectory(src/Cxx/Animation)
add_subdirectory(src/Cxx/CompositeData)
if(VTK_USE_MYSQL)
  add_subdirectory(src/Cxx/Databases)
endif()
# add_subdirectory(src/Cxx/DataStructures)
if(${VTK_VERSION} VERSION_GREATER "6.10")
  add_subdirectory(src/Cxx/Developers)
endif()
#add_subdirectory(src/Cxx/Filtering)
add_subdirectory(src/Cxx/GeometricObjects)
if(VTK_USE_INFOVIS)
  if(VTK_USE_BOOST)
    add_subdirectory(src/Cxx/Graphs)
  endif()
endif()
add_subdirectory(src/Cxx/Annotation)
add_subdirectory(src/Cxx/ImageData)
add_subdirectory(src/Cxx/Geovis)
add_subdirectory(src/Cxx/HyperTreeGrid)
add_subdirectory(src/Cxx/ImageProcessing)
add_subdirectory(src/Cxx/Images)
#add_subdirectory(src/Cxx/ImplicitFunctions)
#add_subdirectory(src/Cxx/InfoVis)
#add_subdirectory(src/Cxx/Interaction)
#add_subdirectory(src/Cxx/IO)
add_subdirectory(src/Cxx/Lighting)
#add_subdirectory(src/Cxx/Math)
add_subdirectory(src/Cxx/Medical)
#add_subdirectory(src/Cxx/Meshes)
add_subdirectory(src/Cxx/Modelling)
if(VTK_USE_PARALLEL)
  add_subdirectory(src/Cxx/Parallel)
endif()
#add_subdirectory(src/Cxx/Picking)
#add_subdirectory(src/Cxx/Plotting)
#add_subdirectory(src/Cxx/PolyData)
add_subdirectory(src/Cxx/Points)
add_subdirectory(src/Cxx/RectilinearGrid)
#add_subdirectory(src/Cxx/Remote)
#add_subdirectory(src/Cxx/Rendering)
#add_subdirectory(src/Cxx/RenderMan)
add_subdirectory(src/Cxx/SimpleOperations)
add_subdirectory(src/Cxx/Shaders)
add_subdirectory(src/Cxx/StructuredGrid)
add_subdirectory(src/Cxx/StructuredPoints)
#add_subdirectory(src/Cxx/Texture)
add_subdirectory(src/Cxx/UnstructuredGrid)
add_subdirectory(src/Cxx/Utilities)
add_subdirectory(src/Cxx/Video)
#if(VTK_USE_VIEWS)
#  add_subdirectory(src/Cxx/Views)
#endif()
#add_subdirectory(src/Cxx/Visualization)
add_subdirectory(src/Cxx/VisualizationAlgorithms)
#add_subdirectory(src/Cxx/VolumeRendering)
add_subdirectory(src/Cxx/Widgets)
```

4.76.9 More Resources

Reference document:

http://ic-openlabs.huawei.com/openlab/#/knowledgebasequery?task_id=R5S1584D116053085202007221018289551313340

 Copyright © Huawei Technologies Co., Ltd.

4.77 BWA 0.7.17 Porting Guide (openEuler 21.03)

4.77.1 Introduction

Burrows-Wheeler Aligner (BWA) is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW, and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100 bp. The rest two are for longer sequences ranged from 70 bp to 1 Mbp. BWA-MEM and BWA-SW share similar features, such as long-read support and split alignment. However, BWA-MEM is generally recommended for high-quality queries as it is faster and more accurate. Compared with BWA-backtrack, BWA-MEM has better performance in 70–100 bp Illumina reads.

For more information about BWA, visit the **BWA official website**.

Programming language: C

Brief description: gene comparison.

Open source license: GPL 3.0

Recommended Software Version

BWA 0.7.17

4.77.2 Environment Requirements

Hardware Requirements

Table 4-378 lists the hardware requirements.

Table 4-378 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-379 lists the software requirements.

Table 4-379 Software requirements

Item	Version	Download URL
BWA	0.7.17	**https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	1.3.3	**https://www.hikunpeng.com/en/developer/devkit/compiler?data=bisheng**
Test case file 1	ERR1044518_1.fastq	**https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_1.fastq.gz**
Test case file 2	ERR1044518_2.fastq	**https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_2.fastq.gz**
Test case file 3	hs38DH.fasta	**https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz** (Download the package, decompress it using **gzip -d**, and rename it **hs38DH.fasta**.)
SSE2NEON.h	-	**https://github.com/jratcliff63367/sse2neon.git**

OS Requirements

Table 4-380 lists the OS requirements.

Table 4-380 OS requirements

Item	Version	Download URL
openEuler	openEuler 21.03 LTS	**https://repo.openeuler.org/openEuler-21.03/**
Kernel	4.14.0-115	Included in the OS image

4.77.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BWA software porting.

Table 4-381 Paths for software porting

No.	Software Installation Path	Usage	Remarks
1	**/opt/compiler**	Installation path of the BiSheng Compiler.	See **Installing and Using BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Usage	Remarks
2	**/path/to/BWA**	Installation path of BWA.	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	**/path/to/CASE**	Installation path of test files.	

4.77.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-382 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing and Using BiSheng Compiler**.

Configuring the Environment

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Edit the environment variable file.

```
vi ~/.bashrc
```

Step 3 Press **i** to enter the edit mode and run the following commands to load the BiSheng Compiler and Hyper MPI environment variables.

```
export PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib:$LD_LIBRARY_PATH
```

NOTE

Replace the previous installation paths with the actual ones.

Step 4 Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the environment variables take effect.

```
source ~/.bashrc
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.77.5 Obtaining the Source Code

Procedure

Step 1 Download the BWA installation package.

https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2

Step 2 Use an SFTP tool to upload the BWA installation package to the */path/to/BWA* directory on the server.

----End

4.77.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the BWA installation package directory.

```
cd /path/to/BWA
```

Step 3 Decompress the BWA installation package.

```
tar -xvf bwa-0.7.17.tar.bz2
```

Step 4 Go to the directory generated after the decompression.

```
cd bwa-0.7.17
```

Step 5 Obtain the **SSE2NEON.h** file.

```
git clone https://github.com/jratcliff63367/sse2neon.git
```

Step 6 Copy **SSE2NEON.h** to the installation directory of the compiler.

```
cp SSE2NEON.h /opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib/clang/10.0.1/include
```

NOTE

You need to change the compiler path based on the site requirements, and add or adjust paths strictly according to the ones described in this document. Otherwise, an error is reported.

Step 7 Modify the **Makefile** file of BWA.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode and modify lines 1 and 14 in the **Makefile** file.
   ```
   CC=clang
   INCLUDES= -I/opt/compiler/bisheng-compiler-1.3.3-aarch64-linux/lib/clang/10.0.1/include
   ```

   NOTE

   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Modify the **ksw.c** file.

1. Open the **ksw.c** file.
   ```
   vi ksw.c
   ```

 Copyright © Huawei Technologies Co., Ltd.

2. Press **i** to enter the edit mode, remove the comment of line 29 in the **ksw.c** file (using **/* */**), and edit the following content:
```
#include <SSE2NEON.h>
```

NOTE

After performing **Step 8.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Perform the compilation.
```
make
```

Step 10 Load the BWA environment.
```
export PATH=/path/to/BWA/bwa-0.7.17:$PATH
```

Step 11 Check whether the compilation is successful.
```
bwa
```

The compilation is successful if the following information is displayed:
```
Usage:   bwa <command> [options]
```

----End

4.77.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the working directory.
```
cd /path/to/CASE
```

Step 3 Obtain the test case.
```
wget https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_1.fastq.gz
wget https://ftp.sra.ebi.ac.uk/vol1/fastq/ERR104/008/ERR1044518/ERR1044518_2.fastq.gz
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_full_analysis_set.fna.gz -O hs38DH.fasta
```

Step 4 Decompress the test case file.
```
gzip -d ERR1044518_1.fastq.gz
gzip -d ERR1044518_2.fastq.gz
```

Step 5 Create an index file.
```
bwa index -a bwtsw hs38DH.fasta
```

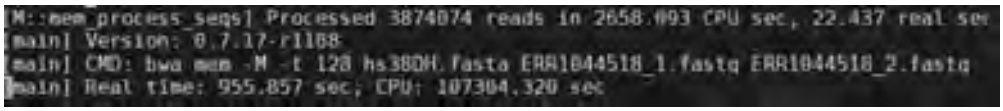
Step 6 Perform the test.
```
bwa mem -t 128 hs38DH.fasta ERR1044518_1.fastq ERR1044518_2.fastq > bwa.sam 2 > bwa_log
```

NOTE

The number **128** indicates the number of used cores. You can change the number according to the server configuration.

Check the value of **Real time** (unit: s) in the **bwa_log** log. A smaller value indicates higher performance.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.78 ABSOLUTE 1.0.6 Porting Guide (CentOS 7.6)

4.78.1 Introduction

ABSOLUTE can estimate purity/ploidy, and from that compute absolute copy-number and mutation multiplicities.

When DNA is extracted from an admixed population of cancer and normal cells, the information on absolute copy number per cancer cell is lost in the mixing. The purpose of ABSOLUTE is to re-extract these data from the mixed DNA population. This process begins by generation of segmented copy number data, which is input to the ABSOLUTE algorithm together with pre-computed models of recurrent cancer karyotypes and, optionally, allelic fraction values for somatic point mutations. The output of ABSOLUTE then provides re-extracted information on the absolute cellular copy number of local DNA segments and, for point mutations, the number of mutated alleles.

For more information about ABSOLUTE, visit the **ABSOLUTE official website**.

Programming language: R

Brief description: ABSOLUTE can estimate purity/ploidy, and from that compute absolute copy-number and mutation multiplicities.

Open source license: custom open source license

Recommended Software Version

ABSOLUTE 1.0.6

4.78.2 Environment Requirements

Hardware Requirements

Table 4-383 lists the hardware requirements.

Table 4-383 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-384 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-384 Software requirements

Item	Version	Download Address
ABSOLUTE	1.0.6	http://software.broadinstitute.org/cancer/cga/absolute_download

OS Requirements

Table 4-385 lists the OS requirements.

Table 4-385 OS requirements

Item	Version	Download URL
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image

4.78.3 Paths for Software Porting

This chapter lists software installation paths involved in the ABSOLUTE software porting.

Table 4-386 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/absolute	Installation path of ABSOLUTE.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

 Copyright © Huawei Technologies Co., Ltd.

4.78.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-387 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.78.5 Obtaining the Source Code

Procedure

Step 1 Log in to the official website, register an account, and download the ABSOLUTE installation package.

http://software.broadinstitute.org/cancer/cga/absolute_download

Step 2 Use an SFTP tool to upload the ABSOLUTE installation package to the */path/to/absolute* directory on the server.

----End

4.78.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation package directory.

```
cd /path/to/absolute
```

Step 3 Install R.

```
yum install R
```

Step 4 Add a cran image.

1. Create a **/root/.Rprofile** file.
   ```
   vi /root/.Rprofile
   ```
2. Press **i** to enter the edit mode and add the following content. If the file already exists, add it to the end of the file.
   ```
   options("repos" = c(CRAN="https://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 5 Start R.

 Copyright © Huawei Technologies Co., Ltd.

```
R
```

Step 6 Install numDeriv.

```
install.packages("numDeriv")
```

Step 7 Install ABSOLUTE.

```
install.packages("ABSOLUTE_1.0.6.tar.gz")
```

----End

4.78.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to R.

```
R
```

Step 3 Run ABSOULUTE.

```
library(ABSOLUTE)
RunAbsolute
```

```
> library(ABSOLUTE)
Loading required package: numDeriv
> RunAbsolute
function (seg.dat.fn, sigma.p, max.sigma.h, min.ploidy, max.ploidy,
    primary.disease, platform, sample.name, results.dir, max.as.seg.count,
    max.non.clonal, max.neg.genome, copy_num_type, maf.fn = NULL,
    min.mut.af = NULL, output.fn.base = NULL, verbose = FALSE)
{
```

----End

4.79 GDS3D 1.9 Porting Guide (CentOS 7.6)

4.79.1 Introduction

GDS3D is an application that can interpret so called IC layouts and render them in 3D. The program accepts standard GDSII files as input data. Along with the layout file, it requires a so-called process definition file which contains the 3D parameters of the process being used. These files combined allow the program to create a 3D representation of the layout, where the user has full, real-time control over the camera position and angle, much like in a 3D video game.

For more information about GDS3D, visit the **GDS3D official website**.

Programming languages: C/C++

Brief description: GDS3D is an application that can interpret so called IC layouts and render them in 3D.

Open source license: GPL-2.0

Recommended Software Version

GDS3D v1.9

 Copyright © Huawei Technologies Co., Ltd.

4.79.2 Environment Requirements

Hardware Requirements

Table 4-388 lists the hardware requirements.

Table 4-388 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-389 lists the software requirements.

Table 4-389 Software requirements

Item	Version	Download Address
GDS3D	1.9	https://github.com/trilomix/GDS3D.git

OS Requirements

Table 4-390 lists the OS requirements.

Table 4-390 OS requirements

Item	Version	Download URL
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image

4.79.3 Paths for Software Porting

This chapter lists software installation paths involved in the GDS3D software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-391 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/GDS3D	Installation path of GDS3D.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.79.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-392 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.79.5 Obtaining the Source Code

Procedure

Step 1 Download the GDS3D installation package.

https://github.com/trilomix/GDS3D.git

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the GDS3D installation package to the */path/to/GDS3D* directory on the server.

----End

4.79.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
git clone https://github.com/trilomix/GDS3D.git
```

Step 3 Go to the installation directory.

```
cd GSD3D /
```

Step 4 Install the dependencies.

```
yum install -y xcb* libxcb*
yum install -y libX11-devel libXext-devel libXtst-devel libxkbcommon-devel libXrender-devel libgcrypt-devel pciutils-devel nss-devel cups-devel pulseaudio-libs-devel libgudev1-devel systemd-devel libcap-devel alsa-lib-devel ruby libXrandr-devel libXcomposite-devel libXcursor-devel dbus-devel fontconfig-devel
yum install -y mesa* libglvnd-opengl freeglut libGLEW
```

Step 5 Perform the compilation.

```
make -C linux
```

----End

4.79.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

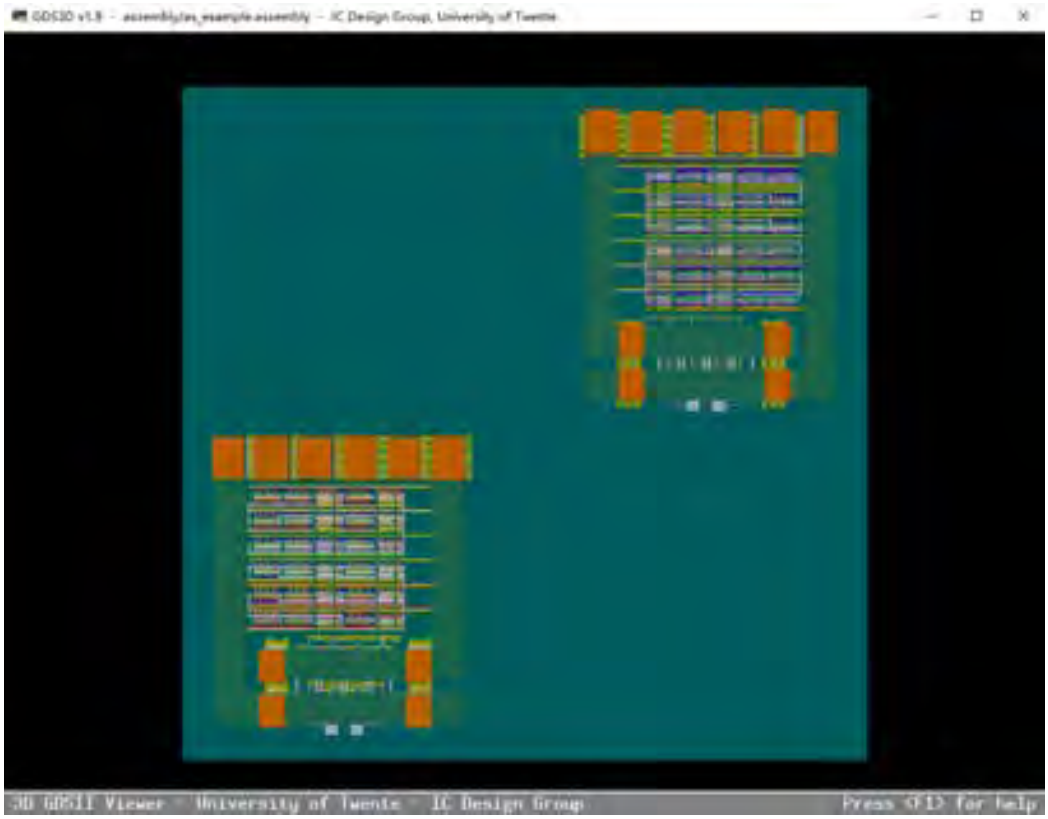
Step 2 Go to the */path/to/GDS3D* directory.

```
cd /path/to/GDS3D
```

Step 3 Run GDS3D.

```
./linux/GDS3D -a assembly/as_example.assembly
```

 Copyright © Huawei Technologies Co., Ltd.



----End

4.80 HOMER 4.11 Porting Guide (CentOS 8.2)

4.80.1 Introduction

Hypergeometric Optimization of Motif EnRichment (HOMER) is a suite of tools for Motif Discovery and next-gen sequencing analysis. It is a collection of command line programs for UNIX-style operating systems written in Perl and C++. HOMER was primarily written as a de novo motif discovery algorithm and is well suited for finding 8-20 bp motifs in large scale genomics data. HOMER contains many useful tools for analyzing ChIP-Seq, GRO-Seq, RNA-Seq, DNase-Seq, Hi-C and numerous other types of functional genomics sequencing data sets.

For more information about HOMER, visit the **HOMER official website**.

Programming languages: Perl, C++

Brief description: HOMER is a collection of command line programs for UNIX-style operating systems written in Perl and C++.

Open source license: custom open source license

Recommended Software Version

HOMER 4.11

 Copyright © Huawei Technologies Co., Ltd.

4.80.2 Environment Requirements

Hardware Requirements

Table 4-393 lists the hardware requirements.

Table 4-393 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-394 lists the software requirements.

Table 4-394 Software requirements

Item	Version	Download Address
Perl	5.26	Contained in the OS

OS Requirements

Table 4-395 lists the OS requirements.

Table 4-395 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.80.3 Paths for Software Porting

This chapter lists software installation paths involved in the HOMER software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-396 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/homer	Installation path of HOMER.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.80.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-397 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.80.5 Obtaining the Source Code

Procedure

Step 1 Download the HOMER installation file.

http://homer.ucsd.edu/homer/configureHomer.pl

Step 2 Use an SFTP tool to upload the installation file to the */path/to/homer* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.80.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/homer
```

Step 3 Start the installation.

```
perl ./configureHomer.pl -install
```

The following files are generated in the **configureHomer.pl** directory:

Step 4 Add the environment variables.

```
export PATH=/path/to/homer/bin:$PATH
```

----End

4.80.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/homer
```

Step 3 Download the test file.

```
wget https://phoenixnap.dl.sourceforge.net/project/rseqc/BED/Human_Homo_sapiens/hg19_RefSeq.bed.gz
```

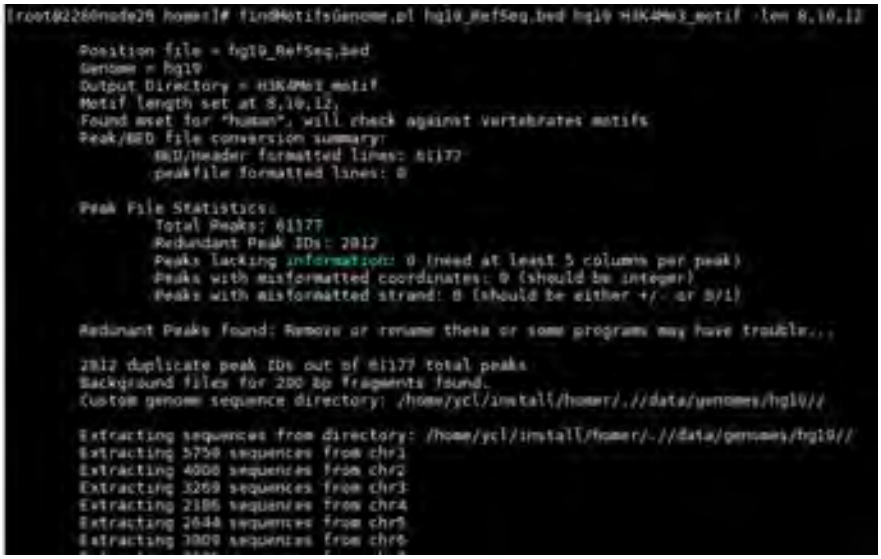
Step 4 Decompress the test file.

```
gzip -d hg19_RefSeq.bed.gz
```

Step 5 Perform the test.

```
findMotifsGenome.pl hg19_RefSeq.bed hg19 H3K4Me3_motif -len 8,10,12
```

The test takes approximately 100 minutes.



 Copyright © Huawei Technologies Co., Ltd.

The test result is stored in this directory:

```
[root@2280node26 homer]# ll H3K4Me3_motif/
total 692
-rw-r--r-- 1 root root      0 Jan 17 14:08 0.706096721886755.tmp
-rw-r--r-- 1 root root  30004 Jan 17 15:45 homerMotifs.all.motifs
-rw-r--r-- 1 root root   9997 Jan 17 15:09 homerMotifs.motifs10
-rw-r--r-- 1 root root  11288 Jan 17 15:45 homerMotifs.motifs12
-rw-r--r-- 1 root root   8719 Jan 17 15:01 homerMotifs.motifs8
drwxr-xr-x 2 root root  12288 Jan 17 15:50 homerResults
-rw-r--r-- 1 root root 218533 Jan 17 15:50 homerResults.html
drwxr-xr-x 2 root root   8192 Jan 17 14:58 knownResults
-rw-r--r-- 1 root root 327218 Jan 17 14:58 knownResults.html
-rw-r--r-- 1 root root  49356 Jan 17 14:58 knownResults.txt
-rw-r--r-- 1 root root     54 Jan 17 14:33 motifFindingParameters.txt
-rw-r--r-- 1 root root   1888 Jan 17 14:34 seq.autonorm.tsv
[root@2280node26 homer]#
```

----End

4.81 mCaller 1.0.3 Porting Guide (CentOS 8.2)

4.81.1 Introduction

The mCaller program is designed to call m6A from nanopore data using the differences between measured and expected currents.

For more information about mCaller, visit the **mCaller official website**.

Programming language: Python

Brief description: mCaller is designed to call m6A from nanopore data using the differences between measured and expected currents.

Open source license: MIT License

Recommended Software Version

mCaller v1.0.3

4.81.2 Environment Requirements

Hardware Requirements

Table 4-398 lists the hardware requirements.

Table 4-398 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-399 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-399 Software requirements

Item	Version	Download Address
Python	3.7.5	https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz

OS Requirements

Table 4-400 lists the OS requirements.

Table 4-400 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.81.3 Paths for Software Porting

This chapter lists software installation paths involved in the mCaller software porting.

Table 4-401 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/usr/local/python3.7.5	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/mCaller	Installation path of mCaller.	

4.81.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configure Process

Table 4-402 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the dependencies	See **4.81.4.1 Installing the Dependencies**.
3	Installing Python	See **4.81.4.2 Installing Python**.

4.81.4.1 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y epel-release gcc gcc-c++ cmake make libX11 libX11-devel
```

----End

4.81.4.2 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the required dependencies.

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel
```

Step 3 Download and decompress the Python 3 installation package.

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
tar -zxvf Python-3.7.5.tgz
```

Step 4 Go to the installation directory.

```
cd Python-3.7.5
```

Step 5 Perform the compilation and installation.

```
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
make install
```

Step 6 Set the soft links.

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.81.5 Obtaining the Source Code

Procedure

Step 1 Download the mCaller package.

https://github.com/al-mcintyre/mCaller

Step 2 Use an SFTP tool to upload the mCaller package to the */path/to/mCaller* directory on the server.

----End

4.81.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
pip3 install -i https://repo.huaweicloud.com/repository/pypi/simple wheel
pip3 install -i https://repo.huaweicloud.com/repository/pypi/simple scikit-learn h5py biopython matplotlib seaborn numpy pysam scipy pandas
```

Step 3 Add the folder to the path of the Python 3 running module.

```
python3
import sys
sys.path.append('/path/to/mCaller')
```

----End

4.81.7 Running and Verification

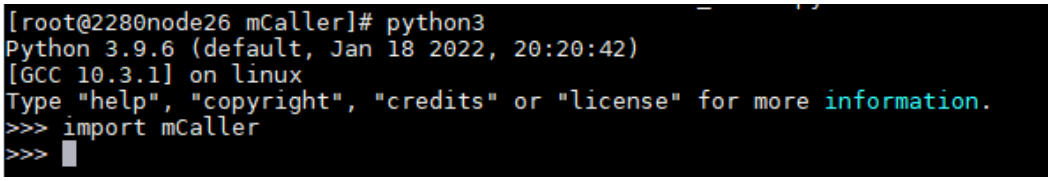
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Import mCaller.

```
python3
import mCaller
```

If no error is displayed in the command output, the import is successful.



Step 3 Run mCaller. (The following is an example.)

```
cd /path/to/mCaller
python3 mCaller.py -p testdata/test_positions.txt -r testdata/pb_ecoli_polished_assembly.fasta -e testdata/masonread1.eventalign.tsv -t 4 --train -f testdata/masonread1.fastq
```

 Copyright © Huawei Technologies Co., Ltd.

python3 make_bed.py -f testdata/masonread1.eventalign.diffs.6 -d 1 -t 0.5

----End

4.82 Muscle 5.0.1428 Porting Guide (CentOS 7.9)

4.82.1 Introduction

MUltiple Sequence Comparison by Log-Expectation (MUSCLE) is computer software for multiple sequence alignment of protein and nucleotide sequences.

For more information about MUSCLE, visit the **MUSCLE official website**.

Programming languages: C/C++

Brief description: Computer software for multiple sequence alignment of protein and nucleotide sequences.

Open source license: GPL-3.0

Recommended Software Version

Muscle 5.0.1428

4.82.2 Environment Requirements

Hardware Requirements

Table 4-403 lists the hardware requirements.

Table 4-403 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-404 lists the software requirements.

Table 4-404 Software requirements

Item	Version	Download Address
MUSCLE	5.0.1428	https://github.com/rcedgar/muscle/releases/tag/v5.0.1428

OS Requirements

Table 4-405 lists the OS requirements.

Table 4-405 OS requirements

Item	Version	Download URL
CentOS	7.9	https://www.centos.org/download/
Kernel	4.18.0-193	https://www.centos.org/download/

4.82.3 Paths for Software Porting

This chapter lists software installation paths involved in the MUSCLE software porting.

Table 4-406 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/muscle	Installation path of MUSCLE.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.82.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-407 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.

4.82.5 Obtaining the Source Code

Procedure

Step 1 Download the MUSCLE installation package.

https://github.com/rcedgar/muscle/releases/tag/v5.0.1428

Step 2 Use an SFTP tool to upload the MUSCLE installation package to the */path/to/muscle* directory on the server.

----End

4.82.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Download the source code to the **/path/to** directory:

```
cd /path/to
git clone -b v5.0.1428 https://github.com/rcedgar/muscle.git
```

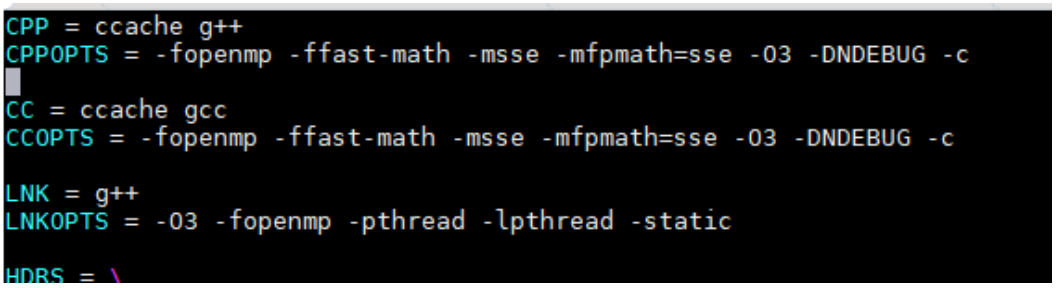
Step 3 Go to the **src** directory.

```
cd /path/to/muscle/src
```

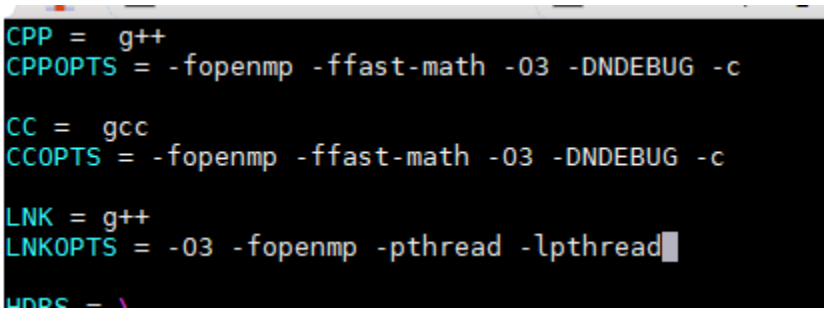
Step 4 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode and delete the **ccache**, **-msse**, **-mfpmath=sse**, and **-static** keywords from lines 1, 2, 4, 5, and 8.

   Before the modification:

   After the modification:

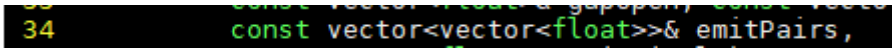
   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

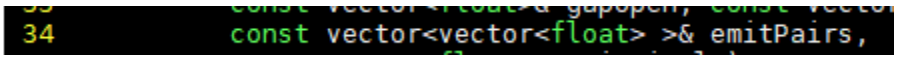
Step 5 Modify the **pairhmm.h** file.

1. Open the **pairhmm.h** file.
   ```
   vi pairhmm.h
   ```
2. Press **i** to enter the edit mode and add a space between the two right arrows (>>) in line 34.

   Before the modification:

   After the modification:

 Copyright © Huawei Technologies Co., Ltd.

NOTE

After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Modify the **pairhmm.cpp** file.

1. Open the **pairhmm.cpp** file.
```
vi pairhmm.cpp
```
2. Press **i** to enter the edit mode and add a space between the two right arrows (>>) in line 70.

   Before the modification:

```
70  const vector<vector<float>>& emitPairs, const vector<float>& emitSingle)
```

   After the modification:

```
70  const vector<vector<float> >& emitPairs, const vector<float>& emitSingle)
```

NOTE

After performing Step 6.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 7 Modify the **perturbhmm.cpp** file.

1. Open the **perturbhmm.cpp** file.
```
vi perturbhmm.cpp
```
2. Press **i** to enter the edit mode and change **abs** in lines 51 and 62 to **fabs**.

   Before the modification:

```
51                  SumT += abs(P1 - P2);
52                  }
53          MeanTransDelta = SumT/NT;
54
55          float SumE = 0;
56          for (uint i = 0; i < AlphaSize; ++i)
57                  {
58                  for (uint j = 0; j < AlphaSize; ++j)
59                          {
60                          float P1 = HP1.m_Emits[i][j];
61                          float P2 = HP2.m_Emits[i][j];
62                          SumE += abs(P1 - P2);
```

   After the modification:

```
51                  SumT += fabs(P1 - P2);
52                  }
53          MeanTransDelta = SumT/NT;
54
55          float SumE = 0;
56          for (uint i = 0; i < AlphaSize; ++i)
57                  {
58                  for (uint j = 0; j < AlphaSize; ++j)
59                          {
60                          float P1 = HP1.m_Emits[i][j];
61                          float P2 = HP2.m_Emits[i][j];
62                          SumE += fabs(P1 - P2);
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

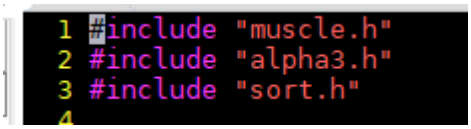
After performing Step 7.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

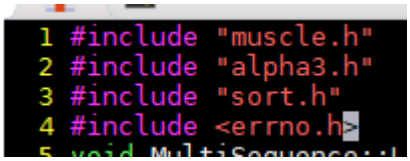
Step 8 Modify the **multisequence.cpp** file.

1. Open the **multisequence.cpp** file.
```
vi multisequence.cpp
```
2. Press **i** to enter the edit mode and add **#include <errno.h>** to the header file list.

   Before the modification:



   After the modification:

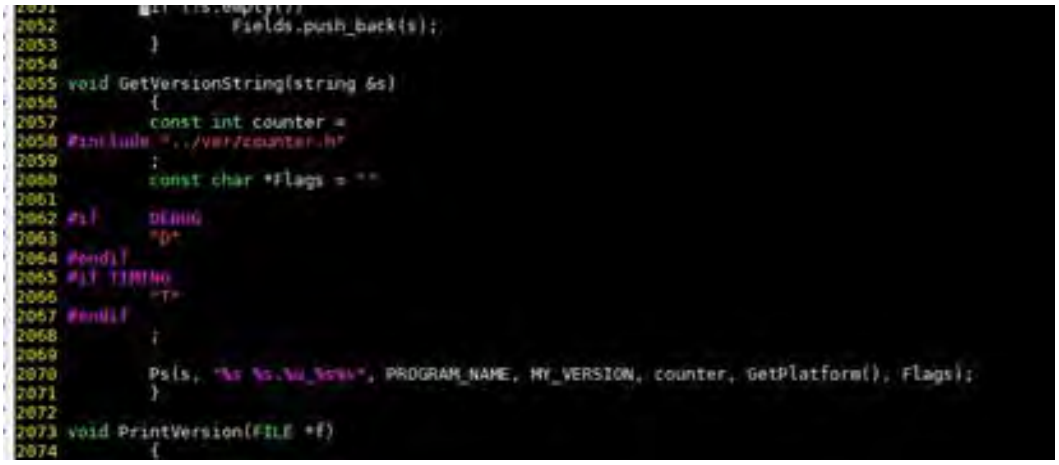


3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 9 Modify the **myutils.cpp** file.

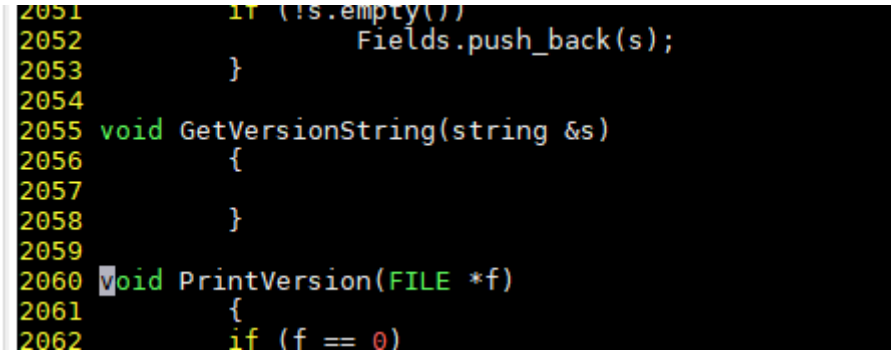
1. Open the **myutils.cpp** file.
```
vi myutils.cpp
```
2. Press **i** to enter the edit mode and delete the entire **GetVersionString** code segment from line 2055. This code segment has defects and the deletion does not affect the program usage.

   Before the modification:



   After the modification:

 Copyright © Huawei Technologies Co., Ltd.



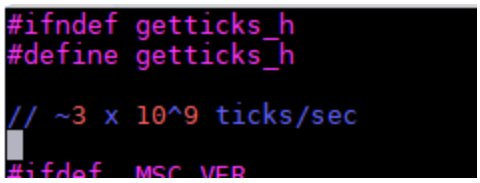
NOTE

After performing Step 9.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 10 Modify the **timing.h** file.

1. Open the **timing.h** file.
```
vi timing.h
```
2. Press **i** to enter the edit mode.
   - Add a macro definition at the beginning of the file.
```
#if defined(__aarch64__)
#include "KunpengTrans.h"
#endif
```
     Before the modification:



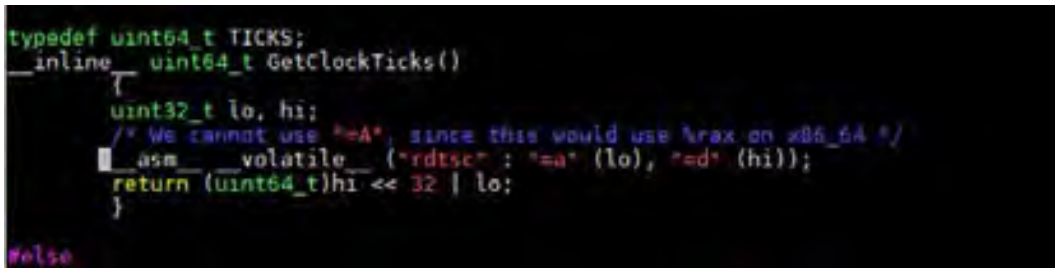
     After the modification:

   - Modify the **__asm__ __volatile__ ("rdtsc" : "=a" (lo), "=d" (hi))** code in line 25 as follows:
```
#if defined(__x86_64__)
        __asm__ __volatile__ ("rdtsc" : "=a" (lo), "=d" (hi));

#elif defined(__aarch64__)
    // Description: Replace with the converted code block suggested. Note:
    // Suggestion:
    //   {
    unsigned int _kp_tmp_CPUFreq = KUNPENG_CPU_FREQUENCY_MHZ;
    unsigned long long _kp_tmp_TSCCount;
    unsigned long long _kp_tmp_TSCFreq;
```

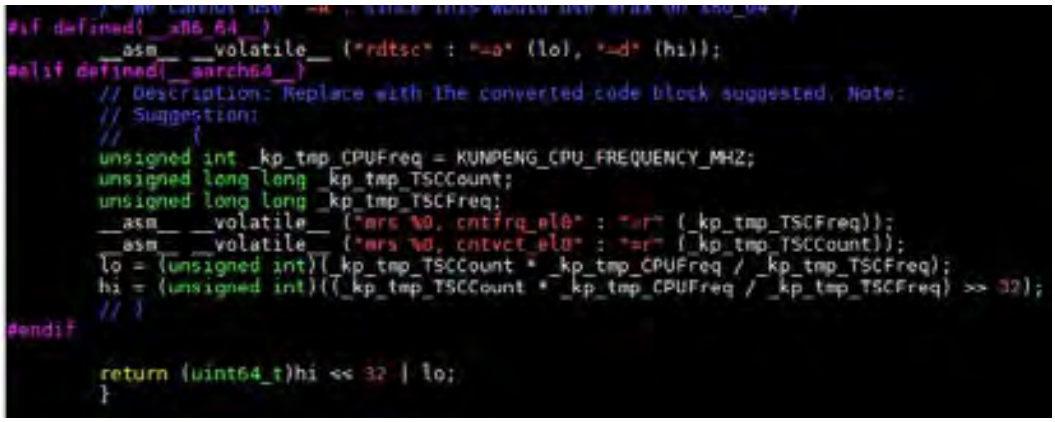
 Copyright © Huawei Technologies Co., Ltd.

```
    __asm__ __volatile__ ("mrs %0, cntfrq_el0" : "=r" (_kp_tmp_TSCFreq));
    __asm__ __volatile__ ("mrs %0, cntvct_el0" : "=r" (_kp_tmp_TSCCount));
    lo = (unsigned int)(_kp_tmp_TSCCount * _kp_tmp_CPUFreq / _kp_tmp_TSCFreq);
    hi = (unsigned int)((_kp_tmp_TSCCount * _kp_tmp_CPUFreq / _kp_tmp_TSCFreq) >> 32);
    // }
#endif
```

Before the modification:



After the modification:



NOTE

After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 11 Create a **KunpengTrans.h** file.

1. Create a **KunpengTrans.h** file.

   vi KunpengTrans.h

2. Press **i** to enter the edit mode and add the following content:

```
/*
 * @Description: KunpengTrans.h
 * @Copyright (c) Huawei Technologies Co., Ltd. 2020-2020. All rights reserved.
 */
#ifndef KUNPENG_TRANS_H
#define KUNPENG_TRANS_H

#include <string.h>


/* ATTENTION:
 * Please set KUNPENG_CPU_FREQUENCY_MHZ to the actual cpu frequecy of your running
environment.
 */
const int KUNPENG_CPU_FREQUENCY_MHZ = 2600;
const int EAX_LEAF = 7;
const int SHIFT_THREE = 3;
/* CASE eax = 0; Highest Function Parameter and Manufacturer ID
```

```
 */
void GetCPUManuID(unsigned int *ebx, unsigned int *edx, unsigned int *ecx);


// %ecx
#define bit_SSE3 (1 << 0)
#define bit_PCLMUL (1 << 1)
#define bit_LZCNT (1 << 5)
#define bit_SSSE3 (1 << 9)
#define bit_FMA (1 << 12)
#define bit_CMPXCHG16B (1 << 13)
#define bit_SSE4_1 (1 << 19)
#define bit_SSE4_2 (1 << 20)
#define bit_MOVBE (1 << 22)
#define bit_POPCNT (1 << 23)
#define bit_AES (1 << 25)
#define bit_XSAVE (1 << 26)
#define bit_OSXSAVE (1 << 27)
#define bit_AVX (1 << 28)
#define bit_F16C (1 << 29)
#define bit_RDRND (1 << 30)

// %edx
#define bit_CMPXCHG8B (1 << 8)
#define bit_CMOV (1 << 15)
#define bit_MMX (1 << 23)
#define bit_FXSAVE (1 << 24)
#define bit_SSE (1 << 25)
#define bit_SSE2 (1 << 26)
/* CASE eax = 1; Processor Info and Feature Bits
 * Skylake:       eax = 0x00050654, ebx = 0x43400800, ecx = 0x7ffefbf7, edx = 0xbfebfbff // Sampled at
6148
 * Cascade Lake : eax = 0x00050657, ebx = 0x08400800, ecx = 0x7ffefbf7, edx = 0xbfebfbff // Sampled
at 6248
*/
void GetCPUFeature(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx);


/* %ebx */
#define bit_FSGSBASE (1 << 0)
#define bit_SGX (1 << 2)
#define bit_BMI (1 << 3)
#define bit_HLE (1 << 4)
#define bit_AVX2 (1 << 5)
#define bit_BMI2 (1 << 8)
#define bit_RTM (1 << 11)
#define bit_MPX (1 << 14)
#define bit_AVX512F (1 << 16)
#define bit_AVX512DQ (1 << 17)
#define bit_RDSEED (1 << 18)
#define bit_ADX (1 << 19)
#define bit_AVX512IFMA (1 << 21)
#define bit_CLFLUSHOPT (1 << 23)
#define bit_CLWB (1 << 24)
#define bit_AVX512PF (1 << 26)
#define bit_AVX512ER (1 << 27)
#define bit_AVX512CD (1 << 28)
#define bit_SHA (1 << 29)
#define bit_AVX512BW (1 << 30)
#define bit_AVX512VL (1u << 31)

/* %ecx */
#define bit_PREFETCHWT1 (1 << 0)
#define bit_AVX512VBMI (1 << 1)
#define bit_PKU (1 << 3)
#define bit_OSPKE (1 << 4)
#define bit_AVX512VBMI2 (1 << 6)
#define bit_SHSTK (1 << 7)
#define bit_GFNI (1 << 8)
```

 Copyright © Huawei Technologies Co., Ltd.

```
#define bit_VAES (1 << 9)
#define bit_AVX512VNNI (1 << 11)
#define bit_VPCLMULQDQ (1 << 10)
#define bit_AVX512BITALG (1 << 12)
#define bit_AVX512VPOPCNTDQ (1 << 14)
#define bit_RDPID (1 << 22)
#define bit_MOVDIRI (1 << 27)
#define bit_MOVDIR64B (1 << 28)

/* %edx */
#define bit_AVX5124VNNIW (1 << 2)
#define bit_AVX5124FMAPS (1 << 3)
#define bit_IBT (1 << 20)
#define bit_PCONFIG (1 << 18)
/* CASE eax = 7, ecx = 0; Extended Features
 * Skylake:       eax = 0x00000000, ebx = 0xd39ffffb, ecx = 0x00000018, edx = 0x9c002400 // Sampled
at 6148
 * Cascade Lake : eax = 0x00000000, ebx = 0xd39ffffb, ecx = 0x00000818, edx = 0xbc000400 //
Sampled at 6248
 */
void GetExtendCPUFeature(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx);


/*
 * GetSupportedCPUID only supported while eax = 0, eax = 1 , eax = 7 and ecx = 0
 * In other cases always return 0x0;
 * chipID support 1,2 -- 1 for skylake server eg. 6148; 2 for Cascade Lake 6248
 */
void GetSupportedCPUID(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx);

void RepStos(void *dest, unsigned long long src, unsigned long long len, unsigned width, int df);


void RepStosB(void *dest, unsigned long long src, unsigned long long len, int DF);


void RepStosW(void *dest, unsigned long long src, unsigned long long len, int DF);


void RepStosD(void *dest, unsigned long long src, unsigned long long len, int DF);


void RepStosQ(void *dest, unsigned long long src, unsigned long long len, int DF);

void RepMovs(void *dest, void *src, unsigned long long len, unsigned width, int df);


#include <arm_neon.h>

#define KP_FORCE_INLINE static inline __attribute__((always_inline))

typedef union {
    int8x16_t vect_s8;
    int16x8_t vect_s16;
    int32x4_t vect_s32;
    int64x2_t vect_s64;
    uint8x16_t vect_u8;
    uint16x8_t vect_u16;
    uint32x4_t vect_u32;
    uint64x2_t vect_u64;
} __kp_m128i;

#define _KP_SIDD_NEGATIVE_POLARITY 0x10        // negate results
#define _KP_SIDD_MASKED_NEGATIVE_POLARITY 0x30 // negate results only before end of string

static uint16_t g_kp_mask_epi16[8] __attribute__((aligned(16))) = { 0x01, 0x02, 0x04, 0x08, 0x10,
0x20, 0x40, 0x80 };
static uint8_t g_kp_mask_epi8[16] __attribute__((aligned(16))) = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x40, 0x80,
```

 Copyright © Huawei Technologies Co., Ltd.

```
                                        0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

#define KP_PCMPSTR_EQ_16x8(a, b, mtx)                                          \
  {                                                                            \
     mtx[0].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 0)), a.vect_u16); \
     mtx[1].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 1)), a.vect_u16); \
     mtx[2].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 2)), a.vect_u16); \
     mtx[3].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 3)), a.vect_u16); \
     mtx[4].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 4)), a.vect_u16); \
     mtx[5].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 5)), a.vect_u16); \
     mtx[6].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 6)), a.vect_u16); \
     mtx[7].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 7)), a.vect_u16); \
  }

#define KP_PCMPSTR_EQ_8x16(a, b, mtx)                                          \
  {                                                                            \
     mtx[0].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 0)), a.vect_u8);   \
     mtx[1].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 1)), a.vect_u8);   \
     mtx[2].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 2)), a.vect_u8);   \
     mtx[3].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 3)), a.vect_u8);   \
     mtx[4].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 4)), a.vect_u8);   \
     mtx[5].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 5)), a.vect_u8);   \
     mtx[6].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 6)), a.vect_u8);   \
     mtx[7].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 7)), a.vect_u8);   \
     mtx[8].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 8)), a.vect_u8);   \
     mtx[9].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 9)), a.vect_u8);   \
     mtx[10].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 10)), a.vect_u8); \
     mtx[11].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 11)), a.vect_u8); \
     mtx[12].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 12)), a.vect_u8); \
     mtx[13].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 13)), a.vect_u8); \
     mtx[14].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 14)), a.vect_u8); \
     mtx[15].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 15)), a.vect_u8); \
  }

#define KP_PCMPSTR_RNG_U16x8(a, b, mtx)                                                \
  {                                                                                    \
     uint16x8_t vect_b[8];                                                             \
     __kp_m128i mask;                                                                  \
     mask.vect_u32 = vdupq_n_u32(0xffff);                                              \
     vect_b[0] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 0));                           \
     vect_b[1] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 1));                           \
     vect_b[2] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 2));                           \
     vect_b[3] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 3));                           \
     vect_b[4] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 4));                           \
     vect_b[5] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 5));                           \
     vect_b[6] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 6));                           \
     vect_b[7] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 7));                           \
     int i;                                                                            \
     for (i = 0; i < 8; i++) {                                                         \
        mtx[i].vect_u16 = vbslq_u16(mask.vect_u16, vcgeq_u16(vect_b[i], a.vect_u16),   \
        vcleq_u16(vect_b[i], a.vect_u16));                                             \
     }                                                                                 \
  }
#define KP_PCMPSTR_RNG_S16x8(a, b, mtx)                                                \
  {                                                                                    \
     int16x8_t vect_b[8];                                                              \
     __kp_m128i mask;                                                                  \
     mask.vect_u32 = vdupq_n_u32(0xffff);                                              \
     vect_b[0] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 0));                           \
     vect_b[1] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 1));                           \
     vect_b[2] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 2));                           \
     vect_b[3] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 3));                           \
     vect_b[4] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 4));                           \
     vect_b[5] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 5));                           \
     vect_b[6] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 6));                           \
     vect_b[7] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 7));                           \
     int i;                                                                            \
     for (i = 0; i < 8; i++) {                                                         \
        mtx[i].vect_u16 = vbslq_u16(mask.vect_u16, vcgeq_s16(vect_b[i], a.vect_s16),   \
```

 Copyright © Huawei Technologies Co., Ltd.

```
        vcleq_s16(vect_b[i], a.vect_s16));                                          \
    }                                                                               \
  }

#define KP_PCMPSTR_RNG_U8x16(a, b, mtx)                                             \
  {                                                                                 \
    uint8x16_t vect_b[16];                                                          \
    __kp_m128i mask;                                                                \
    mask.vect_u16 = vdupq_n_u16(0xff);                                              \
    vect_b[0] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 0));                            \
    vect_b[1] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 1));                            \
    vect_b[2] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 2));                            \
    vect_b[3] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 3));                            \
    vect_b[4] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 4));                            \
    vect_b[5] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 5));                            \
    vect_b[6] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 6));                            \
    vect_b[7] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 7));                            \
    vect_b[8] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 8));                            \
    vect_b[9] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 9));                            \
    vect_b[10] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 10));                          \
    vect_b[11] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 11));                          \
    vect_b[12] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 12));                          \
    vect_b[13] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 13));                          \
    vect_b[14] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 14));                          \
    vect_b[15] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 15));                          \
    int i;                                                                          \
    for (i = 0; i < 16; i++) {                                                      \
      mtx[i].vect_u8 = vbslq_u8(mask.vect_u8, vcgeq_u8(vect_b[i], a.vect_u8), vcleq_u8(vect_b[i],
a.vect_u8)); \
    }                                                                               \
  }

#define KP_PCMPSTR_RNG_S8x16(a, b, mtx)                                             \
  {                                                                                 \
    int8x16_t vect_b[16];                                                           \
    __kp_m128i mask;                                                                \
    mask.vect_u16 = vdupq_n_u16(0xff);                                              \
    vect_b[0] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 0));                            \
    vect_b[1] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 1));                            \
    vect_b[2] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 2));                            \
    vect_b[3] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 3));                            \
    vect_b[4] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 4));                            \
    vect_b[5] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 5));                            \
    vect_b[6] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 6));                            \
    vect_b[7] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 7));                            \
    vect_b[8] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 8));                            \
    vect_b[9] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 9));                            \
    vect_b[10] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 10));                          \
    vect_b[11] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 11));                          \
    vect_b[12] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 12));                          \
    vect_b[13] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 13));                          \
    vect_b[14] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 14));                          \
    vect_b[15] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 15));                          \
    int i;                                                                          \
    for (i = 0; i < 16; i++) {                                                      \
      mtx[i].vect_u8 = vbslq_u8(mask.vect_u8, vcgeq_s8(vect_b[i], a.vect_s8), vcleq_s8(vect_b[i],
a.vect_s8)); \
    }                                                                               \
  }

static int kp_aggregate_equal_any_8x16(int la, int lb, __kp_m128i mtx[16])
{
  int res = 0;
  int j;
  int m = (1 << la) - 1;
  uint8x8_t vect_mask = vld1_u8(g_kp_mask_epi8);
  uint8x8_t t_lo = vtst_u8(vdup_n_u8(m & 0xff), vect_mask);
  uint8x8_t t_hi = vtst_u8(vdup_n_u8(m >> 8), vect_mask);
  uint8x16_t vect = vcombine_u8(t_lo, t_hi);
```

 Copyright © Huawei Technologies Co., Ltd.

```
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vandq_u8(vect, mtx[j].vect_u8);
        mtx[j].vect_u8 = vshrq_n_u8(mtx[j].vect_u8, 7);
        int tmp = vaddvq_u8(mtx[j].vect_u8) ? 1 : 0;
        res |= (tmp << j);
    }
    return res;
}

static int kp_aggregate_equal_any_16x8(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint16x8_t vect = vtstq_u16(vdupq_n_u16(m), vld1q_u16(g_kp_mask_epi16));
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vandq_u16(vect, mtx[j].vect_u16);
        mtx[j].vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 15);
        int tmp = vaddvq_u16(mtx[j].vect_u16) ? 1 : 0;
        res |= (tmp << j);
    }
    return res;
}

static int kp_cal_res_byte_equal_any(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_8x16(a, b, mtx);
    return kp_aggregate_equal_any_8x16(la, lb, mtx);
}

static int kp_cal_res_word_equal_any(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_16x8(a, b, mtx);
    return kp_aggregate_equal_any_16x8(la, lb, mtx);
}

static int kp_aggregate_ranges_16x8(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint16x8_t vect = vtstq_u16(vdupq_n_u16(m), vld1q_u16(g_kp_mask_epi16));
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vandq_u16(vect, mtx[j].vect_u16);
        mtx[j].vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 15);
        __kp_m128i tmp;
        tmp.vect_u32 = vshrq_n_u32(mtx[j].vect_u32, 16);
        uint32x4_t vect_res = vandq_u32(mtx[j].vect_u32, tmp.vect_u32);
        int t = vaddvq_u32(vect_res) ? 1 : 0;
        res |= (t << j);
    }
    return res;
}

static int kp_aggregate_ranges_8x16(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint8x8_t vect_mask = vld1_u8(g_kp_mask_epi8);
    uint8x8_t t_lo = vtst_u8(vdup_n_u8(m & 0xff), vect_mask);
    uint8x8_t t_hi = vtst_u8(vdup_n_u8(m >> 8), vect_mask);
    uint8x16_t vect = vcombine_u8(t_lo, t_hi);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vandq_u8(vect, mtx[j].vect_u8);
        mtx[j].vect_u8 = vshrq_n_u8(mtx[j].vect_u8, 7);
        __kp_m128i tmp;
```

 Copyright © Huawei Technologies Co., Ltd.

```
        tmp.vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 8);
        uint16x8_t vect_res = vandq_u16(mtx[j].vect_u16, tmp.vect_u16);
        int t = vaddvq_u16(vect_res) ? 1 : 0;
        res |= (t << j);
    }
    return res;
}

static int kp_cal_res_ubyte_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_U8x16(a, b, mtx);
    return kp_aggregate_ranges_8x16(la, lb, mtx);
}

static int kp_cal_res_sbyte_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_S8x16(a, b, mtx);
    return kp_aggregate_ranges_8x16(la, lb, mtx);
}

static int kp_cal_res_uword_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_U16x8(a, b, mtx);
    return kp_aggregate_ranges_16x8(la, lb, mtx);
}

static int kp_cal_res_sword_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_S16x8(a, b, mtx);
    return kp_aggregate_ranges_16x8(la, lb, mtx);
}

static int kp_cal_res_byte_equal_each(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    uint8x16_t mtx = vceqq_u8(a.vect_u8, b.vect_u8);
    int m0 = (la < lb) ? 0 : ((1 << la) - (1 << lb));
    int m1 = 0x10000 - (1 << la);
    int tb = 0x10000 - (1 << lb);
    uint8x8_t vect_mask, vect0_lo, vect0_hi, vect1_lo, vect1_hi;
    uint8x8_t tmp_lo, tmp_hi, res_lo, res_hi;
    vect_mask = vld1_u8(g_kp_mask_epi8);
    vect0_lo = vtst_u8(vdup_n_u8(m0), vect_mask);
    vect0_hi = vtst_u8(vdup_n_u8(m0 >> 8), vect_mask);
    vect1_lo = vtst_u8(vdup_n_u8(m1), vect_mask);
    vect1_hi = vtst_u8(vdup_n_u8(m1 >> 8), vect_mask);
    tmp_lo = vtst_u8(vdup_n_u8(tb), vect_mask);
    tmp_hi = vtst_u8(vdup_n_u8(tb >> 8), vect_mask);

    res_lo = vbsl_u8(vect0_lo, vdup_n_u8(0), vget_low_u8(mtx));
    res_hi = vbsl_u8(vect0_hi, vdup_n_u8(0), vget_high_u8(mtx));
    res_lo = vbsl_u8(vect1_lo, tmp_lo, res_lo);
    res_hi = vbsl_u8(vect1_hi, tmp_hi, res_hi);
    res_lo = vand_u8(res_lo, vect_mask);
    res_hi = vand_u8(res_hi, vect_mask);

    int res = vaddv_u8(res_lo) + (vaddv_u8(res_hi) << 8);
    return res;
}

static int kp_cal_res_word_equal_each(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    uint16x8_t mtx = vceqq_u16(a.vect_u16, b.vect_u16);
    int m0 = (la < lb) ? 0 : ((1 << la) - (1 << lb));
    int m1 = 0x100 - (1 << la);
    int tb = 0x100 - (1 << lb);
```

 Copyright © Huawei Technologies Co., Ltd.

```
    uint16x8_t vect_mask = vld1q_u16(g_kp_mask_epi16);
    uint16x8_t vect0 = vtstq_u16(vdupq_n_u16(m0), vect_mask);
    uint16x8_t vect1 = vtstq_u16(vdupq_n_u16(m1), vect_mask);
    uint16x8_t tmp = vtstq_u16(vdupq_n_u16(tb), vect_mask);
    mtx = vbslq_u16(vect0, vdupq_n_u16(0), mtx);
    mtx = vbslq_u16(vect1, tmp, mtx);
    mtx = vandq_u16(mtx, vect_mask);
    return vaddvq_u16(mtx);
}

static int kp_aggregate_equal_ordered_8x16(int bound, int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j, k;
    int m1 = 0x10000 - (1 << la);
    uint8x16_t vect_mask = vld1q_u8(g_kp_mask_epi8);
    uint8x16_t vect1 = vtstq_u8(vdupq_n_u8(m1), vect_mask);
    uint8x16_t vect_minusone = vdupq_n_u8(-1);
    uint8x16_t vect_zero = vdupq_n_u8(0);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vbslq_u8(vect1, vect_minusone, mtx[j].vect_u8);
    }
    for (j = lb; j < bound; j++) {
        mtx[j].vect_u8 = vbslq_u8(vect1, vect_minusone, vect_zero);
    }
    uint8_t enable[16] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    for (j = 0; j < bound; j++) {
        int val = 1;
        uint8x16_t vect_en = vld1q_u8(enable);
        for (k = j; k < bound && val == 1; k++) {
            int t = vaddvq_u8(vandq_u8(mtx[j].vect_u8, vect_en));
            val = (t == bound - j) ? 1 : 0;
        }
        res = (val << j) + res;
        enable[bound - 1 - j] = 0;
    }
    return res;
}

static int kp_aggregate_equal_ordered_16x8(int bound, int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j, k;
    int m1 = 0x100 - (1 << la);
    uint16x8_t vect_mask = vld1q_u16(g_kp_mask_epi16);
    uint16x8_t vect1 = vtstq_u16(vdupq_n_u16(m1), vect_mask);
    uint16x8_t vect_minusone = vdupq_n_u16(-1);
    uint16x8_t vect_zero = vdupq_n_u16(0);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vbslq_u16(vect1, vect_minusone, mtx[j].vect_u16);
    }
    for (j = lb; j < bound; j++) {
        mtx[j].vect_u16 = vbslq_u16(vect1, vect_minusone, vect_zero);
    }
    uint16_t enable[8] = {1, 1, 1, 1, 1, 1, 1, 1};
    for (j = 0; j < bound; j++) {
        int val = 1;
        uint16x8_t vect_en = vld1q_u16(enable);
        for (k = j; k < bound && val == 1; k++) {
            int t = vaddvq_u16(vandq_u16(mtx[j].vect_u16, vect_en));
            val = (t == bound - j) ? 1 : 0;
        }
        res = (val << j) + res;
        enable[bound - 1 - j] = 0;
    }
    return res;
}

static int kp_cal_res_byte_equal_ordered(__kp_m128i a, int la, __kp_m128i b, int lb)
```

```
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_8x16(a, b, mtx);
    return kp_aggregate_equal_ordered_8x16(16, la, lb, mtx);
}

static int kp_cal_res_word_equal_ordered(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_16x8(a, b, mtx);
    return kp_aggregate_equal_ordered_16x8(8, la, lb, mtx);
}

typedef enum {
    KP_CMP_UBYTE_EQUAL_ANY,
    KP_CMP_UWORD_EQUAL_ANY,
    KP_CMP_SBYTE_EQUAL_ANY,
    KP_CMP_SWORD_EQUAL_ANY,
    KP_CMP_UBYTE_RANGES,
    KP_CMP_UWORD_RANGES,
    KP_CMP_SBYTE_RANGES,
    KP_CMP_SWORD_RANGES,
    KP_CMP_UBYTE_EQUAL_EACH,
    KP_CMP_UWORD_EQUAL_EACH,
    KP_CMP_SBYTE_EQUAL_EACH,
    KP_CMP_SWORD_EQUAL_EACH,
    KP_CMP_UBYTE_EQUAL_ORDERED,
    KP_CMP_UWORD_EQUAL_ORDERED,
    KP_CMP_SBYTE_EQUAL_ORDERED,
    KP_CMP_SWORD_EQUAL_ORDERED
} _KP_MM_CMPESTR_ENUM;
typedef int (*KP_CMPESTR)(__kp_m128i a, int la, __kp_m128i b, int lb);
typedef struct {
    _KP_MM_CMPESTR_ENUM cmpintEnum;
    KP_CMPESTR cmpFun;
} KP_CmpestrFuncList;
static KP_CmpestrFuncList g_kp_cmpestrFuncList[] = {{KP_CMP_UBYTE_EQUAL_ANY,
kp_cal_res_byte_equal_any},
                                    {KP_CMP_UWORD_EQUAL_ANY, kp_cal_res_word_equal_any},
                                    {KP_CMP_SBYTE_EQUAL_ANY, kp_cal_res_byte_equal_any},
                                    {KP_CMP_SWORD_EQUAL_ANY, kp_cal_res_word_equal_any},
                                    {KP_CMP_UBYTE_RANGES, kp_cal_res_ubyte_ranges},
                                    {KP_CMP_UWORD_RANGES, kp_cal_res_uword_ranges},
                                    {KP_CMP_SBYTE_RANGES, kp_cal_res_sbyte_ranges},
                                    {KP_CMP_SWORD_RANGES, kp_cal_res_sword_ranges},
                                    {KP_CMP_UBYTE_EQUAL_EACH, kp_cal_res_byte_equal_each},
                                    {KP_CMP_UWORD_EQUAL_EACH, kp_cal_res_word_equal_each},
                                    {KP_CMP_SBYTE_EQUAL_EACH, kp_cal_res_byte_equal_each},
                                    {KP_CMP_SWORD_EQUAL_EACH, kp_cal_res_word_equal_each},
                                    {KP_CMP_UBYTE_EQUAL_ORDERED, kp_cal_res_byte_equal_ordered},
                                    {KP_CMP_UWORD_EQUAL_ORDERED, kp_cal_res_word_equal_ordered},
                                    {KP_CMP_SBYTE_EQUAL_ORDERED, kp_cal_res_byte_equal_ordered},
                                    {KP_CMP_SWORD_EQUAL_ORDERED,
kp_cal_res_word_equal_ordered}};

KP_FORCE_INLINE int kp_neg_fun(int res, int lb, int imm8, int bound)
{
    int m;
    switch (imm8 & 0x30) {
        case _KP_SIDD_NEGATIVE_POLARITY:
            res ^= 0xffffffff;
            break;
        case _KP_SIDD_MASKED_NEGATIVE_POLARITY:
            m = (1 << lb) - 1;
            res ^= m;
            break;
        default:
            break;
    }
```

 Copyright © Huawei Technologies Co., Ltd.

```
    return res & ((bound == 8) ? 0xFF : 0xFFFF);
}

int __remill_simd__mm_cmpestri(__kp_m128i a, int la, __kp_m128i b, int lb, const int imm8, int
*intRes2);


__kp_m128i __remill_simd__mm_cmpestrm(__kp_m128i a, int la, __kp_m128i b, int lb, const int imm8,
int *intRes2);


#endif // KUNPENG_TRANS_H
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 12 Create a **KunpengTrans.cpp** file.

1. Create a **KunpengTrans.cpp** file.

   vi KunpengTrans.cpp

2. Press **i** to enter the edit mode and add the following content:

```
/*
 * @Description: KunpengTrans.h
 * @Copyright (c) Huawei Technologies Co., Ltd. 2020-2020. All rights reserved.
 */
#ifndef KUNPENG_TRANS_H
#define KUNPENG_TRANS_H

#include <string.h>
#include <KunpengTrans.h>

/* ATTENTION:
 * Please set KUNPENG_CPU_FREQUENCY_MHZ to the actual cpu frequecy of your running
environment.
 */
const int KUNPENG_CPU_FREQUENCY_MHZ = 2600;
const int EAX_LEAF = 7;
const int SHIFT_THREE = 3;
/* CASE eax = 0; Highest Function Parameter and Manufacturer ID
 */
void GetCPUManuID(unsigned int *ebx, unsigned int *edx, unsigned int *ecx)
{
    // ID str = "KunpengHisil"
    char b_str[] = "Kunp";
    char d_str[] = "engH";
    char c_str[] = "isil";
    *ebx = *(unsigned int *)b_str;
    *edx = *(unsigned int *)d_str;
    *ecx = *(unsigned int *)c_str;
}

// %ecx
#define bit_SSE3 (1 << 0)
#define bit_PCLMUL (1 << 1)
#define bit_LZCNT (1 << 5)
#define bit_SSSE3 (1 << 9)
#define bit_FMA (1 << 12)
#define bit_CMPXCHG16B (1 << 13)
#define bit_SSE4_1 (1 << 19)
#define bit_SSE4_2 (1 << 20)
#define bit_MOVBE (1 << 22)
#define bit_POPCNT (1 << 23)
#define bit_AES (1 << 25)
#define bit_XSAVE (1 << 26)
#define bit_OSXSAVE (1 << 27)
#define bit_AVX (1 << 28)
#define bit_F16C (1 << 29)
#define bit_RDRND (1 << 30)
```

```
// %edx
#define bit_CMPXCHG8B (1 << 8)
#define bit_CMOV (1 << 15)
#define bit_MMX (1 << 23)
#define bit_FXSAVE (1 << 24)
#define bit_SSE (1 << 25)
#define bit_SSE2 (1 << 26)
/* CASE eax = 1; Processor Info and Feature Bits
 * Skylake:       eax = 0x00050654, ebx = 0x43400800, ecx = 0x7ffefbf7, edx = 0xbfebfbff // Sampled at
6148
 * Cascade Lake : eax = 0x00050657, ebx = 0x08400800, ecx = 0x7ffefbf7, edx = 0xbfebfbff // Sampled
at 6248
*/
void GetCPUFeature(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx)
{
    *eax = 0x0; // Processor Info not defined in kunpeng;
    *ebx = 0x0; // Additional Info not defined in kunpeng;
    *ecx = bit_SSE3 | bit_LZCNT | bit_SSSE3 | bit_SSE4_1 | bit_SSE4_2 | bit_POPCNT;
    *edx = bit_MMX | bit_SSE | bit_SSE2;
}

/* %ebx */
#define bit_FSGSBASE (1 << 0)
#define bit_SGX (1 << 2)
#define bit_BMI (1 << 3)
#define bit_HLE (1 << 4)
#define bit_AVX2 (1 << 5)
#define bit_BMI2 (1 << 8)
#define bit_RTM (1 << 11)
#define bit_MPX (1 << 14)
#define bit_AVX512F (1 << 16)
#define bit_AVX512DQ (1 << 17)
#define bit_RDSEED (1 << 18)
#define bit_ADX (1 << 19)
#define bit_AVX512IFMA (1 << 21)
#define bit_CLFLUSHOPT (1 << 23)
#define bit_CLWB (1 << 24)
#define bit_AVX512PF (1 << 26)
#define bit_AVX512ER (1 << 27)
#define bit_AVX512CD (1 << 28)
#define bit_SHA (1 << 29)
#define bit_AVX512BW (1 << 30)
#define bit_AVX512VL (1u << 31)

/* %ecx */
#define bit_PREFETCHWT1 (1 << 0)
#define bit_AVX512VBMI (1 << 1)
#define bit_PKU (1 << 3)
#define bit_OSPKE (1 << 4)
#define bit_AVX512VBMI2 (1 << 6)
#define bit_SHSTK (1 << 7)
#define bit_GFNI (1 << 8)
#define bit_VAES (1 << 9)
#define bit_AVX512VNNI (1 << 11)
#define bit_VPCLMULQDQ (1 << 10)
#define bit_AVX512BITALG (1 << 12)
#define bit_AVX512VPOPCNTDQ (1 << 14)
#define bit_RDPID (1 << 22)
#define bit_MOVDIRI (1 << 27)
#define bit_MOVDIR64B (1 << 28)

/* %edx */
#define bit_AVX5124VNNIW (1 << 2)
#define bit_AVX5124FMAPS (1 << 3)
#define bit_IBT (1 << 20)
#define bit_PCONFIG (1 << 18)
/* CASE eax = 7, ecx = 0; Extended Features
 * Skylake:       eax = 0x00000000, ebx = 0xd39fffbb, ecx = 0x00000018, edx = 0x9c002400 // Sampled
at 6148
```

 Copyright © Huawei Technologies Co., Ltd.

```
 * Cascade Lake : eax = 0x00000000, ebx = 0xd39ffffb, ecx = 0x00000818, edx = 0xbc000400 //
Sampled at 6248
 */
void GetExtendCPUFeature(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx)
{
    *eax = 0x0;
    *ebx = 0x0;
    *ecx = 0x0;
    *edx = 0x0;
}

/*
 * GetSupportedCPUID only supported while eax = 0, eax = 1 , eax = 7 and ecx = 0
 * In other cases always return 0x0;
 * chipID support 1,2 -- 1 for skylake server eg. 6148; 2 for Cascade Lake 6248
 */
void GetSupportedCPUID(unsigned int *eax, unsigned int *ebx, unsigned int *ecx, unsigned int *edx)
{
    unsigned int leaf = *eax;
    unsigned int count = *ecx;
    if (leaf == 0) {
        *eax = 0x7;
        GetCPUManuID(ebx, edx, ecx);
    } else if (leaf == 1) {
        GetCPUFeature(eax, ebx, ecx, edx);
    } else if (leaf == EAX_LEAF && count == 0) {
        GetExtendCPUFeature(eax, ebx, ecx, edx);
    } else {
        *eax = 0x0;
        *ebx = 0x0;
        *ecx = 0x0;
        *edx = 0x0;
    }
}

void RepStos(void *dest, unsigned long long src, unsigned long long len, unsigned width, int df)
{
    unsigned n = width >> SHIFT_THREE;
    unsigned char *d = (unsigned char *)dest;
    unsigned i;
    unsigned j;
    if (df == 1) {
        for (i = 0; i < len; i++) {
            for (j = 0; j < n; j++) {
                d[j] = src >> (j << SHIFT_THREE);
            }
            d -= n;
        }
    } else {
        for (i = 0; i < len; i++) {
            for (j = 0; j < n; j++) {
                d[j] = src >> (j << SHIFT_THREE);
            }
            d += n;
        }
    }
    return;
}

void RepStosB(void *dest, unsigned long long src, unsigned long long len, int DF)
{
    unsigned char *s = (unsigned char *)dest;
    unsigned char *e = (unsigned char *)dest;
    if (DF) {
        s = s - len;
    } else {
        e = e + len;
    }
    while (s < e) {
```

 Copyright © Huawei Technologies Co., Ltd.

```
        *s++ = src;
    }
}

void RepStosW(void *dest, unsigned long long src, unsigned long long len, int DF)
{
    unsigned short *s = (unsigned short *)dest;
    unsigned short *e = (unsigned short *)dest;
    if (DF) {
        s = s - len;
    } else {
        e = e + len;
    }
    while (s < e) {
        *s++ = src;
    }
}

void RepStosD(void *dest, unsigned long long src, unsigned long long len, int DF)
{
    unsigned int *s = (unsigned int *)dest;
    unsigned int *e = (unsigned int *)dest;
    if (DF) {
        s = s - len;
    } else {
        e = e + len;
    }
    while (s < e) {
        *s++ = src;
    }
}

void RepStosQ(void *dest, unsigned long long src, unsigned long long len, int DF)
{
    unsigned long long *s = (unsigned long long *)dest;
    unsigned long long *e = (unsigned long long *)dest;
    if (DF) {
        s = s - len;
    } else {
        e = e + len;
    }
    while (s < e) {
        *s++ = src;
    }
}

void RepMovs(void *dest, void *src, unsigned long long len, unsigned width, int df)
{
    unsigned n = len * (width >> SHIFT_THREE);
    char *d = NULL;
    char *s = NULL;
    if (df == 1) {
        d = (char *)dest - (n - 1);
        s = (char *)src - (n - 1);
    } else {
        d = (char *)dest;
        s = (char *)src;
    }
    memcpy(d, s, n); // This is an insecure function. If necessary, change it to a secure function.
    return;
}

#include <arm_neon.h>

#define KP_FORCE_INLINE static inline __attribute__((always_inline))

typedef union {
    int8x16_t vect_s8;
    int16x8_t vect_s16;
```

 Copyright © Huawei Technologies Co., Ltd.

```
    int32x4_t vect_s32;
    int64x2_t vect_s64;
    uint8x16_t vect_u8;
    uint16x8_t vect_u16;
    uint32x4_t vect_u32;
    uint64x2_t vect_u64;
} __kp_m128i;

#define _KP_SIDD_NEGATIVE_POLARITY 0x10        // negate results
#define _KP_SIDD_MASKED_NEGATIVE_POLARITY 0x30 // negate results only before end of string

static uint16_t g_kp_mask_epi16[8] __attribute__((aligned(16))) = { 0x01, 0x02, 0x04, 0x08, 0x10,
0x20, 0x40, 0x80 };
static uint8_t g_kp_mask_epi8[16] __attribute__((aligned(16))) = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x40, 0x80,
                                                   0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

#define KP_PCMPSTR_EQ_16x8(a, b, mtx)                                                 \
    {                                                                                 \
        mtx[0].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 0)), a.vect_u16); \
        mtx[1].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 1)), a.vect_u16); \
        mtx[2].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 2)), a.vect_u16); \
        mtx[3].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 3)), a.vect_u16); \
        mtx[4].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 4)), a.vect_u16); \
        mtx[5].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 5)), a.vect_u16); \
        mtx[6].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 6)), a.vect_u16); \
        mtx[7].vect_u16 = vceqq_u16(vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 7)), a.vect_u16); \
    }

#define KP_PCMPSTR_EQ_8x16(a, b, mtx)                                                 \
    {                                                                                 \
        mtx[0].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 0)), a.vect_u8);   \
        mtx[1].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 1)), a.vect_u8);   \
        mtx[2].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 2)), a.vect_u8);   \
        mtx[3].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 3)), a.vect_u8);   \
        mtx[4].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 4)), a.vect_u8);   \
        mtx[5].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 5)), a.vect_u8);   \
        mtx[6].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 6)), a.vect_u8);   \
        mtx[7].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 7)), a.vect_u8);   \
        mtx[8].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 8)), a.vect_u8);   \
        mtx[9].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 9)), a.vect_u8);   \
        mtx[10].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 10)), a.vect_u8); \
        mtx[11].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 11)), a.vect_u8); \
        mtx[12].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 12)), a.vect_u8); \
        mtx[13].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 13)), a.vect_u8); \
        mtx[14].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 14)), a.vect_u8); \
        mtx[15].vect_u8 = vceqq_u8(vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 15)), a.vect_u8); \
    }

#define KP_PCMPSTR_RNG_U16x8(a, b, mtx)                                                   \
    {                                                                                     \
        uint16x8_t vect_b[8];                                                             \
        __kp_m128i mask;                                                                  \
        mask.vect_u32 = vdupq_n_u32(0xffff);                                              \
        vect_b[0] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 0));                           \
        vect_b[1] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 1));                           \
        vect_b[2] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 2));                           \
        vect_b[3] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 3));                           \
        vect_b[4] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 4));                           \
        vect_b[5] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 5));                           \
        vect_b[6] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 6));                           \
        vect_b[7] = vdupq_n_u16(vgetq_lane_u16(b.vect_u16, 7));                           \
        int i;                                                                            \
        for (i = 0; i < 8; i++) {                                                         \
            mtx[i].vect_u16 = vbslq_u16(mask.vect_u16, vcgeq_u16(vect_b[i], a.vect_u16),  \
            vcleq_u16(vect_b[i], a.vect_u16));                                            \
        }                                                                                 \
    }
#define KP_PCMPSTR_RNG_S16x8(a, b, mtx)                                                   \
```

 Copyright © Huawei Technologies Co., Ltd.

```
  {                                                                                \
    int16x8_t vect_b[8];                                                           \
    __kp_m128i mask;                                                               \
    mask.vect_u32 = vdupq_n_u32(0xffff);                                           \
    vect_b[0] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 0));                        \
    vect_b[1] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 1));                        \
    vect_b[2] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 2));                        \
    vect_b[3] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 3));                        \
    vect_b[4] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 4));                        \
    vect_b[5] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 5));                        \
    vect_b[6] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 6));                        \
    vect_b[7] = vdupq_n_s16(vgetq_lane_s16(b.vect_s16, 7));                        \
    int i;                                                                         \
    for (i = 0; i < 8; i++) {                                                      \
      mtx[i].vect_u16 = vbslq_u16(mask.vect_u16, vcgeq_s16(vect_b[i], a.vect_s16),                \
      vcleq_s16(vect_b[i], a.vect_s16));                                           \
    }                                                                              \
  }

#define KP_PCMPSTR_RNG_U8x16(a, b, mtx)                                                          \
  {                                                                                \
    uint8x16_t vect_b[16];                                                         \
    __kp_m128i mask;                                                               \
    mask.vect_u16 = vdupq_n_u16(0xff);                                             \
    vect_b[0] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 0));                                   \
    vect_b[1] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 1));                                   \
    vect_b[2] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 2));                                   \
    vect_b[3] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 3));                                   \
    vect_b[4] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 4));                                   \
    vect_b[5] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 5));                                   \
    vect_b[6] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 6));                                   \
    vect_b[7] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 7));                                   \
    vect_b[8] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 8));                                   \
    vect_b[9] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 9));                                   \
    vect_b[10] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 10));                                  \
    vect_b[11] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 11));                                  \
    vect_b[12] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 12));                                  \
    vect_b[13] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 13));                                  \
    vect_b[14] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 14));                                  \
    vect_b[15] = vdupq_n_u8(vgetq_lane_u8(b.vect_u8, 15));                                  \
    int i;                                                                         \
    for (i = 0; i < 16; i++) {                                                     \
      mtx[i].vect_u8 = vbslq_u8(mask.vect_u8, vcgeq_u8(vect_b[i], a.vect_u8), vcleq_u8(vect_b[i],
a.vect_u8)); \
    }                                                                              \
  }

#define KP_PCMPSTR_RNG_S8x16(a, b, mtx)                                                          \
  {                                                                                \
    int8x16_t vect_b[16];                                                          \
    __kp_m128i mask;                                                               \
    mask.vect_u16 = vdupq_n_u16(0xff);                                             \
    vect_b[0] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 0));                                   \
    vect_b[1] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 1));                                   \
    vect_b[2] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 2));                                   \
    vect_b[3] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 3));                                   \
    vect_b[4] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 4));                                   \
    vect_b[5] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 5));                                   \
    vect_b[6] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 6));                                   \
    vect_b[7] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 7));                                   \
    vect_b[8] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 8));                                   \
    vect_b[9] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 9));                                   \
    vect_b[10] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 10));                                  \
    vect_b[11] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 11));                                  \
    vect_b[12] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 12));                                  \
    vect_b[13] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 13));                                  \
    vect_b[14] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 14));                                  \
    vect_b[15] = vdupq_n_s8(vgetq_lane_s8(b.vect_s8, 15));                                  \
    int i;                                                                         \
```

 Copyright © Huawei Technologies Co., Ltd.

```
        for (i = 0; i < 16; i++) {                                                  \
            mtx[i].vect_u8 = vbslq_u8(mask.vect_u8, vcgeq_s8(vect_b[i], a.vect_s8), vcleq_s8(vect_b[i],
a.vect_s8)); \
        }                                                                           \
    }

static int kp_aggregate_equal_any_8x16(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint8x8_t vect_mask = vld1_u8(g_kp_mask_epi8);
    uint8x8_t t_lo = vtst_u8(vdup_n_u8(m & 0xff), vect_mask);
    uint8x8_t t_hi = vtst_u8(vdup_n_u8(m >> 8), vect_mask);
    uint8x16_t vect = vcombine_u8(t_lo, t_hi);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vandq_u8(vect, mtx[j].vect_u8);
        mtx[j].vect_u8 = vshrq_n_u8(mtx[j].vect_u8, 7);
        int tmp = vaddvq_u8(mtx[j].vect_u8) ? 1 : 0;
        res |= (tmp << j);
    }
    return res;
}

static int kp_aggregate_equal_any_16x8(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint16x8_t vect = vtstq_u16(vdupq_n_u16(m), vld1q_u16(g_kp_mask_epi16));
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vandq_u16(vect, mtx[j].vect_u16);
        mtx[j].vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 15);
        int tmp = vaddvq_u16(mtx[j].vect_u16) ? 1 : 0;
        res |= (tmp << j);
    }
    return res;
}

static int kp_cal_res_byte_equal_any(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_8x16(a, b, mtx);
    return kp_aggregate_equal_any_8x16(la, lb, mtx);
}

static int kp_cal_res_word_equal_any(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_16x8(a, b, mtx);
    return kp_aggregate_equal_any_16x8(la, lb, mtx);
}

static int kp_aggregate_ranges_16x8(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint16x8_t vect = vtstq_u16(vdupq_n_u16(m), vld1q_u16(g_kp_mask_epi16));
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vandq_u16(vect, mtx[j].vect_u16);
        mtx[j].vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 15);
        __kp_m128i tmp;
        tmp.vect_u32 = vshrq_n_u32(mtx[j].vect_u32, 16);
        uint32x4_t vect_res = vandq_u32(mtx[j].vect_u32, tmp.vect_u32);
        int t = vaddvq_u32(vect_res) ? 1 : 0;
        res |= (t << j);
    }
    return res;
```

 Copyright © Huawei Technologies Co., Ltd.

```
}

static int kp_aggregate_ranges_8x16(int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j;
    int m = (1 << la) - 1;
    uint8x8_t vect_mask = vld1_u8(g_kp_mask_epi8);
    uint8x8_t t_lo = vtst_u8(vdup_n_u8(m & 0xff), vect_mask);
    uint8x8_t t_hi = vtst_u8(vdup_n_u8(m >> 8), vect_mask);
    uint8x16_t vect = vcombine_u8(t_lo, t_hi);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vandq_u8(vect, mtx[j].vect_u8);
        mtx[j].vect_u8 = vshrq_n_u8(mtx[j].vect_u8, 7);
        __kp_m128i tmp;
        tmp.vect_u16 = vshrq_n_u16(mtx[j].vect_u16, 8);
        uint16x8_t vect_res = vandq_u16(mtx[j].vect_u16, tmp.vect_u16);
        int t = vaddvq_u16(vect_res) ? 1 : 0;
        res |= (t << j);
    }
    return res;
}

static int kp_cal_res_ubyte_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_U8x16(a, b, mtx);
    return kp_aggregate_ranges_8x16(la, lb, mtx);
}

static int kp_cal_res_sbyte_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_S8x16(a, b, mtx);
    return kp_aggregate_ranges_8x16(la, lb, mtx);
}

static int kp_cal_res_uword_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_U16x8(a, b, mtx);
    return kp_aggregate_ranges_16x8(la, lb, mtx);
}

static int kp_cal_res_sword_ranges(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_RNG_S16x8(a, b, mtx);
    return kp_aggregate_ranges_16x8(la, lb, mtx);
}

static int kp_cal_res_byte_equal_each(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    uint8x16_t mtx = vceqq_u8(a.vect_u8, b.vect_u8);
    int m0 = (la < lb) ? 0 : ((1 << la) - (1 << lb));
    int m1 = 0x10000 - (1 << la);
    int tb = 0x10000 - (1 << lb);
    uint8x8_t vect_mask, vect0_lo, vect0_hi, vect1_lo, vect1_hi;
    uint8x8_t tmp_lo, tmp_hi, res_lo, res_hi;
    vect_mask = vld1_u8(g_kp_mask_epi8);
    vect0_lo = vtst_u8(vdup_n_u8(m0), vect_mask);
    vect0_hi = vtst_u8(vdup_n_u8(m0 >> 8), vect_mask);
    vect1_lo = vtst_u8(vdup_n_u8(m1), vect_mask);
    vect1_hi = vtst_u8(vdup_n_u8(m1 >> 8), vect_mask);
    tmp_lo = vtst_u8(vdup_n_u8(tb), vect_mask);
    tmp_hi = vtst_u8(vdup_n_u8(tb >> 8), vect_mask);

    res_lo = vbsl_u8(vect0_lo, vdup_n_u8(0), vget_low_u8(mtx));
    res_hi = vbsl_u8(vect0_hi, vdup_n_u8(0), vget_high_u8(mtx));
```

 Copyright © Huawei Technologies Co., Ltd.

```
    res_lo = vbsl_u8(vect1_lo, tmp_lo, res_lo);
    res_hi = vbsl_u8(vect1_hi, tmp_hi, res_hi);
    res_lo = vand_u8(res_lo, vect_mask);
    res_hi = vand_u8(res_hi, vect_mask);

    int res = vaddv_u8(res_lo) + (vaddv_u8(res_hi) << 8);
    return res;
}

static int kp_cal_res_word_equal_each(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    uint16x8_t mtx = vceqq_u16(a.vect_u16, b.vect_u16);
    int m0 = (la < lb) ? 0 : ((1 << la) - (1 << lb));
    int m1 = 0x100 - (1 << la);
    int tb = 0x100 - (1 << lb);
    uint16x8_t vect_mask = vld1q_u16(g_kp_mask_epi16);
    uint16x8_t vect0 = vtstq_u16(vdupq_n_u16(m0), vect_mask);
    uint16x8_t vect1 = vtstq_u16(vdupq_n_u16(m1), vect_mask);
    uint16x8_t tmp = vtstq_u16(vdupq_n_u16(tb), vect_mask);
    mtx = vbslq_u16(vect0, vdupq_n_u16(0), mtx);
    mtx = vbslq_u16(vect1, tmp, mtx);
    mtx = vandq_u16(mtx, vect_mask);
    return vaddvq_u16(mtx);
}

static int kp_aggregate_equal_ordered_8x16(int bound, int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j, k;
    int m1 = 0x10000 - (1 << la);
    uint8x16_t vect_mask = vld1q_u8(g_kp_mask_epi8);
    uint8x16_t vect1 = vtstq_u8(vdupq_n_u8(m1), vect_mask);
    uint8x16_t vect_minusone = vdupq_n_u8(-1);
    uint8x16_t vect_zero = vdupq_n_u8(0);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u8 = vbslq_u8(vect1, vect_minusone, mtx[j].vect_u8);
    }
    for (j = lb; j < bound; j++) {
        mtx[j].vect_u8 = vbslq_u8(vect1, vect_minusone, vect_zero);
    }
    uint8_t enable[16] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    for (j = 0; j < bound; j++) {
        int val = 1;
        uint8x16_t vect_en = vld1q_u8(enable);
        for (k = j; k < bound && val == 1; k++) {
            int t = vaddvq_u8(vandq_u8(mtx[j].vect_u8, vect_en));
            val = (t == bound - j) ? 1 : 0;
        }
        res = (val << j) + res;
        enable[bound - 1 - j] = 0;
    }
    return res;
}

static int kp_aggregate_equal_ordered_16x8(int bound, int la, int lb, __kp_m128i mtx[16])
{
    int res = 0;
    int j, k;
    int m1 = 0x100 - (1 << la);
    uint16x8_t vect_mask = vld1q_u16(g_kp_mask_epi16);
    uint16x8_t vect1 = vtstq_u16(vdupq_n_u16(m1), vect_mask);
    uint16x8_t vect_minusone = vdupq_n_u16(-1);
    uint16x8_t vect_zero = vdupq_n_u16(0);
    for (j = 0; j < lb; j++) {
        mtx[j].vect_u16 = vbslq_u16(vect1, vect_minusone, mtx[j].vect_u16);
    }
    for (j = lb; j < bound; j++) {
        mtx[j].vect_u16 = vbslq_u16(vect1, vect_minusone, vect_zero);
    }
```

 Copyright © Huawei Technologies Co., Ltd.

```
    uint16_t enable[8] = {1, 1, 1, 1, 1, 1, 1, 1};
    for (j = 0; j < bound; j++) {
        int val = 1;
        uint16x8_t vect_en = vld1q_u16(enable);
        for (k = j; k < bound && val == 1; k++) {
            int t = vaddvq_u16(vandq_u16(mtx[j].vect_u16, vect_en));
            val = (t == bound - j) ? 1 : 0;
        }
        res = (val << j) + res;
        enable[bound - 1 - j] = 0;
    }
    return res;
}

static int kp_cal_res_byte_equal_ordered(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_8x16(a, b, mtx);
    return kp_aggregate_equal_ordered_8x16(16, la, lb, mtx);
}

static int kp_cal_res_word_equal_ordered(__kp_m128i a, int la, __kp_m128i b, int lb)
{
    __kp_m128i mtx[16];
    KP_PCMPSTR_EQ_16x8(a, b, mtx);
    return kp_aggregate_equal_ordered_16x8(8, la, lb, mtx);
}

typedef enum {
    KP_CMP_UBYTE_EQUAL_ANY,
    KP_CMP_UWORD_EQUAL_ANY,
    KP_CMP_SBYTE_EQUAL_ANY,
    KP_CMP_SWORD_EQUAL_ANY,
    KP_CMP_UBYTE_RANGES,
    KP_CMP_UWORD_RANGES,
    KP_CMP_SBYTE_RANGES,
    KP_CMP_SWORD_RANGES,
    KP_CMP_UBYTE_EQUAL_EACH,
    KP_CMP_UWORD_EQUAL_EACH,
    KP_CMP_SBYTE_EQUAL_EACH,
    KP_CMP_SWORD_EQUAL_EACH,
    KP_CMP_UBYTE_EQUAL_ORDERED,
    KP_CMP_UWORD_EQUAL_ORDERED,
    KP_CMP_SBYTE_EQUAL_ORDERED,
    KP_CMP_SWORD_EQUAL_ORDERED
} _KP_MM_CMPESTR_ENUM;
typedef int (*KP_CMPESTR)(__kp_m128i a, int la, __kp_m128i b, int lb);
typedef struct {
    _KP_MM_CMPESTR_ENUM cmpintEnum;
    KP_CMPESTR cmpFun;
} KP_CmpestrFuncList;
static KP_CmpestrFuncList g_kp_cmpestrFuncList[] = {{KP_CMP_UBYTE_EQUAL_ANY,
kp_cal_res_byte_equal_any},
                                    {KP_CMP_UWORD_EQUAL_ANY, kp_cal_res_word_equal_any},
                                    {KP_CMP_SBYTE_EQUAL_ANY, kp_cal_res_byte_equal_any},
                                    {KP_CMP_SWORD_EQUAL_ANY, kp_cal_res_word_equal_any},
                                    {KP_CMP_UBYTE_RANGES, kp_cal_res_ubyte_ranges},
                                    {KP_CMP_UWORD_RANGES, kp_cal_res_uword_ranges},
                                    {KP_CMP_SBYTE_RANGES, kp_cal_res_sbyte_ranges},
                                    {KP_CMP_SWORD_RANGES, kp_cal_res_sword_ranges},
                                    {KP_CMP_UBYTE_EQUAL_EACH, kp_cal_res_byte_equal_each},
                                    {KP_CMP_UWORD_EQUAL_EACH, kp_cal_res_word_equal_each},
                                    {KP_CMP_SBYTE_EQUAL_EACH, kp_cal_res_byte_equal_each},
                                    {KP_CMP_SWORD_EQUAL_EACH, kp_cal_res_word_equal_each},
                                    {KP_CMP_UBYTE_EQUAL_ORDERED, kp_cal_res_byte_equal_ordered},
                                    {KP_CMP_UWORD_EQUAL_ORDERED, kp_cal_res_word_equal_ordered},
                                    {KP_CMP_SBYTE_EQUAL_ORDERED, kp_cal_res_byte_equal_ordered},
                                    {KP_CMP_SWORD_EQUAL_ORDERED,
kp_cal_res_word_equal_ordered}};
```

 Copyright © Huawei Technologies Co., Ltd.

```
KP_FORCE_INLINE int kp_neg_fun(int res, int lb, int imm8, int bound)
{
    int m;
    switch (imm8 & 0x30) {
        case _KP_SIDD_NEGATIVE_POLARITY:
            res ^= 0xffffffff;
            break;
        case _KP_SIDD_MASKED_NEGATIVE_POLARITY:
            m = (1 << lb) - 1;
            res ^= m;
            break;
        default:
            break;
    }

    return res & ((bound == 8) ? 0xFF : 0xFFFF);
}

int __remill_simd__mm_cmpestri(__kp_m128i a, int la, __kp_m128i b, int lb, const int imm8, int
*intRes2)
{
    int bound = (imm8 & 0x01) ? 8 : 16;
    __asm__ __volatile__ (
        "eor w0, %w[a], %w[a], asr31          \n\t"
        "sub %w[a], w0, %w[a], asr31          \n\t"
        "eor w1, %w[b], %w[b], asr31          \n\t"
        "sub %w[b], w1, %w[b], asr31          \n\t"
        "cmp %w[a], %w[bd]                   \n\t"
        "csel %w[a], %w[bd], %w[a], gt        \n\t"
        "cmp %w[b], %w[bd]                   \n\t"
        "csel %w[b], %w[bd], %w[b], gt        \n\t"
        :[a]"+r"(la), [b]"+r"(lb)
        :[bd]"r"(bound)
        :"w0", "w1");

    int r2 = g_kp_cmpestrFuncList[imm8 & 0x0f].cmpFun(a, la, b, lb);
    r2 = kp_neg_fun(r2, lb, imm8, bound);
    *intRes2 = r2;
    return (r2 == 0) ? bound : ((imm8 & 0x40) ? (31 - __builtin_clz(r2)) : __builtin_ctz(r2));
}

__kp_m128i __remill_simd__mm_cmpestrm(__kp_m128i a, int la, __kp_m128i b, int lb, const int imm8,
int *intRes2)
{
    __kp_m128i dst;
    int bound = (imm8 & 0x01) ? 8 : 16;
    __asm__ __volatile__ (
        "eor w0, %w[a], %w[a], asr31          \n\t"
        "sub %w[a], w0, %w[a], asr31          \n\t"
        "eor w1, %w[b], %w[b], asr31          \n\t"
        "sub %w[b], w1, %w[b], asr31          \n\t"
        "cmp %w[a], %w[bd]                   \n\t"
        "csel %w[a], %w[bd], %w[a], gt        \n\t"
        "cmp %w[b], %w[bd]                   \n\t"
        "csel %w[b], %w[bd], %w[b], gt        \n\t"
        :[a]"+r"(la), [b]"+r"(lb)
        :[bd]"r"(bound)
        :"w0", "w1");

    int r2 = g_kp_cmpestrFuncList[imm8 & 0x0f].cmpFun(a, la, b, lb);
    r2 = kp_neg_fun(r2, lb, imm8, bound);
    *intRes2 = r2;
    dst.vect_u8 = vdupq_n_u8(0);
    if (imm8 & 0x40) {
        if (bound == 8) {
            uint16x8_t tmp = vtstq_u16(vdupq_n_u16(r2), vld1q_u16(g_kp_mask_epi16));
            dst.vect_u16 = vbslq_u16(tmp, vdupq_n_u16(-1), dst.vect_u16);
        } else {
```

 Copyright © Huawei Technologies Co., Ltd.

```
            uint8x16_t vect_r2 = vcombine_u8(vdup_n_u8(r2), vdup_n_u8(r2 >> 8));
            uint8x16_t tmp = vtstq_u8(vect_r2, vld1q_u8(g_kp_mask_epi8));
            dst.vect_u8 = vbslq_u8(tmp, vdupq_n_u8(-1), dst.vect_u8);
        }
    } else {
        if (bound == 16) {
            dst.vect_u16 = vsetq_lane_u16(r2 & 0xffff, dst.vect_u16, 0);
        } else {
            dst.vect_u8 = vsetq_lane_u8(r2 & 0xff, dst.vect_u8, 0);
        }
    }

    return dst;
}

#endif // KUNPENG_TRANS_H
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 13 Upload **KunpengTrans.h** and **KunpengTrans.cpp** to the **src** directory of the MUSCLE source code.

Step 14 Perform the compilation.

```
make
```

After the compilation is successful, the executable file is generated in the **/muscle/src/o** directory.

----End

4.82.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
/path/to/muscle/src/o/muscle5
```

Information similar to the following is displayed:

----End

4.83 Nanopolish 0.13.2 Porting Guide (CentOS 8.2)

4.83.1 Introduction

Nanopolish is a software package for signal-level analysis of Oxford Nanopore sequencing data. Nanopolish can calculate an improved consensus sequence for a draft genome assembly, detect base modifications, call SNPs and indels with respect to a reference genome and more.

For more information about Nanopolish, visit the **Nanopolish official website**.

Programming language: C++

Brief description: A software package for signal-level analysis of Oxford Nanopore sequencing data.

Open source license: MIT License

Recommended Software Version

Nanopolish 0.13.2

4.83.2 Environment Requirements

Hardware Requirements

Table 4-408 lists the hardware requirements.

Table 4-408 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-409 lists the software requirements.

Table 4-409 Software requirements

Item	Version	Download Address
Python	3.6.8	Contained in the OS

OS Requirements

Table 4-410 lists the OS requirements.

Table 4-410 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.83.3 Paths for Software Porting

This chapter lists software installation paths involved in the Nanopolish software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-411 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/nanopolish	Installation path of Nanopolish.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.83.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-412 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

4.83.5 Obtaining the Source Code

Procedure

Step 1 Download the Nanopolish installation package.

```
git clone --recursive https://github.com/jts/nanopolish.git
```

Ensure that the downloaded **recursive** package contains the files **fast5**, **htslib**, **minimap2** and the directories are not empty. If the download speed is slow, you can use other methods to download **fast5**, **htslib**, and **minimap2** and decompress them to the corresponding directories.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

Use the specified branch version, not the latest branch version.

Step 2 Use an SFTP tool to upload the Nanopolish installation package to the */path/to/nanopolish* directory on the server.

----End

4.83.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install lapack lapack-devel blas blas-devel bzip2-devel.aarch64 python-pip zlib-devel.aarch64 xz-devel hdf5-devel libjpeg
```

Step 3 Modify the **Makefile** file.

1. Open the **Makefile** file.
   ```
   vi Makefile
   ```
2. Press **i** to enter the edit mode.
   - Add **ARM ?= 1** to line 18.

     ```
     14 CFLAGS ?= -O3
     15 LDFLAGS ?=
     16 CXX ?= g++
     17 CC ?= gcc
     18 ARM ?= 1
     ```

   - Add **aarch64=1** to line 109.

     ```
     107
     108 minimap2/libminimap2.a:
     109         $(MAKE) -C minimap2 $(MINIMAP2_OPT) aarch64=1 libminimap2.a || exit 255
     110
     ```

   - Added **--build=arm-linux** to the Libhdf5 compilation option.

     ```
     lib/libhdf5.a:
             if [ ! -e hdf5-$(HDF5_VERSION).tar.gz ]; then \
                     version_major_minor=`echo "$(HDF5_VERSION)" | sed -E 's/\.[0-9]+$$//'` \
                     wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8
             fi
             tar -xzf hdf5-$(HDF5_VERSION).tar.gz || exit 255
             cd hdf5-$(HDF5_VERSION) && \
                     ./configure --build=arm-linux --enable-threadsafe --disable-hl --libd
     55
             $(MAKE) -C hdf5-$(HDF5_VERSION) && $(MAKE) -C hdf5-$(HDF5_VERSION) install
     ```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> After performing **Step 3.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 4 Perform the compilation.

```
make
```

Step 5 Install the Python 3 extension components.

```
yum -y install python3 python3-devel
pip3 install wheel
pip3 install -r scripts/requirements.txt --user
```

If a component missing error is reported during the installation, install the component separately:

```
pip install NANOPOLISH
```

----End

4.83.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/nanopolish
```

Step 3 Run Nanopolish.

```
./nanopolish
./nanopolish --version
```

If the following information is displayed, the running is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nanopolish-0.13.2]# ./nanopolish
error: no command provided
usage: nanopolish [command] [options]
  valid commands:
    --help
    --version
    call-methylation
    eventalign
    extract
    getmodel
    help
    index
    methyltrain
    phase-reads
    polya
    scorereads
    variants
    vcf2fasta
  for help on given command, type nanopolish command --help
[root@localhost nanopolish-0.13.2]# ./nanopolish --version
nanopolish version 0.13.2
Written by Jared Simpson.

Copyright 2015-2017 Ontario Institute for Cancer Research
```

----End

4.84 Octave 6.4.0 Porting Guide (CentOS 8.2)

4.84.1 Introduction

GNU Octave is a high-level language primarily intended for numerical computations. It is typically used for such problems as solving linear and nonlinear equations, numerical linear algebra, statistical analysis, and for performing other numerical experiments. It may also be used as a batch-oriented language for automated data processing.

For more information about Octave, visit the **Octave official website**.

Programming language: C++

Brief description: A high-level language.

Open source license: MIT License

Recommended Software Version

Octave 6.4.0

4.84.2 Environment Requirements

Hardware Requirements

Table 4-413 lists the hardware requirements.

Table 4-413 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-414 lists the software requirements.

Table 4-414 Software requirements

Item	Version	Download Address
Octave	6.4.0	**https://mirrors.tuna.tsinghua.edu.cn/gnu/octave/octave-6.4.0.tar.gz**
GCC for openEuler	10.3.1	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz**

OS Requirements

Table 4-415 lists the OS requirements.

Table 4-415 OS requirements

Item	Version	Download URL
CentOS	8.2	**https://vault.centos.org/8.2.2004/isos/aarch64/**
Kernel	4.18.0-193	**https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm**

4.84.3 Paths for Software Porting

This chapter lists software installation paths involved in the Octave software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-416 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/octave	Installation path of Octave.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.

4.84.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-417 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the dependencies	See **4.84.4 Configuring the Compilation Environment**.

4.84.4.1 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install make gcc-gfortran.aarch64 autoconf automake
yum install blas.aarch64 blas-devel.aarch64 lapack.aarch64
yum install lapack-devel.aarch64 pcre pcre-devel readline
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.84.5 Obtaining the Source Code

Procedure

Step 1 Download the Octave installation package.

https://mirrors.tuna.tsinghua.edu.cn/gnu/octave/octave-6.4.0.tar.gz

Step 2 Use an SFTP tool to upload the Octave installation package to the */path/to/octave* directory on the server.

----End

4.84.6 Installing Octave

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the source code to the **/home** directory:

```
cd /home
wget https://mirrors.tuna.tsinghua.edu.cn/gnu/octave/octave-6.4.0.tar.gz
```

Step 3 Decompress the installation package and perform the installation.

```
tar -zxvf octave-6.4.0.tar.gz
cd /home/octave-6.4.0
./configure --prefix=/opt/octave --disable-readline
make
```

```
Octave successfully built.  Now choose from the following:

   ./run-octave    - to run in place to test before installing
   make check      - to run the tests
   make install    - to install (PREFIX=/opt/octave)

   HG ID for this build is "8d7671609955"

make[2]: 离开目录“/home/octave-6.4.0”
make[1]: 离开目录“/home/octave-6.4.0”
```

```
make install
```

Step 4 Set the environment variables.

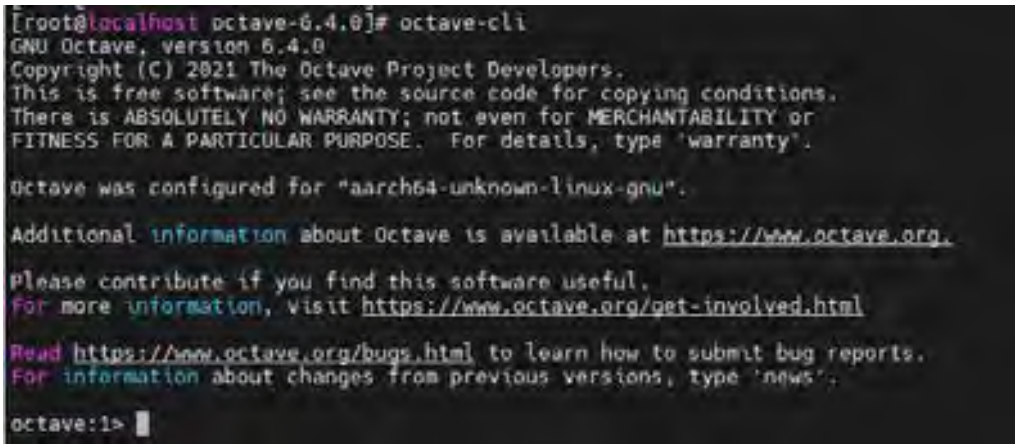
```
export PATH=/path/to/octave/bin:$PATH
export LD_LIBRARY_PATH=/path/to/octave/lib:$LD_LIBRARY_PATH
```

Step 5 Perform the verification.

```
octave-cli
```

If the following information is displayed, the installation is successful:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.85 PanPhlAn 3.1 Porting Guide (openEuler 20.03)

4.85.1 Introduction

Pangenome-based Phylogenomic Analysis (PanPhlAn) is a strain-level metagenomic profiling tool for identifying the gene composition and in-vivo transcriptional activity of individual strains in metagenomic samples. PanPhlAn's ability for strain-tracking and functional analysis of unknown pathogens makes it an efficient tool for culture-free infectious outbreak epidemiology and microbial population studies.

For more information about PanPhlAn, visit the **PanPhlAn official website**.

Programming language: Python

Brief description: A strain-level metagenomic profiling tool.

Open source license: MIT License

Recommended Software Version

PanPhlAn 3.1

4.85.2 Environment Requirements

Hardware Requirements

Table 4-418 lists the hardware requirements.

Table 4-418 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-419 lists the software requirements.

Table 4-419 Software requirements

Item	Version	Download Address
Python	3.7.9	https://www.python.org/ftp/python/3.7.9/Python-3.7.9.tgz
PanPhlAn	3.1	https://github.com/SegataLab/panphlan
Bowtie2	2.41	https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip
Samtools	1.11	https://sourceforge.net/projects/samtools/files/samtools/1.11/samtools-1.11.tar.bz2

OS Requirements

Table 4-420 lists the OS requirements.

Table 4-420 OS requirements

Item	Version	Download URL
openEuler	20.03	https://www.openeuler.org/en/download/

4.85.3 Paths for Software Porting

This chapter lists software installation paths involved in the PanPhlAn software porting.

Table 4-421 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/python3.7.9	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this
3	/path/to/Bowtie2 2.41	Installation path of Bowtie2.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
4	/path/to/samtools1.11	Installation path of Samtools.	document need to be replaced with the actual paths planned during the installation process.
5	/path/to/panphlan	Installation path of PanPhlAn.	

4.85.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-422 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
4	Installing Bowtie2	See **Bowtie2 2.4.1 Porting Guide (CentOS 7.6)**.
5	Installing Samtools	See **CNVnator 0.4.1 Porting Guide (CentOS 7.6)**.

NOTE

The **configure: error: liblzma development files not found** error may be reported during Samtools installation. Run the following command to rectify the error:

```
yum -y install xz-devel
```

4.85.5 Obtaining the Source Code

Procedure

Step 1 Download the PanPhlAn installation package.

https://github.com/SegataLab/panphlan

Step 2 Use an SFTP tool to upload the PanPhlAn installation package to the */path/to/panphlan* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.85.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
pip3 install numpy
pip3 install scipy
pip3 install pandas
```

Step 3 Go to the main program installation directory.

```
cd /path/to/panphlan
```

Step 4 Decompress the installation package.

```
tar -zxvf panphlan-3.1.tar.gz
```

----End

4.85.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory generated after the decompression.

```
cd panphlan-3.1
```

Step 3 Download the macrogenome sample.

```
wget https://www.dropbox.com/s/oi26jg0v7ktlavc/panphlan_tutorial_samples.tar.bz2
```

Step 4 Decompress the sample file.

```
tar -xvjf panphlan_tutorial_samples.tar.bz2
```

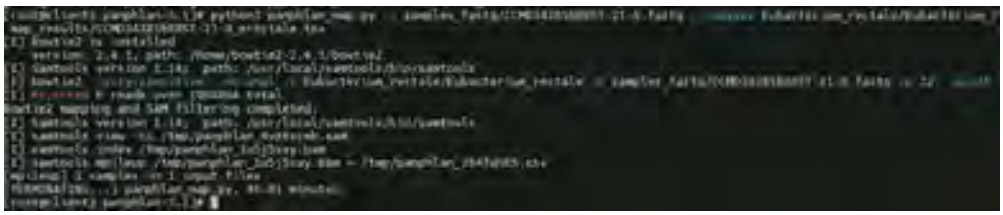
Step 5 Download the reference genome.

```
python3 panphlan_download_pangenome.py -i Eubacterium_rectale
```

Step 6 Map the sample to the pan-genome.

```
mkdir map_results/
python3 panphlan_map.py -i samples_fastq/CCMD34381688ST-21-0.fastq --indexes
Eubacterium_rectale/Eubacterium_rectale  -p Eubacterium_rectale/
Eubacterium_rectale_pangenome.tsv -o map_results/CCMD34381688ST-21-0_erectale.tsv
```

If the following information is displayed, the operation is successful:



Step 7 Analyze the strain.

```
python3 panphlan_profiling.py -i map_results --o_matrix result_profile_erectale.tsv -p
Eubacterium_rectale/Eubacterium_rectale_pangenome.tsv --add_ref
```

If the following information is displayed, the operation is successful:

 Copyright © Huawei Technologies Co., Ltd.

Step 8 View the generated file.

```
cat result_profile_erectale.tsv
```

----End

4.86 PhyloPhlAn 3.0.60 Porting Guide (CentOS 8.2)

4.86.1 Introduction

PhyloPhlAn is an integrated pipeline for large-scale phylogenetic profiling of genomes and metagenomes. PhyloPhlAn is an accurate, rapid, and easy-to-use method for large-scale microbial genome characterization and phylogenetic analysis at multiple levels of resolution. PhyloPhlAn can assign both genomes and metagenome-assembled genomes (MAGs) to species-level genome bins (SGBs). PhyloPhlAn can reconstruct strain-level phylogenies using clade-specific maximally informative phylogenetic markers, and can also scale to very-large phylogenies comprising >17,000 microbial species.

For more information about PhyloPhlAn, visit the **PhyloPhlAn official website**.

Programming languages: Python, Perl

Brief description: PhyloPhlAn is an integrated pipeline for large-scale phylogenetic profiling of genomes and metagenomes.

Open source license: MIT License

Recommended Software Version

PhyloPhlAn 3.0.60

4.86.2 Environment Requirements

Hardware Requirements

Table 4-423 lists the hardware requirements.

Table 4-423 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-424 lists the software requirements.

Table 4-424 Software requirements

Item	Version	Download Address
Python	3.9.6	https://www.python.org/downloads/release/python-396/

OS Requirements

Table 4-425 lists the OS requirements.

Table 4-425 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.86.3 Paths for Software Porting

This chapter lists software installation paths involved in the PhyloPhlAn software porting.

Table 4-426 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/python3.9.6	Installation path of Python.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

4.86.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-427 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Python	See **4.86.4.1 Installing Python**.

4.86.4.1 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package and go to the directory.

```
tar -xvf Python-3.9.6.tgz
cd Python-3.9.6
```

Step 3 Install the dependencies.

```
yum install gcc libffi* openssl libssl zlib* bzip2
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/python3.9.6 --enable-shared --enable-optimizations
make
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Set the environment variables.

```
export PATH=/path/to/python3.9.6/bin:$PATH
export LD_LIBRARY_PATH=/path/to/python3.9.6/lib:$LD_LIBRARY_PATH
```

Step 6 Verify the version.

```
python3 --version
pip3 --version
```

If a Python version number is displayed in the command output, the installation is successful.

```
[root@ceph3 home]# python3 --version
Python 3.9.6
[root@ceph3 home]# pip3 --version
pip 21.1.3 from /usr/local/lib/python3.9/site-packages/pip (python 3.9)
[root@ceph3 home]#
```

----End

4.86.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install PhyloPhlAn.

```
pip3 install phylophlan
```

----End

4.86.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
phylophlan --version
```

If a PhyloPhlAn version number is displayed in the command output, the installation is successful.

```
[root@localhost python3]# phylophlan --version
PhyloPhlAn version 3.0.60 (27 November 2020)
[root@localhost python3]#
```

----End

4.87 PyClone 0.13.1 Porting Guide (CentOS 8.2)

4.87.1 Introduction

PyClone is a probabilistic model for inferring clonal population structure from deep NGS sequencing. PyClone is a Bayesian clustering method for grouping sets

of deeply sequenced somatic mutations into putative clonal clusters while estimating their cellular prevalences and accounting for allelic imbalances introduced by segmental copy-number changes and normal-cell contamination. Single-cell sequencing validation demonstrates PyClone's accuracy.

For more information about PyClone, visit the **PyClone official website**.

Programming language: Python

Brief description: PyClone is a probabilistic model for inferring clonal population structure from deep NGS sequencing.

Open-source license: View license

Recommended Software Version

PyClone 0.13.1

4.87.2 Environment Requirements

Hardware Requirements

Table 4-428 lists the hardware requirements.

Table 4-428 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-429 lists the software requirements.

Table 4-429 Software requirements

Item	Version	Download Address
PyClone	0.13.1	**https://github.com/Roth-Lab/pyclone/archive/refs/tags/0.13.1.tar.gz**
Anaconda	2021.05	**https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh**
PyDP	0.2.4	**https://codeload.github.com/Roth-Lab/pydp/tar.gz/refs/tags/0.2.4**
Numba	0.45.0	**https://files.pythonhosted.org/packages/7e/89/853a1f03b09f1b13b59c3d785678b47daac6ddd24a285f146d09bb723b85/numba-0.45.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
LLVM	8.0.0	http://releases.llvm.org/8.0.0/llvm-8.0.0.src.tar.xz
llvmlite	0.30.0	https://files.pythonhosted.org/packages/8b/b0/df26861e6ce2fc91c8bb93ea808fa2e631ee8a29fc4c3bc96626b78dae74/llvmlite-0.30.0.tar.gz

OS Requirements

Table 4-430 lists the OS requirements.

Table 4-430 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.87.3 Paths for Software Porting

This chapter lists software installation paths involved in the PyClone software porting.

Table 4-431 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in Basic Environment Setup Guide.
2	/path/to/pyclone	Installation path of PyClone.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/root/anaconda3	Installation path of Anaconda.	
4	/path/to/pydp	Installation path of PyDP.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	/path/to/numba	Installation path of Numba.	
6	/path/to/llvm	Installation path of LLVM.	
7	/path/to/llvmlite	Installation path of llvmlite.	

4.87.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-432 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Anaconda	See **4.87.4.1 Installing Anaconda**.
3	Installing PyDP	See **4.87.4.2 Installing PyDP**.
4	Installing Numba	See **4.87.4.3 Installing Numba**.
5	Installing LLVM	See **4.87.4.4 Installing LLVM**.
6	Installing llvmlite	See **4.87.4.5 Installing llvmlite**.

4.87.4.1 Installing Anaconda

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download Anaconda.

```
wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh
```

Step 3 Install Anaconda.

```
bash Anaconda3-2021.05-Linux-aarch64.sh
```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> The default installation directory is **/root/anaconda3**. Press **Enter** and type **yes** as prompted to continue the installation. After the installation is successful, the system displays "Thank you for installing Anaconda3!".

Step 4 Restart the Shell terminal to initialize the Anaconda environment.

Step 5 Check whether the Anaconda environment configuration is initialized successfully.

```
echo $PATH
```

If the following information is displayed at the beginning of the command output, the initialization is successful:

```
/root/anaconda3/bin:/root/anaconda3/condabin:
```

Step 6 Create a PyClone environment.

```
conda create -n pyclone
```

Step 7 Activate the PyClone environment.

```
conda activate pyclone
```

Step 8 Install the dependencies.

```
conda install python==2.7.15 -c conda-forge
pip install numpy  --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
pip install matplotlib --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
conda install pandas -c conda-forge
conda install pyyaml -c conda-forge
pip install scipy  --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
pip install seaborn  --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
conda install singledispatch -c conda-forge
pip install enum34 --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
pip install funcsigs --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

----End

4.87.4.2 Installing PyDP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package and use an SFTP tool to upload it to the server.

Download address: **https://codeload.github.com/Roth-Lab/pydp/tar.gz/refs/tags/0.2.4**

Step 3 Decompress the installation package.

```
tar xvf pydp-0.2.4.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd pydp-0.2.4
```

Step 5 Install PyDP.

```
python setup.py install --single-version-externally-managed --record=pydprecord.txt
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the verification.

```
conda list pydp
```

If the following information is displayed, the installation is successful:

```
# packages in environment at /root/anaconda3/envs/pyclone:
#
# Name                    Version                   Build  Channel
pydp                      0.2.4                    pypi_0    pypi
```

----End

4.87.4.3 Installing Numba

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download Numba.

```
wget https://files.pythonhosted.org/packages/7e/89/853a1f03b09f1b13b59c3d785678b47daac6ddd24a285f146d09bb723b85/numba-0.45.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf numba-0.45.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd numba-0.45.0
```

Step 5 Install Numba.

```
python setup.py install --single-version-externally-managed --record=numbarecord.txt
```

Step 6 Perform the verification.

```
conda list numba
```

If the following information is displayed, the installation is successful:

```
# packages in environment at /root/anaconda3/envs/pyclone2.7:
#
# Name                    Version                   Build  Channel
numba                     0.45.0                   pypi_0    pypi
```

----End

4.87.4.4 Installing LLVM

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download LLVM.

```
wget http://releases.llvm.org/8.0.0/llvm-8.0.0.src.tar.xz
```

Step 3 Decompress LLVM.

```
tar xvf llvm-8.0.0.src.tar.xz
```

Step 4 Go to the directory generated after the decompression.

```
cd llvm-8.0.0.src/
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Go to the **llvm-8.0.0.src/tools** directory and obtain the Clang source code.

```
cd tools/
wget http://releases.llvm.org/8.0.0/cfe-8.0.0.src.tar.xz
```

Step 6 Decompress **cfe-8.0.0.src.tar.xz** and rename the directory.

```
tar vxf cfe-8.0.0.src.tar.xz
mv cfe-8.0.0.src clang
```

Step 7 Go to the **llvm-8.0.0.src/tools** directory and obtain the compiler-rt source code.

```
cd ../projects/
wget http://releases.llvm.org/8.0.0/compiler-rt-8.0.0.src.tar.xz
```

Step 8 Decompress **compiler-rt-8.0.0.src.tar.xz** and rename the directory.

```
tar vxf compiler-rt-8.0.0.src.tar.xz
mv compiler-rt-8.0.0.src compiler-rt
```

Step 9 Return to the **llvm-8.0.0.src** directory and create a **build** directory.

```
cd ..
mkdir build
```

Step 10 Go to the **build** directory.

```
cd build/
```

Step 11 Perform the compilation and installation.

```
cmake -G "Unix Makefiles" -Wno-dev -DCMAKE_BUILD_TYPE=Release ../
make -j32
make install
```

----End

4.87.4.5 Installing llvmlite

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download llvmlite.

```
wget https://files.pythonhosted.org/packages/8b/b0/
df26861e6ce2fc91c8bb93ea808fa2e631ee8a29fc4c3bc96626b78dae74/llvmlite-0.30.0.tar.gz
```

Step 3 Install llvmlite.

```
CXXFLAGS=-fPIC python setup.py install --single-version-externally-managed --
record=llvmliterecord.txt
```

Step 4 Perform the verification.

```
conda list llvmlite
```

If the following information is displayed, the installation is successful:

```
# packages in environment at /root/anaconda3/envs/pyclone2.7:
#
# Name                    Version                   Build  Channel
llvmlite                  0.30.0                   pypi_0    pypi
```

----End

4.87.5 Obtaining the Source Code

Procedure

Step 1 Download the PyClone installation package.

https://codeload.github.com/Roth-Lab/pyclone/tar.gz/refs/tags/0.13.1

Step 2 Use an SFTP tool to upload the PyClone installation package to the */path/to/pyclone* directory on the server.

----End

4.87.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xvf pyclone-0.13.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd pyclone-0.13.1
```

Step 4 Install PyClone.

```
python setup.py install --single-version-externally-managed --record=pyclonerecord.txt
```

Step 5 Perform the verification.

```
conda list pyclone
```

Information similar to the following is displayed:

```
# packages in environment at /root/anaconda3/envs/pyclone:
#
# Name                    Version                   Build  Channel
pyclone                   0.13.1                    pypi_0    pypi
```

Step 6 Perform the verification.

```
PyClone --help
```

Information similar to the following is displayed:

----End

 Copyright © Huawei Technologies Co., Ltd.

4.87.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd pyclone-0.13.1/examples/mixing/tsv/
```

Step 3 Perform the test.

```
PyClone run_analysis_pipeline --in_files SRR385938.tsv SRR385939.tsv SRR385940.tsv SRR385941.tsv --working_dir pyclone_analysis
```

Information similar to the following is displayed:

```
Iteration: 9400
Number of clusters: 9
DP concentration: 1.97853370828
Beta-Binomial precision: 486.879217795

Iteration: 9500
Number of clusters: 9
DP concentration: 2.9766452168
Beta-Binomial precision: 524.198666803

Iteration: 9600
Number of clusters: 9
DP concentration: 1.44941284175
Beta-Binomial precision: 540.761555191

Iteration: 9700
Number of clusters: 9
DP concentration: 3.63858615949
Beta-Binomial precision: 548.07839628

Iteration: 9800
Number of clusters: 9
DP concentration: 3.76142669714
Beta-Binomial precision: 522.394514251
```

Check whether the **pyclone_analysis** directory contains four subdirectories and the **config.yaml** file.

```
(pyclone2.7) [root@2280node27-test tsv]#  ll pyclone_analysis/
total 4
-rw-r--r-- 1 root root 932 Nov 25 09:57 config.yaml
drwxr-xr-x 4 root root  33 Nov 25 10:44 plots
drwxr-xr-x 2 root root  41 Nov 25 10:44 tables
drwxr-xr-x 2 root root 254 Nov 25 09:57 trace
drwxr-xr-x 2 root root  94 Nov 25 09:57 yaml
(pyclone2.7) [root@2280node27-test tsv]#
```

----End

4.88 pyem v0.5 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

4.88.1 Introduction

UCSF pyem is a collection of Python modules and command-line utilities for electron microscopy of biological samples.

For more information about pyem, visit the **pyem official website**.

Programming language: Python

Brief description: A collection of Python modules and command-line utilities for electron microscopy of biological samples.

Open source license: GPL-3.0

Recommended Software Version

pyem v0.5

4.88.2 Environment Requirements

Hardware Requirements

Table 4-433 lists the hardware requirements.

Table 4-433 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-434 lists the software requirements.

Table 4-434 Software requirements

Item	Version	Download Address
pyem	v0.5	**https://codeload.github.com/asarnow/pyem/tar.gz/refs/tags/v0.5**
Anaconda	2021.05	**https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh**

OS Requirements

Table 4-435 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-435 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.88.3 Paths for Software Porting

This chapter lists software installation paths involved in the pyem software porting.

Table 4-436 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/root/anaconda3	Installation path of Anaconda.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/pyem	Installation path of pyem.	
4	/path/to/fftw	Installation path of FFTW.	

4.88.4 Configuring the Compilation Environment

Configure Process

Table 4-437 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing Anaconda	See **4.88.4.1 Installing Anaconda**.

 Copyright © Huawei Technologies Co., Ltd.

4.88.4.1 Installing Anaconda

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Anaconda installation package.

```
wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh
```

Step 3 Install Anaconda.

```
bash Anaconda3-2021.05-Linux-aarch64.sh
```

NOTE

The default installation directory is **/root/anaconda3**. Press **Enter** and type **yes** as prompted to continue the installation. After the installation is successful, the system displays "Thank you for installing Anaconda3!".

Step 4 Restart the Shell terminal to initialize the environment.

Step 5 Check whether the Anaconda environment configuration is initialized successfully.

```
echo $PATH
```

If the following information is displayed at the beginning of the command output, the initialization is successful:

```
/root/anaconda3/bin:/root/anaconda3/condabin:
```

----End

4.88.5 Obtaining the Source Code

Procedure

Step 1 Download the pyem installation package.

https://codeload.github.com/asarnow/pyem/tar.gz/refs/tags/v0.5

Step 2 Use an SFTP tool to upload the pyem installation package to the */path/to/pyem* directory on the server.

----End

4.88.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create the dependencies for installing the conda environment:

```
conda create --name pyem --clone base
conda activate pyem
```

Step 3 Install the FFTW dependencies.

```
yum install fftw3 --installroot=/root/anaconda3/envs/pyem/lib/python3.8/site-packages
wget http://fftw.org/fftw-3.3.8.tar.gz
tar xvf fftw-3.3.8.tar.gz
cd fftw-3.3.8
./configure --prefix=/root/anaconda3/envs/pyem/lib/python3.8/site-packages/FFTW --enable-shared --
```

 Copyright © Huawei Technologies Co., Ltd.

```
enable-static --enable-fma --enable-neon
make -j 32
make install
pip3 install pyfftw --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Step 4 Install the dependencies of healpy, pathos, natsort, and numpy.

```
conda update --all
conda install -c conda-forge healpy pathos
conda install natsort numpy==1.20.1
```

Step 5 Decompress the pyem installation package.

```
tar xvf pyem-0.5.tar.gz
```

Step 6 Go to the directory generated after the decompression.

```
cd pyem-0.5
```

Step 7 Install pyem.

```
pip3 install --no-dependencies -e .
```

NOTE

After the installation is successful, the message "Successfully installed pyem" is displayed.

Step 8 Modify the **projection_subtraction.py** file.

1. Open the **projection_subtraction.py** file.
   ```
   vi projection_subtraction.py
   ```
2. Press **i** to enter the edit mode and delete **version="projection_subtraction.py 2.1b"** from line 291.

   Before the modification:

   ```
   291 parser = argparse.ArgumentParser(version="projection_subtraction.py 2.1b")
   ```

   After the modification:

   ```
   291 parser = argparse.ArgumentParser()
   ```

   NOTE

   After performing Step 8.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

----End

4.88.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Open a new Shell terminal and activate the pyem environment.

```
conda activate pyem
```

Step 3 Go to the **pyem** directory.

```
cd pyem-0.5
```

Step 4 Check whether the **projection_subtraction.py** file is generated.

```
ls
```

If the following information is displayed, the file has been generated:

 Copyright © Huawei Technologies Co., Ltd.

----End

4.89 PyTorch 1.9 Porting Guide (CentOS 8.2)

4.89.1 Introduction

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing. It was developed by Facebook's AI Research lab. A number of pieces of deep learning software are built on top of PyTorch, including Uber's Pyro.

For more information about PyTorch, visit the **PyTorch official website**.

Programming languages: Python, C++, and CUDA

Brief description: An open-source Python machine learning library.

Open source license: custom open source license

Recommended Software Version

PyTorch 1.9

4.89.2 Environment Requirements

Hardware Requirements

Table 4-438 lists the hardware requirements.

Table 4-438 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla V100

Software Requirements

Table 4-439 lists the software requirements.

Table 4-439 Software requirements

Item	Version	Download Address
Docker	19.03.15	**https://download.docker.com/linux/static/stable/aarch64/docker-19.03.15.tgz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download Address
NVIDIA Container Toolkit	Installed using Yum	https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html
PyTorch	1.9	https://github.com/pytorch/pytorch
Cuda	11.1	https://registry.hub.docker.com/r/nvidia/cuda-arm64/tags?page=1&ordering=last_updated&name=11.1
Cudnn	8	https://developer.download.nvidia.cn/compute/machine-learning/repos/rhel8/sbsa/

OS Requirements

Table 4-440 lists the OS requirements.

Table 4-440 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.89.3 Paths for Software Porting

This chapter lists software installation paths involved in the PyTorch software porting.

Table 4-441 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Planning the Installation Paths" in Basic Environment Setup Guide.
2	/path/to/pytorch	Installation path of PyTorch.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.89.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-442 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

Configuring the Docker Environment

See **Docker 19.03.10 Installation Guide (CentOS & openEuler)**. This document uses Docker 19.03.15 as an example.

4.89.5 Obtaining the Source Code

Procedure

Step 1 Download the PyTorch installation package.

https://github.com/pytorch/pytorch

Step 2 Use an SFTP tool to upload the PyTorch installation package to the */path/to/pytorch* directory on the server.

----End

4.89.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the Docker toolkit.

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
yum-config-manager --add-repo https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo
dnf clean expire-cache --refresh
dnf install -y nvidia-docker2
systemctl restart docker
```

Step 3 Obtain the base image.

```
docker pull nvidia/cuda-arm64:11.1-devel-centos8
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Run Docker.

```
docker run -it nvidia/cuda-arm64:11.1-devel-centos8 /bin/bash
```

Step 5 Modify the **/etc/yum.repos.d/nvidia-ml.repo** file.

1. Open the **/etc/yum.repos.d/nvidia-ml.repo** file.
   ```
   vi /etc/yum.repos.d/nvidia-ml.repo
   ```
2. Press **i** to enter the edit mode and add the following information in bold:
   ```
   [nvidia-ml]
   name=nvidia-ml
   baseurl=https://developer.download.nvidia.com/compute/machine-learning/repos/rhel8/sbsa
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA
   sslverify=false
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 6 Modify the **/etc/yum.repos.d/cuda.repo** file.

1. Open the **/etc/yum.repos.d/cuda.repo** file.
   ```
   vi /etc/yum.repos.d/cuda.repo
   ```
2. Press **i** to enter the edit mode and add the following information in bold:
   ```
   [cuda]
   name=cuda
   baseurl=https://developer.download.nvidia.com/compute/cuda/repos/rhel8/sbsa
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA
   sslverify=false
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the settings and exit.

Step 7 Refresh the Yum source cache.

```
yum makecache
```

Step 8 Install the system dependencies.

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel python3 python3-devel gcc wget cmake3 libarchive git which
```

Step 9 Download and install cudnn8.

```
wget https://developer.download.nvidia.cn/compute/machine-learning/repos/rhel8/sbsa/libcudnn8-8.0.4.30-1.cuda11.1.aarch64.rpm --no-check-certificate
rpm -ivh libcudnn8-8.0.4.30-1.cuda11.1.aarch64.rpm
wget https://developer.download.nvidia.cn/compute/machine-learning/repos/rhel8/sbsa/libcudnn8-devel-8.0.4.30-1.cuda11.1.aarch64.rpm --no-check-certificate
rpm –ivh libcudnn8-devel-8.0.4.30-1.cuda11.1.aarch64.rpm
```

Step 10 Deploy Anaconda.

NOTE

When executing the **./Anaconda3-2021.05-Linux-aarch64.sh** script, you need to type **Enter** and **yes**. The script is deployed in the **/root/anaconda3** directory by default.

```
wget https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-aarch64.sh
chmod +x Anaconda3-2021.05-Linux-aarch64.sh
./Anaconda3-2021.05-Linux-aarch64.sh
export PATH=/root/anaconda3/bin/:$PATH
```

Step 11 Install the PyTorch dependencies.

```
conda install astunparse numpy ninja pyyaml setuptools cmake cffi typing_extensions future six requests dataclasses
```

(Optional) Configure the conda proxy as follows:

```
cat > /root/.condarcexport PATH=/root/anaconda3/bin:$PATH
channels:
```

 Copyright © Huawei Technologies Co., Ltd.

```
  - conda-forge
  - defaults
proxy_servers:
  http: xxxxxx
  https: xxxxxx
ssl_verify: false
```

Step 12 Set the environment variables.

```
export PATH=/root/anaconda3/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64
```

Step 13 Download and compile the PyTorch source code.

```
export GIT_SSL_NO_VERIFY=1
git clone --recursive https://github.com/pytorch/pytorch  --depth=1
cd pytorch
git submodule sync
git submodule update --init --recursive --jobs 0
export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-"$(dirname $(which conda))/../"}
python setup.py install
```

Step 14 Create another terminal shell to export the container.

```
docker ps
```

Move large files such as the PyTorch source code out of the container, delete the files, and export the image.

1. Back up the source code.
   ```
   docker cp 4ad81495b7ef:/pytorch .
   ```
2. Delete the PyTorch source code from the container.
   ```
   rm -rf pytorch/
   rm -rf Anaconda3-2021.05-Linux-aarch64.sh
   rm -f libcudnn8-8.0.4.30-1.cuda11.1.aarch64.rp
   rm -f libcudnn8-devel-8.0.4.30-1.cuda11.1.aarch64.rpm
   ```
3. Export the image.
   ```
   docker export -o pytorch_cuda.tar 4ad81495b7ef
   ```

----End

4.89.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Start Docker.

```
docker import pytorch_cuda.tar pytorch/cuda:v3
docker run --gpus all  -it pytorch/cuda:v3 bash
```

Step 3 Set the environment variables.

```
export PATH=/root/anaconda3/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64
```

Step 4 Start Python 3.

```
python3
import torch
torch.cuda.is_available()
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@69369871f558 /]# python3
Python 3.8.8 (default, Apr 23 2021, 03:28:49)
[GCC 10.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
```

----End

4.90 SURVIVOR 1.0.6 Porting Guide (CentOS 8.2)

4.90.1 Introduction

SURVIVOR is a tool set for simulating/evaluating SVs, merging and comparing SVs within and among samples, and includes various methods to reformat or summarize SVs.

For more information about SURVIVOR, visit the **SURVIVOR official website**.

Programming language: C++

Brief description: A tool set for simulating/evaluating SVs, merging and comparing SVs within and among samples.

Open source license: GPLv3

Recommended Software Version

SURVIVOR v1.0.6

4.90.2 Environment Requirements

Hardware Requirements

Table 4-443 lists the hardware requirements.

Table 4-443 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-444 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-444 Software requirements

Item	Version	Download Address
SURVIVOR	1.0.6	https://github.com/fritzsedlazeck/SURVIVOR/archive/refs/tags/v1.0.6.tar.gz

OS Requirements

Table 4-445 lists the OS requirements.

Table 4-445 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/

4.90.3 Paths for Software Porting

This chapter lists software installation paths involved in the SURVIVOR software porting.

Table 4-446 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/home/SURVIVOR	Installation path of SURVIVOR.	The installation paths provided here are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.

 Copyright © Huawei Technologies Co., Ltd.

4.90.4 Configuring the Compilation Environment

Configure Process

Table 4-447 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing GCC and CMake	See **4.90.4.1 Installing GCC and CMake**.

4.90.4.1 Installing GCC and CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC and CMake.

```
yum install gcc-c++
yum install cmake
```

Step 3 Perform the verification.

```
gcc -v
cmake -version
```

----End

4.90.5 Obtaining the Source Code

Procedure

Step 1 Download the SURVIVOR installation package.

```
cd /home
wget https://github.com/fritzsedlazeck/SURVIVOR/archive/refs/tags/v1.0.6.tar.gz
```

Step 2 Use an SFTP tool to upload the SURVIVOR installation package to the **/home/SURVIVOR** directory on the server.

----End

4.90.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **/home** directory.

```
cd /home
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the SURVIVOR installation package.

```
tar -zxvf v1.0.6.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd SURVIVOR-1.0.6/Debug
```

Step 5 Perform the installation.

```
make
```

----End

4.90.7 Running and Verification

Step 1 Use PuTTY to log in to the server as the root user.

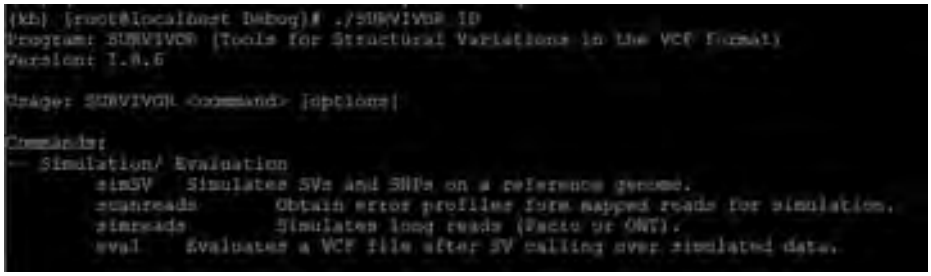
Step 2 Go to the installation directory.

```
cd /home/SURVIVOR-1.0.6/Debug
```

Step 3 Perform the verification.

```
./SURVIVOR ID
```

If the following information is displayed, the installation is successful:



----End

4.91 RELION 3.1.2 Porting Guide (CentOS 8.2)

4.91.1 Introduction

Single particle cryo-electron microscopy (cryo-EM) is one of the most important methods of macromolecular structure. The Regularized Likelihood Optimization (RELION) implements a Bayesian algorithm for cryo-EM structure determination, which is one of the most widely used software in this field.

For more information about RELION, visit the **RELION official website**.

Programming languages: C++, CUDA

Brief description: RELION is a piece of cryo-EM image processing software.

Open source license: GPLv3

Recommended Software Version

RELION 3.1.2

 Copyright © Huawei Technologies Co., Ltd.

4.91.2 Environment Requirements

Hardware Requirements

Table 4-448 lists the hardware requirements.

Table 4-448 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 4-449 lists the software requirements.

Table 4-449 Software requirements

Item	Version	Download Address
RELION	3.1.2	https://github.com/3dem/relion/archive/refs/tags/3.1.2.tar.gz
BiSheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://www.hikunpeng.com/document/detail/zh/kunpenghpcs/hypermpi/userg_huaweimpi_0010.html
FFTW	3.3.8	https://fftw.org/pub/fftw/fftw-3.3.8.tar.gz
CMake	3.8.1	https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
Test cases	relion_benchmark.tar.gz	ftp://ftp.mrc-lmb.cam.ac.uk/pub/scheres/relion_benchmark.tar.gz

OS Requirements

Table 4-450 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-450 OS requirements

Item	Version	Download URL
CentOS	8.2	https://vault.centos.org/8.2.2004/isos/aarch64/
Kernel	4.18.0-193	https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm

4.91.3 Paths for Software Porting

This chapter lists software installation paths involved in the RELION software porting.

Table 4-451 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Planning the Installation Paths" in **Basic Environment Setup Guide**.
2	/path/to/bisheng-1.3.3	Installation path of BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/HMPI	Installation path of Hyper MPI.	
4	/path/to/NVIDIA-CUDA	Installation path of NVIDIA CUDA.	
5	/path/to/FFTW	Installation path of FFTW.	
6	/path/to/CMAKE	Installation path of CMake.	
7	path/to/wxWidgets	Installation path of wxWidgets.	
8	path/to/CTFFIND	Installation path of CTFFIND.	
9	/path/to/RELION	Installation path of RELION.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
10	/path/to/CASE	Path of RELION test cases.	

4.91.4 Configuring the Compilation Environment

4.91.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-452 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing BiSheng Compiler	See **Installing and Using BiSheng Compiler**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI User Guide**.
4	Disabling the Nouveau Driver	See **4.91.4.2 Disabling the Nouveau Driver**.
5	Installing NVIDIA CUDA	See **4.91.4.3 Installing NVIDIA CUDA**.
6	Installing FFTW	See **4.91.4.4 Installing FFTW**.
7	Installing CMake	See **4.91.4.5 Installing CMake**.
8	Installing wxWidgets	See **4.91.4.6 Installing wxWidgets**.
9	Installing CTFFIND	See **4.91.4.7 Installing CTFFIND**.

 Copyright © Huawei Technologies Co., Ltd.

4.91.4.2 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

4.91.4.3 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3** to **Step 5**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the verification.

```
nvcc --version
```

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

4.91.4.4 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the FFTW installation package.

```
wget https://fftw.org/pub/fftw/fftw-3.3.8.tar.gz
```

Step 3 Decompress the FFTW installation package.

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 4 Go to the FFTW installation package directory.

```
cd fftw-3.3.8
```

Step 5 Install FFTW.

```
export CC=clang CXX=clang++ FC=flang
./bootstrap.sh
./configure --prefix=/path/to/FFTW --enable-single --enable-float --enable-neon --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
make clean
./configure --prefix=/path/to/FFTW --enable-long-double --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
make clean
./configure --prefix=/path/to/FFTW --enable-shared --enable-threads --enable-openmp --enable-mpi CFLAGS="-O3 -fomit-frame-pointer -fstrict-aliasing"
make -j && make install
```

Step 6 Set the environment variables.

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

4.91.4.5 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the CMake installation package.

```
wget https://cmake.org/files/v3.8/cmake-3.8.1.tar.gz
```

Step 3 Decompress the CMake installation package.

```
tar zxvf cmake-3.8.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cmake-3.8.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the configuration.

```
./configure --prefix=/path/to/CMAKE
```

Step 6 Perform the compilation and installation.

```
make -j64
make install
```

Step 7 Set the CMake environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.91.4.6 Installing wxWidgets

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the wxWidgets installation package.

```
wget https://github.com/wxWidgets/wxWidgets/archive/refs/tags/v3.0.5.tar.gz
```

Step 3 Decompress the wxWidgets installation package.

```
tar -zxvf v3.0.5.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd wxWidgets-3.0.5
```

Step 5 Perform the configuration.

```
./configure --disable-gui
```

Step 6 Perform the compilation and installation.

```
make -j16
make install
```

Step 7 Set the environment variables.

```
ldconfig
```

Step 8 Perform the verification.

```
wx-config --version
```

----End

4.91.4.7 Installing CTFFIND

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the CTFFIND installation package.

```
wget https://grigoriefflab.umassmed.edu/system/tdf?
path=ctffind-4.1.14.tar.gz&file=1&type=node&id=26
```

Step 3 Decompress the CTFFIND installation package.

```
tar -zxvf  ctffind-4.1.14.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd ctffind-4.1.14
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Modify the **src/core/matrix.cpp** file.

1. Open **src/core/matrix.cpp**.
   ```
   vi src/core/matrix.cpp
   ```
2. Press **i** to enter the insert mode and modify the content in line 7:

   Before the modification:

   ```
   7  #define _AL_SINCOS(x, s, c)  __asm__ ("fsincos" : "=t" (c), "=u" (s) : "0" (x))
   ```

   After the modification:

   ```
   7 #define _AL_SINCOS(x, s, c)   s = sinf(x);  c = cosf(x);
   ```

   NOTE

   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
yum install -y libjpeg* libtiff* fftw*
autoreconf -f -i
./configure --prefix=/path/to/ctffind --disable-mkl --enable-openmp --enable-debugmode
```

Step 7 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and change **-O2** in **CFLAGS CPPFLAGS CXXFLAGS** to **-O0**.

   Before the modification:

   ```
   296 CFLAGS = -g -O2
   297 CPP = clang -E
   298 CPPFLAGS = -O2 -g -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -DDEBUG
   299 CXX = clang++
   300 CXXCPP = clang++ -E
   301 CXXDEPMODE = depmode=gcc3
   302 CXXFLAGS = -O2 -g -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -DDEBUG  -fopenmp
   ```

   After the modification:

   ```
   296 CFLAGS = -g –O0
   297 CPP = clang -E
   298 CPPFLAGS = -O0 -g -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -DDEBUG
   299 CXX = clang++
   300 CXXCPP = clang++ -E
   301 CXXDEPMODE = depmode=gcc3
   302 CXXFLAGS = -O0 -g -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -DDEBUG  -fopenmp
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Perform the compilation and installation.

```
make & make install
```

Step 9 Set the environment variables.

```
export PATH=/path/to/install/ctffind/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.91.5 Obtaining the Source Code

Procedure

Step 1 Download the RELION installation package.

https://github.com/3dem/relion/archive/refs/tags/3.1.2.tar.gz

Step 2 Use an SFTP tool to upload the RELION installation package to the */path/to/RELION* directory on the server.

----End

4.91.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/RELION
```

Step 3 Decompress the installation package.

```
tar -xvf 3.1.2.tar.gz
```

Step 4 Go to the directory generated after the decompression and create a **build** directory.

```
cd relion-3.1.2
mkdir build
```

Step 5 Perform the configuration.

```
yum install -y libtiff-* fltk* fltk-devel*
yum -y install xorg-x11-xauth
cd build
export FFTW_HOME=/path/to/FFTW
export FFTW_LIB=$FFTW_HOME/lib
export FFTW_INCLUDE=$FFTW_HOME/include
cmake  -DBUILD_SHARED_LIBS=OFF -DCUDA=ON  -DCudaTexture=ON -DCUDA_ARCH=80 -DCMAKE_BUILD_TYPE=release  -DCMAKE_INSTALL_PREFIX=/path/to/install/RELION  ..
```

NOTE

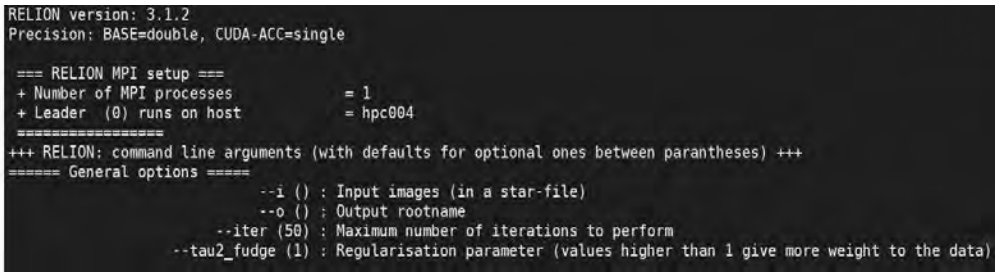
The **-DCUDA_ARCH** value must match the computing capability of the graphics card. The minimum value is 35. For example, the value is 80 for A100 and 70 for V100. For details, visit the **NVIDIA website**.

Step 6 Perform the compilation.

```
make -j40
make -j40 install
```

Step 7 View the RELION version.

```
/path/to/install/RELION/bin/relion_refine_mpi --help
```



 Copyright © Huawei Technologies Co., Ltd.

Step 8 Set the environment variables.

```
export PATH=/path/to/install/RELION/bin:$PATH
```

----End

4.91.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/CASE
```

Step 3 Decompress the test case file.

```
tar xvf relion_benchmark.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd relion_benchmark
```

Step 5 Perform the RELION test.

```
mkdir class3d
yum -y install time
time -p mpirun --allow-run-as-root -n 4 relion_refine_mpi --j 24 --gpu 0 --pool 100 --
dont_combine_weights_via_disc --i Particles/shiny_2sets.star --ref emd_2660.map:mrc --firstiter_cc --
ini_high 60 --ctf --ctf_corrected_ref --iter 25 --tau2_fudge 4 --particle_diameter 360 --K 6 --
flatten_solvent --zero_mask --oversampling 1 --healpix_order 2 --offset_range 5 --offset_step 2 --sym
C1 --norm --scale --random_seed 0 --maxsig 500 --fast_subsets --o class3d/test01
```

----End

4.92 VMD 1.9.4 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

4.92.1 Introduction

VMD is a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting. VMD supports computers running MacOS X, Unix, or Windows, is distributed free of charge, and includes source code.

VMD is designed for modeling, visualization, and analysis of biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. It may be used to view more general molecules, as VMD can read standard Protein Data Bank (PDB) files and display the contained structure. VMD provides a wide variety of methods for rendering and coloring a molecule: simple points and lines, CPK spheres and cylinders, licorice bonds, backbone tubes and ribbons, cartoon drawings, and others. VMD can be used to animate and analyze the trajectory of a molecular dynamics (MD) simulation. In particular, VMD can act as a graphical front end for an external MD program by displaying and animating a molecule undergoing simulation on a remote computer.

For more information about VMD, visit the **VMD official website**.

Programming language: C

Brief description: A molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting.

Open source license: UIUC Open Source License

Recommended Software Version

VMD 1.9.4a55

4.92.2 Environment Requirements

Hardware Requirements

Table 4-453 lists the hardware requirements.

Table 4-453 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 4-454 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-454 Software requirements

Item	Version	Download Address
VMD	1.9.4	**https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD**
NVIDIA CUDA	11.4	**https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run**

OS Requirements

Table 4-455 lists the OS requirements.

Table 4-455 OS requirements

Item	Version	Download URL
CentOS	8.2	**https://vault.centos.org/8.2.2004/isos/aarch64/**
Kernel	4.18.0-193	**https://vault.centos.org/8.2.2004/BaseOS/Source/SPackages/kernel-4.18.0-193.el8.src.rpm**

4.92.3 Paths for Software Porting

This chapter lists software installation paths involved in the VMD software porting.

Table 4-456 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment.	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/VMD	Installation path of VMD.	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.
3	/path/to/rpm	Path of TCL and TK RPM packages.	

 Copyright © Huawei Technologies Co., Ltd.

4.92.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 4-457 Configure process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Disabling the Nouveau driver	See **4.92.4.1 Disabling the Nouveau Driver**.
3	Installing NVIDIA CUDA	See **4.92.4.2 Installing NVIDIA CUDA**.

4.92.4.1 Disabling the Nouveau Driver

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the server.

```
reboot
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.92.4.2 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3** to **Step 5**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 4 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 5 Perform the verification.

```
nvcc --version
```

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

4.92.5 Obtaining the Source Code

Procedure

Step 1 Download the VMD installation package.

https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD

Step 2 Use an SFTP tool to upload the VMD installation package to the */path/to/VMD* directory on the server.

----End

4.92.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/VMD
```

Step 3 Decompress the installation package.

```
tar -xvf vmd-1.9.4a55.src.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Go to the directory generated after the decompression.

```
cd vmd-1.9.4a55
```

Step 5 Install FLTK.

```
yum install fltk fltk-devel
```

Step 6 Install ACTC.

1. Create an **ACTC** directory and switch to the directory.
   ```
   mkdir -p /path/to/VMD/vmd-1.9.4a55/lib/actc
   cd /path/to/VMD/vmd-1.9.4a55/lib/actc
   ```
2. Download the ACTC installation package.

   https://sourceforge.net/projects/actc/
3. Decompress the installation package.
   ```
   tar xvf actc-1.1.tar.gz
   ```
4. Compile ACTC.
   ```
   mv actc-1.1/* ./
   make
   ```
   If the command output contains **libactc.a**, the compilation is successful. For additional information, see README.

Step 7 Install Stride.

1. Go to the **stride** directory.
   ```
   cd /path/to/VMD/vmd-1.9.4a55/lib/stride
   ```
2. Download the Stride installation package.

   http://webclu.bio.wzw.tum.de/stride/stride.tar.gz
3. Decompress the installation package.
   ```
   tar xvf stride.tar.gz
   ```
4. Install Stride.
   ```
   make
   ```
5. Create a soft link.
   ```
   ln -s /path/to/VMD/vmd-1.9.4a55/lib/stride/stride /path/to/VMD/vmd-1.9.4a55/lib/stride/
   stride_LINUXARM64
   ```

Step 8 Install Surf.

1. Go to the Surf directory.
   ```
   cd /path/to/VMD/vmd-1.9.4a55/lib/surf
   ```
2. Decompress the installation package.
   ```
   tar xvf surf.tar.Z
   ```
3. Install Surf.
   ```
   yum install imake
   make depend
   make surf
   ```
4. Create a soft link.
   ```
   ln -s /path/to/VMD/vmd-1.9.4a55/lib/stride/surf /path/to/VMD/vmd-1.9.4a55/lib/stride/
   surf_LINUXARM64
   ```

Step 9 Install VRPN.

1. Create a **VRPN** directory and switch to the directory.
   ```
   mkdir -p /path/to/VMD/vmd-1.9.4a55/lib/vrpn
   cd /path/to/VMD/vmd-1.9.4a55/lib/vrpn
   ```
2. Download the VRPN installation package.

   https://github.com/vrpn/vrpn

 Copyright © Huawei Technologies Co., Ltd.

3. Decompress the installation package.
   ```
   unzip vrpn-master.zip
   ```
4. Install VRPN.
   ```
   mv vrpn-master/* ./
   mkdir build
   cd build
   yum install cmake libarchive
   cmake ../ -DCMAKE_INSTAL_PREFIX=/path/to/vmd/vmd-1.9.4a55/lib/vrpn/
   make && make install
   ```

Step 10 Install Tachyon.

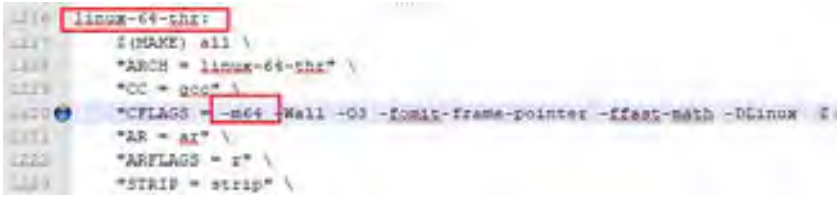
1. Go to the Tachyon directory.
   ```
   cd /path/to/VMD/vmd-1.9.4a55/lib/tachyon
   ```
2. Download the Tachyon installation package.

   https://ftp.sudhip.com/archlinux/other/community/tachyon/tachyon-0.99b6.tar.gz
3. Decompress the installation package.
   ```
   tar xvf tachyon-0.99b6.tar.gz
   ```
4. Go to the **unix** directory of the installation package.
   ```
   cd tachyon/unix
   ```
5. Modify the **Make-config** file.
   a. Open the **Make-config** file.
      ```
      vi Make-config
      ```
   b. Press **i** to enter the edit mode and comment out the JPEG and PNG lines.

      Before the modification:
      ```
      169 USEJPEG=
      170 JPEGINC=
      171 JPEGLIB=
      ...
      189 USEPNG=
      190 PNGINC=
      191 PNGLIB=
      ```
      After the modification:
      ```
      169 #USEJPEG=
      170 #JPEGINC=
      171 #JPEGLIB=
      ...
      189 #USEPNG=
      190 #PNGINC=
      191 #PNGLIB=
      ```

      NOTE

      After performing **Step 10.5.a**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
   c. Press **Esc**, enter **:wq!**, and press **Enter** to save the script and exit.
6. Delete the **-m64** compilation parameters.
   ```
   sed -i "s/-m64//" Make-arch
   ```

7. Modify the **spaceball.c** file.

 Copyright © Huawei Technologies Co., Ltd.

a. Open the **spaceball.c** file.

```
cd ../ demosrc
vi spaceball.c
```

b. Press **i** to enter the edit mode, change **port** in line 35 to **serialport**, and add parentheses to lines 109, 110, and 111.

Before the modification:

```
35   if (port != NULL) {
...
109   if ((buttonchanged & SBALL_BUTTON_PICK) && (buttons & SBALL_BUTTON_PICK)) ||
110       (buttonchanged & SBALL_BUTTON_1) && (buttons & SBALL_BUTTON_1)) ||
111       (buttonchanged & SBALL_BUTTON_LEFT) && (buttons & SBALL_BUTTON_LEFT)) ) {
```

After the modification:

```
35   if (serialport != NULL) {
...
109   if (((buttonchanged & SBALL_BUTTON_PICK) && (buttons & SBALL_BUTTON_PICK)) ||
110       ((buttonchanged & SBALL_BUTTON_1) && (buttons & SBALL_BUTTON_1)) ||
111       ((buttonchanged & SBALL_BUTTON_LEFT) && (buttons & SBALL_BUTTON_LEFT)) ) {
```

NOTE

After performing **Step 10.7.a**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

c. Press **Esc**, enter **:wq!**, and press **Enter** to save the script and exit.

8. Install Tachyon.

```
cd ../unix
make linux-64-thr
```

9. Create a soft link.

```
ln –s /path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon/compile/linux-64-thr/tachyon /
path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon_LINUXARM64
```

Step 11 Install TCL 8.5 and TK 8.5.

1. Download the TCL and TK RPM packages.

   TCL: **http://mirror.centos.org/altarch/7/os/aarch64/Packages/tcl-8.5.13-8.el7.aarch64.rpm**

   http://mirror.centos.org/altarch/7/os/aarch64/Packages/tcl-devel-8.5.13-8.el7.aarch64.rpm

   TK: **http://mirror.centos.org/altarch/7/os/aarch64/Packages/tk-8.5.13-6.el7.aarch64.rpm**

   http://mirror.centos.org/altarch/7/os/aarch64/Packages/tk-devel-8.5.13-6.el7.aarch64.rpm

2. Use an SFTP tool to upload the TCL and TK installation packages to the */path/to/rpm* directory on a server.

3. Install TCL and TK.

```
cd /path/to/rpm
rpm –ivh tcl-8.5.13-8.el7.aarch64.rpm
rpm –ivh tcl-devel-8.5.13-8.el7.aarch64.rpm
rpm –ivh tk-8.5.13-6.el7.aarch64.rpm
rpm –ivh tk-devel-8.5.13-6.el7.aarch64.rpm
```

Step 12 Compile plugins.

1. Go to the **plugins** directory.

```
cd /path/to/VMD/plugins
```

2. Set the environment variables of the TCL header file and library file.

```
export TCL_INCLUDE_DIR=/usr/share/tcl8.5
export TCL_LIBRARY_DIR=/usr/lib64/tcl8.5
```

 Copyright © Huawei Technologies Co., Ltd.

3. Perform the compilation.
```
make LINUXARM64
mkdir /path/to/vmd/vmd-1.9.4a55/plugins
export PLUGINDIR=/path/to/vmd/vmd-1.9.4a55/plugins
make distrib
```

Step 13 Compile VMD.

1. Go to the **vmd-1.9.4a55** directory.
```
cd /path/to/vmd/vmd-1.9.4a55/
```
2. Modify the configuration file.
   a. Open the configuration file.
```
vi configure
```
   b. Press **i** to enter the edit mode.
      - Set the installation path.
```
$install_bin_dir="/path/to/vmd/ vmd-1.9.4a55/bin";
$install_library_dir="/path/to/vmd/ vmd-1.9.4a55/lib/vmd";
```
      - Set the Tachyon environment variables.
```
$libtachyon_dir="/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon";
$libtachyon_include="-I/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon/src";
$libtachyon_library="-L"/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon/compile/linux-64-thr;
```
      - Modify the **$arch_nvccflags** parameter of Linux ARM64 and comment out **"-gencode arch=compute_30,code=compute_30 " .s**.

        Before the modification:
```
16 $install_bin_dir="/usr/local/bin";
17
18 # Directory where VMD files and executables are installed
19 $install_library_dir="/usr/local/lib/$install_name";
...
1354 $libtachyon_dir         = "$vmd_library_dir/tachyon";
1355 $libtachyon_include     = "-I$libtachyon_dir/include";
1356 $libtachyon_library     = "-L$libtachyon_dir/lib_$config_arch";
...
2627                    "-gencode arch=compute_30,code=compute_30 " .
```
        After the modification:
```
16 $install_bin_dir="/path/to/vmd/vmd-1.9.4a55/bin";
17
18 # Directory where VMD files and executables are installed
19 $install_library_dir="/path/to/vmd/vmd-1.9.4a55/lib/vmd";
...
1354 $libtachyon_dir         = "/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon";
1355 $libtachyon_include     = "-I/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon/src";
1356 $libtachyon_library     = "-L/path/to/vmd/vmd-1.9.4a55/lib/tachyon/tachyon/compile/linux-64-thr";
...
2627   #"-gencode arch=compute_30,code=compute_30 " .
```

        NOTE

        After performing Step 13.2.a, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
   c. Press **Esc**, enter **:wq!**, and press **Enter** to save the script and exit.
3. Perform the compilation.
```
./configure LINUXARM64 EGLPBUFFER CUDA IMD LIBTACHYON ZLIB COLVARS TCL PTHREADS SILENT
cd src && make veryclean
make
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

If "No resource compiler required on this platform" is displayed, the compilation is successful.

```
make install
```

If "Make sure /Projects/vmd/ vmd-1.9.4a55/bin is in your path.VMD installation complete. Enjoy!" is displayed, the installation is successful.

----End

4.92.7 Running and Verification

Procedure

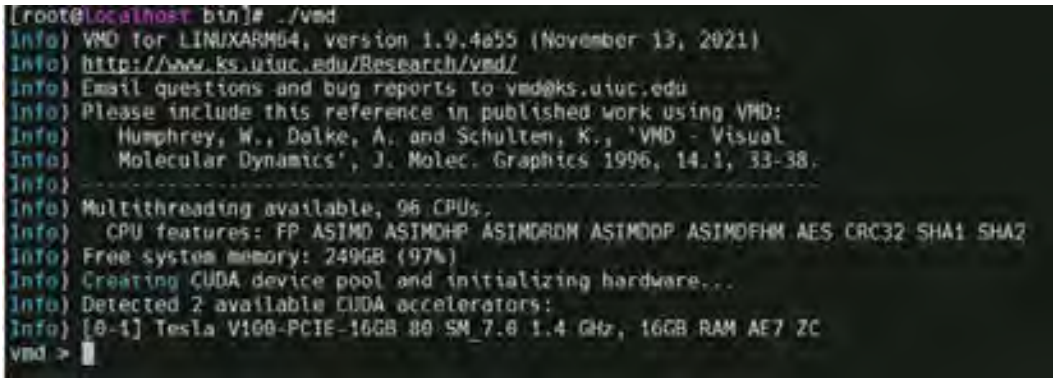
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **bin** directory.

```
cd /path/to/vmd/vmd-1.9.4a55/bin
```

Step 3 Start VMD.

```
./vmd
```



----End

4.93 DBG2OLC Porting Guide (openEuler 20.03)

4.93.1 Introduction

DBG2OLC is a third-generation gene sequencing and assembly software.

Sparc is a third-generation gene sequencing and error correction software.

SparseAssembler is the second-generation gene sequencing and assembly software.

For more information, visit the **DBG2OLC official website** and **DBG2OL related website**.

Programming language: C++

Brief description: third-generation gene sequencing software

Open source license: GPL 3.0

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

N/A

4.93.2 Environmental Requirements

Hardware Requirements

Table 4-458 lists the hardware requirements.

Table 4-458 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-459 lists the software requirements.

Table 4-459 Software requirements

Item	Version	Download URL
DBG2OLC	-	https://codeload.github.com/yechengxi/DBG2OLC/zip/refs/heads/master
BiSheng Compiler	2.1.0	https://www.hikunpeng.com/en/developer/devkit/compiler/bisheng

OS Requirements

Table 4-460 lists the OS requirements.

Table 4-460 OS requirements

Item	Version	Download URL
openEuler	20.03	https://www.openeuler.org/en/download/

4.93.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DBG2OLC software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-461 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

4.93.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-462 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.93.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the DBG2OLC installation package and the Git repository code of related tools.

```
git clone https://github.com/yechengxi/DBG2OLC.git
git clone https://github.com/yechengxi/SparseAssembler.git
git clone https://github.com/yechengxi/Sparc.git
git clone https://github.com/yechengxi/AssemblyUtility.git
git clone https://github.com/mchaisso/blasr.git
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.93.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the location of the DBG2OLC source code and compile the source code.

```
cd DBG2OLC
clang++ -O3 -o DBG2OLC *.cpp
```

Step 3 Go to the location of the Sparc source code and compile the source code.

```
cd Sparc
clang++ -O3 -o Sparc *.cpp
```

Step 4 Go to the location of the AssemblyUtility source code and compile the source code.

```
cd AssemblyUtility
clang++ -o AssemblyStatistics  -O3 AssemblyStatistics.cpp
clang++ -o SelectLongestReads  -O3 SelectLongestReads.cpp
```

Step 5 Go to the location of the SparseAssembler source code and compile the source code.

```
cd SparseAssembler
clang++ -O3 -o SparseAssembler *.cpp
```

Step 6 Go to the location of the blasr-master source code.

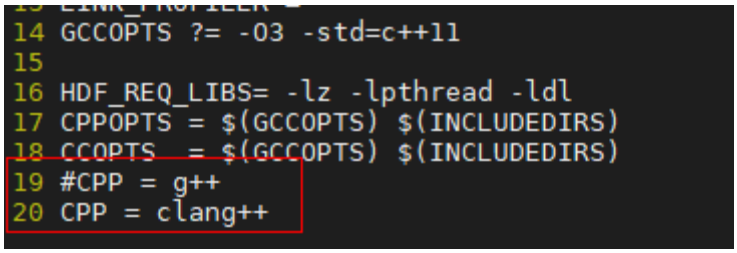
```
cd blasr-master
```

Step 7 Install HDF5.

```
yum install -y hdf5 hdf5-devel
```

Step 8 Modify the **common.mk** file.

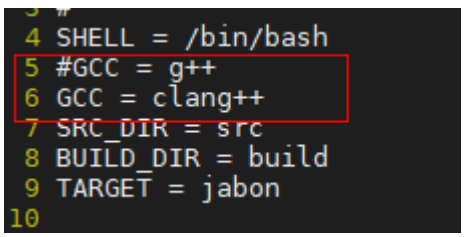
1. Open **common.mk**.
   ```
   vi common.mk
   ```
2. Press **i** to enter the insert mode, comment out line 19, and add line 20.



3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Modify the **jabon/Makefile** file.

1. Open **jabon/Makefile**.
   ```
   vi jabon/Makefile
   ```
2. Press **i** to enter the insert mode, comment out line 5, and add line 6.



 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Perform the compilation.

```
make -j
```

Step 11 In the **DBG2OLC** directory, copy the executable file to the specified path.

```
mkdir compiled
cp ../AssemblyUtility/AssemblyStatistics compiled
cp ../AssemblyUtility/SelectLongestReads compiled
cp ../Sparc/Sparc compiled
cp ../SparseAssembler/SparseAssembler compiled
cp ../blasr-master/alignment/bin/blasrmc compiled
```

Step 12 Set the environment variables.

```
export PATH= /home/hpc/DBG2OLC/compiled:/home/hpc/DBG2OLC/utility:$PATH
```

----End

4.93.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the case files.

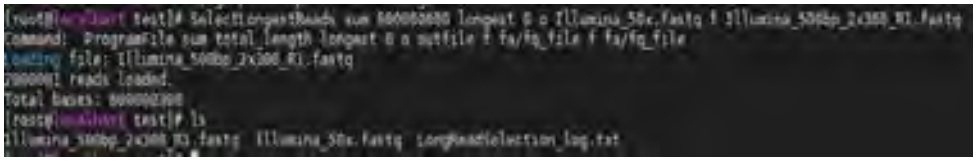
```
wget http://labshare.cshl.edu/shares/schatzlab/www-data/ectools/w303/
Illumina_500bp_2x300_R1.fastq.gz
wget http://labshare.cshl.edu/shares/schatzlab/www-data/ectools/w303/Pacbio.fasta.gz
```

Step 3 Decompress the case files.

```
gzip Pacbio.fasta.gz –d
gzip Illumina_500bp_2x300_R1.fastq.gz -d
```

Step 4 Process the data.

```
SelectLongestReads sum 600000000 longest 0 o Illumina_50x.fastq f Illumina_500bp_2x300_R1.fastq
```



```
SelectLongestReads sum 260000000 longest 0 o Pacbio_20x.fasta f Pacbio.fasta
```

Step 5 Create an **Illumina_data** directory and copy the generated FASTQ file to the **Illumina_data** directory.

```
mkdir Illumina_data && cp Illumina_50x.fastq Illumina_data/
```

Step 6 Create an **Pacbio_data** directory and copy the generated FASTQ file to the **Pacbio_data** directory.

```
mkdir Pacbio_data && cp Pacbio_20x.fasta Pacbio_data/
```

Step 7 Create a **step1** directory and switch to the directory.

```
mkdir step1 && cd step1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 8 Assemble a Contigs sequence using the Illumina fragment library data.

```
SparseAssembler LD 0 k 51 g 15 NodeCovTh 1 EdgeCovTh 0 GS 12000000 f ../Illumina_data/Illumina_50x.fastq
```

```
SparseAssembler LD 1 NodeCovTh 2 EdgeCovTh 1 k 51 g 15 GS 12000000 f ../Illumina_data/Illumina_50x.fastq
```

The following files are generated:

```
[root@localhost step1]# ls -alt
total 99756
-rw-r--r-- 1 root root   212021 Mar 23 19:34 ContigGraph.txt
-rw-r--r-- 1 root root    43456 Mar 23 19:34 Contigs_HP.txt
-rw-r--r-- 1 root root    11549 Mar 23 19:34 Cov_HistContigs.txt
-rw-r--r-- 1 root root    27678 Mar 23 19:34 Contigs_Cov.txt
-rw-r--r-- 1 root root   496401 Mar 23 19:34 Contigs_info.txt
-rw-r--r-- 1 root root    29517 Mar 23 19:34 Contigs_Len.txt
-rw-r--r-- 1 root root       20 Mar 23 19:34 ContigsLog.txt
-rw-r--r-- 1 root root 12178370 Mar 23 19:34 Contigs.txt
-rw-r--r-- 1 root root    17943 Mar 23 19:34 nLB_Contigs.txt
-rw-r--r-- 1 root root    17946 Mar 23 19:34 nRB_Contigs.txt
-rw-r--r-- 1 root root   827574 Mar 23 19:34 MergeHT_content
-rw-r--r-- 1 root root    40028 Mar 23 19:34 MergeHT_idx.txt
-rw-r--r-- 1 root root        0 Mar 23 19:34 BasesCount.txt
drwxr-xr-x 2 root root     4096 Mar 23 19:32 .
-rw-r--r-- 1 root root 84486280 Mar 23 19:32 Asm_HT_content
-rw-r--r-- 1 root root    17488 Mar 23 19:32 CovHist.txt
-rw-r--r-- 1 root root      120 Mar 23 19:32 EdgeCovHist.txt
-rw-r--r-- 1 root root  3692331 Mar 23 19:32 Asm_HT_idx.txt
-rw-r--r-- 1 root root      161 Mar 23 19:32 Assembly_Log.txt
```

 Copyright © Huawei Technologies Co., Ltd.

Step 9 Find the overlap between the Contigs sequence and Pacbio reads and perform layout.

```
DBG2OLC k 17 AdaptiveTh 0.0001 KmerCovTh 2 MinOverlap 20 RemoveChimera 1 Contigs Contigs.txt f ../Pacbio_data/Pacbio_20x.fasta
```

Information similar to the following is displayed:

```
24 branching positions.
403 linear nodes.
Iteration: 3
24 branching positions.
403 linear nodes.
51 edges deleted.
2175 chimeric reads deleted.
5 bad nodes removed in aggressive cleaning.
64 tips in the graph.
Loading contigs.
Collecting information for consensus.
26622 reads.
Calculating reads overlaps.
Avg alignment size: 16
Avg sparse alignment size: 4
123215 alignments calculated.
3 secs.
Loading non-contained sequences.
2624 loaded.
frag sum: 6482443
offset sum: 11642452
Extension warning.
Extension warning.
Assembly finished.
```

The following files are generated:

```
[root@localhost step1]# ls -alt
total 430548
-rw-r--r-- 1 root root  12104164 Mar 23 19:44 backbone_raw.fasta
-rw-r--r-- 1 root root     33115 Mar 23 19:44 full_layout_coords.txt
-rw-r--r-- 1 root root   6518127 Mar 23 19:44 full_layout.fasta
-rw-r--r-- 1 root root      2636 Mar 23 19:44 extension_warning.txt
drwxr-xr-x 2 root root      4096 Mar 23 19:44 .
-rw-r--r-- 1 root root        39 Mar 23 19:44 DBG2OLC_CompressedOverlapGraph.dot
-rw-r--r-- 1 root root        13 Mar 23 19:44 DBG2OLC_CompressedOverlapGraph.txt
-rw-r--r-- 1 root root   2496627 Mar 23 19:44 DBG2OLC_Consensus_info.txt
-rw-r--r-- 1 root root     77747 Mar 23 19:44 DBG2OLC_ContigPaths.txt
-rw-r--r-- 1 root root     60733 Mar 23 19:44 DBG2OLC_Nodes_info.txt
-rw-r--r-- 1 root root       225 Mar 23 19:44 DBG2OLC_NodeSizeEst.txt
-rw-r--r-- 1 root root     24299 Mar 23 19:44 DBG2OLC_Nodes.txt
-rw-r--r-- 1 root root         0 Mar 23 19:44 extension_log.txt
-rw-r--r-- 1 root root    151461 Mar 23 19:44 layout_compressed_reads.txt
-rw-r--r-- 1 root root   6518633 Mar 23 19:44 layout_reads.txt
-rw-r--r-- 1 root root     11829 Mar 23 19:44 FinalBiDirectedBestOverlapGraph.dot
-rw-r--r-- 1 root root      7247 Mar 23 19:44 CleanedBestOverlapGraph.txt
-rw-r--r-- 1 root root     15484 Mar 23 19:44 CleanedBestOverlapGraph.dot
-rw-r--r-- 1 root root     12330 Mar 23 19:44 CleanedBestOverlapGraphWithTies.dot
-rw-r--r-- 1 root root      9746 Mar 23 19:44 CleanedBestOverlapGraphWithTies.txt
-rw-r--r-- 1 root root     14983 Mar 23 19:44 StrongCleanedBestOverlapGraph.dot
-rw-r--r-- 1 root root     13140 Mar 23 19:44 CleanedGraph.dot.dot
-rw-r--r-- 1 root root     14368 Mar 23 19:44 CleanedGraph.dot.txt
-rw-r--r-- 1 root root     13234 Mar 23 19:44 BiDirectedBestOverlapGraph.dot
-rw-r--r-- 1 root root     12117 Mar 23 19:44 ChimeraIdx.txt
-rw-r--r-- 1 root root    144172 Mar 23 19:44 ContainedIdx.txt
-rw-r--r-- 1 root root      7858 Mar 23 19:44 RawBestOverlapGraph.txt
-rw-r--r-- 1 root root     15742 Mar 23 19:44 RawBestOverlapGraph.dot
-rw-r--r-- 1 root root     13229 Mar 23 19:44 RawBestOverlapGraphWithTies.dot
-rw-r--r-- 1 root root     10462 Mar 23 19:44 RawBestOverlapGraphWithTies.txt
-rw-r--r-- 1 root root     14284 Mar 23 19:44 StrongRawBestOverlapGraph.dot
-rw-r--r-- 1 root root     20234 Mar 23 19:44 RawOverlapGraph.dot
-rw-r--r-- 1 root root     21922 Mar 23 19:44 RawOverlapGraph.txt
-rw-r--r-- 1 root root    452166 Mar 23 19:44 CompressedNonContainedReads.txt
-rw-r--r-- 1 root root    170273 Mar 23 19:44 Chimeras.txt
-rw-r--r-- 1 root root   3411555 Mar 23 19:44 selected_reads.txt
-rw-r--r-- 1 root root   3921191 Mar 23 19:44 CleanedLongReads.txt
-rw-r--r-- 1 root root       654 Mar 23 19:43 LongReadContigIndex_Histogram.txt
-rw-r--r-- 1 root root        55 Mar 23 19:43 LongReadContigIndex_log.txt
-rw-r--r-- 1 root root   7190812 Mar 23 19:43 ReadsInfoFrom_Pacbio_20x.fasta
-rw-r--r-- 1 root root 271320984 Mar 23 19:42 ContigKmerIndex_HT_content
-rw-r--r-- 1 root root  23820960 Mar 23 19:42 ContigKmerIndex_HT_idx.txt
drwxr-xr-x 5 root root      4096 Mar 23 19:39 ..
```

Step 10 Use the **python** and **shell** scripts in the **/opt/biosoft/DBG2OLC/utility/** directory to invoke Sparc of the **blasr** and **consensus** modules for calculation.

1. Modify the **split_and_run_sparc.sh** script.
   ```
   vi split_and_run_sparc.sh
   ```
2. Press **i** to enter the insert mode, comment out line 27, and add line 28.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

After performing **Step 10.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Run the following command in the **step1** directory:

```
cp ../Pacbio_20x.fasta .
cat Contigs.txt Pacbio_20x.fasta > ctg_pb.fasta
mkdir consensus_dir
split_and_run_sparc.sh backbone_raw.fasta DBG2OLC_Consensus_info.txt ctg_pb.fasta ./
consensus_dir 2 >cns_log.txt
```

The following files are generated:

----End

4.94 MAKER 2.31.11 Porting Guide (openEuler 20.03)

4.94.1 Introduction

MAKER is a portable and easily configurable genome annotation pipeline. Its purpose is to allow smaller eukaryotic and prokaryotic genome projects to independently annotate their genomes and to create genome databases. MAKER identifies repeats, aligns ESTs and proteins to a genome, produces ab-initio gene predictions and automatically synthesizes these data into gene annotations having evidence-based quality values.

For more information, visit the **MAKER official website**.

Programming language: Perl

Brief description: a portable and easily configurable genome annotation pipeline.

Open source license: Artistic License 2.0

Recommended Software Version

MAKER 2.31.11

 Copyright © Huawei Technologies Co., Ltd.

4.94.2 Environmental Requirements

Hardware Requirements

Table 4-463 lists the hardware requirements.

Table 4-463 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-464 lists the software requirements.

Table 4-464 Software requirements

Item	Version	Download URL
Perl	5.28.0	Provided by openEuler.
GCC	7.3.0	Provided by openEuler.
MPICH	mpich2-1.5	**https://www.mpich.org/static/tarballs/1.5/mpich2-1.5.tar.gz**
BLAST	2.9.0	**https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.9.0/ncbi-blast-2.9.0+-src.tar.gz**
RMBlast	2.2.28	**https://ftp.ncbi.nlm.nih.gov/blast/executables/rmblast/2.2.28/ncbi-rmblastn-2.2.28-src.tar.gz**
SNAP	2013-11-29	**http://korflab.ucdavis.edu/Software/snap-2013-11-29.tar.gz**
TRF	4.09.1	**https://codeload.github.com/Benson-Genomics-Lab/TRF/tar.gz/refs/tags/v4.09.1**
RepeatMasker	4.0.7	**http://www.repeatmasker.org/RepeatMasker-open-4-0-7.tar.gz**
Exonerate	2.2.0	**http://ftp.ebi.ac.uk/pub/software/vertebrategenomics/exonerate/exonerate-2.2.0.tar.gz**
MAKER	2.31.11	**https://www.yandell-lab.org/software/maker.html** (register and download)

OS Requirements

Table 4-465 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-465 OS requirements

Item	Version	Download URL
openEuler	20.03	https://www.openeuler.org/en/download/

4.94.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MAKER software porting.

Table 4-466 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/mpich	Installation path of MPICH	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/blast	Installation path of BLAST	
3	/path/to/rmblast	Installation path of RMBlast	
4	/path/to/snap	Installation path of SNAP	
5	/path/to/trf	Installation path of TRF	
6	/path/to/repeatmasker	Installation path of RepeatMasker	
7	/path/to/exonerate	Installation path of Exonerate	
8	/path/to/maker	Installation path of MAKER	

4.94.4 Configuring the Compilation Environment

4.94.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-467 Configuration process

No.	Configuration Item	Remarks
1	Installing GLib	See **4.94.4.2 Installing GLib**.
2	Installing Exonerate	See **4.94.4.3 Installing Exonerate**.
3	Installing MPICH	See **4.94.4.4 Installing MPICH**.
4	Installing BLAST	See **4.94.4.5 Installing BLAST**.
5	Installing RMBlast	See **4.94.4.6 Installing RMBlast**.
5	Installing SNAP	See **4.94.4.7 Installing SNAP**.
7	Installing TRF	See **4.94.4.8 Installing TRF**.
8	Installing RepeatMasker	See **4.94.4.9 Installing RepeatMasker**.

4.94.4.2 Installing GLib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GLib using yum.

```
yum install glib glib-devel
```

Step 3 Copy the header files to the system directory.

```
cp /usr/include/glib-1.2/glib.h /usr/include/
cp /usr/lib64/glib/include/glibconfig.h /usr/include/
```

----End

4.94.4.3 Installing Exonerate

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf exonerate-2.2.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd exonerate-2.2.0
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/exonerate --build=aarch64-unknown-linux-gnu LIBS=-lglib
make
make install
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Load the environment variable.

```
export PATH=/path/to/exonerate/bin:$PATH
```

----End

4.94.4.4 Installing MPICH

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf mpich2-1.5.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd mpich2-1.5
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/mpich --enable-shared=yes --build=aarch64-unknown-linux-gnu
make -j
make -j install
```

Step 5 Load the environment variables.

```
export PATH=/path/to/mpich/bin:$PATH
export LD_LIBRARY_PATH=/path/to/mpich/lib:$LD_LIBRARY_PATH
```

----End

4.94.4.5 Installing BLAST

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf ncbi-blast-2.9.0+-src.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd ncbi-blast-2.9.0+-src
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/blast
make -j
make -j install
```

Step 5 Load the environment variable.

```
export PATH=/path/to/blast/bin:$PATH
```

----End

4.94.4.6 Installing RMBlast

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf ncbi-rmblastn-2.2.28-src.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd ncbi-rmblastn-2.2.28-src
```

Step 4 Modify the **configure** file.

1. Open **configure**.
   ```
   vi src/build-system/configure
   ```
2. Press **i** to enter the insert mode and modify line 4749.
   ```
   4746 if test "$GCC" = "yes" ; then
   4747    compiler_ver="`$real_CXX -dumpversion 2>&1`"
   4748    case "$compiler_ver" in
   4749      2.95* | 2.96* | 3.* | 4.* | 7.*)
   4750        compiler="GCC"
   4751        ncbi_compiler="GCC"
   4752        ncbi_compiler_ver="$compiler_ver"
   ```

   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the compilation and installation.

```
./configure --prefix=/path/to/rmblas --build=aarch64-unknown-linux-gnu --without-debug --with-mt CFLAGS=-Wno-narrowing CXXFLAGS=-Wno-narrowing
make -j
make -j install
```

Step 6 Load the environment variable.

```
export PATH=/path/to/rmblast/bin:$PATH
```

----End

4.94.4.7 Installing SNAP

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf snap-2013-11-29.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd snap
```

Step 4 Modify the **Makefile** file in the source code root directory.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and delete the **-Werror** option from line 97 to line 101.
   ```
   make $(APP)  CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP2) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP3) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP4) CC="gcc" CFLAGS="-O2 –Wall"
   make $(APP5) CC="gcc" CFLAGS="-O2 –Wall"
   ```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

After performing Step 4.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the **Makefile** file in the **Zoe** directory.

1. Open **Makefile**.
```
vi Makefile
```
2. Press **i** to enter the insert mode and delete the **-Werror** option from line 94 to line 102.
```
gcc:
make $(APP)  CC="gcc" CFLAGS="-O2 -Wall"
all:
make $(APP)  CC="gcc" CFLAGS="-O2 –Wall"
make $(APP2) CC="gcc" CFLAGS="-O2 –Wall"
make $(APP3) CC="gcc" CFLAGS="-O2 –Wall"
make $(APP4) CC="gcc" CFLAGS="-O2 –Wall"
make $(APP5) CC="gcc" CFLAGS="-O2 –Wall"
```

NOTE

After performing Step 5.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.
```
make
```

Step 7 Load the environment variable.
```
export PATH=/path/to/snap:$PATH
```

----End

4.94.4.8 Installing TRF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.
```
tar zxvf TRF-4.09.1.tar.gz
```

Step 3 Go to the directory generated after the decompression.
```
cd TRF-4.09.1
```

Step 4 Perform the compilation and installation.
```
./configure --prefix=/path/to/trf
make
make install
```

Step 5 Load the environment variable.
```
export PATH=/path/to/trf/bin:$PATH
```

----End

4.94.4.9 Installing RepeatMasker

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf RepeatMasker-open-4-0-7.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd RepeatMasker
```

Step 4 Perform the compilation and installation.

```
perl ./configure
```

NOTE

Specify the **perl/rmblast/trf** path as prompted. If environment variables have been loaded for the previous dependencies, the correct path can be identified by default.

Step 5 Load the environment variable for the executable file generated in the source code root directory.

```
export PATH=/path/to/repeatmasker:$PATH
```

Step 6 (Optional) Register and download Repbase (**http://www.girinst.org**), decompress the Repbase package, and save it to the **RepeatMasker/Libraries** directory.

NOTE

Perform this operation when you need to use the Repbase database to save species data.

----End

4.94.5 Obtaining the Source Code

Procedure

Step 1 Download the MAKER source package **maker-2.31.11.tgz**.

https://www.yandell-lab.org/software/maker.html (register and download)

Step 2 Use an SFTP tool to upload the source package to the server.

----End

4.94.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar zxvf maker-2.31.11.tgz
```

Step 3 Go to the directory generated after the decompression.

```
cd maker
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Go to the source code directory and configure MPI.

```
cd src
perl Build.PL
```

NOTE

During the configuration process, when the system asks you whether to support MPI, select **Yes**. If the MPICH environment variable has been loaded, the system automatically identifies the environment variable.

Step 5 Install the Perl package through CPAN.

```
yum install perl-CPAN
./Build installdeps
```

NOTE

During the installation process, when the system asks you whether to install DBD::Pg, select **No**. If the PostgreSQL database has been configured, select **Yes**.

Step 6 Install MAKER.

```
./Build install
```

Step 7 The executable file is generated in the **bin** directory of the source code root directory. Run the following command to add the file to environment variables:

```
export PATH=/path/to/maker/bin:$PATH
```

----End

4.94.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the MAKER source code root directory.

```
cd /path/to/maker
```

Step 3 Create a **test** directory and switch to the directory.

```
mkdir test
cd test
```

Step 4 Copy the test data.

```
cp ../data/dpp_* ./
```

Step 5 Create a configuration file.

```
maker -CTL
```

```
maker_bopts.ctl
maker_exe.ctl
maker_opts.ctl
```

Step 6 Modify the **maker_opts.ctl** file to specify the test data to be copied.

1. Open **maker_opts.ctl**.
   ```
   vi maker_opts.ctl
   ```
2. Press **i** to enter the insert mode and modify lines 2, 16, and 22.

   Before the modification:

```
genome=  #genome sequence (fasta file or fasta embeded in GFF3 file)
organism_type=eukaryotic #eukaryotic or prokaryotic. Default is eukaryotic

#-----Re-annotation Using MAKER Derived GFF3
maker_gff= #MAKER derived GFF3 file
est_pass=0 #use ESTs in maker_gff: 1 = yes, 0 = no
altest_pass=0 #use alternate organism ESTs in maker_gff: 1 = yes, 0 = no
protein_pass=0 #use protein alignments in maker_gff: 1 = yes, 0 = no
rm_pass=0 #use repeats in maker_gff: 1 = yes, 0 = no
model_pass=0 #use gene models in maker_gff: 1 = yes, 0 = no
pred_pass=0 #use ab-initio predictions in maker_gff: 1 = yes, 0 = no
other_pass=0 #passthrough anyything else in maker_gff: 1 = yes, 0 = no

#-----EST Evidence (for best results provide a file for at least one)
est=  #set of ESTs or assembled mRNA-seq in fasta format
altest= #EST/cDNA sequence file in fasta format from an alternate organism
est_gff= #aligned ESTs or mRNA-seq from an external GFF3 file
altest_gff= #aligned ESTs from a closly relate species in GFF3 format

#-----Protein Homology Evidence (for best results provide a file for at least one)
protein=  #protein sequence file in fasta format (i.e. from mutiple oranisms)
protein_gff=  #aligned protein homology evidence from an external GFF3 file
```

After the modification:

```
#-----Genome (these are always required)
genome=dpp_contig.fasta #genome sequence (fasta file or fasta embeded in GFF3 file)
organism_type=eukaryotic #eukaryotic or prokaryotic. Default is eukaryotic

#-----Re-annotation Using MAKER Derived GFF3
maker_gff= #MAKER derived GFF3 file
est_pass=0 #use ESTs in maker_gff: 1 = yes, 0 = no
altest_pass=0 #use alternate organism ESTs in maker_gff: 1 = yes, 0 = no
protein_pass=0 #use protein alignments in maker_gff: 1 = yes, 0 = no
rm_pass=0 #use repeats in maker_gff: 1 = yes, 0 = no
model_pass=0 #use gene models in maker_gff: 1 = yes, 0 = no
pred_pass=0 #use ab-initio predictions in maker_gff: 1 = yes, 0 = no
other_pass=0 #passthrough anyything else in maker_gff: 1 = yes, 0 = no

#-----EST Evidence (for best results provide a file for at least one)
est=dpp_est.fasta #set of ESTs or assembled mRNA-seq in fasta format
altest= #EST/cDNA sequence file in fasta format from an alternate organism
est_gff= #aligned ESTs or mRNA-seq from an external GFF3 file
altest_gff= #aligned ESTs from a closly relate species in GFF3 format

#-----Protein Homology Evidence (for best results provide a file for at least one)
protein=dpp_protein.fasta #protein sequence file in fasta format (i.e. from mutiple oranisms)
protein_gff=  #aligned protein homology evidence from an external GFF3 file
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the test case.

```
mpirun -n 4 maker -l archaea_odb10 maker_opts.ctl maker_bopts.ctl maker_exe.ctl
```

The following figure shows the command output.

NOTE

If an error is reported during the running, see **4.94.8.1 An Error Similar to "cannot locate Text/Soundex.pm in *****" Is Reported When Running a Test Case**.

----End

4.94.8 Troubleshooting

4.94.8.1 An Error Similar to "cannot locate Text/Soundex.pm in *****" Is Reported When Running a Test Case

Symptom

An error similar to "cannot locate Text/Soundex.pm in *****" is reported when running a test case.

Key Process and Cause Analysis

Text/Soundex.pm is not installed.

Conclusion and Solution

Install **Text/Soundex.pm**.

```
cpan -i Text/Soundex.pm
```

4.95 Platanus 1.2.4 Porting Guide (openEuler 20.03)

4.95.1 Introduction

PLATform for Assembling NUcleotide Sequences (Platanus) is a novel de novo sequence assembler.

For more information, visit the **Platanus official website**.

Programming language: C++

Brief description: gene sequence assembler.

Open source license: MIT

Recommended Software Version

Platanus 1.2.4

4.95.2 Environmental Requirements

Hardware Requirements

Table 4-468 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-468 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-469 lists the software requirements.

Table 4-469 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
Platanus	1.2.4	http://platanus.bio.titech.ac.jp/?download=150
Test cases	-	http://platanus.bio.titech.ac.jp/?download=325

OS Requirements

Table 4-470 lists the OS requirements.

Table 4-470 OS requirements

Item	Version	Download URL
openEuler	20.03	https://www.openeuler.org/en/download/

4.95.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Platanus software porting.

Table 4-471 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/PLATANUS	Installation path of Platanus	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/PLATANUS/CASE	Path for storing test cases	

4.95.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-472 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	For details, see "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.95.5 Obtaining the Source Code

Procedure

Step 1 Download the Platanus installation package **Platanus_v1.2.4.tar.gz**.

http://platanus.bio.titech.ac.jp/?download=150

Step 2 Use an SFTP tool to upload the Platanus installation package to the */path/to/PLATANUS* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.95.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/PLATANUS && tar -zxf Platanus_v1.2.4.tar.gz && cd Platanus_v1.2.4
```

Step 3 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and edit lines 1 and 2.

   Before the modification:

   ```
   1 CXX = g++
   2 CXXFLAGS = -std=c++0x -O3 -funroll-loops -Wall -fopenmp -finline-limit-50000 -lm -Dnullptr=0
   ```

   After the modification:

   ```
   1 CXX = clang++
   2 CXXFLAGS = -std=c++0x -O3 -funroll-loops -Wall -fopenmp -lm -Dnullptr=0
   ```

   NOTE

   After performing **Step 3.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Modify the **counter.h** file.

1. Open **counter.h**.
   ```
   vi counter.h
   ```
2. Press **i** to enter the insert mode and modify lines 426, 455, and 684.

   Before the modification:

   ```
   426 countKmerOrWriteTemporary(loopFlag, key, tmpOccurrenceTable, unmappedFP, lock, kmer);
   455 inline void Counter<KMER>::countKmerOrWriteTemporary(bool &loopFlag, const typename KMER::keyType &key, DoubleHash<typename KMER::keyType, unsigned short> tmpOccurrenceTable[], FILE *unmappedFP, omp_lock_t lock[], const KMER &kmer, const unsigned iterateTimes=32)
   684 inline void Counter<KMER>::countKmerOrWriteTemporary(bool &loopFlag, const typename KMER::keyType &key, FILE *unmappedFP, omp_lock_t lock[], const KMER &kmer, const unsigned iterateTimes=32)
   ```

   After the modification:

   ```
   426 countKmerOrWriteTemporary(loopFlag, key, tmpOccurrenceTable, unmappedFP, lock, kmer, 32);
   455 inline void Counter<KMER>::countKmerOrWriteTemporary(bool &loopFlag, const typename KMER::keyType &key, DoubleHash<typename KMER::keyType, unsigned short> tmpOccurrenceTable[], FILE *unmappedFP, omp_lock_t lock[], const KMER &kmer, const unsigned iterateTimes)
   684 inline void Counter<KMER>::countKmerOrWriteTemporary(bool &loopFlag, const typename KMER::keyType &key, FILE *unmappedFP, omp_lock_t lock[], const KMER &kmer, const unsigned iterateTimes)
   ```

   NOTE

   After performing **Step 4.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the **graph.h** file.

 Copyright © Huawei Technologies Co., Ltd.

1. Open **graph.h**.
   vi graph.h
2. Press **i** to enter the insert mode and modify line 1764.
   Before the modification:
   ```
   1764 void BruijnGraph<KMER>::cutBranchIterative(const unsigned long long numThread=1)
   ```
   After the modification:
   ```
   1764 void BruijnGraph<KMER>::cutBranchIterative(const unsigned long long numThread)
   ```
   NOTE
   After performing **Step 5.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Modify the **scaffoldGraph.cpp** file.

1. Open **scaffoldGraph.cpp**.
   vi scaffoldGraph.cpp
2. Press **i** to enter the insert mode and modify line 2280.
   Before the modification:
   ```
   2280        if (abs(insertLength - averageInsSize) > tolerence) continue;
   ```
   After the modification:
   ```
   2280        if (std::fabs(insertLength - averageInsSize) > tolerence) continue;
   ```
   NOTE
   After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Modify the **baseCommand.cpp** file.

1. Open **baseCommand.cpp**.
   vi baseCommand.cpp
2. Press **i** to enter the insert mode and modify line 175.
   Before the modification:
   ```
   175    for (;moveArgs != '\0'; ++moveArgs) {
   ```
   After the modification:
   ```
   175    for (;*moveArgs != '\0'; ++moveArgs) {
   ```
   NOTE
   After performing **Step 7.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Compile and install Platanus.
make

----End

4.95.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Set the environment variable.

```
export PATH=/path/to/PLATANUS/Platanus_v1.2.4:$PATH
```

Step 3 Upload the downloaded test file to the **/path/to/PLATANUS/CASE** directory and decompress it.

Step 4 Run the test case.

```
cd /path/to/PLATANUS/CASE/Platanus_allee_test_data_180619/test_run
platanus assemble -t 64 -tmp ./ -m 8 -o out -f ../reads/Illumina_PE500/R*.fq 2>&1 |tee
out.assemble.log
```

Figure 4-24 shows an example of the running result.

Figure 4-24 An example result

```
TOTAL_MAPPED_READS=236971
TOTAL_UNMAPPED_READS=227111
TOTAL_SHORT_READS(<54)=0
NUM_DELETE_NODE(reads are unmapped)=0
NUM_CUT_NODE=41
assemble completed!

#### PROCESS INFORMATION ####
VmPeak:          10.498 GByte
VmHWM:            6.061 GByte
```

----End

4.96 TASSEL 5.0 Porting Guide (openEuler 20.03)

4.96.1 Introduction

TASSEL is a software package to evaluate traits associations, evolutionary patterns, and linkage disequilibrium.

For more information, visit the **TASSEL official website**.

Programming language: Perl

Brief description: a software package to evaluate traits associations, evolutionary patterns, and linkage disequilibrium.

Open source license: LGPL

Recommended Software Version

TASSEL 5.0

 Copyright © Huawei Technologies Co., Ltd.

4.96.2 Environmental Requirements

Hardware Requirements

Table 4-473 lists the hardware requirements.

Table 4-473 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-474 lists the software requirements.

Table 4-474 Software requirements

Item	Version	Download URL
TASSEL	5.0	https://bitbucket.org/tasseladmin/tassel-5-standalone.git

OS Requirements

Table 4-475 lists the OS requirements.

Table 4-475 OS requirements

Item	Version	Download URL
openEuler	openEuler 20.03 LTS SP1	https://repo.openeuler.org/openEuler-20.03-LTS-SP1/
Kernel	4.19	Included in the OS image

4.96.3 Paths for Software Porting

This chapter lists the software installation paths involved in the TASSEL software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-476 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/tassel	Installation path of TASSEL	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.96.4 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/tassel* directory.

```
cd /path/to/tassel
```

Step 3 Download the TASSEL source package.

```
git clone https://bitbucket.org/tasseladmin/tassel-5-standalone.git
```

----End

4.96.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/tassel/tassel-5-standalone
```

----End

4.96.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/tassel/tassel-5-standalone
```

Step 3 Calculate the kinship matrix.

```
./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -ck -export kinship.txt
```

Step 4 Calculate the LD analysis.

```
./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -ld -ldd png -o
chr_5000sites_ld.png
./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -ld -export LD_5000sites
```

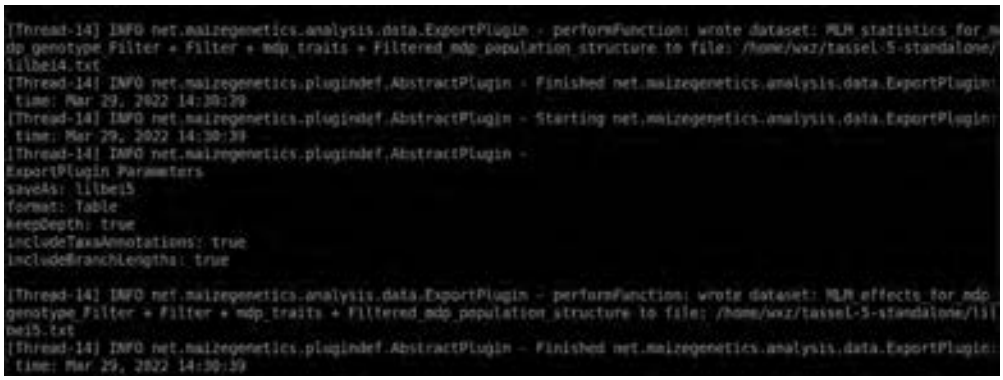
 Copyright © Huawei Technologies Co., Ltd.

Step 5 Extract chromosomes.

- Extract each chromosome.
  ./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -separate -export chromosome
- Extract chromosomes 1 and 3.
  ./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -separate 1,3 -export chromosome

Step 6 Perform MLM analysis.

./run_pipeline.pl -fork1 -h TASSELTutorialData/data/mdp_genotype.hmp.txt -filterAlign -filterAlignMinFreq 0.05 -fork2 -r TASSELTutorialData/data/mdp_traits.txt -fork3 -q TASSELTutorialData/data/mdp_population_structure.txt -excludeLastTrait -fork4 -k TASSELTutorialData/data/mdp_kinship.txt -combine5 -input1 -input2 -input3 -intersect -combine6 -input5 -input4 -mlm -export lilibei -runfork1 -runfork2 -runfork3 -runfork4



----End

4.97 MWASTools 1.20.0 Porting Guide (Kylin V10)

4.97.1 Introduction

MWASTools provides a complete pipeline to perform metabolome-wide association studies. Key functionalities of the package include: quality control analysis of metabonomic data; MWAS using different association models (partial correlations; generalized linear models); model validation using non-parametric bootstrapping; visualization of MWAS results; NMR metabolite identification using STOCSY; and biological interpretation of MWAS results.

For more information, visit the **MWASTools official website**.

Programming language: R

Brief description: a complete pipeline to perform metabolome-wide association studies.

Open source license: custom open source license

Recommended Software Version

MWASTools 1.20.0

 Copyright © Huawei Technologies Co., Ltd.

4.97.2 Environmental Requirements

Hardware Requirements

Table 4-477 lists the hardware requirements.

Table 4-477 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-478 lists the software requirements.

Table 4-478 Software requirements

Item	Version	Download URL
MWASTools	1.20.0	Automatically installed in the R language; no need to download
R language	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-479 lists the OS requirements.

Table 4-479 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.97.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MWASTools software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-480 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/MWASTools	Installation path of MWASTools	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.97.4 Configuring the Compilation Environment

4.97.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-481 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.97.4.2 Installing the R Language**.

4.97.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/MWASTools* directory.

```
cd /path/to/MWASTools
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Download the R language source package.

```
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 4 Decompress the R language source package.

```
tar xvf R-4.2.0.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd R-4.2.0
```

Step 6 Perform the compilation and installation.

```
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/MWASTOOLS/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/MWASTOOLS/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MWASTOOLS/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, run the following command to add an image source of the R language.

1. Create an **~/.Rprofile** file.
   ```
   vi ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.97.5 Obtaining the Source Code

If you use the R language to install MWASTools, you do not need to download the software source package.

4.97.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

```
R
```

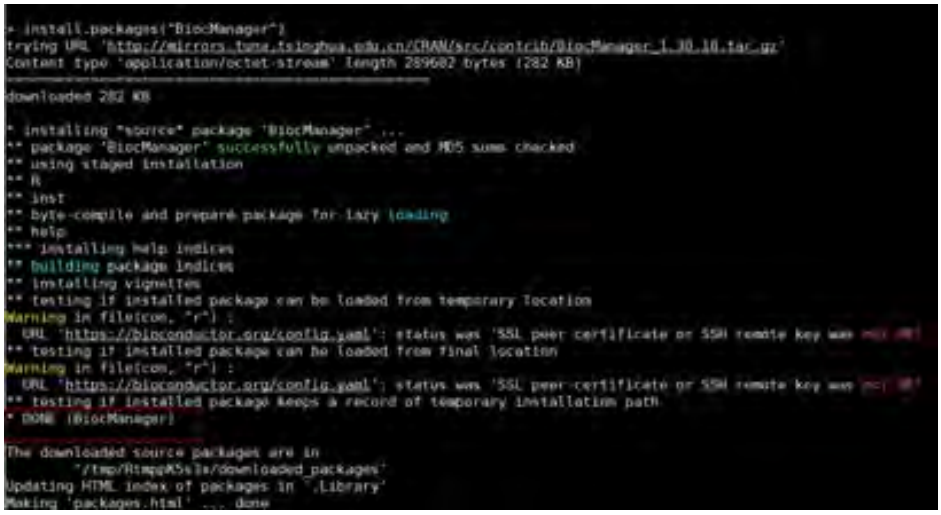
Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Enter **install.packages("BiocManager")**.

If the following information is displayed, the installation is successful:



Step 3 Enter **BiocManager::install("MWASTools")** to install MWASTools.

If the following information is displayed, the installation is successful:

```
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (MWASTools)
```

----End

4.97.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
```

Enter **citation("MWASTools")**.

Information similar to the following is displayed:

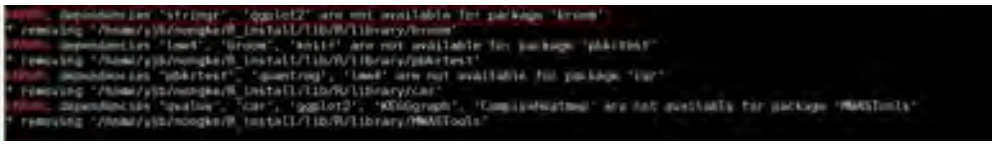
----End

4.97.8 Troubleshooting

4.97.8.1 Failed to Install the broom Dependency

Symptom

An error message similar to "ERROR: dependencies 'stringr', 'ggplot2' are not available for package 'broom'" is reported.



 Copyright © Huawei Technologies Co., Ltd.

Key Process and Cause Analysis

During the MWASTools installation process, the sub-dependencies stringr and ggplot2 of the broom dependency fail to be installed due to network timeout or mismatch between the dependency version and Bioconductor.

Conclusion and Solution

Method 1: Run the **BiocManager::install("stringr")** and **BiocManager::install("ggplot2")** commands to preferentially install the sub-dependencies stringr and ggplot2. After installing the two sub-dependencies, run the **BiocManager::install("broom")** command to install the broom dependency.

Method 2: If the installation still fails, manually install the sub-dependencies from their source code. For example, obtain the source code address of stringr based on the error message. Then download the source code of stringr to the server and run the **R CMD INSTALL xxxxx.tar.gz** command to install stringr.

4.97.8.2 Failed to Install the Rgraphviz Dependency

Symptom

The error message "ERROR: compilation failed for package 'Rgraphviz'" is reported.

Key Process and Cause Analysis

During the MWAS Tools installation process, the AArch64 architecture is not identified because the guess files in the **Rgraphviz_2.40.0** dependency are outdated.

Conclusion and Solution

Obtain the source code address of Rraphviz based on the error message. Download the Rraphviz source code to the server, decompress **Rgraphviz_2.40.0.tar.gz**, download the guess files again to overwrite the original files, and install the Rgraphviz from the source code again.

Step 1 Decompress the **Rgraphviz_2.40.0.tar.gz** source package.

```
tar xvf Rgraphviz_2.40.0.tar.gz
```

Step 2 Go to the directory generated after the decompression.

```
cd Rgraphviz
```

Step 3 Delete the guess files.

```
rm -rf ./src/graphviz/config/config.guess
rm -rf ./src/graphviz/libltdl/config/config.guess
```

Step 4 Download the guess files and rename them.

 Copyright © Huawei Technologies Co., Ltd.

```
wget "http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD" -O
config.guess
```

Step 5 Copy the files.

```
cp ./config.guess ./src/graphviz/config/
cp ./config.guess ./src/graphviz/libltdl/config/
```

Step 6 Delete **Rgraphviz_2.40.0.tar.gz**.

```
cd ..
rm -rf Rgraphviz_2.40.0.tar.gz
```

Step 7 Package the files.

```
tar czvf Rgraphviz_2.40.0.tar.gz Rgraphviz
```

Step 8 Perform the installation.

```
R CMD INSTALL Rgraphviz_2.40.0.tar.gz
```

If the command output contains "DONE (Rgraphviz)", the installation is successful.

```
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** checking absolute paths in shared objects and dynamic libraries
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installat
* DONE (Rgraphviz)
[root@localhost package]#
```

----End

4.98 sRAP 1.26.0 Porting Guide (Kylin V10)

4.98.1 Introduction

The sRAP software package provides a pipeline for gene expression analysis (primarily for RNA-Seq data). The normalization function is specific for RNA-Seq analysis, but all other functions (Quality Control Figures, Differential Expression and Visualization, and Functional Enrichment via BD-Func) will work with any type of gene expression data.

For more information, visit the **sRAP official website**.

Programming language: R

Brief description: gene expression analysis.

Open source license: custom open source license

Recommended Software Version

sRAP 1.26.0

 Copyright © Huawei Technologies Co., Ltd.

4.98.2 Environmental Requirements

Hardware Requirements

Table 4-482 lists the hardware requirements.

Table 4-482 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-483 lists the software requirements.

Table 4-483 Software requirements

Item	Version	Download URL
sRAP	1.26.0	http://www.bioconductor.org/packages/3.10/bioc/src/contrib/sRAP_1.26.0.tar.gz
R language	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-484 lists the OS requirements.

Table 4-484 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.98.3 Paths for Software Porting

This chapter lists the software installation paths involved in the sRAP software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-485 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/SRAP	Installation path of sRAP	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.98.4 Configuring the Compilation Environment

4.98.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-486 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R Language	See **4.98.4.2 Installing the R Language**.
3	Installing BiocManager	See **4.98.4.3 Installing BiocManager**.
4	Installing WriteXLS	See **4.98.4.4 Installing WriteXLS**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
5	Installing gplots	See **4.98.4.5 Installing gplots**.
6	Installing pls	See **4.98.4.6 Installing pls**.
7	Installing ROCR	See **4.98.4.7 Installing ROCR**.

4.98.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/SRAP* directory.

```
cd /path/to/SRAP
```

Step 3 Download the R language source package.

```
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 4 Decompress the R language source package.

```
tar xvf R-4.2.0.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd R-4.2.0
```

Step 6 Perform the compilation and installation.

```
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/SRAP/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/SRAP/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SRAP/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, run the following command to add an image source of the R language.

1. Create an **~/.Rprofile** file.
   ```
   vi ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.98.4.3 Installing BiocManager

Procedure

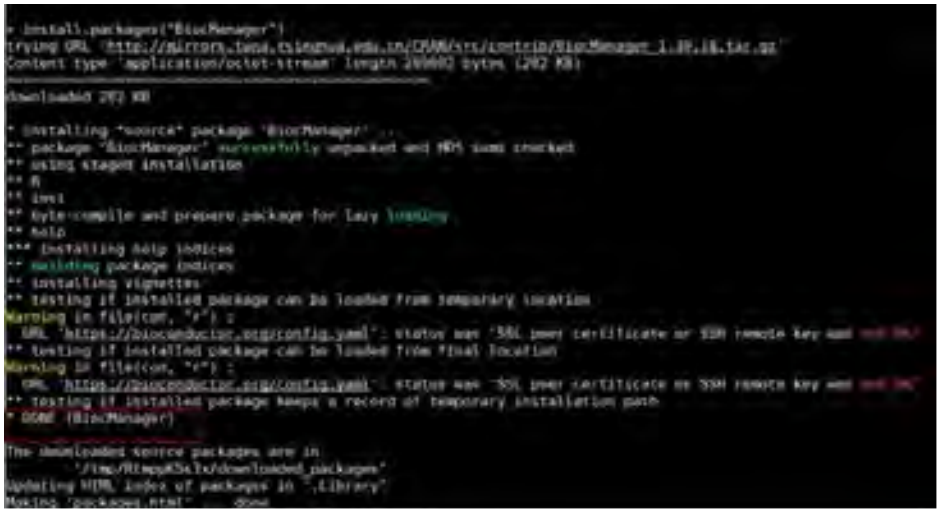
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install BiocManager.

```
R
```

Enter **install.packages("BiocManager")**.

If the following information is displayed, the installation is successful:



----End

4.98.4.4 Installing WriteXLS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/SRAP* directory.

```
cd /path/to/SRAP
```

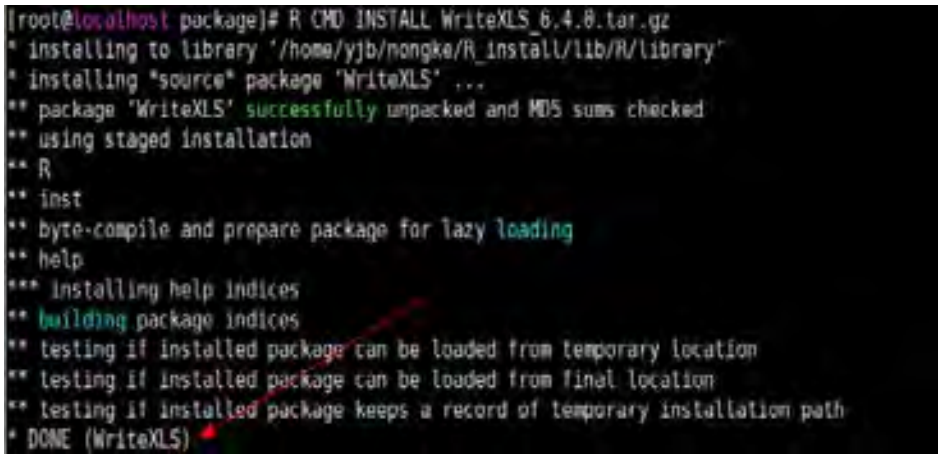
Step 3 Download the WriteXLS source package.

```
wget http://mirrors.tuna.tsinghua.edu.cn/CRAN/src/contrib/WriteXLS_6.4.0.tar.gz
```

Step 4 Install WriteXLS.

```
R CMD INSTALL WriteXLS_6.4.0.tar.gz
```

If the following information is displayed, the installation is successful:



----End

4.98.4.5 Installing gplots

Procedure

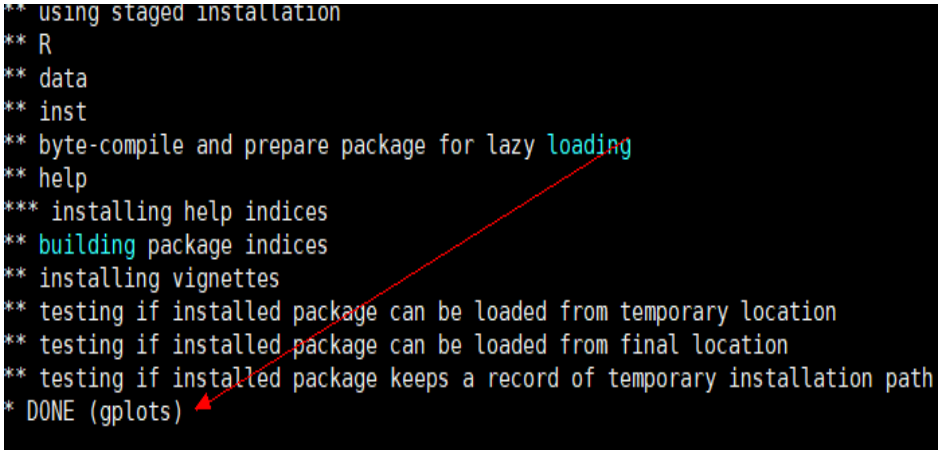
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install gplots.

```
R
```

Enter **BiocManager::install("gplots")**.

If the following information is displayed, the installation is successful:



----End

4.98.4.6 Installing pls

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/SRAP* directory.

```
cd /path/to/SRAP
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Download the pls source package.

```
wget http://mirrors.tuna.tsinghua.edu.cn/CRAN/src/contrib/pls_2.8-0.tar.gz
```

Step 4 Install pls.

```
R CMD INSTALL pls_2.8-0.tar.gz
```

If the following information is displayed, the installation is successful:

```
[root@localhost package]# R CMD INSTALL pls_2.8-0.tar.gz
* installing to library '/home/yjb/nongke/R_install/lib/R/library'
* installing *source* package 'pls' ...
** package 'pls' successfully unpacked and MD5 sums checked
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (pls)
[root@localhost package]#
```

----End

4.98.4.7 Installing ROCR

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

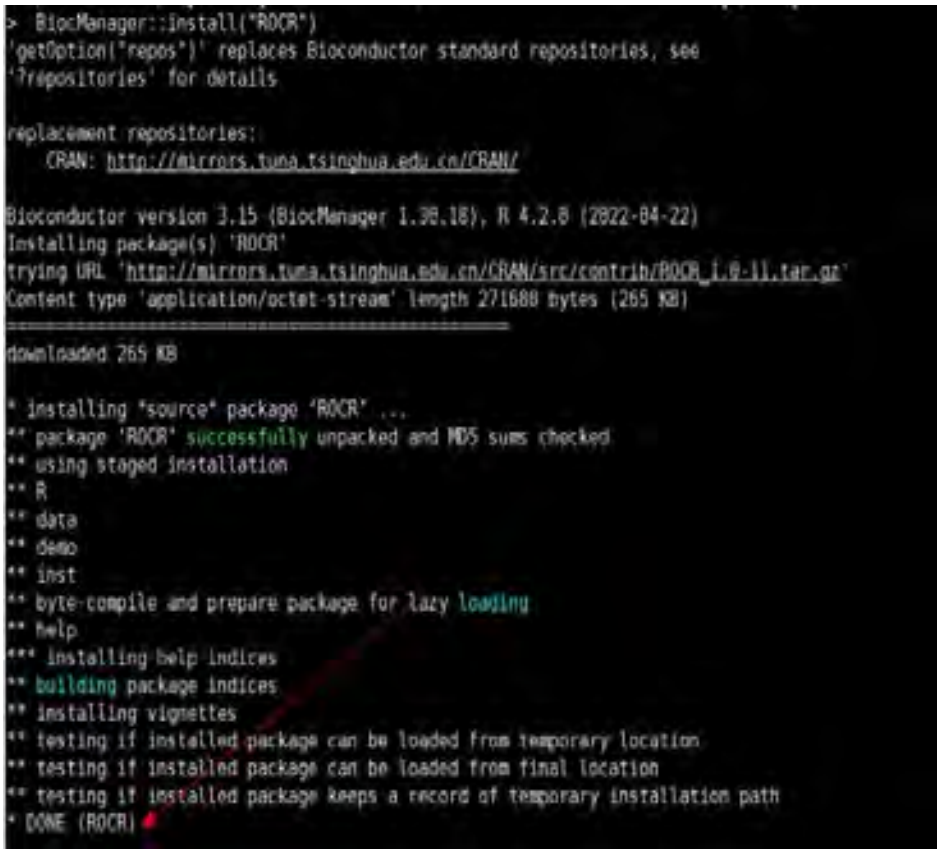
Step 2 Install ROCR.

```
R
```

Enter **BiocManager::install("ROCR")**.

If the following information is displayed, the installation is successful:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.98.5 Obtaining the Source Code

Procedure

Step 1 Download the sRAP installation package **sRAP_1.26.0.tar.gz**.

http://www.bioconductor.org/packages/3.10/bioc/src/contrib/sRAP_1.26.0.tar.gz

Step 2 Use an SFTP tool to upload the sRAP installation package to the */path/to/SRAP* directory on the server.

----End

4.98.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/SRAP
```

Step 3 Decompress the installation package and install sRAP.

```
R CMD INSTALL sRAP_1.26.0.tar.gz
```

If the following information is displayed, the installation is successful:

```
[root@localhost package]# R CMD INSTALL sRAP_1.26.0.tar.gz
* installing to library '/home/yjb/nongke/R_install/lib/R/library'
* installing *source* package 'sRAP' ...
** using staged installation
** R
** data
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (sRAP)
[root@localhost package]#
```

----End

4.98.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
```

Enter **citation("sRAP")**.

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost package]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> citation("sRAP")

When using RNA.bdfunc.fc or RNA.bdfunc.signal, please cite the BD-Func
publication.  For all other functions, please site the fold-change
paper.

  Warden CD et al., Optimal Calculation of RNA-Seq Fold-Change Values,
  Int J Comput Bioinfo In Silico Model (2014) 2(6): 285-292

  Warden CD et al., BD-Func: A Streamlined Algorithm for Predicting
  Activation and Inhibition of Pathways, peerJ (2013) 1:e159

To see these entries in BibTeX format, use 'print(<citation>,
bibtex=TRUE)', 'toBibtex(.)', or set
'options(citation.bibtex.max=999)'.
```

----End

4.99 BRAKER 2.1.6 Porting Guide (Kylin V10)

4.99.1 Introduction

BRAKER2 is an extension of BRAKER1 which allows for fully automated training of the gene prediction tools GeneMark-EX R14, R15, R17, F1 and AUGUSTUS from RNA-Seq and/or protein homology information, and that integrates the extrinsic evidence from RNA-Seq and protein homology information into the prediction.

For more information, visit the **BRAKER official website**.

Programming language: Perl

Brief description: a pipeline for fully automated prediction of protein coding gene structures with GeneMark-ES/ET and AUGUSTUS in novel eukaryotic genomes.

Open source license: custom open source license

Recommended Software Version

BRAKER 2.1.6

4.99.2 Environmental Requirements

Hardware Requirements

Table 4-487 lists the hardware requirements.

Table 4-487 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-488 lists the software requirements.

Table 4-488 Software requirements

Item	Version	Download URL
BRAKER	2.1.6	https://github.com/Gaius-Augustus/BRAKER/archive/refs/tags/v2.1.6.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

OS Requirements

Table 4-489 lists the OS requirements.

Table 4-489 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.99.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BRAKER software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-490 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/BRAKER	Installation path of BRAKER	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.99.4 Configuring the Compilation Environment

4.99.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-491 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing dependencies	See **4.99.4.2 Installing Dependencies**.
4	Installing GeneMark	See **4.99.4.3 Installing GeneMark**.
5	Installing Augustus	See **Augustus 3.3.3 Porting Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
6	Installing DIAMOND	See **DIAMOND 2.0.15 Porting Guide**.
7	Installing CDBtools	See **4.99.4.4 Installing CDBtools**.
8	Installing Samtools	See "Installing Samtools" in **Augustus 3.3.3 Porting Guide**.
9	Installing BamTools	See "Installing BamTools" in **Augustus 3.3.3 Porting Guide**.
10	Installing ProtHint	See **4.99.4.5 Installing ProtHint**.

4.99.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install perl-CPAN
cpan -i File::Spec::Functions
cpan -i Hash::Merge
cpan -i List::Util
cpan -i MCE::Mutex
cpan -i Module::Load::Conditional
cpan -i Parallel::ForkManager
cpan -i POSIX
cpan -i Scalar::Util::Numeric
cpan -i YAML
cpan -i Math::Utils
cpan -i File::HomeDir
cpan -i threads
```

----End

4.99.4.3 Installing GeneMark

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install ExaGear for Server. For details, see "Installing ExaGear for Server on CentOS " in **Binary Translator (ExaGear) User Guide**.

Step 3 Install the dependencies.

```
yum install perl-CPAN
cpan -i YAML
cpan -i Hash::Merge
cpan -i Logger::Simple
cpan -i Parallel:ForkManager
cpan -i MCE::Mutex
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Download the installation package.

http://exon.gatech.edu/GeneMark/license_download.cgi

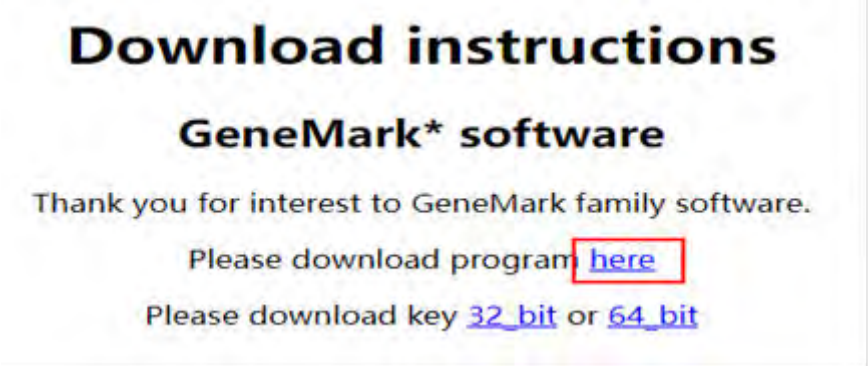
NOTE

You need to submit an application before downloading the software. Download both the software and license file.

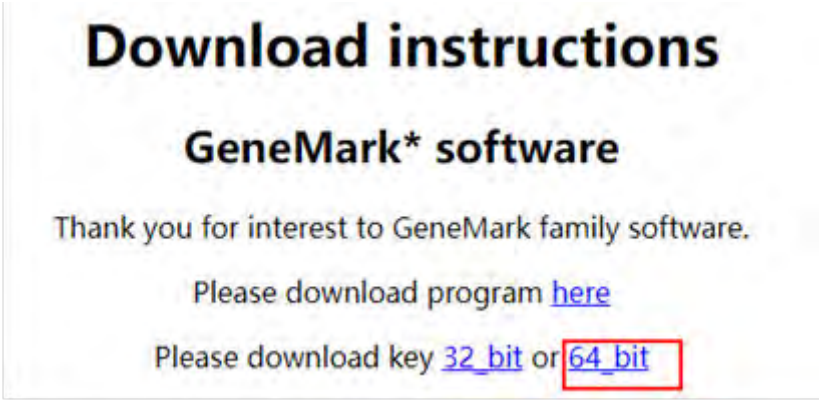
1. Fill in the application form.

2. Download the software.

 Copyright © Huawei Technologies Co., Ltd.



3. Download the license key.



Step 5 Install the software.

1. Go to the installation package directory.
   ```
   cd /path/to/GeneMark
   ```
2. Decompress the installation package.
   ```
   tar xvf gmes_linux_64_4.tar.gz
   ```
3. Decompress the license file.
   ```
   gunzip gm_key_64
   ```
4. Configure the license.
   ```
   cd gmes_linux_64_4
   mv ../gmes_linux_64_4 ./.gm_key
   ```
5. Set the environment variable.
   ```
   export PATH=/path/to/GeneMark/gmes_linux_64_4:$PATH
   ```

----End

4.99.4.4 Installing CDBtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/BRAKER* directory.

```
cd /path/to/BRAKER
```

Step 3 Download the CDBtools installation package.

```
wget https://codeload.github.com/gpertea/cdbfasta/zip/refs/heads/master
```

Step 4 Decompress the CDBtools installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
tar xvf cdbfasta-master.zip
```

Step 5 Go to the directory generated after the decompression.

```
cd cdbfasta
```

Step 6 Perform the compilation and installation.

```
make
```

Step 7 Load the environment variable.

```
export PATH=/path/to/BRAKER/cdbfasta:$PATH
```

----End

4.99.4.5 Installing ProtHint

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/BRAKER* directory.

```
cd /path/to/BRAKER
```

Step 3 Download and decompress the ProtHint installation package.

```
wget https://github.com/gatech-genemark/ProtHint/archive/refs/tags/v2.6.0.tar.gz
tar xvf v2.6.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd ProHint-2.6.0
```

Step 5 Download the dependency package.

```
wget https://github.com/ogotoh/spaln/archive/refs/tags/veer.2.4.9b.tar.gz
```

Step 6 Decompress the dependency package.

```
tar xvf veer.2.4.9b.tar.gz
```

Step 7 Go to the directory generated after the decompression.

```
cd spaln-veer.2.4.9b/src
```

Step 8 Perform the configuration.

```
CXX=g++ ./configure
```

Step 9 Modify the **codepot.h** file.

1. Open **codepot.h**.
   ```
   vi codepot.h
   ```
2. Press **i** to enter the insert mode and modify line 37.

   Before the modification:

   ```
   static  const   EXIN    ZeroExin = {0, 0, 0, 0, 0, 0, -2, -2};
   ```

   After the modification:

   ```
   static  const   EXIN    ZeroExin = {0, 0, 0, 0, 0, 0, (char)-2, (char)-2};
   ```

   NOTE

   After performing **Step 9.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Perform the compilation and installation.

```
make
make install
```

Step 11 Load the environment variable.

```
export PATH=/path/to/BRAKER/ProHint-2.6.0/spaln-veer.2.4.9b/bin:$PATH
```

Step 12 Go to the **ProHint-2.6.0** directory.

```
cd /path/to/BRAKER/ProHint-2.6.0
```

Step 13 Download **spaln-boundary-scorer.git**.

```
git clone https://github.com/gatech-genemark/spaln-boundary-scorer.git
```

Step 14 Decompress the installation package.

```
tar xvf spaln-boundary-scorer.git
```

Step 15 Go to the directory generated after the decompression.

```
cd spaln-boundary-scorer
```

Step 16 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vi Makefile
   ```
2. Press **i** to enter the insert mode and change the content in the first line to **CC=clang++**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 17 Perform the compilation.

```
make
```

Step 18 Set the environment variable.

```
export PATH=/path/to/BRAKER/ProHint-2.6.0/spaln-boundary-scorer:$PATH
```

Step 19 Replace the dependencies.

```
cd /path/to/BRAKER/ProHint-2.6.0/dependencies
rm -rf spaln_table spaln spaln-boundary-scorer diamond
cp -r /path/to/BRAKER/ProHint-2.6.0/spaln-veer.2.4.9b/table spaln_table
cp /path/to/BRAKER/ProHint-2.6.0/spaln-veer.2.4.9b/bin/spaln .
cp /path/to/BRAKER/ProHint-2.6.0/spaln-boundary-scorer/spaln-boundary-scorer .
```

Step 20 Copy the executable file **diamond** (generated after DIAMOND is installed in **Table 4-491**) to the current directory.

----End

4.99.5 Obtaining the Source Code

Procedure

Step 1 Download the BRAKER installation package.

https://github.com/Gaius-Augustus/BRAKER/archive/refs/tags/v2.1.6.tar.gz

Step 2 Use an SFTP tool to upload the software installation package and license file to the **/path/to/BRAKER** directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.99.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/BRAKER* directory.

```
cd /path/to/BRAKER
```

Step 3 Decompress the installation package.

```
tar xvf v2.1.6.tar.gz
```

Step 4 Set the environment variable of the executable file.

```
export PATH=/path/to/BRAKER/BRAKER-2.1.6/scripts:$PATH
```

----End

4.99.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test case directory.

```
cd /path/to/BRAKER/BRAKER-2.1.6/example
```

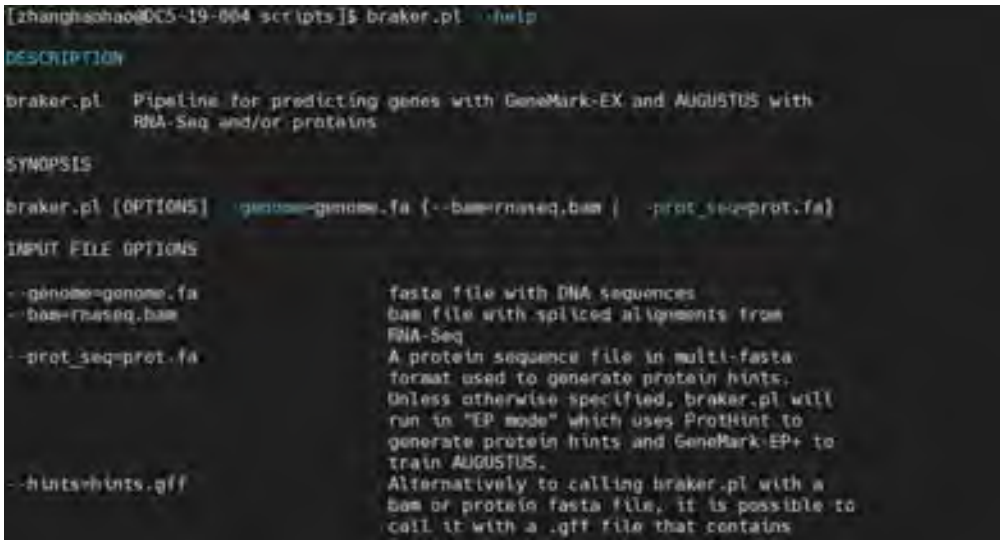
Step 3 Set the environment variable of each dependency.

1. Add the environment variable **AUGUSTUS** to **config** in the installation directory.
   ```
   export AUGUSTUS_CONFIG_PATH=/path/to/augustus_config
   ```
2. Add the environment variable **AUGUSTUS** to **bin** in the installation directory.
   ```
   export AUGUSTUS_BIN_PATH=/path/to/augustus_bin
   ```
3. Add the environment variable **AUGUSTUS** to **scripts** in the installation directory.
   ```
   export AUGUSTUS_SCRIPTS_PATH=/path/to/augustus_scripts
   ```
4. Add the environment variable **DIAMOND** to the directory of the executable file **diamond**.
   ```
   export DIAMOND_PATH=/path/to/diamond
   ```
5. Add the environment variable **GENEMARK** to the directory of the executable file **gmhmme3**.
   ```
   export GENEMARK_PATH=/path/to/GENEMARK
   ```
6. Add the environment variable **BAMTOOLS** to the directory of the executable file **bamtools**.
   ```
   export BAMTOOLS_PATH=/path/to/BAMTOOLS
   ```
7. Add the environment variable **CDBTOOLS** to the directory of the executable file **cdbfasta**.
   ```
   export CDBTOOLS_PATH=/path/to/cdbfasta
   ```
8. Add the environment variable **SAMTOOLS** to the directory of the executable file **samtools**.
   ```
   export SAMTOOLS_PATH=/path/to/samtools
   ```

Step 4 Perform the verification.

```
braker.pl --help
```

 Copyright © Huawei Technologies Co., Ltd.



----End

4.100 Cactus 2.1.0 Porting Guide (Kylin V10)

4.100.1 Introduction

Cactus is a reference-free whole-genome alignment program, as well as a pagenome graph construction toolkit.

For more information, visit the **Cactus official website**.

Programing language: C/Python

Brief description: a reference-free whole-genome alignment program, as well as a pagenome graph construction toolkit.

Open source license: custom open source license

Recommended Software Version

Cactus 2.1.0

4.100.2 Environmental Requirements

Hardware Requirements

Table 4-492 lists the hardware requirements.

Table 4-492 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-493 lists the software requirements.

Table 4-493 Software requirements

Item	Version	Download URL
Cactus	2.1.0	https://github.com/ComparativeGenomicsToolkit/cactus/releases/download/v2.1.0/cactus-v2.1.0.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

OS Requirements

Table 4-494 lists the OS requirements.

Table 4-494 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.100.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Cactus software porting.

Table 4-495 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/Cactus	Installation path of Cactus	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.100.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-496 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.100.5 Obtaining the Source Code

Procedure

Step 1 Download the Cactus installation package.

https://github.com/ComparativeGenomicsToolkit/cactus/releases/download/v2.1.0/cactus-v2.1.0.tar.gz

Step 2 Use an SFTP tool to upload the Cactus installation package to the */path/to/Cactus* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.100.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y python3 python3-pip
```

Step 3 Go to the */path/to/Cactus* directory.

```
cd /path/to/Cactus
```

Step 4 Decompress the installation package.

```
tar -xvf cactus-v2.1.0.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd cactus-v2.1.0
```

Step 6 Perform the compilation and installation.

```
python3 -m pip install setuptools
python3 -m pip install -U -r ./toil-requirement.txt
python3 -m pip install -U .
```

NOTE

If the installation with **pip** fails due to network problems, you can try the Tsinghua source in China and add **--trusted-host pypi.python.org --trusted-host pypi.tuna.tsinghua.edu.cn -i https://pypi.tuna.tsinghua.edu.cn/simple** to the end of the command.

Step 7 Check whether a binary file is generated.

```
ll /usr/local/bin/cactus
```

If the following information is displayed, the installation is successful:

```
[root@localhost cactus-v2.1.0]# ll /usr/local/bin/cactus
-rwxr-xr-x 1 root root 236 Jun 27 18:06 /usr/local/bin/cactus
```

----End

4.100.7 Running and Verification

Procedure

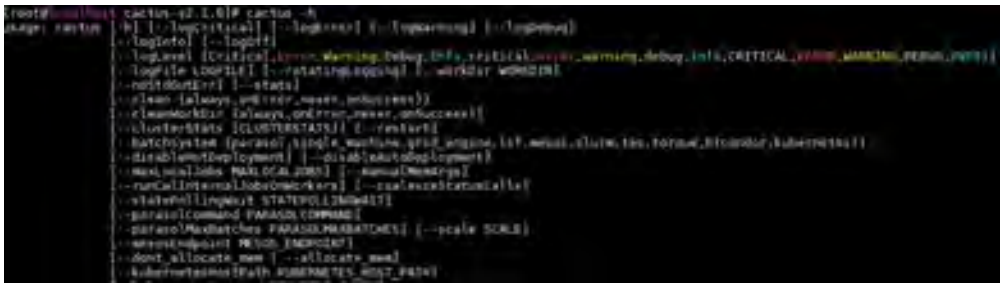
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cactus -h
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.101 CAFE 4.2.1 Porting Guide (Kylin V10)

4.101.1 Introduction

The purpose of CAFE is to analyze changes in gene family size in a way that accounts for phylogenetic history and provides a statistical foundation for evolutionary inferences. The program uses a birth and death process to model gene gain and loss across a user-specified phylogenetic tree. The distribution of family sizes generated under this model can provide a basis for assessing the significance of the observed family size differences among taxa.

For more information, visit the **CAFE official website**.

Programming language: C++/C/Perl

Brief description: CAFE analyzes changes in gene family size and provides a statistical foundation for evolutionary inferences.

Open source license: custom open source license

Recommended Software Version

CAFE 4.2.1

4.101.2 Environmental Requirements

Hardware Requirements

Table 4-497 lists the hardware requirements.

Table 4-497 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-498 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-498 Software requirements

Item	Version	Download URL
CAFE	4.2.1	https://github.com/hahnlab/CAFE/archive/refs/tags/v4.2.1.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

OS Requirements

Table 4-499 lists the OS requirements.

Table 4-499 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.101.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CAFE software porting.

Table 4-500 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/CAFE	Installation path of CAFE	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.101.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-501 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.101.5 Obtaining the Source Code

Procedure

Step 1 Download the CAFE installation package.

https://github.com/hahnlab/CAFE/archive/refs/tags/v4.2.1.tar.gz

Step 2 Use an SFTP tool to upload the CAFE installation package to the */path/to/CAFE* directory on the server.

----End

4.101.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/CAFE* directory.

```
cd /path/to/CAFE
```

Step 3 Decompress the installation package.

```
tar -xvf v4.2.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd CAFE-4.2.1
```

Step 5 Create a compilation file.

```
antoconf
./configure
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Modify the **cafe/cafe_tree.c** file.

1. Open **cafe/cafe_tree.c**.
   ```
   vi cafe/cafe_tree.c
   ```
2. Press **i** to enter the insert mode and modify the file as follows:
   - Delete **if (aq1)** from lines 74 and 81.

     Before the modification:

     ```
     71 void __cafe_tree_free_node(pTree ptree, pTreeNode ptnode, va_list ap1)
     72 {
     73     va_list ap;
     74     if (ap1) va_copy(ap, ap1);
     75     pCafeNode pcnode = (pCafeNode)ptnode;
     76     if (pcnode->likelihoods) memory_free(pcnode->likelihoods);
     77     pcnode->likelihoods = NULL;
     78     if (pcnode->viterbi) memory_free(pcnode->viterbi);
     79     pcnode->viterbi = NULL;
     80     __phylogeny_free_node(ptree, ptnode, ap);
     81     if (ap1) va_end(ap1);
     ```

     After the modification:

     ```
     71 void __cafe_tree_free_node(pTree ptree, pTreeNode ptnode, va_list ap1)
     72 {
     73     va_list ap;
     74     va_copy(ap, ap1);
     75     pCafeNode pcnode = (pCafeNode)ptnode;
     76     if (pcnode->likelihoods) memory_free(pcnode->likelihoods);
     77     pcnode->likelihoods = NULL;
     78     if (pcnode->viterbi) memory_free(pcnode->viterbi);
     79     pcnode->viterbi = NULL;
     80     __phylogeny_free_node(ptree, ptnode, ap);
     81     va_end(ap1);
     82 }
     ```

   - Add **va_list a;** and **va_end (a);** under line 91.
   - Change **NULL** in line 94 to **a**.

     Before the modification:

     ```
     85 void cafe_tree_free(pCafeTree pcafe)
     86 {
     87     if ( pcafe->super.nlist )
     88     {
     89         int i;
     90         for ( i = 0 ; i < pcafe->super.nlist->size ; i++ )
     91         {
     92             __cafe_tree_free_node((pTree)pcafe, (pTreeNode)pcafe->super.nlist->array[i], NULL );
     93         }
     94     }
     ```

     After the modification:

     ```
     85 void cafe_tree_free(pCafeTree pcafe)
     86 {
     87     if ( pcafe->super.nlist )
     88     {
     89         int i;
     90         for ( i = 0 ; i < pcafe->super.nlist->size ; i++ )
     91         {
     92             va_list a;
     93             va_end (a);
     94             __cafe_tree_free_node((pTree)pcafe, (pTreeNode)pcafe->super.nlist->array[i], a );
     95         }
     96     }
     ```

   - Add **va_list b;** and **va_end (b);** under line 318.
   - Change **NULL** in line 321 to **b**.

     Before the modification:

```
317 #pragma omp taskwait
318     compute_node_likelihoods(ptree, ptnode, NULL);
319 }
```

After the modification:

```
318 #pragma omp taskwait
319     va_list b;
320     va_end (b);
321     compute_node_likelihoods(ptree, ptnode, b);
322 }
```

NOTE

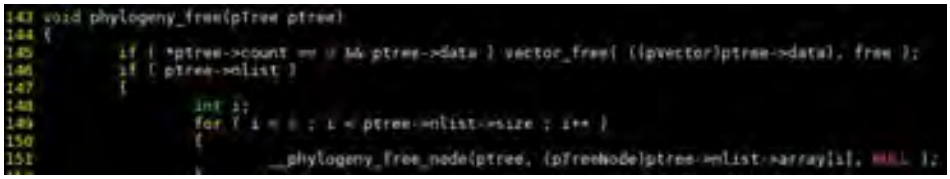
After performing **Step 6.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

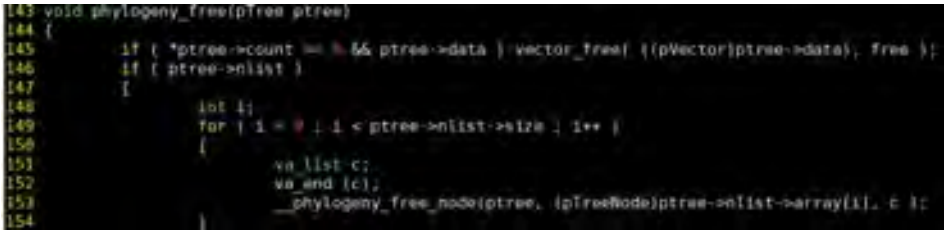
Step 7 Modify the **libtree/phylogeny.c** file.

1. Open **libtree/phylogeny.c**.
   vi libtree/phylogeny.c
2. Press **i** to enter the insert mode and modify the file as follows:
   - Add **va_list c;** and **va_end (c);** under line 150.
   - Change **NULL** in line 153 to **c**.

     Before the modification:

     After the modification:

   - Add **va_list d;** and **va_end (d);** under line 589.
   - Change **NULL** in line 592 to **d**.
   - Add **va_list e;** and **va_end (e);** under line 609.
   - Change **NULL** in line 612 to **e**.

     Before the modification:

 Copyright © Huawei Technologies Co., Ltd.

```
573 pTree phylogeny_split_tree(pTree ptree, int idx, tree_func_node freenode )
574 {
575         pTreeNode pnode = (pTreeNode)ptree->nlist->array[idx];
576         if ( tree_is_root(ptree,(pTreeNode)pnode) ) return NULL;
577         pTree psubtree = (pTree)memory_new(1,ptree->size);
578 //      memcpy(psubtree,ptree, sizeof(ptree->size));
579         psubtree->count = ptree->count;
580         (*psubtree->count)++;
581         psubtree->size = ptree->size;
582         psubtree->nlist = NULL;
583         psubtree->postfix = NULL;
584         psubtree->prefix = NULL;
585
586         psubtree->root = pnode;
587         pTreeNode sibling = phylogeny_get_sibling(ptree,pnode);
588         if ( tree_is_root(ptree,pnode->parent) )
589         {
590                 freenode(ptree, ptree->root,NULL);
591                 ptree->root = sibling;
592                 ptree->root->parent = NULL;
593         }
594         else
595         {
596                 ((pPhylogenyNode)sibling)->branchlength += ((pPhylogenyNode)s
597                 pTreeNode parent = pnode->parent;
598                 pTreeNode grand = parent->parent;
599                 if ( grand->children->head->data == parent )
600                 {
601                         grand->children->head->data = sibling;
602                 }
603                 else
604                 {
605                         grand->children->tail->data = sibling;
606                 }
607                 sibling->parent = grand;
608                 freenode(ptree,parent,NULL);
609         }
```

After the modification:

```
586         psubtree->root = pnode;
587         pTreeNode sibling = phylogeny_get_sibling(ptree,pnode);
588         if ( tree_is_root(ptree,pnode->parent) )
589         {
590                 va_list d;
591                 va_end (d);
592                 freenode(ptree, ptree->root,d);
593                 ptree->root = sibling;
594                 ptree->root->parent = NULL;
595         }
596         else
597         {
598                 ((pPhylogenyNode)sibling)->branchlength += ((pPhyloe
599                 pTreeNode parent = pnode->parent;
600                 pTreeNode grand = parent->parent;
601                 if ( grand->children->head->data == parent )
602                 {
603                         grand->children->head->data = sibling;
604                 }
605                 else
606                 {
607                         grand->children->tail->data = sibling;
608                 }
609                 sibling->parent = grand;
610                 va_list e;
611                 va_end (e);
612                 freenode(ptree,parent,e);
```

- Add **va_list f;** and **va_end (f);** under line 626.
- Change **sp1** in line 629 to **f**.
- Add **va_list g;** and **va_end (g);** under line 646.

- Change **sp1** in line 649 to **g**.

  Before the modification:

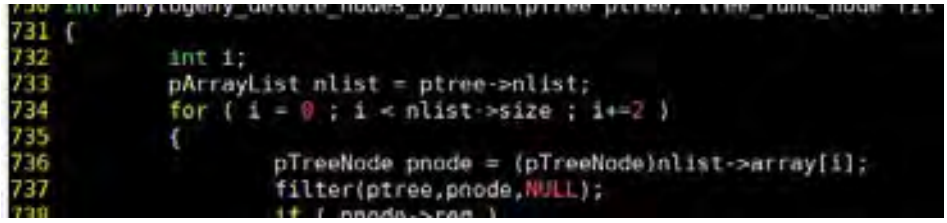
```
618 void __phylogeny_delete_nodes(pTree ptree, pTreeNode pnode, tree_func_node freene
619 {
620         va_list sp1;
621         pTreeNode sibling = phylogeny_get_sibling(ptree,pnode);
622         if ( tree_is_root(ptree,pnode->parent) )
623         {
624                 freenode(ptree, ptree->root,sp1);
625                 ptree->root = sibling;
626                 ptree->root->parent = NULL;
627         }
628         else
629         {
630                 ((pPhylogenyNode)sibling)->branchlength += ((pPhylogenyNode)sibl
631                 pTreeNode parent = pnode->parent;
632                 pTreeNode grand = parent->parent;
633                 if ( grand->children->head->data == parent )
634                 {
635                         grand->children->head->data = sibling;
636                 }
637                 else
638                 {
639                         grand->children->tail->data = sibling;
640                 }
641                 sibling->parent = grand;
642                 freenode(ptree,parent,sp1);
643         }
```

  After the modification:

```
622 void __phylogeny_delete_nodes(pTree ptree, pTreeNode pnode, tree_func_node freenode )
623 {
624         pTreeNode sibling = phylogeny_get_sibling(ptree,pnode);
625         if ( tree_is_root(ptree,pnode->parent) )
626         {
627                 va_list f;
628                 va_end (f);
629                 freenode(ptree, ptree->root,f);
630                 ptree->root = sibling;
631                 ptree->root->parent = NULL;
632         }
633         else
634         {
635                 ((pPhylogenyNode)sibling)->branchlength += ((pPhylogenyNode)sibling->pa
636                 pTreeNode parent = pnode->parent;
637                 pTreeNode grand = parent->parent;
638                 if ( grand->children->head->data == parent )
639                 {
640                         grand->children->head->data = sibling;
641                 }
642                 else
643                 {
644                         grand->children->tail->data = sibling;
645                 }
646                 sibling->parent = grand;
647                 va_list g;
648                 va_end (g);
649                 freenode(ptree,parent,g);
```

- Add **va_list h;** and **va_end (h);** under line 736.
- Change **NULL** in line 739 to **h**.

  Before the modification:



```
731 {
732         int i;
733         pArrayList nlist = ptree->nlist;
734         for ( i = 0 ; i < nlist->size ; i+=2 )
735         {
736                 pTreeNode pnode = (pTreeNode)nlist->array[i];
737                 filter(ptree,pnode,NULL);
```

  After the modification:

 Copyright © Huawei Technologies Co., Ltd.

```
735         {
736                 pTreeNode pnode = (pTreeNode)nlist->array[i];
737                 va_list h;
738                 va_end (h);
739                 filter(ptree,pnode,h);
740                 if ( pnode->reg )
741                 {
```

NOTE

After performing Step 7.1, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Modify the **cafe/viterbi.cpp** file.

1. Open **cafe/viterbi.cpp**.
   vi cafe/viterbi.cpp
2. Press **i** to enter the insert mode and modify the file as follows:
   - Add **va_list m;** and **va_end (m);** under line 502.
   - Change **NULL** in line 505 to **m**.
   - Add **va_list j;** and **va_end (j);** under line 508.
   - Change **NULL** in line 511 to **j**.

     Before the modification:

```
496  if (pcafe->super.postfix)
497  {
498    pArrayList postfix = pcafe->super.postfix;
499    pArrayList prefix = pcafe->super.prefix;
500    int i = 0;
501    for (i = 0; i < postfix->size; i++)
502    {
503      __cafe_tree_node_compute_viterbi((pTree)pcafe, (pTreeNode)postfix->array[i], NULL);
504    }
505    for (i = 0; i < prefix->size; i++)
506    {
507      __cafe_tree_node_backtrack_viterbi((pTree)pcafe, (pTreeNode)prefix->array[i], NULL);
```

     After the modification:

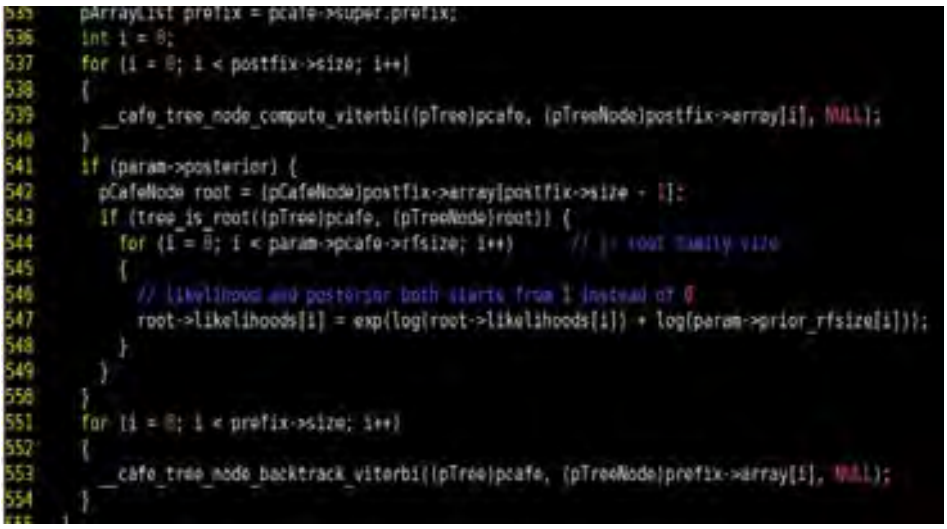
```
500    int i = 0;
501    for (i = 0; i < postfix->size; i++)
502    {
503      va_list m;
504      va_end (m);
505      __cafe_tree_node_compute_viterbi((pTree)pcafe, (pTreeNode)postfix->array[i], m);
506    }
507    for (i = 0; i < prefix->size; i++)
508    {
509      va_list j;
510      va_end (j);
511      __cafe_tree_node_backtrack_viterbi((pTree)pcafe, (pTreeNode)prefix->array[i], j);
512    }
513  }
514  else
515  {
```

   - Add **va_list k;** and **va_end (k);** under line 538.
   - Change **NULL** in line 541 to **k**.

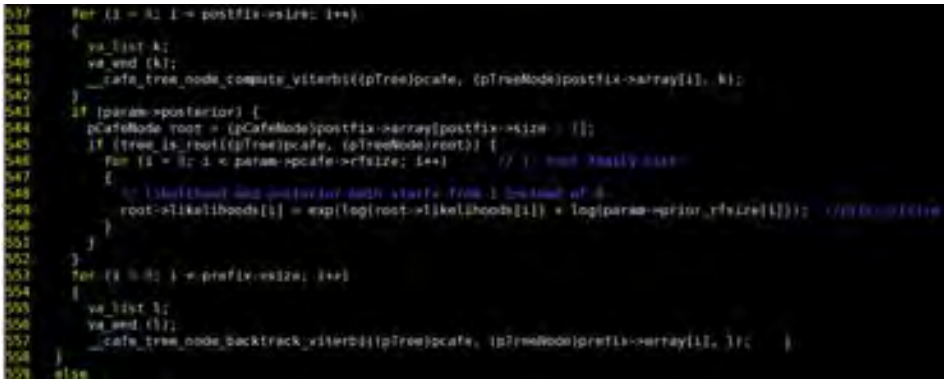
 Copyright © Huawei Technologies Co., Ltd.

- Add **va_list l;** and **va_end (l);** under line 554.
- Change **NULL** in line 557 to **l**.

  Before the modification:



  After the modification:



  NOTE

  After performing **Step 8.1**, press **Esc**, type **:set nu**, and press **Enter** to display the line number.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Perform the compilation.

```
make
```

----End

4.101.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

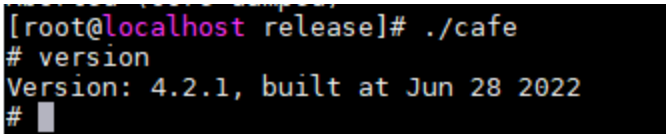
Step 2 Perform the verification.

```
cd /path/to/CAFE/CAFE-4.2.1/release
./cafe
```

Enter **version**.

 Copyright © Huawei Technologies Co., Ltd.

Information similar to the following is displayed:



----End

4.102 Clustal X 2.1 Porting Guide (Kylin V10)

4.102.1 Introduction

The Clustal series of programs are widely used in molecular biology for the multiple alignment of both nucleic acid and protein sequences and for preparing phylogenetic trees. The most familiar version is ClustalW, which uses a simple text menu system that is portable to more or less all computer systems. ClustalX features a graphical user interface and some powerful graphical utilities for aiding the interpretation of alignments and is the preferred version for interactive usage.

For more information, visit the **Clustal X official website**.

Programming language: C++

Brief description: multiple alignment of nucleic acid and protein sequences.

Open source license: LGPL-3.0

Recommended Software Version

Clustal X 2.1

4.102.2 Environmental Requirements

Hardware Requirements

Table 4-502 lists the hardware requirements.

Table 4-502 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-503 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-503 Software requirements

Item	Version	Download URL
Clustal X	2.1	**https://github.com/moble/clustalx.git**
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**

OS Requirements

Table lists the OS requirements.

Table 4-504 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.18.0 (4.14)	Included in the OS image

4.102.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Clustal X software porting.

Table 4-505 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/clustalx	Installation path of Clustal X	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.102.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-506 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.102.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
git clone https://github.com/moble/clustalx.git
```

----End

4.102.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the Qt5 dependency.

```
yum install qt5 qt5-devel
```

Step 3 Go to the main program installation directory.

```
cd clustalx/
```

Step 4 Perform the compilation and installation.

```
qmake
make
mkdir bin/
cp col*xml bin/ && cp clustalx bin/
```

Step 5 Load the environment variable.

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/clustalx/bin:PATH
```

----End

4.103 CNVfilteR 1.10.0 Porting Guide (Kylin V10)

4.103.1 Introduction

CNVfilteR identifies those copy number variants (CNVs) that can be discarded by using the single nucleotide variant (SNV) calls that are usually obtained in common NGS pipelines. Many tools for germline CNV detection from NGS data have been developed. Usually, these tools were designed for different input data like WGS, WES or panel data, and their performance may depend on the CNV size. Available benchmarks show that all these tools obtain false positives, sometimes reaching a very high number of them.

With the aim of reducing the number of false positives, CNVfilteR identifies those germline CNVs that can be discarded. This task is performed by using the germline SNV calls that are usually obtained in common NGS pipelines. As VCF field interpretation is key when working with these files, CNVfilteR specifically supports VCFs produced by VarScan2, Strelka/Strelka2, freeBayes, HaplotypeCaller, UnifiedGenotyper and Torrent Variant Caller. Additionally, results can be plotted using the functions provided by the R/Bioconductor packages karyoploteR and CopyNumberPlots.

For more information, visit the **CNVfilteR official website**.

Programming language: R

Brief description: identifies those CNVs that can be discarded by using the SNV calls that are usually obtained in common NGS pipelines.

Open source license: Artistic License 2.0

Recommended Software Version

CNVfilteR 1.10.0

4.103.2 Environmental Requirements

Hardware Requirements

Table 4-507 lists the hardware requirements.

Table 4-507 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-508 lists the software requirements.

Table 4-508 Software requirements

Item	Version	Download URL
R language	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-509 lists the OS requirements.

Table 4-509 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.103.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CNVfilteR software porting.

Table 4-510 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/CNVFILTER	Installation path of CNVfilteR	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.103.4 Configuring the Compilation Environment

4.103.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-511 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R Language	See **4.103.4.2 Installing the R Language**.

4.103.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/CNVFILTER* directory.

```
cd /path/to/CNVFILTER
```

Step 3 Download the R language source package.

```
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 4 Decompress the source package.

```
tar xvf R-4.2.0.tar.gz
```

Step 5 Go to the directory generated after the decompression.

```
cd R-4.2.0
```

Step 6 Perform the compilation and installation.

```
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/CNVFILTER/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/CNVFILTER/R_install/bin:$PATH
export LB_LIBRARY_PATH=/path/to/CNVFILTER/R_install/lib/R/lib:$ LB_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, run the following command to add an image source of the R language.

 Copyright © Huawei Technologies Co., Ltd.

1. Create an **~/.Rprofile** file.
   ```
   vi ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.103.5 Obtaining the Source Code

If you use the R language to install CNVfilteR, you do not need to download the software source package.

4.103.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

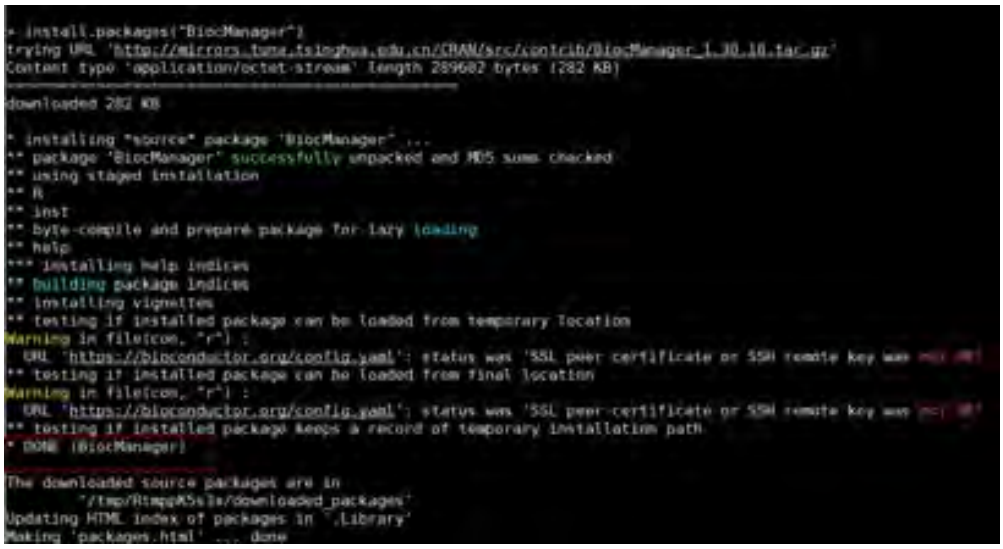
```
R
```

Information similar to the following is displayed:

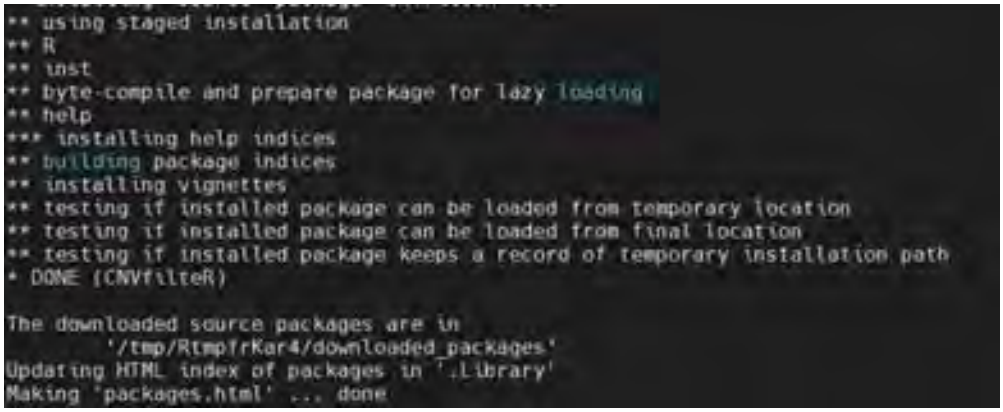
```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Enter **install.packages("BiocManager")**.

If the following information is displayed, the installation is successful:



Enter **BiocManager::install("CNVfilteR")** to install CNVfilteR.



----End

4.103.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
```

Enter **citation("CNVfilteR")**.

Information similar to the following is displayed:

----End

4.104 DIAMOND 2.0.15 Porting Guide (Kylin V10)

4.104.1 Introduction

DIAMOND is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data.

For more information, visit the **DIAMOND official website**.

Programming language: C/C++

Brief description: a sequence aligner for protein and translated DNA searches.

Open source license: custom open source license

Recommended Software Version

DIAMOND 2.0.15

4.104.2 Environmental Requirements

Hardware Requirements

Table 4-512 lists the hardware requirements.

Table 4-512 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-513 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-513 Software requirements

Item	Version	Download URL
DIAMOND	2.0.15	**https://github.com/bbuchfink/diamond/archive/refs/tags/v2.0.15.tar.gz**
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**

OS Requirements

Table 4-514 lists the OS requirements.

Table 4-514 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.104.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DIAMOND software porting.

Table 4-515 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/DIAMOND	Installation path of DIAMOND	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.104.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-516 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.104.5 Obtaining the Source Code

Procedure

Step 1 Download the DIAMOND installation package.

https://github.com/bbuchfink/diamond/archive/refs/tags/v2.0.15.tar.gz

Step 2 Use an SFTP tool to upload the DIAMOND installation package to the */path/to/DIAMOND* directory on the server.

----End

4.104.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/DIAMOND* directory.

```
cd /path/to/DIAMOND
```

Step 3 Decompress the installation package.

```
tar xvf v2.0.15.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd DIAMOND-2.0.15
```

Step 5 Create a **build** directory and switch to the directory.

```
mkdir build
cd build
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/DIAMOND/DIAMOND_install
make
make install
```

Step 7 Load the environment variable.

```
export PATH=/path/to/DIAMOND/DIAMOND_install/bin:$PATH
```

----End

4.104.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/DIAMOND/DIAMOND_install/bin
./diamond --help
```

Information similar to the following is displayed:

```
[root@localhost bin]# ./diamond --help
diamond v2.0.15.153 (C) Max Planck Society for the Advancement of Science
Documentation, support and updates available at http://www.diamondsearch.org
Please cite: http://dx.doi.org/10.1038/s41592-021-01101-x Nature Methods (2021

Syntax: diamond COMMAND [OPTIONS]

Commands:
makedb  Build DIAMOND database from a FASTA file
blastp  Align amino acid query sequences against a protein reference database
blastx  Align DNA query sequences against a protein reference database
view    View DIAMOND alignment archive (DAA) formatted file
help    Produce help message
version Display version information
getseq  Retrieve sequences from a DIAMOND database file
dbinfo  Print information about a DIAMOND database file
test    Run regression tests
makeidx Make database index

General options:
--threads (-p)          number of CPU threads
--db (-d)               database file
--out (-o)              output file
--outfmt (-f)           output format
        0   = BLAST pairwise
        5   = BLAST XML
```

----End

4.105 Exonerate 2.4.0 Porting Guide (Kylin V10)

4.105.1 Introduction

Exonerate is a generic tool for pairwise sequence comparison. It allows you to align sequences using a many alignment models, either exhaustive dynamic programming or a variety of heuristics.

For more information, visit the **Exonerate official website**.

 Copyright © Huawei Technologies Co., Ltd.

Programming language: C

Brief description: a generic tool for pairwise sequence comparison.

Open source license: GPL 3.0

Recommended Software Version

Exonerate 2.4.0

4.105.2 Environmental Requirements

Hardware Requirements

Table 4-517 lists the hardware requirements.

Table 4-517 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-518 lists the software requirements.

Table 4-518 Software requirements

Item	Version	Download URL
Exonerate	2.4.0	https://launchpadlibrarian.net/509766225/exonerate_2.4.0.orig.tar.gz
GCC for openEuler	9.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz

OS Requirements

Table 4-519 lists the OS requirements.

Table 4-519 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.105.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Exonerate software porting.

Table 4-520 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/EXONERATE	Installation path of Exonerate	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.105.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-521 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.105.5 Obtaining the Source Code

Procedure

Step 1 Download the Exonerate installation package.

 Copyright © Huawei Technologies Co., Ltd.

https://launchpadlibrarian.net/509766225/exonerate_2.4.0.orig.tar.gz

Step 2 Use an SFTP tool to upload the Exonerate installation package to the */path/to/EXONERATE* directory on the server.

----End

4.105.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/EXONERATE
```

Step 3 Decompress the installation package.

```
tar -xvf exonerate_2.4.0.orig.tar.gz
```

Step 4 Perform the compilation and installation.

```
./configure --build=aarch-unknown-linux-gnu --prefix=/path/to/EXONERATE
make
```

----End

4.105.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **bin** directory of Exonerate.

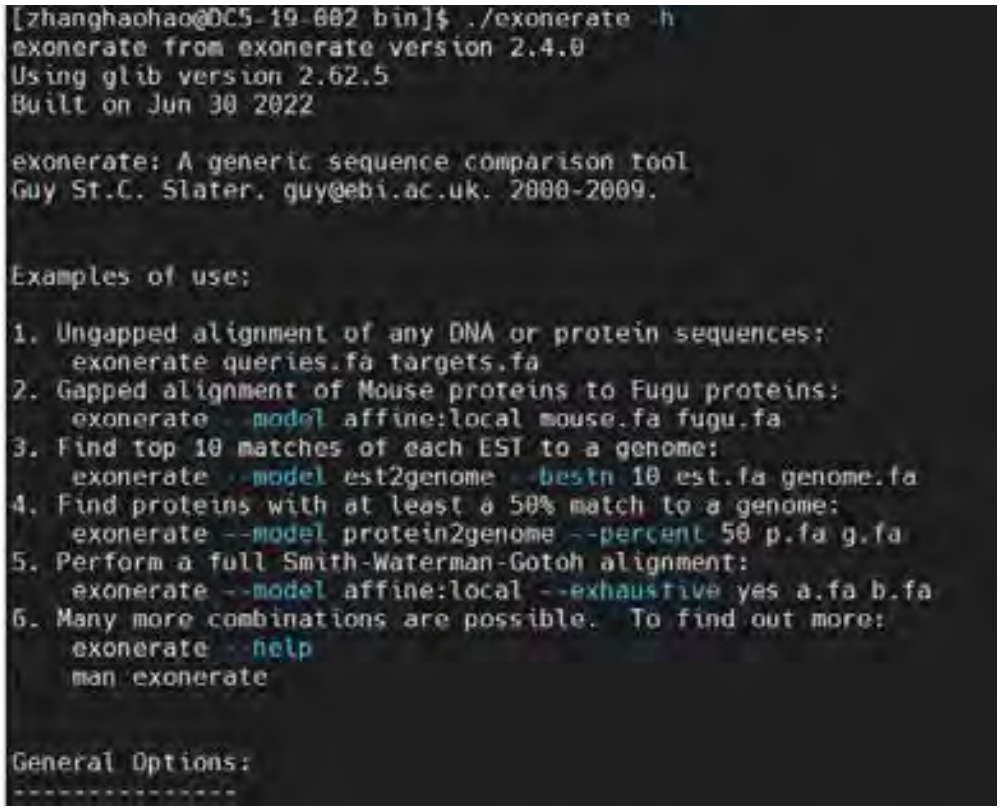
```
cd /path/to/EXONERATE/exonerate/bin
```

Step 3 Perform the verification.

```
./exonerate -h
```

If the following information is displayed, the program is running properly.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.106 FreeBayes 1.3.6 Porting Guide (Kylin V10)

4.106.1 Introduction

FreeBayes is a Bayesian genetic variant detector designed to find small polymorphisms.

For more information, visit the **FreeBayes official website**.

Programming language: C++

Brief description: a Bayesian genetic variant detector.

Open source license: MIT

Recommended Software Version

FreeBayes 1.3.6

4.106.2 Environmental Requirements

Hardware Requirements

Table 4-522 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-522 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-523 lists the software requirements.

Table 4-523 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
Python	3.7	https://www.python.org/
Samtools	1.15.1	https://github.com/samtools/samtools/archive/refs/tags/1.15.1.tar.gz
Parallel	20220622	https://mirror.tuna.tsinghua.edu.cn/gnu/parallel/parallel-latest.tar.bz2
FreeBayes	1.3.6	https://github.com/freebayes/freebayes.git

OS Requirements

Table lists the OS requirements.

Table 4-524 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.106.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FreeBayes software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-525 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/freebayes	Installation path of FreeBayes	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/parallel	Installation path of Parallel	

4.106.4 Configuring the Compilation Environment

4.106.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-526 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Samtools	See "Installing Samtools" in **PASA 2.4.1 Porting Guide (CentOS 7.6)**.
3	Installing Parallel	See **4.106.4.2 Installing Parallel**.

 Copyright © Huawei Technologies Co., Ltd.

4.106.4.2 Installing Parallel

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
pip3 install parallel ninja meson
```

Step 3 Download the Parallel installation package.

```
wget https://mirror.tuna.tsinghua.edu.cn/gnu/parallel/parallel-latest.tar.bz2
```

Step 4 Decompress the Parallel installation package.

```
tar xvf parallel-latest.tar.bz2
```

Step 5 Go to the directory generated after the decompression.

```
cd parallel-latest
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/parallel
```

Step 7 Perform the compilation and installation.

```
make -j16 && make install
```

Step 8 Load the environment variable.

```
export PATH=/path/to/parallel:$PATH
```

----End

4.106.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the FreeBayes installation package.

```
git clone --recursive https://github.com/freebayes/freebayes.git
```

NOTE

Dependencies can be completely downloaded only in **git clone --recursive** mode.

----End

4.106.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Compile vcflib.

```
cd freebayes/vcflib
make
```

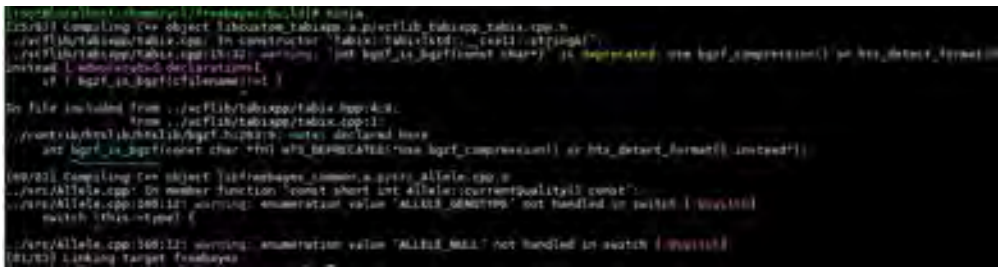
Step 3 Add the environment variable.

```
export PATH=/path/to/freebayes/vcflib/build:$PATH
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Compile and install FreeBayes.

```
cd ../
meson build
cd build && ninja
```



----End

4.106.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the built-in test script.

```
ninja test
```

```
[root@localhost:/home/ycl/freebayes/build]# ninja test
[0/1] Running all tests.
1/5 T03          OK              0.20s
2/5 T00          OK              0.22s
3/5 T02          OK              0.36s
4/5 T01b         OK              4.45s
5/5 T01          OK              9.65s

Ok:                 5
Expected Fail:      0
Fail:               0
Unexpected Pass:    0
Skipped:            0
Timeout:            0

Full log written to /home/ycl/freebayes/build/meson-logs/testlog.txt
```

NOTE

The Parallel, Samtools, and vcflib environment variables must be loaded.

----End

4.107 GMAP 2021-12-17 Porting Guide (Kylin V10)

4.107.1 Introduction

GMAP is a genomic mapping and alignment program for mRNA and EST sequences.

For more information, visit the **GMAP official website**.

Programming language: C

 Copyright © Huawei Technologies Co., Ltd.

Brief description: a genomic mapping and alignment program for mRNA and EST sequences.

Open source license: custom open source license

Recommended Software Version

GMAP 2021-12-17

4.107.2 Environmental Requirements

Hardware Requirements

Table 4-527 lists the hardware requirements.

Table 4-527 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-528 lists the software requirements.

Table 4-528 Software requirements

Item	Version	Download URL
GMAP	2021-12-17	http://research-pub.gene.com/gmap/src/gmap-gsnap-2021-12-17.tar.gz

OS Requirements

Table 4-529 lists the OS requirements.

Table 4-529 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.107.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GMAP software porting.

Table 4-530 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/Gmap	Installation path of GMAP	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.107.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-531 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

 Copyright © Huawei Technologies Co., Ltd.

4.107.5 Obtaining the Source Code

Procedure

Step 1 Download the GMAP installation package.

http://research-pub.gene.com/gmap/src/gmap-gsnap-2021-12-17.tar.gz

Step 2 Use an SFTP tool to upload the GMAP installation package to the */path/to/Gmap* directory on the server.

----End

4.107.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/Gmap* directory.

```
cd /path/to/Gmap
```

Step 3 Decompress the installation package.

```
tar -xvf gmap-gsnap-2021-12-17.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd gmap-2021-12-17
```

Step 5 Create a compilation file.

```
./configure --prefix=/path/to/Gmap/gmap_install
```

Step 6 Perform the compilation.

```
make -j
make install
```

----End

4.107.7 Running and Verification

Procedure

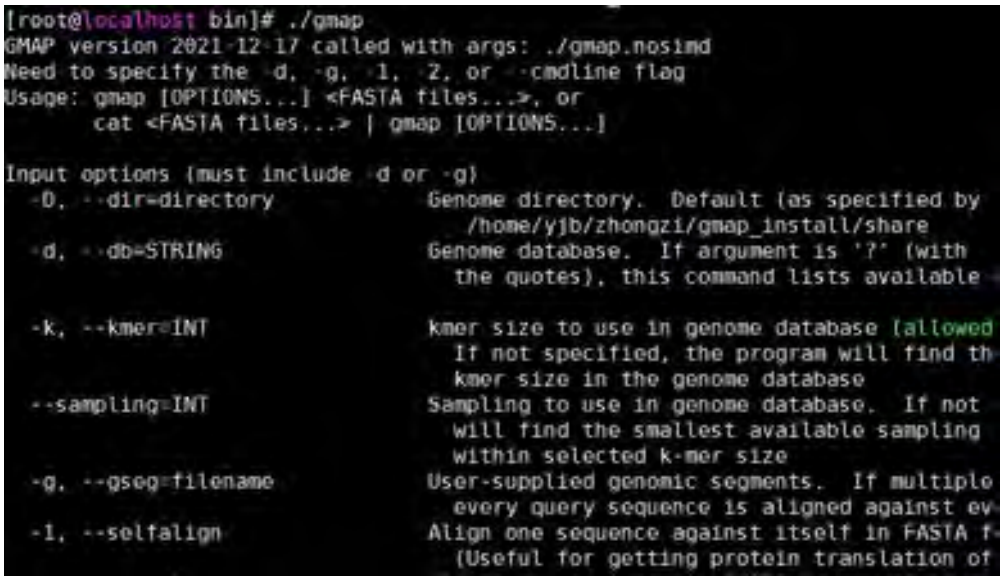
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/Gmap/gmap_install/bin
./gmap
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.108 GUIDEseq 1.26.0 Porting Guide (Kylin V10)

4.108.1 Introduction

GUIDEseq implements GUIDE-seq and PEtag-seq analysis workflow including functions for filtering UMI and reads with low coverage, obtaining unique insertion sites (proxy of cleavage sites), estimating the locations of the insertion sites, aka, peaks, merging estimated insertion sites from plus and minus strand, and performing off target search of the extended regions around insertion sites.

For more information, visit the **GUIDEseq official website**.

Programming language: R

Brief description: GUIDEseq implements GUIDE-seq and PEtag-seq analysis workflow.

Open source license: GPL 3.0

Recommended Software Version

GUIDEseq 1.26.0

4.108.2 Environmental Requirements

Hardware Requirements

Table 4-532 lists the hardware requirements.

Table 4-532 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-533 lists the software requirements.

Table 4-533 Software requirements

Item	Version	Download URL
GUIDEseq	1.26.0	Automatically installed in the R language; no need to download

OS Requirements

Table 4-534 lists the OS requirements.

Table 4-534 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.108.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GUIDEseq software porting.

Table 4-535 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/GUIDESEQ	Installation path of GUIDEseq	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.108.4 Configuring the Compilation Environment

4.108.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-536 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R Language	See **4.108.4.2 Installing the R Language**.

4.108.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the */path/to/GUIDESEQ* directory.

```
cd /path/to/GUIDESEQ
```

Step 3 Download the R language source package.

```
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 4 Decompress the R language source package.

```
tar xvf R-4.2.0.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Go to the directory generated after the decompression.

```
cd R-4.2.0
```

Step 6 Perform the compilation and installation.

```
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/GUIDESEQ/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/GUIDESEQ/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GUIDESEQ/R_install/lib/R/lib:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, run the following command to add an image source of the R language.

1. Create an **~/.Rprofile** file.
   ```
   vi ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.108.5 Obtaining the Source Code

If you use the R language to install GUIDEseq, you do not need to download the software source package.

4.108.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the installation.

```
R
```

Information similar to the following is displayed:

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Enter **install.packages("BiocManager")**.

If the following information is displayed, the installation is successful:

```
> install.packages("BiocManager")
trying URL 'http://mirrors.tuna.tsinghua.edu.cn/CRAN/src/contrib/BiocManager_1.30.16.tar.gz'
Content type 'application/octet-stream' length 289682 bytes (282 KB)
==================================================
downloaded 282 KB

* installing *source* package 'BiocManager' ...
** package 'BiocManager' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package can be loaded from final location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package keeps a record of temporary installation path
* DONE (BiocManager)

The downloaded source packages are in
        '/tmp/RtmpqM5s1k/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Enter **BiocManager::install("GUIDEseq")** to install GUIDEseq.

 Copyright © Huawei Technologies Co., Ltd.

```
* installing *source* package 'GUIDEseq' ...
** using staged installation
** R
** data
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (GUIDEseq)

The downloaded source packages are in
        '/tmp/RtmpfrKar4/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

----End

4.108.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
```

Enter **citation("GUIDEseq")**.

Information similar to the following is displayed:

----End

4.109 HiCUP v0.8.3 Porting Guide (Kylin V10)

4.109.1 Introduction

Hi-C User Pipeline (HiCUP) is a bioinformatics pipeline for processing Hi-C data. The pipeline takes and maps FASTQ data against a reference genome and filters

out frequently encountered experimental artefacts. The pipeline produces paired-read files in SAM/BAM format, each read pair corresponding to a putative Hi-C di-tag. HiCUP also produces summary statistics at each stage of the pipeline providing quality control, helping pinpoint potential problems and refine the experimental protocol. HiCUP comprises five Perl scripts (in addition to the control script) for analysing Hi-C sequence data.

For more information, visit the **HiCUP official website**.

Programming language: Perl

Brief description: a bioinformatics pipeline for processing Hi-C data.

Open source license: LGPL-3.0

Recommended Software Version

HiCUP v0.8.3

4.109.2 Environmental Requirements

Hardware Requirements

Table 4-537 lists the hardware requirements.

Table 4-537 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-538 lists the software requirements.

Table 4-538 Software requirements

Item	Version	Download URL
HiCUP	v0.8.3	**https://github.com/StevenWingett/HiCUP/archive/refs/tags/v0.8.3.tar.gz**

OS Requirements

Table lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-539 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.109.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HiCUP software porting.

Table 4-540 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

4.109.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-541 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.109.5 Obtaining the Source Code

Procedure

Step 1 Download the HiCUP installation package.

https://github.com/StevenWingett/HiCUP/archive/refs/tags/v0.8.3.tar.gz

Step 2 Use an SFTP tool to upload the HiCUP installation package to the server.

----End

4.109.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HiCUP installation package.

```
tar -xvf HiCUP-0.8.3.tar.gz
```

----End

4.109.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
./hicup -h
```

 Copyright © Huawei Technologies Co., Ltd.

----End

4.110 Jellyfish 2.3.0 Porting Guide (Kylin V10)

4.110.1 Introduction

Jellyfish is a tool for fast, memory-efficient counting of k-mers in DNA. A k-mer is a substring of length k, and counting the occurrences of all such substrings is a central step in many analyses of DNA sequence. Jellyfish can count k-mers using an order of magnitude less memory and an order of magnitude faster than other k-mer counting packages by using an efficient encoding of a hash table and by exploiting the "compare-and-swap" CPU instruction to increase parallelism.

For more information, visit the **Jellyfish official website**.

Programming language: C++

Brief description: a tool for fast, memory-efficient counting of k-mers in DNA.

Open source license: GPL 3.0

Recommended Software Version

Jellyfish 2.3.0

4.110.2 Environmental Requirements

Hardware Requirements

Table 4-542 lists the hardware requirements.

Table 4-542 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software requirements

Table 4-543 lists the software requirements.

Table 4-543 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
Jellyfish	2.3.0	https://github.com/gmarcais/Jellyfish/releases/download/v2.3.0/jellyfish-2.3.0.tar.gz

OS Requirements

Table lists the OS requirements.

Table 4-544 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.110.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Jellyfish software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-545 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/jellyfish	Installation path of Jellyfish	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.110.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-546 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.110.5 Obtaining the Source Code

Procedure

Step 1 Download the Jellyfish source package.

https://github.com/gmarcais/Jellyfish/releases/download/v2.3.0/jellyfish-2.3.0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Jellyfish source package to the server.

----End

4.110.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
tar -xvf jellyfish-2.3.0.tar.gz
```

Step 3 Go to the directory generated after the decompression.

```
cd jellyfish-2.3.0
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/jellyfish
make -j 4
make install
```

----End

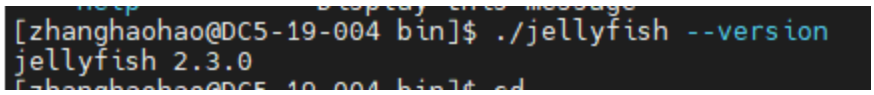
4.110.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/jellyfish/bin
./jellyfish --version
```



----End

4.111 KOBAS 3.0.3 Porting Guide (Kylin V10)

4.111.1 Introduction

KEGG Orthology-Based Annotation System (KOBAS) is a tool for the annotation of sequences by KEGG Orthology terms. KOBAS also identifies enriched pathways and uses KEGG Pathway, PID, BioCyc, Reactome, Panther and human data from OMIM, KEGG Disease, FunDO, GAD, NHGRI, and GWAS databases.

For more information, visit the **KOBAS official website**.

Programming language: Python

Brief description: a tool for the annotation of sequences by KEGG Orthology terms.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

KOBAS 3.0.3

4.111.2 Environmental Requirements

Hardware Requirements

Table 4-547 lists the hardware requirements.

Table 4-547 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table 4-548 lists the software requirements.

Table 4-548 Software requirements

Item	Version	Download URL
Anaconda	3-2021.05	https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh
KOBAS	3.0.3	https://anaconda.org/bioconda/kobas/3.0.3/download/noarch/kobas-3.0.3-py_3.tar.bz2

OS Requirements

Table lists the OS requirements.

Table 4-549 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.111.3 Paths for Software Porting

This chapter lists the software installation paths involved in the KOBAS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-550 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/root/anaconda3	Installation path of Anaconda	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/kobas	Installation path of KOBAS	

4.111.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-551 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Anaconda	See section "Installing Anaconda" in **EMAN2 2.91 Porting Guide (CentOS 8.2)**.

4.111.5 Obtaining the Source Code

Procedure

Step 1 Download the KOBAS installation package.

https://anaconda.org/bioconda/kobas/3.0.3/download/noarch/kobas-3.0.3-py_3.tar.bz2

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the KOBAS installation package to the */path/to/kobas* directory on the server.

----End

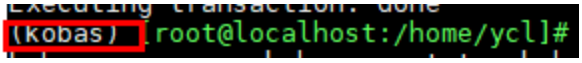
4.111.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a Conda environment and go to the environment.

```
conda create -n kobas && conda activate kobas
```



Step 3 Install the Python environment.

```
conda install -c bioconda -c conda-forge python=2.7.15
```

Step 4 Install the dependencies.

```
conda install -c conda-forge -c bioconda panda
/path/to/pip install blast
```

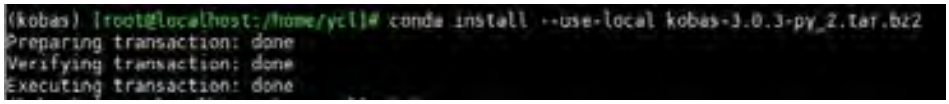
NOTE

/path/to/pip indicates the **pip** directory in the Conda environment. You can run the **which pip** command to view the specific path.

```
conda install -c conda-forge -c bioconda r-base
/path/to/pip install rpy2==2.8.5
/path/to/pip install biopython==1.75 --trusted-host pypi.tuna.tsinghua.edu.cn -i https://pypi.tuna.tsinghua.edu.cn/simple
conda install -c conda-forge -c bioconda bioconductor-qvalue
```

Step 5 Install KOBAS.

```
conda install --use-local kobas-3.0.3-py_2.tar.bz2
```



----End

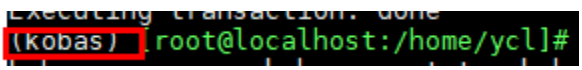
4.111.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Conda environment.

```
conda activate kobas
```



Step 3 Perform the verification.

```
kobas-run
```

Information similar to the following is displayed:

----End

4.112 LTR_Finder Porting Guide (Kylin V10)

4.112.1 Introduction

LTR_Finder is an efficient program for finding full-length LTR retrotranspsons in genome sequences.

The Program first constructs all exact match pairs by a suffix-array based algorithm and extends them to long highly similar pairs. Then Smith-Waterman algorithm is used to adjust the ends of LTR pair candidates to get alignment boundaries. These boundaries are subject to re-adjustment using supporting information of TG..CA box and TSRs and reliable LTRs are selected. Next, LTR_FINDER tries to identify PBS, PPT and RT inside LTR pairs by built-in aligning and counting modules. RT identification includes a dynamic programming to process frame shift. For other protein domains, LTR_FINDER calls ps_scan (from PROSITE, **http://www.expasy.org/prosite/**) to locate cores of important enzymes if they occur. Then possible ORFs are constructed based on that. At last, the program reports possible LTR retrotransposon models in different confidence levels according to how many signals and domains they hit.

For more information, visit the **LTR_Finder official website**.

Programming language: C++/C/Perl

Brief description: LTR_Finder is an efficient program for finding full-length LTR retrotranspsons in genome sequences.

Open source license: MIT

4.112.2 Environmental Requirements

Hardware Requirements

Table 4-552 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-552 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-553 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
LTR_Finder	-	https://github.com/xzhub/LTR_Finder/archive/refs/heads/master.zip

OS Requirements

Table lists the OS requirements.

Table 4-554 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.112.3 Paths for Software Porting

This chapter lists the software installation paths involved in the LTR_Finder software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-555 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/LTR_Finder	Installation path of LTR_Finder	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.112.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-556 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.112.5 Obtaining the Source Code

Procedure

Step 1 Download the LTR_Finder installation package **LTR_Finder-master.zip**.

https://github.com/xzhub/LTR_Finder/archive/refs/heads/master.zip

Step 2 Use an SFTP tool to upload the LTR_Finder installation package to the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.112.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
unzip LTR_Finder-master.zip
```

Step 3 Go to the installation package directory.

```
cd LTR_Finder-master/source
```

Step 4 In the **makefile** file, change **CC = gcc** and **CXX = g++** to **CC = clang** and **CXX = clang++**, respectively.

```
CC = clang
CXX = clang++
```

Step 5 Perform the compilation and installation.

```
make
```

----End

4.113 LTR_retriever 2.9.0 Porting Guide (Kylin V10)

4.113.1 Introduction

LTR_retriever is a command line program (in Perl) for accurate identification of LTR retrotransposons (LTR-RTs) from outputs of LTRharvest, LTR_FINDER, MGEScan 3.0.0, LTR_STRUC, and LtrDetector, and generates non-redundant LTR-RT library for genome annotations.

For more information, visit the **LTR_retriever official website**.

Programming language: Perl

Brief description: a command line program (in Perl) for accurate identification of LTR-RTs.

Open source license: custom open source license

Recommended Software Version

LTR_retriever 2.9.0

4.113.2 Environmental Requirements

Hardware Requirements

Table 4-557 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-557 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-558 Software requirements

Item	Version	Download URL
LTR_retriever	2.9.0	https://github.com/oushujun/LTR_retriever/archive/refs/tags/v2.9.0.tar.gz

OS Requirements

Table 4-559 lists the OS requirements.

Table 4-559 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.113.3 Paths for Software Porting

This chapter lists the software installation paths involved in the LTR_retriever software porting.

Table 4-560 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/LTR_retriever	Installation path of LTR_retriever	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.113.4 Configuring the Compilation Environment

4.113.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool. HMMER has been installed in the */path/to/HMMER* directory.

Configuration Process

Table 4-561 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Perl	See **4.113.4.2 Installing Perl**.
3	Installing RepeatMasker 4.1.2	See **RepeatMasker 4.1.2 Porting Guide (Kylin V10)**.
4	Installing rmblast 2.10.0	See **RMBlast 2.10.0 Porting Guide (Kylin V10)**.
5	Installing GenomeTools	See **4.113.4.3 Installing GenomeTools**.
6	Installing CD-HIT	See **4.113.4.4 Installing CD-HIT**.

 Copyright © Huawei Technologies Co., Ltd.

4.113.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Perl.

```
yum install perl perl-CPAN
```

----End

4.113.4.3 Installing GenomeTools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/LTR_retriever
wget http://genometools.org/pub/genometools-1.6.2.tar.gz
tar xvf genometools-1.6.2.tar.gz
```

Step 3 Go to the installation directory and perform the compilation and installation.

```
cd genometools-1.6.2/
make cairo=0
```

Step 4 Load the environment variable.

```
export PATH=/path/to/LTR_retriever/genometools-1.6.2/bin:$PATH
```

----End

4.113.4.4 Installing CD-HIT

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/LTR_retriever
wget https://github.com/weizhongli/cdhit/archive/refs/tags/V4.8.1.tar.gz
tar xvf V4.8.1.tar.gz
```

Step 3 Go to the installation directory and perform the compilation and installation.

```
cd cdhit-4.8.1
make
```

----End

4.113.5 Obtaining the Source Code

Procedure

Step 1 Download the LTR_retriever installation package **v2.9.0.tar.gz**.

https://github.com/oushujun/LTR_retriever/archive/refs/tags/v2.9.0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the LTR_retriever installation package to the */path/to/LTR_retriever* directory on the server.

----End

4.113.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/LTR_retriever
tar -xvf v2.9.0.tar.gz
cd LTR_retriever-2.9.0
```

Step 3 Configure the dependencies.

```
vim paths
```

Step 4 Add the installation directories of BLAST+, RepeatMasker, HMMER, and CD-HIT.

```
BLAST+=/path/to/rmblast/rmblast_install/bin
RepeatMasker=/path/to/RepeatMasker/RepeatMasker
HMMER=/path/to/HMMER/bin
CDHIT=/path/to/LTR_retriever/cdhit-4.8.1
```

Step 5 Load the environment variable.

```
export PATH=/path/to/LTR_retriever LTR_retriever-2.9.0:$PATH
```

----End

4.113.7 Running and Verification

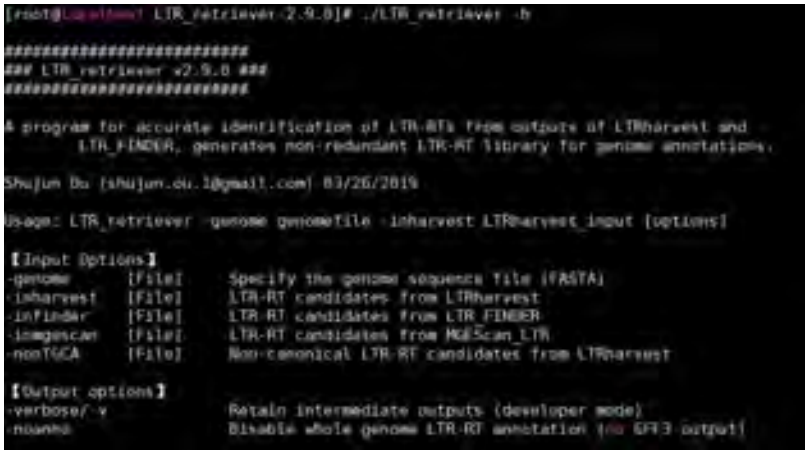
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the installation is successful.

```
cd /path/to/LTR_retriever/LTR_retriever-2.9.0
./LTR_retriever -h
```

Information similar to the following is displayed:

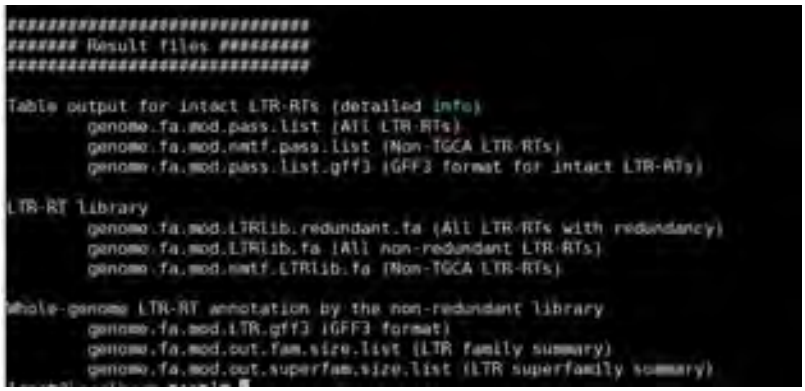


 Copyright © Huawei Technologies Co., Ltd.

Step 3 Start the running.

```
mkdir test && cd test
wget http://ftp.ensembl.org/pub/release-106/fasta/gallus_gallus/dna/
Gallus_gallus.GRCg6a.dna.chromosome.16.fa.gz
gunzip Gallus_gallus.GRCg6a.dna.chromosome.16.fa.gz
mv Gallus_gallus.GRCg6a.dna.chromosome.16.fa Gallus.fa
gt suffixerator -db genome.fa -indexname genome.fa -tis -suf -lcp -des -ssp -sds -dna
gt ltrharvest -index genome.fa -minlenltr 100 -maxlenltr 7000 -mintsd 4 -maxtsd 6 -motif TGCA -
motifmis 1 -similar 85 -vic 10 -seed 20 -seqids yes > genome.fa.harvest.scn
LTR_retriever -genome genome.fa -inharvest genome.fa.harvest.scn
```

Information similar to the following is displayed:



----End

4.114 MAFFT 7.490 Porting Guide (Kylin V10)

4.114.1 Introduction

Multiple Alignment using Fast Fourier Transform (MAFFT) is a high-speed multiple sequence alignment program.

For more information, visit the **MAFFT official website**.

Programming language: C

Brief description: a high-speed multiple sequence alignment program.

Open source license: custom open source license

Recommended Software Version

MAFFT 7.490

4.114.2 Environmental Requirements

Hardware Requirements

Table 4-562 lists the hardware requirements.

Table 4-562 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-563 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
MAFFT	7.490	https://mafft.cbrc.jp/alignment/software/mafft-7.490-with-extensions-src.tgz

OS Requirements

Table lists the OS requirements.

Table 4-564 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.114.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MAFFT software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-565 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/MAFFT	Installation path of MAFFT	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.114.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-566 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.114.5 Obtaining the Source Code

Procedure

Step 1 Download the MAFFT installation package **mafft-7.490-with-extensions-src.tgz**.

https://mafft.cbrc.jp/alignment/software/mafft-7.490-with-extensions-src.tgz

Step 2 Use an SFTP tool to upload the MAFFT installation package to the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.114.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

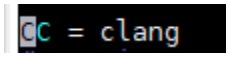
Step 2 Decompress the installation package.

```
tar zxvf mafft-7.490-with-extensions-src.tgz
```

Step 3 Modify the **Makefile** file.

```
cd mafft-7.490-with-extensions/core
```

Step 4 In the **Makefile** file, change **CC = gcc** to **CC = clang**.



Step 5 Perform the compilation and installation.

```
make clean
make
```

----End

4.115 Meryl 1.3 Porting Guide (Kylin V10)

4.115.1 Introduction

Meryl is a utility for working with sequence files

For more information, visit the **Meryl official website**.

Programming language: C

Brief description: Meryl is a utility for working with sequence files.

Open source license: custom open source license

Recommended Software Version

Meryl 1.3

4.115.2 Environmental Requirements

Hardware Requirements

Table 4-567 lists the hardware requirements.

Table 4-567 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 4-568 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz
Meryl	1.3	https://github.com/marbl/meryl/releases/download/v1.3/meryl-1.3.tar.xz

OS Requirements

Table lists the OS requirements.

Table 4-569 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.115.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Meryl software porting.

Table 4-570 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/Meryl	Installation path of Meryl	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.115.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-571 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.115.5 Obtaining the Source Code

Procedure

Download the Meryl installation package.

```
git clone https://github.com/marbl/meryl.git
```

4.115.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

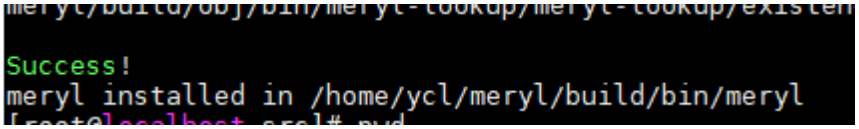
Step 2 Go to the main program installation directory.

```
cd meryl/src
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Perform the compilation and installation.

```
make
```



Step 4 Load the environment.

```
export PATH=/home/ycl/mery/build/bin:$PATH
```

----End

4.116 Minigraph 0.19 Porting Guide (Kylin V10)

4.116.1 Introduction

Minigraph is a sequence-to-graph mapper and graph constructor. For graph generation, it aligns a query sequence against a sequence graph and incrementally augments an existing graph with long query subsequences diverged from the graph.

For more information, visit the **Minigraph official website**.

Programming language: C

Brief description: Minigraph is a sequence-to-graph mapper and graph constructor.

Open source license: MIT

Recommended Software Version

Minigraph 0.19

4.116.2 Environmental Requirements

Hardware Requirements

Table 4-572 lists the hardware requirements.

Table 4-572 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-573 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

OS Requirements

Table lists the OS requirements.

Table 4-574 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.116.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Minigraph software porting.

Table 4-575 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/Minigraph	Installation path of Minigraph	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.116.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-576 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.116.5 Obtaining the Source Code

Procedure

Download the installation package **minigraph-v0.19.tar.gz**.

```
wget https://github.com/lh3/minigraph/archive/refs/tags/v0.19.tar.gz -O minigraph-v0.19.tar.gz
```

4.116.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar zxvf minigraph-v0.19.tar.gz
```

Step 3 In the **Makefile** file, change **CC = gcc** to **CC = clang**.

Step 4 Perform the compilation and installation.

```
make
```

----End

4.116.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Perform the verification.

```
./minigraph test/MT-human.fa test/MT-orangA.fa > out.paf
```

----End

4.117 MISA 25082020 Porting Guide (Kylin V10)

4.117.1 Introduction

MIcroSAtellite Identification Tool (MISA) allows the identification and localization of perfect microsatellites as well as compound microsatellites which are interrupted by a certain number of bases. In order to design primers flanking the SSR loci, two Perl scripts serve as interface modules for the program-to-program data interchange between MISA and the primer modeling software Primer3 (Whitehead Institute).

For more information, visit the **MISA official website**.

Programming language: Perl

Brief description: MISA allows the identification and localization of perfect microsatellites as well as compound microsatellites which are interrupted by a certain number of bases.

Open source license: custom open source license

Recommended Software Version

MISA 25082020

4.117.2 Environmental Requirements

Hardware Requirements

Table 4-577 lists the hardware requirements.

Table 4-577 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 4-578 Software requirements

Item	Version	Download URL
MISA	25082020	https://webblast.ipk-gatersleben.de/misa/misa_sourcecode_25082020.zip

OS Requirements

Table 4-579 lists the OS requirements.

Table 4-579 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.117.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MISA software porting.

Table 4-580 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/MISA	Installation path of MISA	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.117.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-581 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.117.5 Obtaining the Source Code

Procedure

Step 1 Download the MISA installation package **misa_sourcecode_25082020.zip**.

https://webblast.ipk-gatersleben.de/misa/misa_sourcecode_25082020.zip

Step 2 Use an SFTP tool to upload the MISA installation package to the */path/to/MISA* directory on the server.

----End

4.117.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/MISA
```

Step 3 Decompress the installation package.

```
unzip misa_sourcecode_25082020.zip
```

Step 4 Perform the compilation and installation.

```
chmod +x misa.pl
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.117.7 Running and Verification

Procedure

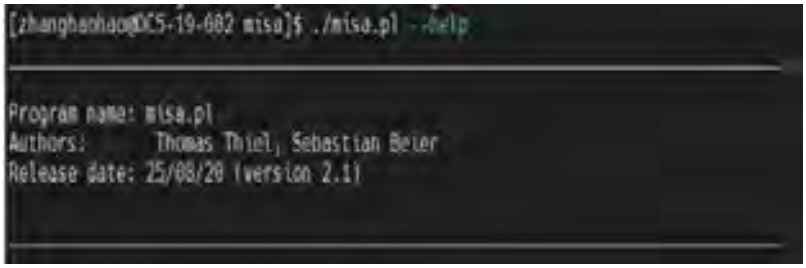
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the MISA directory.

```
cd /path/to/MISA
```

Step 3 Perform the verification.

```
./misa.pl --help
```



The program is running properly.

----End

4.118 MODELTEST 0.1.7 Porting Guide (Kylin V10)

4.118.1 Introduction

ModelTest-NG is a tool for selecting the best-fit model of evolution for DNA and protein alignments. ModelTest-NG supersedes jModelTest and ProtTest in one single tool, with graphical and command console interfaces.

For more information, visit the **MODELTEST official website**.

Programming language: C++

Brief description: ModelTest-NG is a tool for selecting the best-fit model of evolution for DNA and protein alignments.

Open source license: GPL 3.0

Recommended Software Version

MODELTEST 0.1.7

4.118.2 Environmental Requirements

Hardware Requirements

Table 4-582 lists the hardware requirements.

Table 4-582 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-583 Software requirements

Item	Version	Download URL
MODELTEST	0.1.7	https://github.com/ddarriba/modeltest/archive/refs/tags/v0.1.7.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

OS Requirements

Table 4-584 lists the OS requirements.

Table 4-584 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.118.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MODELTEST software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-585 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bisheng-1.3.3	Installation path of the BiSheng Compiler	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/MODELTEST	Installation path of MODELTEST	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.118.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-586 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.118.5 Obtaining the Source Code

Download the MODELTEST installation package.

```
cd /path/to/MODELTEST
git clone https://github.com/ddarriba/modeltest
```

4.118.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the main program installation directory.

```
cd /path/to/MODELTEST/modeltest
```

Step 3 Perform the compilation and installation.

```
mkdir build
cd build/
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/MODELTEST/modeltest/modeltest_install
make
make install
```

----End

4.118.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **modeltest** directory.

```
cd /path/to/MODELTEST/modeltest_install/bin
```

Step 3 Perform the verification.

```
./modeltest-ng -v
```

The program is running properly.

----End

4.119 NINJA 1.2.2 Porting Guide (Kylin V10)

4.119.1 Introduction

NINJA is a fast neighbor-joining method for phylogenetic tree inference. According to benchmark tests, at the time of release, NINJA is the fastest available tool for computing correct neighbor-joining phylogenies for inputs of more than 10,000

 Copyright © Huawei Technologies Co., Ltd.

sequences. It is more than 10x faster than the fastest implementation of the canonical neighbor-joining algorithm (**QuickTree**).

For more information, visit the **NINJA official website**.

Programming language: C++

Brief description: NINJA is a fast neighbor-joining method for phylogenetic tree inference.

Open source license: custom open source license

Recommended Software Version

NINJA 1.2.2

4.119.2 Environmental Requirements

Hardware Requirements

Table 4-587 lists the hardware requirements.

Table 4-587 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-588 lists the software requirements.

Table 4-588 Software requirements

Item	Version	Download URL
NINJA	1.2.2	**https://wheelerlab.org/software/ninja/old_distros/ninja_1.2.2.tgz**

OS Requirements

Table 4-589 lists the OS requirements.

Table 4-589 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.119.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NINJA software porting.

Table 4-590 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/NINJA	Installation path of NINJA	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.119.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-591 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.

4.119.5 Obtaining the Source Code

Procedure

Step 1 Download the NINJA installation package **ninja_1.2.2.tgz**.

 Copyright © Huawei Technologies Co., Ltd.

https://wheelerlab.org/software/ninja/old_distros/ninja_1.2.2.tgz

Step 2 Use an SFTP tool to upload the NINJA installation package to the */path/to/NINJA* directory on the server.

----End

4.119.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/NINJA
```

Step 3 Download the dependency package.

```
yum install -y java*
```

Step 4 Decompress the installation package.

```
tar -zxf ninja_1.2.2.tgz
```

----End

4.119.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the NINJA directory.

```
cd /path/to/NINJA/ninja_1.2.2
```

Step 3 Perform the verification.

```
./ninja -help
```

The program is running properly.

----End

4.120 NOVOPlasty 4.3.1 Porting Guide (Kylin V10)

4.120.1 Introduction

NOVOPlasty is a de novo assembler and heteroplasmy/variance caller for short circular genomes.

For more information, visit the **NOVOPlasty official website**.

Programming language: Perl

Brief description: a de novo assembler and heteroplasmy/variance caller for short circular genomes.

Open source license: custom open source license

Recommended Software Version

NOVOPlasty 4.3.1

4.120.2 Environmental Requirements

Hardware Requirements

Table 4-592 lists the hardware requirements.

Table 4-592 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-593 Software requirements

Item	Version	Download URL
NOVOPlasty	4.3.1	**https://github.com/ndierckx/NOVOPlasty/archive/refs/tags/NOVOPlasty4.3.1.tar.gz**

OS Requirements

Table 4-594 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-594 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.120.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NOVOPlasty software porting.

Table 4-595 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "GCC for openEuler Installation and Operation" in **User Guide (GCC for openEuler)**.
2	/path/to/NOVOPLASTY	Installation path of NOVOPlasty	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.120.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-596 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Setting up the GCC for openEuler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.120.5 Obtaining the Source Code

Procedure

Step 1 Download the NOVOPlasty installation package **NOVOPlasty4.3.1.tar.gz**.

https://github.com/ndierckx/NOVOPlasty/archive/refs/tags/NOVOPlasty4.3.1.tar.gz

Step 2 Use an SFTP tool to upload the NOVOPlasty installation package to the */path/to/NOVOPLASTY* directory on the server.

----End

4.120.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/NOVOPLASTY
```

Step 3 Decompress the installation package.

```
tar -xvf NOVOPlasty4.3.1.tar.gz
```

Step 4 Perform the compilation and installation.

```
chmod +x NOVOPlasty.pl
```

----End

4.120.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **NOVOPlasty** directory.

```
cd /path/to/NOVOPLASTY/NOVOPlasty-NOVOPlasty4.3.1
```

Step 3 Perform the verification.

```
./perl NOVOPlasty4.3.1.pl -c config.txt
```

 Copyright © Huawei Technologies Co., Ltd.

The program is running properly.

----End

4.121 OrthoMCL 2.0.9 Porting Guide (Kylin V10)

4.121.1 Introduction

OrthoMCL is a genome-scale algorithm for grouping orthologous protein sequences. OrthoMCL-DB is a public database that allows users to browse and view ortholog groups that were pre-computed using the OrthoMCL algorithm. Version 4 of this database contained 116,536 ortholog groups clustered from 1,270,853 proteins obtained from 88 eukaryotic genomes, 16 archaeal genomes and 34 bacterial genomes. Future versions of OrthoMCL-DB will include more proteomes as more genomes are sequenced. The OrthoMCL-DB website has a tool for uploading and grouping a set of protein sequences, typically representing a proteome.

For more information, visit the **OrthoMCL official website**.

Programming language: Perl

Brief description: OrthoMCL is a genome-scale algorithm for grouping orthologous protein sequences.

Open source license: GPL 3.0

Recommended Software Version

OrthoMCL 2.0.9

4.121.2 Environmental Requirements

Hardware Requirements

Table 4-597 lists the hardware requirements.

Table 4-597 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-598 Software requirements

Item	Version	Download URL
OrthoMCL	2.0.9	https://github.com/apetkau/orthomclsoftware-custom.git

OS Requirements

Table 4-599 lists the OS requirements.

Table 4-599 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.121.3 Paths for Software Porting

This chapter lists the software installation paths involved in the OrthoMCL software porting.

Table 4-600 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/OrthoMCL	Installation path of OrthoMCL	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.121.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-601 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.121.5 Obtaining the Source Code

Procedure

Download OrthoMCL.

```
git clone https://github.com/apetkau/orthomclsoftware-custom.git
```

4.121.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **OrthoMCL** directory.

```
cd /path/to/ORTHOMCLSOFTWARE/orthomclsoftware-custom/bin
```

Step 3 Perform the verification.

```
./orthomclAdjustFasta --help
```

The program is running properly.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.122 PAL2NAL 14 Porting Guide (Kylin V10)

4.122.1 Introduction

PAL2NAL is a program that converts a multiple sequence alignment of proteins and the corresponding DNA (or mRNA) sequences into a codon alignment. The program automatically assigns the corresponding codon sequence even if the input DNA sequence has mismatches with the input protein sequence, or contains UTRs, polyA tails. It can also deal with frame shifts in the input alignment, which is suitable for the analysis of pseudogenes. The resulting codon alignment can further be subjected to the calculation of synonymous (d_S) and non-synonymous (d_N) substitution rates. If the input is a pair of sequences, PAL2NAL automatically calculates d_S and d_N by the codeml program in PAML.

For more information, visit the **PAL2NAL official website**.

Programming language: Perl

Brief description: PAL2NAL is a program that converts a multiple sequence alignment of proteins and the corresponding DNA (or mRNA) sequences into a codon alignment.

Open source license: GPL 2.0

Recommended Software Version

PAL2NAL 14

4.122.2 Environmental Requirements

Hardware Requirements

Table 4-602 lists the hardware requirements.

Table 4-602 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-603 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-603 Software requirements

Item	Version	Download URL
PAL2NAL	14	http://www.bork.embl.de/pal2nal/distribution/pal2nal.v14.tar.gz

OS Requirements

Table 4-604 lists the OS requirements.

Table 4-604 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.122.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PAL2NAL software porting.

Table 4-605 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/PAL2NAL	Installation path of PAL2NAL	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.122.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-606 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.122.5 Obtaining the Source Code

Procedure

Step 1 Download the PAL2NAL installation package **pal2nal.v14.tar.gz**.

http://www.bork.embl.de/pal2nal/distribution/pal2nal.v14.tar.gz

Step 2 Use an SFTP tool to upload the PAL2NAL installation package to the */path/to/PAL2NAL* directory on the server.

----End

4.122.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/PAL2NAL
```

Step 3 Decompress the installation package.

```
tar xvf pal2nal.v14.tar.gz
```

----End

4.122.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **pal2nal** directory.

```
cd /path/to/PAL2NAL/pal2nal.v14
```

Step 3 Perform the verification.

```
./pal2nal.pl test.aln test.nuc -output paml -nogap > for_paml/test.codon
```

 Copyright © Huawei Technologies Co., Ltd.

The program is running properly.

----End

4.123 PASA r20130907 Porting Guide (Kylin V10)

4.123.1 Introduction

The Program to Assemble Spliced Alignments (PASA) is used to automatically incorporate ESTs and full-length cDNAs into gene structure annotations, in the process annotating UTRs, alternative splicing variations, and polyadenylation sites.

For more information, visit the **PASA official website**.

Programming language: Perl

Brief description: used to automatically incorporate ESTs and full-length cDNAs into gene structure annotations.

Open source license: BSD

Recommended Software Version

PASA r20130907

4.123.2 Environmental Requirements

Hardware Requirements

Table 4-607 lists the hardware requirements.

Table 4-607 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-608 Software requirements

Item	Version	Download URL
PASA	r20130907	**https://sourceforge.net/projects/pasa/files/OLD_VERSIONS/PASA_r20130907.tgz**
GCC for openEuler	9.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-609 lists the OS requirements.

Table 4-609 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.123.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PASA software porting.

Table 4-610 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "GCC for openEuler Installation and Operation" in User Guide (GCC for openEuler).
2	/path/to/PASA	Installation path of PASA	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.123.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-611 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Single-Node System Environment" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

4.123.5 Obtaining the Source Code

Procedure

Step 1 Download the PASA installation package **PASA_r20130907.tgz**.

https://sourceforge.net/projects/pasa/files/OLD_VERSIONS/PASA_r20130907.tgz

Step 2 Use an SFTP tool to upload the PASA installation package to the */path/to/PASA* directory on the server.

----End

4.123.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/PASA
```

Step 3 Decompress the installation package.

```
tar xvf PASA_r20130907.tgz
```

Step 4 Modify the configuration file.

```
cd PASA_r20130907
sed -i "22s/$/ -std="c++98"/g" pasa_cpp/Makefile
```

Step 5 Perform the compilation.

```
make
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.123.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **PASA** directory.

```
cd /path/to/PASA/PASA_r20130907/bin
```

Step 3 Perform the verification.

```
./PASA
```

The program is running properly.

----End

4.124 Picard 2.27.3 Porting Guide (Kylin V10)

4.124.1 Introduction

Picard is implemented using the HTSJDK Java library HTSJDK to support accessing file formats that are commonly used for high-throughput sequencing data such as SAM and VCF.

For more information, visit the **official Picard website**.

Programming language: Java

Brief description: A set of Java command line tools for manipulating high-throughput sequencing (HTS) data and formats.

Open source license: custom open source license

Recommended Software Version

Picard 2.27.3

4.124.2 Environmental Requirements

Hardware Requirements

Table 4-612 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-612 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-613 Software requirements

Item	Version	Download URL
Picard	2.27.3	https://github.com/broadinstitute/picard

OS Requirements

Table 4-614 lists the OS requirements.

Table 4-614 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.124.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Picard software porting.

Table 4-615 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/picard	Installation path of Picard	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.124.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-616 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.124.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/picard
```

Step 3 Obtain the main program source code.

```
git clone https://github.com/broadinstitute/picard.git
```

----End

4.124.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the main program installation directory.

```
cd /path/to/picard/picard
```

Step 3 Install the dependencies.

```
yum install -y java-1.8.0-*
```

Step 4 Perform the compilation.

```
./gradlew shadowJar
```

After the compilation is complete, the message "BUILD SUCCESSFUL" is displayed.

----End

4.124.7 Running and Verification

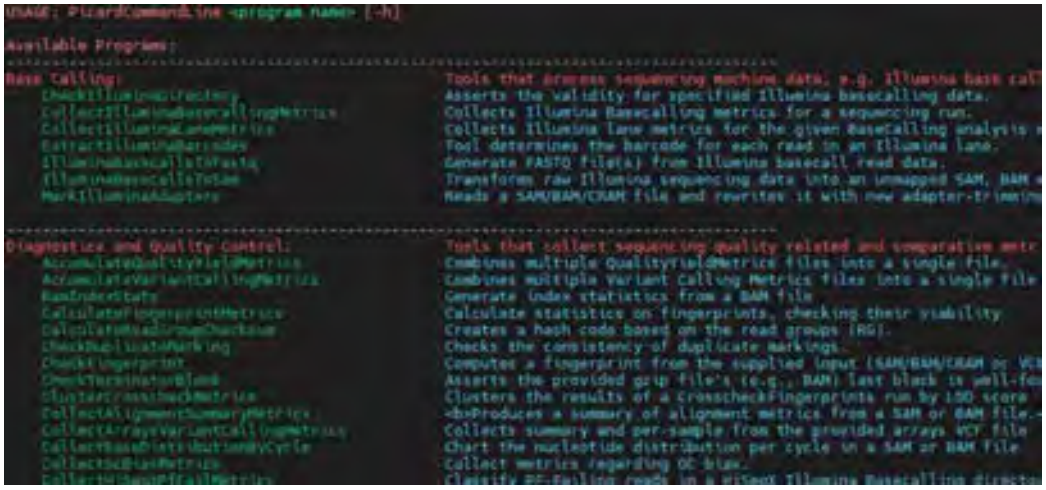
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
java -jar build/libs/picard.jar
```

Information similar to the following is displayed:



----End

4.124.8 Troubleshooting

4.124.8.1 Failed to Download gradle-5.6-bin.zip

Symptom

An error is reported if the network is inaccessible when you run the **/gradlew shadowJar** command in step 4 of **6**.

Key Process and Cause Analysis

Network access fails.

Conclusion and Solution

Step 1 Manually download the file to the current directory.

```
wget https://services.gradle.org/distributions/gradle-5.6-bin.zip
```

Step 2 Modify the file.

```
./gradle/wrapper/gradle-wrapper.properties
```

Step 3 Change the value of **distributionUrl** to **gradle-5.6-bin.zip** in the local directory and run the following command again:

```
./gradlew shadowJar
```

----End

4.125 ProtTest 3.4.2 Porting Guide (Kylin V10)

4.125.1 Introduction

ProtTest is a bioinformatic tool for the selection of best-fit models of aminoacid replacement for the data at hand. ProtTest makes this selection by finding the model in the candidate list with the smallest Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) score or Decision Theory Criterion (DT). At the same time, ProtTest obtains model-averaged estimates of different parameters (including a model-averaged phylogenetic tree) and calculates their importance (Posada and Buckley 2004).

For more information, visit the **ProtTest official website**.

Programming language: Java

Brief description: a bioinformatic tool for the selection of best-fit models of aminoacid replacement for the data at hand.

Open source license: custom open source license

Recommended Software Version

ProtTest 3.4.2

4.125.2 Environmental Requirements

Hardware Requirements

Table 4-617 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-617 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 2-2 lists the software requirements.

Table 4-618 Software requirements

Item	Version	Download URL
ProtTest	3.4.2	**https://github.com/ddarriba/prottest3/archive/refs/tags/3.4.2-release.tar.gz**

OS Requirements

Table 4-619 lists the OS requirements.

Table 4-619 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.125.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ProtTest software porting.

Table 4-620 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/prottest	Installation path of ProtTest	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.125.4 Configuring the Compilation Environment

4.125.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-621 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing PhyML	See **4.125.4.2 Installing PhyML**.

4.125.4.2 Installing PhyML

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the following directory:

```
cd /path/to/prottest
```

Step 3 Download the installation package.

```
git clone https://github.com/stephaneguindon/phyml.git
```

Step 4 Perform the compilation and installation.

```
cd phyml
sh autogen.sh
./configure --prefix=/path/to/prottest/phyml_install --enable-phyml
```

 Copyright © Huawei Technologies Co., Ltd.

```
make
make install
```

----End

4.125.5 Obtaining the Source Code

Procedure

Step 1 Download the ProtTest installation package **3.4.2-release.tar.gz**.

https://github.com/ddarriba/prottest3/archive/refs/tags/3.4.2-release.tar.gz

Step 2 Use an SFTP tool to upload the ProtTest installation package to the */path/to/prottest* directory on the server.

----End

4.125.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y ant java-1.8.0-* autoconf automake
```

Step 3 Go to the installation directory and decompress the installation package.

```
cd /path/to/prottest
tar -xvf 3.4.2-release.tar.gz
cd prottest-3.4.2-release
```

Step 4 Create a build file.

```
ant jar
```

The message "BUILD SUCCESSFUL" is displayed.

----End

4.125.7 Running and Verification

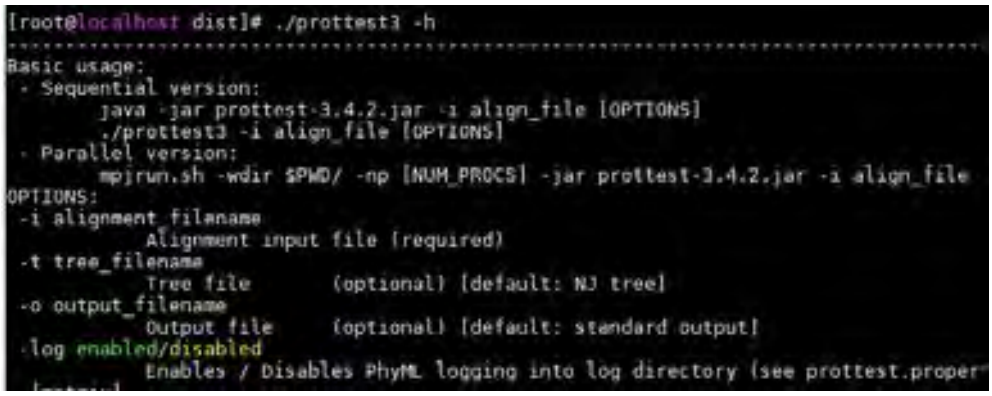
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/prottest/prottest-3.4.2-release/dist
cp /path/to/prottest/phyml_install/bin/phyml ./dist/bin
./prottest3 -h
```

Information similar to the following is displayed:



----End

4.126 Purge_Dups 1.2.6 Porting Guide (Kylin V10)

4.126.1 Introduction

Purge_Dups is designed to remove haplotigs and contig overlaps in a de novo assembly based on read depth.

For more information, visit the **Purge_Dups official website**.

Programing language: C/Python

Brief description: Purge_Dups is designed to remove haplotigs and contig overlaps in a de novo assembly based on read depth.

Open source license: custom open source license

Recommended Software Version

Purge_Dups 1.2.6

4.126.2 Environmental Requirements

Hardware Requirements

Table 4-622 lists the hardware requirements.

Table 4-622 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-623 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-623 Software requirements

Item	Version	Download URL
Purge_Dups	1.2.6	**https://github.com/dfguan/purge_dups/archive/refs/tags/v1.2.6.tar.gz**

OS Requirements

Table 4-624 lists the OS requirements.

Table 4-624 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.126.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Purge_Dups software porting.

Table 4-625 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/purge_dups	Installation path of Purge_Dups	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.126.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-626 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.126.5 Obtaining the Source Code

Procedure

Step 1 Download the Purge_Dups installation package **v1.2.6.tar.gz**.

https://github.com/dfguan/purge_dups/archive/refs/tags/v1.2.6.tar.gz

Step 2 Use an SFTP tool to upload the Purge_Dups installation package to the */path/to/purge_dups* directory on the server.

----End

4.126.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y ant java-1.8.0-* autoconf automake
```

Step 3 Go to the installation directory and decompress the installation package.

```
cd /path/to/purge_dups
tar -xvf v1.2.6.tar.gz
cd purge_dups-1.2.6
```

Step 4 Modify the **makefile** file.

1. Open **makefile**.
   ```
   cd src
   vim makefile
   ```
2. Press **i** to enter the insert mode and change **CC = gcc** in the first line to **CC = clang**.

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the compilation.

```
make
```

----End

4.126.7 Running and Verification

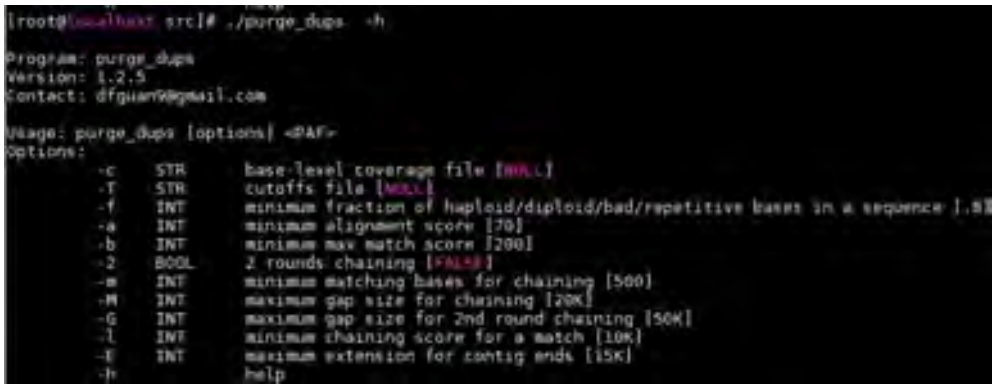
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/purge_dups/purge_dups-1.2.6/src
./purge_dups -h
```

Information similar to the following is displayed:



----End

4.127 Ragout 2.3 Porting Guide (Kylin V10)

4.127.1 Introduction

Reference-Assisted Genome Ordering UTility (Ragout) is a tool for chromosome-level scaffolding using multiple references. Given initial assembly fragments (contigs/scaffolds) and one or multiple related references (complete or draft), it produces a chromosome-scale assembly (as a set of scaffolds).

Programming language: Python/C++

For more information, visit the **Ragout official website**.

Brief description: a tool for chromosome-level scaffolding using multiple references.

Open source license: custom open source license

Recommended Software Version

Ragout 2.3

 Copyright © Huawei Technologies Co., Ltd.

4.127.2 Environmental Requirements

Hardware Requirements

Table 4-627 lists the hardware requirements.

Table 4-627 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-628 lists the software requirements.

Table 4-628 Software requirements

Item	Version	Download URL
Ragout	2.3	https://github.com/fenderglass/Ragout/archive/refs/tags/2.3.tar.gz

OS Requirements

Table 4-629 lists the OS requirements.

Table 4-629 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.127.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Ragout software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-630 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/Ragout	Installation path of Ragout	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.127.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-631 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.127.5 Obtaining the Source Code

Procedure

Step 1 Download the Ragout installation package **2.3.tar.gz**.

https://github.com/fenderglass/Ragout/archive/refs/tags/2.3.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Ragout installation package to the */path/to/Ragout* directory on the server.

----End

4.127.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y python3 python3-pip
pip3 install network==2.2
```

Step 3 Go to the installation directory and decompress the installation package.

```
cd /path/to/Ragou
tar -xvf 2.3.tar.gz
cd Ragout -2.3
```

Step 4 Perform the compilation.

```
python3 setup.py build
python3 setup.py install
```

----End

4.127.7 Running and Verification

Procedure

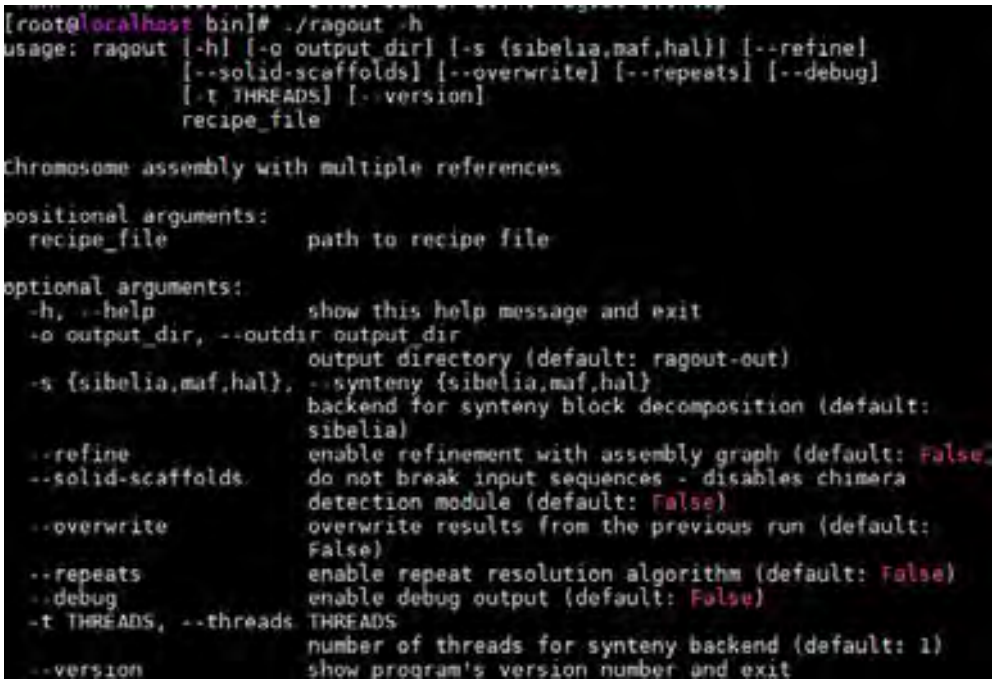
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/Ragout/Ragout-2.3/bin
./ragout -h
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.128 RepeatMasker 4.1.2 Porting Guide (Kylin V10)

4.128.1 Introduction

RepeatMasker is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences.

For more information, visit the **RepeatMasker official website**.

Programming language: Perl

Brief description: RepeatMasker is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences.

Open source license: custom open source license

Recommended Software Version

RepeatMasker 4.1.2

4.128.2 Environmental Requirements

Hardware Requirements

Table 4-632 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-632 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-633 lists the software requirements.

Table 4-633 Software requirements

Item	Version	Download URL
RepeatMasker	4.1.2	http://www.repeatmasker.org/RepeatMasker/RepeatMasker-4.1.2-p1.tar.gz

OS Requirements

Table 4-634 lists the OS requirements.

Table 4-634 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.128.3 Paths for Software Porting

This chapter lists the software installation paths involved in the RepeatMasker software porting.

Table 4-635 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/RepeatMasker	Installation path of RepeatMasker	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.128.4 Configuring the Compilation Environment

4.128.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-636 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Perl	See **4.128.4.2 Installing Perl**.
4	Installing RMBlast	See **RMBlast 2.10.0 Porting Guide (Kylin V10)**.
5	Installing TRF	See **4.128.4.3 Installing TRF**.

4.128.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Perl.

 Copyright © Huawei Technologies Co., Ltd.

```
yum install perl perl-CPAN
```

----End

4.128.4.3 Installing TRF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
git clone https://github.com/Benson-Genomics-Lab/TRF.git
```

Step 3 Go to the installation directory and perform the configuration and installation.

```
cd TRF
mkdir build
cd build
../configure --prefix=/path/to/RepeatMasker/TRF_install
make -j
make install
```

----End

4.128.5 Obtaining the Source Code

Procedure

Step 1 Download the RepeatMasker installation package **RepeatMasker-4.1.2-p1.tar.gz**.

http://www.repeatmasker.org/RepeatMasker/RepeatMasker-4.1.2-p1.tar.gz

Step 2 Use an SFTP tool to upload the RepeatMasker installation package to the */path/to/RepeatMasker* directory on the server.

----End

4.128.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/RepeatMasker
tar -xvf RepeatMasker-4.1.2-p1.tar.gz
cd RepeatMasker
```

Step 3 Install the dependencies.

```
yum install python3 python3-pip
pip3 install h5py
```

Step 4 Install the software.

```
chmod -R 755 *
./configure
```

The installation information is displayed:

```
The full path including the name for the TRF program.
TRF_PRGM:
```

Input */path/to/RepeatMasker/TRF_install/bin/trf* and then press **Enter**.

```
Add a Search Engine:
   1. Crossmatch: [ Un-configured ]
   2. RMBlast: [ Un-configured ]
   3. HMMER3.1 & DFAM: [ Un-configured ]
   4. ABBlast: [ Un-configured ]

   5. Done


Enter Selection: 2
```

Input **2** and press **Enter**.

```
The path to the installation of the RMBLAST sequence alignment program.
RMBLAST_DIR:
```

Input */path/to/rmblast/rmblast_install/bin* and then press **Enter**.

```
Do you want RMBlast to be your default
search engine for Repeatmasker? (Y/N)  [ Y ]: Y
```

Input **Y** and press **Enter**.

```
Add a Search Engine:
   1. Crossmatch: [ Un-configured ]
   2. RMBlast: [ Configured, Default ]
   3. HMMER3.1 & DFAM: [ Un-configured ]
   4. ABBlast: [ Un-configured ]

   5. Done


Enter Selection: 5
```

RMBlast has been configured. Input **5** and press **Enter** to start the installation.

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
Database: Dfam
Version: 3.3
Date: 2020-11-09

Dfam - A database of transposable element (TE) sequence alignments and HMMs.

Total consensus sequences: 6953
Total HMMs: 6915
```

----End

4.128.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/RepeatMasker/RepeatMasker
./RepeatMasker -h
```

Information similar to the following is displayed:

```
[root@localhost RepeatMasker]# ./RepeatMasker -h
RepeatMasker version 4.1.2-p1
Option h is ambiguous (help, hmmer_dir, html)
./RepeatMasker - 4.1.2-p1
NAME
    RepeatMasker - Mask repetitive DNA

SYNOPSIS
      RepeatMasker [-options] <seqfiles(s) in fasta format>

DESCRIPTION
    The options are:
```

----End

4.129 RepeatModeler 2.0.1 Porting Guide (Kylin V10)

4.129.1 Introduction

RepeatModeler is a de novo transposable element (TE) family identification and modeling package. At the heart of RepeatModeler are three de-novo repeat finding programs (RECON, RepeatScout and LtrHarvest/Ltr_retriever) which employ complementary computational methods for identifying repeat element boundaries and family relationships from sequence data.

For more information, visit the **RepeatModeler official website**.

Programming language: Perl

Brief description: a de novo transposable element (TE) family identification and modeling package.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

RepeatModeler 2.0.1

4.129.2 Environmental Requirements

Hardware Requirements

Table 4-637 lists the hardware requirements.

Table 4-637 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-638 lists the software requirements.

Table 4-638 Software requirements

Item	Version	Download URL
RepeatModeler	2.0.1	https://github.com/Dfam-consortium/RepeatModeler/archive/refs/tags/2.0.1.tar.gz

OS Requirements

Table 4-639 lists the OS requirements.

Table 4-639 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.129.3 Paths for Software Porting

This chapter lists the software installation paths involved in the RepeatModeler software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-640 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/RepeatModeler	Installation path of RepeatModeler	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.129.4 Configuring the Compilation Environment

4.129.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-641 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Perl	See **4.129.4.2 Installing Perl**.
5	Installing RepeatMasker	See **RepeatMasker 4.1.2 Porting Guide (Kylin V10)**.
6	Installing RepeatScout	See **RepeatScout 1.0.6 Porting Guide (Kylin V10)**.
7	Installing RECON	See **4.129.4.3 Installing RECON**.

 Copyright © Huawei Technologies Co., Ltd.

4.129.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Perl.

```
yum install perl perl-CPAN
```

----End

4.129.4.3 Installing RECON

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/RepeatMasker
wget http://www.repeatmasker.org/RepeatModeler/RECON-1.08.tar.gz
tar xvf RECON-1.08.tar.gz
```

Step 3 Go to the installation directory and perform the installation.

```
cd RECON-1.08/src
vim Makefile
```

In line 15, change **CC=gcc** to **CC=clang**.

Step 4 Perform the compilation.

```
make
make install
```

----End

4.129.5 Obtaining the Source Code

Procedure

Step 1 Download the RepeatModeler installation package **2.0.1.tar.gz**.

https://github.com/Dfam-consortium/RepeatModeler/archive/refs/tags/2.0.1.tar.gz

Step 2 Use an SFTP tool to upload the RepeatModeler installation package to the */path/to/RepeatModeler* directory on the server.

----End

4.129.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/RepeatModeler
tar -xvf 2.0.1.tar.gz
cd RepeatModeler-2.0.1
```

Step 3 Compile the file.

```
perl ./configure
```

If the system displays a message indicating that a module is missing, run **cpan -i *** to install the module (* indicates the module name). For example:

```
[root@localhost RepeatModeler-2.0.1]# perl ./configure

The following perl modules required by RepeatModeler are missing from
your system.  Please install these first:
    JSON
    LWP::UserAgent
```

```
cpan -i JSON
cpan -i LWP::UserAgent
```

After all modules are installed, run the **perl ./configure** command again.

```
RepeatModeler Configuration Program

This program assists with the configuration of the
RepeatModeler program.  The next set of screens will ask
you to enter information pertaining to your system
configuration.  At the end of the program your RepeatModeler
installation will be ready to use.



<PRESS ENTER TO CONTINUE>
```

Press **Enter**.

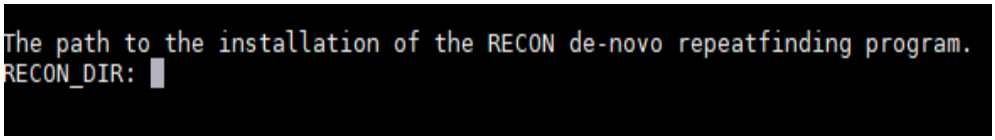
```
**PERL INSTALLATION PATH**

  This is the full path to the Perl interpreter.
  ie. /usr/local/bin/perl









Enter path [ /usr/bin/perl ]: /usr/bin/perl
```

Input **/usr/bin/perl** to use the built-in Perl of the system, and then press **Enter**.

```
The path to the installation of RepeatMasker.
REPEATMASKER_DIR:
```

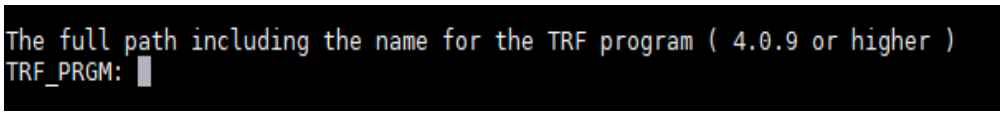
 Copyright © Huawei Technologies Co., Ltd.

Input */path/to/RepeatMasker/RepeatMasker* and press **Enter**.

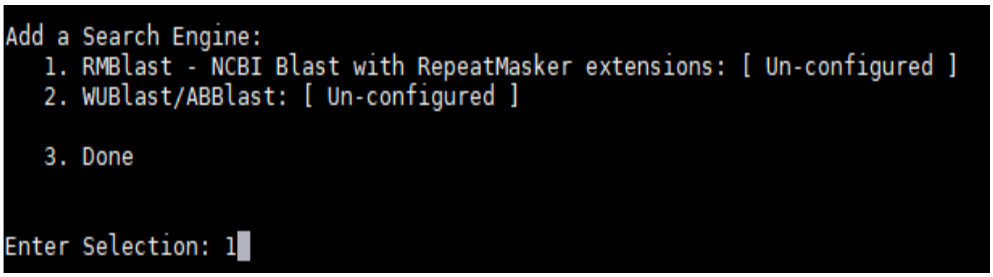


Input */path/to/RepeatMasker/RECON-1.08/src* and press **Enter**.

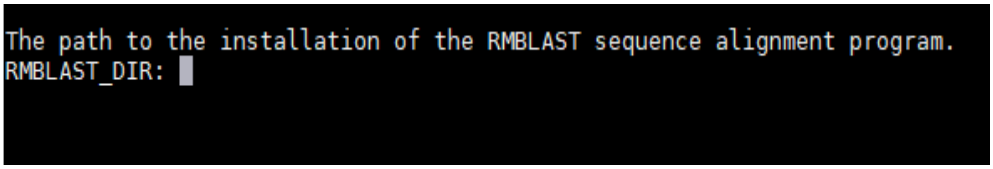
Input */path/to/RepeatScout/RepeatScout_install* and press **Enter**.



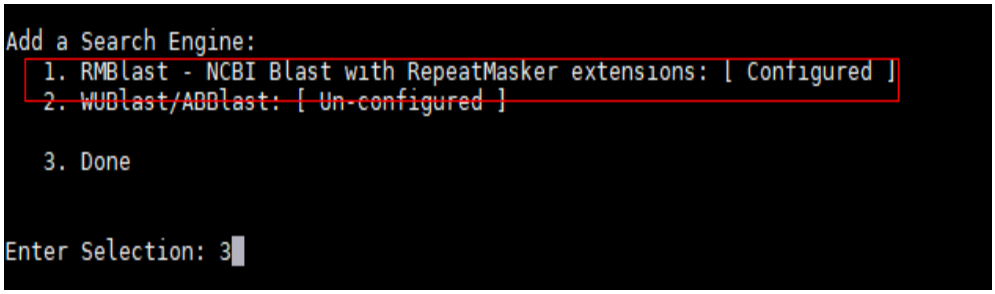
Input */path/to/RepeatMasker/TRF_install/bin/trf* and press **Enter**.



Input **1** and press **Enter** to use RMBlast.



Input */path/to/rmblast/rmblast_install/bin* and press **Enter**.



After RMBlast is configured, input **3** and press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

```
LTR Structural Identication Pipeline [optional]

In addition to RECON/RepeatScout this version of RepeatModeler
has the option of running an additional analysis to identify
structural features of LTR sequences.

Do you wish to configure RepeatModeler for this type
of analysis [y] or n?: n
```

Input **n** and press **Enter**.

The installation is successful, and the message "Congratulations! RepeatModeler is now ready to use" is displayed.

```
LTR Structural Identication Pipeline [optional]

In addition to RECON/RepeatScout this version of RepeatModeler
has the option of running an additional analysis to identify
structural features of LTR sequences.

Do you wish to configure RepeatModeler for this type
of analysis [y] or n?: n
 -- Setting perl interpreter and version...

Congratulations!  RepeatModeler is now ready to use.
```

----End

4.129.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/RepeatModeler/RepeatModeler-2.0.1
./RepeatModeler --help
```

Information similar to the following is displayed:

```
[root@localhost RepeatModeler-2.0.1]# ./RepeatModeler --help
No database indicated

./RepeatModeler - 2.0.1
NAME
    RepeatModeler - Model repetitive DNA

SYNOPSIS
      RepeatModeler [-options] -database <XDF Database>

DESCRIPTION
    The options are:

    -h(elp)
        Detailed help

    -database
        The name of the sequence database to run an analysis on. This is the
        name that was provided to the BuildDatabase script using the "-name"
        option.

    -pa #
        Specify the number of parallel search jobs to run. RMBlast jobs will
        use 4 cores each and ABBlast jobs will use a single core each. i.e.
        on a machine with 12 cores and running with RMBlast you would use
        -pa 3 to fully utilize the machine.

    -recoverDir <Previous Output Directory>
        If a run fails in the middle of processing, it may be possible
        recover some results and continue where the previous run left off.
        Simply supply the output directory where the results of the failed
        run were saved and the program will attempt to recover and continue
        the run.

    -srand #
        Optionally set the seed of the random number generator to a known
        value before the batches are randomly selected ( using Fisher Yates
        Shuffling ). This is only useful if you need to reproduce the sample
        choice between runs. This should be an integer number.

    -LTRStruct
```

----End

4.130 RepeatScout 1.0.6 Porting Guide (Kylin V10)

4.130.1 Introduction

RepeatMasker is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences.

For more information, visit the **RepeatScout official website**.

Programming language: C

Brief description: RepeatMasker is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: custom open source license

Recommended Software Version

RepeatScout 1.0.6

4.130.2 Environmental Requirements

Hardware Requirements

Table 4-642 lists the hardware requirements.

Table 4-642 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-643 lists the software requirements.

Table 4-643 Software requirements

Item	Version	Download URL
RepeatScout	1.0.6	http://www.repeatmasker.org/RepeatScout-1.0.6.tar.gz

OS Requirements

Table 4-644 lists the OS requirements.

Table 4-644 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.130.3 Paths for Software Porting

This chapter lists the software installation paths involved in the RepeatScout software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-645 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/ RepeatScout	Installation path of RepeatScout	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.130.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-646 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.130.5 Obtaining the Source Code

Procedure

Step 1 Download the RepeatScout installation package **RepeatScout-1.0.6.tar.gz**.

http://www.repeatmasker.org/RepeatScout-1.0.6.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the RepeatScout installation package to the */path/to/RepeatScout* directory on the server.

----End

4.130.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/RepeatScout
tar -xvf RepeatScout-1.0.6.tar.gz
cd RepeatScout-1.0.6.
```

Step 3 Modify the **Makefile** file.

1. Open **Makefile**.
   ```
   vim Makefile
   ```
2. Press **i** to enter the insert mode and modify line 17.

   Before the modification:

   ```
   # Installation Directory
   INSTDIR = /usr/local/RepeatScout-$(VERSION)
   ```

   After the modification:

   ```
   # Installation Directory
   INSTDIR=/path/to/RepeatScout/RepeatScout_install
   ```

Step 4 Perform the compilation.

```
make -j
make install
```

----End

4.130.7 Running and Verification

Procedure

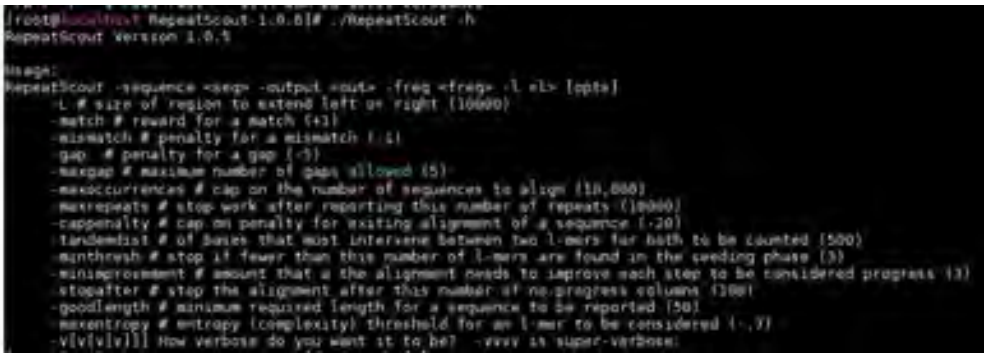
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/RepeatScout/RepeatScout_install
./RepeatScout -h
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.131 RMBlast 2.10.0 Porting Guide (Kylin V10)

4.131.1 Introduction

RMBlast is a RepeatMasker compatible version of the standard NCBI blastn program. The primary difference between this distribution and the NCBI distribution is the addition of a new program "rmblastn" for use with RepeatMasker and RepeatModeler.

For more information, visit the **RMBlast official website**.

Programming language: C/C++

Brief description: RMBlast is a RepeatMasker compatible version of the standard NCBI blastn program.

Open source license: custom open source license

Recommended Software Version

RMBlast 2.10.0

4.131.2 Environmental Requirements

Hardware Requirements

Table 4-647 lists the hardware requirements.

Table 4-647 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-648 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-648 Software requirements

Item	Version	Download URL
blast+	2.10.0	**https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.0/ncbi-blast-2.10.0+-src.tar.gz**
isb-2.10.0+-rmblast	2.10.0	**https://www.repeatmasker.org/isb-2.10.0+-rmblast.patch.gz**

OS Requirements

Table 4-649 lists the OS requirements.

Table 4-649 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.131.3 Paths for Software Porting

This chapter lists the software installation paths involved in the RMBlast software porting.

Table 4-650 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/rmblast	Installation path of RMBlast	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.131.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-651 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.131.5 Obtaining the Source Code

Procedure

Step 1 Download the blast+ and isb-2.10.0+-rmblas installation packages.

https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.0/ncbi-blast-2.10.0+-src.tar.gz

https://www.repeatmasker.org/isb-2.10.0+-rmblast.patch.gz

Step 2 Use an SFTP tool to upload the two installation packages to the */path/to/rmblast* directory on the server.

----End

4.131.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y patch
```

Step 3 Go to the installation directory and decompress the installation package.

```
cd /path/to/rmblast
tar -xvf ncbi-blast-2.10.0+-src.tar.gz
gunzip isb-2.10.0+-rmblast.patch.gz
cd ncbi-blast-2.10.0+-src
patch -p1 < ../isb-2.10.0+-rmblast.patch
cd c++
```

Step 4 Modify the configuration file.

 Copyright © Huawei Technologies Co., Ltd.

```
vim src/build-system/configure
```

Comment out lines 8236 to 8238 and add lines 8239 to 8243. The modification is as follows:

Step 5 Create a compilation file.

```
./configure --with-mt --without-debug --without-krb5 --without-openssl --with-projects=scripts/
projects/rmblastn/project.lst --prefix=/path/to/rmblast/rmblast_install
```

Step 6 Perform the compilation.

```
make -j64
make install
```

----End

4.131.7 Running and Verification

Procedure

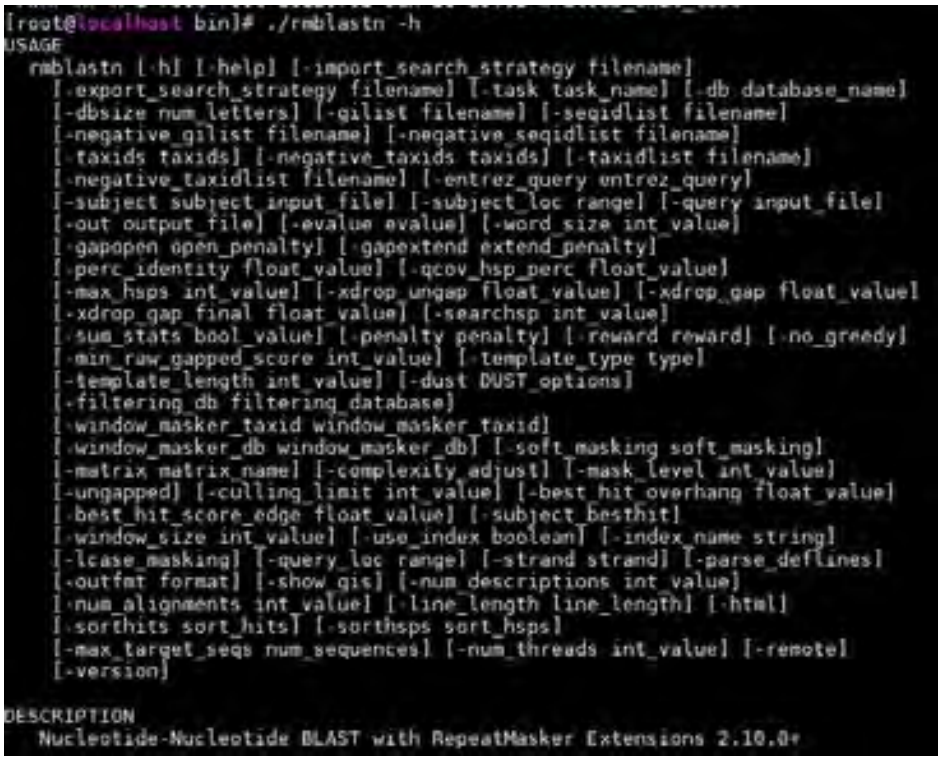
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/rmblast/rmblast_install/bin
./rmblastn -h
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.132 RSEM 1.3.3 Porting Guide (Kylin V10)

4.132.1 Introduction

RSEM is a software package for estimating gene and isoform expression levels from RNA-Seq data. The RSEM package provides a user-friendly interface, supports threads for parallel computation of the EM algorithm, single-end and paired-end read data, quality scores, variable-length reads and RSPD estimation. In addition, it provides posterior mean and 95% credibility interval estimates for expression levels. For visualization, It can generate BAM and Wiggle files in both transcript-coordinate and genomic-coordinate. Genomic-coordinate files can be visualized by both UCSC Genome browser and Broad Institute's Integrative Genomics Viewer (IGV). Transcript-coordinate files can be visualized by IGV. RSEM also has its own scripts to generate transcript read depth plots in PDF format. The unique feature of RSEM is, the read depth plots can be stacked, with read depth contributed to unique reads shown in black and contributed to multi-reads shown in red. In addition, models learned from data can also be visualized. Last but not least, RSEM contains a simulator.

For more information, visit the **RSEM official website**.

Programming language: C/C++

Brief description: a software package for estimating gene and isoform expression levels from RNA-Seq data.

Open source license: custom open source license

Recommended Software Version

RSEM 1.3.3

4.132.2 Environmental Requirements

Hardware Requirements

Table 4-652 lists the hardware requirements.

Table 4-652 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-653 lists the software requirements.

Table 4-653 Software requirements

Item	Version	Download URL
RSEM	1.3.3	https://github.com/deweylab/RSEM/archive/refs/tags/v1.3.3.tar.gz

OS Requirements

Table 4-654 lists the OS requirements.

Table 4-654 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.132.3 Paths for Software Porting

This chapter lists the software installation paths involved in the RSEM software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-655 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/RSEM	Installation path of RSEM	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.132.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-656 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.132.5 Obtaining the Source Code

Procedure

Step 1 Download the RSEM installation package **v1.3.3.tar.gz**.

https://github.com/deweylab/RSEM/archive/refs/tags/v1.3.3.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the RSEM installation package to the */path/to/RSEM* directory on the server.

----End

4.132.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
cd /path/to/RSEM
tar xvf v1.3.3.tar.gz
cd RESM-1.3.3
```

Step 3 Modify the configuration file.

```
vim Makefile
```

In line 14, change **CXX = g++** to **CXX = clang++**.

In line 28, change **prefix ?= /usr/local** to **prefix ?= /path/to/RSEM/RSEM_install**.

Step 4 Perform the compilation.

```
make
make install
```

----End

4.132.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/RSEM/RSEM_install/bin
./rsem-prepare-reference --help
```

Information similar to the following is displayed:

----End

4.133 CANU 1.8 Porting Guide (CentOS 7.6)

4.133.1 Introduction

CANU is a third-generation data assembly tool designed by Celera Assembler. It supports PacBio RSII and Oxford Nanopore MinION and is used for high-noise single-molecule sequencing. The main functions include correction, trimming, and assembly.

For more information about CANU, visit the **official CANU website**.

Programming language: C/Perl

Brief description: high-noise single-molecule sequencing tool

Open source license: GPL 2.0

Recommended Software Version

CANU V1.8

4.133.2 Environment Requirements

Hardware Requirements

Table 4-657 lists the hardware requirements.

Table 4-657 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-658 lists the OS requirements.

Table 4-658 OS requirements

Item	Version	Download Address
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image

4.133.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CANU software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-659 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/CANU	Installation path of CANU	The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.

4.133.4 Configuring the Compilation Environment

Step 1 Install the GNU and configure environment variables.

For details, see **GNU 9.1 Installation Guide**.

Step 2 Use PuTTY to log in to a server as the **root** user.

Step 3 Run the following commands to load the environment variables of GNU 9.1:

```
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following command to install Boost:

```
yum install boost* -y
```

NOTE

The Yum source is a local source of the CentOS 7.6 image. You can modify the files in the **/etc/yum.repos.d** directory as required.

----End

4.133.5 Obtaining the Source Code

Step 1 Download the CANU source code package **canu-1.8.tar.gz**.

https://github.com/marbl/canu/archive/refs/tags/v1.8.tar.gz

Step 2 Use an SFTP tool to upload the CANU source code package to the */path/to/CANU* directory on the server.

----End

4.133.6 Compiling and Installing CANU

Step 1 Use PuTTY to log in to a server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to decompress the CANU installation package:

```
tar -xvf canu-1.8.tar.gz
```

Step 3 Run the following command to switch to CANU source code directory:

```
cd canu-1.8/src
```

Step 4 Run the following command to compile CANU:

```
make -j8
```

Step 5 Go to the CANU 1.8 directory. If the CANU executable file is generated in the **Linux-aarch64/bin** directory, as shown in the following command output, the compilation is complete.

```
-rwxrwxr-x 1 root root 31114 Oct 23  2018 canu
```

Step 6 Run the following command to add CANU to the environment variables:

```
export PATH=/path/to/CANU/canu-1.8/Linux-aarch64/bin:$PATH
```

----End

4.133.7 Running and Verifying CANU

Step 1 Use PuTTY to log in to a server as the **root** user.

Step 2 Run the following command to install Java 1.8.0:

```
yum install java-1.8.0-openjdk* gnuplot -y
```

NOTE

The Yum source is a local source of the CentOS 7.6 image. You can modify the files in the **/etc/yum.repos.d** directory as required.

Step 3 Run the following command to obtain the case file:

```
curl -L -o oxford.fastq http://gembox.cbcb.umd.edu/mhap/raw/ecoli_p6_25x.filtered.fastq
```

NOTE

Access to the external network is required. You can download and convert the case file by using a server that is connected to the external network, and then copy the case file to the test directory.

Step 4 Run the following command to start CANU:

```
{ time canu -p ecoli -d ecoli-pacbio genomeSize=4.8m corPartitions=384 corMemory=3
corPartitionMin=1000 corThreads=1 useGrid=false obtovlThreads=96
obtOvlHashBlockLength=682496000 utgOvlHashBlockLength=682496000 utgovlThreads=96
gridEngineMemoryOption="-l vf=MEMORY" -nanopore-raw oxford.fastq ; } 2>&1 |tee -a canu.log
```

Table 4-660 describes the command parameters.

Table 4-660 CANU command parameters

Parameter	Description
useGrid	Whether to run under Grid Control. The default value is **true**. Change the value to **false**.
-p	assembly-prefix component index. Set this parameter based on the case.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
-d	assembly-directory component directory. The value of this parameter is user-defined.
genmoeSize	Genome size. Set this parameter based on the case.
-nanopore-raw	Select the required gene file format based on the case.
corPartitions	Number of tasks. You are advised to set this parameter to a multiple of the number of cores. The recommended value is 4 times the number of cores.
corThreads	Number of threads in each process. The recommended value is **1**.
obtovlThreads	Number of threads for running obtovl. You are advised to set this parameter to the number of cores.
utgovlThreads	Number of threads for running utgov. You are advised to set this parameter to the number of cores.

----End

4.134 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6)

4.134.1 Introduction

Ont-Tombo is a suite of tools primarily for the identification of modified nucleotides from nanopore sequencing data. Ont-Tombo also provides tools for the analysis and visualization of raw nanopore signal.

Ont-Tombo provides the following functions:

- Improved gauze detection
- Visualization of the original reference anchoring signal
- Python API for original model analysis
- User-friendly model estimation methods with tutorials

For more information about Ont-Tombo, visit the **official Hive website**.

Programming language: Python

Brief description: the tool for the identification of modified nucleotides from nanopore sequencing data

Open source license: Mozilla Public License Version 2.0

Recommended Software Version

Ont-tombo 1.5.1

 Copyright © Huawei Technologies Co., Ltd.

4.134.2 Environment Requirements

Hardware Requirements

Table 4-661 lists the hardware requirements.

Table 4-661 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 4-662 lists the software requirements.

Table 4-662 Software Requirements

Item	Version	Download Address
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
OpenBLAS	0.3.7	https://codeload.github.com/xianyi/OpenBLAS/tar.gz/refs/tags/v0.3.7
Python	2.7.16	https://www.python.org/ftp/python/2.7.16/Python-2.7.16.tgz

OS Requirements

Table 4-663 lists the OS requirements.

Table 4-663 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/

4.134.3 Planning the Paths for Software Porting

This chapter lists the software installation paths involved in the Ont-tombo software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-664 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
3	/path/to/OPENBLAS	Installation path of OpenBLAS	
4	/path/to/PYTHON	Installation path of Python	

4.134.4 Configuring the Compilation Environment

4.134.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing dependencies	See **4.134.4.2 Installing Dependencies**.
3	Installing Python	See **4.134.4.3 Installing Python**.
4	Installing PIP	See **4.134.4.4 Installing PIP**.
5	Installing OpenBLAS	See **4.134.4.5 Installing OpenBLAS**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
6	Installing HDF5	See **4.134.4.6 Installing HDF5**.

4.134.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel
```

----End

4.134.4.3 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the Python installation package:

```
tar zxvf Python-2.7.16.tgz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd Python-2.7.16
```

Step 4 Run the following command to perform configuration:

```
CXX=g++ ./configure --prefix=/path/to/PYTHON
```

Step 5 Run the following command to perform compilation:

```
make
make install
```

Step 6 Run the following commands to set the Python environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 7 Run the following command to check whether the installation is successful:

```
python -V
```

If information similar to the following is displayed, the installation is successful:

```
Python 2.7.16
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.134.4.4 Installing PIP

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to download the distribution file of PIP:

```
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py -o get-pip.py
```

Step 3 Run the following command to install PIP.

```
python get-pip.py
```

Step 4 Run the following command to check whether the installation is successful:

```
which pip
```

If information similar to the following is displayed, the installation is successful:

```
/path/to/PYTHON/bin/pip
```

----End

4.134.4.5 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the OpenBLAS installation package:

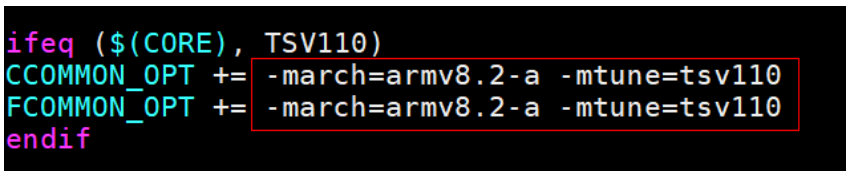
```
tar -xvf OpenBLAS-0.3.7.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.7
```

Step 4 (Optional) If the GCC compiler version is not GNU 9 or later, modify the **Makefile.arm64** file and run the **gcc -v** command to query the GCC compiler version.

1. Open **Makefile.arm64**.
   ```
   vi Makefile.arm64
   ```
2. Press **i** to enter the insert mode and delete the content in the red box shown in the following figure:

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

Step 5 Run the following commands to install OpenBLAS:

```
make BINARY=64 FC=/usr/bin/gfortran USE_THREAD=1
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following command to set the OpenBLAS environment variable:

```
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.134.4.6 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
FC=/usr/bin/gfortran ./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-shared
```

Step 5 Run the following commands to compile and install HDF5:

```
make -j 16
make install
```

Step 6 Run the following commands to set the HDF5 environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

4.134.5 Obtaining the Source Code

Procedure

The Ont-Tombo installation package is downloaded and installed through the link specified by the pip command. You do not need to download the source code package separately.

https://pypi.tuna.tsinghua.edu.cn/simple

4.134.6 Compiling and Installing Ont-Tombo

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install Numpy:

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple numpy
```

Step 3 Run the following command to install Scipy:

```
LAPACK=/path/to/OPENBLAS/lib/libopenblas.so BLAS=/path/to/OPENBLAS/lib/libopenblas.so pip install -i https://pypi.tuna.tsinghua.edu.cn/simple scipy
```

Step 4 Run the following command to install Ont-Tombo:

```
CFLAGS="-I/path/to/HDF5/include -L/path/to/HDF5/lib" pip install -i https://pypi.tuna.tsinghua.edu.cn/simple ont-tombo
```

If information similar to the following is displayed, the installation is successful:

```
Successfully installed cython-0.29.15 future-0.18.2 h5py-2.10.0 mappy-2.17 ont-tombo-1.5.1 tqdm-4.43.0
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.135 CPAT 3.0.4 Porting Guide (Kylin V10)

4.135.1 Introduction

CPAT is a bioinformatics tool to predict RNA's coding probability based on the RNA sequence characteristics. To achieve this goal, CPAT calculates scores of these 4 linguistic features from a set of known protein-coding genes and another set of non-coding genes.

- ORF size
- ORF coverage
- Fickett TESTCODE
- Hexamer usage bias

CPAT will then build a logistic regression model using these 4 features as predictor variables and the "protein-coding status" as the response variable. After evaluating the performance and determining the probability cutoff, the model can be used to predict new RNA sequences.

For more information, visit the **CPAT official website**.

Programming language: Python

Brief description: CPAT is a bioinformatics tool to predict RNA's coding probability based on the RNA sequence characteristics.

Open source license: GNU General Public License

Recommended Software Version

CPAT 3.0.4

4.135.2 Environment Requirements

Hardware Requirements

Table 4-665 lists the hardware requirements.

Table 4-665 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-666 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz**
Python	3.7.5	**https://www.python.org/downloads/**

OS Requirements

Table 4-667 lists the OS requirements.

Table 4-667 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.135.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CPAT software porting.

Table 4-668 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

4.135.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-669 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.135.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install CPAT.

```
pip3 install cpat
```

----End

4.135.6 Running and Verification

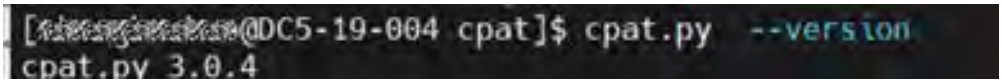
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify the version.

```
cpat.py --version
```

If the correct version is displayed, the installation is successful.



----End

4.136 MrBayes 3.2.7a Porting Guide (Kylin V10)

4.136.1 Introduction

MrBayes is a program for Bayesian inference and model choice across a wide range of phylogenetic and evolutionary models. MrBayes uses Markov chain Monte Carlo (MCMC) methods to estimate the posterior distribution of model parameters.

 Copyright © Huawei Technologies Co., Ltd.

For more information, visit the **MrBayes web page**.

Programming language: C

Brief description: A program for Bayesian inference and model choice.

Open source license: GPL 3.0

Recommended Software Version

MrBayes 3.2.7a

4.136.2 Environment Requirements

Hardware Requirements

Table 4-670 lists the hardware requirements.

Table 4-670 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-671 Software requirements

Item	Version	Download URL
MrBayes	3.2.7a	**https://github.com/NBISweden/MrBayes.git**
GCC for openEuler	9.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz**

OS Requirements

Table 4-672 lists the OS requirements.

Table 4-672 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.136.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MrBayes software porting.

Table 4-673 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	/path/to/MRBAYES	Installation path of MrBayes	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.136.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-674 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.

 Copyright © Huawei Technologies Co., Ltd.

4.136.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the MrBayes installation package.

```
cd /path/to/MRBAYES
git clone --depth=1 https://github.com/NBISweden/MrBayes.git
```

----End

4.136.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source code directory.

```
cd /path/to/MRBAYES/MrBayes
```

Step 3 Perform the compilation.

```
./configure --prefix=/path/to/MRBAYES/mybayes
make
make install
```

----End

4.136.7 Running and Verification

Procedure

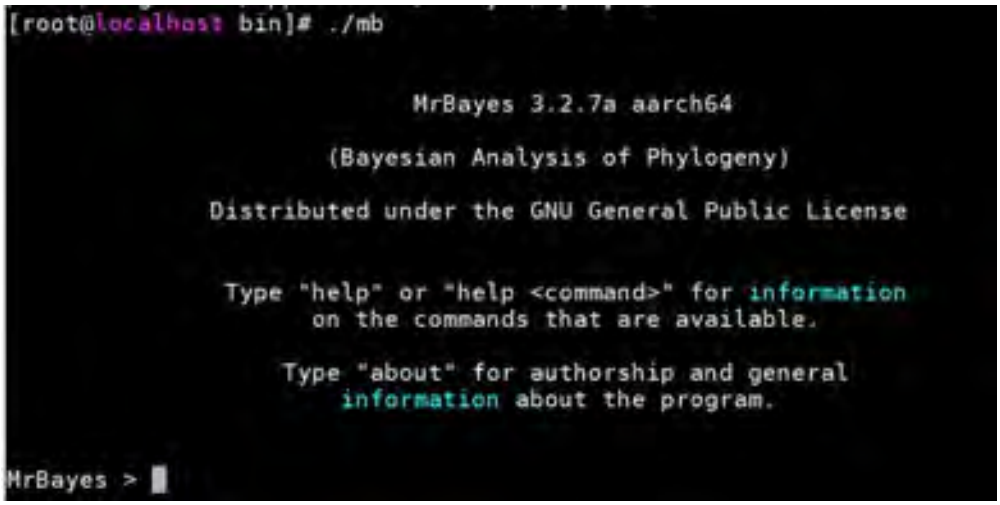
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the MrBayes directory.

```
cd /path/to/MRBAYES/mybayes/bin
```

Step 3 Start the running.

```
./mb
```



 Copyright © Huawei Technologies Co., Ltd.

The program is running properly.

----End

4.137 MSMC 1.1.0 Porting Guide (Kylin V10)

4.137.1 Introduction

This software implements MSMC, a method to infer population size and gene flow from multiple genome sequences.

In short, MSMC can infer:

- the scaled population size of a single population as a function of time
- the timing and nature of population separations between two populations

from multiple phased haplotypes. When only two haplotypes are given, MSMC is similar to PSMC, and we call it PSMC' because of subtle differences in the method and the underlying model, which allows PSMC' to infer more accurately the recombination rate.

For more information, visit the **MSMC page at GitHub**.

Programming language: D

Brief description: A method to infer population size and gene flow from multiple genome sequences.

Open source license: GPL 3.0

Recommended Software Version

MSMC 1.1.0

4.137.2 Environment Requirements

Hardware Requirements

Table 4-675 lists the hardware requirements.

Table 4-675 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-676 Software requirements

Item	Version	Download URL
MSMC	1.1.0	**https://github.com/stschiff/msmc/archive/refs/tags/v1.1.0.tar.gz**
GCC for openEuler	9.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz**

OS Requirements

Table 4-677 lists the OS requirements.

Table 4-677 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.137.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MSMC software porting.

Table 4-678 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	/path/to/MSMC	Installation path of MSMC	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.137.4 Configuring the Compilation Environment

4.137.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-679 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.
3	Installing GSL	See **4.137.4.2 Installing GSL**.
4	Installing the D compiler	See **4.137.4.3 Installing the D Compiler**.

4.137.4.2 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/MSMC
wget http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar -xvf gsl-2.6.tar.gz
cd gsl-2.6
```

Step 4 Perform the compilation.

```
./configure --prefix=/path/to/MSMC/gsl
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.137.4.3 Installing the D Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the D compiler.

```
curl -fsS https://dlang.org/install.sh | bash -s ldc
```

----End

4.137.5 Obtaining the Source Code

Procedure

Step 1 Download the MSMC installation package **v1.1.0.tar.gz**.

https://github.com/stschiff/msmc/archive/refs/tags/v1.1.0.tar.gz

Step 2 Use an SFTP tool to upload the MSMC installation package to the */path/to/MSMC* directory on the server.

----End

4.137.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/MSMC
```

Step 3 Load the D compiler.

```
source ~/dlang/ldc-1.30.0/activate
```

Step 4 Decompress the installation package.

```
tar xvf v1.1.0.tar.gz
```

Step 5 Modify the configuration file.

```
cd msmc-1.1.0
sed -i "s/dmd/ldmd2/g" Makefile
sed -i "2s/usr\/lib/path\/to\/MSMC\/gsl/g" Makefile
```

Step 6 Perform the compilation.

```
make
```

----End

4.137.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

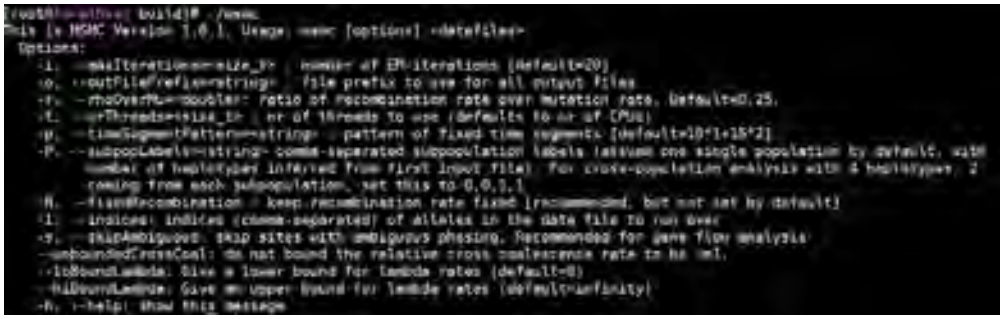
Step 2 Go to the MSMC directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/MSMC/msmc-1.1.0/build
```

Step 3 Start the running.

```
./msmc
```



The program is running properly.

----End

4.138 Prodigal 2.6.3 Porting Guide (Kylin V10)

4.138.1 Introduction

Prodigal is a protein-coding gene prediction software tool for bacterial and archaeal genomes. It performs the following functions:

- Predicts protein-coding genes: Prodigal provides fast, accurate protein-coding gene predictions in GFF3, Genbank, or Sequin table format.
- Handles draft genomes and metagenomes: Prodigal runs smoothly on finished genomes, draft genomes, and metagenomes.
- Runs quickly: Prodigal analyzes the E. coli K-12 genome in 10 seconds on a modern MacBook Pro.
- Runs unsupervised: Prodigal is an unsupervised machine learning algorithm. It does not need to be provided with any training data, and instead automatically learns the properties of the genome from the sequence itself, including genetic code, RBS motif usage, start codon usage, and coding statistics.
- Handles gaps, scaffolds, and partial genes: The user can specify how Prodigal should deal with gaps and has numerous options for allowing or forbidding genes to run into or span gaps.
- Identifies translation initiation sites: Prodigal predicts the correct translation initiation site for most genes, and can output information about every potential start site in the genome, including confidence score, RBS motif, and much more.
- Outputs detailed summary statistics for each genome: Prodigal makes available many statistics for each genome, including contig length, gene length, GC content, GC skew, RBS motifs used, and start and stop codon usage.

For more information, visit the **Prodigal page at GitHub**.

Programming language: C

 Copyright © Huawei Technologies Co., Ltd.

Brief description: A protein-coding gene prediction software tool for bacterial and archaeal genomes.

Open source license: GPL 3.0

Recommended Software Version

Prodigal 2.6.3

4.138.2 Environment Requirements

Hardware Requirements

Table 4-680 lists the hardware requirements.

Table 4-680 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-681 Software requirements

Item	Version	Download URL
Prodigal	2.6.3	https://github.com/hyattpd/Prodigal/archive/refs/tags/v2.6.3.tar.gz
GCC for openEuler	9.3.1	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-linux.tar.gz

OS Requirements

Table 4-682 lists the OS requirements.

Table 4-682 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.138.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Prodigal software porting.

Table 4-683 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/KPGCC	Installation path of GCC for openEuler	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	/path/to/PRODIGAL	Installation path of Prodigal	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.138.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-684 Configuration process

No.	Configuration Item	Remarks
1	Setting up the open source environment	See "Setting Up the Environment for the Single-Node Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Setting up the GCC for openEuler environment	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.

 Copyright © Huawei Technologies Co., Ltd.

4.138.5 Obtaining the Source Code

Procedure

Step 1 Download the Prodigal installation package **v2.6.3.tar.gz**.

https://github.com/hyattpd/Prodigal/archive/refs/tags/v2.6.3.tar.gz

Step 2 Use an SFTP tool to upload the Prodigal installation package to the */path/to/PRODIGAL* directory on the server.

----End

4.138.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/PRODIGAL
```

Step 3 Decompress the installation package.

```
tar xvf v2.6.3.tar.gz
```

Step 4 Perform the compilation.

```
cd Prodigal-2.6.3/
make
```

----End

4.138.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

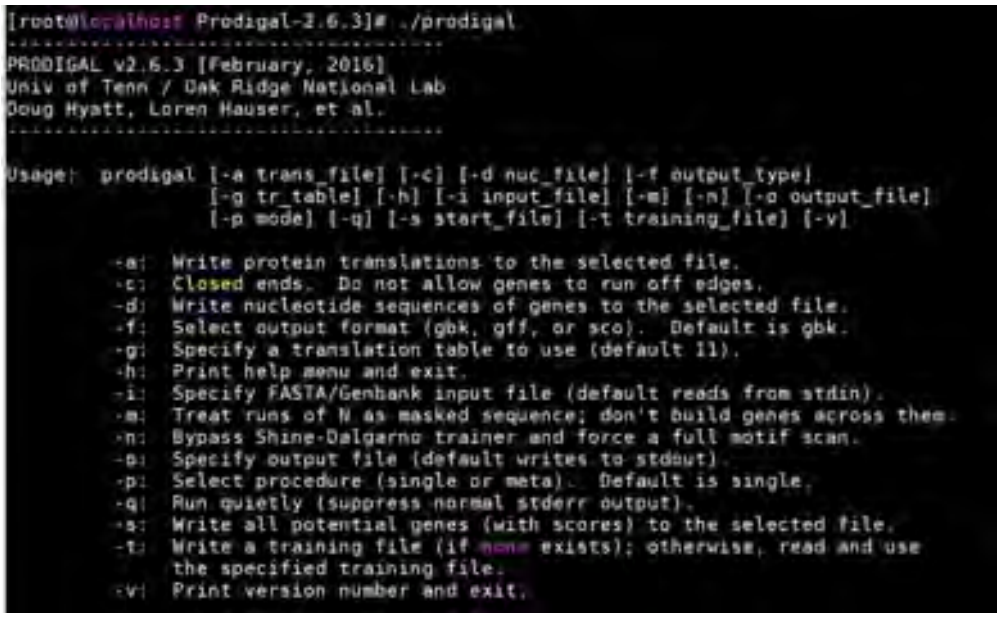
Step 2 Go to the Prodigal directory.

```
cd /path/to/PRODIGAL/Prodigal-2.6.3
```

Step 3 Start the running.

```
./prodigal
```

 Copyright © Huawei Technologies Co., Ltd.



The program is running properly.

----End

4.139 Vaa3D 2.2.0.0 Porting Guide (openEuler 20.03)

4.139.1 Introduction

Vaa3D is open-source software for 3D/4D/5D image visualization and analysis. It is a tool for visualizing large-scale image volumes and surface objects, and supports gigabytes and 64-bit data. It is also a platform to develop new 3D image analysis algorithms for high-throughput processing.

It is a container of powerful modules/plugins for high dimensional image analysis (cell segmentation, neuron tracing, brain registration, annotation, quantitative measurement and statistics, etc), which makes Vaa3D suitable for various bioimage informatics applications.

For more information, visit the **Vaa3D page at GitHub**.

Brief description: Open-source software for 3D/4D/5D image visualization and analysis.

Programming language: C++

Open source license: MIT License

Recommended Software Version

Vaa3D 2.2.0.0

 Copyright © Huawei Technologies Co., Ltd.

4.139.2 Environment Requirements

Hardware Requirements

Table 4-685 lists the hardware requirements.

Table 4-685 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-686 Software requirements

Item	Version	Download URL
Vaa3D	2.2.0.0	https://github.com/Vaa3D/v3d_external.git
szip	2.1.1	https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-2.1.1.tar.gz
teem	1.11.0	http://prdownloads.sourceforge.net/teem/teem-1.11.0-src.tar.gz

OS Requirements

Table lists the OS requirements.

Table 4-687 OS requirements

Item	Version	Download URL
openEuler	20.03 LTS SP2	https://repo.openeuler.org/openEuler-20.03-LTS-SP2/

4.139.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Vaa3D software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-688 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

4.139.4 Configuring the Compilation Environment

4.139.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-689 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing szip	See **4.139.4.2 Installing szip**.
3	Installing teem	See **4.139.4.3 Installing teem**.

4.139.4.2 Installing szip

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/szip
wget https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-2.1.1.tar.gz
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf szip-2.1.1.tar.gz
cd szip-2.1.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Perform the compilation.

```
./configure --prefix=/path/to/szip
make -j && make install
```

----End

4.139.4.3 Installing teem

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/teem
wget http://prdownloads.sourceforge.net/teem/teem-1.11.0-src.tar.gz
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf teem-1.11.0-src.tar.gz
cd teem-1.11.0-src
```

Step 4 Perform the compilation.

```
cmake . -DCMAKE_INSTALL_PREFIX=/path/to/teem
make -j && make install
```

----End

4.139.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the dependency package.

```
yum install hdf5 hdf5-devel qt qt-devel
```

Step 3 Download the installation package.

```
git clone --recursive https://github.com/Vaa3D/v3d_external.git
```

Step 4 Configure the environment.

1. Open the **./build.linux** file.
   ```
   cd v3d_external
   vim ./build.linux
   ```
2. Press **i** to go to the insert mode, and add the following content to the beginning of the file:
   ```
   QT_DIR=/usr/lib64/qt4
   PATH=$QT_DIR/bin:$PATH
   QMAKESPEC=$QT_DIR/bin
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Modify the configuration file.

1. Open **v3d_main/build.redhat**.
   ```
   vim v3d_main/build.redhat
   ```
2. Press **i** to enter the insert mode and add **--build=aarch64-unknown-linux-gnu** to the end of the **./configure** file.
   ```
   36  ../src_packages/tiff-4.0.2/configure --prefix=$tifpath --build=aarch64-unknown-linux-gnu
   ...
   ```

 Copyright © Huawei Technologies Co., Ltd.

```
61   ../src_packages/fftw-3.1.2/configure --prefix=$fftwpath --enable-float --enable-threads --enable-
shared --build=aarch64-unknown-linux-gnu
```

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Open **v3d_main/v3d/v3d_essential.pro**.
   vim v3d_main/v3d/v3d_essential.pro
5. Press i to enter the **insert** mode and add **-lpng** to line 448.
   448 LIBS += -L../common_lib/lib_unix64 -lpng -lteem -lbz2 -lz -lGLU #for nrrd support
6. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Configure szip and teem.

```
cd v3d_main/common_lib/lib_unix64/
mkdir bak
mv *.a bak/
ln -s /path/to/szip/lib/libsz.a libszip.a
ln -s /path/to/teem/lib/libteem.a libteem.a
```

Step 7 Perform the compilation.

```
cd /path/to/v3d_external
./build.linux
```

----End

4.139.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run Vaa3D.

```
cd /path/to/v3d_external/bin
./vaa3d
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.140 WGCNA 1.70 Porting Guide (Kylin V10)

4.140.1 Introduction

The WGCNA R software package is a comprehensive collection of R functions for performing various aspects of weighted correlation network analysis. The package includes functions for network construction, module detection, gene selection, calculations of topological properties, data simulation, visualization, and interfacing with external software. Along with the R package we also present R software tutorials. While the methods development was motivated by gene expression data, the underlying data mining approach can be applied to a variety of different settings.

For more information, visit the **WGCNA official website**.

Programming language: R

Brief description: An R package for weighted correlation network analysis.

Open source license: GPL

Recommended Software Version

WGCNA 1.70

4.140.2 Environment Requirements

Hardware Requirements

Table 4-690 lists the hardware requirements.

Table 4-690 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-691 Software requirements

Item	Version	Download URL
WGCNA	1.70	Install it using the R language. You do not need to download the source package.
R	4.2.0	**https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-692 lists the OS requirements.

Table 4-692 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.140.3 Paths for Software Porting

This chapter lists the software installation paths involved in the WGCNA software porting.

Table 4-693 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/WGCNA	Installation path of WGCNA	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.140.4 Configuring the Compilation Environment

4.140.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-694 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.140.4.2 Installing R-4.2.0**.

4.140.4.2 Installing R-4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/WGCNA
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
```

Step 3 Install the dependencies.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 4 Perform the compilation.

```
./configure --prefix=/path/to/WGCNA/R_install
make
make install
```

Step 5 Load the environment variables.

```
export PATH=/path/to/WGCNA/R_install/bin:$PATH
export LB_LIBRARY_PATH=/path/to/WGCNA/R_install/lib/R/lib:$ LB_LIBRARY_PATH
```

Step 6 (Optional) If the network speed is slow, add a mirrored source of the R language.

1. Create an **~/.Rprofile** file.
   ```
   vi ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.140.5 Obtaining the Source Code

Use the R language to install WGCNA. You do not need to download the source package.

 Copyright © Huawei Technologies Co., Ltd.

4.140.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the R CLI.

```
R
```

Information similar to the following is displayed:

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Enter **install.packages("BiocManager")**.

If the following information is displayed, the installation is successful:

```
> install.packages("BiocManager")
trying URL 'http://mirrors.tuna.tsinghua.edu.cn/CRAN/src/contrib/BiocManager_1.30.16.tar.gz'
Content type 'application/octet-stream' length 289602 bytes (282 KB)
==================================================
downloaded 282 KB

* installing *source* package 'BiocManager' ...
** package 'BiocManager' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package can be loaded from final location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package keeps a record of temporary installation path
* DONE (BiocManager)

The downloaded source packages are in
        '/tmp/RtmpqM5sIe/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Step 3 Install WGCNA.

Enter **BiocManager::install("WGCNA")**.

----End

4.140.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
library(WGCNA)
```

Information similar to the following is displayed:

```
> library(WGCNA)
Loading required package: dynamicTreeCut
Loading required package: fastcluster

Attaching package: 'fastcluster'

The following object is masked from 'package:stats':

    hclust


Attaching package: 'WGCNA'

The following object is masked from 'package:stats':

    cor
```

----End

4.141 AutoDock 4.2.6 Porting Guide (Kylin V10)

4.141.1 Introduction

AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure. It has applications in X-ray crystallography, structure-based

drug design, lead optimization, virtual screening (HTS), combinatorial library design, protein-protein docking, and chemical mechanism studies.

For more information, visit the **AutoDock official website**.

Programming language: C++

Brief description: A suite of automated docking tools.

Open source license: GNU General Public License

Recommended Software Version

AutoDock 4.2.6

4.141.2 Environment Requirements

Hardware Requirements

Table 4-695 lists the hardware requirements.

Table 4-695 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-696 Software requirements

Item	Version	Download URL
AutoDock	4.2.6	**http://autodock.scripps.edu/downloads/autodock-registration/tars/dist426/autodocksuite-4.2.6-src.tar.gz**

OS Requirements

Table 4-697 lists the OS requirements.

Table 4-697 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.141.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AutoDock software porting.

Table 4-698 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/autodock	Installation path of AutoDock	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.141.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-699 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

4.141.5 Obtaining the Source Code

Procedure

Step 1 Download the AutoDock installation package **autodocksuite-4.2.6-src.tar.gz**.

http://autodock.scripps.edu/downloads/autodock-registration/tars/dist426/autodocksuite-4.2.6-src.tar.gz

Step 2 Use an SFTP tool to upload the AutoDock installation package to the */path/to/autodock* directory on the server.

----End

4.141.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/autodock
tar -xvf autodocksuite-4.2.6-src.tar.gz
cd autodocksuite-4.2.6-src/src/
```

Step 3 Create a build file.

```
./autodock/configure --prefix=/path/to/autodock/autodock_install
./autogrid/configure --prefix=/path/to/autodock/autogrid_install
```

Step 4 Perform the compilation.

```
autodock/make -j
autodock/make install
autogrid/make -j
autogrid/make install
```

----End

4.141.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/autodock
./autodock_install/bin/autodock4 -version
./autogrid_install/bin/autogrid4 -version
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

----End

4.142 bioconductor-manta 1.32.0 Porting Guide (Kylin V10)

4.142.1 Introduction

bioconductor-manta is used for microbial assemblage normalized transcript analysis.

For more information, visit the **bioconductor-manta official website**.

Programming language: R

Brief description: Tools for robust comparative metatranscriptomics.

Open source license: custom open source license

Recommended Software Version

bioconductor-manta 1.32.0

4.142.2 Environment Requirements

Hardware Requirements

Table 4-700 lists the hardware requirements.

Table 4-700 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-701 Software requirements

Item	Version	Download URL
bioconductor-manta	1.32.0	http://www.bioconductor.org/packages/3.10/bioc/src/contrib/manta_1.32.0.tar.gz

OS Requirements

Table 4-702 lists the OS requirements.

Table 4-702 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.142.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bioconductor-manta software porting.

Table 4-703 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/bioconductor-manta	Installation path of bioconductor-manta	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.142.4 Configuring the Compilation Environment

4.142.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-704 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.142.4.2 Installing R-4.2.0**.

4.142.4.2 Installing R-4.2.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/bioconductor-manta
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/bioconductor-manta/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/bioconductor-manta/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bioconductor-manta/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirrored source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the first line of the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.142.5 Obtaining the Source Code

Procedure

Step 1 Download the bioconductor-manta installation package **manta_1.32.0.tar.gz**.

http://www.bioconductor.org/packages/3.10/bioc/src/contrib/manta_1.32.0.tar.gz

Step 2 Use an SFTP tool to upload the bioconductor-manta installation package to the */path/to/bioconductor-manta* directory on the server.

----End

4.142.6 Compilation and Installation

Procedure

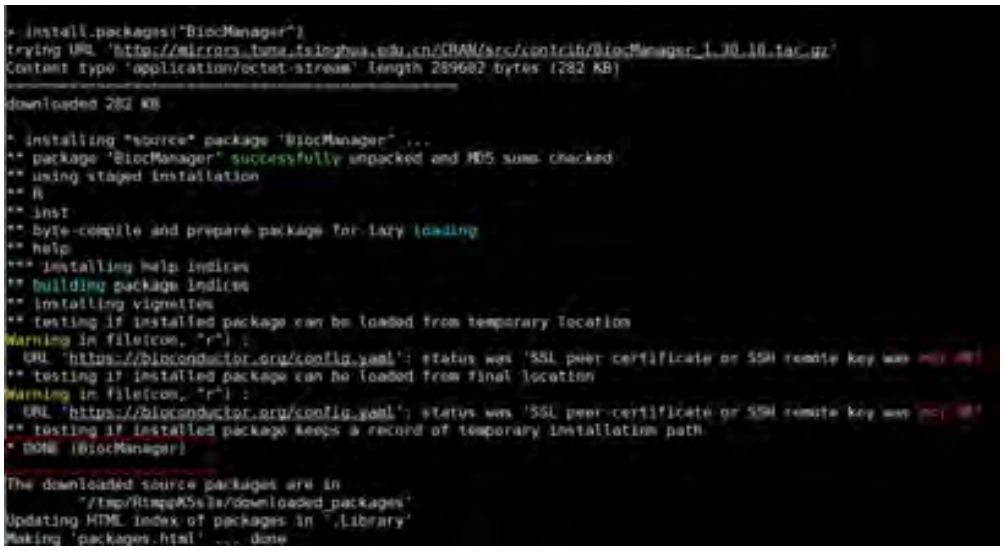
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
R
install.packages("BiocManager")
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



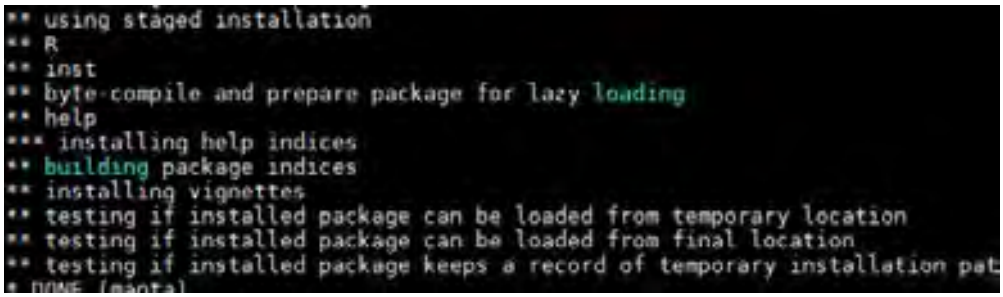
Step 3 Install the dependencies for bioconductor-manta.

```
BiocManager::install("edgeR")
BiocManager::install("Hmisc")
BiocManager::install("caroline")
q()
```

Step 4 Install bioconductor-manta.

```
cd /path/to/bioconductor-manta
R CMD INSTALL manta_1.32.0.tar.gz
```

Information similar to the following is displayed:



----End

4.142.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("manta")
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
> citation("manta")

To cite package 'manta' in publications use:

  David Schruth & Adrian Marchetti (2012). Microbial Assemblage
  Normalized Transcript Analysis. R package version 1.32.0.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {Microbial Assemblage Normalized Transcript Analysis},
    author = {David Schruth and Adrian Marchetti},
    year = {2012},
    note = {R package version 1.32.0},
  }
```

----End

4.143 CellSys 0.1.0 Porting Guide (Kylin V10)

4.143.1 Introduction

CellSys is an open-source tool for building initial structures for bio-membranes and drug-delivery systems.

For more information, visit the **CellSys page at GitHub**.

Programming language: Python

Brief description: An open-source tool for building initial structures for bio-membranes and drug-delivery systems.

Open source license: custom open source license

Recommended Software Version

CellSys 0.1.0

4.143.2 Environment Requirements

Hardware Requirements

Table 4-705 lists the hardware requirements.

Table 4-705 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-706 Software requirements

Item	Version	Download URL
cellsys	0.1.0	**https://github.com/314arhaam/cellsys** (obtained using **git clone**)

OS Requirements

Table 4-707 lists the OS requirements.

Table 4-707 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.143.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CellSys software porting.

Table 4-708 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/cellsys	Installation path of CellSys	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.143.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.143.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-709 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing dependencies	See **4.143.4.2 Installing Dependencies**.

4.143.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Install the dependencies.

```
pip3 install scipy
pip3 install pandas
pip3 install matplotlib
```

----End

4.143.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the CellSys source package.

```
cd /path/to/cellsys
git clone https://github.com/314arhaam/cellsys.git
```

----End

4.143.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the source code directory.

```
cd /path/to/cellsys/cellsys/main
```

Step 3 Perform the installation.

```
python3
import cellsys
```

Information similar to the following is displayed:

----End

4.143.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify CellSys.

```
cd /path/to/cellsys/cellsys/main
python3 systems.py
```

Information similar to the following is displayed:

----End

4.144 cutadapt 4.1 Porting Guide (Kylin V10)

4.144.1 Introduction

Cutadapt finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

For more information, visit the **cutadapt page at GitHub**.

Programming language: Python

Brief description: A genome editing tool.

Open source license: custom open source license

Recommended Software Version

cutadapt 4.1

4.144.2 Environment Requirements

Hardware Requirements

Table 4-710 lists the hardware requirements.

Table 4-710 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-711 Software requirements

Item	Version	Download URL
cutadapt	4.1	**http://mirrors.aliyun.com/pypi/packages/14/a5/b990e46d1c3d309e63357cb730797594b5b7071aaa57b8a4c3d30d007c7f/xopen-1.6.0-py3-none-any.whl** (pip recommended for the installation)

OS Requirements

Table 4-712 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-712 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.144.3 Paths for Software Porting

This chapter lists the software installation paths involved in the cutadapt software porting.

Table 4-713 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/cutadapt	Installation path of cutadapt	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.144.4 Configuring the Compilation Environment

4.144.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-714 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing dependencies	See **4.144.4.2 Installing Dependencies**.

4.144.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Install the dependencies.

```
pip3 install setuptools
pip3 install setuptools_scm
```

----End

4.144.5 Obtaining the Source Code

Run the **pip3 install** command for the installation. You do not need to obtain the source code.

4.144.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/cutadapt
```

Step 3 Perform the installation.

```
pip3 install cutadapt==4.1
```

If the installation fails due to network problems, run the following command to install it:

```
pip3 install cutadapt -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

----End

4.144.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify cutadapt.

```
cutadapt
```

Information similar to the following is displayed:

```
[root@localhost package]# cutadapt
This is cutadapt 4.1 with Python 3.7.9
Command line parameters:
Run "cutadapt --help" to see command-line options.
See https://cutadapt.readthedocs.io/ for full documentation.
```

----End

4.145 deepTools 3.5.1 Porting Guide (Kylin V10)

4.145.1 Introduction

deepTools addresses the challenge of handling the large amounts of data that are now routinely generated from DNA sequencing centers. deepTools contains useful modules to process the mapped reads data for multiple quality checks, creating normalized coverage files in standard bedGraph and bigWig file formats, that allow comparison between different files (for example, treatment and control). Finally, using such normalized and standardized files, deepTools can create many publication-ready visualizations to identify enrichments and for functional annotations of the genome.

For more information, visit the **deepTools page at GitHub**.

Programming language: Python

Brief description: A tool for analyzing and processing deep sequencing data.

Open source license: custom open source license

Recommended Software Version

deepTools 3.5.1

 Copyright © Huawei Technologies Co., Ltd.

4.145.2 Environment Requirements

Hardware Requirements

Table 4-715 lists the hardware requirements.

Table 4-715 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-716 Software requirements

Item	Version	Download URL
deepTools	3.5.1	https://github.com/deeptools/deepTools/archive/refs/tags/3.5.1.tar.gz

OS Requirements

Table 4-717 lists the OS requirements.

Table 4-717 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.145.3 Paths for Software Porting

This chapter lists the software installation paths involved in the deepTools software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-718 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/deeptools	Installation path of deepTools	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.145.4 Configuring the Compilation Environment

4.145.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-719 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Python 3	See **4.145.4.2 Installing Python 3**.

4.145.4.2 Installing Python 3

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Python 3 installation package.

```
cd /path/to/Python3
wget https://www.python.org/ftp/python/3.9.8/Python-3.9.8.tar.xz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the installation package.

```
tar xvf Python-3.9.8.tar.xz
```

Step 4 Install the Python dependencies.

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel
```

Step 5 Install Python.

```
cd Python-3.9.8
./configure --prefix=/path/to/Python-3.9.8-install
make
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/Python-3.9.8-install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/Python-3.9.8-install/lib:$LD_LIBRARY_PATH
```

----End

4.145.5 Obtaining the Source Code

Procedure

Step 1 Download the deepTools installation package **3.5.1.tar.gz**.

https://github.com/deeptools/deepTools/archive/refs/tags/3.5.1.tar.gz

Step 2 Use an SFTP tool to upload the deepTools installation package to the */path/to/deeptools* directory on the server.

----End

4.145.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/deeptools
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf 3.5.1.tar.gz
cd deepTools-3.5.1
```

Step 4 Perform the installation.

```
python3 setup.py install
```

----End

4.145.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Verify deepTools.

```
deeptools
```

Information similar to the following is displayed:

----End

4.146 DESeq2 1.36.0 Porting Guide (Kylin V10)

4.146.1 Introduction

DESeq2 estimates variance-mean dependence in count data from high-throughput sequencing assays and tests for differential expression based on a model using the negative binomial distribution.

For more information, visit the **DESeq2 web page**.

Programming language: R

Brief description: Differential expression based on a model using the negative binomial distribution.

Open source license: custom open source license

Recommended Software Version

DESeq2 1.36.0

4.146.2 Environment Requirements

Hardware Requirements

Table 4-720 lists the hardware requirements.

Table 4-720 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-721 Software requirements

Item	Version	Download URL
DESeq2	1.36.0	Install it using the R language. You do not need to download the source package.

OS Requirements

Table 4-722 lists the OS requirements.

Table 4-722 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.146.3 Paths for Software Porting

This chapter lists the software installation paths involved in the DESeq2 software porting.

Table 4-723 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/DESEQ2	Installation path of DESeq2	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.146.4 Configuring the Compilation Environment

4.146.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-724 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.146.4.2 Installing R-4.2.0**.

4.146.4.2 Installing R-4.2.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/DESEQ2
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/DESEQ2/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/DESEQ2/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/DESEQ2/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirrored source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.146.5 Obtaining the Source Code

Use the R language to install DESeq2. You do not need to download the source package.

4.146.6 Compilation and Installation

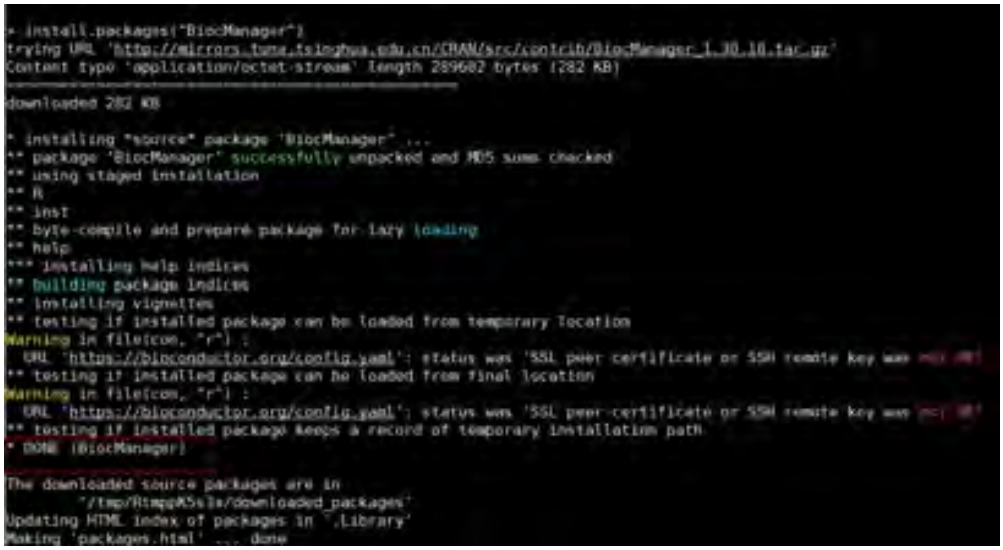
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
R
install.packages("BiocManager")
```

Information similar to the following is displayed:



Step 3 Install DESeq2.

```
BiocManager::install("DESeq2")
```

----End

4.146.7 Running and Verification

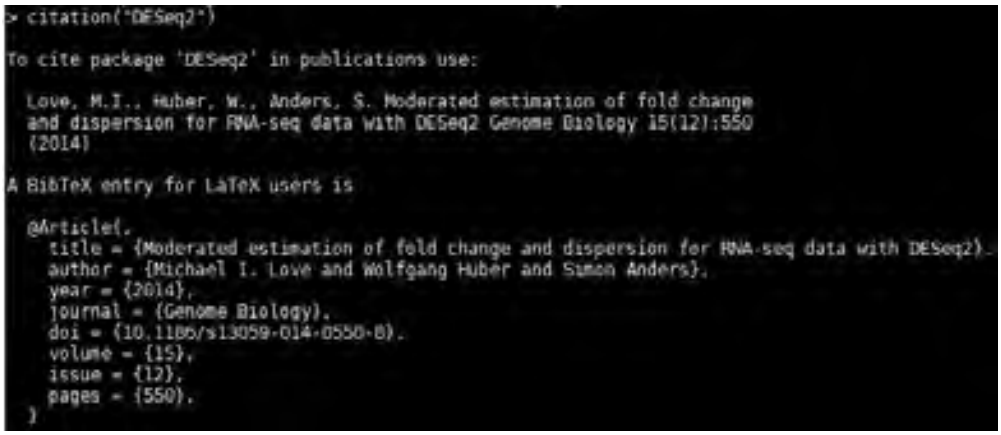
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("DESeq2")
```

Information similar to the following is displayed:



----End

4.147 featurecounts 2.0.3 Porting Guide (Kylin V10)

4.147.1 Introduction

featureCounts is a software program developed for counting reads to genomic features such as genes, exons, promoters and genomic bins. It is included in the Subread package.

For more information, visit the **featureCounts web page**.

Programming language: C

Brief description: Counts reads to genomic features.

Open source license: custom open source license

Recommended Software Version

featureCounts 2.0.3

4.147.2 Environment Requirements

Hardware Requirements

Table 4-725 lists the hardware requirements.

Table 4-725 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-726 Software requirements

Item	Version	Download URL
featureCounts	2.0.3	https://sourceforge.net/projects/subread/files/subread-2.0.3/subread-2.0.3-source.tar.gz

OS Requirements

Table 4-727 lists the OS requirements.

Table 4-727 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.147.3 Paths for Software Porting

This chapter lists the software installation paths involved in the featureCounts software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-728 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/featurecounts	Installation path of featureCounts	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.147.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-729 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.147.5 Obtaining the Source Code

Procedure

Step 1 Download the featureCounts installation package **subread-2.0.3-source.tar.gz**.

https://sourceforge.net/projects/subread/files/subread-2.0.3/subread-2.0.3-source.tar.gz

Step 2 Use an SFTP tool to upload the featureCounts installation package to the */path/to/featureCounts* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.147.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/featurecounts
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf subread-2.0.3-source.tar.gz
cd subread-2.0.3-source
```

Step 4 Modify the following file.

1. Open **Makefile.Linux**.
   ```
   vim src/Makefile.Linux
   ```
2. Press **i** to enter the insert mode and change **core2** in line 10 to **tsv110**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

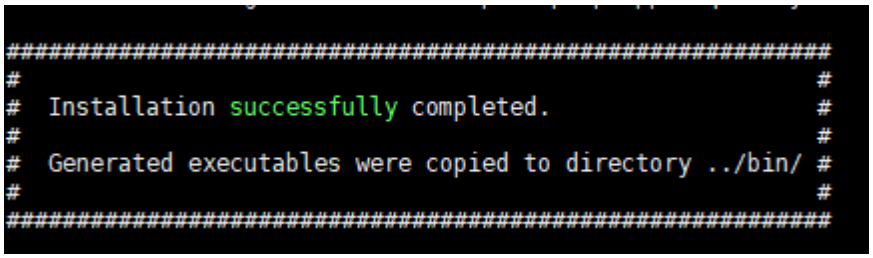
Step 5 Modify the following file.

1. Open **Makefile**.
   ```
   vim src/longread-one/Makefile
   ```
2. Press **i** to enter the insert mode and change **core2** in line 7 to **tsv110**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
cd src
make -f Makefile.Linux
```

Information similar to the following is displayed:



----End

4.147.7 Running and Verification

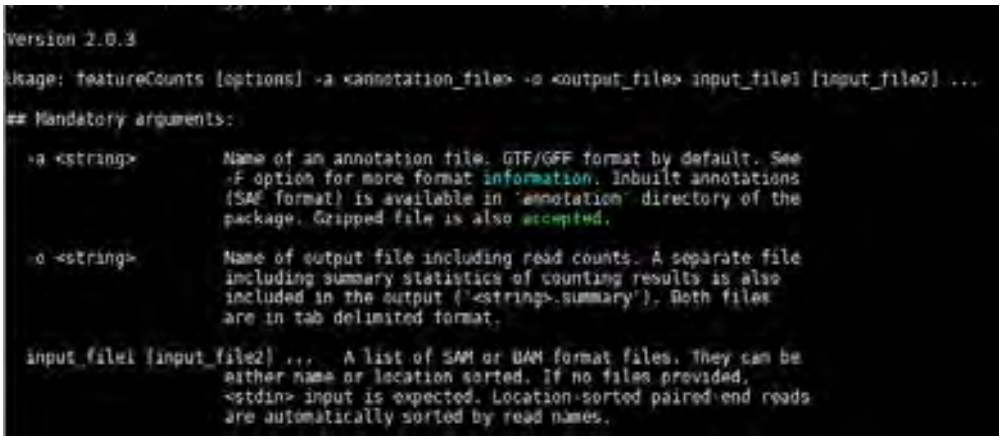
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify featureCounts.

```
cd /path/to/featurecounts/subread-2.0.3-source/bin
./featurecounts
```

Information similar to the following is displayed:



----End

4.148 ggplot2 3.3.6 Porting Guide (Kylin V10)

4.148.1 Introduction

ggplot2 is an R package used for statistical computing and data representation using data visualization.

For more information, visit the **ggplot2 page at GitHub**.

Programming language: R

Brief description: An R package used for statistical computing and data representation using data visualization.

Open source license: custom open source license

Recommended Software Version

ggplot2 3.3.6

4.148.2 Environment Requirements

Hardware Requirements

Table 4-730 lists the hardware requirements.

Table 4-730 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-731 Software requirements

Item	Version	Download URL
ggplot2	3.3.6	Install it using the R language. You do not need to download the source package.

OS Requirements

Table 4-732 lists the OS requirements.

Table 4-732 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.148.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ggplot2 software porting.

Table 4-733 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/ggplot2	Installation path of ggplot2	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.148.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.148.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-734 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.148.4.2 Installing R-4.2.0**.

4.148.4.2 Installing R-4.2.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/ggplot2
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/ggplot2/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/ggplot2/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/ggplot2/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirrored source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.148.5 Obtaining the Source Code

Use the R language to install ggplot2. You do not need to download the source package.

4.148.6 Compilation and Installation

Procedure

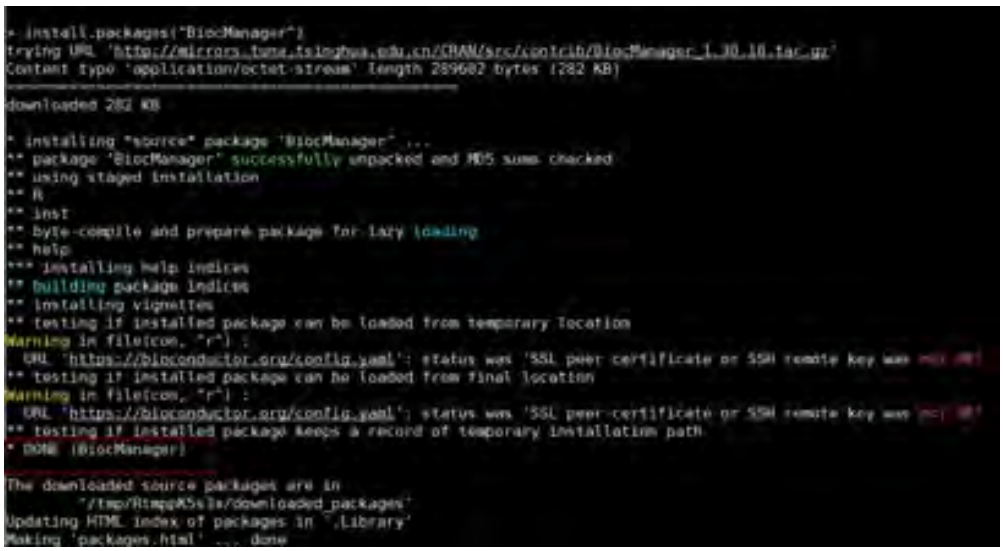
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to R.

```
R
```

Enter **install.packages("BiocManager")**.

Information similar to the following is displayed:



Step 3 Install ggplot2.

```
BiocManager::install("ggplot2")
```

----End

4.148.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("ggplot2")
```

Information similar to the following is displayed:

```
> citation("ggplot2")

To cite ggplot2 in publications, please use:

  H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
  Springer-Verlag New York, 2016.

A BibTeX entry for LaTeX users is

  @Book{,
    author = {Hadley Wickham},
    title = {ggplot2: Elegant Graphics for Data Analysis},
    publisher = {Springer-Verlag New York},
    year = {2016},
    isbn = {978-3-319-24277-4},
    url = {https://ggplot2.tidyverse.org},
  }
```

----End

4.149 MultiQC 1.12 Porting Guide (Kylin V10)

4.149.1 Introduction

MultiQC is a tool to create a single report with interactive plots for multiple bioinformatics analyses across many samples.

For more information, visit the **MultiQC page at GitHub**.

Programming language: Python

Brief description: A tool to create a single report with interactive plots for multiple bioinformatics analyses across many samples.

Open source license: custom open source license

Recommended Software Version

MultiQC 1.12

4.149.2 Environment Requirements

Hardware Requirements

Table 4-735 lists the hardware requirements.

Table 4-735 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software requirements

Table lists the software requirements.

Table 4-736 Software requirements

Item	Version	Download URL
MultiQC	1.12	https://github.com/ewels/MultiQC/archive/refs/tags/v1.12.tar.gz

OS Requirements

Table 4-737 lists the OS requirements.

Table 4-737 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.149.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MultiQC software porting.

Table 4-738 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/multiqc	Installation path of MultiQC	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.149.4 Configuring the Compilation Environment

4.149.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-739 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Python 3	See **4.149.4.2 Installing Python 3**.

4.149.4.2 Installing Python 3

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the Python 3 installation package.

```
cd /path/to/Python3
wget https://www.python.org/ftp/python/3.9.8/Python-3.9.8.tar.xz
```

Step 3 Decompress the installation package.

```
tar xvf Python-3.9.8.tar.xz
```

Step 4 Install the Python 3 dependencies.

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel
gdbm-devel db4-devel libpcap-devel xz-devel
```

Step 5 Install Python 3.

```
cd Python-3.9.8
./configure --prefix=/path/to/Python-3.9.8-install
make
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/Python-3.9.8-install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/Python-3.9.8-install/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.149.5 Obtaining the Source Code

Procedure

Step 1 Download the MultiQC installation package **v1.12.tar.gz**.

https://github.com/ewels/MultiQC/archive/refs/tags/v1.12.tar.gz

Step 2 Use an SFTP tool to upload the MultiQC installation package to the */path/to/multiqc* directory on the server.

----End

4.149.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/multiqc
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf v1.12.tar.gz
cd MultiQC-1.12
```

Step 4 Install the software.

```
python3 setup.py install
```

Information similar to the following is displayed:

```
Finished processing dependencies for multiqc==1.12

--------------------------------
 MultiQC installation complete!
--------------------------------
For help in running MultiQC, please see the documentation available
at http://multiqc.info or run: multiqc --help
```

----End

4.149.7 Running and Verification

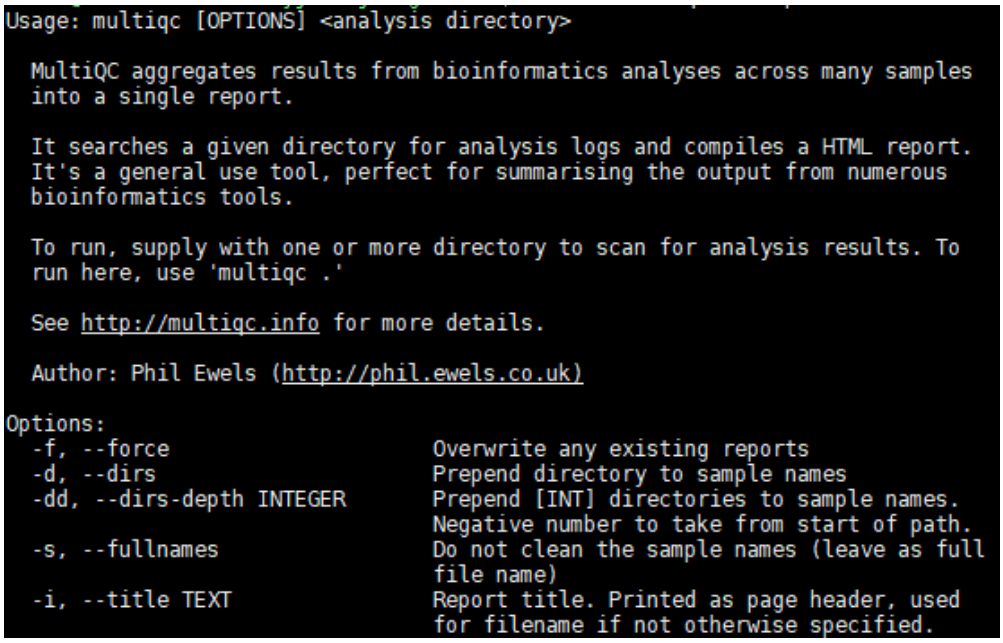
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify MultiQC.

```
multiqc --help
```

Information similar to the following is displayed:



----End

4.150 Open Babel 3.1.1 Porting Guide (Kylin V10)

4.150.1 Introduction

Open Babel is a chemical toolbox designed to speak the many languages of chemical data. It's an open, collaborative project allowing anyone to search, convert, analyze, or store data from molecular modeling, chemistry, solid-state materials, biochemistry, or related areas.

For more information, visit the **Open Babel page at GitHub**.

Programming languages: C/C++

Brief description: A chemical toolbox.

Open source license: custom open source license

Recommended Software Version

Open Babel 3.1.1

4.150.2 Environment Requirements

Hardware Requirements

Table 4-740 lists the hardware requirements.

Table 4-740 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-741 Software requirements

Item	Version	Download URL
Open Babel	3.1.1	https://github.com/openbabel/openbabel/archive/refs/tags/openbabel-3-1-1.tar.gz

OS Requirements

Table 4-742 lists the OS requirements.

Table 4-742 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.150.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Open Babel software porting.

Table 4-743 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/openbabel	Installation path of Open Babel.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.150.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-744 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.150.5 Obtaining the Source Code

Procedure

Step 1 Download the Open Babel installation package **openbabel-3-1-1.tar.gz**.

https://github.com/openbabel/openbabel/archive/refs/tags/openbabel-3-1-1.tar.gz

Step 2 Use an SFTP tool to upload the Open Babel installation package to the */path/to/openbabel* directory on the server.

----End

4.150.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the source package directory.

```
cd /path/to/openbabel
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf openbabel-3-1-1.tar.gz
cd openbabel-3.1.1
```

Step 4 Download the Eigen3 package and decompress it.

```
wget https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz
tar xvf eigen-3.4.0.tar.gz
```

Step 5 Create a build directory and go to it.

```
mkidr build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake .. -DEIGEN3_INCLUDE_DIR=/path/to/openbabel/openbabel-3.1.1/eigen-3.4.0 -
DCMAKE_INSTALL_PREFIX=/path/to/openbabel/openbabel-3.1.1-install
make
make install
```

----End

4.150.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify Open Babel.

```
cd /path/to/openbabel/openbabel-3.1.1-install/bin
./obabel
```

Information similar to the following is displayed:

```
No input file or format spec or possibly a misplaced option.
Most options must come after the input files. (-i -o -O -m can be anywhwere.)

Open Babel 3.1.0 -- Aug  6 2022 -- 16:15:24
Usage:
obabel [-i<input-type>] <infilename> [-o<output-type>] -O<outfilename> [Options]
Try  -H option for more information.
```

----End

4.151 PBSuite 15.8.24 Porting Guide (Kylin V10)

4.151.1 Introduction

This currently hosts two projects created and maintained by Adam English.

PBJelly - the genome upgrading tool. PBJelly is a highly automated pipeline that aligns long sequencing reads (such as PacBio RS reads or long 454 reads in fasta format) to high-confidence draft assembles. PBJelly fills or reduces as many captured gaps as possible to produce upgraded draft genomes.

PBHoney - the structural variation discovery tool. PBHoney is an implementation of two variant-identification approaches designed to exploit the high mappability

 Copyright © Huawei Technologies Co., Ltd.

of long reads (i.e., greater than 10,000 bp). PBHoney considers both intra-read discordance and soft-clipped tails of long reads to identify structural variants.

For more information, visit the **PBSuite web page**.

Programming language: Python

Brief description: The genome upgrading tool and the structural variation discovery tool.

Open source license: custom open source license

Recommended Software Version

PBSuite 15.8.24

4.151.2 Environment Requirements

Hardware Requirements

Table 4-745 lists the hardware requirements.

Table 4-745 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-746 Software requirements

Item	Version	Download URL
PBSuite	15.8.24	**https://sourceforge.net/projects/pb-jelly/files/PBSuite_15.8.24.tgz**

OS Requirements

Table 4-747 lists the OS requirements.

Table 4-747 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.151.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PBSuite software porting.

Table 4-748 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/PBSuite	Installation path of PBSuite	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.151.4 Configuring the Compilation Environment

4.151.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-749 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing Python 2	See **4.151.4.2 Installing Python 2**.
3	Installing dependencies	See **4.151.4.3 Installing Dependencies**.
4	Installing blasr	See **blasr Porting Guide (Kylin V10)**.

4.151.4.2 Installing Python 2

Procedure

Step 1 Download the installation package.

```
cd /path/to/Python2
wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz
```

Step 2 Decompress the installation package.

```
tar xvf Python-2.7.18.tgz
```

Step 3 Install the dependencies.

```
yum install gcc libffi* openssl libssl zlib* bzip2
```

Step 4 Install Python.

```
cd Python-2.7.18
./configure --prefix=/path/to/Python-2.7.18-install --with-ensurepip=install
make
make install
```

Step 5 Load the environment variables.

```
export PATH=/path/to/Python-2.7.18-install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/Python-2.7.18-install/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/Python-2.7.18-install/include:$INCLUDE
```

----End

4.151.4.3 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies using the **pip** command.

```
pip install intervaltree==2.1.0
pip install intervaltree-bio==1.0.1
pip install network==1.1
pip install numpy==1.16.6
pip install PyIntervalTree==0.5
pip install pysam==0.8.0
pip install PyVCF==0.6.8
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.151.5 Obtaining the Source Code

Procedure

Step 1 Download the PBSuite installation package **PBSuite_15.8.24.tgz**.

https://sourceforge.net/projects/pb-jelly/files/PBSuite_15.8.24.tgz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/PBSuite* directory on the server.

----End

4.151.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/PBSuite
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf PBSuite_15.8.24.tgz
mv PBSuite_15.8.24 PBSuite
cd PBSuite
```

Step 4 Modify the installation script.

1. Open **setup.sh**.
   ```
   vim setup.sh
   ```
2. Press **i** to enter the insert mode and change the content in line 8 to **export SWEETPATH=/path/to/PBSuite/PBSuite**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Install the script.

```
./setup.sh
```

----End

4.151.7 Running and Verification

Procedure

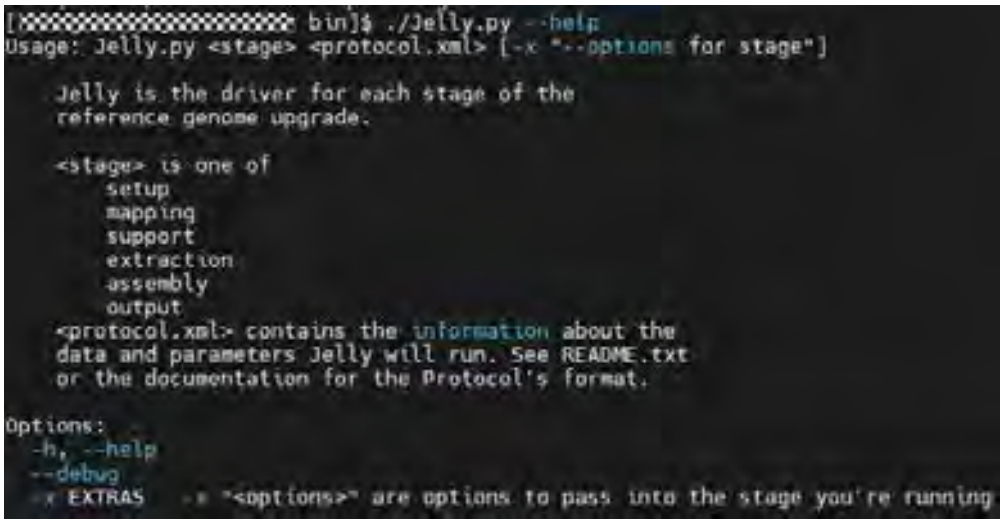
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify PBSuite.

```
cd /path/to/PBSuite/PBSuite/bin
./Jelly.py --help
```

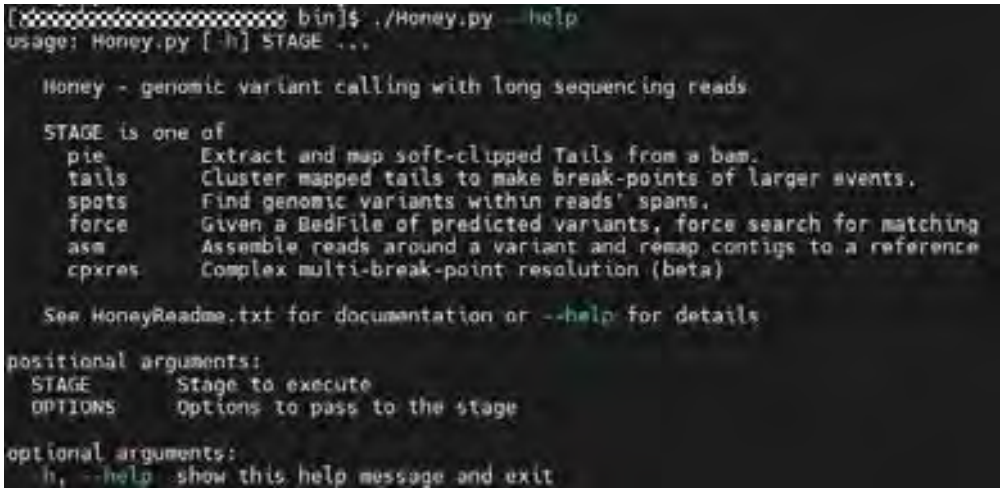
Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



./Honey.py --help

Information similar to the following is displayed:



----End

4.152 psipred 4.0 Porting Guide (Kylin V10)

4.152.1 Introduction

psipred is a simple and accurate protein secondary structure prediction method, incorporating two feed-forward neural networks which perform an analysis on output obtained from PSI-BLAST.

For more information, visit the **psipred page at GitHub**.

Programming languages: Java/C

Brief description: A protein secondary structure prediction tool.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

psipred 4.0

4.152.2 Environment Requirements

Hardware Requirements

Table 4-750 lists the hardware requirements.

Table 4-750 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-751 Software requirements

Item	Version	Download URL
psipred	4.0	https://github.com/psipred/psipred/archive/refs/tags/v4.0.tar.gz
blast+	2.13	https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.13.0+-src.tar.gz

OS Requirements

Table 4-752 lists the OS requirements.

Table 4-752 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.152.3 Paths for Software Porting

This chapter lists the software installation paths involved in the psipred software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-753 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/psipred	Installation path of psipred	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/blast+	Installation path of blast+	

4.152.4 Configuring the Compilation Environment

4.152.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-754 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
4	Installing blast+	See **4.152.4.2 Installing blast+**.

 Copyright © Huawei Technologies Co., Ltd.

4.152.4.2 Installing blast+

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/blast+
wget https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.13.0+-src.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf ncbi-blast-2.13.0+-src.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd ncbi-blast-2.13.0+-src
export BLAST_TOP=`pwd`
```

Step 5 Download the LMDB package and decompress it.

```
wget https://github.com/LMDB/lmdb/archive/refs/tags/LMDB_0.9.24.tar.gz
tar xvf LMDB_0.9.24.tar.gz
export LMDB_PATH=$BLAST_TOP/lmdb-LMDB_0.9.24
```

Step 6 Switch to the **c++** subdirectory under **blast+**.

```
cd $BLAST_TOP/c++
```

Step 7 Create a configuration file.

```
./configure
```

Step 8 Perform the compilation.

```
cd ReleaseMT/build
make all_r -j40
```

Step 9 Set the environment variable.

```
export PATH=/path/to/blast+/ncbi-blast-2.13.0+-src/c++/ReleaseMT/bin:$PATH
```

----End

4.152.5 Obtaining the Source Code

Procedure

Step 1 Download the psipred installation package **v4.0.tar.gz**.

https://github.com/psipred/psipred/archive/refs/tags/v4.0.tar.gz

Step 2 Use an SFTP tool to upload the psipred installation package to the */path/to/psipred* directory on the server.

----End

4.152.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/psipred
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf v4.0.tar.gz
cd psipred-4.0/src
```

Step 4 Perform the compilation.

```
make -j
make install
```

Step 5 Go to the directory generated after the decompression.

```
cd ../BLAST+
```

Step 6 Create a database directory and go to it.

```
mkdir database && cd database
```

Step 7 Download and use the database.

```
wget http://ftp.ncbi.nih.gov/blast/db/FASTA/swissprot.gz
gunzip swissprot.gz
makeblastdb -dbtype prot -in swissprot -out swissprot
```

Step 8 Modify the run file.

1. Open **runpsipredplus**.
   ```
   cd ..
   vim runpsipredplus
   ```
2. Press **i** to enter the insert mode and modify lines 14 and 17.
   ```
   set dbname = /path/to/psipred-4.0/BLAST+/database/swissprot
   set ncbidir = /path/to/blast+/ncbi-blast-2.13.0+-src/c++/ReleaseMT/bin
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.152.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
export PATH=/path/to/psipred-4.0/bin:$PATH
cd psipred-4.0/BLAST+
./runpsipredplus ../example/example.fasta
```

Information similar to the following is displayed:

```
[root@localhost BLAST+]# ./runpsipredplus ../example/example.fasta
Running PSI-BLAST with sequence ../example/example.fasta ...
Predicting secondary structure...
Pass1 ...
Pass2 ...
Cleaning up ...
Final output files: example.ss2 example.horiz
Finished.
[root@localhost BLAST+]#
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.153 SNPhylo 20180901 Porting Guide (Kylin V10)

4.153.1 Introduction

Phylogenetic trees are a good tool to infer evolutionary relationships among various organisms and have been used in many evolutionary studies. Consequently, phylogenetic trees based on SNP data have been determined in resequencing projects. However, there was no simple way to determine phylogenetic trees with the huge number of variants determined from resequencing data. Thus, a new pipeline, SNPhylo, is developed to construct phylogenetic trees based on SNP data. With this pipeline, users can construct a phylogenetic tree from a file containing huge SNP data.

For more information, visit the **SNPhylo page at GitHub**.

Programming language: R

Brief description: A tool to infer evolutionary relationships among various organisms.

Open source license: custom open source license

Recommended Software Version

SNPhylo 20180901

4.153.2 Environment Requirements

Hardware Requirements

Table 4-755 lists the hardware requirements.

Table 4-755 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-756 Software requirements

Item	Version	Download URL
SNPhylo	20180901	**https://github.com/thlee/SNPhylo/archive/refs/tags/20180901.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-757 lists the OS requirements.

Table 4-757 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.153.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SNPhylo software porting.

Table 4-758 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/snphylo	Installation path of SNPhylo	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.153.4 Configuring the Compilation Environment

4.153.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-759 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the R language	See **4.153.4.2 Installing R-4.2.0**.
3	Installing dependencies	See **4.153.4.3 Installing Dependencies**.
4	Installing Python 2	See **4.153.4.4 Installing Python 2**.
5	Installing Muscle	See **Muscle 5.0.1428 Porting Guide (CentOS 7.9)**.
6	Installing phylip	See **4.153.4.5 Installing phylip**.

4.153.4.2 Installing R-4.2.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/snphylo
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/snphylo/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/snphylo/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/snphylo/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirrored source of the R language.

1. Create an **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

4.153.4.3 Installing Dependencies

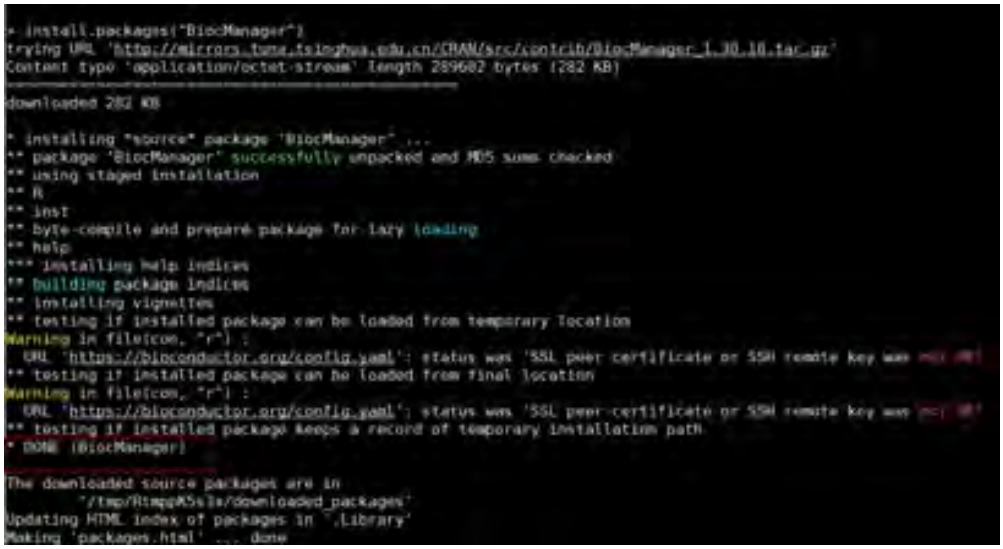
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
R
install.packages("BiocManager")
```

Information similar to the following is displayed:



Enter **BiocManager::install(c("getopt","gdsfmt","SNPRelate","ape","phangorn"))** and wait until the installation is complete.

----End

4.153.4.4 Installing Python 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/Python2
wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz
```

Step 3 Decompress the installation package.

```
tar xvf Python-2.7.18.tgz
```

Step 4 Install the dependencies.

```
yum install gcc libffi* openssl libssl zlib* bzip2
```

Step 5 Install Python.

```
cd Python-2.7.18
./configure --prefix=/path/to/Python-2.7.18-install --with-ensurepip=install
make
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/Python-2.7.18-install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/Python-2.7.18-install/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/Python-2.7.18-install/include:$INCLUDE
```

----End

4.153.4.5 Installing phylip

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/phylip
wget http://evolution.gs.washington.edu/phylip/download/phylip-3.697.tar.gz
```

Step 3 Decompress the installation package.

```
tar -xvf phylip-3.697.tar.gz
```

Step 4 Go to the file directory.

```
cd phylip-3.697/src
```

Step 5 Compile the program.

```
make -f Makefile.unx install
```

After the compilation is complete, the executable file **dnaml** is generated in the upper-level directory **exe**.

----End

4.153.5 Obtaining the Source Code

Procedure

Step 1 Download the SNPhylo installation package **20180901.tar.gz**.

https://github.com/thlee/SNPhylo/archive/refs/tags/20180901.tar.gz

Step 2 Use an SFTP tool to upload the SNPhylo installation package to the */path/to/snphylo* directory on the server.

----End

4.153.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/snphylo
tar xvf 20180901.tar.gz
cd SNPhylo-20180901
```

Step 3 Modify the configuration file.

1. Open **snphylo.cfg**.
   ```
   vim snphylo.cfg
   ```
2. Press **i** to enter the insert mode and modify the file as follows:

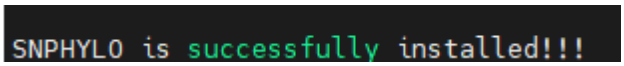
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Run the installation script.

```
sh setup.sh
```

Type **y** as prompted. If the system displays a message indicating that Muscle and dnaml are not found, type the paths of the Muscle and dnaml executable files.

If the following information is displayed, the installation is successful:



----End

4.153.7 Running and Verification

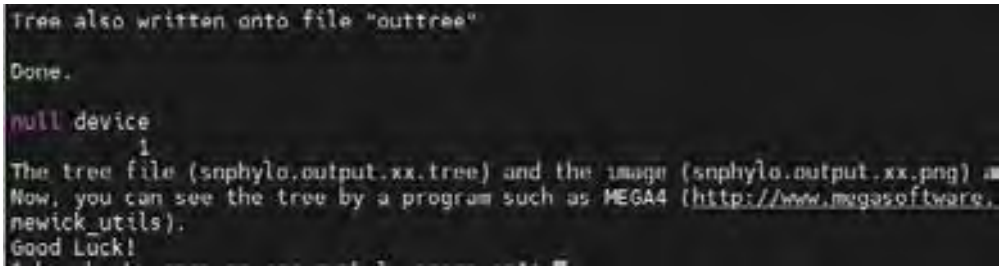
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/snphylo/SNPhylo-20180901/samples
bzip2 -d soybean.partial.hapmap.bz2
cd ../
./snophylp.sh -H samples/soybean.partial.hapmap
```

Information similar to the following is displayed:



----End

4.154 TreeBeST 1.9.2 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

4.154.1 Introduction

TreeBeST, which stands for (gene) Tree Building guided by Species Tree, is a versatile program that builds, manipulates and displays phylogenetic trees. It is particularly designed for building gene trees with a known species tree and is highly efficient and accurate.

For more information, visit the **TreeBeST web page**.

Programming language: C

Brief description: A multifunctional program for building, manipulating, and displaying phylogenetic trees.

Open source license: custom open source license

Recommended Software Version

TreeBeST 1.9.2

4.154.2 Environment Requirements

Hardware Requirements

Table 4-760 lists the hardware requirements.

Table 4-760 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-761 Software requirements

Item	Version	Download URL
TreeBeST	1.9.2	**https://sourceforge.net/projects/treesoft/files/treebest/1.9.2/treebest-1.9.2.tar.gz**

OS Requirements

Table 4-762 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-762 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.154.3 Paths for Software Porting

This chapter lists the software installation paths involved in the TreeBeST software porting.

Table 4-763 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/treebest	Installation path of TreeBeST	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.154.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-764 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.154.5 Obtaining the Source Code

Procedure

Step 1 Download the TreeBeST installation package **treebest-1.9.2.tar.gz**.

https://sourceforge.net/projects/treesoft/files/treebest/1.9.2/treebest-1.9.2.tar.gz

Step 2 Use an SFTP tool to upload the TreeBeST installation package to the */path/to/treebest* directory on the server.

----End

4.154.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory and decompress the installation package.

```
cd /path/to/treebest
tar -xvf treebest-1.9.2.tar.gz
mv treebest-1.9.2 treebest
```

Step 3 Go to the directory generated after decompression.

```
cd treebest
```

Step 4 Modify the following file.

1. Open **common/hash_com.h**.
   ```
   vim common/hash_com.h
   ```
2. Press **i** to enter the insert mode and change **false** in line 197 to **0**.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Open **Makefile**.
   ```
   vim Makefile
   ```
5. Press **i** to enter the insert mode. Change lines 1 and 2 to **CC=clang** and **CXX=clang++** respectively, and add **-Wl,-z,-muldefs** to line 3.
6. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 5 Perform the compilation.

```
make
```

----End

4.154.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/treebest/treebest
./treebest
```

Information similar to the following is displayed:

```
[root@localhost treebest-1.9.2]# ./treebest

Program: TreeBeST (gene Tree Building guided by Species Tree)
Version: 1.9.2 build 28Sep2022
Contact: Heng Li <lh3@sanger.ac.uk>

Usage:   treebest <command> [options]

Command: nj         build neighbour-joining tree, SDI, rooting
         best       build tree with the help of a species tree
         phyml      build phyml tree
         sdi        speciation vs. duplication inference
         spec       print species tree
         format     reformat a tree
         filter     filter a multi-alignment
         trans      translate coding nucleotide alignment
         backtrans  translate aa alignment back to nt
         leaf       get external nodes
         mfa2aln    convert MFA to ALN format
         ortho      ortholog/paralog inference
         distmat    distance matrix
         treedist   topological distance between two trees
         pwalign    pairwise alignment
         mmerge     merge a forest
         export     export a tree to EPS format
         subtree    extract the subtree
         simulate   simulate a gene tree
         sortleaf   sort leaf order
         estlen     estimate branch length
         trimpoor   trim out leaves that affect the quality of a tree
         root       root a tree by minimizing height
```

----End

4.155 BLASR Porting Guide (Kylin V10)

4.155.1 Introduction

BLASR it the PacBio® long read aligner.

For more information, visit the **BLASR page at GitHub**.

Programming language: C

 Copyright © Huawei Technologies Co., Ltd.

Brief description: The PacBio® long read aligner.

Open source license: custom open source license

Recommended Software Version

N/A

4.155.2 Environment Requirements

Hardware Requirements

Table 4-765 lists the hardware requirements.

Table 4-765 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-766 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz

OS Requirements

Table lists the OS requirements.

Table 4-767 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.18.0 (4.14)	Included in the OS image

4.155.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BLASR software porting.

Table 4-768 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/HDF5	Installation path of HDF5	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.155.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-769 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
4	Installing HDF5	See "Installing HDF5" in **FVCOM 4.1 Porting Guide (openEuler 21.03)**.

4.155.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the BLASR installation package.

```
git clone https://github.com/BioinformaticsArchive/blasr.git
```

----End

4.155.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

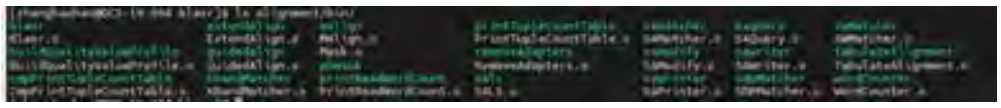
```
cd blasr
```

Step 3 Load the HDF5 environment.

```
export HDF5INCLUDEDIR=/path/to/HDF5/include
export HDF5LIBDIR=/path/to/HDF5/lib
```

Step 4 Perform the compilation and installation.

```
make
```



A BLASR executable file is generated in **alignment/bin**.

----End

4.155.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the following directory:

```
cd alignment/bin
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Perform the verification.

```
./blasr -h
```

----End

4.156 BreakDancer Porting Guide (Kylin V10)

4.156.1 Introduction

BreakDancer, released under GPLv3, is a Cpp package that provides genome-wide detection of structural variants from next generation paired-end sequencing reads. It includes two complementary programs. BreakDancerMax predicts five types of structural variants: insertions, deletions, inversions, inter- and intra-chromosomal translocations from next-generation short paired-end sequencing reads using read pairs that are mapped with unexpected separation distances or orientation.

For more information, visit the **BreakDancer page at GitHub**.

Programming language: C++

Brief description: A Cpp package that provides genome-wide detection of structural variants from next generation paired-end sequencing reads.

Open source license: GPLv3

Recommended Software Version

master

4.156.2 Environment Requirements

Hardware Requirements

Table 4-770 lists the hardware requirements.

Table 4-770 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table lists the software requirements.

Table 4-771 Software requirements

Item	Version	Download URL
breakdancer	master	https://github.com/genome/breakdancer

OS Requirements

Table lists the OS requirements.

Table 4-772 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.156.3 Paths for Software Porting

This chapter lists the software installation paths involved in the BreakDancer software porting.

Table 4-773 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/breakdancer	Installation path of BreakDancer	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.156.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-774 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.156.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the BreakDancer installation package.

```
cd /path/to/breakdancer
git clone --recursive https://github.com/genome/breakdancer.git
```

----End

4.156.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/breakdancer/breakdancer
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Perform the compilation and installation.

```
mkdir build && cd build
cmake .. -DCMAKE_BUILD_TYPE=release -DCMAKE_INSTALL_PREFIX=/path/to/breakdancer
make && make install
```

Step 4 Load the environment variable.

```
export PATH=/path/to/breakdancer/bin:$PATH
```

----End

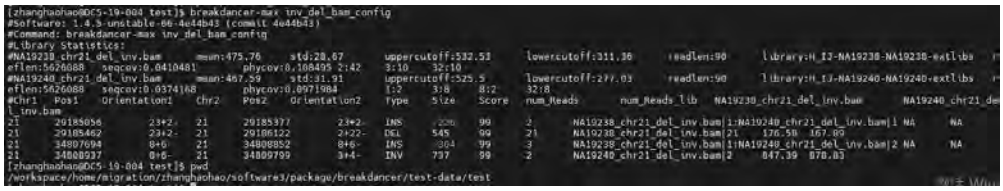
4.156.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd breakdancerd/test-data
./breakdancer_max inv_del_bam_config
```



----End

4.157 Mugsy 1.2.3 Porting Guide (Kylin V10)

4.157.1 Introduction

Mugsy is a multiple whole genome aligner. Mugsy uses Nucmer for pairwise alignment, a custom graph based segmentation procedure for identifying collinear regions.

For more information, visit the **Mugsy web page**.

Programming language: Perl

Brief description: A multiple whole genome aligner.

Open source license: custom open source license

Recommended Software Version

Mugsy 1.2.3

4.157.2 Environment Requirements

Hardware Requirements

Table 4-775 lists the hardware requirements.

Table 4-775 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table lists the software requirements.

Table 4-776 Software requirements

Item	Version	Download URL
Anaconda3	2021.05	https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2021.05-Linux-aarch64.sh

OS Requirements

Table lists the OS requirements.

Table 4-777 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.18.0 (4.14)	Included in the OS image

4.157.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Mugsy software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-778 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/Anaconda	Installation path of Anaconda	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.157.4 Configuring the Compilation Environment

4.157.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-779 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Anaconda	See **4.157.4.2 Installing Anaconda**.

4.157.4.2 Installing Anaconda

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Anaconda3.

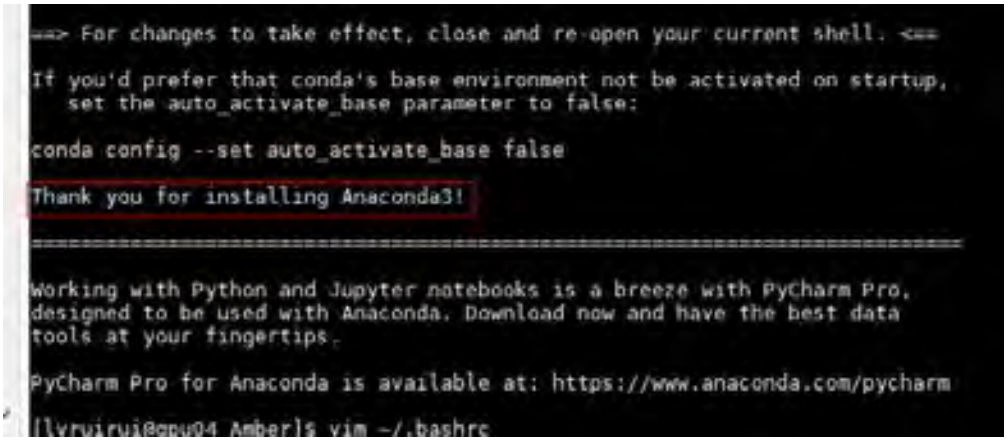
```
bash Anaconda3-2021.05-Linux-aarch64.sh
```

 Copyright © Huawei Technologies Co., Ltd.

Perform the installation as instructed.

NOTE

During the installation, press **Enter** and type **yes** when prompted. When a message is displayed stating "Do you wish the installer to initialize Anaconda3 by running conda init?", type **yes**. After the installation is complete, you will see the message "Thank you for installing Anaconda3!".



Step 3 Set the Anaconda3 environment variables.

```
source ~/.bashrc
```

----End

4.157.5 Obtaining the Source Code

Use **Anaconda** for an automatic installation.

4.157.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

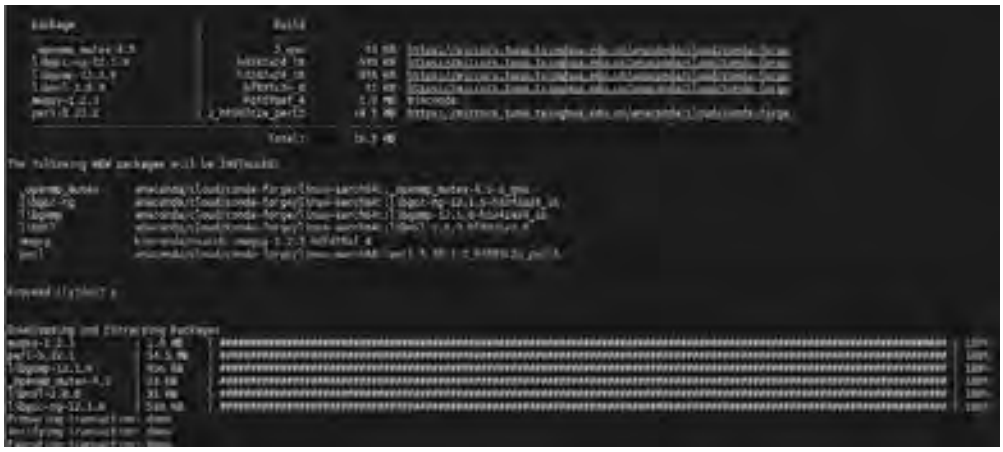
Step 2 Activate the Anaconda environment.

```
conda activate mugsy
```

Step 3 Perform the installation.

```
conda install -c bioconda mugsy
```

 Copyright © Huawei Technologies Co., Ltd.



----End

4.157.7 Running and Verification

Procedure

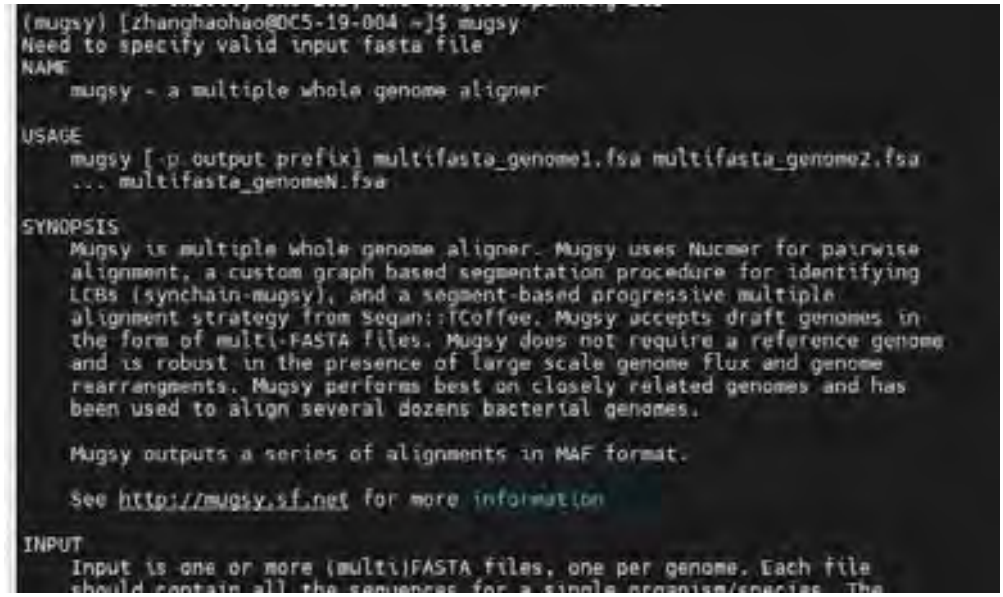
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Activate the Anaconda environment for Mugsy.

```
conda activate mugsy
```

Step 3 Perform the verification.

```
mugsy
```



----End

4.158 AlphaFold Porting Guide (CentOS 8.2)

4.158.1 Introduction

AlphaFold is an AI system developed by DeepMind that predicts a protein's 3D structure from its amino acid sequence. It regularly achieves accuracy competitive with experiment.

For more information, visit the **AlphaFold page at GitHub**.

Programming language: Python

Brief description: An AI system developed by DeepMind that predicts a protein's 3D structure from its amino acid sequence.

Open source license: Apache 2.0

Recommended Software Version

AlphaFold 2

4.158.2 Environment Requirements

Hardware Requirements

Table 4-780 lists the hardware requirements.

Table 4-780 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor
GPU	NVIDIA Tesla A100

Software Requirements

Table lists the software requirements.

Table 4-781 Software requirements

Item	Version	Download URL
Bazel	3.7.2	**https://github.com/bazelbuild/bazel/releases/download/0.26.0/bazel-0.26.0-dist.zip**
GCC	9.3.0	**https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/**
TensorFlow	2.4.3	**https://github.com/tensorflow/tensorflow/archive/v2.4.3.tar.gz**
NVIDIA V100 driver	11.2	**https://www.nvidia.cn/Download/index.aspx?lang=cn**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
NVIDIA CUDA	11.4	https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux_sbsa.run
jaxlib	0.17.0	https://github.com/google/jax/archive/refs/tags/jaxlib-v0.1.70.tar.gz
Python	3.8	https://www.python.org/downloads/
Miniconda	3	https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

OS Requirements

Table lists the OS requirements.

Table 4-782 OS requirements

Item	Version	Download URL
CentOS	8.2	https://www.centos.org/download/
Kernel	4.18.0-193.el8.aarch64	https://www.centos.org/download/

4.158.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AlphaFold software porting.

Table 4-783 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in HPC Solution Basic Environment Setup Guide.
2	/path/to/python	Installation path of Python	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual
4	/path/to/NVIDIA-CUDA	Installation path of NVIDIA CUDA.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
5	/path/to/tensorflow	Installation path of TensorFlow	paths planned during the installation process.
6	/path/to/bazel	Installation path of Bazel	
7	/path/to/jaxlib	Installation path of jaxlib	
8	/path/to/AlphaFold	Installation path of AlphaFold	
9	/path/to/CASE	Path for AlphaFold test cases	
10	/usr/local/cuda-11.4	Installation path of CUDA	

4.158.4 Configuring the Compilation Environment

4.158.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-784 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Conda	See **4.158.4.2 Installing Conda**.
3	Installing Bazel	See **4.158.4.3 Installing Bazel**.
4	Installing TensorFlow	See **4.158.4.4 Installing TensorFlow**.
5	Installing jaxlib	See **4.158.4.5 Installing jaxlib**.
6	Install AlphaFold dependencies	See **4.158.4.6 Install AlphaFold Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
7	Installing NVIDIA CUDA	See **4.158.4.7 Installing NVIDIA CUDA**.

4.158.4.2 Installing Conda

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download Conda.

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

Step 3 Install Conda.

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Step 4 Install the software as prompted. After the installation is complete, close the current shell window and open it again.

Step 5 After the installation is complete, check whether the installation is successful.

```
conda -version
```

If the correct version is displayed, the installation is successful.

----End

4.158.4.3 Installing Bazel

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the Bazel dependencies.

```
yum -y install build-essential openjdk-8-jdk python zip unzip gcc python-pip libhdf5-serial-dev git pkg-config
pip install numpy future keras_preprocessing mock Cython
```

Step 3 Decompress, install, and compile Bazel.

```
mkdir bazel
unzip bazel-0.26.0-dist.zip -d bazel
cd bazel
EXTRA_BAZEL_ARGS='--host_javabase=@local_jdk//:jdk' ./compile.sh
```

Step 4 Create a soft link.

```
ln -s ${bazel root directory}/output/bazel /usr/bin/bazel
```

Step 5 Perform the verification.

```
bazel --version
```

If the correct version is displayed, the installation is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.158.4.4 Installing TensorFlow

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the TenforFlow source code.

```
git clone https://github.com/tensorflow/tensorflow.git
```

Step 3 For the GPU version, you need to install the the CUDA Toolkit by yourself.

Step 4 Configure TensorFlow compilation options.

```
cd tensorflows-xxx
./configure
```

The configuration for the GPU version is as follows:

```
Extracting Bazel installation...
WARNING: --batch mode is deprecated. Please instead explicitly shut down your Bazel server using the command "bazel shutdown".
You have bazel 0.26.0- (@non-git) installed.
Please specify the location of python. [Default is /usr/bin/python]: [Enter]
Please input the desired Python library path to use.  Default is [/usr/local/lib/python2.7/dist-packages] [Enter]
Do you wish to build TensorFlow with XLA JIT support? [Y/n]: n
No XLA JIT support will be enabled for TensorFlow.

Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: n
No OpenCL SYCL support will be enabled for TensorFlow.

Do you wish to build TensorFlow with ROCm support? [y/N]: n
No ROCm support will be enabled for TensorFlow.

Do you wish to build TensorFlow with CUDA support? [y/N]: y
CUDA support will be enabled for TensorFlow.

Do you wish to build TensorFlow with TensorRT support? [y/N]: n
No TensorRT support will be enabled for TensorFlow.

Found CUDA 10.2 in:
  /usr/local/cuda/lib64
  /usr/local/cuda/include
Found cuDNN 7 in:
  /usr/local/cuda/lib64
  /usr/local/cuda/include
Please specify a list of comma-separated CUDA compute capabilities you want to build with.
You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.
```

 Copyright © Huawei Technologies Co., Ltd.

```
Please note that each additional compute capability significantly increases your build time and binary size, and that TensorFlow only supports
compute capabilities >= 3.5 [Default is: 3.5,7.0]: T4为7.5，V100为7

Do you want to use clang as CUDA compiler? [y/N]: n
nvcc will be used as CUDA compiler.
Please specify which gcc should be used by nvcc as the host compiler. [Default is /usr/bin/gcc]: [Enter]

Do you wish to build TensorFlow with MPI support? [y/N]: n
No MPI support will be enabled for TensorFlow.

Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native -Wno-sign-
compare]:  [Enter]
Would you like to interactively configure ./WORKSPACE for Android builds? [y/N]: n
Not configuring the WORKSPACE for Android builds.
```

Step 5 Create a **.whl** file.

```
bazel clean –expunge
```

Step 6 The compilation commands for the GPU version are as follows:

```
bazel build --config=opt --config=cuda //tensorflow/tools/pip_package:build_pip_package
mkdir tensorflow-pkg
bazel-bin/tensorflow/tools/pip_package/build_pip_package --gpu ./tensorflow-pkg
```

Step 7 The generated **.whl** file is stored in the **tensorflow_pkg** directory. Run the following command to install it:

```
pip install tensorflow-xxx.whl
```

----End

4.158.4.5 Installing jaxlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install jax.

```
pip install jax
```

Step 3 Download the jaxlib source code.

```
wget https://github.com/google/jax/archive/refs/tags/jaxlib-v0.1.70.tar.gz
```

Step 4 Install the jaxlib dependencies.

```
pip install numpy scipy cython six
```

Step 5 Configure the compiler.

```
export CC=gcc CXX=g++ FC=gfortran
```

Step 6 Compile the CUDA version.

```
python build/build.py --enable_cuda
pip install -e build  # installs jaxlib (includes XLA)
```

Step 7 Install **jaxlib.whl**.

```
pip install ./disk/jaxlib-0.17.0.whl
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.158.4.6 Install AlphaFold Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the AlphaFold dependencies.

```
conda install -y -c conda-forge openmm==7.5.1 cudnn==8.2.1.32 cudatoolkit==11.0.3 pdbfixer==1.7
conda install -y -c bioconda hmmer==3.3.2 hhsuite==3.3.0 kalign2==2.04
```

----End

4.158.4.7 Installing NVIDIA CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. In this case, perform **Step 4**.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **wq!**, and press **Enter** to save the file and exit.

Step 4 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo shcuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 5 Set the environment variables.

```
export PATH=/usr/local/cuda-11.4/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
```

Step 6 Perform the verification.

```
nvcc --version
```

If the command output contains the NVIDIA CUDA component version information, the installation is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.158.5 Obtaining the Source Code

Procedure

Step 1 Download the AlphaFold source code.

https://github.com/deepmind/alphafold

Step 2 Use an SFTP tool to upload the AlphaFold installation package to the */path/to/AlphaFold* directory on the server.

----End

4.158.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/AlphaFold
```

Step 3 Decompress the installation package.

```
tar -xvf alphaFold-xxx.tar.gz
```

NOTE

xxx indicates the software version.

Step 4 Go to the directory generated after the decompression.

```
cd alphafold
```

Step 5 Perform the installation.

```
pip3 install --upgrade pip
&& pip3 install -r ./requirements.txt
&& pip3 install --upgrade "jax[cuda111]" -f
https://storage.googleapis.com/jax-releases/jax_releases.html
&& pip3 install jaxlib==0.1.70+cuda111 -f
https://storage.googleapis.com/jax-releases/jax_releases.html
# work_path=/path/to/alphafold-code
work_path=$(PWD)
# update openmm
a=$(which python)
cd $(dirname $(dirname $a))/lib/python3.8/site-packages
patch -p0 < $work_path/docker/openmm.patch
```

Step 6 Download the database script.

```
wget https://github.com/kalininalab/alphafold_non_docker/blob/main/download_db.sh
```

Step 7 Download the **alphafold_data** file.

```
bash download_db.sh -d </home/johndoe/alphafold_data>
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.158.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

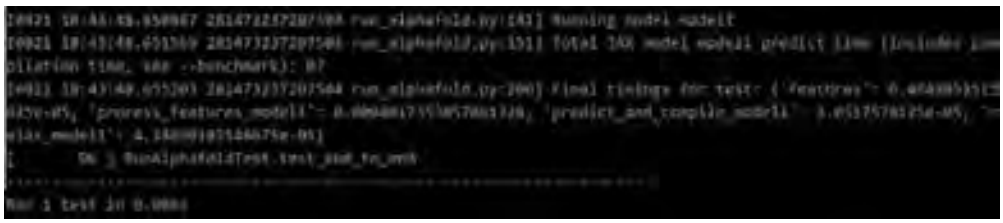
Step 2 Go to the test directory.

```
cd /path/to/AlphaFold
```

Step 3 Perform the test.

```
python run_alphafold_test.py
```

The following is an example of the test result:



----End

4.159 Glimmer v3.02 Porting Guide (Kylin V10)

4.159.1 Introduction

Glimmer is a system for finding genes in microbial DNA, especially the genomes of bacteria, archaea, and viruses. Glimmer uses interpolated Markov models (IMMs) to identify the coding regions and distinguish them from noncoding DNA.

For more information, visit the **Glimmer official website**.

Programming language: C++

Brief description: A system for finding genes in microbial DNA, especially the genomes of bacteria, archaea, and viruses.

Open source license: OSI

Recommended Software Version

Glimmer v3.02

4.159.2 Environment Requirements

Hardware Requirements

Table 4-785 lists the hardware requirements.

Table 4-785 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-786 Software requirements

Item	Version	Download URL
Glimmer	v3.02	http://ccb.jhu.edu/software/glimmer/glimmer302b.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

OS Requirements

Table 4-787 lists the OS requirements.

Table 4-787 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.159.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Glimmer software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-788 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/bisheng-2.1.0	Installation path of BiSheng Compiler.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/Glimmer	Installation path of Glimmer	
4	/path/to/CASE	Path for Glimmer test cases	

4.159.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-789 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.159.5 Obtaining the Source Code

Procedure

Step 1 Download the Glimmer installation package **glimmer302b.tar.gz**.

http://ccb.jhu.edu/software/glimmer/glimmer302b.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the CLUSTALW installation package to the */path/to/Glimmer* directory on the server.

----End

4.159.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/Glimmer
```

Step 3 Decompress the installation package.

```
tar -xvf glimmer302.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd glimmer3.02/src
```

Step 5 Perform the compilation and installation.

```
make
make install
```

Step 6 Check whether an executable file is generated.

```
ll /path/to/Glimmer/glimmer3.02/bin/
```

Step 7 Set the environment variable.

```
export PATH=/path/to/Glimmer/glimmer3.02/bin/:$PATH
```

----End

4.159.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run Glimmer3.

```
./glimmer3
```

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost bin]# ./glimmer3
Starting at Wed Jun  1 15:33:51 2022

USAGE:  glimmer3 [options] <sequence-file> <icm-file> <tag>

Read DNA sequences in <sequence-file> and predict genes
in them using the Interpolated Context Model in <icm-file>.
Output details go to file <tag>.detail and predictions go to
file <tag>.predict

Options:
 -A <codon-list>
 --start_codons <codon-list>
    Use comma-separated list of codons as start codons
    Sample format:  -A atg,gtg
    Use -P option to specify relative proportions of use.
    If -P not used, then proportions will be equal
 -b <filename>
 --rbs_pwm <filename>
    Read a position weight matrix (PWM) from <filename> to identify
    the ribosome binding site to help choose start sites
 -C <p>
 --gc_percent <p>
    Use <p> as GC percentage of independent model
    Note:  <p> should be a percentage, e.g., -C 45.2
 -E <filename>
 --entropy <filename>
    Read entropy profiles from <filename>.  Format is one header
    line, then 20 lines of 3 columns each.  Columns are amino acid,
    positive entropy, negative entropy.  Rows must be in order
    by amino acid code letter
```

The program is running properly.

----End

4.160 HaploView 4.1 Porting Guide (Kylin V10)

4.160.1 Introduction

HaploView is a Java based tool for use by biologists in the study of genetic haplotype data. It provides a quick, easy interface to many common tasks involved in such analyses.

For more information, visit the **HaploView official website**.

Programming language: Java

Brief description: HaploView is a Java based tool for use by biologists in the study of genetic haplotype data.

Open source license: MIT

Recommended Software Version

HaploView 4.1

 Copyright © Huawei Technologies Co., Ltd.

4.160.2 Environment Requirements

Hardware Requirements

Table 4-790 lists the hardware requirements.

Table 4-790 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-791 Software requirements

Item	Version	Download URL
HaploView	4.1	https://master.dl.sourceforge.net/project/haploview/haploview-source/4.1/Haploview4.1.tar.gz?viasf=1
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

OS Requirements

Table 4-792 lists the OS requirements.

Table 4-792 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.160.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HaploView software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-793 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/bisheng-2.1.0	Installation path of BiSheng Compiler.	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/Haploview	Installation path of HaploView	
4	/path/to/CASE	Path for HaploView test cases	

4.160.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-794 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.160.5 Obtaining the Source Code

Procedure

Step 1 Download the HaploView 4.1 JAR package **Haploview4.1.jar**.

https://sourceforge.net/projects/haploview/

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the HaploView 4.1 JAR package to the */path/to/Haploview* directory on a server.

----End

4.160.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the following directory. The JAR package does not need to be installed.

```
cd /path/to/Haploview
```

----End

4.160.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run HaploView4.

```
java -jar Haploview4.1.jar
```

The program is running properly.

----End

4.161 HH-suite v3.3.0 Porting Guide (Kylin V10)

4.161.1 Introduction

The HH-suite is an open-source software package for sensitive protein sequence searching based on the pairwise alignment of hidden Markov models (HMMs). It contains HHsearch and HHblits among other programs and utilities. HHsearch takes as input a multiple sequence alignment (MSA) or profile HMM and searches a database of HMMs (e.g. PDB, Pfam, or InterPro) for homologous proteins. HHsearch is often used for protein structure prediction to detect homologous templates and to build highly accurate query-template pairwise alignments for homology modeling.

For more information, visit the **HH-suite page at GitHub**.

Programming languages: C/C++/Perl

Brief description: A remote protein homology detection suite.

Open source license: GPL 3.0

Recommended Software Version

HH-suite v3.3.0

4.161.2 Environment Requirements

Hardware Requirements

Table 4-795 lists the hardware requirements.

Table 4-795 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 4-796 Software requirements

Item	Version	Download URL
HH-suite	v3.3.0	**https://github.com/soedinglab/hh-suite/archive/refs/tags/v3.3.0.tar.gz**
CMake	3.21.1	**https://github.com/Kitware/CMake/archive/refs/tags/v3.21.1.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-797 lists the OS requirements.

Table 4-797 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.161.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HH-suite software porting.

Table 4-798 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/HHSUITE	Installation path of HH-suite	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/CMAKE	Installation path of CMake	

4.161.4 Configuring the Compilation Environment

4.161.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-799 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
3	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.
4	Installing CMake	See **4.161.4.2 Installing CMake**.

4.161.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/CMAKE
wget https://github.com/Kitware/CMake/archive/refs/tags/v3.21.1.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf v3.21.1.tar.gz
```

Step 4 Go to the directory generated after decompression.

```
cd cmake-3.21.1
```

Step 5 Install the software.

```
./bootstrap --prefix=/path/to/CMAKE
make -j
make install
```

Step 6 Set the environment variables.

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.161.5 Obtaining the Source Code

Procedure

Step 1 Download the HH-suite installation package **v3.3.0.tar.gz**.

https://github.com/soedinglab/hh-suite/archive/refs/tags/v3.3.0.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the HH-suite installation package to the */path/to/HHSUITE* directory on the server.

----End

4.161.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/HHSUITE
```

Step 3 Decompress the installation package.

```
tar xvf v3.3.0.tar.gz
```

Step 4 Go to the directory generated after decompression.

```
cd hh-suite-3.3.0
```

Step 5 Create a build directory and go to it.

```
mkdir build && cd build
```

Step 6 Create a build file.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/opt/hhsuite -DHAVE_ARM8=ON
```

Step 7 Perform the compilation and installation.

```
make -j
make install
```

Step 8 Add the environment variable.

```
export PATH="/opt/hhsuite/bin:/opt/hhsuite/scripts:$PATH"
```

----End

4.161.7 Running and Verification

Procedure

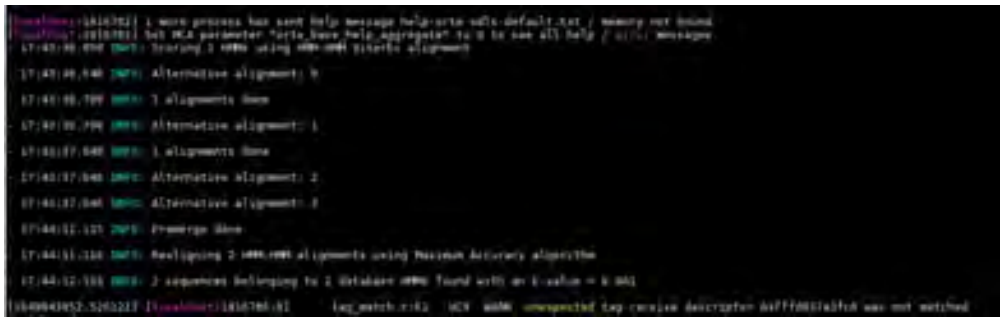
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
cd /path/to/HHSUITE/hh-suite-3.3.0/data
./test.sh
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

4.161.8 Troubleshooting

4.161.8.1 An Error Is Reported When Running MPI as the Root User

Symptom

When you run the **test.sh** script, the following error information is displayed:

```
--------------------------------------------------------------------------
mpirun has detected an attempt to run as root.

Running as root is *strongly* discouraged as any mistake (e.g., in
defining TMPDIR) or bug can result in catastrophic damage to the OS
file system, leaving your system in an unusable state.

We strongly suggest that you run mpirun as a non-root user.

You can override this protection by adding the --allow-run-as-root option
to the cmd line or by setting two environment variables in the following way:
the variable OMPI_ALLOW_RUN_AS_ROOT=1 to indicate the desire to override this
protection, and OMPI_ALLOW_RUN_AS_ROOT_CONFIRM=1 to confirm the choice and
add one more layer of certainty that you want to do so.
We reiterate our advice against doing so - please proceed at your own risk.
--------------------------------------------------------------------------
```

Key Process and Cause Analysis

To run MPI as the root user, add **--allow-run-as-root**.

Conclusion and Solution

Add **--allow-run-as-root** to **mpirun** in the script.

4.162 art-nextgen-simulation-tools 20160605 Porting Guide (Kylin V10)

4.162.1 Introduction

art-nextgen-simulation-tools is a set of simulation tools to generate synthetic next-generation sequencing reads. It simulates sequencing readings by simulating a real sequencing process using an empirical error model or a quality profile summarized from a large amount of recalibrated sequencing data.

 Copyright © Huawei Technologies Co., Ltd.

For more information, visit the **art-nextgen-simulation-tools website**.

Programming language: Perl

Brief description: A set of simulation tools to generate synthetic next-generation sequencing reads.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Version

20160605

4.162.2 Environment Requirements

Hardware Requirements

Table 4-800 lists the hardware requirements.

Table 4-800 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-801 lists the OS requirements.

Table 4-801 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-802 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-802 Software requirements

Item	Version	Download URL
art-nextgen-simulation-tools	20160605	https://mirror.tuna.tsinghua.edu.cn/raspbian/raspbian/pool/main/a/art-nextgen-simulation-tools/art-nextgen-simulation-tools_20160605%2Bdfsg.orig.tar.xz
GCC	7.3.0	It is built in the OS and does not need to be installed.

4.162.3 Paths for Software Porting

This chapter lists the software installation paths involved in the art-nextgen-simulation-tools software porting.

Table 4-803 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/art-nextgen-simulation-tools	Installation path of art-nextgen-simulation-tools.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.162.4 Configuring the Compilation Environment

4.162.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-804 Configuration process

No.	Configuration Item	Remarks
1	Installing dependencies	See **4.162.4.2 Installing Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

4.162.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y perl perl-devel perl-CPAN
yum install -y autoconf gsl gsl-devel
yum install -y gcc gcc-c++ gcc-gfortran
```

----End

4.162.5 Obtaining the Source Code

Procedure

Step 1 Download the art-nextgen-simulation-tools installation package.

https://mirror.tuna.tsinghua.edu.cn/raspbian/raspbian/pool/main/a/art-nextgen-simulation-tools/art-nextgen-simulation-tools_20160605%2Bdfsg.orig.tar.xz

Step 2 Use an SFTP too to upload the art-nextgen-simulation-tools installation package to the */path/to/art-nextgen-simulation-tools* directory on the server.

----End

4.162.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/art-nextgen-simulation-tools
tar -xvf art-nextgen-simulation-tools_20160605+dfsg.orig.tar.xz
cd art_src_MountRainier_Linux
```

Step 2 Create a configuration file.

```
autoreconf --install
```

Step 3 Create **Makefile**.

```
./configure --prefix=/path/to/art-nextgen-simulation-tools
```

Step 4 Perform the installation.

```
make
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.162.7 Running and Verification

Procedure

Step 1 Verify the **aln2bed.pl** test case.

```
cd /path/to/art-nextgen-simulation-tools/bin
./aln2bed.pl
```

If the following information is displayed, the verification is successful.

```
aln2bed converts ALN files to a BED file

USAGE: aln2bed out_bed_file.bed in_aln_file_1 [ in_aln_file_2 ...]
```

Step 2 Verify the **art_454** test case.

```
./art_454
```

If the following information is displayed, the verification is successful.

Step 3 Verify the **art_illumina** test case.

```
./art_illumina
```

If the following information is displayed, the verification is successful.

----End

4.162.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

4.163 bioconductor-appreci8r 1.14.0 Porting Guide (Kylin V10)

4.163.1 Introduction

appreci8R is an R/Bioconductor package for filtering SNVs and short indels.

appreci8R is an R version of the appreci8-algorithm - a pipeline for PREcise variant Calling Integrating 8 tools.

For more information, visit the **official website**.

Programming language: R

Brief description: An R/Bioconductor package for filtering SNVs and short indels.

Open source license: LGPL 3

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.14.0

4.163.2 Environment Requirements

Hardware Requirements

Table 4-805 lists the hardware requirements.

Table 4-805 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-806 lists the software requirements.

Table 4-806 Software requirements

Item	Version	Download URL
Appreci8r	1.14.0	**http://bioconductor.org/packages/release/bioc/src/contrib/appreci8R_1.14.0.tar.gz**

OS Requirements

Table 4-807 lists the OS requirements.

Table 4-807 OS requirements

Item	Version	Download URL
Kyin	v10	**https://www.centos.org/download/**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.163.3 Paths for Software Porting

This chapter lists the software installation paths involved in the appreci8R software porting.

Table 4-808 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/Appreci8r	Installation path of appreci8R.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.163.4 Configuring the Compilation Environment

4.163.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-809 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.163.4.2 Installing R 4.2.0**.

 Copyright © Huawei Technologies Co., Ltd.

4.163.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/Appreci8r
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the package and perform the installation.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/Appreci8r/R_install
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/Appreci8r/R_install/bin:$PATH
export LB_LIBRARY_PATH=/path/to/APPRECI8R/R_install/lib/R/lib:$ LB_LIBRARY_PATH
```

Step 4 (Optional) Add an R language mirror source if needed.

```
vim ~/.Rprofile
```

Step 5 Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.163.5 Obtaining the Source Code

Procedure

Step 1 Use the R language to install the software. You do not need to download the source package.

Step 2 (Optional for source code installation) Download the appreci8R installation package **appreci8R_1.14.0.tar.gz**.

http://bioconductor.org/packages/release/bioc/src/contrib/appreci8R_1.14.0.tar.gz

Step 3 (Optional for source code installation) Use an SFTP tool to upload the appreci8R installation package to the */path/to/APPRECI8R* directory on the server.

----End

4.163.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the R CLI.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Step 3 Install BiocManager.

```
install.packages("BiocManager")
```

If **DONE (BiocManager)** is displayed in the command output, the installation is successful.

```
> install.packages("BiocManager")
trying URL 'http://mirrors.tuna.tsinghua.edu.cn/CRAN/src/contrib/BiocManager_1.30.18.tar.gz'
Content type 'application/octet-stream' length 289602 bytes (282 KB)
==================================================
downloaded 282 KB

* installing *source* package 'BiocManager' ...
** package 'BiocManager' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package can be loaded from final location
Warning in file(con, "r") :
  URL 'https://bioconductor.org/config.yaml': status was 'SSL peer certificate or SSH remote key was
** testing if installed package keeps a record of temporary installation path
* DONE (BiocManager)

The downloaded source packages are in
        '/tmp/RtmpA5s3e/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Step 4 Install appreci8R.

```
BiocManager::install("appreci8R")
```

If **DONE (appreci8R)** is displayed in the command output, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.

----End

4.163.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("appreci8R")
```

If **appreci8R** is displayed in the command output and the R package version is 1.14.0, the verification is successful.

----End

4.163.8 Troubleshooting

4.163.8.1 Failed to Download the Installation Package of the Dependency Software

This case is an example and is for reference only. The actual situation prevails.

Symptom

An error occurs during compilation. The error message is as follows:

Key Process and Cause Analysis

The installation package is too large. As a result, the download fails.

Conclusion and Solution

Step 1 Copy the download link and manually download the installation package.

```
wget http://mirrors.tuna.tsinghua.edu.cn/bioconductor//packages/3.15/data/annotation/src/contrib/BSgenome.Hsapiens.UCSC.hg19_1.4.3.tar.gz
```

Step 2 Install the dependency.

```
R CMD INSTALL BSgenome.Hsapiens.UCSC.hg19
```

----End

4.163.9 Change History

Table 4-810

Date	Description
2023-04-27	This issue is the first official release.

4.164 ANGSD 0.938 Porting Guide (Kylin V10)

4.164.1 Introduction

ANGSD is a piece of software for analyzing next generation sequencing data. The software can handle a number of different input types from mapped reads to imputed genotype probabilities. Most methods take genotype uncertainty into account instead of basing the analysis on called genotypes. This is especially useful for low and medium depth data.

For more information, visit the **ANGSD page at GitHub**.

Programming language: C++

 Copyright © Huawei Technologies Co., Ltd.

Brief description: A piece of software for analyzing next generation sequencing data.

Open source license: custom open source license

Recommended Software Version

0.938

4.164.2 Environment Requirements

Hardware Requirements

Table 4-811 lists the hardware requirements.

Table 4-811 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-812 lists the software requirements.

Table 4-812 Software requirements

Item	Version	Download URL
htslib	1.16	https://github.com/samtools/htslib/releases/download/1.16/htslib-1.16.tar.bz2
ANGSD	0.938	https://github.com/ANGSD/angsd/archive/refs/heads/master.zip

OS Requirements

Table 4-813 lists the OS requirements.

Table 4-813 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.164.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ANGSD software porting.

Table 4-814 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/angsd	Installation path of ANGSD.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.164.4 Configuring the Compilation Environment

4.164.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-815 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing HTSlib	See **4.164.4.2 Installing HTSlib 1.16**.

4.164.4.2 Installing HTSlib 1.16

Procedure

Step 1 Download the HTSlib source package.

```
cd /path/to/angsd/htslib
wget https://github.com/samtools/htslib/releases/download/1.16/htslib-1.16.tar.bz2
```

Step 2 Decompress the installation package.

```
tar -xvf htslib-1.16.tar.bz2
cd htslib-1.16
```

Step 3 Perform the compilation and installation.

```
./configure
make -j && make install
```

----End

4.164.5 Obtaining the Source Code

Procedure

Step 1 Download the installation package **angsd0.938.tar.gz**.

http://popgen.dk/software/download/angsd/angsd0.938.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/angsd* directory on the server.

----End

4.164.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory and decompress the installation package.

```
cd /path/to/angsd
tar -xvf angsd0.938.tar.gz
cd angsd
```

Step 3 Perform the compilation and installation.

```
make -j HTSSRC=../htslib-1.16
```

----End

4.164.7 Running and Verification

Procedure

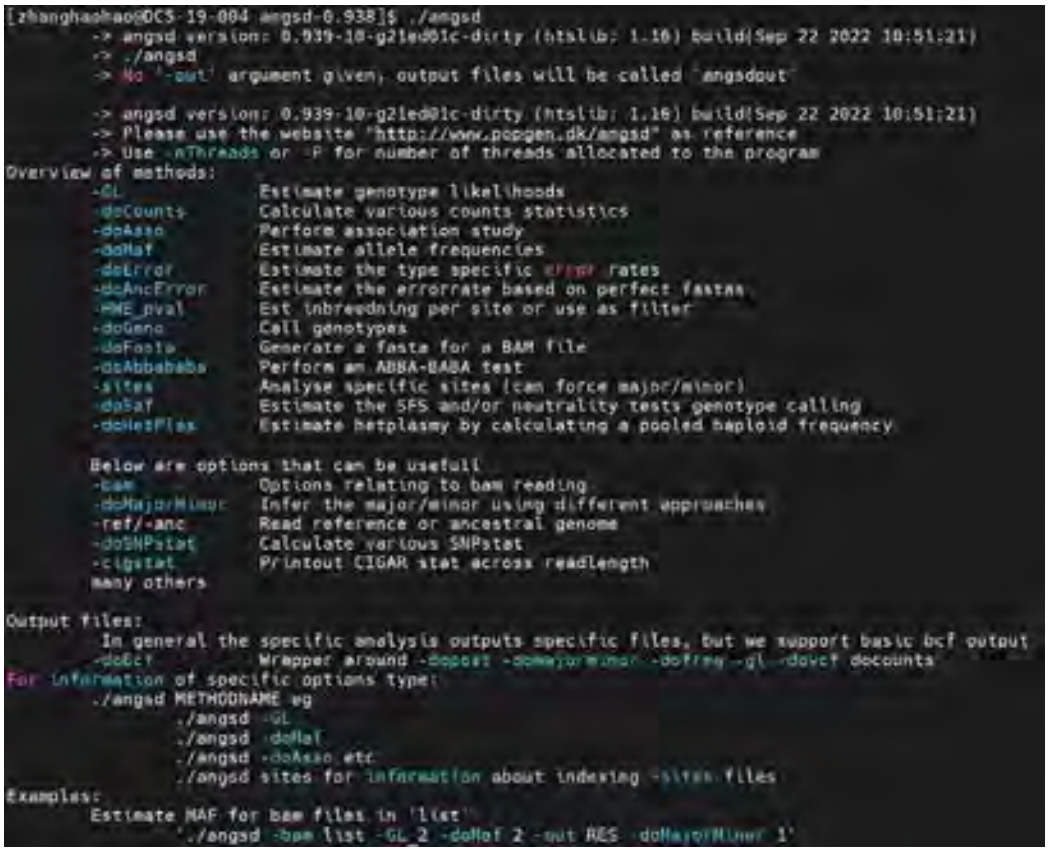
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 View the ANGSD parameters.

```
cd /path/to/angsd
./angsd
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.164.8 Change History

Table 4-816

Date	Description
2023-04-27	This issue is the first official release.

4.165 bioconductor-decontam 1.16.0 Porting Guide (Kylin V10)

4.165.1 Introduction

The decontam package provides simple statistical methods to identify and visualize contaminating DNA features. It provides a simple interface that takes in your table of sequence features (ASVs, OTUs, MAGs, genera, etc), and classifies each sequence feature as a contaminant based on signatures of contamination that have been demonstrated over many previous studies.

For more information, visit the **decontam website**.

Programming language: R

Brief description: Provides simple statistical methods to identify and visualize contaminating DNA features.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.16.0

4.165.2 Environment Requirements

Hardware Requirements

Table 4-817 lists the hardware requirements.

Table 4-817 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-818 lists the software requirements.

Table 4-818 Software requirements

Item	Version	Download URL
decontam	1.16.0	http://bioconductor.org/packages/release/bioc/src/contrib/decontam_1.16.0.tar.gz

OS Requirements

Table 4-819 lists the OS requirements.

Table 4-819 OS requirements

Item	Version	Download URL
Kyin	v10	https://www.centos.org/download/
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.165.3 Paths for Software Porting

This chapter lists the software installation paths involved in the decontam software porting.

Table 4-820 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/DECONTAM	Installation path of decontam.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.165.4 Configuring the Compilation Environment

4.165.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-821 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.165.4.2 Installing R 4.2.0**.

4.165.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/DECONTAM
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
```

 Copyright © Huawei Technologies Co., Ltd.

```
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/DECONTAM/R_install
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/DECONTAM/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/DECONTAM/R_install/lib/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add an R language mirror source if needed.

```
vim ~/.Rprofile
```

Step 5 Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.165.5 Obtaining the Source Code

Procedure

Step 1 Use the R language to install the software. You do not need to download the source package.

Step 2 (Optional for source code installation) Download the decontam installation package **decontam_1.16.0.tar.gz**.

http://bioconductor.org/packages/release/bioc/src/contrib/decontam_1.16.0.tar.gz

Step 3 (Optional for source code installation) Use an SFTP tool to upload the decontam installation package to the */path/to/DECONTAM* directory on the server.

----End

4.165.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the R CLI.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

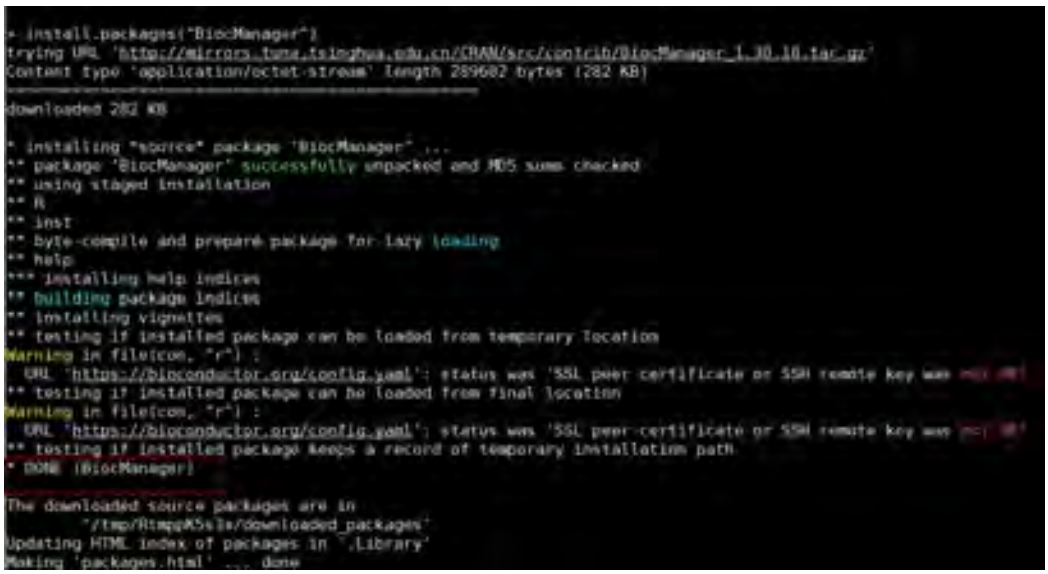
 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Step 3 Perform the verification.

```
install.packages("BiocManager")
```

If **DONE (BiocManager)** is displayed in the command output, the installation is successful.



Step 4 Install decontam.

```
BiocManager::install("decontam")
```

If **DONE (decontam)** is displayed in the command output, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.

----End

4.165.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("decontam")
```

If **decontam** is displayed in the command output, the verification is successful.

----End

4.165.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

4.166 bioconductor-clippda 1.46.0 Porting Guide (Kylin V10)

4.166.1 Introduction

bioconductor-clippda provides methods for analyzing data from clinical proteomic profiling studies. The focus is on the studies of human subjects, which are often observational case-control by design and have technical replicates. A method for sample size determination for planning these studies is proposed. It incorporates routines for adjusting for the expected heterogeneities and imbalances in the data and the within-sample replicate correlations.

For more information, visit the **official website**.

Programming language: R

Brief description: Methods for analyzing data from clinical proteomic profiling studies.

Open source license: GPL (≥ 2)

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.46.0

4.166.2 Environment Requirements

Hardware Requirements

Table 4-822 lists the hardware requirements.

Table 4-822 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-823 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-823 Software requirements

Item	Version	Download URL
R	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-824 lists the OS requirements.

Table 4-824 OS requirements

Item	Version	Download URL
Kyin	v10	https://www.centos.org/download/
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.166.3 Paths for Software Porting

This chapter lists the software installation paths involved in the clippda software porting.

Table 4-825 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/R	Installation path of R.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.166.4 Configuring the Compilation Environment

4.166.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

The server must be connected to the Internet.

Configuration Process

Table 4-826 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.166.4.2 Installing R 4.2.0**.

4.166.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/R
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the package.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/R
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/R/bin:$PATH
export LD_LIBRARY_PATH=/path/to/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add an R language mirror source if needed.

```
vim ~/.Rprofile
```

Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.166.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the R CLI.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Step 3 Install the dependency rgl.

```
install.packages("rgl", INSTALL_opts="--no-test-load")
```

If **DONE (rgl)** is displayed in the command output, the installation is successful.

```
** R
** demo
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
*** copying figures
** building package indices
** installing vignettes
** checking absolute paths in shared objects and dynamic libraries
* DONE (rgl)

The downloaded source packages are in
        '/tmp/RtmpeNDj4M/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Step 4 Add the following two lines to the **~/.Rprofile** file:

```
vim ~/.Rprofile
```

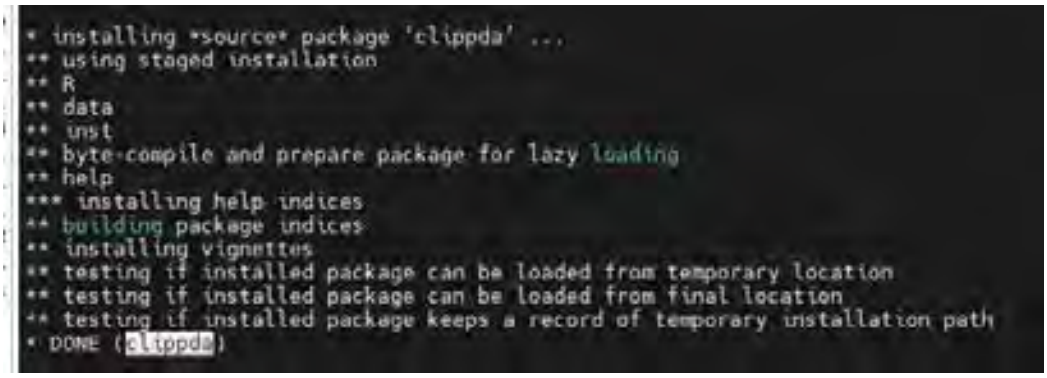
```
options(rgl.useNULL = TRUE)
options(rgl.debug = TRUE)
~
```

Step 5 Install clippda.

```
BiocManager::install("clippda")
```

If **DONE (clippda)** is displayed in the command output, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.166.6 Running and Verification

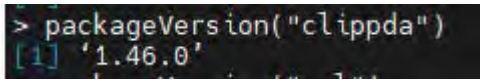
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
package.Version("clippda")
```

If the version **1.46.0** in displayed in the command output, the verification is successful.



----End

4.166.7 Change History

Table 4-827

Date	Description
2023-04-27	This issue is the first official release.

4.167 bioconductor-bubbletree 2.26.0 Porting Guide (Kylin V10)

4.167.1 Introduction

bioconductor-bubbletree is an intuitive visualization method used to elucidate tumoral aneuploidy and clonality in somatic mosaicism using next generation sequencing data.

For more information, visit the **bioconductor-bubbletree website**.

Programming language: R

Brief description: Sequencing data analysis.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.26.0

4.167.2 Environment Requirements

Hardware Requirements

Table 4-828 lists the hardware requirements.

Table 4-828 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-829 lists the OS requirements.

Table 4-829 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-830 lists the software requirements.

Table 4-830 Software requirements

Item	Version	Download URL
bioconductor-bubbletree	2.26.0	Install it using the R language. You do not need to download the source package.

 Copyright © Huawei Technologies Co., Ltd.

4.167.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bioconductor-bubbletree software porting.

Table 4-831 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bioconductor-bubbletree	Installation path of bioconductor-bubbletree.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.167.4 Configuring the Compilation Environment

4.167.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-832 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.167.4.2 Installing the R Language**.

4.167.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/bioconductor-bubbletree
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/bioconductor-bubbletree/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/bioconductor-bubbletree/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bioconductor-bubbletree/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirror source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

----End

4.167.5 Obtaining the Source Code

Use the R language to install bioconductor-bubbletree. You do not need to download the source package.

4.167.6 Compilation and Installation

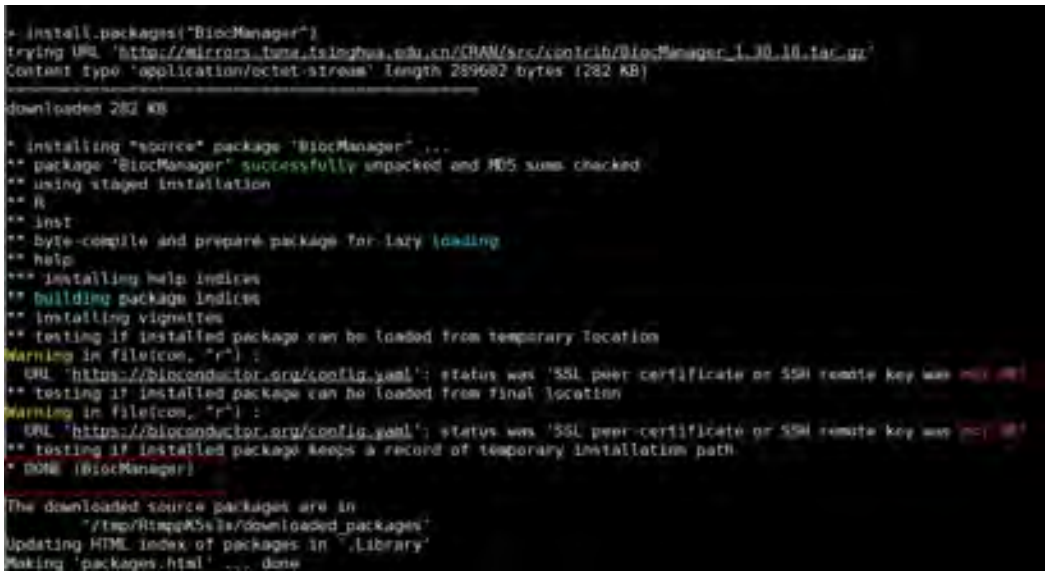
Procedure

Step 1 Install the dependencies.

```
R
install.packages("BiocManager")
```

If **DONE (BiocManager)** is displayed in the command output, the installation is successful.

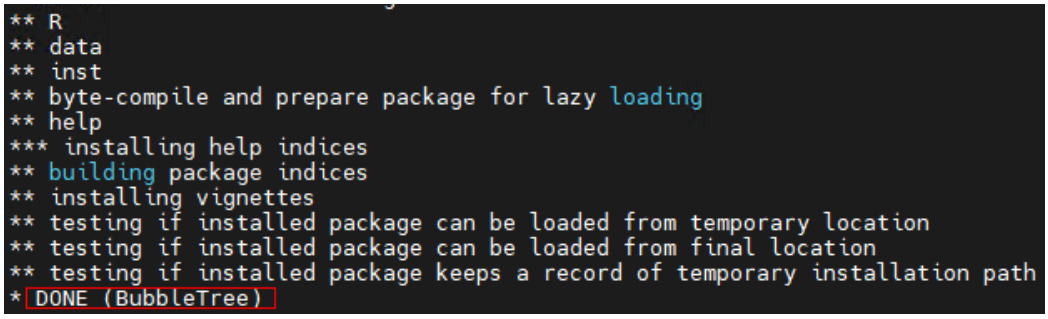
 Copyright © Huawei Technologies Co., Ltd.



Step 2 Install bioconductor-bubbletree.

```
BiocManager::install("BubbleTree")
```

If **DONE (BubbleTree)** is displayed in the command output, the installation is successful.



----End

4.167.7 Running and Verification

Procedure

Perform the verification.

```
R
citation("BubbleTree")
```

Command output:

If **BubbleTree** is displayed in the command output and the R package version is 2.26.0, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.

4.167.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

4.168 AMAS 1.2 Porting Guide (Kylin V10)

4.168.1 Introduction

AMAS works on amino acid and nucleotide alignments and combines capabilities of sequence manipulation with a function that calculates basic statistics.

For more information, visit the **AMAS page at GitHub**.

Programming language: Python

Brief description: A fast tool for alignment manipulation and computing of summary statistics.

Open source license: GNU General Public License

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.2

4.168.2 Environment Requirements

Hardware Requirements

Table 4-833 lists the hardware requirements.

Table 4-833 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-834 lists the software requirements.

Table 4-834 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz
Python	3.7.5	https://www.python.org/downloads/
AMAS	1.2	https://github.com/marekborowiec/AMAS.git

OS Requirements

Table 4-835 lists the OS requirements.

Table 4-835 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.168.3 Paths for Software Porting

This chapter lists the software installation paths involved in the AMAS software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-836 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/AMAS	Installation path of AMAS.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.168.4 Obtaining the Source Code

Procedure

Step 1 Download the AMAS installation package **AMAS**.

Step 2 Obtain the AMAS source code.

```
cd /path/to/amas
git clone https://github.com/marekborowiec/AMAS.git
```

----End

4.168.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 (Optional) Configure the virtual environment if you do not want to use system dependencies.

```
pip install virtualenv
virtualenv AMAS
source AMAS/bin/activate
```

Step 3 Go to the installation directory.

```
cd /path/to/AMAS/AMAS
```

Step 4 Install AMAS.

```
python3 setup.py install
```

----End

4.168.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the AMAS directory.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/AMAS/AMAS/amas
```

Step 3 Run the test case. If no error information is displayed, the verification is successful.

```
python3 AMAS.py concat -f fasta -d dna -i ../tests/fasta1.fas -u nexus
```

----End

4.168.7 Change History

Date	Description
2022-2-17	This issue is the first official release.

4.169 bioconductor-ebcoexpress 1.42.0 Porting Guide (Kylin V10)

4.169.1 Introduction

bioconductor-ebcoexpress is used for differential co-expression analysis.

For more information, visit the **bioconductor-ebcoexpress website**.

Programming language: R

Brief description: Used for differential co-expression analysis.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Version

1.42.0

4.169.2 Environment Requirements

Hardware Requirements

Table 4-837 lists the hardware requirements.

Table 4-837 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-838 lists the OS requirements.

Table 4-838 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-839 lists the software requirements.

Table 4-839 Software requirements

Item	Version	Download URL
bioconductor-ebcoexpress	1.42.0	Install it using the R language. You do not need to download the source package.

4.169.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bioconductor-ebcoexpress software porting.

Table 4-840 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bioconductor-ebcoexpress	Installation path of bioconductor-ebcoexpress.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.169.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.169.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-841 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.169.4.2 Installing the R Language**.

4.169.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/bioconductor-ebcoexpress
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/bioconductor-ebcoexpress/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/bioconductor-ebcoexpress/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bioconductor-ebcoexpress/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirror source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.169.5 Obtaining the Source Code

Use the R language to install bioconductor-ebcoexpress. You do not need to download the source package.

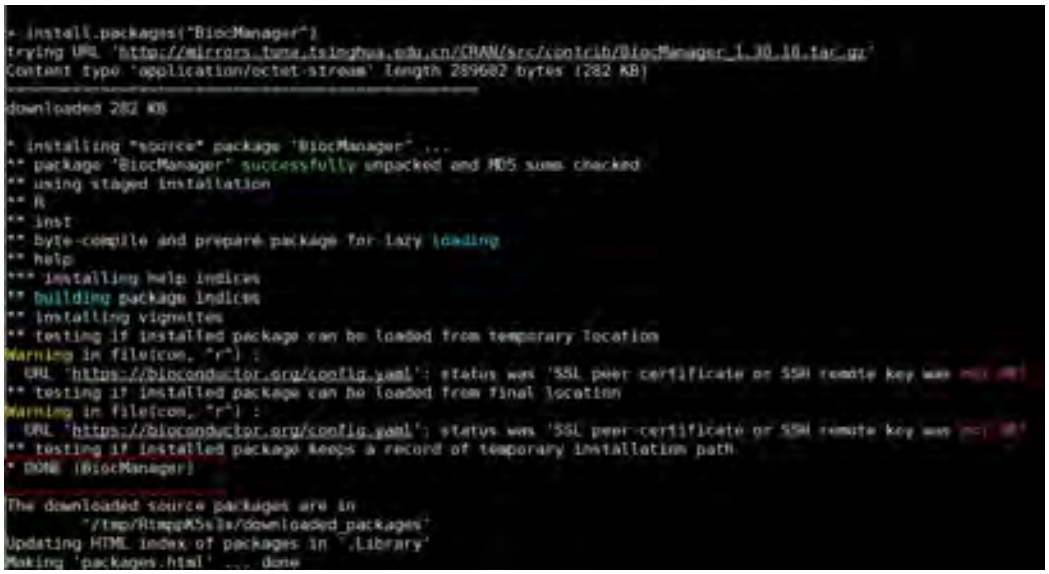
4.169.6 Compilation and Installation

Procedure

Step 1 Install the dependencies.

```
R
install.packages("BiocManager")
```

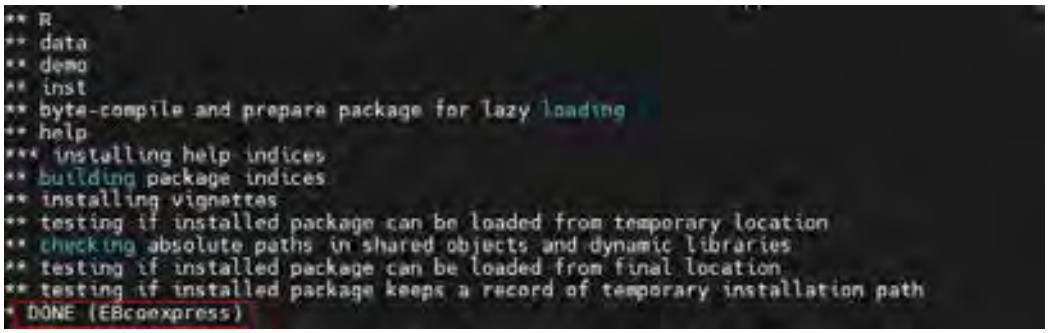
If **DONE (BiocManager)** is displayed in the command output, the installation is successful.



Step 2 Install bioconductor-ebcoexpress.

```
BiocManager::install("EBcoexpress")
```

If **DONE (EBcoexpress)** is displayed in the command output, the installation is successful.



----End

 Copyright © Huawei Technologies Co., Ltd.

4.169.7 Running and Verification

Procedure

Perform the verification.

```
R
citation("EBcoexpress")
```

If **EBcoexpress** is displayed in the command output and the R package version is 1.42.0, the verification is successful.

```
> citation("EBcoexpress")

To cite package 'EBcoexpress' in publications use:

  Dawson JA (2022). _EBcoexpress: EBcoexpress for Differential
  Co-Expression Analysis_. R package version 1.42.0.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {EBcoexpress: EBcoexpress for Differential Co-Expression Analysis},
    author = {John A. Dawson},
    year = {2022},
    note = {R package version 1.42.0},
  }
```

4.169.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

4.170 athena_meta 1.3 Porting Guide (Kylin V10)

4.170.1 Introduction

athena_meta is a read cloud assembler for metagenomes.

For more information, visit the **athena_meta page at GitHub**.

Programming language: Python

Brief description: A read cloud assembler for metagenomes.

Open source license: MIT License

Recommended Software Version

1.3

4.170.2 Environment Requirements

Hardware Requirements

Table 4-842 lists the hardware requirements.

Table 4-842 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-843 lists the software requirements.

Table 4-843 Software requirements

Item	Version	Download URL
athena_meta	1.3	https://github.com/abishara/athena_meta/archive/refs/heads/main.zip

OS Requirements

Table 4-844 lists the OS requirements.

Table 4-844 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.170.3 Paths for Software Porting

This chapter lists the software installation paths involved in the athena_meta software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-845 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/athena_meta	Installation path of athena_meta.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/athena_meta/dependences	Installation path of the dependencies for athena_meta.	

4.170.4 Configuring the Compilation Environment

4.170.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-846 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing Python 2.7	athena_meta depends on Python 2.7 and is incompatible with Python 3.x. For details, see **4.170.4.2 Installing Python 2.7**.
3	Installing IDBA 1.1.3a1	See **4.170.4.3 Installing IDBA 1.1.3a1**.
4	Installing Samtools 1.16.1	See **4.170.4.4 Installing Samtools 1.16.1**.
5	Installing HTSlib 1.16	See **4.170.4.5 Installing HTSlib 1.16**.
6	Installing minimap2	See **4.170.4.6 Installing minimap2**.
7	Installing BWA 0.7.17	See **4.170.4.7 Installing BWA 0.7.17**.
8	Installing Flye 2.3.1	See **4.170.4.8 Installing Flye 2.3.1**.

 Copyright © Huawei Technologies Co., Ltd.

4.170.4.2 Installing Python 2.7

Procedure

Step 1 Download the Python language source package.

```
cd /path/to/athena_meta/dependences
wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tar.xz
```

Step 2 Decompress the installation package.

```
tar -xvf Python-2.7.18.tar.xz
cd python-2.7.18
```

Step 3 Perform the compilation and installation.

```
./configure --prefix=/path/to/athena_meta/dependences/python-2.7.18
make -j && make -install
```

Step 4 Add the environment variable.

```
vi ~/.bashrc
```

Add the following content to the file:

```
export PATH=/path/to/athena_meta/dependences/python-2.7.18/bin:$PATH
```

Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Make the modification to take effect.

```
source ~/.bashrc
```

Step 6 View the Python version.

```
python -V
```

----End

4.170.4.3 Installing IDBA 1.1.3a1

Procedure

Step 1 Download the IDBA source package.

```
cd /path/to/athena_meta/dependences
wget https://github.com/abishara/idba/archive/refs/tags/1.1.3a1.tar.gz
```

Step 2 Decompress the installation package.

```
tar -xvf 1.1.3a1.tar.gz
cd idba-1.1.3a1/
```

Step 3 Execute **build.sh** to perform the installation.

```
./build.sh
```

----End

4.170.4.4 Installing Samtools 1.16.1

Procedure

Step 1 Download the Samtools source package.

```
cd /path/to/athena_meta/dependences
wget https://github.com/samtools/samtools/releases/download/1.16.1/samtools-1.16.1.tar.bz2
```

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Decompress the source package and go to the directory.

```
tar -xvf samtools-1.16.1.tar.bz2
cd samtools-1.16.1
```

Step 3 Perform the compilation and installation.

```
make -j && make install
```

----End

4.170.4.5 Installing HTSlib 1.16

Procedure

Step 1 Download the HTSlib source package.

```
cd /path/to/athena_meta/dependences
wget https://github.com/samtools/htslib/releases/download/1.16/htslib-1.16.tar.bz2
```

Step 2 Decompress the installation package.

```
tar -xvf htslib-1.16.tar.bz2
cd htslib-1.16
```

Step 3 Perform the compilation and installation.

```
make -j && make install
```

----End

4.170.4.6 Installing minimap2

Procedure

Step 1 Download the minimap2 source package.

```
cd /path/to/athena_meta/dependences
git clone https://github.com/lh3/minimap2
```

Step 2 Perform the compilation and installation.

```
cd minimap2
make arm_neno=1 aarch64=1
```

----End

4.170.4.7 Installing BWA 0.7.17

Procedure

Step 1 Download the BWA source package.

```
cd /path/to/athena_meta/dependences
git clone https://github.com/lh3/bwa.git
```

Step 2 Modify the configuration.

```
cd bwa
wget https://raw.githubusercontent.com/DLTcollab/sse2neon/master/sse2neon.h
vi ksw.c
```

Modify line 30.

```
#include <emmintrin.h>
```

After the modification:

```
#include "sse2neon.h"
```

Step 3 Perform the compilation and installation.

```
make -j
```

Step 4 Change the directory name.

```
cd ..
mv bwa/ bwa-0.7.17/
```

----End

4.170.4.8 Installing Flye 2.3.1

Procedure

Step 1 Download the Flye source package.

```
cd /path/to/athena_meta/dependences
git clone https://github.com/fenderglass/Flye
```

Step 2 Perform the installation.

```
cd flye
python setup.py install
```

Step 3 Change the directory name.

```
cd ..
mv flye/ flye-2.3.1/
```

----End

4.170.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the source code.

```
cd /path/to/athena_meta/
git clone https://github.com/abishara/athena_meta.git
```

----End

4.170.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the Python virtual environment.

```
cd /path/to/athena_meta/
python -m pip install virtualenv
python -m virtualenv .env
source .env/bin/activate
```

Step 3 Add the environment variable.

```
vi ~/.bashrc
```

Add the following content to the opened file:

 Copyright © Huawei Technologies Co., Ltd.

```
export ATHENA_HOME=/path/to/athena_meta/dependences
export PATH=$ATHENA_HOME/idba-1.1.3a1/bin:$PATH
export PATH=$ATHENA_HOME/samtools-1.16.1/bin:$PATH
export PATH=$ATHENA_HOME/htslib-1.16/bin:$PATH
export PATH=$ATHENA_HOME/minimap2:$PATH
export PATH=$ATHENA_HOME/bwa-0.7.17:$PATH
export PATH=$ATHENA_HOME/flye-2.31/bin:$PATH
```

Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Make the modification to take effect.

```
source ~/.bashrc
```

Step 5 Perform the compilation and installation.

```
cd athena_meta
pip install .
```

----End

4.170.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check the **athena_meta** parameter.

```
cd /path/to/athena_meta/
source .env/bin/activate
athena_meta
```

If the following information is displayed, the verification is successful.

```
(.env) [zhanghaohao@DC5-19-004 dependences]$ athena-meta
error: one of either --config, --check_prereqs, or --test must be specified

usage: athena-meta [-h] [--config CONFIG] [--check_prereqs] [--test]
                   [--force_reads] [--threads THREADS]

optional arguments:
  -h, --help          show this help message and exit
  --config CONFIG     input JSON config file for run, NOTE:
                      dirname(config.json) specifies root output directory
  --check_prereqs     test if external deps visible in environment
  --test              run tiny assembly test to check setup and prereqs
  --force_reads       proceed with subassembly even if input *bam and *fastq do
                      not pass QC
  --threads THREADS   number of multiprocessing threads
```

Step 3 Verify athena_meta.

```
athena_meta --check_prereqs
```

If **ok** is displayed in the command output, the verification is successful.

```
(.env) [zhanghaohao@DC5-19-004 dependences]$ athena-meta --check_prereqs
flye [ok]
bwa [ok]
samtools [ok]
idba_subasm [ok]
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.170.8 Change History

Table 4-847

Date	Description
2023-04-27	This issue is the first official release.

4.171 bioconductor-GARS 1.16.0 Porting Guide (Kylin V10)

4.171.1 Introduction

bioconductor-GARS provides methods for analyzing data from clinical proteomic profiling studies. The focus is on the studies of human subjects, which are often observational case-control by design and have technical replicates. A method for sample size determination for planning these studies is proposed. It incorporates routines for adjusting for the expected heterogeneities and imbalances in the data and the within-sample replicate correlations.

For more information, visit the **bioconductor-GARS website**.

Brief description: Methods for analyzing data from clinical proteomic profiling studies.

Programming language: R

Open source license: GPL (≥ 2)

NOTE

When using open source software, comply with the applicable licenses.

1.16.0

4.171.2 Environment Requirements

Hardware Requirements

Table 4-848 lists the hardware requirements.

Table 4-848 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-849 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-849 Software requirements

Item	Version	Download URL
R	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-850 lists the OS requirements.

Table 4-850 OS requirements

Item	Version	Download URL
Kylin	v10	https://www.centos.org/download/
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.171.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GARS software porting.

Table 4-851 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/R	Installation path of R.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.171.4 Configuring the Compilation Environment

4.171.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

The server must be connected to the Internet.

Configuration Process

Table 4-852 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.171.4.2 Installing R 4.2.0**.

4.171.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/R
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the package and perform the installation.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/R
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/R/bin:$PATH
export LD_LIBRARY_PATH=/path/to/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add an R language mirror source if needed.

```
vim ~/.Rprofile
```

Step 5 Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.171.5 Obtaining the Source Code

Procedure

Step 1 Use the R language to install the software. You do not need to download the source package.

----End

4.171.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Enter the R command.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Step 3 Install rgl.

```
install.packages("rgl", INSTALL_opts="--no-test-load")
```

If **DONE (rgl)** is displayed in the command output, the installation is successful.

```
** R
** demo
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
*** copying figures
** building package indices
** installing vignettes
** checking absolute paths in shared objects and dynamic libraries
* DONE (rgl)

The downloaded source packages are in
        '/tmp/RtmpeNDj4M/downloaded_packages'
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Step 4 Add the following two lines to the **~/.Rprofile** file:

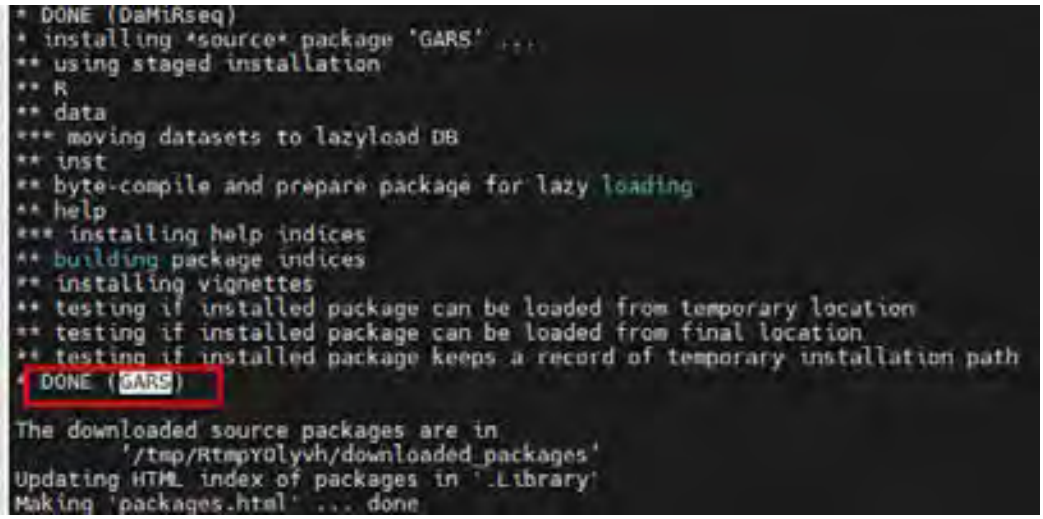
```
vim ~/.Rprofile
```

```
options(rgl.useNULL = TRUE)
options(rgl.debug = TRUE)
~
```

Step 5 Install clippda.

```
BiocManager::install("GARS")
```

If **DONE (GARS)** is displayed in the command output, the installation is successful.



----End

4.171.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("GARS")
```

Command output:

If **GARS** is displayed in the command output, the verification is successful.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.171.8 Troubleshooting

4.171.8.1 Failed to Download the Installation Package of the Dependency Software

This case is an example and is for reference only. The actual situation prevails.

Symptom

An error is reported during compilation. The error information is similar to the following:

```
Error in download.file(url, destfile,method, mode = "wb", failed
download from 'http://mirrors.tuna.tsinghua.edu.cn/bioconductor//packaJes/3.15/data/annotation/src/
contrib/BSgenome.Hsapiens.UCSC.hg19 1.4.3.tar.gz'
```

Key Process and Cause Analysis

The installation package is too large. As a result, the download fails.

Conclusion and Solution

Step 1 Copy the download link and manually download the installation package.

```
wget http://mirrors.tuna.tsinghua.edu.cn/bioconductor//packages/3.15/data/annotation/src/contrib/
BSgenome.Hsapiens.UCSC.hg19_1.4.3.tar.gz
```

Step 2 Install the dependency.

```
R CMD INSTALL BSgenome.Hsapiens.UCSC.hg19
```

----End

4.171.9 Change History

Date	Description
2023-04-27	This issue is the first official release.

4.172 CD-HIT 4.8.1 Porting Guide (Kylin V10)

4.172.1 Introduction

CD-HIT is a very widely used program for clustering and comparing protein or nucleotide sequences.

CD-HIT is very fast and can handle extremely large databases. CD-HIT helps to significantly reduce the computational and manual efforts in many sequence analysis tasks and aids in understanding the data structure and correct the bias within a dataset.

 Copyright © Huawei Technologies Co., Ltd.

The CD-HIT package has CD-HIT, CD-HIT-2D, CD-HIT-EST, CD-HIT-EST-2D, CD-HIT-454, CD-HIT-PARA, PSI-CD-HIT, CD-HIT-OTU, CD-HIT-LAP, CD-HIT-DUP, and over a dozen scripts.

Programming language: Python

Brief description: A program for clustering and comparing protein or nucleotide sequences.

Recommended Software Version

CD-HIT 4.8.1

4.172.2 Environment Requirements

Hardware Requirements

Table 4-853 lists the hardware requirements.

Table 4-853 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-854 lists the OS requirements.

Table 4-854 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.htmlhttps://www.kylinos.cn/scheme/server/1.html
Kernel	4.19/4.14	Included in the OS image

Software Requirements

Table 4-855 lists the software requirements.

Table 4-855 Software requirements

Item	Version	Download URL
CD-HIT	4.8.1	https://github.com/weizhongli/cdhit/archive/refs/tags/V4.8.1.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

4.172.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CD-HIT software porting.

Table 4-856 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bisheng-2.1.0	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/CDHIT	Installation path of CD-HIT.	

4.172.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-857 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

4.172.5 Obtaining the Source Code

Procedure

Step 1 Download the CD-HIT installation package **Cdhit-2.1.tar.gz**.

```
wget https://github.com/weizhongli/cdhit/archive/refs/tags/V4.8.1.tar.gz
```

----End

4.172.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/CDHIT
```

Step 3 Decompress the installation package.

```
tar -xvf V4.8.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd cdhit-4.8.1
```

Step 5 Modify the configuration file.

```
sed -i "1,3s/g++/clang/g" Makefile
```

Step 6 Perform the compilation and installation.

```
make
```

----End

4.172.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the CD-HIT directory.

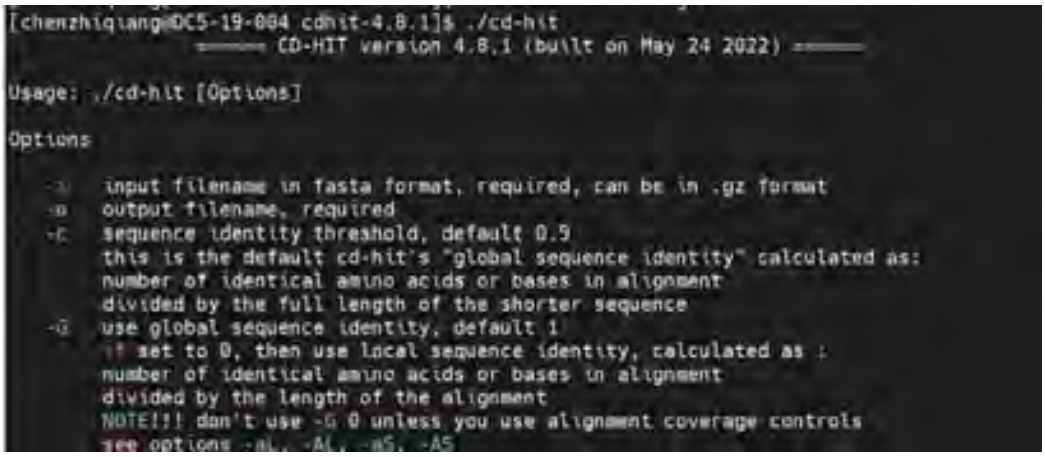
```
cd /path/to/CDHIT/cdhit-4.8.1
```

Step 3 Run CD-HIT.

```
./cdhit
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.173 bioconductor-grohmm 1.30.1 Porting Guide (Kylin V10)

4.173.1 Introduction

Grohmm is a GRO-seq analysis pipeline.

For more information, visit the **official website**.

Programming language: R

Brief description: A GRO-seq analysis pipeline.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.30.1

4.173.2 Environment Requirements

Hardware Requirements

Table 4-858 lists the hardware requirements.

Table 4-858 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-859 lists the software requirements.

Table 4-859 Software requirements

Item	Version	Download URL
Grohmm	1.30.1	http://bioconductor.org/packages/release/bioc/src/contrib/groHMM_1.30.1.tar.gz

OS Requirements

Table 4-860 lists the OS requirements.

Table 4-860 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.173.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Grohmm software porting.

Table 4-861 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/GROHMM	Installation path of the R language and Grohmm.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.173.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.173.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-862 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.173.4.2 Installing R 4.2.0**.

4.173.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/GROHMM
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the installation package and perform the installation.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/GROHMM/R_install
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/GROHMM/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GROHMM/R_install/lib/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add a mirror source of the R language.

```
vim ~/.Rprofile
```

Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.173.5 Obtaining the Source Code

Procedure

Step 1 Use the R language to install the software. You do not need to download the source package.

Step 2 (Optional) Download the Grohmm installation package **groHMM_1.30.1.tar.gz**.

 Copyright © Huawei Technologies Co., Ltd.

http://bioconductor.org/packages/release/bioc/src/contrib/groHMM_1.30.1.tar.gz

(Optional) Use an SFTP tool to upload the Grohmm installation packages to the */path/to/GROHMM* directory on the server.

----End

4.173.6 Compilation and Installation

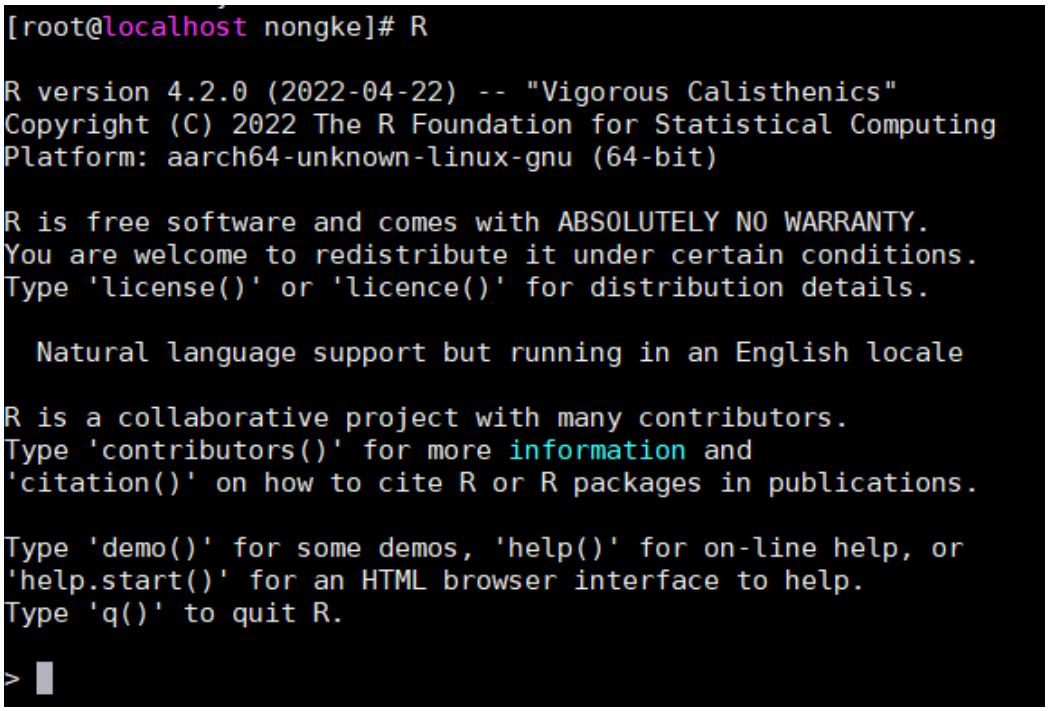
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the R CLI.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

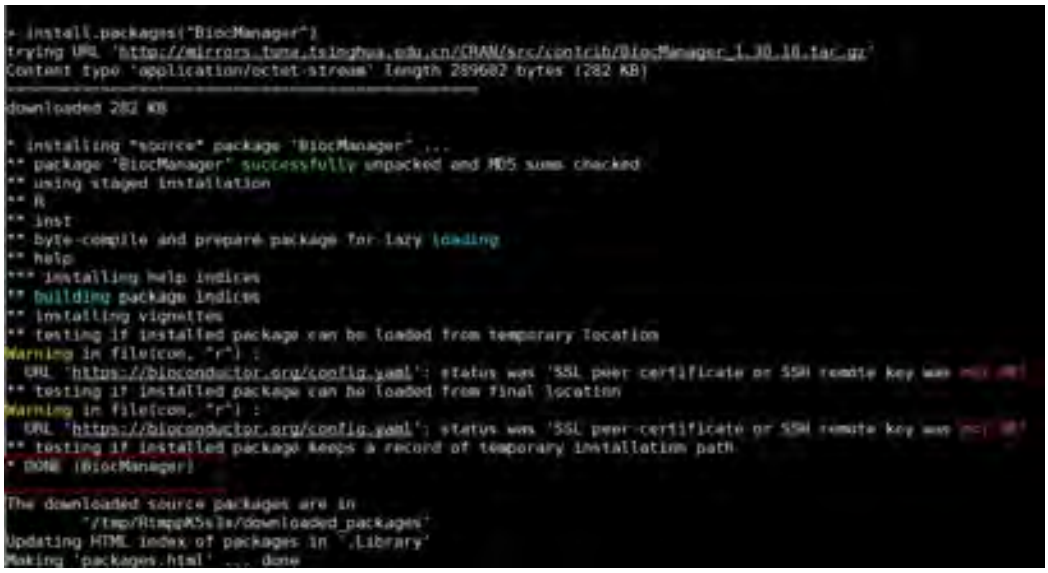


Step 3 Install BiocManager.

```
install.packages("BiocManager")
```

If **DONE (BiocManager)** is displayed in the command output, the installation is successful.

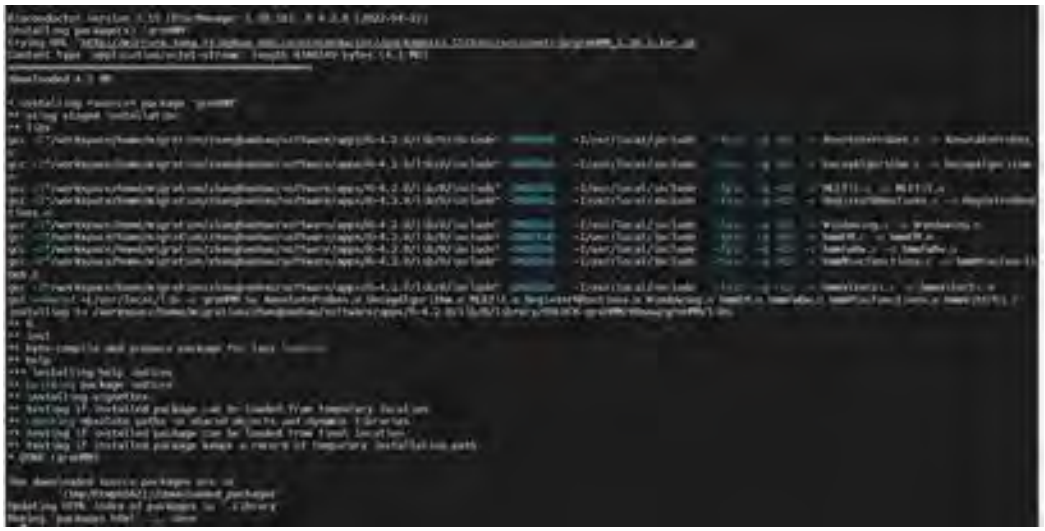
 Copyright © Huawei Technologies Co., Ltd.



Step 4 Install Grohmm.

```
BiocManager::install("groHMM")
```

If **DONE (groHMM)** is displayed in the command output, the installation is successful.



----End

4.173.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("groHMM")
```

If **groHMM** is displayed in the command output, the verification is successful.

----End

4.174 bioconductor-MAGeCKFlute 2.0.0 Porting Guide (Kylin V10)

4.174.1 Introduction

MAGeCKFlute provides several strategies to remove potential biases within sgRNA-level read counts and gene-level beta scores. The downstream analysis with the package includes identifying essential, non-essential, and target-associated genes, and performing biological functional category analysis, pathway enrichment analysis and protein complex enrichment analysis of these genes. The package also visualizes genes in multiple ways to benefit users exploring screening data. Collectively, MAGeCKFlute enables accurate identification of essential, non-essential, and targeted genes, as well as their related biological functions.

For more information, visit the **MAGeCKFlute website**.

Programming language: R

Brief description: Provides several strategies to remove potential biases within sgRNA-level read counts and gene-level beta scores.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.0.0

4.174.2 Environment Requirements

Hardware Requirements

Table 4-863 lists the hardware requirements.

Table 4-863 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-864 lists the software requirements.

Table 4-864 Software requirements

Item	Version	Download URL
R	4.2.0	https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz

OS Requirements

Table 4-865 lists the OS requirements.

Table 4-865 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.174.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MAGeCKFlute software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-866 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/R	Installation path of the R language and MAGeCKFlute.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.174.4 Configuring the Compilation Environment

4.174.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-867 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.174.4.2 Installing R 4.2.0**.

4.174.4.2 Installing R 4.2.0

Procedure

Step 1 Download the R language source package.

```
cd /path/to/R
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the installation package and perform the installation.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/R
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/R/bin:$PATH
export LD_LIBRARY_PATH=/path/to/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add a mirror source of the R language.

```
vim ~/.Rprofile
```

Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.174.5 Obtaining the Source Code

Use the R language to install the software. You do not need to download the source package.

4.174.6 Compilation and Installation

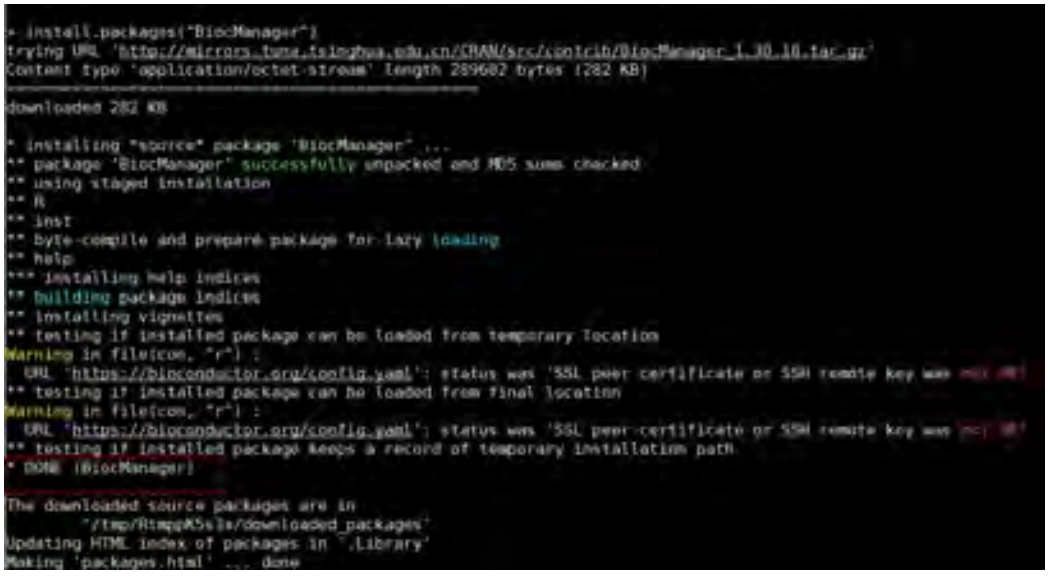
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
R
install.packages("BiocManager")
```

If **DONE (BiocManager)** is displayed in the command output, the installation is successful.



Step 3 Install MAGeCKFlute.

```
BiocManager::install("MAGeCKFlute")
```

If **DONE (MAGeCKFlute)** is displayed in the command output, the installation is successful.

```
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
snapshotDate(): 2022-04-26
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
snapshotDate(): 2022-04-26
** testing if installed package can be loaded from final location
snapshotDate(): 2022-04-26
** testing if installed package keeps a record of temporary installation path
* DONE (MAGeCKFlute)
```

----End

4.174.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("MAGeCKFlute")
```

If **MAGeCKFlute** is displayed in the command output and the R package version is 2.0.0, the verification is successful.

```
> citation("MAGeCKFlute")

To cite package 'MAGeCKFlute' in publications use:

  Wang B, Zhang W, Wu F, Liu WL&XS (2022). _MAGeCKFlute: Integrative
  Analysis Pipeline for Pooled CRISPR Functional Genetic Screens_. R
  package version 2.0.0.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {MAGeCKFlute: Integrative Analysis Pipeline for Pooled CRISPR Functional
Genetic Screens},
    author = {Binbin Wang and Wubing Zhang and Feizhen Wu and Wei Li & X. Shirley Liu},
    year = {2022},
    note = {R package version 2.0.0},
  }

ATTENTION: This citation information has been auto-generated from the
package DESCRIPTION file and may need manual editing, see
'help("citation")'.
```

----End

4.174.8 Troubleshooting

4.174.8.1 Failed to Download the Installation Package of the Dependency Software

This case is an example and is for reference only. The actual situation prevails.

 Copyright © Huawei Technologies Co., Ltd.

Symptom

An error occurs during compilation. The error message is as follows:

Key Process and Cause Analysis

The installation package is too large. As a result, the download fails.

Conclusion and Solution

Step 1 Copy the download link and manually download the installation package.

```
wget http://mirrors.tuna.tsinghua.edu.cn/bioconductor//packages/3.15/data/annotation/src/contrib/BSgenome.Hsapiens.UCSC.hg19_1.4.3.tar.gz
```

Step 2 Install the dependency.

```
R CMD INSTALL BSgenome.Hsapiens.UCSC.hg19
```

----End

4.175 bioconductor-intansv 1.36.0 Porting Guide (Kylin V10)

4.175.1 Introduction

bioconductor-intansv is used for integrative analysis of structural variations.

For more information, visit the **bioconductor-intansv official website**.

Programming language: R

Brief description: Integrative analysis of structural variations.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.36.0

4.175.2 Environment Requirements

Hardware Requirements

Table 4-868 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-868 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-869 lists the OS requirements.

Table 4-869 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-870 lists the software requirements.

Table 4-870 Software requirements

Item	Version	Download URL
bioconductor-intansv	1.36.0	Install it using the R language. You do not need to download the source package.

4.175.3 Paths for Software Porting

This chapter lists the software installation paths involved in the bioconductor-intansv software porting.

Table 4-871 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bioconductor-intansv	Installation path of bioconductor-intansv.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.175.4 Configuring the Compilation Environment

4.175.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-872 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.175.4.2 Installing the R Language**.

4.175.4.2 Installing the R Language

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the R language source package.

```
cd /path/to/bioconductor-intansv
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf R-4.2.0.tar.gz
```

Step 4 Go to the R source code directory.

```
cd R-4.2.0
```

Step 5 Install the dependencies using the **yum** command.

```
yum install libXt-devel libX11-devel readline-devel
```

Step 6 Compile and install R.

```
./configure --prefix=/path/to/bioconductor-intansv/R_install
make
make install
```

Step 7 Load the environment variables.

```
export PATH=/path/to/bioconductor-intansv/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/bioconductor-intansv/R_install/lib/R/library:$LD_LIBRARY_PATH
```

Step 8 (Optional) If the network speed is slow, add a mirror source of the R language.

1. Open the **~/.Rprofile** file.
   ```
   vim ~/.Rprofile
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
   options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
   ```

 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the changes and exit.

----End

4.175.5 Obtaining the Source Code

Use the R language to install bioconductor-intansv. You do not need to download the source package.

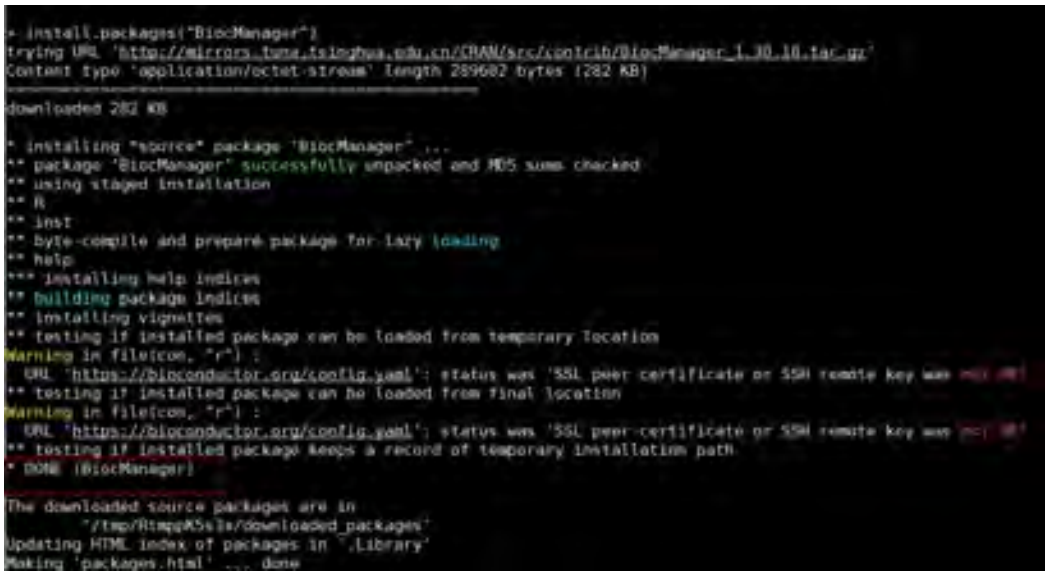
4.175.6 Compilation and Installation

Procedure

Step 1 Install the dependencies.

```
R
install.packages("BiocManager")
```

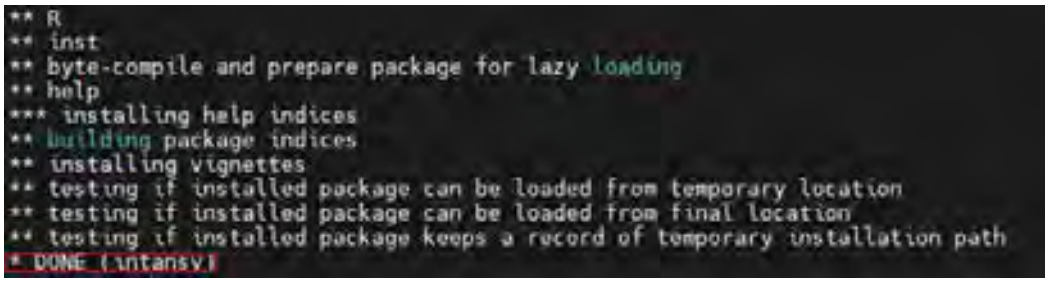
If **DONE (BiocManager)** is displayed in the command output, the installation is successful.



Step 2 Install bioconductor-intansv.

```
BiocManager::install("intansv")
```

If **DONE (intansv)** is displayed in the command output, the installation is successful.



----End

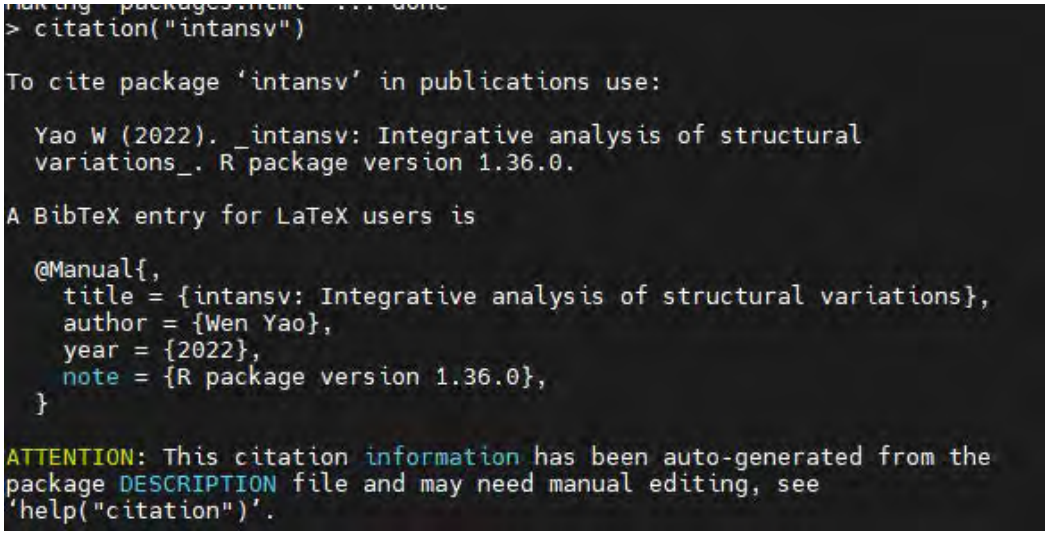
4.175.7 Running and Verification

Procedure

Perform the verification.

```
R
citation("intansv")
```

If **intansv** is displayed in the command output and the R package version is 1.36.0, the verification is successful.



4.175.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

4.176 bioconductor-multihiccompare 1.14.0 Porting Guide (Kylin V10)

4.176.1 Introduction

multiHiCcompare provides functions for joint normalization and difference detection in multiple Hi-C datasets. This extension of the original HiCcompare package now allows for Hi-C experiments with more than 2 groups and multiple samples per group. multiHiCcompare operates on processed Hi-C data in the form of sparse upper triangular matrices. It accepts four column (chromosome, region1, region2, IF) tab-separated text files storing chromatin interaction matrices. multiHiCcompare provides cyclic loess and fast loess (fastlo) methods adapted to jointly normalizing Hi-C data. Additionally, it provides a general linear model (GLM) framework adapting the edgeR package to detect differences in Hi-C data in a distance dependent manner.

 Copyright © Huawei Technologies Co., Ltd.

For more information, visit the **official website**.

Programming language: R

Brief description: A generalized linear model (GLM) frame.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.14.0

4.176.2 Environment Requirements

Hardware Requirements

Table 4-873 lists the hardware requirements.

Table 4-873 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-874 lists the software requirements.

Table 4-874 Software requirements

Item	Version	Download URL
multiHiCcompare	1.14.0	**http://bioconductor.org/packages/release/bioc/src/contrib/multiHiCcompare_1.14.0.tar.gz**

OS Requirements

Table 4-875 lists the OS requirements.

Table 4-875 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

4.176.3 Paths for Software Porting

This chapter lists the software installation paths involved in the multiHiCcompare software porting.

Table 4-876 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/MULTIHICCOMPARE	Installation path of the R language and multiHiCcompare.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.176.4 Configuring the Compilation Environment

4.176.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-877 Configuration process

No.	Configuration Item	Remarks
1	Installing the R language	See **4.176.4.2 Installing R 4.2.0**.

 Copyright © Huawei Technologies Co., Ltd.

4.176.4.2 Installing R 4.2.0

Procedure

Step 1 Go to the installation directory and download the R language source package.

```
cd /path/to/MULTIHICCOMPARE
wget https://cran.r-project.org/src/base/R-4/R-4.2.0.tar.gz
```

Step 2 Decompress the installation package and perform the installation.

```
tar xvf R-4.2.0.tar.gz
cd R-4.2.0
yum install libXt-devel libX11-devel readline-devel
./configure --prefix=/path/to/MULTIHICCOMPARE/R_install
make
make install
```

Step 3 Load the environment variables.

```
export PATH=/path/to/MULTIHICCOMPARE/R_install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MULTIHICCOMPARE/R_install/lib/R/lib:$LD_LIBRARY_PATH
```

Step 4 (Optional) Add a mirror source of the R language.

```
vim ~/.Rprofile
```

Add the following content to the file:

```
options("repos"= c(CRAN="http://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
```

----End

4.176.5 Obtaining the Source Code

Procedure

Step 1 Use the R language to install the software. You do not need to download the source package.

Step 2 (Optional) Download the multiHiCcompare installation package **multiHiCcompare_1.14.0.tar.gz**.

http://bioconductor.org/packages/release/bioc/src/contrib/multiHiCcompare_1.14.0.tar.gz

Step 3 (Optional) Use an SFTP tool to upload the multiHiCcompare installation packages to the */path/to/MULTIHICCOMPARE* directory on the server.

----End

4.176.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Start the R parser.

```
R
```

If **R Version 4.2.0** is displayed in the command output, the R parser is started successfully.

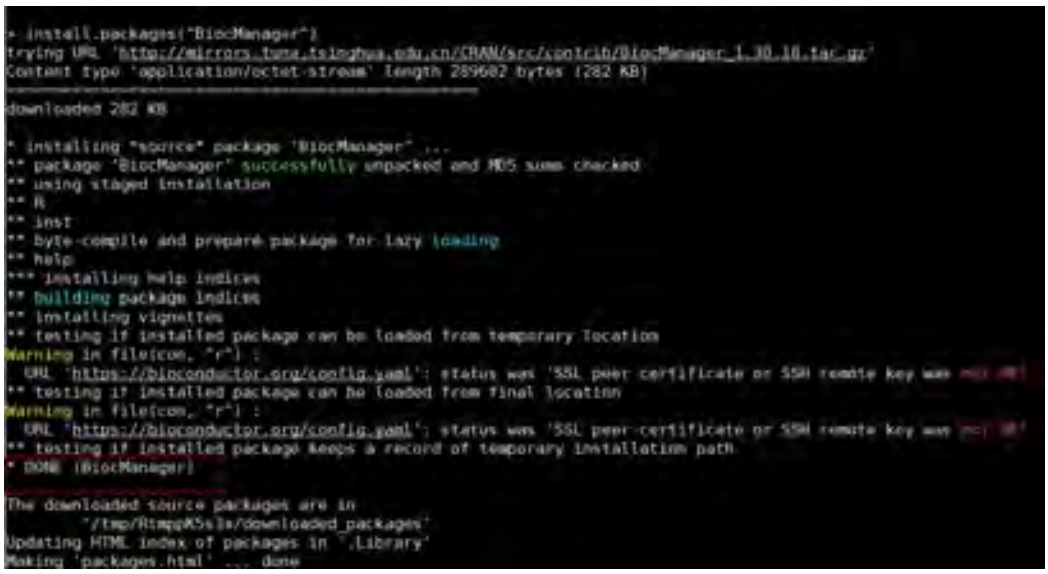
 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost nongke]# R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Step 3 Install BiocManager.

```
install.packages("BiocManager")
```

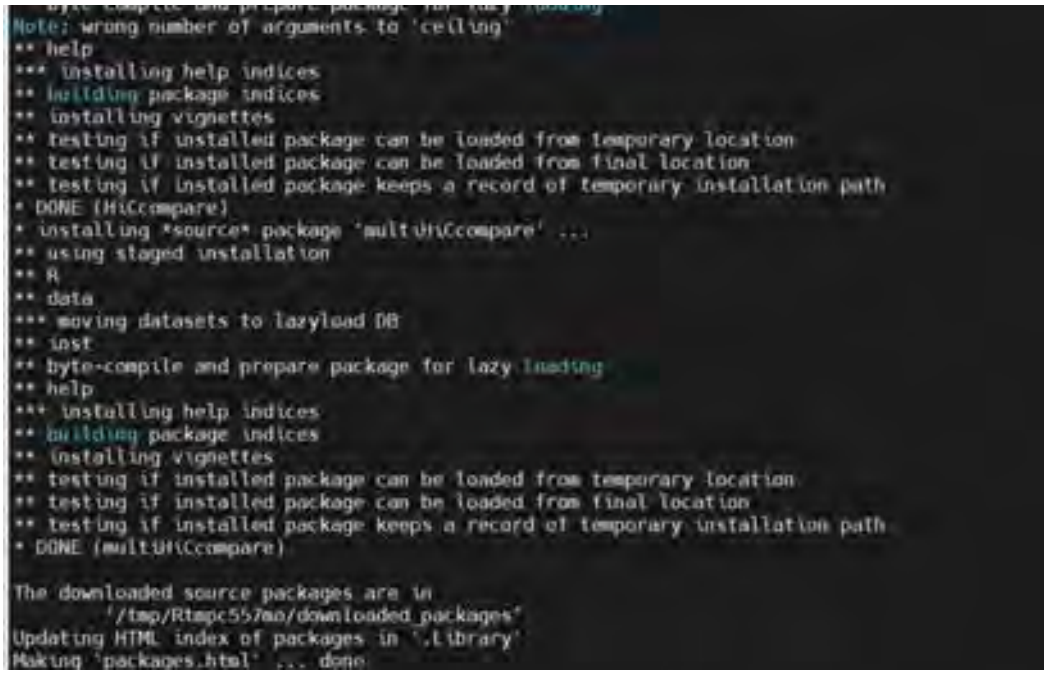
If **DONE (BiocManager)** is displayed in the command output, the installation is successful.



Step 4 Install multiHiCcompare.

```
BiocManager::install("multiHiCcompare")
```

If **DONE (multiHiCcompare)** is displayed in the command output, the installation is successful.



----End

4.176.7 Running and Verification

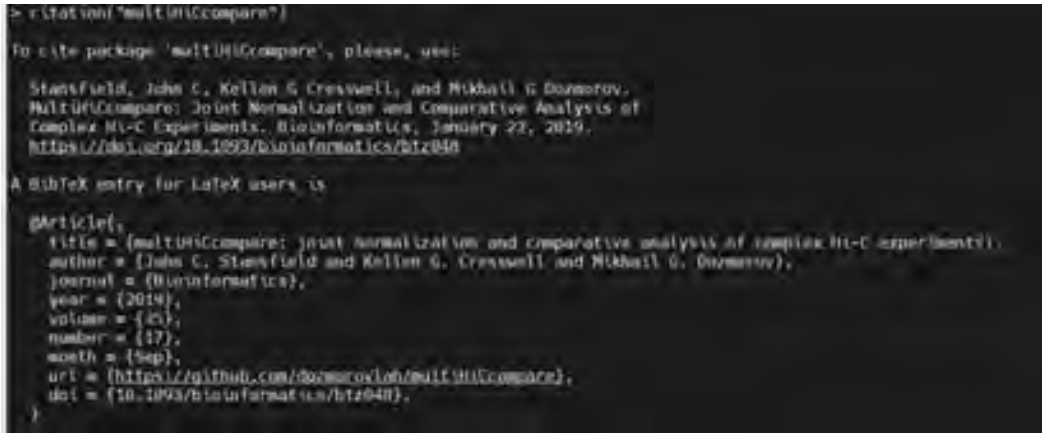
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
R
citation("multiHiCcompare")
```

If **multiHiCcompare** is displayed in the command output, the verification is successful.



----End

4.177 ClustalW 2.1 Porting Guide (Kylin V10)

4.177.1 Introduction

ClustalW is a general purpose multiple alignment program for DNA or proteins. It uses progressive alignment methods, which align the most similar sequences first and work their way down to the least similar sequences until a global alignment is created. ClustalW is a matrix-based algorithm, whereas tools like T-Coffee and Dialign are consistency-based. ClustalW has a fairly efficient algorithm that competes well against other software. This program requires three or more sequences in order to calculate a global alignment, for pairwise sequence alignment (2 sequences) use tools similar to EMBOSS or LALIGN.

For more information, visit the **Clustal official website**.

Programming language: C++

Brief description: A general purpose multiple alignment program for DNA or proteins.

Recommended Software Version

ClustalW 2.1

4.177.2 Environment Requirements

Hardware Requirements

Table 4-878 lists the hardware requirements.

Table 4-878 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-879 lists the OS requirements.

Table 4-879 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19/4.14	Included in the OS image

Software Requirements

Table 4-880 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-880 Software requirements

Item	Version	Download URL
ClustalW	2.1	http://ftp.ClustalW.org/pub/ClustalW/ClustalW-2.1.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

4.177.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ClustalW software porting.

Table 4-881 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/bisheng-2.1.0	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/CLUSTALW	Installation path of ClustalW.	
3	/path/to/CASE	Path for storing ClustalW test cases.	

4.177.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-882 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

4.177.5 Obtaining the Source Code

Procedure

Step 1 Download the ClustalW installation package **ClustalW-2.1.tar.gz**.

```
cd /path/to/CLUSTALW
wget http://www.clustal.org/download/current/clustalw-2.1.tar.gz
```

----End

4.177.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/CLUSTALW
```

Step 3 Decompress the installation package.

```
tar -xvf clustalw-2.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd clustalw-2.1
```

Step 5 Modify the configuration file.

```
mv config.guess config.guess_bak
mv config.sub config.sub_bak
cp /usr/share/automake-1.16/config.* .
```

Step 6 Perform the configuration.

```
./configure --prefix=/path/to/CLUSTALW
```

Step 7 Perform the compilation and installation.

```
make
make install
```

Step 8 Check whether an executable file is generated.

```
ll /path/to/CLUSTALW/bin/clustalw2
```

Step 9 Set the environment variable.

```
export PATH=/path/to/CLUSTALW/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.177.7 Running and Verification

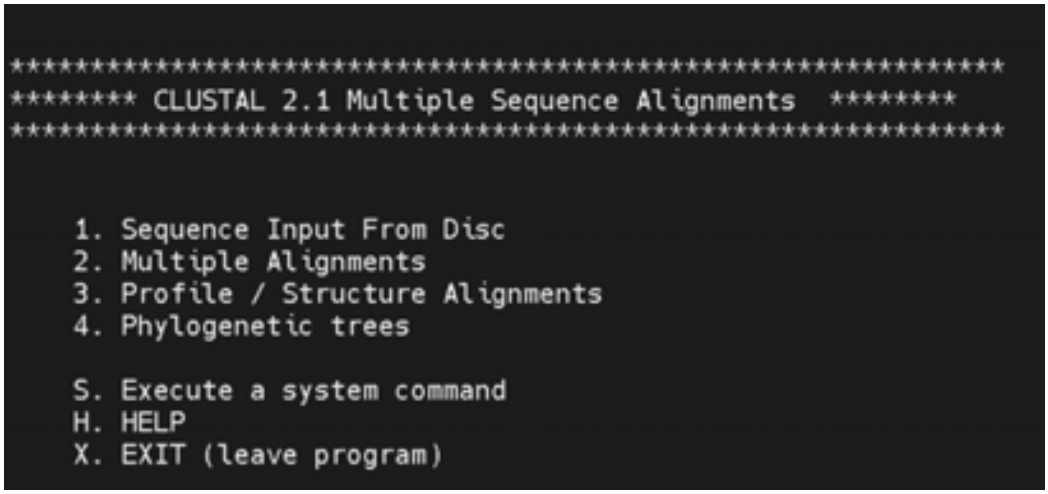
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run ClustalW.

```
./clustalw2
```

If the following information is displayed, the verification is successful.



----End

4.178 HMMER 3.3.2 Porting Guide (Kylin V10)

4.178.1 Introduction

HMMER searches biological sequence databases for homologous sequences, using either single sequences or multiple sequence alignments as queries. HMMER implements a technology called "profile hidden Markov models" (profile HMMs).

For more information, visit the **HMMER page at GitHub**.

Programming language: C

Brief description: HMMER implements a technology called "profile hidden Markov models" (profile HMMs).

Open source license: BSD

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

3.3.2

 Copyright © Huawei Technologies Co., Ltd.

4.178.2 Environment Requirements

Hardware Requirements

Table 4-883 lists the hardware requirements.

Table 4-883 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-884 lists the OS requirements.

Table 4-884 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-885 lists the software requirements.

Table 4-885 Software requirements

Item	Version	Download URL
HMMER	3.3.2	https://github.com/EddyRivasLab/hmmer/archive/refs/heads/h3-arm.zip
Easel	Latest	https://github.com/EddyRivasLab/easel/archive/refs/heads/master.zip
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz
HMPI	1.2.1	https://gitee.com/kunpengcompute/hmpi/archive/refs/tags/v1.2.1-huawei.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

4.178.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HMMER software porting.

Table 4-886 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/hmmer	Installation path of HMMER.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.178.4 Configuring the Compilation Environment

4.178.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-887 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing HMPI	See **Installing HMPI**.
3	Installing Dependencies	See **4.178.4.2 Installing Dependencies**.

4.178.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Install the dependencies.

```
yum install -y autoconf
```

----End

4.178.5 Obtaining the Source Code

Procedure

Step 1 Download the HMMER and Easel installation packages.

https://github.com/EddyRivasLab/hmmer/archive/refs/heads/h3-arm.zip

https://github.com/EddyRivasLab/easel/archive/refs/heads/master.zip

NOTICE

- The official HMMER3 source package does not support the Arm architecture. Therefore, you are advised to download the H3-Arm source package from the GitHub repository. Compared with 3.3.2, the H3-Arm source package adapts to the Arm architecture.
- See **HMMER User's Guide (eddylab.org)** for reference.

Step 2 Use an SFTP tool to upload the HMMER and Easel installation packages to the */path/to/hmmer* directory on the server.

----End

4.178.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
unzip hmmer-h3-arm.zip -d /path/to/hmmer
unzip easel-master.zip -d /path/to/hmmer/hmmer-h3-arm
```

Step 2 Go to the source code directory and change the related directory name.

```
cd /path/to/hmmer
mv hmmer-h3-arm/ hmmer-3.3.2/
cd hmmer-3.3.2/
mv easel-master/ easel/
```

Step 3 Set the environment variables.

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
```

Step 4 Generate **Makefile** and modify the related file.

```
autoconf
./configure --prefix=/path/to/hmmer/hmmer --enable-mpi --enable-neon CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
sed -i 's/esl_FCompare/esl_FCompareNew/g' src/p7_domain.c
sed -i 's/esl_DCompare/esl_DCompareNew/g' src/p7_domain.c
```

 Copyright © Huawei Technologies Co., Ltd.

```
sed -i 's/esl_FCompare/esl_FCompareNew/g' src/p7_hit.c
sed -i 's/esl_DCompare/esl_DCompareNew/g' src/p7_hit.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/p7_hmm.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/p7_trace.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/impl_neon/p7_oprofile.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/generic_vtrace.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/p7_bg.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/p7_gmx.c
sed -i '41 d' src/impl_neon/impl_neon.h
sed -i '29c static inline float' src/impl_neon/impl_neon.h
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/./generic_stotrace.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/./generic_viterbi.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/./modelconfig.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/impl_neon/./stotrace.c
sed -i 's/esl_FCompare_old/esl_FCompare/g' src/impl_neon/./optacc.c
```

Step 5 Perform the compilation and installation.

```
make -j$(nproc)
make check
make -j$(nproc) install
```

Step 6 Load the HMMER environment variable.

```
export PATH=/path/to/hmmer/hmmer/bin:$PATH
```

Step 7 (Optional) Install the Easel library.

```
cd easel/
make install
```

----End

4.178.7 Running and Verification

Procedure

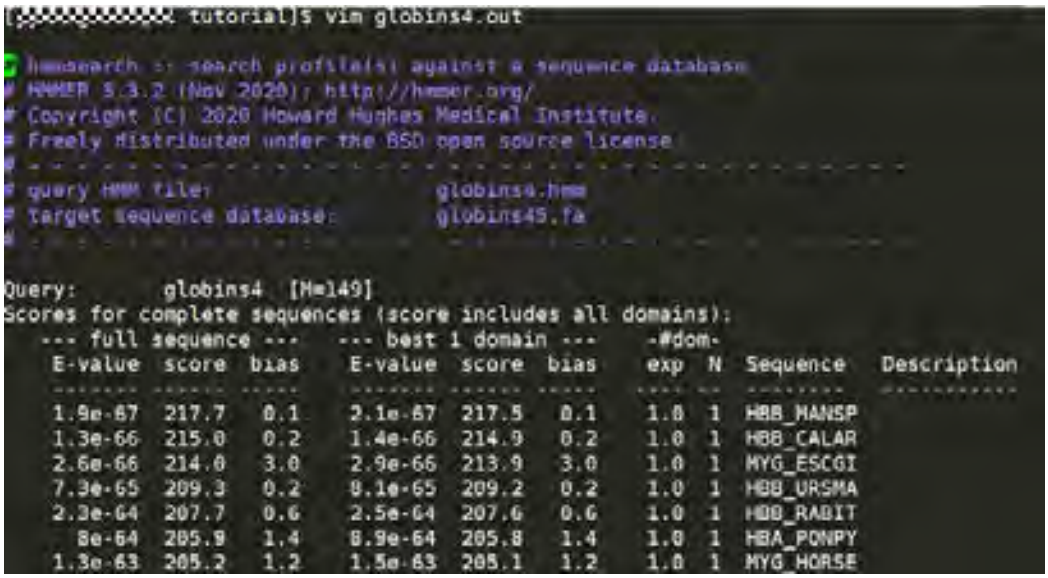
Step 1 Verify the built-in test cases.

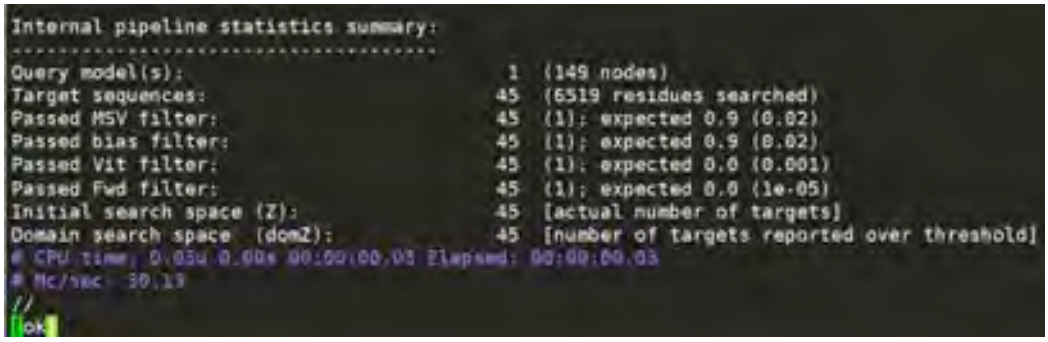
```
cd /path/to/hmmer/hmmer--3.3.2/tutorial
hmmbuild globins4.hmm globins4.sto
```

If the following information is displayed, the verification is successful.

```
hmmsearch globins4.hmm globins45.fa > globins4.out
```

If the following information is displayed, the verification is successful.





----End

4.178.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

4.179 IDBA-UD 1.0 Porting Guide (Kylin V10)

4.179.1 Introduction

IDBA_UD is the basic iterative de Bruijn graph assembler for second-generation sequencing reads. IDBA-UD, an extension of IDBA, is designed to utilize paired-end reads to assemble low-depth regions and use progressive depth on contigs to reduce errors in high-depth regions. It is a generic purpose assembler and especially good for single-cell and metagenomic sequencing data.

For more information, visit the **IDBA_UD page at GitHub**.

Programming language: C++

Brief description: Basic iterative de Bruijn graph assembler for second-generation sequencing reads.

Open source license: unknown

NOTE

When using open source software, comply with the applicable licenses.

Recommended Version

Master

4.179.2 Environment Requirements

Hardware Requirements

Table 4-888 lists the hardware requirements.

Table 4-888 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-889 lists the OS requirements.

Table 4-889 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 4-890 lists the software requirements.

Table 4-890 Software requirements

Item	Version	Download URL
IDBA_UD	--	https://github.com/1928d/idba_ud

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
CMake	3.22.5	https://github.com/Kitware/CMake/releases/download/v3.22.5/cmake-3.22.5-linux-aarch64.tar.gz

4.179.3 Paths for Software Porting

This chapter lists the software installation paths involved in the IDBA_UD software porting.

Table 4-891 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	*/path/to/bisheng-2.1.0*	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	*/path/to/cmake*	Installation path of CMake.	
3	*/path/to/idba_ud*	Installation path of IDBA_UD.	

4.179.4 Configuring the Compilation Environment

4.179.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-892 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **4.1**.
2	Installing CMake	See **4.2**.

4.179.4.2 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
mkdir -p /opt/compiler
cp -r bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz /opt/compiler
cd /opt/compiler
tar -zxvf bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
```

Step 3 Set the BiSheng Compiler environment variable.

```
export PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 4 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

4.179.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory and decompress the package.

```
cd /path/to/cmake
tar xvf cmake-3.22.5-linux-aarch64.tar.gz
```

Step 3 Set the CMake environment variable.

```
export PATH=/path/to/cmake/cmake-3.22.5-linux-aarch64/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.179.5 Obtaining the Source Code

Procedure

Step 1 Download IDBA_UD.

https://github.com/1928d/idba_ud

Step 2 Download command:

```
git clone https://github.com/1928d/idba_ud.git
```

----End

4.179.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd idba_ud/
```

Step 3 Modify the **idba_ud/CMakeLists.txt** file.

Change **-march=native** to **-march=armv8.4-a+dotprod**.

Step 4 Perform the compilation and installation.

```
mkdir build
cd build
cmake ../
make
```

If the IDBA_UD executable file is generated in the current directory, the installation is complete.

Step 5 Set the environment variables.

```
export PATH=/path/to/idba_ud/build:$PATH
```

----End

4.179.7 Running and Verification

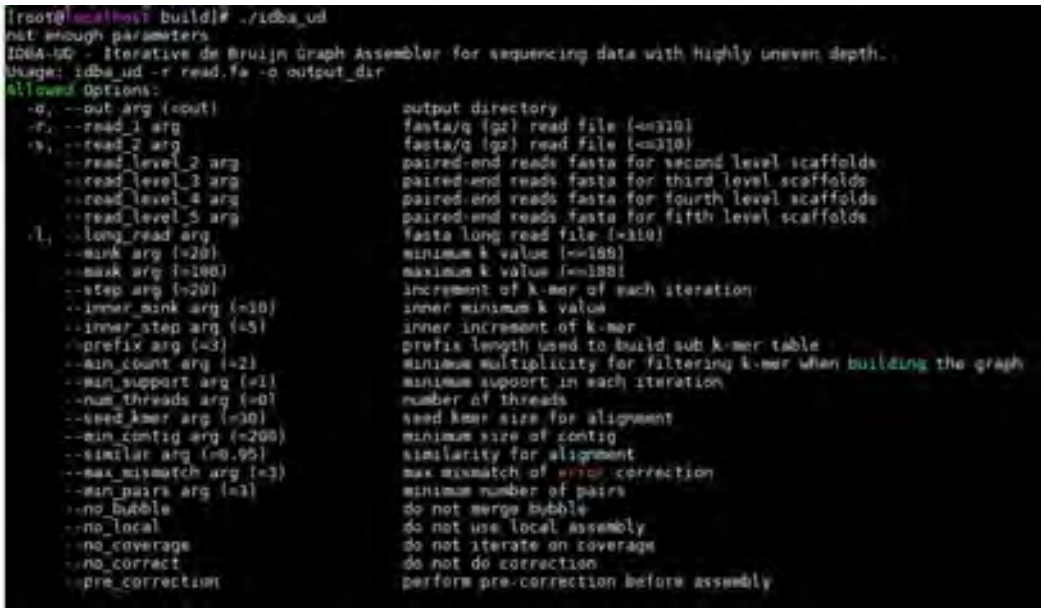
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

```
cd idba_ud/build
./idba_ud
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.179.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

4.180 InterProScan 5.53-87.0 Porting Guide (Kylin V10)

4.180.1 Introduction

InterPro is a database which integrates together predictive information about proteins' function from a number of partner resources, giving an overview of the families that a protein belongs to and the domains and sites it contains. InterProScan is a piece of software that can use the InterPro database.

For more information, visit the **InterProScan website**.

Programming language: Java/Python

Brief description: A piece of software that can use the InterPro database.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

5.53-78.0

4.180.2 Environment Requirements

Hardware Requirements

Table 4-893 lists the hardware requirements.

Table 4-893 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-894 lists the OS requirements.

Table 4-894 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-895 lists the software requirements.

Table 4-895 Software requirements

Item	Version	Download URL
interproscan	5.53-87.0	https://ftp.ebi.ac.uk/pub/software/unix/iprscan/5/5.53-87.0/interproscan-5.53-87.0-64-bit.tar.gz

4.180.3 Paths for Software Porting

This chapter lists the software installation paths involved in the InterProScan software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-896 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/interproscan	Installation path of InterProScan.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.180.4 Configuring the Compilation Environment

4.180.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server is connected to the Internet.

Configuration Process

Table 4-897 Configuration process

No.	Configuration Item	Remarks
1	Installing HMMER	See **4.178 HMMER 3.3.2 Porting Guide (Kylin V10)**.
2	Installing dependencies	See **4.180.4.2 Installing Dependencies**.

4.180.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y python3 python3-devel perl perl-CPAN java-11-openjdk java-11-openjdk-devel
```

----End

4.180.5 Obtaining the Source Code

Procedure

Step 1 Download the InterProScan installation package.

https://ftp.ebi.ac.uk/pub/software/unix/iprscan/5/5.53-87.0/interproscan-5.53-87.0-64-bit.tar.gz

Step 2 Use an SFTP tool to upload the InterProScan installation package to the *path/to/interproscan* directory on the server.

----End

4.180.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/interproscan
tar xf interproscan-5.53-87.0-64-bit.tar.gz
cd interproscan-5.53-87.0
```

Step 2 Modify the executable file path in the **interproscan.properties** file.

```
vim interproscan.properties
```

Modify it as follows:

```
#hmmer 3
binary.hmmer3.path=/to/path/HMMER
binary.hmmer3.hmmscan.path=/to/path/HMMER/hmmscan
binary.hmmer3.hmmsearch.path=/to/path/HMMER/hmmsearch

binary.hmmer33.path=/to/path/HMMER
binary.hmmer33.hmmscan.path=/to/path/HMMER/mmscan
binary.hmmer33.hmmsearch.path=/to/path/HMMER/hmmsearch
```

CAUTION

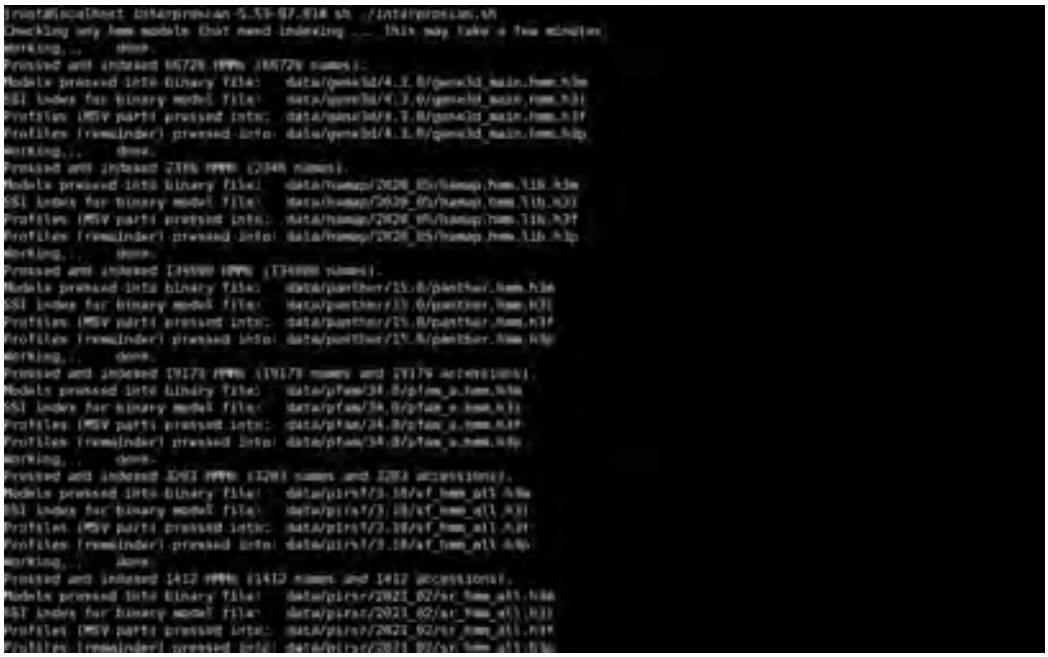
Replace the example path **/to/path/** with the actual one.

Step 3 Perform the installation.

```
sh ./interproscan.sh
```

If the following information is displayed, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.180.7 Change History

Date	Description
2023-04-26	This issue is the first official release.

4.181 MEME 5.4.0 Porting Guide (Kylin V10)

4.181.1 Introduction

The MEME Suite supports motif-based analysis of DNA, RNA, and protein sequences. You can also use it with sequences over novel sequence alphabets that you define. The Suite provides motif discovery algorithms using both probabilistic (MEME) and discrete models (STREME), which have complementary strengths. It also allows discovery of motifs with arbitrary insertions and deletions (GLAM2). The MEME Suite provides three tools for motif enrichment analysis – measuring the enrichment of known motifs in sets of sequences. The motif enrichment may be anywhere in the sequences (SEA, AME), or concentrated in the central regions of the sequences (CentriMo). The Suite also provides an algorithm for measuring the similarity between motifs (Tomtom). These three types of analysis are combined in a pipeline in two MEME Suite tools – XSTREME and MEME-ChIP – that perform comprehensive motif analysis in general sequences (XSTREME) or in ChIP-seq peaks (MEME-ChIP). In addition to motif discovery, the MEME Suite provides tools for scanning sequences for matches to motifs (FIMO, MAST and GLAM2Scan), scanning for clusters of motifs (MCAST), finding preferred spacings between motifs (SpaMo), predicting the biological roles of motifs (GOMo), and predicting the regulatory targets of transcription factors.

For more information, visit the **MEME official website**.

Programming language: C

Brief description: A tool for motif-based analysis of DNA, RNA and protein sequences.

Open source license: GPL 3.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

MEME 5.4.0

4.181.2 Environment Requirements

Hardware Requirements

Table 4-898 lists the hardware requirements.

Table 4-898 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-899 lists the OS requirements.

Table 4-899 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 4-900 lists the software requirements.

Table 4-900 Software requirements

Item	Version	Download URL
MEME	5.4.0	https://meme-suite.org/meme/meme-software/5.4.0/meme-5.4.0.tar.gz

4.181.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MEME software porting.

Table 4-901 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/MEME	Installation path of MEME.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

4.181.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-902 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	For details, see "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

4.181.5 Obtaining the Source Code

Procedure

Step 1 Download the MEME installation package **meme-5.4.0.tar.gz**.

https://meme-suite.org/meme/meme-software/5.4.0/meme-5.4.0.tar.gz

Step 2 Use an SFTP tool to upload the MEME installation package to the */path/to/MEME* directory on the server.

----End

4.181.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/MEME
```

Step 3 Decompress the installation package.

```
tar -xvf meme-5.4.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd meme-5.4.0
```

Step 5 Install the dependencies.

```
yum install -y libtool* autoconf  automake nodejs*  perl*  libxml2* libxslt*
cpan XML::Simple
cpan Sys::info
cpan Math::CDF
cpan HTML::TreeBuilder.pm
cpan JSON
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Compile and install MEME.

```
./configure --prefix=/path/to/MEME
make
make install
```

----End

4.181.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test case directory and go to the directory.

```
cd /path/to/MEME/bin
```

Step 3 Run MEME-ChIP.

```
./meme-chip
```

If the following information is displayed, the verification is successful.

----End

4.181.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

4.182 MetaVelvet 1.2.01 Porting Guide (Kylin V10)

4.182.1 Introduction

MetaVelvet is a de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454. The fundamental ideas of MetaVelvet are first decomposing de Bruijn graph constructed from mixed short reads into individual sub-graphs and second building scaffolds based on every decomposed de Bruijn sub-graph as isolated species genome. MetaVelvet has been proven to generate assemblies of longer N50 and higher quality than Velvet when applied to metagenomic sequences, and recognized as one of frequently-used metagenomic assemblers in this research community.

For more information, visit the **MetaVelvet official website**.

Programming language: C

Brief description: A de novo genomic assembler specially designed for short read sequencing technologies.

Open source license: GPL 3.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

MetaVelvet 1.2.01

4.182.2 Environment Requirements

Hardware Requirements

Table 4-903 lists the hardware requirements.

Table 4-903 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-904 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-904 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 4-905 lists the software requirements.

Table 4-905 Software requirements

Item	Version	Download URL
MetaVelvet	1.2.01	**https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10**
Test case	HMP.small	**http://metavelvet.dna.bio.keio.ac.jp/data/HMP.small.tar.gz**

4.182.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MetaVelvet software porting.

Table 4-906 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/Velvet	Installation path of MetaVelvet.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.182.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

4.182.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-907 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **4.1**.

4.182.4.2 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
mkdir -p /opt/compiler
cp -r bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz /opt/compiler
cd /opt/compiler
tar -zxvf bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
```

Step 3 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 4 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

4.182.5 Obtaining the Source Code

Procedure

Step 1 Download the MetaVelvet installation package **v1.2.10.tar.gz**.

https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10

NOTE

The MetaVelvet source code is included in the Velvet installation package.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Velvet installation package to the */path/to/Velvet* directory on the server.

----End

4.182.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/VELVET
```

Step 3 Decompress the installation package.

```
tar -xvf velvet-1.2.10.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd velvet-1.2.10
```

Step 5 Modify **Makefile**.

```
sed -i '1s/gcc/clang/g' Makefile
```

Step 6 Compile and install MetaVelvet.

```
make 'CATEGORIES=10' 'MAXKMERLENGTH=57' 'LONGSEQUENCES=1' 'OPENMP=1' 'BUNDLEDZLIB=1'
```

After the compilation is complete, two executable files, **Velveth** and **Velvetg**, are generated.

Step 7 Load the environment variable.

```
export PATH=`pwd`:$PATH
```

Step 8 Go to the MetaVelvet installation directory.

```
cd contrib/MetaVelvet-v1.2.01/
```

Step 9 Modify **Makefile**.

```
sed -i '1s/gcc/clang/g' Makefile
sed -i '2s/g++/clang++/g' Makefile
sed -i "3s%$% -I../../third-party/zlib-1.2.3/%g" Makefile
```

Step 10 Compile and install MetaVelvet.

```
make 'CATEGORIES=10' 'MAXKMERLENGTH=57'
```

Step 11 Load the environment variable.

```
export PATH=`pwd`:$PATH
```

----End

4.182.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test case directory and go to the directory.

```
mkdir /path/to/VELVET/testcase
cd /path/to/VELVET/testcase
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Upload the downloaded test case file to the directory.

Step 4 Decompress the test case file.

```
tar zxvf HMP.small.tar.gz
```

Step 5 Run the **velveth** executable file to prepare data.

```
velveth out-dir 51  -fasta -shortPaired HMP.small/SRR041654_shuffled.fasta HMP.small/
SRR041655_shuffled.fasta
```

Step 6 Run the **velvetg** executable file to assemble the genome.

```
velvetg out-dir -exp_cov auto -ins_length 260
```

Step 7 Run **meta-velvetg** to output the assembly result.

```
meta-velvetg out-dir -ins_length 260
```

----End

4.182.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

4.183 mothur 1.48 Porting Guide (Kylin V10)

4.183.1 Introduction

mothur is an open source software package for bioinformatics data processing. The package is frequently used in the analysis of DNA from uncultured microbes. Since its initial version released in February 2009, mothur has been upgraded to version 1.48.

It incorporates the functionality of dotur, sons, treeclimber, s-libshuff, unifrac, and much more. mothur is capable of processing data generated from several DNA sequencing methods including 454 pyrosequencing, Illumina HiSeq and MiSeq, Sanger, PacBio, and IonTorrent.

For more information, visit the **mothur official website**.

Programming language: C++

Brief description: An open source software package for bioinformatics data processing.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

mothur 1.48

 Copyright © Huawei Technologies Co., Ltd.

4.183.2 Environment Requirements

Hardware Requirements

Table 4-908 lists the hardware requirements.

Table 4-908 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-909 lists the OS requirements.

Table 4-909 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 4-910 lists the software requirements.

Table 4-910 Software requirements

Item	Version	Download URL
mothur	1.48	https://github.com/mothur/mothur/archive/refs/tags/v1.48.0.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

4.183.3 Paths for Software Porting

This chapter lists the software installation paths involved in the mothur software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-911 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/mothur	Installation path of mothur.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.183.4 Configuring the Compilation Environment

4.183.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-912 Configuration process

No.	Configuration Item	Remarks
1	Installing BiSheng Compiler 2.1.0	See "Installing and Using BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

4.183.5 Obtaining the Source Code

Procedure

Step 1 Download the mothur installation package **mothur-1.48.tar.gz**.

https://github.com/mothur/mothur/archive/refs/tags/v1.48.0.tar.gz

Step 2 Use an SFTP tool to upload the mothur installation package to the */path/to/mothur* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.183.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/mothur
```

Step 3 Decompress the installation package.

```
tar -xvf  v1.48.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd mothur-1.48.0
```

Step 5 Perform the compilation and installation.

```
make
```

Step 6 Check whether an executable file is generated.

```
ll /path/to/mothur/mothur
```

----End

4.183.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run mothur.

```
./mothur
```

If the following information is displayed, the running is successful.

----End

4.183.8 Change History

Date	Description
2023-04-28	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

4.184 PHYLIP 3.697 Porting Guide (Kylin V10)

4.184.1 Introduction

PHYLogeny Inference Package (PHYLIP) is a free computational phylogenetics package of programs for inferring evolutionary trees (phylogenies).

For more information, visit the **PHYLIP website**.

Programming language: C

Brief description: A free computational phylogenetics package of programs for inferring evolutionary trees (phylogenies).

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

3.697

4.184.2 Environment Requirements

Hardware Requirements

Table 4-913 lists the hardware requirements.

Table 4-913 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-914 lists the OS requirements.

Table 4-914 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-915 lists the software requirements.

Table 4-915 Software requirements

Item	Version	Download URL
Phylip	3.697	http://evolution.gs.washington.edu/phylip/download/phylip-3.697.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

4.184.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PHYLIP software porting.

Table 4-916 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/Phylip	Installation path of PHYLIP.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.184.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-917 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

4.184.5 Obtaining the Source Code

Procedure

Step 1 Download the PHYLIP installation package.

http://evolution.gs.washington.edu/phylip/download/phylip-3.697.tar.gz

Step 2 Use an SFTP tool to upload the PHYLIP installation package to the */path/to/PHYLIP* directory on the server.

----End

4.184.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/Phylip
tar -xvf phylip-3.697.tar.gz
cd phylip-3.697/src
```

Step 2 Modify **Makefile**.

```
vim Makefile.unx
```

Change line 84 to **CC=clang**.

Change line 94 to **CFLAGS=-Wl,-z,muldefs**.

Step 3 Perform the compilation and installation.

```
make -f Makefile.unx install
```

----End

4.184.7 Running and Verification

Procedure

Step 1 Verify the **seqboot** test case.

```
cd ../exe
./seqboot
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
seqboot: can't find input file "infile"
Please enter a new file name> 
```

Step 2 Verify the **retree** test case.

```
./retree
```

If the following information is displayed, the verification is successful.

```
Tree Rearrangement, version 3.697

Settings for this run:
  U          Initial tree (arbitrary, user, specify)?  User tree from tree file
  N   Format to write out trees (PHYLIP, Nexus, XML)?  PHYLIP
  0              Graphics type (IBM PC, ANSI, none)?  ANSI
  W       Width of terminal screen, of plotting area?  80, 80
  L                         Number of lines on screen?  24

Are these settings correct? (type Y or the letter for one to change)
```

----End

4.184.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

4.185 Primer3 2.4.0 Porting Guide (Kylin V10)

4.185.1 Introduction

Primer3 is a software tool commonly used for designing Polymerase Chain Reaction (PCR) primers.

For more information, visit the **Primer3 website**.

Programming language: C

Brief description: A software tool commonly used for designing PCR primers.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.4.0

 Copyright © Huawei Technologies Co., Ltd.

4.185.2 Environment Requirements

Hardware Requirements

Table 4-918 lists the hardware requirements.

Table 4-918 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-919 lists the OS requirements.

Table 4-919 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-920 lists the software requirements.

Table 4-920 Software requirements

Item	Version	Download URL
Primer3	2.4.0	https://sourceforge.net/projects/primer3/files/primer3/2.4.0/primer3-2.4.0.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

4.185.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Primer software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-921 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/PRIMER	Installation path of Primer.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.185.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-922 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

4.185.5 Obtaining the Source Code

Procedure

Step 1 Download the Primer installation package.

https://sourceforge.net/projects/primer3/files/primer3/2.4.0/primer3-2.4.0.tar.gz

Step 2 Use an SFTP tool to upload the Primer installation package to the */path/to/PRIMER* directory on the server.

----End

4.185.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to/PRIMER
tar -xvf primer3-2.4.0.tar.gz
cd primer3-2.4.0/src
```

Step 2 Modify **Makefile**.

```
vim Makefile
```

In line 56, change **CC=gcc** to **CC=clang**.

In line 57, change **CPP=g++** to **CXX=clang++**.

Step 3 Perform the compilation and installation.

```
make -j
```

----End

4.185.7 Running and Verification

Procedure

Perform the verification.

```
./primer3_core --help
```

If the following information is displayed, the verification is successful.

4.185.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

4.186 Sambamba 0.8.2 Porting Guide (openEuler 20.03)

4.186.1 Introduction

Sambamba is a high performance, highly parallel, robust, and fast tool (and library), written in the D programming language, for working with SAM and BAM files. Because of its efficiency, Sambamba is an important work horse running in many sequencing centers around the world today.

For more information, visit the **Sambamba page at GitHub**.

Brief description: A high performance, highly parallel, robust, and fast tool.

Programming language: D

Open source license: GPL 2.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

0.8.2

4.186.2 Environment Requirements

Hardware Requirements

Table 4-923 lists the hardware requirements.

Table 4-923 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-924 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-924 OS requirements

Item	Version	Download URL
openEuler	20.03 LTS SP2	https://www.openeuler.org/en/
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-925 lists the software requirements.

Table 4-925 Software requirements

Item	Version	Download URL
Sambamba	0.8.2	https://github.com/biod/sambamba/archive/refs/tags/v0.8.2.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz
LLVM-based D Compiler (LDC)	1.29.0	https://github.com/ldc-developers/ldc/releases/download/v1.29.0/ldc2-1.29.0-linux-aarch64.tar.xz
Test cases	-	http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00100/alignment/HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.bam

4.186.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Sambamba software porting.

Table 4-926 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/sambamba	Installation path of Sambamba.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/ldc	Installation path of LDC.	

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
3	/path/to/CASE	Path for storing test cases.	

4.186.4 Configuring the Compilation Environment

4.186.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-927 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing LDC	See **4.186.4.2 Installing LDC**.

4.186.4.2 Installing LDC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
cd /path/to/ldc
tar xf ldc2-1.29.0-linux-aarch64.tar.xz
```

Step 3 Set the environment variables of LDC.

```
export PATH=/path/to/ldc/ldc2-1.30.0-beta1-linux-aarch64/bin:$PATH
export LIBRARY_PATH=/path/to/ldc/ldc2-1.30.0-beta1-linux-aarch64/lib:$LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.186.5 Obtaining the Source Code

Procedure

Step 1 Download the Sambamba installation package **sambamba-0.8.2.tar.gz**.

https://github.com/biod/sambamba/archive/refs/tags/v0.8.2.tar.gz

Step 2 Use an SFTP tool to upload the Sambamba installation package to the */path/to/sambamba* directory on the server.

----End

4.186.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/Sambamba
```

Step 3 Decompress the installation package.

```
tar -xvf sambamba-0.8.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd sambamba-0.8.2
```

Step 5 Perform the compilation and installation.

```
make
```

Step 6 Check whether an executable file is generated.

```
ll /path/to/Sambamba/sambamba-0.8.2/bin/
```

If the following information is displayed, the installation is successful.

```
[root@localhost sambamba]# ll bin/
total 45020
-rwxr-xr-x 1 root root 16447112 May 26 10:58 sambamba-0.8.2
-rw-r--r-- 1 root root 29686016 May 26 10:58 sambamba-0.8.2.o
```

Step 7 Set the environment variables.

```
export PATH=/path/to/Sambamba/sambamba-0.8.2/bin:$PATH
```

----End

4.186.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory.

```
cd /path/to/CASE
```

Step 3 Download the test cases.

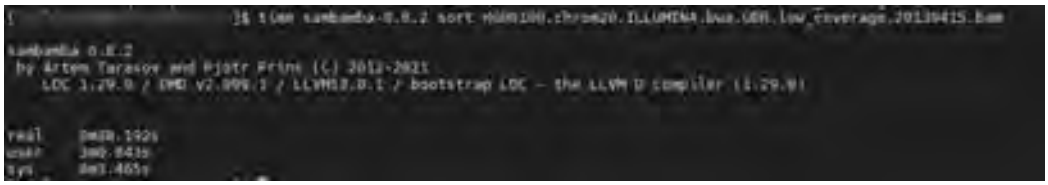
 Copyright © Huawei Technologies Co., Ltd.

```
wget http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00100/alignment/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.bam
```

Step 4 Perform a test.

```
time sambamba-0.8.2 sort HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.bam
```

Check the **real** value (unit: s). A smaller value indicates higher performance.



----End

4.186.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

4.187 SOAPnuke 2.1.7 Porting Guide (Kylin V10)

4.187.1 Introduction

As a novel analysis tool developed for quality control and preprocessing of FASTQ and SAM/BAM data, SOAPnuke includes 5 modules for different usage scenarios, namely filter, filterHts, filterStLFR, filtersRNA and filterMeta.

For more information, visit the **SOAPnuke page at GitHub**.

Programming language: C++

Brief description: A novel analysis tool developed for quality control and preprocessing of FASTQ and SAM/BAM data

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

SOAPnuke 2.1.7

4.187.2 Environment Requirements

Hardware Requirements

Table 4-928 lists the hardware requirements.

Table 4-928 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-929 lists the software requirements.

Table 4-929 Software requirements

Item	Version	Download URL
SOAPnuke	2.1.7	https://github.com/BGI-flexlab/SOAPnuke/archive/refs/tags/SOAPnuke2.1.7.tar.gz
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

OS Requirements

Table 4-930 lists the OS requirements.

Table 4-930 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.187.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SOAPnuke software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-931 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/compiler	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/SOAPnuke	Installation path of SOAPnuke.	

4.187.4 Configuring the Compilation Environment

4.187.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-932 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **4.187.4.2 Installing the BiSheng Compiler**.

4.187.4.2 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
mkdir -p /opt/compiler
cp -r bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz /opt/compiler
cd /opt/compiler
tar -zxvf bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
```

Step 3 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/lib:
```

 Copyright © Huawei Technologies Co., Ltd.

```
$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 4 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

4.187.5 Obtaining the Source Code

Procedure

Step 1 Download the **SOAPnuke2.1.7.tar.gz** package.

https://github.com/BGI-flexlab/SOAPnuke/archive/refs/tags/SOAPnuke2.1.7.tar.gz

Step 2 Use the SFTP tool to upload the **SOAPnuke2.1.7.tar.gz** package to the ***/path/to/SOAPnuke*** directory on the server.

----End

4.187.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar –xvf SOAPnuke2.1.7.tar.gz
```

Step 3 Go to the installation directory of the main program.

```
cd /path/to/SOAPnuke/SOAPnuke-SOAPnuke2.1.7
```

Step 4 Perform the compilation.

```
make
```

----End

4.187.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run SOAPnuke.

```
./SOAPnuke
```

If the following information is displayed, the program is running properly.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost SOAPnuke-SOAPnuke2.1.7]# ^C
[root@localhost SOAPnuke-SOAPnuke2.1.7]# ./SOAPnuke

Program: SOAPnuke
Version: 2.1.7
Contact: GongChun<gongchun@genomics.cn>  ChenYuXin<chenyuxin@genomics.cn>
Command:
         filter        preprocessing normal Fastq files
         filterStLFR   preprocessing stLFR Fastq files
         filtersRNA    preprocessing sRNA Fastq files
         filterMeta    preprocessing Meta Fastq files

[root@localhost SOAPnuke-SOAPnuke2.1.7]#
```

----End

4.187.8 Change History

Date	Description
2022-05-24	This issue is the first official release.

4.188 SOAPsnp Master Porting Guide (Kylin V10)

4.188.1 Introduction

SOAPsnp is a member of the Short Oligonucleotide Analysis Package (SOAP). Despite its name, the program is a resequencing utility that can assemble consensus sequence for the genome of a newly sequenced individual based on the alignment of the raw sequencing reads on the known reference. The SNPs can then be identified on the consensus sequence through the comparison with the reference. In the first Asian genome re-sequencing project, evaluation of SOAPsnp result on Illumina HapMap 1M BeadChip Duo genotyping sites showed great accuracy. Over 99% of the genotyping sites were covered at over 99.9% consistency. Further PCR plus Sanger sequencing of the inconsistent SNP sites confirmed majority of the SOAPsnp results.

For more information, visit the **SOAPsnp official website**.

Programming language: C++

Brief description: A member of the Short Oligonucleotide Analysis Package (SOAP).

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

SOAPsnp-master

 Copyright © Huawei Technologies Co., Ltd.

4.188.2 Environment Requirements

Hardware Requirements

Table 4-933 lists the hardware requirements.

Table 4-933 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-934 lists the software requirements.

Table 4-934 Software requirements

Item	Version	Download URL
SOAPsnp	Master	https://github.com/zzhangjii/soapsnp/archive/refs/heads/master.zip
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz

OS Requirements

Table 4-935 lists the OS requirements.

Table 4-935 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.188.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SOAPsnp software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-936 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/compiler/	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/SOAPsnp	Installation path of SOAPsnp.	

4.188.4 Configuring the Compilation Environment

4.188.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-937 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **4.188.4.2 Installing the BiSheng Compiler**.

4.188.4.2 Installing the BiSheng Compiler

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
mkdir -p /opt/compiler
cp -r bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz /opt/compiler
cd /opt/compiler
tar -zxvf bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
```

Step 3 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/lib:
$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Verify the installation.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

4.188.5 Obtaining the Source Code

Procedure

Step 1 Download the **soapsnp-master.zip** package.

https://github.com/zzhangjii/soapsnp/archive/refs/heads/master.zip

Step 2 Use an SFTP tool to upload the **soapsnp-master.zip** package to the */path/to/SOAPsnp* directory on the server.

----End

4.188.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the package.

```
unzip soapsnp-master.zip
```

Step 3 Go to the installation directory of the main program.

```
cd /path/to/SOAPsnp/soapsnp-master
```

Step 4 Perform the compilation.

```
make
```

----End

4.188.7 Running and Verification

Procedure

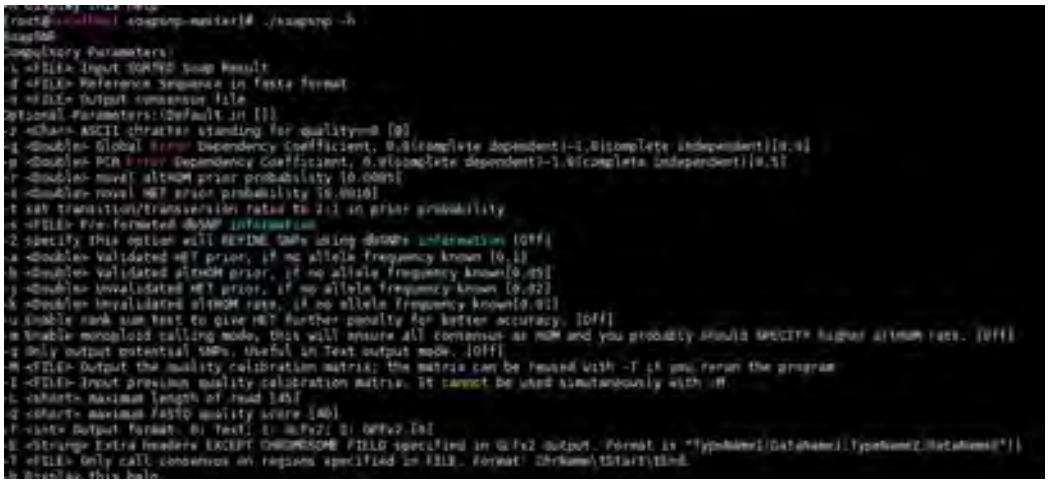
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run SOAPsnp.

```
./soapsnp -h
```

If the following information is displayed, the program is running properly.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.188.8 Change History

Date	Description
2022-05-24	This issue is the first official release.

4.189 trimmomatic 0.33.0 Porting Guide (Kylin V10)

4.189.1 Introduction

A flexible read trimming tool for Illumina NGS data.

For more information, visit the **trimmomatic page at GitHub**.

Programming language: Java

Brief description: A flexible read trimming tool for Illumina NGS data.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

0.33.0

4.189.2 Environment Requirements

Hardware Requirements

Table 4-938 lists the hardware requirements.

Table 4-938 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-939 lists the OS requirements.

Table 4-939 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-940 lists the software requirements.

Table 4-940 Software requirements

Item	Version	Download URL
trimmomatic	0.33.0	https://github.com/timflutre/trimmomatic/archive/refs/heads/master.zip
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

4.189.3 Paths for Software Porting

This chapter lists the software installation paths involved in the trimmomatic software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-941 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/trimmomatic	Installation path of trimmomatic.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.189.4 Configuring the Compilation Environment

4.189.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-942 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing Java	See **4.189.4.2 Installing Java**.

4.189.4.2 Installing Java

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Java.

```
yum install java-1.8.0*
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.189.5 Obtaining the Source Code

Procedure

Step 1 Download the trimmomatic installation package.

https://github.com/timflutre/trimmomatic/archive/refs/heads/master.zip

Step 2 Use an SFTP tool to upload the trimmomatic installation package to the */path/to/trimmomatic* directory on the server.

----End

4.189.6 Compilation and Installation

Procedure

Step 1 Go to the installation package directory.

```
cd /path/to/trimmomatic
```

Step 2 Decompress the installation package.

```
unzip master.zip
cd trimmomatic-master
```

Step 3 Perform the compilation and installation.

```
make -j
make install INSTALL=/path/to/trimmomatic
```

----End

4.189.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/trimmomatic/bin
java -jar trimmomatic.jar
```

If the following information is displayed, the verification is successful.

4.189.8 Change History

Date	Description
2023-05-04	This issue is the first official release.

4.190 tRNAscan-SE 2.0.9 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

4.190.1 Introduction

tRNAscan-SE was written in the Perl script language. Input consists of DNA or RNA sequences in FASTA format. tRNA predictions are output in standard tabular or ACeDB format. tRNAscan-SE does no tRNA detection itself, but instead combines the strengths of three independent tRNA prediction programs by negotiating the flow of information between them, performing a limited amount of post-processing, and outputting the results in one of several formats. For more information, visit the **tRNAscan-SE page at GitHub**.

Programming language: C/Perl

Brief description: A program for detection of tRNA genes.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.0.9

4.190.2 Environment Requirements

Hardware Requirements

Table 4-943 lists the hardware requirements.

Table 4-943 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-944 lists the software requirements.

Table 4-944 Software requirements

Item	Version	Download URL
tRNAscan-SE	2.0.9	**https://github.com/UCSC-LoweLab/tRNAscan-SE/archive/refs/tags/v2.0.9.tar.gz**
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 4-945 lists the OS requirements.

Table 4-945 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

4.190.3 Paths for Software Porting

This chapter lists the software installation paths involved in the tRNAscan-SE software porting.

Table 4-946 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/opt/compiler	Installation path of the BiSheng Compiler.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/tRNAscan-SE	Installation path of tRNAscan-SE.	

4.190.4 Configuring the Compilation Environment

4.190.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 4-947 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **4.190.4.2 Installing the BiSheng Compiler**.

4.190.4.2 Installing the BiSheng Compiler

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the BiSheng Compiler.

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
mkdir -p /opt/compiler
cp -r bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz /opt/compiler
cd /opt/compiler
tar -zxvf bisheng-compiler-2.1.0.1.B002-aarch64-linux.tar.gz
```

Step 3 Set the environment variables of the BiSheng Compiler.

```
export PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/bin:$PATH
export LD_LIBRARY_PATH=/opt/compiler/bisheng-compiler-2.1.0.1.B002-aarch64-linux/lib:$LD_LIBRARY_PATH
export CC=clang CXX=clang++ FC=flang
```

Step 4 Perform the verification.

```
clang -v
```

If the command output contains the BiSheng Compiler version information, the installation is successful.

----End

4.190.5 Obtaining the Source Code

Procedure

Step 1 Download the **tRNAscan-SE-2.0.9.tar.gz** package.

https://github.com/UCSC-LoweLab/tRNAscan-SE/archive/refs/tags/v2.0.9.tar.gz

Step 2 Use an SFTP tool to upload the **tRNAscan-SE-2.0.9.tar.gz** package to the */path/to/tRNAscan-SE* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.190.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd path/to/tRNAscan-SE
```

Step 3 Decompress the package.

```
tar -xvf tRNAscan-SE-2.0.9.tar.gz
```

Step 4 Create an installation directory.

```
mkdir /path/to/tRNAscan-SE/tRNAscan-SE_install
```

Step 5 Perform the compilation.

```
make && make install
```

----End

4.190.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the **bin** installation directory.

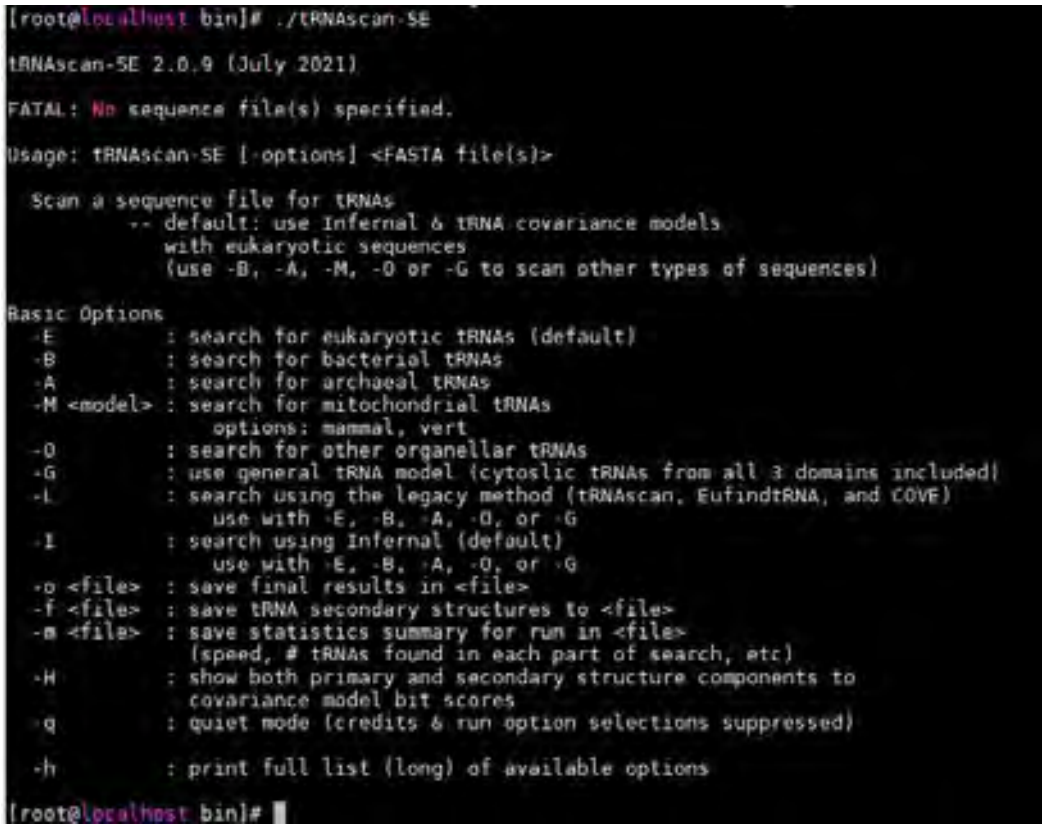
```
cd /path/to/tRNAscan-SE/tRNAscan-SE_install/bin
```

Step 3 Run tRNAscan-SE.

```
./tRNAscan-SE
```

If the following information is displayed, the program is running properly.

 Copyright © Huawei Technologies Co., Ltd.



----End

4.190.8 Change History

Date	Description
2022-05-24	This issue is the first official release.

4.191 vcfSampleCompare 2.013 Porting Guide (Kylin V10)

4.191.1 Introduction

VcfSampleCompare answers the question "What's different?" between my samples. It does this by sorting and filtering the variant records of a VCF file (containing data for 2 or more samples) based on the differences in the variant data between samples.

For more information, visit the **VcfSampleCompare page at GitHub**.

Programming language: Perl

Brief description: It sorts and filters the variant records of a VCF file based on the differences in the variant data between samples.

Open source license: custom open source license

 Copyright © Huawei Technologies Co., Ltd.

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.013

4.191.2 Environment Requirements

Hardware Requirements

Table 4-948 lists the hardware requirements.

Table 4-948 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-949 lists the OS requirements.

Table 4-949 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-950 lists the software requirements.

Table 4-950 Software requirements

Item	Version	Download URL
VcfSampleCompare	2.013	https://github.com/hepcat72/vcfSampleCompare/archive/refs/tags/2.013.tar.gz
gcc	7.3.0	It is built in the OS and does not need to be installed.

 Copyright © Huawei Technologies Co., Ltd.

4.191.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VcfSampleCompare software porting.

Table 4-951 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/VcfSampleCompare	Installation path of VcfSampleCompare.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.191.4 Configuring the Compilation Environment

4.191.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-952 Configuration process

No.	Configuration Item	Remarks
1	Installing Dependencies	See **4.191.4.2 Installing Dependencies**.

4.191.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install -y perl perl-devel perl-CPAN
yum install -y gcc gcc-c++ gcc-gortran
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4.191.5 Obtaining the Source Code

Procedure

Step 1 Download the VcfSampleCompare installation package.

https://github.com/hepcat72/vcfSampleCompare/archive/refs/tags/2.013.tar.gz

Step 2 Use an SFTP tool to upload the VcfSampleCompare installation package to the */path/to/VcfSampleCompare* directory on the server.

----End

4.191.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/VcfSampleCompare
tar -xvf 2.013.tar.gz
cd vcfSampleCompare-2.013
```

Step 2 Compile the file.

```
perl Makefile.PL
make
make install
```

----End

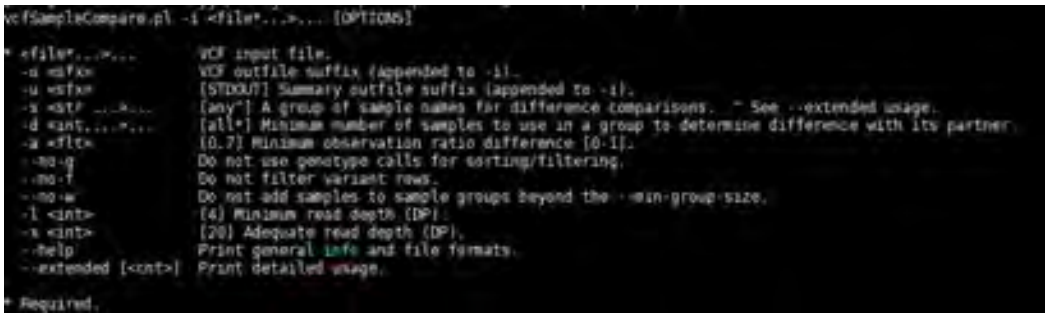
4.191.7 Running and Verification

Procedure

Verify the **vcfSampleCompare.pl** test case.

```
vcfSampleCompare.pl
```

If the following information is displayed, the verification is successful.



4.191.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

4.192 VelvetOptimiser 2.2.6 Porting Guide (Kylin V10)

4.192.1 Introduction

VelvetOptimiser is designed to run as a wrapper script for the Velvet assembler (Zerbino and Birney 2008) and to assist with optimizing the assembly. It searches a supplied hash value range for the optimum, estimates the expected coverage, and then searches for the optimum coverage cutoff. It uses Velvet's internal mechanism for estimating insert lengths for paired end libraries. It can optimize the assemblies by either the default optimization condition or by a user supplied one. It outputs the results to a subdirectory and records all its operations in a log file.

For more information, visit the **VelvetOptimiser page at GitHub**.

Programming language: Perl

Brief description: A wrapper script for the Velvet assembler, to assist with optimizing the assembly.

Open source license: GPL 2.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.2.6

4.192.2 Environment Requirements

Hardware Requirements

Table 4-953 lists the hardware requirements.

Table 4-953 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-954 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-954 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-955 lists the software requirements.

Table 4-955 Software requirements

Item	Version	Download URL
VelvetOptimiser	2.2.6	https://github.com/tseemann/VelvetOptimiser/archive/refs/tags/2.2.6.tar.gz

4.192.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VelvetOptimiser software porting.

Table 4-956 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/VelvetOptimiser	Installation path of VelvetOptimiser.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.192.4 Configuring the Compilation Environment

4.192.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

The server must be connected to the Internet.

Configuration Process

Table 4-957 Configuration process

No.	Configuration Item	Remarks
1	Installing Perl	See **4.192.4.2 Installing Perl**.
2	Installing Velvet	See **4.192.4.3 Installing Velvet**.

4.192.4.2 Installing Perl

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://www.cpan.org/src/5.0/perl-5.34.0.tar.gz
```

Step 2 Decompress the installation package and perform the installation.

```
tar xvf perl-5.34.0.tar.gz
cd perl-5.34.0
./Configure -des -Dprefix=/path/to/perl -Dusethreads -Uinstalluserbinperl -Dcc=gcc
make
make test
make install
```

Step 3 Set the environment variables.

```
export PATH=/path/to/perl/bin:$PATH
export PERL5LIB=/path/to/perl/lib/site_perl/5.34.0
```

Step 4 Configure the Perl source.

```
cpan
o conf urllist unshift http://mirrors.163.com/cpan/
o conf commit
quit
```

If the following information is displayed, the configuration is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost VelvetOptimiser-2.2.6]# cpan
Loading internal logger. Log::Log4perl recommended for better logging
Terminal does not support AddHistory.

To fix that, maybe try>  install Term::ReadLine::Perl


cpan shell -- CPAN exploration and modules installation (v2.28)
Enter 'h' for help.

cpan[1]> o conf urllist unshift http://mirrors.163.com/cpan/
Please use 'o conf commit' to make the config permanent!


cpan[2]> o conf commit
commit: wrote '/root/.cpan/CPAN/MyConfig.pm'

cpan[3]> quit
Terminal does not support GetHistory.
Lockfile removed.
```

----End

4.192.4.3 Installing Velvet

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10 -O velvet-1.2.10.tar.gz
```

Step 2 Decompress the installation package and perform the installation.

```
tar xvf velvet-1.2.10.tar.gz
cd velvet-1.2.10
export OMP_NUM_THREADS=96
export OMP_THREAD_LIMIT=95
sed -ri 's/\-m64/\-mabi=lp64/g' Makefile
make 'GATEGORIES=10' 'MAXKMERLENGTH=57' 'LONGSEQUENCES=1' 'OPENMP=1' 'BUNDLEDZLIB=1'
```

Step 3 Set the environment variables.

```
export PATH=/path/to/velvet-1.2.10:$PATH
```

----End

4.192.5 Obtaining the Source Code

Procedure

Step 1 Download the VelvetOptimiser installation package.

https://github.com/tseemann/VelvetOptimiser/archive/refs/tags/2.2.6.tar.gz

Step 2 Use an SFTP tool to upload the VelvetOptimiser installation package to the */path/to/VelvetOptimiser* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

4.192.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/VelvetOptimiser
tar -xvf 2.2.6.tar.gz
```

Step 2 Install the module.

```
yum install libdb libdb-devel
cpan App::cpanminus
cpanm Bio/SeqIO.pm
```

Step 3 Set the environment variables.

```
export PATH=/path/to/VelvetOptimiser/VelvetOptimiser-2.2.6/bin:$PATH
```

----End

4.192.7 Running and Verification

Procedure

Perform the verification.

```
./VelvetOptimiser.pl --help
```

If the following information is displayed, the verification is successful.

4.192.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

4.193 wtdbg 1.1.006 Porting Guide (Kylin V10)

4.193.1 Introduction

wtdbg is designed to assemble huge genomes in a very limited time. It requires a PowerPC with multiple cores and a large RAM (more than 1 TB). wtdbg can assemble a 100 X human pacbio dataset within one day.

For more information, visit the **wtdbg page at GitHub**.

Programming language: C

Brief description: A fuzzy Bruijn graph (FBG) approach to long noisy reads assembly.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.1.006

4.193.2 Environment Requirements

Hardware Requirements

Table 4-958 lists the hardware requirements.

Table 4-958 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-959 lists the OS requirements.

Table 4-959 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 4-960 lists the software requirements.

Table 4-960 Software requirements

Item	Version	Download URL
wtdbg	1.1.006	https://github.com/ruanjue/wtdbg.git
GCC	7.3.0	You do not need to download it. Use yum to install it.

4.193.3 Paths for Software Porting

This chapter lists the software installation paths involved in the wtdbg software porting.

Table 4-961 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/wtdbg	Installation path of wtdbg.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.193.4 Configuring the Compilation Environment

4.193.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-962 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC	See 4.193.4.2 Installing GCC.

 Copyright © Huawei Technologies Co., Ltd.

4.193.4.2 Installing GCC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC using yum.

```
yum install gcc gcc-c++ gcc-gfortran
```

----End

4.193.5 Obtaining the Source Code

Procedure

Download the source code.

```
cd /path/to/wtdbg
git clone https://github.com/ruanjue/wtdbg.git
```

4.193.6 Compilation and Installation

Procedure

Step 1 Download the sse2neon header file.

```
cd /path/to/wtdbg
wget https://github.com/jratcliff63367/sse2neon/archive/master.zip -O sse2neon-master.zip
unzip sse2neon-master.zip
mv ./sse2neon-master/SSE2NEON.h ./sse2neon-master/sse2neon.h
```

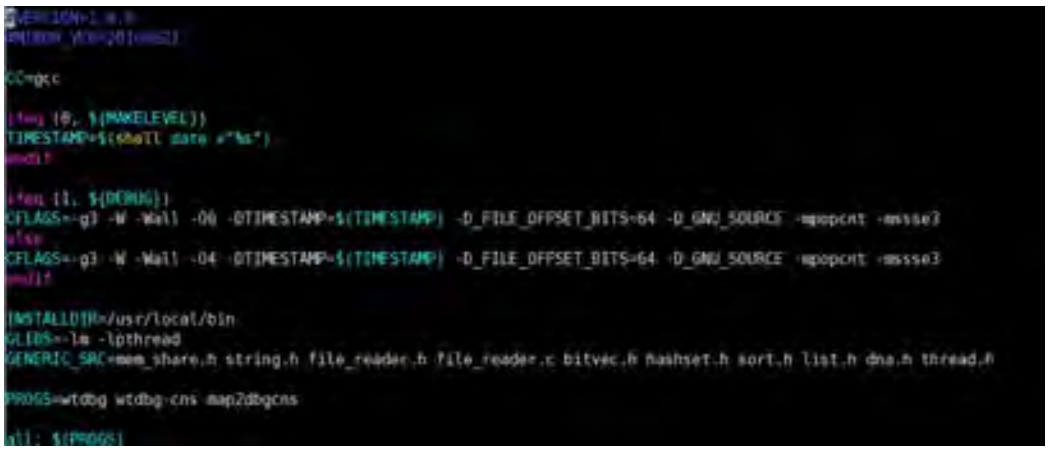
Step 2 Import the sse2neon header file to the installation directory.

```
cp ./sse2neon-master/sse2neon.h ./wtdbg
```

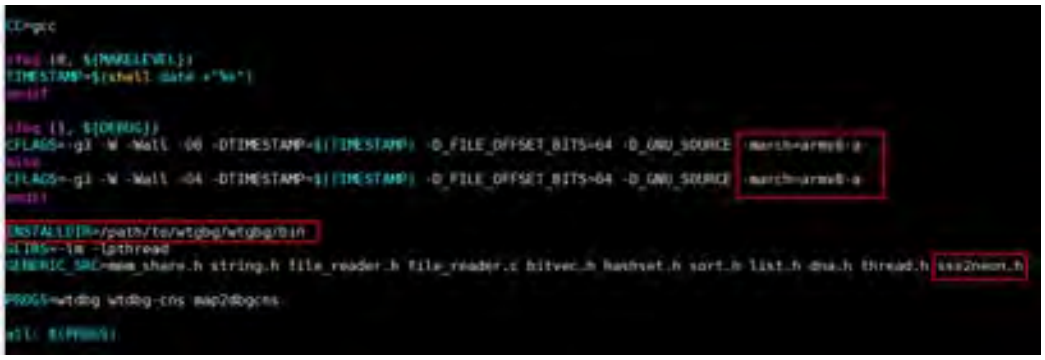
Step 3 Modify **Makefile**.

```
vim Makefile
```

Before:



After:



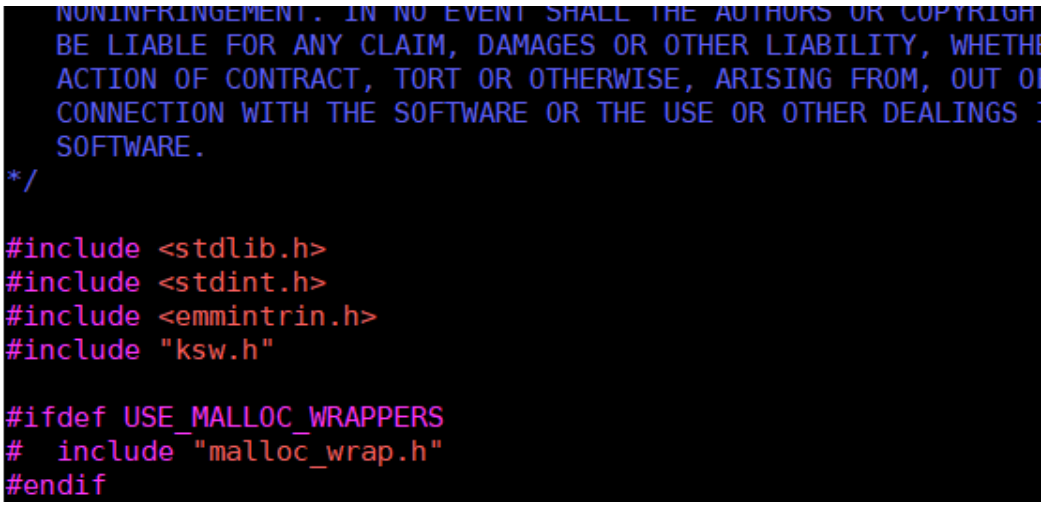
NOTE

Replace the example path **/path/to/wtdbg** with the actual one.

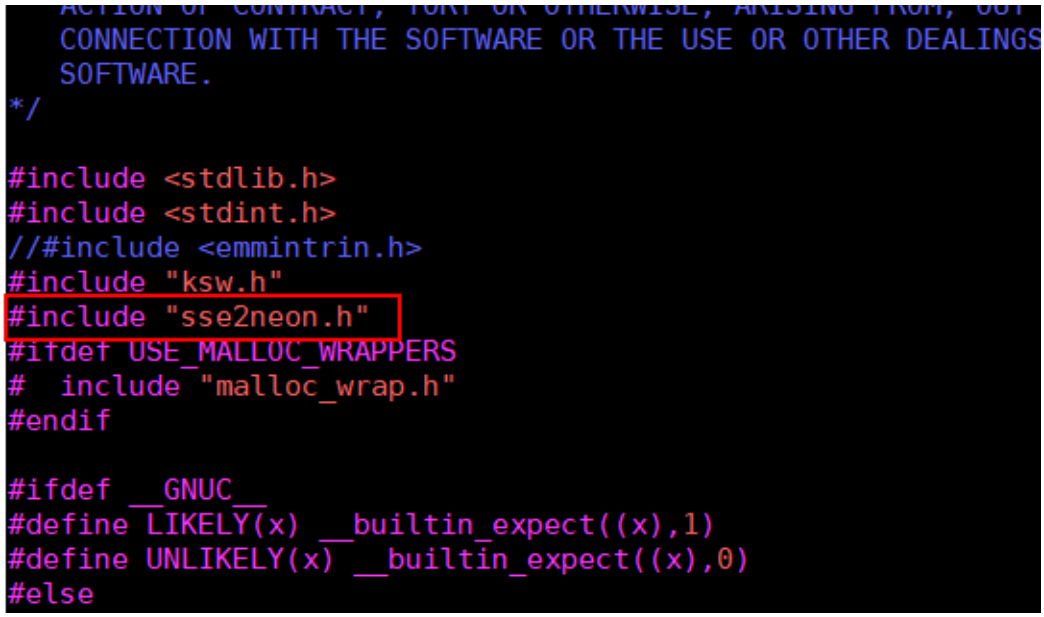
Step 4 Modify **ksw.c**.

```
vim ksw.c
```

Before:



After:



Step 5 Create an installation directory.

```
mkdir bin
```

Step 6 Perform the installation.

```
make
make install
```

----End

4.193.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/wtdbg/wtdbg/bin
./wtdbg version
```

If the software version **1.1.006** is displayed in the command output, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost bin]# ./wtdbg version
--
-- total memory    534578752.000 kB
-- available       524969856.000 kB
-- 128 cores
-- Starting program: ./wtdbg version
-- pid                   4006262
--
WTDBG: De novo assembler for long noisy sequences
Author: Jue Ruan <ruanjue@gmail.com>
Version: 1.1.006
Compiled: 1676975595
Usage: wtdbg [options]
Options:
 -t <int>    Number of threads, 0: all cores, [1]
 -i <string> Long reads sequences file, + *
 -I <string> Accurate contig sequences file, +
 -r          Search nodes on accurate contig sequences only
 -T <int>    Limitation of total base-pairs in Mb, usually for test [0]
 -R          Rapid subsampling instead of aligning all intervals
 -j <int>    Expected length of node, [1000] bp
 -J <int>    Distance of next node start, 0: the same as <-j> [0] bp
 -X <float>  Max overlapped fraction of two adjacent nodes in a read, [0
 -e <int>    Min cov of edges, [3]
```

4.193.8 Change History

Date	Description
2023-04-23	This issue is the first official release.

4.194 wtdbg2 2.5 Porting Guide (Kylin V10)

4.194.1 Introduction

wtdbg2 is a de novo sequence assembler for long noisy reads produced by PacBio or Oxford Nanopore Technologies (ONT). It assembles raw reads without error correction and then builds the consensus from intermediate assembly output. wtdbg2 is able to assemble the human and even the 32Gb Axolotl genome at a speed tens of times faster than CANU and FALCON while producing contigs of comparable base accuracy.

For more information, visit the **wtdbg2 page at GitHub**.

Programming language: C

Brief description: A fuzzy Bruijn graph (FBG) approach to long noisy reads assembly.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

2.5

4.194.2 Environment Requirements

Hardware Requirements

Table 4-963 lists the hardware requirements.

Table 4-963 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 4-964 lists the OS requirements.

Table 4-964 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 4-965 lists the software requirements.

Table 4-965 Software requirements

Item	Version	Download URL
wtdbg2	2.5	https://github.com/ruanjue/wtdbg2.git

4.194.3 Paths for Software Porting

This chapter lists the software installation paths involved in the wtdbg2 software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 4-966 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/wtdbg2	Installation path of wtdbg2.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.194.4 Configuring the Compilation Environment

4.194.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-967 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC	See **4.194.4.2 Installing GCC**.

4.194.4.2 Installing GCC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install GCC using yum.

```
yum install gcc gcc-c++ gcc-gfortran
```

----End

4.194.5 Obtaining the Source Code

Procedure

Download the source code.

```
cd /path/to/wtdbg2
git clone https://github.com/ruanjue/wtdbg2.git
```

 Copyright © Huawei Technologies Co., Ltd.

4.194.6 Compilation and Installation

Procedure

Step 1 Download the sse2neon header file.

```
cd /path/to/wtdbg2
wget https://github.com/jratcliff63367/sse2neon/archive/master.zip -O sse2neon-master.zip
unzip sse2neon-master.zip
mv ./sse2neon-master/SSE2NEON.h ./sse2neon-master/sse2neon.h
```

Step 2 Import the sse2neon header file to the installation directory.

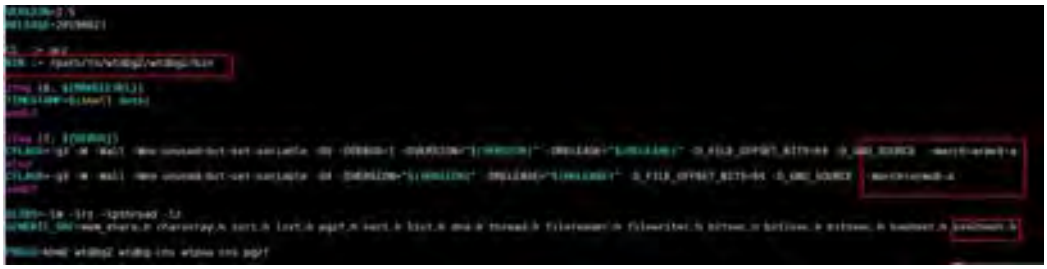
```
cp ./sse2neon-master/sse2neon.h ./wtdbg2
```

Step 3 Modify **Makefile**.

```
vim Makefile
```

Before:

After:



NOTE

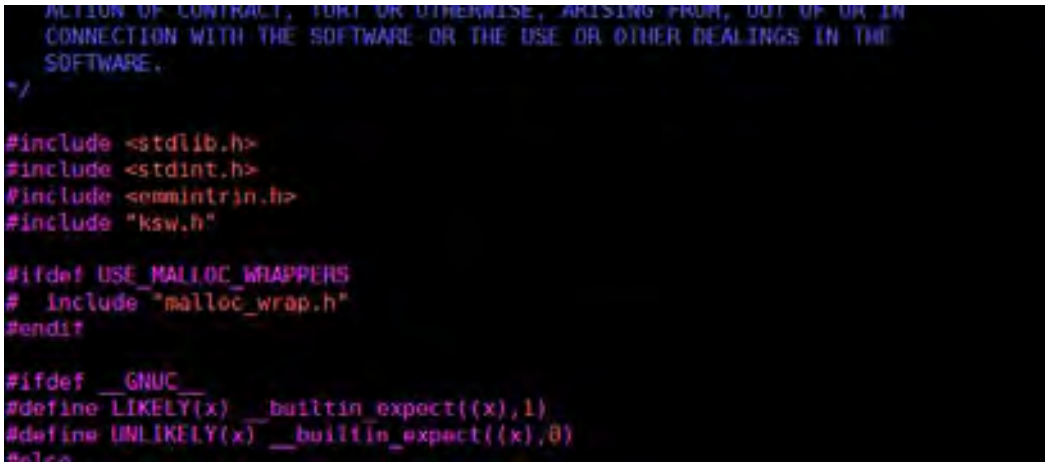
Replace the example path **/path/to/wtdbg2** with the actual one.

Step 4 Modify **ksw.c**.

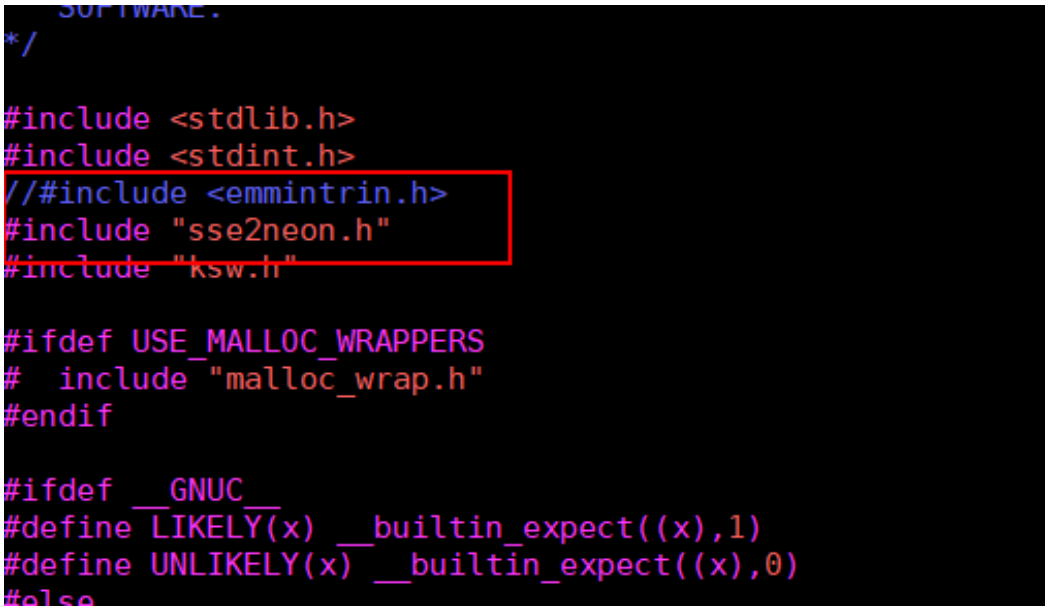
```
vim ksw.c
```

Before:

 Copyright © Huawei Technologies Co., Ltd.



After:



Step 5 Modify **poacns.h**.

```
vim poacns.h
```

Before:

```
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#ifndef PO_MSA_CNS_RJ_H
#define PO_MSA_CNS_RJ_H

#include "bit2vec.h"
#include "dna.h"
#include "chararray.h"
#include "list.h"
#include "hashset.h"
#include <emmintrin.h>
#include <tmmintrin.h>

#if __BYTE_ORDER == 1234
//#pragma message(" ** " __FILE__ " has been tested in LITTLE_ENDIAN **\n")
#else
#pragma message(" ** " __FILE__ " hasn't been tested in BIG_ENDIAN **\n")
#endif
static int cns_debug = 0;
```

```
        seqlen = g->seqs->rdlens->buffer[rid];
        seqlex = roundup_times(seqlen, 8); // 128 = 8 * sizeof(RJ1)
        slen = seqlex >> 3;
        seqinc = seqlex + 8; // seqlex, max_id, beg, end, and paddings
#if __BYTE_ORDER == 1234
        SMASK = _mm_setr_epi8(0, 2, 4, 6, 8, 10, 12, 14, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80);
#else
        // TODO: need to test in BIG_ENDIAN
        SMASK = _mm_setr_epi8(1, 3, 5, 7, 9, 11, 13, 15, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80);
#endif
        // set query prof
        // fit buffer to 16 bytes aligned
```

After:

```
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#ifndef PO_MSA_CNS_RJ_H
#define PO_MSA_CNS_RJ_H

#include "bit2vec.h"
#include "dna.h"
#include "chararray.h"
#include "list.h"
#include "hashset.h"
//#include <emmintrin.h>
//#include <tmmintrin.h>
#include "sse2neon.h"
#if __BYTE_ORDER == 1234
//#pragma message(" ** " __FILE__ " has been tested in LITTLE_ENDIAN **\n")
#else
#pragma message(" ** " __FILE__ " hasn't been tested in BIG_ENDIAN **\n")
#endif
static int cns_debug = 0;
```

```
        u4i i, j;
        segoff = g->seqs->rdoffs->buffer[rid];
        seqlen = g->seqs->rdlens->buffer[rid];
        seqlex = roundup_times(seqlen, 8); // 128 = 8 * sizeof(RJ1)
        slen = seqlex >> 3;
        seqinc = seqlex + 8; // seqlex, max_id, beg, end, and paddings
#if __BYTE_ORDER == 1234
//      SMASK = _mm_setr_epi8(0, 2, 4, 6, 8, 10, 12, 14, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80);
#else
        // TODO: need to test in BIG_ENDIAN
//      SMASK = _mm_setr_epi8(1, 3, 5, 7, 9, 11, 13, 15, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80);
#endif
        // set query prof
        // fit buffer to 16 bytes aligned
        set_rd_query_prof(g, rid);
        prepare_rd_align_pog(g, rid);
        clear_u4v(g->stack);
```

Step 6 Create an installation directory.

```
mkdir bin
```

Step 7 Perform the installation.

```
make
make install
```

----End

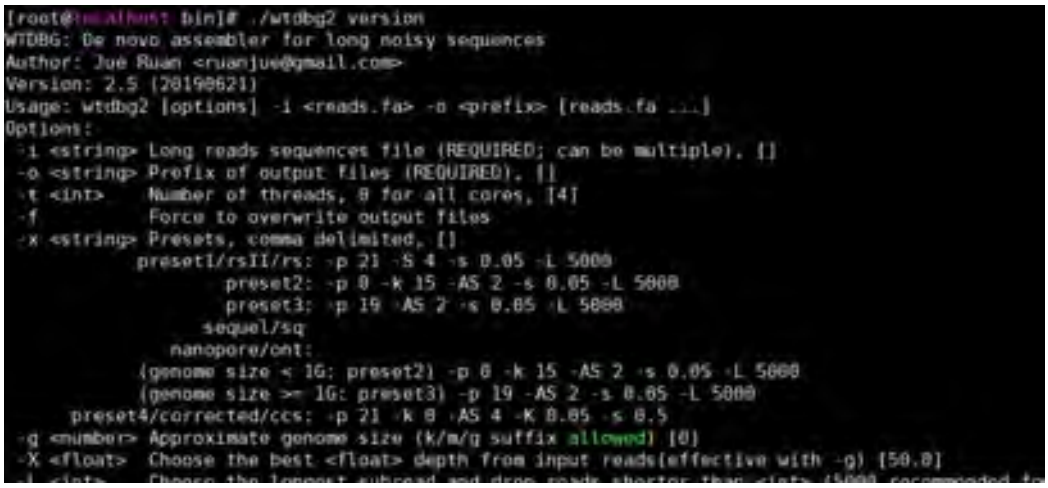
4.194.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/wtdbg2/wtdbg2/bin
./wtdbg2 version
```

If the software version **2.5** is displayed in the command output, the installation is successful.



4.194.8 Change History

Date	Description
2023-04-23	This issue is the first official release.

4.195 xpressplot 0.2.5 Porting Guide (Kylin V10)

4.195.1 Introduction

xpressplot is a toolkit for navigating and analyzing gene expression datasets.

For more information, visit the **xpressplot website**.

Programming language: Python

One-sentence description: A toolkit for navigating and analyzing gene expression datasets.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: GPL 3.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

0.2.5

4.195.2 Environment Requirements

Hardware Requirements

Table 4-968 lists the hardware requirements.

Table 4-968 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 4-969 lists the software requirements.

Table 4-969 Software requirements

Item	Version	Download URL
Python	2.7 or later	https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz

OS Requirements

Table 4-970 lists the OS requirements.

Table 4-970 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

4.195.3 Paths for Software Porting

This chapter lists the software installation paths involved in the xpressplot software porting.

Table 4-971 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/XPRESSPLOT	Path for deploying the virtual environment.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

4.195.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 4-972 Configuration process

No.	Configuration Item	Remarks
1	Installing Python	The built-in Python of the OS is used. No additional deployment is required. If you want to deploy it manually, see section "Installing Python 2" in **SNPhylo 20180901 Porting Guide (Kylin V10)**.

4.195.5 Obtaining the Source Code

Use pip for the installation. You do not need to download the source code.

 Copyright © Huawei Technologies Co., Ltd.

4.195.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 (Optional) Configure the virtual environment.

```
pip install virtualenv
cd /path/to/XPRESSPLOT
virtualenv xpressplot
source /path/to/XPRESSPLOT/xpressplot/bin/activate
```

Step 3 Install xpressplot.

```
pip install xpressplot==0.2.5
```

If the following information is displayed, the installation is successful.

```
Installing collected packages: xpressplot
    Running setup.py install for xpressplot ... done
Successfully installed xpressplot-0.2.5
```

----End

4.195.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Perform the verification.

```
python
import XPRESSplot
```

----End

4.195.8 Change History

Date	Description
2022-2-17	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

5 Other

5.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)
5.2 SGE 8.1.9 Porting Guide (CentOS 7.6)
5.3 Conda 4.3.16 Porting Guide (CentOS 7.6)
5.4 NPB 3.4 Porting Guide (CentOS 7.6)
5.5 Double-conversion 3.2.0 Porting Guide (Kylin V10)
5.6 LIBPNG 1.6.37 Porting Guide (Kylin V10)
5.7 Linenoise-NG 1.0.1 Porting Guide (Kylin V10)
5.8 NCO 5.1.0 Porting Guide (Kylin V10)
5.9 PETSc 3.16 Porting Guide (Kylin V10)
5.10 Chainer 7.8.1 Porting Guide (Kylin V10)
5.11 Horovod 0.25.0 Porting Guide (Kylin V10)
5.12 Keras 2.9.0 Porting Guide (Kylin V10)
5.13 Lasagne 0.1 Porting Guide (Kylin V10)
5.14 mpi4py 3.1.3 Porting Guide (Kylin V10)
5.15 MXNet 2.0.0 Porting Guide (Kylin V10)
5.16 sparsehash 2.0.4 Porting Guide (Kylin V10)
5.17 SUNDIALS 6.3.0 Porting Guide (Kylin V10)
5.18 Theano 1.0.5 Porting Guide (Kylin V10)
5.19 Thrust 1.12 Porting Guide (Kylin V10)
5.20 Tinker 8.10.3 Porting Guide (Kylin V10)
5.21 west Porting Guide (Kylin V10)
5.22 Crystal 1.7.2 Porting Guide (Kylin V10)
5.23 GRIB API 1.21.0 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

5.24 IPM 2.0.6 Porting Guide (Kylin V10)

5.25 ncview 2.1.5 Porting Guide (Kylin V10)

5.26 paddle 2.3.1 Porting Guide (Kylin V10)

5.27 ploticus 242 Porting Guide (Kylin V10)

5.28 Rosetta Porting Guide (Kylin V10)

5.29 SimpleITK Master Porting Guide (openEuler 20.03)

5.30 Singularity 3.9.9 Porting Guide (Kylin V10)

5.31 szip 2.1.1 Porting Guide (Kylin V10)

5.32 VTK 7.1.1 Porting Guide (openEuler 20.03)

5.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)

5.1.1 Introduction

Hadoop is an Apache open source framework written in Java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

For more information, visit the **official Hadoop website**.

Programming language: Java

Brief description: open source code framework

Open source license: Apache 2.0

5.1.2 Environment Requirements

Hardware Requirements

Table 5-1 lists the hardware requirements.

Table 5-1 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 5-2 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-2 Software Requirements

Item	Version	How to Obtain
Hadoop	3.1.2	**https://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2.tar.gz**
Spark	2.4.4	**https://spark.apache.org/releases/spark-release-2-4-4.html**
Scala	2.12.4	**https://www.scala-lang.org/download/2.12.4.html**
JDK	1.8.0_171	**https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**

OS Requirements

Table 5-3 lists the OS requirements.

Table 5-3 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

5.1.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Hadoop software porting.

Table 5-4 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	*/path/to/HADOOP*	Installation path of Hadoop	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.1.4 Configuring the Compilation Environment

5.1.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-5 Configuration process

No.	Configuration Item	Remarks
1	Installing the Java environment	See **5.1.4.2 Installing the Java Environment**.

5.1.4.2 Installing the Java Environment

Procedure

Step 1 Obtain the Java package **jdk-8u152-linux-arm64-vfp-hflt.tar.gz**. For details, see the **software requirements**.

Step 2 Use PuTTY to log in to the server as the **root** user.

Step 3 Run the following commands to install the Java environment:

```
mkdir /path/to/HADOOP
cd path/to/HADOOP
tar -zxvf jdk-8u152-linux-arm64-vfp-hflt.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

Differences between JRE and JDK:

- The Java Runtime Environment (JRE) is the environment required for Java running.
- The Java Development Kit (JDK) includes the JRE and the tools and class libraries required for Java development.

Step 4 Configure environment variables and modify the **/root/.bashrc** file.

1. Open the file.
   ```
   vi /root/.bashrc
   ```
2. Press **i** to enter the insert mode and add the following path to the end of the **/root/.bashrc** file:
   ```
   export JAVA_HOME=path/to/HADOOP/jdk1.8.0_171
   export JRE_HOME=$JAVA_HOME/jre
   export PATH=:$JAVA_HOME/bin:$PATH
   export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

----End

5.1.5 Obtaining the Source Code

Procedure

Step 1 Download the Hadoop installation package **hadoop-3.1.2.tar.gz**.

Download address: **https://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2.tar.gz**

Step 2 Use SFTP to upload the Hadoop installation package to the */path/to/HADOOP* directory on the server.

----End

5.1.6 Compiling and Installing Hadoop and Spark

5.1.6.1 Configuring Hadoop

Hadoop can run on a single node in pseudo-distributed mode. The Hadoop process runs as a separated Java process. The node functions as both the NameNode and DataNode and reads files in the HDFS.

The configuration file is stored in **hadoop-3.1.2/etc/hadoop**. For a pseudo-distributed cluster, you need to modify the **core-site.xml** and **hdfs-site.xml** configuration files. The Hadoop configuration file is in XML format. Each configuration is implemented by declaring the name and value of property.

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to switch to the directory, in which the Hadoop installation package is stored:

```
cd path/to/HADOOP
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Run the following command to decompress the Hadoop installation package:

```
tar -xvf hadoop-3.1.2.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hadoop-3.1.2
```

Step 5 Run the following commands to create the following four folders in the **hadoop-3.1.2** folder as the HDFS file paths:

```
mkdir hdfs
mkdir hdfs/tmp
mkdir hdfs/name
mkdir hdfs/data
```

Step 6 Run the following command to go to the directory in which the configuration file is stored:

```
cd etc/hadoop/
```

Step 7 Run the following command to modify the **core-site.xml** file:

1. Open the file.
   ```
   vi core-site.xml
   ```
2. Press **i** to enter the insert mode and edit the **core-site.xml** file.

   Before the modification:

   ```
   <configuration>
   </configuration>
   ```

   After the modification:

   ```
   <configuration>
    <property>
         <name>hadoop.tmp.dir</name>
         <value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/tmp</value>
         <description>Abase for other temporary directories.</description>
    </property>
    <property>
         <name>io.file.buffer.size</name>
         <value>4096</value>
    </property>
    <property>
         <name>fs.defaultFS</name>
         <value>hdfs://armnode2:9000</value>
    </property>
    <property>
         <name>fs.trash.interval</name>
         <value>1440</value>
    </property>
   </configuration>
   ```

   NOTE

   armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the **hostname** command to query the hostname of the installation environment.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to modify the **hdfs-site.xml** configuration file:

1. Open the file.
   ```
   vi hdfs-site.xml
   ```
2. Press **i** to enter the insert mode and edit the **hdfs-site.xml** file.

   Before the modification:

```
<configuration>
</configuration>
```

After the modification:

```
<configuration>
  <property>
<name>dfs.namenode.http-address</name>
<value>armnode2:50070</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/name</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/data</value>
</property>
<property>
<name>dfs.webmnt.enabled</name>
<value>true</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
</configuration>
```

NOTE

armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the **hostname** command to query the hostname of the installation environment.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Run the following commands to modify the **workers** configuration file:

1. Open the file.
   ```
   vi workers
   ```
2. Press **i** to enter the insert mode and edit the **workers** file to add the hostname of the installation environment.
   ```
   armnode2
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 10 Run the following commands to create the **master** and **slaves** configuration files:

```
cd /path/to/HADOOP/hadoop-3.1.2/etc/hadoop
cp workers master
cp workers slaves
```

----End

5.1.6.2 Configuring Spark

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to download Scala.

```
cd /path/to/HADOOP
wget https://downloads.lightbend.com/scala/2.12.4/scala-2.12.4.tgz
tar xvf scala-2.12.4.tgz
```

Step 3 Run the following command to decompress the Spark installation package:

```
tar -xvf  spark-2.4.4-bin-hadoop2.7.tgz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/HADOOP/spark-2.4.4-bin-hadoop2.7/conf
```

Step 5 Run the following commands to modify the **spark-defaults.conf** configuration file:

```
mv spark-defaults.conf.template spark-defaults.conf
vi spark-defaults.conf
```

1. Press **i** to enter the insert mode and add the following content to the **spark-defaults.conf** file:

```
spark.master                    spark://armnode2:7077
spark.scheduler.mode            FAIR
spark.eventLog.enabled          true
spark.eventLog.dir              hdfs://armnode2:9000/sparklog
spark.shuffle.consolidateFiles  true
spark.shuffle.manager           SORT
spark.sql.hive.convertMetastoreOrc false
```

   NOTE

   armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the **hostname** command to query the hostname of the installation environment.

2. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Run the following commands to modify the **slaves** configuration file:

```
mv slaves.template slaves
vi slaves
```

1. Press **i** to enter the insert mode and edit the **slaves** file to add the hostname of the installation environment.

```
armnode2
```

2. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 7 Run the following commands to modify the **spark-env.sh** configuration file:

```
mv spark-env.sh.template spark-env.sh
vi spark-env.sh
```

1. Press **i** to enter the insert mode and add the following content to the **spark-env.sh** file:

```
export JAVA_HOME=/path/to/HADOOP/jdk1.8.0_171
export SCALA_HOME=/path/to/HADOOP/scala-2.12.4
export SPARK_HOME=/path/to/HADOOP/spark-2.4.4-bin-hadoop2.7
export SPARK_MASTER_IP=armnode2
export HADOOP_HOME=/path/to/HADOOP/hadoop-3.1.2
export HADOOP_CONF_DIR=/path/to/HADOOP/hadoop-3.1.2/etc/hadoop
export SPARK_DIST_CLASSPATH=$(/path/to/HADOOP/hadoop-3.1.2/bin/hadoop classpath)
export SPARK_DRIVER_MEMORY=30g
export SPARK_WORKER_INSTANCES=10
export SPARK_WORKER_CORES=16
export SPARK_WORKER_MEMORY=20g
export SPARK_EXECUTOR_MEMORY=10g
export SPARK_LOCAL_DIRS=/path/to/HADOOP/hadoop-3.1.2/hdfs/spark/tmp
export SPARK_WORKER_DIR=/path/to/HADOOP/hadoop-3.1.2/hdfs/spark/work
```

 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> **armnode2** indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the **hostname** command to query the hostname of the installation environment.

2. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 8 Run the following command to go to the directory in which the configuration file is stored:

```
cd /path/to/HADOOP
```

Step 9 Run the following command to configure environment variables:

1. Open the file.
   ```
   vi env.sh
   ```
2. Press **i** to enter the insert mode, create an environment variable file, and add the following content to the file:
   ```
   export JAVA_HOME=/path/to/HADOOP/jdk1.8.0_171
   export JRE_HOME=$JAVA_HOME/jre
   export PATH=:$JAVA_HOME/bin:$PATH
   export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
   export HADOOP_HOME=/path/to/HADOOP/hadoop-3.1.2
   export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
   export HDFS_DATANODE_USER=root
   export HDFS_NAMENODE_USER=root
   export HDFS_SECONDARYNAMENODE_USER=root
   export YARN_RESOURCEMANAGER_USER=root
   export YARN_NODEMANAGER_USER=root
   export SCALA_HOME=/path/to/HADOOP/scala-2.12.4
   export PATH=$SCALA_HOME/bin:$PATH
   export SPARK_HOME=/path/to/HADOOP/spark-2.4.4-bin-hadoop2.7
   export PATH=$SPARK_HOME/bin:$PATH
   ```

   > **NOTE**
   >
   > Hadoop configuration file description:
   >
   > The running mode of Hadoop is determined by the configuration file read when Hadoop is running. Therefore, if you need to switch from the pseudo-distributed mode to the non-distributed mode, you need to delete the configuration items in the **core-site.xml** file.
   >
   > In addition, pseudo-distributed Hadoop can run after **fs.defaultFS** and **dfs.replication** are configured (as described in the official tutorial). However, if **hadoop.tmp.dir** is not configured, the default temporary directory is **/tmp/hadoo-hadoop**, the directory may be deleted by the system during the restart. As a result, you must run the format command again. Therefore, you need to specify **dfs.namenode.name.dir** and **dfs.datanode.data.dir**. Otherwise, errors may occur in the following steps.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
4. Run the following command to make the environment variables take effect:
   ```
   source env.sh
   ```

Step 10 After the configuration is complete, format the NameNode.

```
hdfs namenode -format
```

If the following information is displayed, the process ends:

```
/root/GATK/hadoop-3.1.2/hdfs/name has been successfully formatted.
```

Step 11 Run the following commands to start the NameNode and DataNode daemon processes:

```
start-dfs.sh
jps
```

 Copyright © Huawei Technologies Co., Ltd.

Run the **jps** command to check whether the NameNode, DataNode, and SecondaryNameNode processes are successfully started. If they are successfully started, the processes shown in **Figure 5-1** are displayed. If the SecondaryNameNode process is not started, run the **sbin/stop-dfs.sh** command to stop the process and try again. If NameNode or DataNode is not started, the configuration fails. Check the previous steps or view the startup logs to locate the fault.

Figure 5-1 Example

```
[root@armnode2 hadoop-3.1.2]# jps
16081 NameNode
16295 DataNode
16666 SecondaryNameNode
28926 Jps
```

Step 12 Run the following command to go to the Spark directory:

```
cd /path/to/HADOOP/spark-2.4.4-bin-hadoop2.7/sbin
```

Step 13 Run the following command to start the Spark process:

```
./start-all.sh
jps
```

Run the **jps** command to check whether the Spark process is successfully started. If the Spark process is successfully started, multiple Worker processes are displayed, as shown in **Figure 5-2**.

Figure 5-2 Example

```
[root@armnode2 sbin]# jps
18113 Worker
19777 Worker
16295 DataNode
17832 Worker
19500 Worker
16081 NameNode
17201 Master
20052 Worker
18677 Worker
19225 Worker
16666 SecondaryNameNode
17563 Worker
20380 Jps
18942 Worker
18399 Worker
```

----End

5.1.7 Running and Verifying Hadoop and Spark

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to create a user directory in the HDFS:

```
cd path/to/HADOOP
./bin/hdfs dfs -mkdir -p /user/hadoop
```

 Copyright © Huawei Technologies Co., Ltd.

NOTE

The directories and files created by HDFS DFS operations can be identified only in Hadoop mode. They cannot be viewed by running the local **ls** command.

Step 3 Run the following commands to create the **input** directory and copy the files in **etc/hadoop** to the **input** directory:

```
hdfs dfs -mkdir -p input
hdfs dfs -put ./etc/hadoop/*.xml input
```

Step 4 Run the following command to view the file list:

```
hdfs dfs -ls input
```

Step 5 Run the following command to stop Hadoop:

```
stop-dfs.sh
```

----End

5.1.8 Troubleshooting

Problem 1: An Error Is Reported When hdfs namenode -format Is Run

Symptom

The error message "Error: JAVA_HOME is not set and could not be found" is displayed when **hdfs namenode -format** is run.

Possible Causes

The **JAVA_HOME** environment variable is incorrectly set.

Procedure

Set the **JAVA_HOME** variable by referring to **Configuring the Compilation Environment**.

If **JAVA_HOME** is correctly set, modify the **./etc/hadoop/hadoop-env.sh** file.

1. Open the file.
   ```
   vi ./etc/hadoop/hadoop-env.sh
   ```
2. Press **i** to enter the insert mode and add the following environment variable to the environment variable file:
   ```
   expect JAVA_HOME=JDK installation path
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Problem 2: An Error Is Reported When hdfs namenode -format Is Run

Symptom

The error message "Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password)" is displayed when **hdfs namenode -format** is run.

Possible Causes

The password-free function of the local host is faulty.

Procedure

Run the **ssh-keygen -t rsa** command to enable password-free function again.

 Copyright © Huawei Technologies Co., Ltd.

Problem 3: "Could not resolve hostname" Displayed When Hadoop Is Started

Symptom

The error message "ssh: Could not resolve hostname xxx" is displayed when Hadoop is started.

Procedure

You can set the Hadoop environment variable to resolve this problem.

Step 1 Press **Ctrl+C** to stop the startup.

Step 2 Add the following two lines to **~/.bashrc**:

1. Open the file.
   ```
   vi ~/.bashrc
   ```
2. Press **i** to enter the insert mode and add the following commands to **~/.bashrc**:
   ```
   export HADOOP_HOME=/usr/local/hadoop
   export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
   ```

   NOTE

   The setting process is the same as that of the **JAVA_HOME** variable. In the variable, **HADOOP_HOME** indicates the Hadoop installation directory.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Run the following command for the setting to take effect:

```
source ~/.bashrc
```

Step 4 Run the following command to start Hadoop:

```
start-dfs.sh
```

----End

5.1.9 More Information

Official Hadoop website:

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Prepare_to_Start_the_Hadoop_Cluster

5.2 SGE 8.1.9 Porting Guide (CentOS 7.6)

5.2.1 Introduction

Sun Grid Engine (SGE) is a grid-based resource management software developed by Sun. It receives jobs submitted by users and uses resource management policies to schedule jobs to run on appropriate systems in the grid. Users can submit batch, interactive, and parallel jobs to the grid. It is an advanced resource management tool used to manage different types of distributed computing environments.

The SGE software consists of three modules: sge_qmaster, sge_schedd, and sge_execd. The servers in the cluster are classified into the master hosts,

 Copyright © Huawei Technologies Co., Ltd.

administration hosts, execution hosts, and submission hosts, depending on the module running on them.

For more information, visit the **official SGE website**.

Programming language: C/Java

Brief description: scheduler software based on grid resource management

Open source license: Sun Industry Standards Source License (SISSL)

Recommended Software Version

SGE 8.1.9

5.2.2 Environment Requirements

Hardware Requirements

Table 5-6 lists the hardware requirements.

Table 5-6 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 5-7 lists the software requirements.

Table 5-7 Software requirements

Item	Version	Download URL
SGE	8.1.9	**http://l4u-00.jinr.ru/pub/misc/jobc/batch/SGE/8.1.9/sge-8.1.9.tar.gz**

OS Requirements

Table 5-8 lists the OS requirements.

Table 5-8 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

 Copyright © Huawei Technologies Co., Ltd.

5.2.3 Planning the Paths for Software Porting

Table 5-9 lists the software installation paths involved in the SGE 8.1.9 porting process and the hosts in the SGE cluster.

Table 5-9 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	*/path/to/SGE*	Directory in which the SGE source code is compiled	The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.
2	*/path/to/SGE_ROOT*	Installation path of SGE	

Table 5-10 SGE host description

No.	Host	Description
1	Master host	The master host is central for the overall cluster activities. It runs the sge_qmaster daemon and the sge_schedd daemon. These daemons control grid engine system components, such as queues and jobs. The daemons maintain the tables related to component status, user access permissions, and other tasks. By default, the master host also manages the administration hosts and submission hosts. In this document, the node name of the master host is **armnode2**.
2	Administration host	The administration host runs any type of management operations in the grid engine system.
3	Submission host	Users can submit and control batch processing jobs on the submission hosts. Specifically, after logging in to a submission host, a user can use the **qsub** command to submit a job, and use the **qstat** command to monitor the job status.
4	Execution host	The execution hosts run jobs. Therefore, they host queue instances. The execution hosts run the sge_execd daemon. In this document, the node name of the execution host is **armnode4**.

5.2.4 Configuring the Compilation Environment

Step 1 Use PuTTY to log in to the server as the **root** user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run the following command to install dependencies:

```
yum install -y java-1.8.0-openjdk* ant hwloc-devel openssl-devel libdb-devel pam-devel junit* javacc* libXt-devel motif-devel ncurses-devel
```

NOTE

In this example, the CentOS 7.6 image is used as the local Yum source. You can change the Yum source in the **/etc/yum.repos.d** directory as required.

Step 3 Load the environment variables of Java 1.8.0 that has been installed.

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-7.b13.el7.aarch64
export PATH=$JAVA_HOME/bin:$PATH
```

Step 4 Create a soft link for the JRE library of ARM64.

```
cd /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-7.b13.el7.aarch64/jre/lib
ln -s aarch64 arm64
```

----End

5.2.5 Obtaining the Source Code

Step 1 Download the SGE source code package **sge-8.1.9.tar.gz**.

http://l4u-00.jinr.ru/pub/misc/jobc/batch/SGE/8.1.9/sge-8.1.9.tar.gz

Step 2 Use an SFTP tool to upload the SGE source code package to the */path/to/SGE* directory on the server.

----End

5.2.6 Compiling and Installing SGE

5.2.6.1 Compiling SGE

Step 1 Use PuTTY to log in to the SGE master host as the **root** user.

Step 2 Run the following commands on PuTTY to decompress the SGE installation package:

```
cd /path/to/SGE
tar -xvf sge-8.1.9.tar.gz
cd sge-8.1.9/source
```

Step 3 Run the **bootstrap.sh** script.

```
sh scripts/bootstrap.sh
```

Step 4 Modify the **aimk** file.

1. Open the file.
   ```
   vi aimk
   ```
2. Press **i** to enter the insert mode, and add "-L/usr/lib64 -lcurses" to the end of **-lpthread** in lines 330, 1852, and 2664 in the **aimk** file.
   ```
   330: set LIBS = "-lm -lpthread -L/usr/lib64 -lcurses"
   1852:set LIBS - "-lsocket -lnsl -lm -lpthread -L/usr/lib64 -lcurses -lthread -lkstat"
   2664:set SGE_LIBS = "-lsge -lpthread -L/usr/lib64 -lcurses"
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Set the environment variable **SGE_ROOT**.

```
export SGE_ROOT=/path/to/SGE_ROOT
```

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Run the following commands to compile the software:

```
./aimk -parallel 32
echo y | ./scripts/distinst -all -local -noexit
```

----End

5.2.6.2 Installing SGE on the Master Host

Step 1 Run the following command to install the NIS package:

```
yum install ypserv -y
```

Step 2 Configure the NIS domain.

```
nisdomainname huawei
```

NOTE

In the command, *huawei* indicates the domain name of the cluster. Replace it with the actual domain name.

Step 3 Modify the **/etc/sysconfig/network** configuration file.

1. Open the file.
   ```
   vi /etc/sysconfig/network
   ```
2. Press **i** to enter the insert mode and enter the following information:
   ```
   HOSTNAME=Name of the master host
   NETWORKING=yes
   NISDOMAIN=huawei
   ```

   NOTE

   HOSTNAME indicates the name of the master node.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Add a user, for example, user **test**.

```
useradd -d /path/test test
```

NOTE

In this example, **/path** is a shared directory. All nodes in the cluster are mounted to the **/path** directory. Set the path based on actual situation.

Step 5 Change the password of the **test** user, for example, to **Huawei12#$**.

```
echo "Huawei12#$" | passwd --stdin test
```

Step 6 Restart the ypserv service to update the database.

```
systemctl start ypserv
systemctl enable ypserv
cd /var/yp
make
```

Step 7 Run the following commands to start the installation:

```
cd /path/to/SGE_ROOT
./install_qmaster
```

Step 8 On the **Welcome to the Grid Engine installation** screen, press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

```
Welcome to the Grid Engine installation
---------------------------------------

Grid Engine qmaster host installation
-------------------------------------

Before you continue with the installation please read these hints:

   - Your terminal window should have a size of at least
     80x24 characters

   - The INTR character is often bound to the key Ctrl-C.
     The term >Ctrl-C< is used during the installation if you
     have the possibility to abort the installation

The qmaster installation procedure will take approximately 5-10 minutes.

Hit <RETURN> to continue >>
```

Step 9 On the **Choosing Grid Engine admin user account** screen, enter **n** and press **Enter** twice.

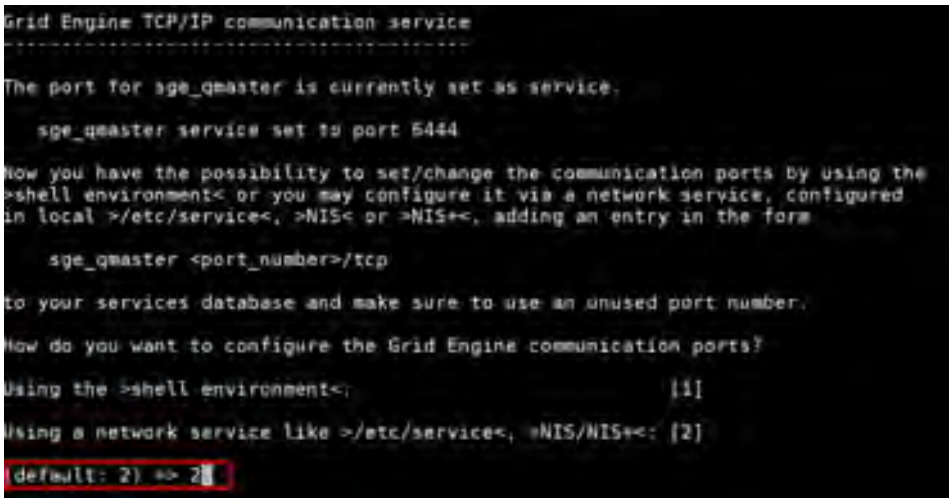
```
Choosing Grid Engine admin user account
---------------------------------------

You may install Grid Engine so that all files are created with the user id of an
unprivileged user.

This will make it possible to install and run Grid Engine in directories
where user >root< has no permissions to create and write files and directories.

   - Grid Engine still has to be started by user >root<

   - this directory should be owned by the Grid Engine administrator

Do you want to install Grid Engine
under a user ID other than >root< (y/n) [y] >> n


Installing Grid Engine as user >root<

Hit <RETURN> to continue >>
```

Step 10 On the **Checking $SGE_ROOT directory** screen, use the path planned and press **Enter** twice.

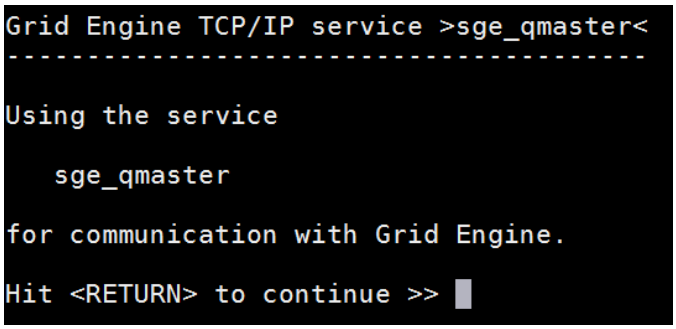
```
Checking $SGE_ROOT directory
----------------------------

The Grid Engine root directory is:

   $SGE_ROOT = /path/to/SGE_ROOT

If this directory is not correct (e.g. it may contain an automounter
prefix) enter the correct path to this directory or hit <RETURN>
to use default [/path/to/SGE_ROOT] >>

Your $SGE_ROOT directory: /path/to/SGE_ROOT

Hit <RETURN> to continue >>
```

Step 11 On the **Grid Engine TCP/IP communication service** screen, enter **2** and press **Enter**.

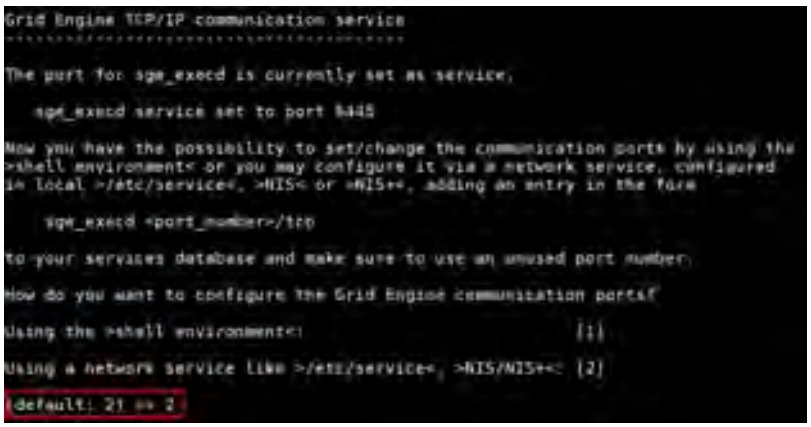
 Copyright © Huawei Technologies Co., Ltd.



Step 12 On the **Grid Engine TCP/IP service** screen, press **Enter**.



Step 13 On the **Grid Engine TCP/IP communication service** screen, enter **2** and press **Enter**.



Step 14 On the **Grid Engine TCP/IP communication service** screen, press **Enter**.

 Copyright © Huawei Technologies Co., Ltd.

```
Grid Engine TCP/IP communication service
----------------------------------------

Using the service

   sge_execd

for communication with Grid Engine.

Hit <RETURN> to continue >>
```

Step 15 On the **Grid Engine cells** screen, use the default cell name and press **Enter** twice.

Step 16 On the **Unique Cluster name** screen, enter the cluster name and press **Enter** twice.

```
Unique cluster name
-------------------

The cluster name uniquely identifies a specific Sun Grid Engine cluster.
The cluster name must be unique throughout your organization. The name
is not related to the SGE cell.

The cluster name must start with a letter ([A-Za-z]), followed by letters,
digits ([0-9]), dashes (-) or underscores (_).

Enter new cluster name or hit <RETURN>
to use default [p6444] >> hpccluster
creating directory: /path/to/SGE_ROOT/default/common

Your $SGE_CLUSTER_NAME: hpccluster

Hit <RETURN> to continue >>
```

Step 17 On the **Grid Engine qmaster spool directory** screen, specify the spool directory (the default directory is recommended) and press **Enter** twice.

 Copyright © Huawei Technologies Co., Ltd.

```
Grid Engine qmaster spool directory
-----------------------------------

The qmaster spool directory is the place where the qmaster daemon stores
the configuration and the state of the queueing system.

User >root< on this host must have read/write access to the qmaster
spool directory.

If you will install shadow master hosts or if you want to be able to start
the qmaster daemon on other hosts (see the corresponding section in the
Grid Engine Installation and Administration Manual for details) the account
on the shadow master hosts also needs read/write access to this directory.

Enter a qmaster spool directory [/path/to/SGE_ROOT/default/spool/qmaster] >>

Using qmaster spool directory >/path/to/SGE_ROOT/default/spool/qmaster<.
Hit <RETURN> to continue >>
```

Step 18 On the **Windows Execution Host Support** screen, enter **n** and press **Enter**.

```
Windows Execution Host Support
------------------------------

Are you going to install Windows Execution Hosts? (y/n) [n] >> n
```

Step 19 On the **Verifying and setting file permissions** screen, enter **y** and press **Enter** twice.

```
Verifying and setting file permissions
--------------------------------------

Did you either install this version with a package manager,
or otherwise verify and set the file permissions of your
distribution (enter: y) (y/n) [y] >> y

We do not verify file permissions. Hit <RETURN> to continue >>
```

Step 20 On the **Select default Grid Engine hostname resolving method** screen, enter **y** and press **Enter** twice.

```
Select default Grid Engine hostname resolving method
----------------------------------------------------

Are all hosts of your cluster in one DNS domain? If this is
the case the hostnames

   >hostA< and >hostA.foo.com<

would be treated as equal, because the DNS domain name >foo.com<
is ignored when comparing hostnames.

Are all hosts of your cluster in a single DNS domain (y/n) [y] >> y

Ignoring domain name when comparing hostnames.

Hit <RETURN> to continue >>
```

Step 21 On the **Grid Engine JMX MBean server** screen, enter **n** and press **Enter** twice.

 Copyright © Huawei Technologies Co., Ltd.

```
Grid Engine JMX MBean server
----------------------------

In order to use the SGE Inspect or the Service Domain Manager (SDM)
SGE adapter you need to configure a JMX server in qmaster. Qmaster
will then load a Java Virtual Machine through a shared library.
NOTE: Java  or later is required for the JMX MBean server.

Do you want to enable the JMX MBean server (y/n) [n] >> n


Making directories
------------------

creating directory: /path/to/SGE_ROOT/default/spool/qmaster
creating directory: /path/to/SGE_ROOT/default/spool/qmaster/job_scripts
Hit <RETURN> to continue >>
```

Step 22 On the **Setup spooling** screen, enter **classic** and press **Enter**.

```
Setup spooling
--------------

Your SGE binaries are compiled to link the spooling libraries
during runtime (dynamically). So you can choose between Berkeley DB
spooling and Classic spooling methods.
Please choose a spooling method (berkeleydb|classic) [classic] >> classic
Dumping bootstrapping information
Initializing spooling database
```

Step 23 On the **Grid Engine group id range** screen, use the default settings and press **Enter** twice.

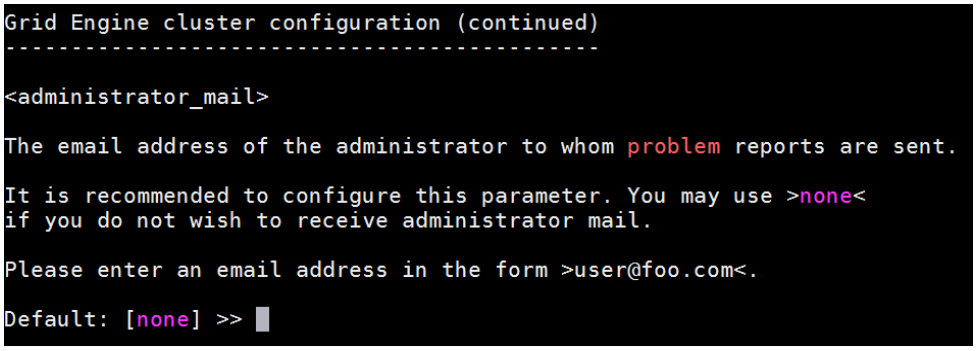
```
Grid Engine group id range
--------------------------

When jobs are started under the control of Grid Engine an additional group id
is set on platforms which do not support jobs. This is done to provide maximum
control for Grid Engine jobs.

This additional UNIX group id range must be unused group ids in your system.
Each job will be assigned a unique id during the time it is running.
Therefore you need to provide a range of ids which will be assigned
dynamically for jobs.

The range must be big enough to provide enough numbers for the maximum number
of Grid Engine jobs running at a single moment on a single host. E.g. a range
like >20000-20100< means that Grid Engine will use the group ids from
20000-20100 and provides a range for 100 Grid Engine jobs at the same time
on a single host.

You can change at any time the group id range in your cluster configuration.

Please enter a range [20000-20100] >>

Using >20000-20100< as gid range. Hit <RETURN> to continue >>
```

Step 24 On the **Grid Engine cluster configuration** screen, use the default settings and press **Enter**.

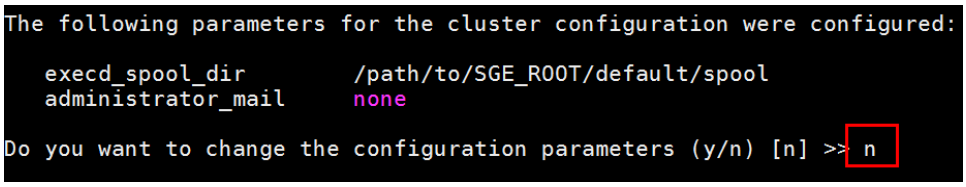
```
Grid Engine cluster configuration
---------------------------------

Please give the basic configuration parameters of your Grid Engine
installation:

   <execd_spool_dir>

The pathname of the spool directory of the execution hosts. User >root<
must have the right to create this directory and to write into it.

Default: [/path/to/SGE_ROOT/default/spool] >>
```

 Copyright © Huawei Technologies Co., Ltd.

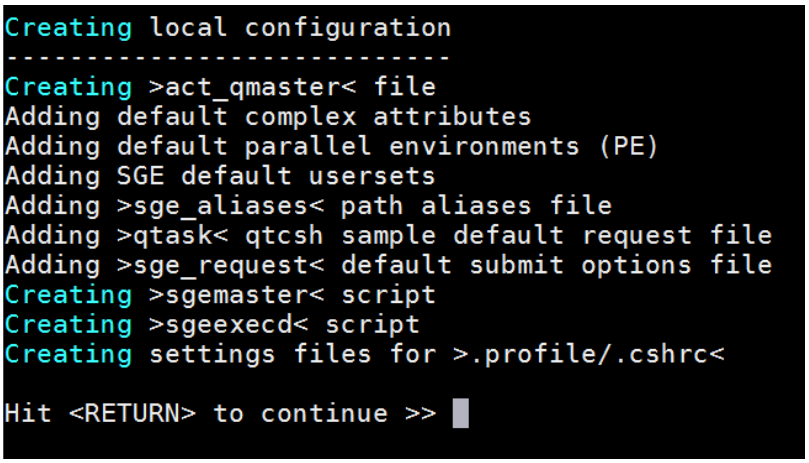
Step 25 On the **Grid Engine cluster configuration (continued)** screen, use the default settings and press **Enter**.



Step 26 When "The following parameters for the cluster configuration were configured" is displayed, enter **n** and press **Enter**.

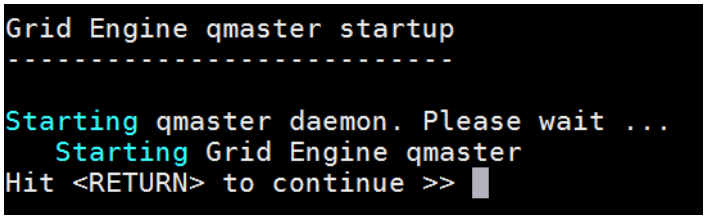


Step 27 On the **Creating local configuration** screen, press **Enter**.



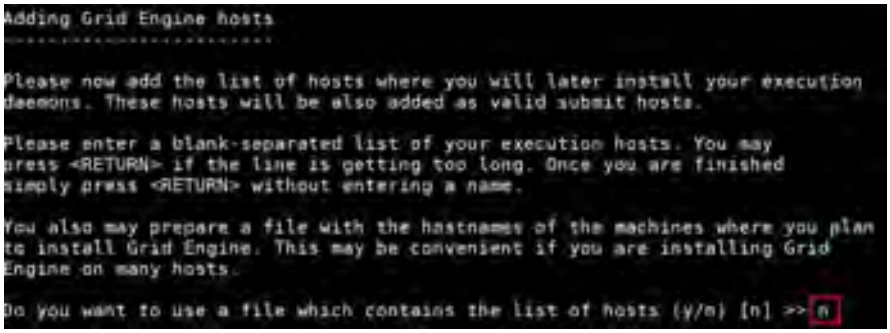
Step 28 On the **qmaster startup script** screen, enter **y** and press **Enter** twice.

Step 29 On the **Grid Engine qmaster startup** screen, press **Enter**.

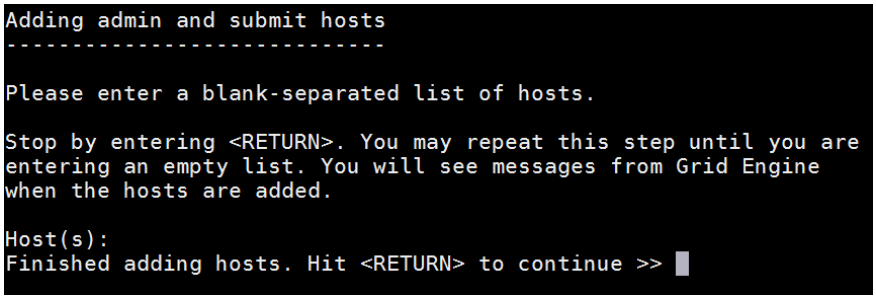


 Copyright © Huawei Technologies Co., Ltd.

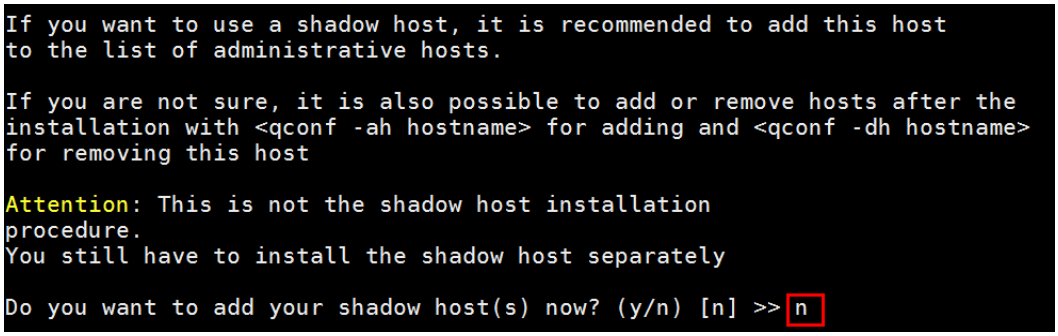
Step 30 On the **Adding Grid Engine hosts** screen, enter **n** and press **Enter**.



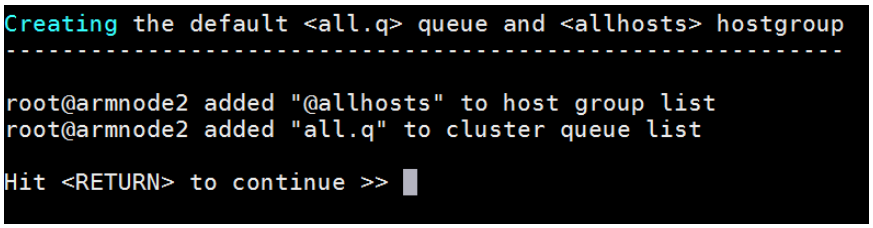
Step 31 On the **Adding admin and submit hosts** screen, press **Enter** twice.



Step 32 On the **Shadow host** screen, enter **n** and press **Enter**.



Step 33 On the **Creating the default queue and hostgroup** screen, press **Enter**.

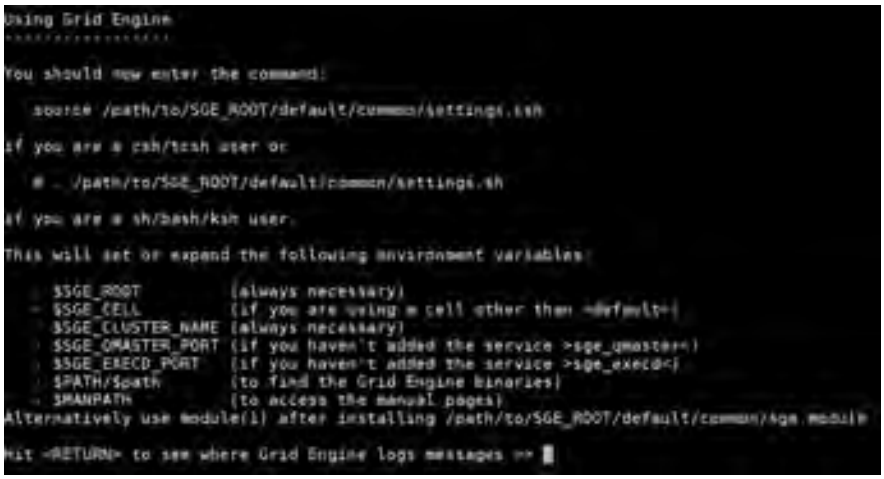


Step 34 On the **Scheduler Tuning** screen, enter **1** and press **Enter**. Then, enter **y** and press **Enter**.

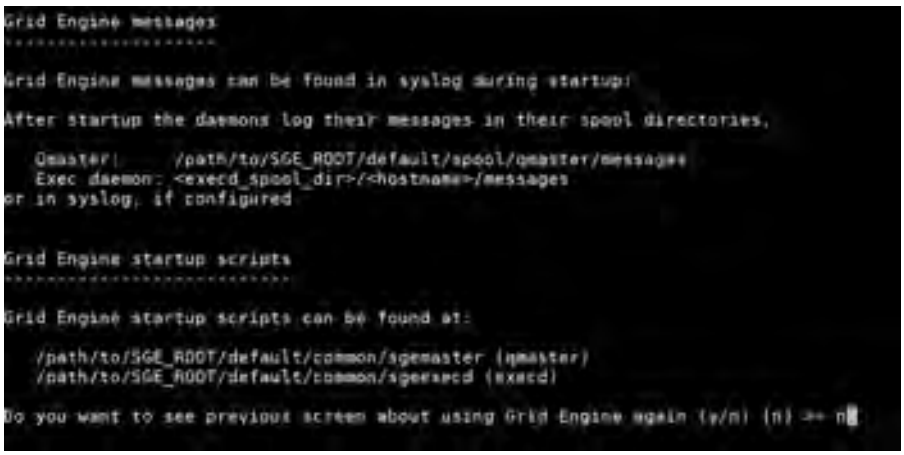
 Copyright © Huawei Technologies Co., Ltd.

```
Scheduler Tuning
----------------

The details on the different options are described in the manual.

Configurations
--------------
1) Normal
          Fixed interval scheduling, report limited scheduling information,
          actual + assumed load

2) High
          Fixed interval scheduling, report limited scheduling information,
          actual load

3) Max
          Immediate Scheduling, report no scheduling information,
          actual load

Enter the number of your preferred configuration and hit <RETURN>!
Default configuration is [1] >> 1
```

Step 35 On the **Using Grid Engine** screen, press **Enter**.

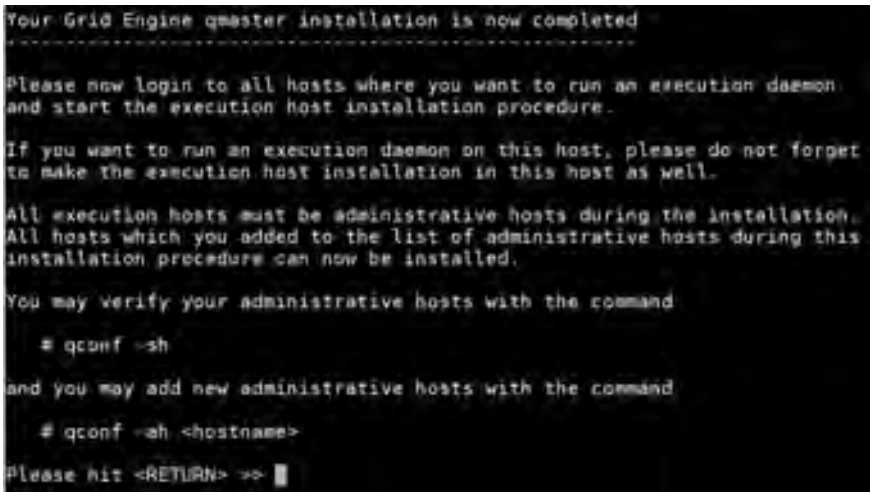


Step 36 On the **Grid Engine messages** screen, enter **n** and press **Enter**.



Step 37 When "Your Grid Engine qmaster installation is now completed" is displayed, press **Enter** to complete the installation.

 Copyright © Huawei Technologies Co., Ltd.



Step 38 Load the SGE environment variables.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

Step 39 Add the SGE environment variables to the **.bashrc** file for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /root/.bashrc
```

Step 40 Add the master host as the submission host.

```
qconf -as armnode2
```

Step 41 Add the master host as the administration host.

```
qconf -ah armnode4
```

----End

5.2.6.3 Installing SGE on the Execution Host

Step 1 On the execution host, run the following command to install the NIS client installation package ypbind:

yum install ypbind -y

NOTE

In this example, the CentOS 7.6 image is used as the local Yum source. You can change the Yum source in the **/etc/yum.repos.d** directory as required.

Step 2 Modify the **/etc/sysconfig/network** file on the execution host.

1. Open the file.
   ```
   vi /etc/sysconfig/network
   ```
2. Press **i** to enter the insert mode and enter the following information:
   ```
   HOSTNAME=Name of the execution host
   NETWORKING=yes
   NISDOMAIN=huawei
   ```

   NOTE

   HOSTNAME specifies the name of the execution host, and **NISDOMAIN** specifies the domain name of the cluster. Set them based on actual situation.
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Modify the **/etc/yp.conf** file on the execution host.

1. Open the file.
   ```
   vi /etc/yp.conf
   ```
2. Press **i** to enter the insert mode and add the following information to the end of the file:
   ```
   domain huawei server 192.168.40.50
   ```

   NOTE

   In this example, *192.168.40.50* indicates the IP address of the master host. Replace it with the actual IP address.
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Modify the **/etc/nsswitch.conf** file.

1. Open the file.
   ```
   vi /etc/nsswitch.conf
   ```
2. Press **i** to enter the insert mode and enter the following information:
   ```
   passwd:    files nis
   shadow:    files nis
   group:     files nis
   hosts:     files nis dns
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Start the ypbind service.

```
systemctl start ypbind
systemctl enable ypbind
```

Step 6 Install hwloc.

```
yum install hwloc* -y
```

NOTE

In this example, the CentOS 7.6 image is used as the local Yum source. You can change the Yum source in the **/etc/yum.repos.d** directory as required.

Step 7 Set the environment variable **SGE_ROOT**.

```
export SGE_ROOT=/path/to/SGE_ROOT
```

Step 8 Load the SGE environment variables.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

Step 9 Add the SGE environment variables to the **.bashrc** file for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /root/.bashrc
```

Step 10 Start the execd process on the execution host.

```
/path/to/SGE_ROOT/default/common/sgeexecd
```

Step 11 Enable the execd process to automatically start as the system boots.

```
echo "/path/to/SGE_ROOT/default/common/sgeexecd" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.2.7 Running and Verifying SGE

Procedure

Step 1 Use PuTTY to log in to the SGE master host as the **root** user.

Step 2 Run the following command to add the execution host:

```
qconf -as armnode4
```

Step 3 Query the default host name.

```
qconf -shgrpl
@allhosts
```

Step 4 Modify the host group information and add the execution host to the host group.

1. **qconf -mhgrp @allhosts**
2. Press **i** to enter the insert mode and enter the following information:
   ```
   group_name @allhosts
   hostlist armnode4
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Switch to a non-**test** user.

```
su - test
```

Step 6 Load the SGE environment variables as the **test** user.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

Step 7 Add the SGE environment variables to the **.bashrc** file of the **test** user for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /path/test/.bashrc
```

Step 8 Create an execution script **run.sh**.

Run the **vi run.sh** command.

Press **i** to enter the insert mode and add the following information:

```
#!/bin/bash
#$ -S /bin/bash
nodeinfo=`hostname`
echo "This is the SGE test from $nodeinfo" >> sge-test.log
```

Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 9 Submit a job.

```
qsub -V -cwd -o stdout.txt -e stderr.txt run.sh
```

Table 5-11 describes the command parameters.

Table 5-11 Common parameters in the **qsub** command

Parameter	Description
-V	Exports the environment variables in the current shell to the job to be submitted.
-cwd	Runs the program in the current working directory. By default, it is the home directory on the compute node of the current user.

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Description
-o	Adds the standard output to the end of the specified file. The default file name is **$job_name.o$job_id**.
-e	Adds the standard error output to the end of the specified file. The default file name is **$job_name.e$job_id**.
-q	Specifies the queue to be delivered. If this parameter is not specified, the system searches for the queue with the permission and minimum load to execute the job.
-S	Specifies the software that runs the commands in **run.sh**. The default value is **tcsh**. You are advised to use bash. Set this parameter to **/bin/bash**, or add "#$ -S /bin/bash" at the beginning of the **run.sh** file. If the parameter is not set to bash, the output contains "Warning: no access to tty (Bad file descriptor)."
-hold_jid	Specifies the jobs to be executed before the current job. A comma (,) is used to separate multiple job_ids.
-N	Sets the job name. The default job name is the input file name of qsub.
-p	Sets the job priority. The value ranges from -1023 to 1024. A larger value indicates a higher priority. However, a higher permission is required to set this parameter to a positive number. Common users cannot set this parameter to a positive number.
-j y|n	Specifies whether to merge the standard output and standard error output streams into the **-o** parameter result.
-pe	Sets parallel environment (PE) information.

Step 10 Manage jobs. **Table 5-12** describes the commands and parameter for job management.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-12 Commands and parameter for job management

Command	Description
qstat -f	Queries all jobs submitted by the current user on the current node. The job status can be: • **qw**: waiting state. A job is in **qw** state after being submitted. The job runs immediately once computing resources are available. • **hqw**: pending state. A job in this state starts to run only after the previous jobs are executed. The job submitted by using **qsub** with **-hold_jid** specified is in this state. • **Eqw**: job waiting in error state. • **r**: running state. • **s**: job suspended temporarily because resources are used by the job with a higher priority. • **dr**: job deleted after a node exits unexpectedly. The job disappears only after the node is restarted.
qstat jobID	Queries information by job ID.
qstat -u user	Queries information by user.
qdel jobID	Deletes a job.

----End

5.2.8 SGE Common Commands

Table 5-13 SGE common commands

Command	Description
qconf -ae hostname	Adds an execution host.
qconf -de hostname	Deletes an execution host.
qconf -sel	Displays the list of execution hosts.
qconf -ah hostname	Adds an administration host.
qconf -dh hostname	Deletes an administration host.
qconf -sh	Displays the list of administration hosts.
qconf -as hostname	Adds a submission host.
qconf -ds hostname	Deletes a submission host.
qconf -ss	Displays the list of submission hosts.
qconf -ahgrp groupname	Adds a host user group.
qconf -mhgrp groupname	Modifies a host user group.

 Copyright © Huawei Technologies Co., Ltd.

Command	Description
qconf -shgrp groupname	Display information about a host user group.
qconf -shgrpl	Display the list of host user groups.
qconf -aq queuename	Adds a cluster queue.
qconf -dq queuename	Deletes a cluster queue.
qconf -mq queuename	Modifies cluster queue configuration.
qconf -sq queuename	Displays cluster queue configuration.
qconf -sql	Displays the list of cluster queues.
qconf -ap PE_name	Adds a parallel environment (PE).
qconf -mp PE_name	Modifies a PE.
qconf -dp PE_name	Deletes a PE.
qconf -sp PE_name	Displays information about a PE.
qconf -spl	Displays the list of PE names.
qstat -f	Displays the execution host status.
qstat -u user	Queries user Jobs.
qhost	Displays information about the execution host resources.

5.3 Conda 4.3.16 Porting Guide (CentOS 7.6)

5.3.1 Introduction

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Conda, as a package manager, helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because Conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.

For more information, visit the **official Conda website**.

Programming language: Python

Brief description: Open source package and environment management system.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: 3-Clause BSD

Recommended Software Version

Conda 4.3.16

5.3.2 Environment Requirements

Hardware Requirements

Table 5-14 lists the hardware requirements.

Table 5-14 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 5-15 lists the software requirements.

Table 5-15 Software requirements

Item	Version	How to Obtain
Python	2.7.5	Use yum commands to install it.

OS Requirements

Table 5-16 lists the OS requirements.

Table 5-16 OS requirements

Item	Version	How to Obtain
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	Included in the OS image.

5.3.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Conda software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-17 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each installation package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

5.3.4 Configuring the Compilation Environment

5.3.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Single-Node Scenario" in **Basic Environment Setup Guide**.
2	Installing dependencies	See **5.3.4.2 Installing Dependencies**.

5.3.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install the dependency:

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel python-devel
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.3.5 Obtaining the Source Code

Procedure

Step 1 The Conda software package is downloaded and installed through the link specified by the pip command. You do not need to download the source code package separately.

URL: **https://pypi.tuna.tsinghua.edu.cn/simple**

----End

5.3.6 Compiling and Installing Conda

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to install Conda.

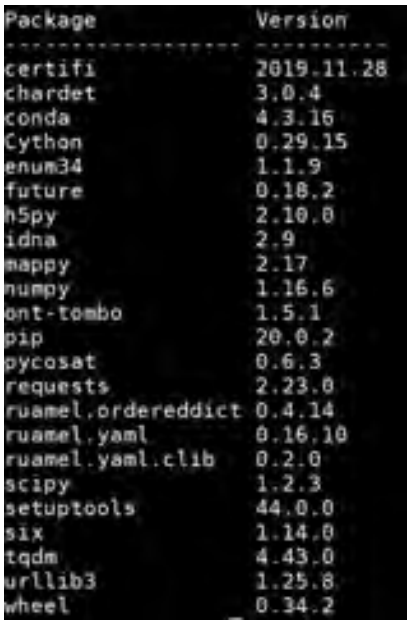
```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple conda
```

Step 3 Run the following command to check whether Conda is installed successfully:

```
pip list
```

If **conda** is displayed in the command output, the installation is successful, as shown in **Figure 5-3**.

Figure 5-3 Result example



----End

5.4 NPB 3.4 Porting Guide (CentOS 7.6)

5.4.1 Introduction

NAS Parallel Benchmarks (NPB) are a set of benchmarks targeting performance evaluation of highly parallel supercomputers. They are developed and maintained

 Copyright © Huawei Technologies Co., Ltd.

by the NASA Advanced Supercomputing (NAS) Division (formerly the NASA Numerical Aerodynamic Simulation Program) based at the NASA Ames Research Center. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original "pencil-and-paper" specification (NPB 1). The benchmark suite has been extended to include new benchmarks for unstructured adaptive meshes, parallel I/O, multi-zone applications, and computational grids. Problem sizes in NPB are predefined and indicated as different classes. Reference implementations of NPB are available in commonly-used programming models like MPI and OpenMP (NPB 2 and NPB 3).

For more information, visit the **official NPB website**.

Programming language: C/Fortran

Brief description: NPB is a set of benchmarks targeting performance evaluation of highly parallel supercomputers.

Open source license: NOSA

Recommended Software Version

NPB 3.4

5.4.2 Environment Requirements

Hardware Requirements

Table 5-18 lists the hardware requirements.

Table 5-18 Hardware requirements

Item	Description
CPU	Kunpeng 920

Software Requirements

Table 5-19 lists the software requirements.

Table 5-19 Software requirements

Item	Version	Download Address
NPB	3.4	**https://www.nas.nasa.gov/publications/npb.html**

OS Requirements

Table 5-20 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-20 OS requirements

Item	Version	How to Obtain
CentOS	7.6	**https://www.centos.org/download/**
Kernel	4.14.0-115	Included in the OS image.

5.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NPB software porting.

Table 5-21 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package required for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	*/path/to/NPB*	Installation path of NPB	The installation paths listed in this table are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.4.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-22 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

5.4.5 Obtaining the Source Code

Procedure

Step 1 Download the NPB installation package **NPB3.4.tar.gz**.

Download address: **https://www.nas.nasa.gov/publications/npb.html**.

Step 2 Use SFTP to upload the NPB installation package to the ***/path/to/NPB*** directory on the server.

----End

5.4.6 Compiling and Installing NPB

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/NPB
```

Step 3 Copy the installation package to the installation directory.

```
cp NPB3.4.tar.gz /path/to/NPB
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/NPB
```

Step 5 Run the following command to decompress the installation package:

```
tar -xvf NPB3.4.tar.gz
```

Step 6 Run the following command to switch to the directory generated after the package is decompressed

```
cd NPB3.4
```

Step 7 Run the following command to generate a configuration file:

```
cd NPB3.4-MPI
cp config/make.def.template config/make.def
```

Step 8 Run the following command to load **OPENMPI** and **GNU** environmental variables:

```
export PATH=/path/to/OPENMPI/bin:/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:/path/to/GNU/lib64:$LD_LIBRARY_PATH
```

Step 9 Run the following command to perform compilation.

```
make EP CLASS=D NPROCS=96
```

NOTE

In the preceding command:

- **EP** indicates the benchmark. The NPB has 12 benchmarks: IS, EP, CG, MG, FT, BT, BT-10, SP, LU, UA, DC, and DT.
- **CLASS=D** indicates the problem size. NPB has eight problem sizes: S, W, A, B, C, D, E, and F.
- **NPROCS=96** indicates the number of cores.

You can select the benchmark, problem size, and number of cores based on the site requirements. For details about the parameters, see **5.4.8 More Resources**.

----End

 Copyright © Huawei Technologies Co., Ltd.

5.4.7 Running and Verifying NPB

Procedure

Step 1 Use PuTTY to log in to the server as the **root** user.

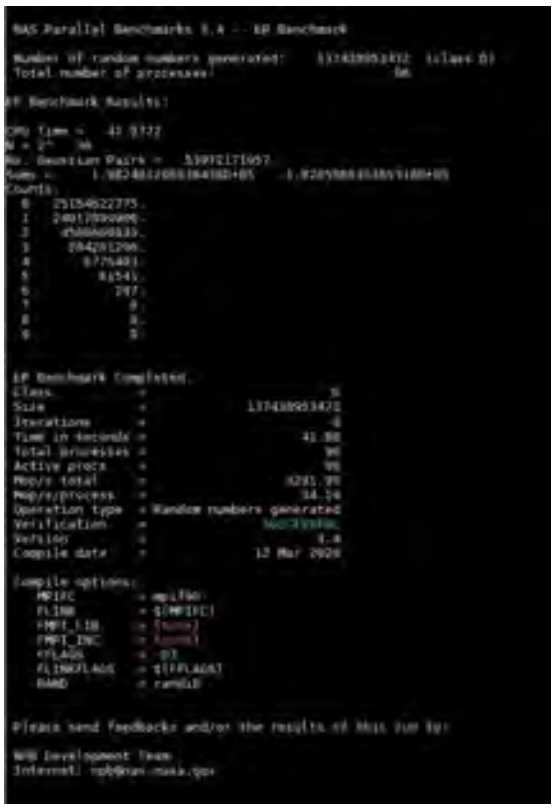
Step 2 Switch to the test directory.

```
cd /path/to/NPB/NPB3.4/NPB3.4-MPI/bin
```

Step 3 Run the following command to start the test. If the following information is displayed, the test is successful.

```
mpirun  --allow-run-as-root --mca btl ^openib -np 96 ep.D.x
```

Figure 5-4 Result example



----End

5.4.8 More Resources

- 12 NPB benchmarks:

 Copyright © Huawei Technologies Co., Ltd.

Table 5-23 NPB benchmarks

Benchmark	Full Name	Description
IS	Integer Sort	It is used to solve two-dimensional large integer sorting based on the bucket sort, and requires full-switching communication.
EP	Embarrassingly Parallel	It is used to calculate the Gauss pseudo random number and mainly performs floating-point number calculation. The prominent feature of EP is that no communication between processors is performed. Therefore, different results obtained through interconnection show the same execution characteristics.
CG	Conjugate Gradient	It is used to solve the approximation of the minimum eigenvalue of a large sparse symmetric positive definite matrix.
MG	MultiGrid	It is used to monitor short- and long-distance communication. MG is a simplified multi-grid core.
FT	Fast Fourier Transform	It is used to detect long-distance communication and solve 3-dimensional partial differential equation using fast Fourier transform.
BT	Block Tri-Diagonal	It is used to solve the tri-diagonal equation set.

 Copyright © Huawei Technologies Co., Ltd.

Benchmark	Full Name	Description
BT-IO	Block Tri-Diagonal-IO	It is used to test different parallel I/O technologies.
SP	Scalar Penta-diagonal	It is used to solve the penta-diagonal equation set.
LU	Lower Upper Triangular	It is used to solve sparse block equations based on symmetric super-relaxation method.
UA	Unstructured Adaptive	It is used to test unstructured adaptive networks, dynamic and irregular memory access.
DC	Data Cube	It is used to test data movement.
DT	Data Traffic	

- Eight NPB problem sizes:

  Table 5-24 NPB problem sizes

Problem Size	Description
S	The S (sample) class is for sample programs.
W	The W (workstation) class is usually used for workstations.
A-F	The size of A class is the smallest and increases in ascending order.

5.5 Double-conversion 3.2.0 Porting Guide (Kylin V10)

5.5.1 Introduction

The Double-conversion package contains a library that facilitates binary-to-decimal and decimal-to-binary routines for IEEE doubles.

For more information, visit the **official Double-conversion website**.

Programming language: C++

 Copyright © Huawei Technologies Co., Ltd.

Brief description: A library that facilitates binary-to-decimal and decimal-to-binary routines for IEEE doubles.

Open source license: custom open source license

Recommended Software Version

Double-conversion 3.2.0

5.5.2 Environment Requirements

Hardware Requirements

Table 5-25 lists the hardware requirements.

Table 5-25 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-26 lists the software requirements.

Table 5-26 Software requirements

Item	Version	Download URL
Double-conversion	3.2.0	https://github.com/google/double-conversion/archive/refs/tags/v3.2.0.tar.gz

OS Requirements

Table 5-27 lists the OS requirements.

Table 5-27 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

5.5.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Double-conversion software porting.

Table 5-28 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/double-conversion	Installation path of Double-conversion	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.5.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-29 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

5.5.5 Obtaining the Source Code

Procedure

Step 1 Download the Double-conversion installation package.

https://github.com/google/double-conversion/archive/refs/tags/v3.2.0.tar.gz

Step 2 Use an SFTP tool to upload the Double-conversion installation package to the */path/to/double-conversion* directory on the server.

----End

5.5.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/double-conversion
```

Step 3 Decompress the installation packages.

```
tar xvf v3.2.0.tar.gz
```

Step 4 Go to the directory generated after decompression.

```
cd double-conversion-3.2.0
```

Step 5 Create a **build** directory and go to the directory.

```
mkdir build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/double-conversion -DBUILD_SHARED_LIBS=ON -DBUILD_TESTING=ON
make -j
make install
```

----End

5.5.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify Double-conversion.

```
cd /path/to/double-conversion-3.2.0/build/test/cctest
./cctest
```

Information similar to the following is displayed:

```
Ran 84 tests.
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.6 LIBPNG 1.6.37 Porting Guide (Kylin V10)

5.6.1 Introduction

The LIBPNG package contains a library of functions for creating and manipulating Portable Network Graphics (PNG) image format files.

For more information, visit the **official LIBPNG website**.

Programming language: C

Brief description: A library of functions for creating and manipulating Portable Network Graphics (PNG) image format files.

Open source license: custom open source license

Recommended Software Version

LIBPNG 1.6.37

5.6.2 Environmental Requirements

Hardware Requirements

Table 5-30 lists the hardware requirements.

Table 5-30 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-31 lists the software requirements.

Table 5-31 Software requirements

Item	Version	Download URL
LIBPNG	1.6.37	**https://nchc.dl.sourceforge.net/project/libpng/libpng16/1.6.37/libpng-1.6.37.tar.gz**

OS Requirements

Table 5-32 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-32 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.6.3 Paths for Software Porting

This chapter lists the software installation paths involved in the LIBPNG software porting.

Table 5-33 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/libpng	Installation path of LIBPNG	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.6.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-34 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.

5.6.5 Obtaining the Source Code

Procedure

Step 1 Download the LIBPNG installation package.

https://nchc.dl.sourceforge.net/project/libpng/libpng16/1.6.37/libpng-1.6.37.tar.gz

Step 2 Use an SFTP tool to upload the LIBPNG installation package to the */path/to/libpng* directory on the server.

----End

5.6.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/libpng
```

Step 3 Decompress the installation package.

```
tar xvf libpng-1.6.37.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd libpng-1.6.37
```

Step 5 Perform the compilation and installation.

```
./configure --prefix=/path/to/libpng --build=aarch64-unknown-linux-gnu
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.6.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify LIBPNG.

```
cd /path/to/libpng/bin
./libpng16-config
```

Information similar to that in the following figure is displayed:

```
[root@localhost bin]# ./libpng16-config
Usage: ./libpng16-config [OPTION] ...

Known values for OPTION are:

  --prefix        print libpng prefix
  --libdir        print path to directory containing library
  --libs          print library linking information
  --ccopts        print compiler options
  --cppflags      print pre-processor flags
  --cflags        print preprocessor flags, I_opts, and compiler options
  --I_opts        print "-I" include options
  --L_opts        print linker "-L" flags for dynamic linking
  --R_opts        print dynamic linker "-R" or "-rpath" flags
  --ldopts        print linker options
  --ldflags       print linker flags (ldopts, L_opts, R_opts, and libs)
  --static        revise subsequent outputs for static linking
  --help          print this help and exit
  --version       print version information
```

----End

5.7 Linenoise-NG 1.0.1 Porting Guide (Kylin V10)

5.7.1 Introduction

Linenoise Next Generation (Linenoise-NG) is a small, portable GNU readline replacement for Linux, Windows and MacOS which is capable of handling UTF-8 characters. Unlike GNU readline, which is GPL, this library uses a BSD license and can be used in any kind of program.

For more information, visit the **official Linenoise-NG website**.

Programming language: C++

Brief description: A small, portable GNU readline replacement.

Open source license: custom open source license

Recommended Software Version

Linenoise-NG 1.0.1

 Copyright © Huawei Technologies Co., Ltd.

5.7.2 Environment Requirements

Hardware Requirements

Table 5-35 lists the hardware requirements.

Table 5-35 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-36 lists the software requirements.

Table 5-36 Software requirements

Item	Version	Download URL
Linenoise-NG	1.0.1	https://github.com/qingqibing/linenoise-ng/archive/refs/tags/v1.0.1.tar.gz

OS Requirements

Table 5-37 lists the OS requirements.

Table 5-37 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.7.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Linenoise-NG software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-38 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/ linenoise-ng	Installation path of Linenoise-NG	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-39 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.

5.7.5 Obtaining the Source Code

Procedure

Step 1 Download the Linenoise-NG installation package.

https://github.com/qingqibing/linenoise-ng/archive/refs/tags/v1.0.1.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Use an SFTP tool to upload the Linenoise-NG installation package to the */path/to/linenoise-ng* directory on the server.

----End

5.7.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/linenoise-ng
```

Step 3 Decompress the installation package.

```
tar xvf v1.0.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd linenoise-ng-1.0.1
```

Step 5 Create a **build** directory and go to the directory.

```
mkdir build
cd build
```

Step 6 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/linenoise-ng -DCMAKE_BUILD_TYPE=Release
make -j
make install
```

----End

5.7.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify Linenoise-NG.

```
cd /path/to/linenoise-ng-1.0.1/build
./example
```

Type **hello,world!** and press **Enter**. Then type **this is linux** and press **Enter**. After that, press **Ctrl+C** to exit.

```
cat history
```

Information similar to the following is displayed:

```
# ./example
starting...
linenoise> hello,world!
thanks for the input.
linenoise> this is linux
thanks for the input.
linenoise> ^C
# cat history
hello,world!
this is linux
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.8 NCO 5.1.0 Porting Guide (Kylin V10)

5.8.1 Introduction

The NCO toolkit manipulates and analyzes data stored in netCDF-accessible formats, including DAP, HDF4, and HDF5. It exploits the geophysical expressivity and logic of many Climate & Forecast (CF) metadata conventions, the flexible description of physical dimensions translated by UDUnits, the network transparency of OPeNDAP, the storage capabilities (e.g., compression, chunking, groups) of the Hierarchical Data Format (HDF), and many powerful mathematical and statistical algorithms of the GNU Scientific Library (GSL). NCO is fast, powerful, and free.

For more information, visit the **official NCO website**.

Programming languages: C/C++

Brief description: Manipulates and analyzes data stored in netCDF-accessible formats.

Open source license: custom open source license

Recommended Software Version

NCO 5.1.0

5.8.2 Environmental Requirements

Hardware Requirements

Table 5-40 lists the hardware requirements.

Table 5-40 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-41 lists the software requirements.

Table 5-41 Software Requirements

Item	Version	Download URL
NCO	5.1.0	**https://github.com/nco/nco/archive/refs/tags/5.1.0.tar.gz**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
PnetCDF	1.11.2	http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
NetCDF-Fortran	4.4.5	https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5
NetCDF-C	4.7.0	https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0
GSL	2.6	http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz
UDUNITS2	2.2.28	https://artifacts.unidata.ucar.edu/repository/downloads-udunits/2.2.28/udunits-2.2.28.tar.gz

OS Requirements

Table 5-42 lists the OS requirements.

Table 5-42 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.8.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NCO software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-43 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.
2	/path/to/nco	Installation path of NCO	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
3	/path/to/pnetcdf	Installation path of PnetCDF	
4	/path/to/hdf5	Installation path of HDF5	
5	/path/to/netcdf	Installation path of NetCDF-C	
		Installation path of NetCDF-Fortran	
6	/path/to/gsl	Installation path of GSL	
7	/path/to/udunits	Installation path of UDUNITS2	

5.8.4 Configuring the Compilation Environment

5.8.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-44 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.
3	Installing Hyper MPI	See "Compiling the Hyper MPI Software Package Using GCC" in the "Installing Hyper MPI Using Source Code" section of the **Hyper MPI User Guide**
4	Installing the dependencies	See **5.8.4.2 Installing the Dependencies**.
5	Installing PnetCDF	See **5.8.4.3 Installing PnetCDF**.
6	Installing HDF5	See **5.8.4.4 Installing HDF5**.
7	Installing NetCDF	See **5.8.4.5 Installing NetCDF**.
8	Installing GSL	See **5.8.4.6 Installing GSL**.
9	Installing UDUNITS	See **5.8.4.7 Installing UDUNITS**.

5.8.4.2 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install antlr antlr-static expat-devel libxml2-devel
```

----End

5.8.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Download the PnetCDF installation package.

```
cd /path/to/pnetcdf
wget http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz
```

Step 3 Decompress the PnetCDF installation package.

```
tar xvf pnetcdf-1.11.2.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd pnetcdf-1.11.2
```

Step 5 Compile and install PnetCDF.

```
export CC=mpicc CXX=mpicxx FC=mpifort
./configure --prefix=/path/to/pnetcdf --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test
make -j
make install
```

----End

5.8.4.4 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the HDF5 installation package.

```
cd /path/to/hdf5
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
```

Step 3 Decompress the HDF5 installation package.

```
tar xvf hdf5-1.10.1.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd hdf5-1.10.1
```

Step 5 Compile and install HDF5.

```
export CC=mpicc CXX=mpicxx FC=mpifort
./configure --prefix=/path/to/hdf5 --build=aarch64-unknown-linux-gnu --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id"
sed -i '11835c wl="-Wl,"' libtool
make -j
make install
```

----End

5.8.4.5 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation packages.

```
cd /path/to/netcdf
wget https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5 -O netcdf-fortran-4.4.5.tar.gz
wget https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0 -O netcdf-c-4.7.0.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Decompress the NetCDF-C installation packages.

```
tar xvf netcdf-c-4.7.0.tar.gz
tar xvf netcdf-fortran-4.4.5.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd netcdf-c-4.7.0
```

Step 5 Set the environment variables.

```
export CC=mpicc CXX=mpicxx FC=mpifort
export HDF5_DIR=/path/to/hdf5
export PNETCDF_DIR=/path/to/pnetcdf
```

Step 6 Compile and install NetCDF-C.

```
./configure --prefix=/path/to/netcdf --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CPPFLAGS="-I${HDF5_DIR}/include -I${PNETCDF_DIR}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${PNETCDF_DIR}/lib" CFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -I${HDF5_DIR}/include -I${PNETCDF_DIR}/include"
make -j
make install
```

Step 7 Set the environment variables.

```
export PATH=/path/to/netcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/netcdf/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/netcdf
```

Step 8 Compile and install NetCDF-Fortran.

```
cd ../netcdf-fortran-4.4.5
./configure --prefix=/path/to/netcdf --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS="-I${HDF5_DIR}/include -I${NETCDF}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${NETCDF}/lib" CFLAGS="-L${HDF5_DIR}/HDF5/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" CXXFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" FCFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include"
sed -i '11838c wl="-Wl,"' libtool
make -j
make install
```

----End

5.8.4.6 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the GSL installation package.

```
cd /path/to/gsl
wget http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz
```

Step 3 Decompress the GSL installation package.

```
tar xvf gsl-2.6.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd gsl-2.6
```

Step 5 Compile and install GSL.

```
./configure --prefix=/path/to/gsl
make -j
make install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.8.4.7 Installing UDUNITS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the UDUNITS installation package.

```
cd /path/to/udunits
wget https://artifacts.unidata.ucar.edu/repository/downloads-udunits/2.2.28/udunits-2.2.28.tar.gz
```

Step 3 Decompress the UDUNITS installation package.

```
tar xvf udunits-2.2.28.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd udunits-2.2.28
```

Step 5 Compile and install UDUNITS.

```
./configure --prefix=/path/to/udunits
make -j
make install
```

----End

5.8.5 Obtaining the Source Code

Procedure

Step 1 Download the NCO installation package.

https://github.com/nco/nco/archive/refs/tags/5.1.0.tar.gz

Step 2 Use an SFTP tool to upload the NCO installation package to the */path/to/nco* directory on the server.

----End

5.8.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/nco
```

Step 3 Decompress the installation package.

```
tar xvf 5.1.0.tar.gz
```

Step 4 Go to the directory generated after the decompression.

```
cd nco-5.1.0
```

Step 5 Load the environment variable.

```
export PATH=/path/to/gsl/bin:/path/to/netcdf/bin:$PATH
```

Step 6 Perform the compilation and installation.

```
./configure --prefix=/path/to/nco NETCDF_INC=/path/to/netcdf/include NETCDF_LIB=/path/to/netcdf/lib
NETCDF_ROOT=/path/to/netcdf UDUNITS2_PATH=/path/to/udunits
```

 Copyright © Huawei Technologies Co., Ltd.

```
make -j
make install
```

----End

5.8.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify NCO.

```
cd /path/to/nco/bin
./ncbo --version
```

Information similar to that in the following figure is displayed:

----End

5.9 PETSc 3.16 Porting Guide (Kylin V10)

5.9.1 Introduction

The Portable, Extensible Toolkit for Scientific Computation (PETSc) is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.

For more information, visit the **official PETSc website**.

Programing language: C/Python

Brief description: A suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.

Open source license: custom open source license

Recommended Software Version

PETSc 3.16

5.9.2 Environmental Requirements

Hardware Requirements

Table 5-45 lists the hardware requirements.

Table 5-45 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-46 lists the software requirements.

Table 5-46 Software Requirements

Item	Version	Download URL
PETSc	3.16	https://github.com/petsc/petsc/archive/refs/heads/release-3.16.zip

OS Requirements

Table 5-47 lists the OS requirements.

Table 5-47 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.9.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PETSc software porting.

Table 5-48 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Paths for Software Porting" in **Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/petsc	Installation path of PETSc	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.9.4 Configuring the Compilation Environment

5.9.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-49 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **Basic Environment Setup Guide**.
2	Installing the BiSheng compiler	See **Installing and Using BiSheng Compiler**.
3	Installing Hyper MPI	See "Compiling the Hyper MPI Software Package Using GCC" in the "Installing Hyper MPI Using Source Code" section of the **Hyper MPI User Guide**
4	Installing the dependencies	See **5.9.4.2 Installing the Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

5.9.4.2 Installing the Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install python3
```

----End

5.9.5 Obtaining the Source Code

Procedure

Step 1 Download the PETSc installation package.

https://github.com/petsc/petsc/archive/refs/heads/release-3.16.zip

Step 2 Use an SFTP tool to upload the PETSc installation package to the */path/to/petsc* directory on the server.

----End

5.9.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/petsc
```

Step 3 Decompress the installation package.

```
unzip release-3.16.zip
```

Step 4 Go to the directory generated after the decompression.

```
cd petsc-release-3.16
```

Step 5 Perform the compilation and installation.

```
./configure CC=mpicc CXX=mpicxx FC=mpifort
make all
```

----End

5.9.7 Running and Verification

Procedure

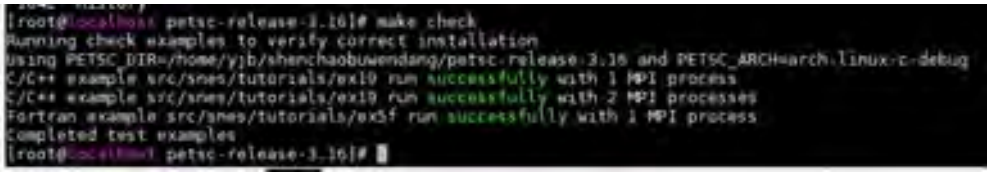
Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify PETSc.

```
cd /path/to/petsc/petsc-release-3.16
export OMPI_ALLOW_RUN_AS_ROOT=1
export OMPI_ALLOW_RUN_AS_ROOT_CONFIRM=1
make check
```

 Copyright © Huawei Technologies Co., Ltd.

Information similar to that in the following figure is displayed:



----End

5.10 Chainer 7.8.1 Porting Guide (Kylin V10)

5.10.1 Introduction

Chainer is a Python-based deep learning framework aiming at flexibility. It provides automatic differentiation APIs based on the define-by-run approach (a.k.a. dynamic computational graphs) as well as object-oriented high-level APIs to build and train neural networks. It also supports CUDA/cuDNN using CuPy for high performance training and inference.

For more information, visit the **Chainer page at GitHub**.

Programming language: Python

Brief description: A deep learning framework aiming at flexibility.

Open source license: custom open source license

Recommended Software Version

Chainer 7.8.1

5.10.2 Environment Requirements

Hardware Requirements

Table 5-50 lists the hardware requirements.

Table 5-50 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-51 Software requirements

Item	Version	Download URL
Chainer	7.8.1	pip install. No downloads are needed.

OS Requirements

Table 5-52 lists the OS requirements.

Table 5-52 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.10.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Chainer software porting.

Table 5-53 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/chainer	Installation path of Chainer	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.10.4 Configuring the Compilation Environment

5.10.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-54 Configuration process

No.	Configuration Item	Remarks
1	Installing dependencies	See **5.10.4.2 Installing Dependencies**.

5.10.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Check whether Python is successfully installed.

```
python3 --version
```

If the correct version is displayed, the installation is successful.

Step 4 Create a Chainer virtual environment.

```
pip3 install virtualenv
mkdir -p /path/to/chainer
python3 -m venv /path/to/chainer
```

----End

5.10.5 Obtaining the Source Code

Use **pip install** for installation. You do not need to download the source code.

5.10.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Activate the environment.

```
source /path/to/chainer/bin/activate
```

Step 3 Go to the Chainer environment.

```
cd /path/to/chainer
```

Step 4 Install Chainer.

```
pip3 install chainer==7.8.1 --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

----End

5.10.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify Chainer.

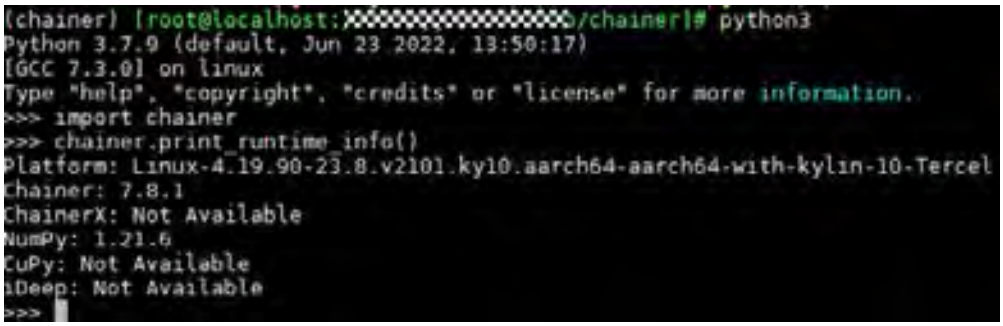
```
python3
```

Type **import chainer** and press **Enter**.

Type **chainer.print_runtime_info()** and press **Enter**.

If no error is reported, the Chainer module is successfully imported.

Information similar to the following is displayed:



----End

5.11 Horovod 0.25.0 Porting Guide (Kylin V10)

5.11.1 Introduction

Horovod is a distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet. The goal of Horovod is to make distributed deep learning fast and easy to use.

For more information, visit the **Horovod page at GitHub**.

Programming language: Python

Brief description: A distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: Apache License v2.0

Recommended Software Version

Horovod 0.25.0

5.11.2 Environment Requirements

Hardware Requirements

Table 5-55 lists the hardware requirements.

Table 5-55 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-56 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz
Hyper MPI	1.1.1	https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz
Python	3.7.5	https://www.python.org/downloads/
Horovod	0.25.0	https://github.com/horovod/horovod

OS Requirements

Table 5-57 lists the OS requirements.

Table 5-57 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

5.11.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Horovod software porting.

Table 5-58 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/HOROVOD	Installation path of Horovod	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.11.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-59 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing Hyper MPI	See "Installing Hyper MPI Using Source Code" in **Hyper MPI Installation Guide**.

 Copyright © Huawei Technologies Co., Ltd.

5.11.5 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 (Optional) Configure the virtual environment if needed.

```
pip install virtualenv
virtualenv Horovod
source Horovod/bin/activate
```

Step 3 Go to the installation directory.

```
cd /path/to/HOROVOD
```

Step 4 Install TensorFlow.

```
pip install tensorflow-2.4.1-cp37-cp37m-manylinux_2_24_aarch64.whl --trusted-host
mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Step 5 Update NumPy.

```
pip install -U numpy  --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/
simple
python3 -c "import tensorflow as tf; print(tf.__version__); print(tf.sysconfig.get_include()); print('
'.join(tf.sysconfig.get_link_flags())); print(' '.join(tf.sysconfig.get_compile_flags()))"
```

If the following information is displayed, the update is successful:

Step 6 Install PyTorch.

```
git clone --depth=1 --recursive  https://github.com/pytorch/pytorch
cd pytorch
pip install pyyaml --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/
simple
python setup.py install
```

Step 7 Install the dependencies.

```
pip install pytest ninja mock --trusted-host mirrors.tools.huawei.com -i http://
mirrors.tools.huawei.com/pypi/simple
```

Step 8 Install Horovod.

```
HOROVOD_WITH_PYTORCH=1 HOROVOD_WITH_TENSORFLOW=1 HOROVOD_WITH_MPI=1 pip install
--no-cache-dir git+https://github.com/horovod/horovod.git@master --trusted-host
mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

----End

5.11.6 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the test script.

```
git clone --recursive https://github.com/horovod/horovod.git
```

Step 3 Go to the test directory.

```
cd horovod/test/parallel
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Execute the script.

```
horovodrun -np 2 pytest -v test_tensorflow.py
```

----End

5.12 Keras 2.9.0 Porting Guide (Kylin V10)

5.12.1 Introduction

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

For more information, visit the **Keras page at GitHub** and **Keras official website**.

Programming language: Python

Brief description: A deep learning API written in Python.

Open source license: custom open source license

Recommended Software Version

Keras 2.9.0

5.12.2 Environment Requirements

Hardware Requirements

Table 5-60 lists the hardware requirements.

Table 5-60 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-61 Software requirements

Item	Version	Download URL
Keras	2.9.0	pip install. No downloads are needed.

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 5-62 lists the OS requirements.

Table 5-62 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.12.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Keras software porting.

Table 5-63 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/keras	Installation path of Keras	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.12.4 Configuring the Compilation Environment

5.12.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-64 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing dependencies	See **5.12.4.2 Installing Dependencies**.

5.12.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Check whether Python is successfully installed.

```
python3 --version
```

If the correct version is displayed, the installation is successful.

Step 4 Create a Keras virtual environment.

```
pip3 install virtualenv
mkdir -p /path/to/keras
python3 -m venv /path/to/keras
source /path/to/keras/bin/activate
```

Step 5 Go to the Keras environment.

```
cd /path/to/keras
```

Step 6 Download the TensorFlow package.

```
wget https://github.com/lhelontra/tensorflow-on-arm/releases/download/v2.4.0/tensorflow-2.4.0-cp37-none-linux_aarch64.whl
```

Step 7 Install TensorFlow.

```
pip3 install tensorflow-2.4.0-cp37-none-linux_aarch64.whl --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

TensorFlow may fail to be installed due to h5py.

Step 8 Download the h5py package and decompress it.

```
wget http://mirrors.tools.huawei.com/pypi/packages/5f/97/a58afbcf40e8abecededd9512978b4e4915374e5b80049af082f49cebe9a/h5py-2.10.0.tar.gz
tar xvf h5py-2.10.0.tar.gz
```

Step 9 Install h5py.

```
cd h5py-2.10.0
python3 setup.py install
```

Information similar to the following is displayed:

```
Finished processing dependencies for h5py==2.10.0
```

 Copyright © Huawei Technologies Co., Ltd.

Step 10 Install TensorFlow again.

```
cd .. pip3 install tensorflow-2.4.0-cp37-none-linux_aarch64.whl --trusted-host
mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Information similar to the following is displayed:

----End

5.12.5 Obtaining the Source Code

Use **pip install** for installation. You do not need to download the source code.

5.12.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the main program installation directory.

```
cd /path/to/keras
```

Step 3 Install Keras.

```
pip3 install keras==2.9.0 --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/
pypi/simple
```

Information similar to the following is displayed:

----End

5.12.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify Keras.

```
python3
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

If no error is reported, the Keras module is successfully imported.

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

```
(keras) [root@localhost:[redacted]/keras]# python3
Python 3.7.9 (default, Jun 23 2022, 13:50:17)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import tensorflow as tf
>>> from tensorflow import keras
>>> from tensorflow.keras import layers
>>> quit()
(keras) [root@localhost:[redacted]/keras]#
```

----End

5.13 Lasagne 0.1 Porting Guide (Kylin V10)

5.13.1 Introduction

Lasagne is a lightweight library to build and train neural networks in Theano. Its main features are:

- Supports feed-forward networks such as Convolutional Neural Networks (CNNs), recurrent networks including Long Short-Term Memory (LSTM), and any combination thereof
- Allows architectures of multiple inputs and multiple outputs, including auxiliary classifiers
- Many optimization methods including Nesterov momentum, RMSprop and ADAM
- Freely definable cost function and no need to derive gradients due to Theano's symbolic differentiation
- Transparent support of CPUs and GPUs due to Theano's expression compiler

For more information, visit the **Lasagne page at GitHub**.

Programming language: Python

Brief description: A lightweight library.

Open source license: custom open source license

Recommended Software Version

Lasagne 0.1

5.13.2 Environment Requirements

Hardware Requirements

Table 5-65 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-65 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-66 Software requirements

Item	Version	Download URL
Lasagne	0.1	https://github.com/Lasagne/Lasagne/archive/master.zip

OS Requirements

Table 5-67 lists the OS requirements.

Table 5-67 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.13.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Lasagne software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-68 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/Lasagne	Installation path of Lasagne	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.13.4 Configuring the Compilation Environment

5.13.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-69 Configuration process

No.	Configuration Item	Remarks
1	Installing Python	See **5.13.4.2 Installing Python**.
2	Installing Theano	See **Theano 1.0.5 Porting Guide (Kylin V10)**.

5.13.4.2 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Check whether Python is successfully installed.

```
python3 --version
```

If the correct version is displayed, the installation is successful.

Step 4 Create a Lasagne virtual environment.

 Copyright © Huawei Technologies Co., Ltd.

```
pip3 install virtualenv
mkdir -p /path/to/Lasagne
python3 -m venv /path/to/Lasagne
```

----End

5.13.5 Obtaining the Source Code

Procedure

Step 1 Download the Lasagne installation package **Lasagne-master.zip**.

https://github.com/Lasagne/Lasagne/archive/master.zip

Step 2 Use an SFTP tool to upload the installation package to the */path/to/Lasagne* directory on the server.

----End

5.13.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Activate the environment.

```
source /path/to/Lasagne/bin/activate
```

Step 3 Go to the Lasagne environment.

```
cd /path/to/Lasagne
```

Step 4 Install Lasagne.

```
pip3 install master.zip --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Information similar to the following is displayed:

----End

5.13.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the master package.

```
unzip master.zip
```

Step 3 Go to the test case directory.

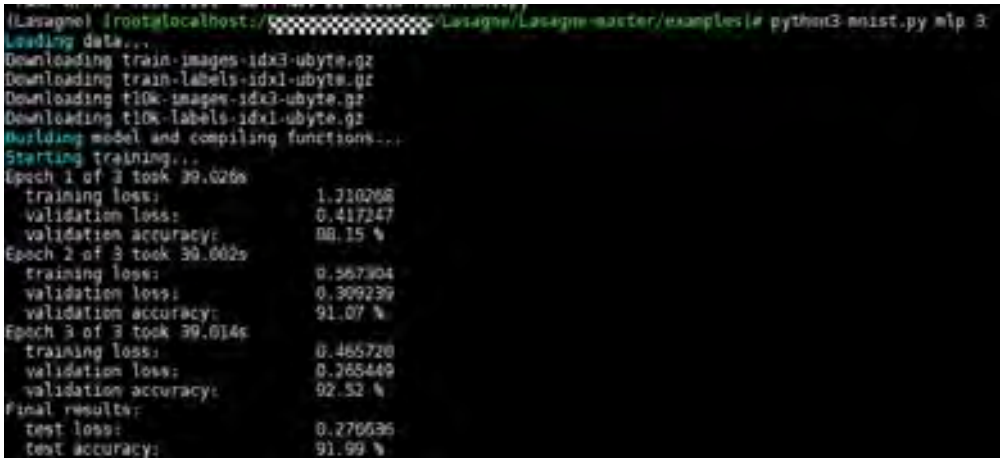
```
cd Lasagne-master/examples
```

 Copyright © Huawei Technologies Co., Ltd.

Step 4 Execute the test case.

```
python3 mnist.py mlp 3
```

Information similar to the following is displayed:



----End

5.14 mpi4py 3.1.3 Porting Guide (Kylin V10)

5.14.1 Introduction

This package provides Python bindings for the Message Passing Interface (MPI) standard. It is implemented on top of the MPI specification and exposes an API which grounds on the standard MPI-2 C++ bindings.

For more information, visit the **mpi4py page at GitHub**.

Programming languages: C/Python/Cyphon

Brief description: Python bindings for the MPI standard.

Open source license: custom open source license

Recommended Software Version

mpi4py 3.1.3

5.14.2 Environment Requirements

Hardware Requirements

Table 5-70 lists the hardware requirements.

Table 5-70 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-71 Software requirements

Item	Version	Download URL
mpi4py	3.1.3	https://github.com/mpi4py/mpi4py/archive/refs/tags/3.1.3.tar.gz

OS Requirements

Table 5-72 lists the OS requirements.

Table 5-72 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.14.3 Paths for Software Porting

This chapter lists the software installation paths involved in the mpi4py software porting.

Table 5-73 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.
2	/path/to/mpi4py	Installation path of mpi4py	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

 Copyright © Huawei Technologies Co., Ltd.

5.14.4 Configuring the Compilation Environment

5.14.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-74 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing Open MPI	See "Compiling and Installing Open MPI" in **OpenMPI 4.0.1 Installation Guide**.
4	Install dependencies	See **5.14.4.2 Installing Dependencies**.

5.14.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install python3-devel python3-pip
pip3 install cython
```

----End

5.14.5 Obtaining the Source Code

Procedure

Step 1 Download the mpi4py installation package **mpi4py-3.1.3.tar.gz**.

https://github.com/mpi4py/mpi4py/archive/refs/tags/3.1.3.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/mpi4py* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

5.14.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/mpi4py
```

Step 3 Decompress the installation package.

```
tar xvf 3.1.3.tar.gz
mv mpi4py-3.1.3 mpi4py
cd mpi4py
```

Step 4 Modify the configuration file.

1. Open **mpi.cfg**.
   ```
   vim mpi.cfg
   ```
2. Press **i** to enter the insert mode and change the path in lines 55 to 61 to the OpenMPI installation path.

   ```
   55 [openmpi]
   56 mpi_dir              = /path/to/OPENMPI
   57 mpicc                = %(mpi_dir)s/bin/mpicc
   58 mpicxx               = %(mpi_dir)s/bin/mpicxx
   59 include_dirs         = %(mpi_dir)s/include
   60 libraries            = mpi
   61 library_dirs         = %(mpi_dir)s/lib
   62 runtime_library_dirs = %(library_dirs)s
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform the compilation and installation.

```
python3 setup.py install
```

If "error:vt-mpi" or "lmpe" is displayed during the compilation and installation, ignore it.

----End

5.14.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify mpi4py.

```
cd /path/to/mpi4py
```

Step 3 Create an example.

1. Create a **test.py** file.
   ```
   vim test.py
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   from mpi4py import MPI
   comm = MPI.COMM_WORLD
   rank = comm.Get_rank()
   print ('My rank is ', rank)
   ```

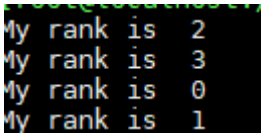
 Copyright © Huawei Technologies Co., Ltd.

3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 4 Perform the verification.

```
mpirun --allow-run-as-root -np 4 python3 test.py
```

Information similar to the following is displayed:



----End

5.15 MXNet 2.0.0 Porting Guide (Kylin V10)

5.15.1 Introduction

MXNet is a deep learning framework designed for both efficiency and flexibility.

For more information, visit the **MXNet official website**.

Programming languages: C++/Python

Brief description: A deep learning framework designed for both efficiency and flexibility.

Open source license: custom open source license

Recommended Software Version

MXNet 2.0.0

5.15.2 Environment Requirements

Hardware Requirements

Table 5-75 lists the hardware requirements.

Table 5-75 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-76 Software requirements

Item	Version	Download URL
MXNet	2.0.0	Run the **git clone** command.

OS Requirements

Table 5-77 lists the OS requirements.

Table 5-77 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.15.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MXNet software porting.

Table 5-78 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/mxnet	Installation path of MXNet	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/opencv	Installation path of OpenCV	

5.15.4 Configuring the Compilation Environment

5.15.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-79 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Install dependencies	See **5.15.4.2 Installing Dependencies**.
3	Installing OpenCV	See **5.15.4.3 Installing OpenCV**.

5.15.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install python3-devel python3-pip lapack-devel openblas-devel openssl-devel zeromq-devel atlas-devel protobuf-devel
```

----End

5.15.4.3 Installing OpenCV

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package.

```
cd /path/to/opencv
wget https://github.com/opencv/opencv/archive/refs/tags/4.5.5.tar.gz -O opencv-4.4.5.tar.gz
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf opencv-4.4.5.tar.gz
cd opencv-4.4.5
```

Step 4 Create a build directory.

```
mkdir build
cd build
```

Step 5 Perform the compilation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/opencv
make -j
make install
```

Step 6 Load the environment variables.

```
export PATH=/path/to/opencv/bin:$PATH
export LD_LIBRARY_PATH=/path/to/opencv/lib64:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.15.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the source code.

```
cd /path/to/mxnet
git clone --recursive https://github.com/apache/incubator-mxnet mxnet
```

----End

5.15.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source code directory.

```
cd /path/to/mxnet/mxnet
```

Step 3 Create a build directory.

```
mkdir build
cd build
```

Step 4 Create a build file.

```
cmake .. -DUSE_CUDA=OFF
```

Step 5 Compile the installation file.

```
make -j
```

Step 6 Load the environment variable.

```
export LD_LIBRARY_PATH=/path/to/mxnet/mxnet/build:$LD_LIBRARY_PATH
```

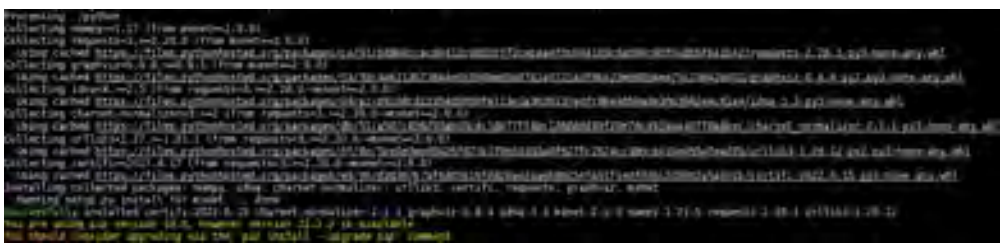
Step 7 Create a MXNet-Python virtual environment and activate it.

```
pip3 install virtualenv
mkdir -p /path/to/mxnet-python
python3 -m venv /path/to/mxnet-python
source /path/to/mxnet-python/bin/activate
```

Step 8 Install MXNet.

```
cd /path/to/mxnet/mxnet
rm -rf python/mxnet/space.py
cp contrib/tvmop/space.py python/mxnet/
python3 -m pip install ./python
```

Information similar to the following is displayed:



----End

 Copyright © Huawei Technologies Co., Ltd.

5.15.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

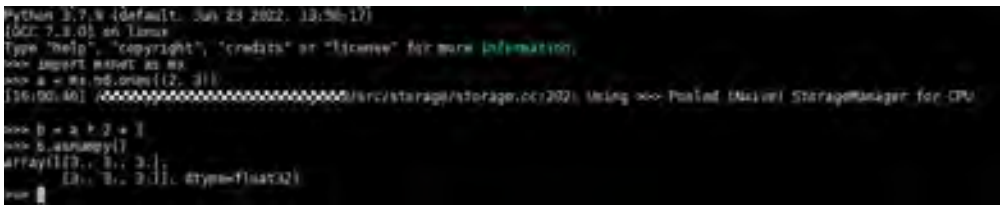
Step 2 Go to the Python interpreter.

```
python3
```

Step 3 Perform the verification.

```
import mxnet as mx
a = mx.nd.ones((2, 3))
b = a * 2 + 1
b.asnumpy()
```

Information similar to the following is displayed:



----End

5.16 sparsehash 2.0.4 Porting Guide (Kylin V10)

5.16.1 Introduction

Sparsehash is an extremely memory-efficient hash_map implementation by Google. The difference between hash and map lies in the implementation at the bottom layer. Hash is generally comprised of array and hash, while map is a red-black tree or another type of tree. You do not need to install the STL library when using hash_map and hash_set. Google provides the entire implementation process. Google uses templates and generic programming during the implementation.

For more information, visit the **sparsehash page at GitHub**.

Programming languages: C/C++

Brief description: An extremely memory-efficient hash_map implementation.

Open source license: custom open source license

Recommended Software Version

sparsehash 2.0.4

5.16.2 Environment Requirements

Hardware Requirements

Table 5-80 lists the hardware requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-80 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-81 Software requirements

Item	Version	Download URL
sparsehash	2.0.4	https://github.com/sparsehash/sparsehash/archive/refs/tags/sparsehash-2.0.4.tar.gz

OS Requirements

Table 5-82 lists the OS requirements.

Table 5-82 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.16.3 Paths for Software Porting

This chapter lists the software installation paths involved in the sparsehash software porting.

Table 5-83 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Software Installation Path	Description	Remarks
2	/path/to/sparsehash	Installation path of sparsehash	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.16.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-84 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.
2	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

5.16.5 Obtaining the Source Code

Procedure

Step 1 Download the sparsehash installation package **sparsehash-2.0.4.tar.gz**.

https://github.com/sparsehash/sparsehash/archive/refs/tags/sparsehash-2.0.4.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/sparsehash* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

5.16.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/sparsehash
```

Step 3 Decompress the installation package.

```
tar xvf sparsehash-2.0.4.tar.gz
cd sparsehash-sparsehash-2.0.4
```

Step 4 Perform the compilation and installation.

```
./configure --prefix=/path/to/sparsehash
make -j
make install
```

----End

5.16.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify sparsehash.

```
cd /path/to/sparsehash/sparsehash-sparsehash-2.0.4
./hashtable_test
./libc_allocator_with_realloc_test
./simple_compat_test
./simple_test
./sparsetable_unittest
./template_util_unittest
./type_traits_unittest
```

The command output of the preceding programs is **pass**.

```
./time_hash_map
```

Information similar to the following is displayed:

```
STANDARD MAP (256 byte objects, 312500 iterations):
map_grow              233.6 ns  (       0 hashes,   312500 copies)
map_predict/grow      259.0 ns  (       0 hashes,   312500 copies)
map_replace           120.6 ns  (       0 hashes,        0 copies)
map_fetch_random      655.1 ns  (       0 hashes,        0 copies)
map_fetch_sequential  141.6 ns  (       0 hashes,        0 copies)
map_fetch_empty         0.8 ns  (       0 hashes,        0 copies)
map_remove            129.6 ns  (       0 hashes,        0 copies)
map_toggle             57.2 ns  (       0 hashes,   312500 copies)
map_iterate            49.5 ns  (       0 hashes,        0 copies)
```

----End

5.17 SUNDIALS 6.3.0 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

5.17.1 Introduction

SUNDIALS is a family of software packages providing robust and efficient time integrators and nonlinear solvers that can easily be incorporated into existing simulation codes. The packages are designed to require minimal information from the user, allow users to supply their own data structures underneath the packages, and enable interfacing with user-supplied or third-party algebraic solvers and preconditioners.

For more information, visit the **SUNDIALS page at GitHub**.

Programming languages: C/Fortran

Brief description: A family of software packages providing robust and efficient time integrators and nonlinear solvers.

Open source license: custom open source license

Recommended Software Version

SUNDIALS 6.3.0

5.17.2 Environment Requirements

Hardware Requirements

Table 5-85 lists the hardware requirements.

Table 5-85 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-86 Software requirements

Item	Version	Download URL
SUNDIALS	6.3.0	**https://github.com/LLNL/sundials/releases/download/v6.3.0/sundials-6.3.0.tar.gz**

OS Requirements

Table 5-87 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-87 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.17.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SUNDIALS software porting.

Table 5-88 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/sundials	Installation path of SUNDIALS	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.17.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-89 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See "Installing BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

 Copyright © Huawei Technologies Co., Ltd.

5.17.5 Obtaining the Source Code

Procedure

Step 1 Download the SUNDIALS installation package **sundials-6.3.0.tar.gz**.

https://github.com/LLNL/sundials/releases/download/v6.3.0/sundials-6.3.0.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/sundials* directory on the server.

----End

5.17.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source package directory.

```
cd /path/to/sundials
```

Step 3 Decompress the installation package and go to the directory generated after the decompression.

```
tar xvf sundials-6.3.0.tar.gz
cd sundials-6.3.0
```

Step 4 Create a build directory and go to it.

```
mkdir build
cd build
```

Step 5 Create a build file.

```
export CC=clang CXX=clang++ FC=flang
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/sundials -DEXAMPLES_INSTALL_PATH=/path/to/sundials/examples
```

Step 6 Perform the compilation and installation.

```
make -j
make install
```

----End

5.17.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the directory of an example.

```
cd /path/to/sundials/examples/arkode/C_manyvector
```

Step 3 Compile the example.

```
make
```

Step 4 Run the example.

```
./ark_brusselator1D_manyvec
```

 Copyright © Huawei Technologies Co., Ltd.

Information similar to the following is displayed:

```
Final Solver Statistics:
   Internal solver steps = 398 (attempted = 471)
   Total RHS evals:  Fe = 2829,  Fi = 8040
   Total linear solver setups = 0
   Total linear iterations = 17325
   Total linear convergence failures = 0
   Total J*v evaluations = 17325
   Total RHS evals in linear solver = 0
   Total number of Newton iterations = 5211
   Total number of nonlinear solver convergence failures = 0
   Total number of error test failures = 73
```

Step 5 Go to the directory of another example.

```
cd ../C_serial
```

Step 6 Compile the example.

```
make
```

Step 7 Run the example.

```
./ark_robertson
```

Information similar to the following is displayed:

```
Final Statistics:
Current time                 = 100378753296.6599
Steps                        = 896
Step attempts                = 1040
Stability limited steps      = 0
Accuracy limited steps       = 1037
Error test fails             = 141
NLS step fails               = 3
Inequality constraint fails  = 0
Initial step size            = 1e-08
Last step size               = 15100776005.32107
Current step size            = 23327099826.86393
Explicit RHS fn evals        = 0
Implicit RHS fn evals        = 30366
NLS iters                    = 25171
NLS fails                    = 54
NLS iters per step           = 28.09263392857143
LS setups                    = 426
Jac fn evals                 = 63
LS RHS fn evals              = 0
Prec setup evals             = 0
Prec solves                  = 0
LS iters                     = 0
LS fails                     = 0
Jac-times setups             = 0
Jac-times evals              = 0
LS iters per NLS iter        = 0
Jac evals per NLS iter       = 0.002502880298756506
Prec evals per NLS iter      = 0
```

----End

5.18 Theano 1.0.5 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

5.18.1 Introduction

Theano is a Python library that allows you to define, optimize, and efficiently evaluate mathematical expressions involving multi-dimensional arrays. It is built on top of NumPy.

For more information, visit the **Theano web page**.

Programming language: Python

Brief description: A Python library that allows you to define, optimize, and efficiently evaluate mathematical expressions involving multi-dimensional arrays.

Open source license: custom open source license

Recommended Software Version

Theano 1.0.5

5.18.2 Environment Requirements

Hardware Requirements

Table 5-90 lists the hardware requirements.

Table 5-90 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-91 Software requirements

Item	Version	Download URL
Theano	1.0.5	pip install. No downloads are needed.

OS Requirements

Table 5-92 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-92 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.18.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Theano software porting.

Table 5-93 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/theano	Installation path of Theano	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.18.4 Configuring the Compilation Environment

5.18.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-94 Configuration process

No.	Configuration Item	Remarks
1	Installing dependencies	See **5.18.4.2 Installing Dependencies**.

 Copyright © Huawei Technologies Co., Ltd.

5.18.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Python.

```
yum install python3-devel python3-pip
```

Step 3 Check whether Python is successfully installed.

```
python3 --version
```

If the correct version is displayed, the installation is successful.

Step 4 Create a Theano virtual environment.

```
pip3 install virtualenv
mkdir -p /path/to/theano
python3 -m venv /path/to/theano
```

----End

5.18.5 Obtaining the Source Code

Use **pip install** for installation. You do not need to download the source code.

5.18.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Activate the environment.

```
source /path/to/theano/bin/activate
```

Step 3 Go to the Theano environment.

```
cd /path/to/theano
```

Step 4 Download SciPy.

```
wget https://github.com/scipy/scipy/releases/download/v1.7.3/scipy-1.7.3-cp37-cp37m-manylinux_2_17_aarch64.manylinux2014_aarch64.whl -O scipy-1.7.3-cp37-cp37m-linux_aarch64.linux_aarch64.whl
```

Step 5 Install SciPy.

```
pip3 install scipy-1.7.3-cp37-cp37m-linux_aarch64.linux_aarch64.whl --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Information similar to the following is displayed:

Step 6 Install Theano.

```
pip3 install Theano==1.0.5 --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.

----End

5.18.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Create a test file, **test.py**.

1. Create **test.py**.
   ```
   vim test.py
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   import theano
   from theano import tensor
   ##declare two symbolic floating-point scalars
   a = tensor.dscalar()
   b = tensor.dscalar()
   ##create a simple expression
   c = a + b
   ##convert the expression into a callable object that takes (a,b)
   ##values as input and computes a value for c
   f = theano.function([a,b], c)
   ##bind 1.5 to 'a', 2.5 to 'b', and evaluate 'c'
   assert 4.0 == f(1.5, 2.5)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 3 Perform a test.

```
python3 test.py
```

No command output is displayed, indicating that the conclusion "1.5 + 2.5 = 4.0" is true.

Step 4 Create a test file, **test2.py**.

1. Create **test2.py**.
   ```
   vim test2.py
   ```
2. Press **i** to enter the insert mode and add the following content to the file:
   ```
   import theano
   from theano import tensor
   ##declare two symbolic floating-point scalars
   a = tensor.dscalar()
   b = tensor.dscalar()
   ##create a simple expression
   c = a + b
   ##convert the expression into a callable object that takes (a,b)
   ##values as input and computes a value for c
   f = theano.function([a,b], c)
   ##bind 1.5 to 'a', 2.5 to 'b', and evaluate 'c'
   assert 4.5== f(1.5, 2.5)
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 5 Perform a test again.

```
python3 test2.py
```

An error is reported, indicating that the conclusion "1.5 + 2.5 = 4.5" is incorrect.

 Copyright © Huawei Technologies Co., Ltd.

```
(theano) [root@localhost:[redacted]/theano]# python3 test2.py
Traceback (most recent call last):
  File "test2.py", line 12, in <module>
    assert 4.5 == f(1.5, 2.5)
AssertionError
```

----End

5.19 Thrust 1.12 Porting Guide (Kylin V10)

5.19.1 Introduction

Thrust is the C++ parallel algorithms library which inspired the introduction of parallel algorithms to the C++ Standard Library. Thrust's high-level interface greatly enhances programmer productivity while enabling performance portability between GPUs and multicore CPUs. It builds on top of established parallel programming frameworks (such as CUDA, TBB, and OpenMP). It also provides a number of general-purpose facilities similar to those found in the C++ Standard Library.

For more information, visit the **Thrust page at GitHub**.

Programming languages: C++/C/CUDA

Brief description: A C++ parallel algorithms library.

Open source license: custom open source license

Recommended Software Version

Thrust 1.12

5.19.2 Environment Requirements

Hardware Requirements

Table lists the hardware requirements.

Table 5-95 Hardware requirements

Item	Remarks
CPU	Huawei Kunpeng 920 processor
GPU	NVIDIA Corporation GV100GL

Software Requirements

Table lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-96 Software requirements

Item	Version	Download URL
Thrust	1.12	Run the **git clone --recursive** command.

OS Requirements

Table 5-97 lists the OS requirements.

Table 5-97 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.19.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Thrust software porting.

Table 5-98 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/thrust	Installation path of Thrust	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.19.4 Configuring the Compilation Environment

5.19.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-99 Configuration process

No.	Configuration Item	Remarks
1	Installing CUDA	See **5.19.4.2 Installing CUDA**.
2	Installing CMake	See **5.19.4.3 Installing CMake**.

5.19.4.2 Installing CUDA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the Nouveau driver is disabled.

```
lsmod |grep nouveau
```

- If a command output is displayed, the Nouveau driver is not disabled. In this case, perform **Step 3**.
- If no command output is displayed, the Nouveau driver is disabled. No further action is required.

Step 3 Disable the Nouveau driver.

1. Create a **disable-nouveau.conf** file.
   ```
   vi /etc/modprobe.d/disable-nouveau.conf
   ```
2. Press **i** to enter the insert mode and add the following content:
   ```
   blacklist nouveau
   options nouveau modeset=0
   ```
3. Press **Esc**, type **wq!**, and press **Enter** to save the file and exit.

Step 4 Back up the **initramfs** file and create a new one.

```
cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
dracut -f /boot/initramfs-$(uname -r).img $(uname -r)
```

Step 5 Restart the computer.

```
reboot
```

Step 6 Check whether the Nouveau driver is disabled (no command output is displayed).

```
lsmod |grep nouveau
```

Step 7 Install NVIDIA CUDA.

```
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/
cuda_11.4.0_470.42.01_linux_sbsa.run
sudo sh cuda_11.4.0_470.42.01_linux_sbsa.run
```

Step 8 Set the environment variables.

```
export CUDA_HOME=/usr/local/cuda
export PATH=$CUDA_HOME/bin:$PATH
export LD_LIBRARY_PATH=$CUDA_HOME/lib64:$LD_LIBRARY_PATH
```

Step 9 Verify the version.

```
nvcc -V
```

If the following information is displayed, the installation is successful:

 Copyright © Huawei Technologies Co., Ltd.

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Wed_Jun__2_19:15:41_PDT_2021
Cuda compilation tools, release 11.4, V11.4.48
Build cuda_11.4.r11.4/compiler.30033411_0
```

----End

5.19.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the CMake installation package.

```
wget https://github.com/Kitware/CMake/releases/download/v3.24.0-rc4/cmake-3.24.0-rc4-linux-aarch64.tar.gz
```

Step 3 Decompress the installation package.

```
tar xvf cmake-3.24.0-rc4-linux-aarch64.tar.gz
```

Step 4 Set the environment variables.

```
cd cmake-3.24.0-rc4-linux-aarch64
export PATH=`pwd`/bin:$PATH
```

Step 5 Verify CMake.

```
cmake --version
```

Information similar to the following is displayed:

```
cmake version 3.24.0-rc4

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

----End

5.19.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the Thrust source code.

```
cd /path/to/thrust
git clone --recursive https://github.com/NVIDIA/thrust.git
```

----End

5.19.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the source code directory.

```
cd /path/to/thrust/thrust
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Create a build directory and go to it.

```
mkdir build
cd build
```

Step 4 Perform the compilation and installation.

```
cmake .. -DCMAKE_INSTALL_PREFIX=/path/to/thrust
make -j 64
make -j install
```

----End

5.19.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify the Thrust version.

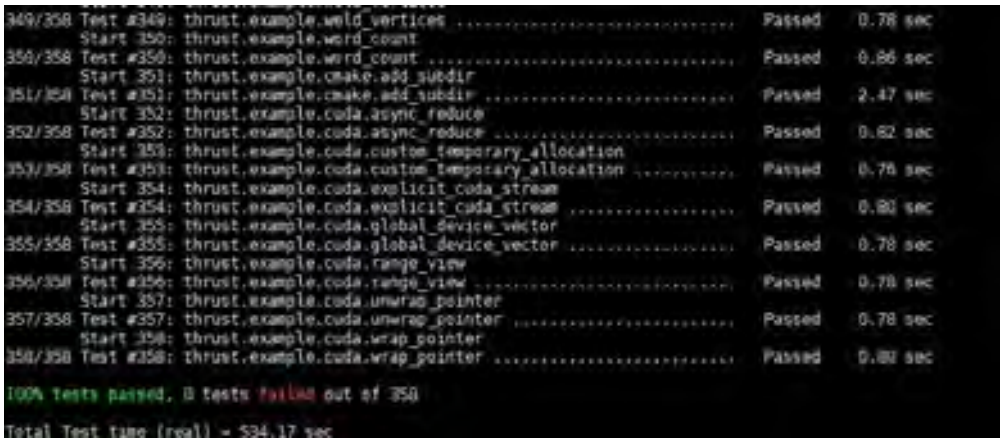
1. Go to the directory.
   ```
   cd /path/to/thrust/include
   ```
2. Create a **version.cu** file.
   ```
   vim version.cu
   ```
3. Press **i** to enter the insert mode and add the following content:
   ```
   #include <thrust/version.h>
   #include <iostream>
   int main(void)
   {
   int major = THRUST_MAJOR_VERSION;
   int minor = THRUST_MINOR_VERSION;
   std::cout << "Thrust v" << major << "." << minor << std::endl;
   return 0;
   }
   ```
4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.
5. Perform the verification.
   ```
   nvcc version.cu -o version
   ./version
   ```
   Information similar to the following is displayed:
   ```
   [root@localhost include]# ./version
   Thrust v1.12
   ```

Step 3 Verify the built-in test case.

```
cd /path/to/thrust/thrust/build/
ctest
```

Information similar to the following is displayed:

 Copyright © Huawei Technologies Co., Ltd.



----End

5.20 Tinker 8.10.3 Porting Guide (Kylin V10)

5.20.1 Introduction

The Tinker molecular modeling software is a complete and general package for molecular mechanics and dynamics, with some special features for biopolymers. Tinker has the ability to use any of several common parameter sets, such as Amber (ff94, ff96, ff98, ff99, ff99SB), CHARMM (19, 22, 22/CMAP), Allinger MM (MM2-1991 and MM3-2000), OPLS (OPLS-UA, OPLS-AA), Merck Molecular Force Field (MMFF), Liam Dang's polarizable model, and the AMOEBA, AMOEBA+ and HIPPO polarizable atomic multipole force fields.

For more information, visit the **Tinker page at GitHub**.

Programming languages: C/Fortran

Brief description: A molecular modeling tool.

Open source license: custom open source license

Recommended Software Version

Tinker 8.10.3

5.20.2 Environment Requirements

Hardware Requirements

Table 5-100 lists the hardware requirements.

Table 5-100 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table lists the software requirements.

Table 5-101 Software requirements

Item	Version	Download URL
Tinker	8.10.3	Run the **git clone** command.

OS Requirements

Table 5-102 lists the OS requirements.

Table 5-102 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-23.8.v2101.ky10.aarch64	Included in the OS image

5.20.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Tinker software porting.

Table 5-103 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/tinker	Installation path of Tinker	The installation paths provided here are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.20.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

 Copyright © Huawei Technologies Co., Ltd.

Configuration Process

Table 5-104 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

5.20.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Obtain the source code.

```
cd /path/to/tinker
git clone https://github.com/TinkerTools/tinker.git
```

----End

5.20.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies.

```
yum install fftw*
```

Step 3 Go to the source code directory.

```
cd /path/to/tinker/tinker
```

Step 4 Copy the build files to the build directory.

```
cp make/Makefile source/
```

Step 5 Modify the build file.

1. Go to the **source** directory.
   ```
   cd source
   ```
2. Open **Makefile**.
   ```
   vim Makefile
   ```
3. Press **i** to enter the insert mode, change line 44 to **TINKERDIR = /path/to/tinker**, and change line 90 to **OPTFLAGS = -Ofast -fopenmp**.
4. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

Step 6 Perform the compilation and installation.

```
make -j
make install
```

Step 7 Load the environment variable.

```
export PATH=/path/to/tinker/tinker/bin:$PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.20.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the test directory.

```
cd /path/to/tinker/tinker/test
```

Step 3 Run the test case.

```
./water.run
```

Information similar to the following is displayed:

```
Average Electrostatic Potential over Atoms :
     (Kcal/mole per unit charge)

  Structure    Atom      Points      Potential        Target        RMS Diff

       1         1         1887        -9.2072       -9.1967         0.0610
       1         2         1237         7.0443        7.0521         0.0399
       1         3         1233         7.0424        7.0503         0.0413

Electrostatic Potential over all Grid Points :

Average Potential Value for Model :                  0.0053
Average Potential Magnitude for Model :              8.1794
Average Potential Value for Target :                 0.0143
Average Potential Magnitude for Target :             8.1777

Average Signed Potential Difference :               -0.0090
Average Unsigned Potential Difference :              0.0353
Root Mean Square Potential Difference :              0.0505
```

Step 4 Run the test case.

```
./salt.run
```

Information similar to the following is displayed:

```
Optimally Conditioned Variable Metric Optimization :

VM Iter     F Value      G RMS      F Move   X Move      Angle   FG Call

    0     -751.1121     9.3253                                       1
    1     -752.4582     6.9569      1.3461   0.0166     0.0000       2
    2     -754.3516     1.0706      1.8934   0.0487     0.0000       3
    3     -754.4054     0.1518      0.0538   0.0089     0.0000       4
    4     -754.4065     0.0041      0.0011   0.0015     0.0000       5
    5     -754.4065     0.0000      0.0000   0.0000     0.0008       6

OCVM  --  Normal Termination due to SmallGrad

Final Potential Function Value :       -754.4065
Final RMS Coordinate Gradient :       0.6493D-07
Final Coordinate Gradient Norm :      0.1836D-06
Final RMS Lattice Gradient :          0.2063D-04
Final Lattice Gradient Norm :         0.5054D-04

Final Lattice Dimensions :      a          5.7660
                                b          5.7660
                                c          5.7660
                             Alpha        90.0000
                             Beta         90.0000
                             Gamma        90.0000
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.21 west Porting Guide (Kylin V10)

5.21.1 Introduction

West lets you manage multiple Git repositories under a single directory using a single file, called the west manifest file, or manifest for short.

For more information, visit the **west page at GitHub**.

Programming language: Python

Brief description: A Zephyr RTOS meta tool.

Open source license: Apache License 2.0

Recommended Software Version

N/A

5.21.2 Environment Requirements

Hardware Requirements

Table 5-105 lists the hardware requirements.

Table 5-105 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table lists the software requirements.

Table 5-106 Software requirements

Item	Version	Download URL
GCC for openEuler	10.3.1	**https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2021.09-aarch64-linux.tar.gz**
Python	3.7.5	**https://www.python.org/downloads/**

OS Requirements

Table 5-107 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-107 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

5.21.3 Paths for Software Porting

This chapter lists the software installation paths involved in the west software porting.

Table 5-108 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	-	Installation path of each software package for setting up the basic environment	See "Planning the Installation Paths" in **HPC Solution Basic Environment Setup Guide**.

5.21.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-109 Configuration process

No.	Configuration Item	Remarks
1	Setting up the basic environment	See "Setting Up the Environment for the Cluster Scenario" in **HPC Solution Basic Environment Setup Guide**.

5.21.5 Obtaining the Source Code

Run the **pip** command for the installation. You do not need to obtain the source code in advance.

 Copyright © Huawei Technologies Co., Ltd.

5.21.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 (Optional) Configure the virtual environment if needed.

```
pip install virtualenv
virtualenv west
source west/bin/activate
```

Step 3 Install west.

```
pip3 install west
```

Information similar to the following is displayed:

----End

5.21.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify the version.

```
west -V
```

If the following information is displayed, the installation is successful:

```
West version: v0.14.0
```

----End

5.22 Crystal 1.7.2 Porting Guide (Kylin V10)

5.22.1 Introduction

Crystal is a programming language with the following goals:

- Have a syntax similar to Ruby (but compatibility with it is not a goal).

 Copyright © Huawei Technologies Co., Ltd.

- Statically type-checked but without having to specify the type of variables or method arguments.
- Be able to call C code by writing bindings to it in Crystal.
- Have compile-time evaluation and generation of code, to avoid boilerplate code.
- Compile to efficient native code.

For more information, visit the **Crystal official website**.

Programming language: Crystal

Brief description: A programming language.

Open source license: Apache 2.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

1.7.2

5.22.2 Environment Requirements

Hardware Requirements

Table 5-110 lists the hardware requirements.

Table 5-110 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-111 lists the OS requirements.

Table 5-111 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19	Included in the OS image

Software Requirements

Table 5-112 lists the software requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-112 Software requirements

Item	Version	Download URL
Crystal	1.7.2	Obtain it from the Docker image. You do not need to download it.
docker	18.09.8	**https://download.docker.com/linux/static/stable/aarch64/docker-18.09.8.tgz**

5.22.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Crystal software porting.

Table 5-113 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/crystal	Installation path of Crystal.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.22.4 Configuring the Compilation Environment

5.22.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-114 Configuration process

No.	Configuration Item	Remarks
1	Installing Docker	See **5.22.4.2 Installing Docker**.

 Copyright © Huawei Technologies Co., Ltd.

5.22.4.2 Installing Docker

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the installation package and decompress it.

```
cd /path/to
wget https://download.docker.com/linux/static/stable/aarch64/docker-18.09.8.tgz
tar -xzvf docker-18.09.8.tgz
```

Step 3 Configure Docker.

```
cp -p docker/* /usr/bin/
vim /lib/systemd/system/docker.service
```

Add the following content and save the file.

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target
[Service]
Type=notify
ExecStart=/usr/bin/dockerd
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=infinity
LimitNPROC=infinity
TimeoutStartSec=0
Delegate=yes
KillMode=process
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s
[Install]
WantedBy=multi-user.target
```

Step 4 Start the Docker service.

```
systemctl start docker.service
```

Step 5 View the Docker version.

```
docker version
```

If the version **18.09.8** in displayed in the command output, the installation is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
Client: Docker Engine - Community
 Version:           18.09.8
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        0dd43dd
 Built:             Wed Jul 17 17:39:22 2019
 OS/Arch:           linux/arm64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.8
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       0dd43dd
  Built:            Wed Jul 17 17:53:39 2019
  OS/Arch:          linux/arm64
  Experimental:     false
```

----End

5.22.5 Obtaining the Source Code

The software is installed using Docker. You do not need to download the source code.

5.22.6 Compilation and Installation

Procedure

Step 1 Pull a Crystal image.

```
docker pull 84codes/crystal:latest-fedora-37
```

Step 2 View the image.

```
docker images
```

If the following information is displayed, the pull is successful.

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
84codes/crystal     latest-fedora-37    5758b6ca216f        7 days ago          564MB
```

----End

5.22.7 Running and Verification

Procedure

Step 1 Obtain the built-in test cases.

```
cd /path/to/
git clone https://github.com/crystal-lang/crystal.git
```

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Run Crystal using a container.

```
docker run -it 84codes/crystal:latest-fedora-37
```

Step 3 View the container ID.

```
docker ps -a
```

If **CONTAINER ID** is displayed in the command output, the container is running properly.

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                              COMMAND             CREATED
caf67ade6d37        84codes/crystal:latest-fedora-37   "/usr/bin/crystal"   5 seconds ago
```

Step 4 Export the container and import the test case to re-create a container.

```
docker export caf67ade6d37 > crystal.tar
mkdir crystal-example
tar xf crystal.tar -C crystal-example
rm -rf crystal.tar
cp -r crystal/samples/* crystal-example/home
cd crystal-example
tar cf crystal.tar *
```

Step 5 Import the container.

```
docker import - crystal:example < crystal.tar
docker images
```

If the following information is displayed, the import is successful.

```
[root@localhost crystal-example]# docker import - crystal:example < crystal.tar
sha256:11f6d7a6c06990051eaaa8f2e111cb11fa8d634cc8234990363f4dc27ee22b74
[root@localhost crystal-example]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
crystal             example             11f6d7a6c069        5 seconds ago       538MB
84codes/crystal     latest-fedora-37    5758b6ca216f        7 days ago          564MB
```

Step 6 Use Crystal to run the **binary-trees.cr**, **red_black_tree.cr**, and **pig.cr** test cases.

```
docker run -it crystal:example /usr/bin/crystal run /home/binary-trees.cr
```

If the following information is displayed, the running is successful.

```
[root@localhost crystal-example]# docker run -it crystal:example /usr/bin/crystal run /home/binary-trees.cr
stretch tree of depth 16        check: -1
65536     trees of depth 4      check: -65536
16384     trees of depth 6      check: -16384
4096      trees of depth 8      check: -4096
1024      trees of depth 10     check: -1024
256       trees of depth 12     check: -256
64        trees of depth 14     check: -64
16        trees of depth 16     check: -16
long lived tree of depth 15     check: -1
```

```
docker run -it crystal:example /usr/bin/crystal run /home/red_black_tree.cr
```

If the following information is displayed, the running is successful.

```
[root@localhost crystal-example]# docker run -it crystal:example /usr/bin/crystal run /home/red_black_tree.cr
delete: 00:00:01.318483000, res: 0
add: 00:00:00.665806070, res: 100000
search: 00:00:00.390581850, res: [63464, 100000]
walk: 00:00:01.122804670, res: [99933, 100000]
reverse_walk: 00:00:01.135076430, res: [99933, 100000]
min: 00:00:02.678430529, res: 100000
max: 00:00:03.193816510, res: 999900000
summary time: 00:00:09.516368530
```

```
docker run -it crystal:example /usr/bin/crystal run /home/pig.cr
```

If the following information is displayed, the running is successful.

 Copyright © Huawei Technologies Co., Ltd.

```
[root@localhost crystal-example]# docker run -it crystal:example /usr/bin/crystal run /home/pig.cr
Wins, losses staying at k = 1: 241/990 %24.3, 749/990 %75.6
Wins, losses staying at k = 2: 256/990 %25.8, 734/990 %74.1
Wins, losses staying at k = 3: 276/990 %27.8, 714/990 %72.1
Wins, losses staying at k = 4: 346/990 %34.9, 644/990 %65.0
Wins, losses staying at k = 5: 400/990 %40.4, 590/990 %59.5
Wins, losses staying at k = 6: 456/990 %46.0, 534/990 %53.9
Wins, losses staying at k = 7: 489/990 %49.3, 501/990 %50.6
Wins, losses staying at k = 8: 541/990 %54.6, 449/990 %45.3
Wins, losses staying at k = 9: 582/990 %58.7, 408/990 %41.2
Wins, losses staying at k = 10: 605/990 %61.1, 385/990 %38.8
Wins, losses staying at k = 11: 594/990 %60.0, 396/990 %40.0
Wins, losses staying at k = 12: 633/990 %63.9, 357/990 %36.0
Wins, losses staying at k = 13: 652/990 %65.8, 338/990 %34.1
Wins, losses staying at k = 14: 652/990 %65.8, 338/990 %34.1
```

----End

5.22.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

5.23 GRIB API 1.21.0 Porting Guide (Kylin V10)

5.23.1 Introduction

GRIB API is the encoding/decoding software for GRIB editions 1 and 2 developed at ECMWF.

For more information, visit the **GRIB API page at GitHub**.

Programming language: C

Brief description: Encoding/decoding software for GRIB.

Open source license: Apache 2.0

Recommended Software Version

GRIB API 1.21.0

5.23.2 Environment Requirements

Hardware Requirements

Table 5-115 lists the hardware requirements.

Table 5-115 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

 Copyright © Huawei Technologies Co., Ltd.

OS Requirements

Table 5-116 lists the OS requirements.

Table 5-116 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90	Included in the OS image

Software Requirements

Table 5-117 lists the software requirements.

Table 5-117 Software requirements

Item	Version	Download URL
BiSheng Compiler	2.4.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/
Hyper MPI	1.2.0	https://github.com/kunpengcompute/hmpi/tags
GRIB API	1.21.0	https://github.com/weathersource/grib_api/archive/refs/tags/v1.21.0.tar.gz
HDF5	1.10.1	https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
PnetCDF	1.11.2	http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz
NetCDF-C	4.7.0	https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0
NetCDF-Fortran	4.4.5	https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5

5.23.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GRIB API software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-118 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/grib_api	Installation path of GRIB API.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.23.4 Configuring the Compilation Environment

5.23.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-119 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing HMPI	See **Installing HMPI**.
3	Installing dependencies	See **5.23.4.2 Installing Dependencies**.
4	Installing HDF5	See **5.23.4.3 Installing HDF5**.
5	Installing PnetCDF	See **5.23.4.4 Installing PnetCDF**.
6	Installing NetCDF	See **5.23.4.5 Installing NetCDF**.

5.23.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

 Copyright © Huawei Technologies Co., Ltd.

```
yum install jasper-devel perl-devel libaec-devel openjpeg-devel
```

----End

5.23.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the HDF5 installation package and decompress it.

```
cd /path/to
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
tar xf hdf5-1.10.1.tar.gz
```

Step 3 Go to the source package directory and perform the installation.

```
cd hdf5-1.10.1
CC=mpicc CXX=mpicxx FC=mpif90 F77=mpif90  ./configure --prefix=/path/to/hdf5 --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id"
sed -i '11835c wl="-Wl,"' libtool
make -j 64
make install
```

Step 4 Set the environment variables.

```
export PATH=/path/to/hdf5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/hdf5/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hdf5/include:$INCLUDE
```

----End

5.23.4.4 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the PnetCDF installation package and decompress it.

```
cd /path/to
wget http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz
tar xf pnetcdf-1.11.2.tar.gz
```

Step 3 Go to the source package directory and perform the installation.

```
cd pnetcdf-1.11.2
CC=mpicc CXX=mpicxx FC=mpif90 F77=mpif90 ./configure --prefix=/path/to/pnetcdf --enable-shared --enable-fortran --enable-large-file-test
make -j 64
make install
```

Step 4 Set the environment variables.

```
export PATH=/path/to/pnetcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/pnetcdf/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/pnetcdf/include:$INCLUDE
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.23.4.5 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the NetCDF installation package and decompress it.

```
cd /path/to
wget https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5 -O netcdf-fortran-4.4.5.tar.gz
wget https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0 -O netcdf-c-4.7.0.tar.gz
tar xf netcdf-fortran-4.4.5.tar.gz
tar xf netcdf-c-4.7.0.tar.gz
```

Step 3 Go to the NetCDF-C source package directory and perform the installation.

```
export HDF5_DIR=/path/to/hdf5
export PNETCDF_DIR=/path/to/pnetcdf
cd netcdf-c-4.7.0
CC=mpicc ./configure --prefix=/path/to/netcdf --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CPPFLAGS="-I${HDF5_DIR}/include -I${PNETCDF_DIR}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${PNETCDF_DIR}/lib" CFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -I${HDF5_DIR}/include -I${PNETCDF_DIR}/include"
make -j 64
make install
```

Step 4 Go to the NetCDF-Fortran source package directory and perform the installation.

```
export NETCDF=/path/to/netcdf
export PATH=/path/to/netcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/netcdf/lib:$LD_LIBRARY_PATH
cd ../netcdf-fortran-4.4.5
CC=mpicc FC=mpif90 F77=mpif90 ./configure --prefix=${NETCDF} --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS="-I${HDF5_DIR}/include -I${NETCDF}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${NETCDF}/lib" CFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" CXXFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" FCFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include"
make -j 64
make install
```

Step 5 Set the environment variables.

```
export INCLUDE=/path/to/netcdf/include:$INCLUDE
```

----End

5.23.5 Obtaining the Source Code

Procedure

Download the GRIB API installation package.

```
wget https://github.com/weathersource/grib_api/archive/refs/tags/v1.21.0.tar.gz
```

5.23.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the GRIB API source package.

 Copyright © Huawei Technologies Co., Ltd.

```
cd /path/to
tar xf v1.21.0.tar.gz
```

Step 3 Install GRIB API.

```
cd grib_api-1.21.0
CC=mpicc F77=mpifort FC=mpifort ./configure --with-netcdf=/path/to/netcdf --prefix=/path/to/
grib_api --build=aarch64-unknown-linux-gnu
sed -i '10270c wl="-Wl,"' libtool
make
make install
```

----End

5.23.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify GRIB API.

```
cd /path/to/grib_api/bin
./grib_ls /path/to/grib_api/share/grib_api/samples/sh_sfc_grib1.tmpl
./grib_compare  /path/to/grib_api/share/grib_api/samples/sh_sfc_grib1.tmpl  /path/to/grib_api/share/
grib_api/samples/sh_sfc_grib2.tmpl
```

If the following information is displayed, the verification is successful.

----End

5.24 IPM 2.0.6 Porting Guide (Kylin V10)

5.24.1 Introduction

IPM: Integrated Performance Monitoring for High Performance Computing.

For more information, visit the **IPM page at GitHub**.

Programming language: C/Perl

 Copyright © Huawei Technologies Co., Ltd.

Brief description: Integrated Performance Monitoring for High Performance Computing.

Open source license: LGPL 2.1

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.0.6

5.24.2 Environment Requirements

Hardware Requirements

Table 5-120 lists the hardware requirements.

Table 5-120 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-121 lists the OS requirements.

Table 5-121 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 5-122 lists the software requirements.

Table 5-122 Software requirements

Item	Version	Download URL
IPM	2.0.6	https://github.com/nerscadmin/IPM/archive/refs/tags/2.0.6.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz
HMPI	1.2.1	https://gitee.com/kunpengcompute/hmpi/archive/refs/tags/v1.2.1-huawei.tar.gz

5.24.3 Paths for Software Porting

This chapter lists the software installation paths involved in the IPM software porting.

Table 5-123 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/IPM	Installation path of IPM.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.24.4 Configuring the Compilation Environment

5.24.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-124 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

 Copyright © Huawei Technologies Co., Ltd.

No.	Configuration Item	Remarks
2	Installing HMPI	See **Installing HMPI**.
3	Installing Perl	See **5.24.4.2 Installing Perl**.

5.24.4.2 Installing Perl

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install Perl.

```
yum install -y perl perl-devel perl-CPAN
```

----End

5.24.5 Obtaining the Source Code

Procedure

Step 1 Download the IPM installation package.

https://github.com/nerscadmin/IPM/archive/refs/tags/2.0.6.tar.gz

Step 2 Use an SFTP tool to upload the IPM installation package to the */path/to/IPM* directory on the server.

----End

5.24.6 Compilation and Installation

Procedure

Step 1 Decompress the installation package.

```
cd /path/to/IPM
tar xvf IPM-2.0.6.tar.gz
cd IPM-2.0.6
```

Step 2 Modify the related file and generate **Makefile**.

```
/bootstrap.sh
sed -i "5749cint main(int argc, char *argv[]) {" configure
sed -i '11837c wl="-Wl,"' libtool
sed -i '11989c wl="-Wl,"' libtool
CC=mpicc CXX=mpicxx FC=mpif90 F77=mpif90 ./configure --prefix=/path/to/IPM --build=aarch64-unknown-linux-gnu --enable-mpi
```

Step 3 Perform the compilation and installation.

```
make
make install
```

Step 4 Set the environment variables.

```
export LD_LIBRARY_PATH=/path/to/IPM/lib:$LD_LIBRARY_PATH
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.24.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/IPM/IPM-2.0.6/test/test.hello
mpicc main.c -o main.x -L/path/to/IPM/lib -lipm
./main.x
```

If the following information is displayed, the verification is successful.

```
[root@localhost test.hello]# ./main.x
Hello from rank 0 of 1
##IPMv2.0.6############################################################
#
# command   : ./main.x
# start     : Mon Jan 16 15:14:41 2023   host      : localhost.local
# stop      : Mon Jan 16 15:14:41 2023   wallclock : 0.14
# mpi_tasks : 1 on 1 nodes               %comm     : 0.00
# mem [GB]  : 0.04                       gflop/sec : 0.00
#
#           :       [total]        <avg>          min          max
# wallclock :          0.14         0.14         0.14         0.14
# MPI       :          0.00         0.00         0.00         0.00
# %wall     :
#   MPI     :                       0.00         0.00         0.00
# #calls    :
#   MPI     :             4            4            4            4
# mem [GB]  :          0.04         0.04         0.04         0.04
#
########################################################################
```

5.24.8 Change History

Date	Description
2023-04-25	This issue is the first official release.

5.25 ncview 2.1.5 Porting Guide (Kylin V10)

5.25.1 Introduction

ncview is a visual browser for NetCDF format files. Typically you would use ncview to get a quick and easy, push-button look at your NetCDF files.

For more information, visit the **ncview website**.

Programming language: C

Brief description: A visual browser for NetCDF format files.

Open source license: GNU GPL v3

NOTE

When using open source software, comply with the applicable licenses.

 Copyright © Huawei Technologies Co., Ltd.

Recommended Software Version

2.1.5

5.25.2 Environment Requirements

Hardware Requirements

Table 5-125 lists the hardware requirements.

Table 5-125 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-126 lists the OS requirements.

Table 5-126 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 5-127 lists the software requirements.

Table 5-127 Software requirements

Item	Version	Download URL
ncview	2.1.5	ftp://cirrus.ucsd.edu/pub/ncview/ncview-2.1.5.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz
HMPI	1.2.1	https://gitee.com/kunpengcompute/hmpi/archive/refs/tags/v1.2.1-huawei.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

5.25.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ncview software porting.

Table 5-128 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/ncview	Installation path of ncview.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.25.4 Configuring the Compilation Environment

5.25.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

The server is connected to the Internet.

Configuration Process

Table 5-129 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing HMPI	See **Installing HMPI**.
3	Installing Dependencies	See **5.25.4.2 Installing Dependencies**.
3	Installing HDF5	See **5.25.4.3 Installing HDF5**.
4	Installing PnetCDF	See **5.25.4.4 Installing PnetCDF**.
5	Installing NetCDF	See **5.25.4.5 Installing NetCDF**.
6	Installing libexpat	See **5.25.4.6 Installing libexpat**.
7	Installing UDUNITS	See **5.25.4.7 Installing UDUNITS**.

 Copyright © Huawei Technologies Co., Ltd.

5.25.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install cmake curl curl-devel zlib zlib-devel libXt-devel libXaw-devel libpng-devel xorg-x11-xauth xorg-x11-server-utils xorg-x11-server-Xnest libXtst
```

----End

5.25.4.3 Installing HDF5

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz
tar xf hdf5-1.10.1.tar.gz
```

Step 2 Go to the source package directory and perform the installation.

```
cd hdf5-1.10.1
CC=mpicc CXX=mpicxx FC=mpif90 F77=mpif90  ./configure --prefix=/path/to/hdf5 --enable-fortran --enable-static=yes --enable-parallel --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-null-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id"
sed -i '11835c wl="-Wl,"' libtool
make -j 64
make install
```

Step 3 Set the environment variables.

```
export PATH=/path/to/hdf5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/hdf5/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/hdf5/include:$INCLUDE
```

----End

5.25.4.4 Installing PnetCDF

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz
tar xf pnetcdf-1.11.2.tar.gz
```

Step 2 Go to the source package directory and perform the installation.

```
cd pnetcdf-1.11.2
CC=mpicc CXX=mpicxx FC=mpif90 F77=mpif90 ./configure --prefix=/path/to/pnetcdf --enable-shared --enable-fortran --enable-large-file-test
make -j 64
make install
```

Step 3 Set the environment variables.

```
export PATH=/path/to/pnetcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/pnetcdf/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/pnetcdf/include:$INCLUDE
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.25.4.5 Installing NetCDF

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5 -O netcdf-fortran-4.4.5.tar.gz
wget https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0 -O netcdf-c-4.7.0.tar.gz
tar xf netcdf-fortran-4.4.5.tar.gz
tar xf netcdf-c-4.7.0.tar.gz
```

Step 2 Go to the NetCDF-C source package directory and perform the installation.

```
export HDF5_DIR=/path/to/hdf5
export PNETCDF_DIR=/path/to/pnetcdf
cd netcdf-c-4.7.0
CC=mpicc ./configure --prefix=/path/to/netcdf --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CPPFLAGS="-I${HDF5_DIR}/include -I${PNETCDF_DIR}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${PNETCDF_DIR}/lib" CFLAGS="-L${HDF5_DIR}/lib -L${PNETCDF_DIR}/lib -I${HDF5_DIR}/include -I${PNETCDF_DIR}/include"
make -j 64
make install
```

Step 3 Go to the NetCDF-Fortran source package directory and perform the installation.

```
export NETCDF=/path/to/netcdf
export PATH=/path/to/netcdf/bin:$PATH
export LD_LIBRARY_PATH=/path/to/netcdf/lib:$LD_LIBRARY_PATH
cd ../netcdf-fortran-4.4.5
CC=mpicc FC=mpif90 F77=mpif90 ./configure --prefix=${NETCDF} --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS="-I${HDF5_DIR}/include -I${NETCDF}/include" LDFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -Wl,-rpath=${HDF5_DIR}/lib -Wl,-rpath=${NETCDF}/lib" CFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" CXXFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include" FCFLAGS="-L${HDF5_DIR}/lib -L${NETCDF}/lib -I${HDF5_DIR}/include -I${NETCDF}/include"
make -j 64
make install
```

Step 4 Set the environment variables.

```
export INCLUDE=/path/to/netcdf/include:$INCLUDE
```

----End

5.25.4.6 Installing libexpat

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://github.com/libexpat/libexpat/archive/refs/tags/R_2_4_7.tar.gz -O libexpat-2.4.7.tar.gz
tar xf libexpat-2.4.7.tar.gz
```

Step 2 Go to the source package directory and perform the installation.

```
cd libexpat-R_2_4_7/expat
cmake . -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_INSTALL_PREFIX=/path/to/libexpat
make -j 64
make install
```

Step 3 Set the environment variables.

 Copyright © Huawei Technologies Co., Ltd.

```
export PATH=/path/to/libexpat/bin:$PATH
export LD_LIBRARY_PATH=/path/to/libexpat/lib64:$LD_LIBRARY_PATH
export INCLUDE=/path/to/libexpat/include:$INCLUDE
```

----End

5.25.4.7 Installing UDUNITS

Procedure

Step 1 Download the installation package.

```
cd /path/to
wget https://artifacts.unidata.ucar.edu/repository/downloads-udunits/2.2.28/udunits-2.2.28.tar.gz
tar xf udunits-2.2.28.tar.gz
```

Step 2 Go to the source package directory and perform the installation.

```
cd udunits-2.2.28
CC=clang CXX=clang++ ./configure LDFLAGS="-L/path/to/libexpat/lib64" CPPFLAGS="-I/path/to/libexpat/
include" --prefix=/path/to/udunits
make -j 64
make install
```

Step 3 Set the environment variables.

```
export PATH=/path/to/udunits/bin:$PATH
export LD_LIBRARY_PATH=/path/to/udunits/lib:$LD_LIBRARY_PATH
export INCLUDE=/path/to/udunits/include:$INCLUDE
```

----End

5.25.5 Obtaining the Source Code

Procedure

Step 1 Download the ncview installation package.

ftp://cirrus.ucsd.edu/pub/ncview/ncview-2.1.5.tar.gz

Step 2 Use an SFTP tool to upload the ncview installation package to the */path/to/ncview* directory on the server.

----End

5.25.6 Compilation and Installation

Procedure

Step 1 Decompress the source package.

```
cd /path/to
tar xf ncview-2.1.5.tar.gz
```

Step 2 Go to the source package directory and then perform the compilation and installation.

```
cd ncview-2.1.5
CC=mpicc CXX=mpicxx FC=mpifort ./configure LDFLAGS="-L/path/to/libexpat/lib64" CPPFLAGS="-I/
path/to/libexpat/include" --prefix=/path/to/ncview --with-nc-config=/path/to/netcdf/bin/nc-config --
with-udunits2_incdir=/path/to/udunits/include --with-udunits2_libdir=/path/to/udunits/lib
```

 Copyright © Huawei Technologies Co., Ltd.

```
make -j 64
make install
```

----End

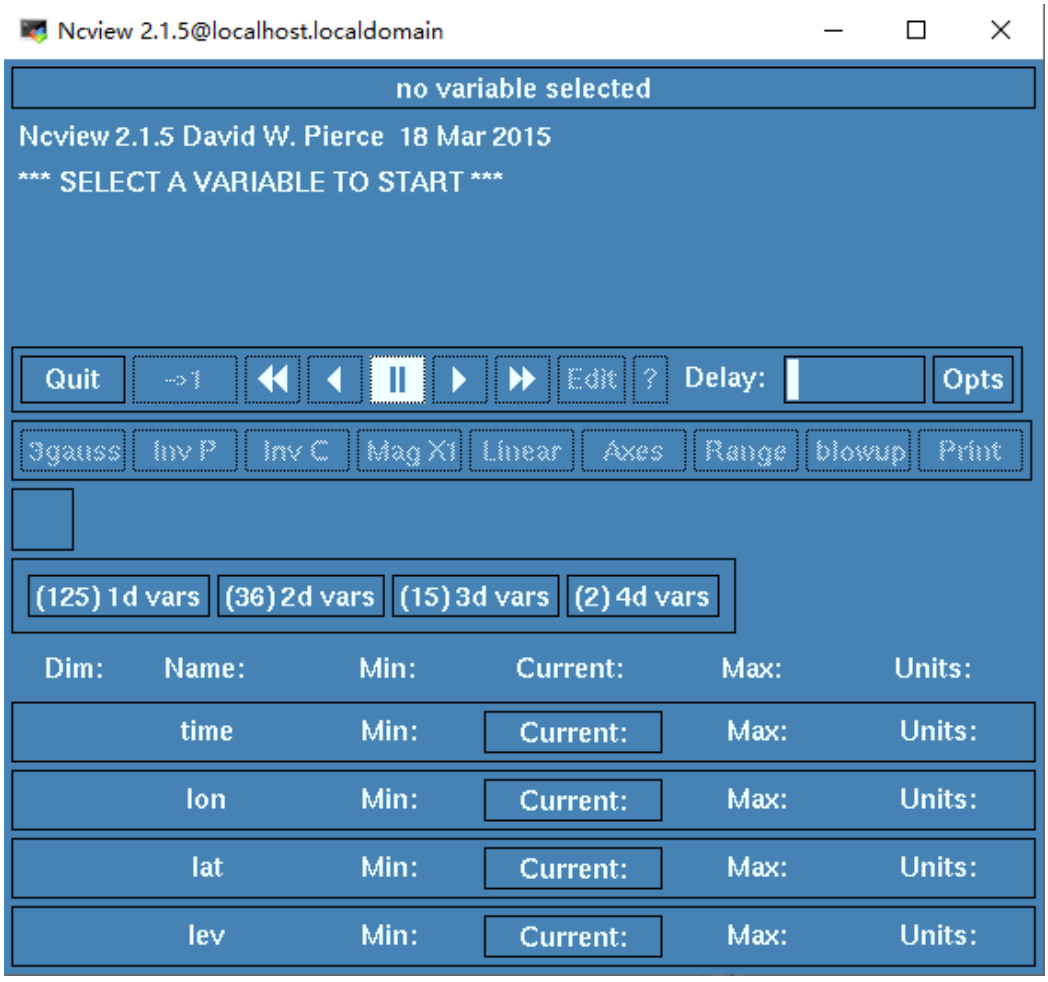
5.25.7 Running and Verification

Procedure

Perform the verification.

```
cd /path/to/ncview/bin
./ncview /path/to/netcdf-c-4.7.0/ncdump/ref_nccopy3_subset.nc
```

If the following information is displayed, the verification is successful.



5.25.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

5.26 paddle 2.3.1 Porting Guide (Kylin V10)

5.26.1 Introduction

PaddlePaddle, as the first proprietary deep learning platform in China, has been officially open-sourced to professional communities since 2016. It is an industrial platform with advanced technologies and rich features that cover core deep learning frameworks, basic model libraries, end-to-end development kits, tools & components as well as service platforms.

For more information, visit the **PaddlePaddle page at GitHub**.

Programming language: C++/Python

Brief description: An industrial platform with advanced technologies and rich features that cover core deep learning frameworks, basic model libraries, end-to-end development kits, tools & components as well as service platforms.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.3.1

5.26.2 Environment Requirements

Hardware Requirements

Table 5-130 lists the hardware requirements.

Table 5-130 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-131 lists the OS requirements.

Table 5-131 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 5-132 lists the software requirements.

Table 5-132 Software requirements

Item	Version	Download URL
paddle	2.3.1	**https://github.com/PaddlePaddle/Paddle/archive/refs/tags/v2.3.1.tar.gz**
BiSheng Compiler	2.1.0	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz**

5.26.3 Paths for Software Porting

This chapter lists the software installation paths involved in the paddle software porting.

Table 5-133 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/paddle	Installation path of paddle.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.26.4 Configuring the Compilation Environment

 Copyright © Huawei Technologies Co., Ltd.

5.26.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-134 Configuration process

No.	Configuration Item	Remarks
1	Installing BiSheng Compiler 2.1.0	See "Installing and Using BiSheng Compiler" in **User Guide (BiSheng Compiler)**.
2	Installing dependencies	See **5.26.4.2 Installing Dependencies**.

5.26.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies.

```
yum install python3 python3-pip openblas-devel lapack-devel git
pip3 install protobuf numpy wheel patchelf pyyaml jinja2 --trusted-host mirrors.tools.huawei.com -i http://mirrors.tools.huawei.com/pypi/simple
```

----End

5.26.5 Obtaining the Source Code

Procedure

Step 1 (Optional) Download the paddle installation package.

https://github.com/PaddlePaddle/Paddle/archive/refs/tags/v2.3.1.tar.gz

It is a better practice to obtain the source code using **git clone**.

Step 2 Use an SFTP tool to upload the installation package to the */path/to/paddle* directory on the server.

----End

5.26.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/paddle
```

Step 3 Decompress the installation package.

```
git clone https://github.com/PaddlePaddle/Paddle.git
cd Paddle
git checkout develop
ulimit -n 4096
tar xvf v2.3.1.tar.gz
cd Paddle-2.3.1
```

Step 4 Modify the configuration file.

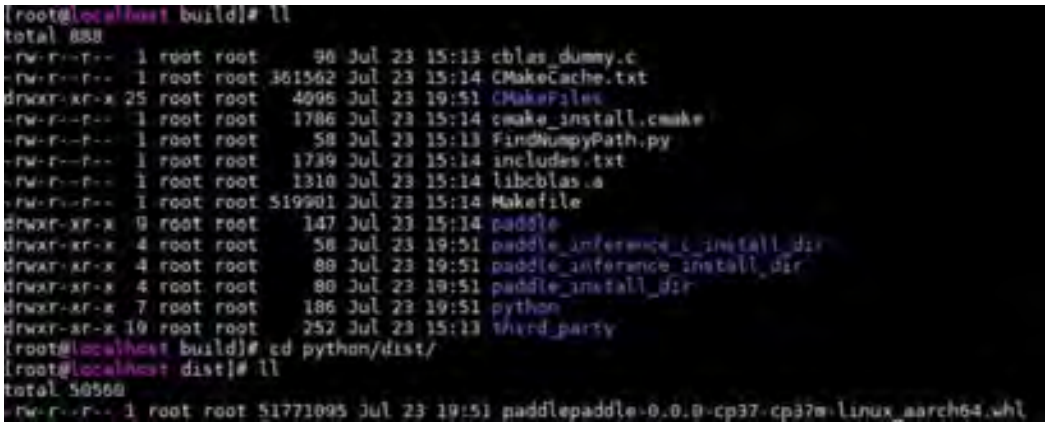
```
sed -i "144c\      #-Werror " cmake/flags.cmake
sed -i "227c\  set(GPU_COMMON_FLAGS -Wall -Wextra ${GPU_COMMON_FLAGS}) " cmake/
flags.cmake
```

Step 5 Perform the compilation.

```
mkdir build
cd build
git init
cmake .. -DWITH_GPU=OFF -DWITH_TESTING=OFF -DCMAKE_BUILD_TYPE=Release -DPY_VERSION=3 -
DWITH_MKL=OFF -DWITH_AVX=OFF -DON_INFER=ON -DWITH_XBAYAK=OFF -DWITH_ARM=ON
git config --global http.postBuffer 524288000
make TARGET=ARMV8 -j$(nproc)
```

Note: In this step, you need to **git clone** the dependencies and then they are automatically installed. Ensure that your computer is connected to the Internet. This step may take a long time.

After the compilation is complete, a **whl** file is generated in **python/dist** of the current directory.



Step 6 Perform the installation.

```
cd python/dist
pip3 install -U *.whl
```

----End

5.26.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify paddle.

```
python3
```

Access the Python 3 interpreter.

Input the following command:

```
import paddle
```

Then input the following command:

```
paddle.utils.run_check()
```

If the following information is displayed, the verification is successful.

----End

5.26.8 Change History

Date	Description
2023-04-28	This issue is the first official release.

5.27 ploticus 242 Porting Guide (Kylin V10)

5.27.1 Introduction

ploticus is a free GPL software utility that can produce various types of plots and graphs.

For more information, visit the **ploticus website**.

Programming language: C

Brief description: GPL software utility.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

242

5.27.2 Environment Requirements

Hardware Requirements

Table 5-135 lists the hardware requirements.

Table 5-135 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-136 lists the OS requirements.

Table 5-136 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

Software Requirements

Table 5-137 lists the software requirements.

Table 5-137 Software requirements

Item	Version	Download URL
ploticus	242	https://github.com/timflutre/ploticus/archive/refs/heads/master.zip
BiSheng Compiler	2.1.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-2.1.0-aarch64-linux.tar.gz

5.27.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ploticus software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-138 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/ploticus	Installation path of ploticus.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.27.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-139 Configuration process

No.	Configuration Item	Remarks
1	Installing BiSheng Compiler 2.1.0	See "Installing and Using BiSheng Compiler" in **User Guide (BiSheng Compiler)**.

5.27.5 Obtaining the Source Code

Procedure

Step 1 Download the ploticus installation package.

https://sourceforge.net/projects/ploticus/files/ploticus/2.42/ploticus242_src.tar.gz

Step 2 Use an SFTP tool to upload the installation package to the */path/to/ploticus* directory on the server.

----End

5.27.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/ploticus
```

Step 3 Decompress the installation package.

```
tar xvf ploticus242_src.tar.gz
cd ploticus242/src
```

Step 4 Modify **Makefile**.

```
sed -i "10c\CC = clang" Makefile
sed -i "143c\INSTALLBIN = /path/to/ploticus242-install" Makefile
```

Step 5 Create an installation directory.

```
mkdir -p /path/to/ploticus242-install
```

Step 6 Perform the compilation and installation.

```
make
make install
```

----End

5.27.7 Running and Verification

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Verify ploticus.

```
cd /path/to/ploticus-install
./pl
```

If the following information is displayed, the verification is successful.

```
[root@localhost ploticus242-install]# ./pl
usage: pl scriptfile [options] ...or...  pl -prefab prefabname [options]
ploticus 2.42-May2013 (unix).  This build can produce: PS EPS SVG SVGZ PNG
Copyright 1998-2009 Steve Grubb, http://ploticus.sourceforge.net
```

----End

5.27.8 Change History

Date	Description
2023-04-28	This issue is the first official release.

5.28 Rosetta Porting Guide (Kylin V10)

5.28.1 Introduction

Rosetta is a privacy-preserving framework based on TensorFlow. It integrates with mainstream privacy-preserving computation technologies, including cryptography, federated learning and trusted execution environment. Rosetta aims to provide privacy-preserving solutions for artificial intelligence without requiring expertise in cryptography, federated learning and trusted execution environment. Rosetta

 Copyright © Huawei Technologies Co., Ltd.

reuses the APIs of TensorFlow and allows transferring traditional TensorFlow codes into a privacy-preserving manner with minimal changes, just with one line of code. The current version integrates the secure multi-party computation protocols for 3 parties. The underlying protocols are SecureNN and Helix. They are secure in the semi-honest model with honest majority.

For more information, visit the **Rosetta page at GitHub**.

Programming language: C++/Python

Brief description: A framework for implementing privacy protection and computing.

Open source license: LGPL v3

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

You are advised to clone data from the GitHub project.

5.28.2 Environment Requirements

Hardware Requirements

Table 5-140 lists the hardware requirements.

Table 5-140 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

2 lists the OS requirements.

Table 5-141 OS requirements

Item	Version	Download URL
Kylin	V10	**https://www.kylinos.cn/scheme/server/1.html**
Kernel	4.19.90-24.4.v2101.ky10.aarch64	Included in the OS image

 Copyright © Huawei Technologies Co., Ltd.

Software Requirements

Table 5-142 lists the software requirements.

Table 5-142 Software requirements

Item	Version	Download URL
Rosetta	Latest version available at GitHub	**https://github.com/LatticeX-Foundation/Rosetta.git**
Python	3.7.6	**https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tar.xz**
JDK 8	8_341	**https://www.oracle.com/java/technologies/downloads**
Bazel	0.25.0	**https://github.com/bazelbuild/bazel/releases/download/0.25.0/bazel-0.25.0-dist.zip**
TensorFlow	1.14.0	**https://github.com/tensorflow/tensorflow/archive/refs/tags/v1.14.0.tar.gz**

5.28.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Rosetta software porting.

Table 5-143 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/Rosetta	Installation path of Rosetta.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/Python3.7.6	Installation path of Python 3.7.6.	
3	/path/to/JDK8	Installation path of JDK 8.	
4	/path/to/bazel	Installation path of Bazel.	
5	/path/to/tensorflow-1.14.0	Installation path of TensorFlow.	

 Copyright © Huawei Technologies Co., Ltd.

5.28.4 Configuring the Compilation Environment

5.28.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-144 Configuration process

No.	Configuration Item	Remarks
1	Installing Python 3.7.6	See **5.28.4.2 Installing Python 3.7.6**.
2	Installing JDK 8	See **5.28.4.3 Installing JDK 8**.
3	Installing Bazel 0.25.0	See **5.28.4.4 Installing Bazel 0.25.0**.
4	Installing TensorFlow 1.14.0	See **5.28.4.5 Installing TensorFlow 1.14.0**.

5.28.4.2 Installing Python 3.7.6

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd path/to/python3.7.6
```

Step 3 Decompress the package.

```
tar -xvf Python-3.7.6.tar.xz
```

Step 4 Go to the **Python-3.7.6** directory.

```
cd Python-3.7.6
```

Step 5 Perform the compilation and installation.

```
mkdir /path/to/Python3.7.6
./configure –prefix=/path/to/Python3.7.6
make -j && make install
```

Step 6 Create a soft link that points to Python.

```
cd /usr/bin
```

Run the **python -V** command. If another version is displayed in the command output, run the following command to delete Python:

```
rm python
```

 Copyright © Huawei Technologies Co., Ltd.

If no Python version is displayed in the command output, run the following command to create a soft link:

```
ln -s /path/to/Python3.7.6/bin/python3 ./python
```

Step 7 Check the Python version.

```
python -V
```

----End

5.28.4.3 Installing JDK 8

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory.

```
cd /path/to/JDK8
```

Step 3 Decompress the package.

```
tar -zxvf jdk-8u341-linux-aarch64.tar.gz
```

Step 4 Set the environment variables.

```
vim ~/.bashrc
```

Add the following content to the end of the **.bashrc** file:

```
#set java environment
JAVA_HOME=/path/to/JDK8/jdk1.8.0_341/
JRE_HOME=/path/to/JDK8/jdk1.8.0_341/jre
CLASS_PATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
export JAVA_HOME JRE_HOME CLASS_PATH PATH
```

Make the environment variables take effect.

```
source ~/.bashrc
```

Step 5 Perform the verification.

```
java -version
```

----End

5.28.4.4 Installing Bazel 0.25.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/bazel
```

Step 3 Decompress the package.

```
unzip -d bazel-0.25.0 bazel-0.25.0-dist.zip
```

Step 4 Perform the compilation and installation.

```
cd bazel-0.25.0
EXTRA_BAZEL_ARGS='--host_javabase=@local_jdk//:jdk' ./compile.sh
```

Step 5 Set the environment variables.

```
export PATH=/path/to/bazel/bazel-0.25.0/output:$PATH
```

Step 6 Perform the verification.

```
bazel version
```

----End

5.28.4.5 Installing TensorFlow 1.14.0

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the package.

```
tar -zxvf v1.14.0.tar.gz
```

Step 3 Build **.whl** files for TensorFlow 1.14.0 and Python 3.7.

```
cd tensorflow-1.14.0
./configure
bazel build -j 4 //tensorflow/tools/pip_package:build_pip_package (It takes about 5 hours. Please wait.)
./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
```

Step 4 Install the TensorFlow **.whl** file using pip.

```
pip3 install /tmp/tensorflow_pkg/*.whl --user
```

Step 5 Perform the verification.

```
pip3 list
```

----End

5.28.5 Obtaining the Source Code

Procedure

Download Rosetta.

```
git clone --recurse https://github.com/LatticeX-Foundation/Rosetta.git
```

5.28.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the installation directory of the main program.

```
cd /path/to/Rosetta
```

Step 3 Perform the compilation and installation.

```
./rosetta.sh compile --enable-protocol-mpc-securenn
./rosetta.sh install
```

----End

 Copyright © Huawei Technologies Co., Ltd.

5.28.7 Change History

Date	Description
2023-04-28	This issue is the first official release.

5.29 SimpleITK Master Porting Guide (openEuler 20.03)

5.29.1 Introduction

SimpleITK is an image analysis toolkit, providing a simplified programming interface to the algorithms and data structures of the Insight Segmentation and Registration Toolkit. It supports image operations on 2D, 3D and 4D multi-channel images. The toolkit is written in C++, and also provides programming interfaces for other languages including Python, R, and Java.

For more information, visit the **SimpleITK official website**.

Programming language: C++

Brief description: An image analysis toolkit.

Open source license: Apache 2.0

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

Master

5.29.2 Environment Requirements

Hardware Requirements

Table 5-145 lists the hardware requirements.

Table 5-145 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-146 lists the OS requirements.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-146 OS requirements

Item	Version	Download URL
openEuler	20.03 LTS SP3	**https://repo.openeuler.org/openEuler-20.03-LTS-SP3/ISO/aarch64/openEuler-20.03-LTS-SP3-everything-aarch64-dvd.iso**
Kernel	kernel-4.19.90-2212.4.0.0183.oe1.aarch64	Included in the OS image

Software Requirements

Table 5-147 lists the software requirements.

Table 5-147 Software requirements

Item	Version	Download URL
SimpleITK	master	**https://github.com/SimpleITK/SimpleITK.git**
GCC for openEuler	10.3.1	**https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2022.12-aarch64-linux.tar.gz**

5.29.3 Paths for Software Porting

This chapter lists the software installation paths involved in the SimpleITK software porting.

Table 5-148 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/SimpleITK	Installation path of SimpleITK.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.
2	/path/to/R	Installation path of the R language.	
3	/path/to/MINICONDA	Installation path of Miniconda.	

 Copyright © Huawei Technologies Co., Ltd.

5.29.4 Configuring the Compilation Environment

5.29.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-149 Configuration process

No.	Configuration Item	Remarks
1	Installing GCC for openEuler	See "Installing and Using GCC for openEuler" in **User Guide (GCC for openEuler)**.
2	Installing dependencies	See **5.29.4.2 Installing Dependencies**.
3	Installing the R language	See **5.29.4.3 Installing the R Language**.
4	Installing Miniconda	See **5.29.4.4 Installing Miniconda**.

5.29.4.2 Installing Dependencies

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the dependencies using the **yum** command.

```
yum install vim iputils net-tools openssh openssh-clients openssh-server glibc glibc-common glibc-devel glibc-help libstdc++ libstdc++-devel libatomic bison wget libgcc gcc_secure readline readline-devel libXt libXt-devel bzip2 bzip2-devel xz xz-libs xz-devel pcre2 pcre2-devel curl libcurl libcurl-devel git lua lua-devel java-11-openjdk java-11-openjdk-devel tcl tcl-devel ruby ruby-devel make cmake
```

----End

5.29.4.3 Installing the R Language

Procedure

Step 1 Download the R language installation package and decompress it.

```
cd /path/to/R
wget https://cran.r-project.org/src/base/R-4/R-4.1.2.tar.gz --no-check-certificate
mkdir src_R_4.1.2
tar zxvf R-4.1.2.tar.gz -C ./src_R_4.1.2 --strip-components 1
cd src_R_4.1.2
```

Step 2 Compile and install R.

 Copyright © Huawei Technologies Co., Ltd.

```
./configure --prefix=/path/to/R CC=`which gcc` CXX=`which g++` FC=`which gfortran`
make -j
make install
```

Step 3 Add R to the environment variables.

```
export PATH=/path/to/R/bin:$PATH
```

Step 4 Perform the verification.

```
R --version
```

----End

5.29.4.4 Installing Miniconda

Procedure

Step 1 Download the installation package.

```
cd /path/to/MINICONDA
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-aarch64.sh --no-check-certificate
```

Step 2 Install Miniconda.

```
bash Miniconda3-latest-Linux-aarch64.sh
```

(You only need to enter **yes** to confirm throughout the installation process.)

Step 3 Deactivate the base environment by default.

```
source ~/miniconda3/bin/activate
conda config --set auto_activate_base False
conda deactivate
```

Step 4 Install the Python virtual environment and activate it.

```
conda create -n py39_SimpleITK python=3.9
conda activate py39_SimpleITK
```

Step 5 Install NumPy.

```
conda install numpy
```

----End

5.29.5 Obtaining the Source Code

Procedure

Obtain the source code.

```
cd /path/to/SimpleITK
git config --global http.sslVerify false
git clone https://github.com/SimpleITK/SimpleITK.git
```

5.29.6 Compilation and Installation

Procedure

Step 1 Go to the following directory.

```
cd /path/to/SimpleITK/SimpleITK
```

Step 2 Create an installation directory and perform the compilation and installation.

```
mkdir build
cd build
```

 Copyright © Huawei Technologies Co., Ltd.

```
CC=`which gcc` CXX=`which g++` FC=`which gfortran` cmake ../SuperBuild
make -j
```

----End

5.29.7 Running and Verification

Procedure

Step 1 Switch to the directory where SimpleITK test cases are stored.

```
cd /path/to/SimpleITK/SimpleITK/build/SimpleITK-build
```

Step 2 Run the test cases.

```
ctest .
```

NOTE

Some test cases may fail, including **ProcessObject_NumberOfThreads**, **SLICImageFilter.minimum**, and **ImageRegistrationMethodDisplacement1**.

The cause is that these official test cases have bugs inside, but the software will not be affected. For details, visit https://github.com/SimpleITK/SimpleITK/issues/1571.

Step 3 Using Python as an example, bind the Python interface of SimpleITK.

```
cd /path/to/SimpleITK/SimpleITK/build/SimpleITK-build/Wrapping/Python
python setup.py install
```

NOTE

Run **pip list**. You can see that SimpleITK has been installed in the current Python interpreter.

For details about how to bind other languages, visit https://simpleitk.readthedocs.io/en/v2.1.0/building.html#installation-from-build-tree.

----End

5.29.8 Change History

Date	Description
2023-04-27	This issue is the first official release.

5.30 Singularity 3.9.9 Porting Guide (Kylin V10)

5.30.1 Introduction

Singularity is an open source container platform designed to be simple, fast, and secure. Many container platforms are available, but Singularity is designed for ease-of-use on shared systems and in high performance computing (HPC) environments.

For more information, visit the **Singularity page at GitHub**.

Programming language: C/Go

Brief description: Container.

 Copyright © Huawei Technologies Co., Ltd.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

3.9.9

5.30.2 Environment Requirements

Hardware Requirements

Table 5-150 lists the hardware requirements.

Table 5-150 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-151 lists the OS requirements.

Table 5-151 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 5-152 lists the software requirements.

Table 5-152 Software requirements

Item	Version	Download URL
Singularity	3.9.9	https://github.com/sylabs/singularity/releases/download/v3.9.9/singularity-ce-3.9.9.tar.gz

 Copyright © Huawei Technologies Co., Ltd.

Item	Version	Download URL
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

5.30.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Singularity software porting.

Table 5-153 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/singularity	Installation path of Singularity.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.30.4 Configuring the Compilation Environment

5.30.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 5-154 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.
2	Installing Go	See **5.30.4.2 Installing Go**.

 Copyright © Huawei Technologies Co., Ltd.

5.30.4.2 Installing Go

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download Go.

```
cd /path/to
wget https://golang.google.cn/dl/go1.19.2.linux-arm64.tar.gz
tar xvf go1.19.2.linux-arm64.tar.gz
```

Step 3 Set the Go environment variables.

```
cd go
export PATH=`pwd`:$PATH
```

----End

5.30.5 Obtaining the Source Code

Procedure

Step 1 Download the Singularity installation package.

https://github.com/sylabs/singularity/releases/download/v3.9.9/singularity-ce-3.9.9.tar.gz

Step 2 Use an SFTP tool to upload the Singularity installation package to the */path/to/Singularity* directory on the server.

----End

5.30.6 Compilation and Installation

Procedure

Step 1 Decompress the source package.

```
cd /path/to/singularity
tar -xvf singularity-ce-3.9.9.tar.gz
cd singularity-ce-3.9.9
```

Step 2 Create **Makefile**.

```
./mconfig -c clang -x clang++
```

Step 3 Perform the compilation and installation.

```
make -C builddir
make -C builddir install
```

----End

5.30.7 Running and Verification

Procedure

Perform the verification.

```
singularity
```

If the following information is displayed, the verification is successful.

 Copyright © Huawei Technologies Co., Ltd.

5.30.8 Change History

Date	Description
2023-04-24	This issue is the first official release.

5.31 szip 2.1.1 Porting Guide (Kylin V10)

5.31.1 Introduction

szip is a common open-source compression library.

For more information, visit the **szip website**.

Programming language: C

Brief description: Open-source compression library.

Open source license: custom open source license

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

2.1.1

5.31.2 Environment Requirements

Hardware Requirements

Table 5-155 lists the hardware requirements.

Table 5-155 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

OS Requirements

Table 5-156 lists the OS requirements.

Table 5-156 OS requirements

Item	Version	Download URL
Kylin	V10	https://www.kylinos.cn/scheme/server/1.html
Kernel	4.19	Included in the OS image

Software Requirements

Table 5-157 lists the software requirements.

Table 5-157 Software requirements

Item	Version	Download URL
szip	2.1.1	https://distfiles.macports.org/szip/szip-2.1.1.tar.gz
BiSheng Compiler	2.5.0	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiSheng-compiler-2.5.0-aarch64-linux.tar.gz

5.31.3 Paths for Software Porting

This chapter lists the software installation paths involved in the szip software porting.

 Copyright © Huawei Technologies Co., Ltd.

Table 5-158 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/path/to/szip	Installation path of szip.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.31.4 Configuring the Compilation Environment

5.31.4.1 Configuration Description

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

The server must be connected to the Internet.

Configuration Process

Table 5-159 Configuration process

No.	Configuration Item	Remarks
1	Installing the BiSheng Compiler	See **Installing the BiSheng Compiler**.

5.31.5 Obtaining the Source Code

Procedure

Step 1 Download the szip installation package.

https://distfiles.macports.org/szip/szip-2.1.1.tar.gz

Step 2 Use an SFTP tool to upload the szip installation package to the */path/to/szip* directory on the server.

----End

 Copyright © Huawei Technologies Co., Ltd.

5.31.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the source package.

```
cd /path/to/szip
tar -xvf szip-2.1.1.tar.gz
cd szip-2.1.1
```

Step 3 Set the environment variables.

```
export CC=clang CXX=clang++ FC=flang
```

Step 4 Create **Makefile**.

```
./configure --prefix=/path/to/szip
```

Step 5 Perform the compilation and installation.

```
make
make install
```

----End

5.31.7 Running and Verification

Procedure

Check whether a dynamic library is generated.

```
ll /path/to/szip/lib
```

If the following information is displayed, the generation is successful.

```
total 128
-rw-r--r-- 1 root root 67402 Oct 26 16:22 libsz.a
-rwxr-xr-x 1 root root   941 Oct 26 16:22 libsz.la
lrwxrwxrwx 1 root root    14 Oct 26 16:22 libsz.so -> libsz.so.2.0.0
lrwxrwxrwx 1 root root    14 Oct 26 16:22 libsz.so.2 -> libsz.so.2.0.0
-rwxr-xr-x 1 root root 73400 Oct 26 16:22 libsz.so.2.0.0
```

5.31.8 Change History

Date	Description
2023-04-24	This issue is the first official release.

5.32 VTK 7.1.1 Porting Guide (openEuler 20.03)

5.32.1 Introduction

The Visualization Toolkit (VTK) is an open-source software system for image processing, 3D graphics, volume rendering and visualization. VTK includes many advanced algorithms (e.g., surface reconstruction, implicit modeling, decimation) and rendering techniques (e.g., hardware-accelerated volume rendering, LOD

 Copyright © Huawei Technologies Co., Ltd.

control). VTK is used by academicians for teaching and research and by many commercial firms who use VTK to build or extend products.

For more information, visit the **VTK official website**.

Programming language: C++

Brief description: An open-source software system for image processing, 3D graphics, volume rendering and visualization.

Open source license: BSD 3-clause

NOTE

When using open source software, comply with the applicable licenses.

Recommended Software Version

7.1.1

5.32.2 Environment Requirements

Hardware Requirements

Table 5-160 lists the hardware requirements.

Table 5-160 Hardware requirements

Item	Description
CPU	Kunpeng 920 processor

Software Requirements

Table 5-161 lists the software requirements.

Table 5-161 Software requirements

Software	Version	How to Install
openEuler	20.03 LTS SP1	**https://docs.openeuler.org/en/docs/20.03_LTS_SP1/docs/Installation/installation-guideline.html**
GCC for openEuler	9.3	**https://www.hikunpeng.com/document/detail/en/kunpengdevps/compiler/ug-hgcc/kunpenghgcc_06_0005.html**
VTK	7.1.1	-
cmake	3.16.5	-

 Copyright © Huawei Technologies Co., Ltd.

5.32.3 Paths for Software Porting

This chapter lists the software installation paths involved in the VTK software porting.

Table 5-162 Paths for software porting

No.	Software Installation Path	Description	Remarks
1	/usr/local/vtk7.1.1	Installation path of VTK.	The installation paths provided here are only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.

5.32.4 Configuring the Compilation Environment

5.32.4.1 Configuration Description

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 5-163 Configuration process

No.	Configuration Item	Remarks
1	Configuring a yum source	See **5.32.4.2 Configuring the Huawei Yum Source**.
2	Disabling the firewall	See **5.32.4.3 Disabling the Firewall**.
3	Installing GCC for openEuler	See "Installing GCC for openEuler" in **User Guide (GCC for openEuler)**.

5.32.4.2 Configuring the Huawei Yum Source

Procedure

Step 1 Back up the base source file.

```
mv /etc/yum.repos.d/openEuler.repo /etc/yum.repos.d/openEuler.repo.bak
```

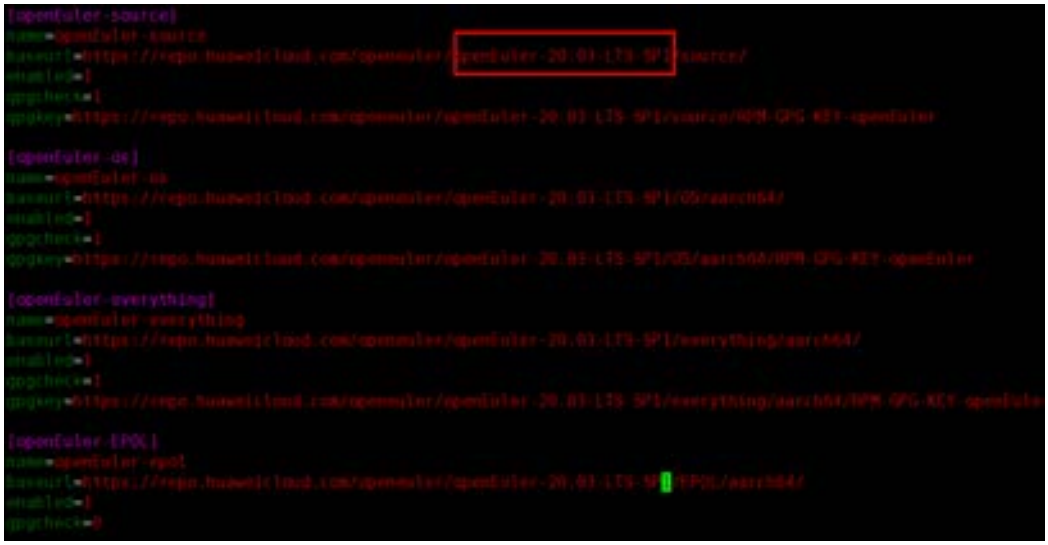
 Copyright © Huawei Technologies Co., Ltd.

Step 2 Download the **openeuler_aarch64.repo** file.

```
cd /etc/yum.repos.d/
wget -O /etc/yum.repos.d/openEulerOS.repo https://repo.huaweicloud.com/repository/conf/
openeuler_aarch64.repo
```

Step 3 Modify the repo link and replace **openEuler-20.03-LTS** with **openEuler-20.03-LTS-SP1**.

```
vim openEulerOS.repo
```



Step 4 Clear the original yum source cache.

```
yum clean all
```

Step 5 Refresh the yum source cache.

```
yum makecache
```

----End

5.32.4.3 Disabling the Firewall

Procedure

Step 1 Stop the firewall.

```
systemctl stop firewalld.service
```

Step 2 Disable the firewall.

```
systemctl disable firewalld.service
```

----End

5.32.4.4 Installing GCC for openEuler

Step 1 Download GCC for openEuler to the **/home** directory.

```
wget https://mirror.iscas.ac.cn/kunpeng/archive/compiler/kunpeng_gcc/gcc-9.3.1-2021.03-aarch64-
linux.tar.gz
```

Step 2 Decompress the package.

```
mkdir -p /opt/aarch64/compiler
cp -rf gcc-9.3.1-2021.03-aarch64-linux.tar.gz /opt/aarch64/compiler
cd /opt/aarch64/compiler
tar -xf gcc-9.3.1-2021.03-aarch64-linux.tar.gz
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Set the environment variables and add the following content to the end of the **profile** file.

```
vi /etc/profile
export PATH=/opt/aarch64/compiler/gcc-9.3.1-2021.03-aarch64-linux/bin:$PATH
export INCLUDE=/opt/aarch64/compiler/gcc-9.3.1-2021.03-aarch64-linux/include:$INCLUDE
export LD_LIBRARY_PATH=/opt/aarch64/compiler/gcc-9.3.1-2021.03-aarch64-linux/
lib64:$LD_LIBRARY_PATH
```

Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

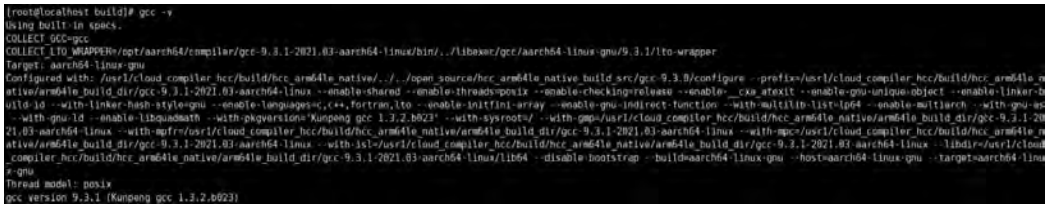
Run the following command for the environment variables to take effect. After reconnecting to the server, you do not need to run the following command again. The system automatically loads the environment configuration.

```
source /etc/profile
```

Step 4 After setting and loading the Kunpeng GCC environment variables, verify the Kunpeng GCC compiler version.

```
gcc –v
```

If the command output contains the Kunpeng GCC version information, the Kunpeng GCC compiler has been successfully installed.



----End

5.32.5 Obtaining the Source Code

Procedure

Step 1 Download the **VTK-7.1.1.tar.gz** package.

https://www.vtk.org/files/release/7.1/VTK-7.1.1.tar.gz

Step 2 Use an SFTP tool to upload the VTK source package to the **/home** directory on the server.

----End

5.32.6 Compilation and Installation

Procedure

Step 1 Decompress the source package and go to the source code directory.

```
tar -zxvf VTK-7.1.1.tar.gz
cd VTK-7.1.1
```

Step 2 Set the compilation information.

```
mkdir build
cd build
cmake .. -D CMAKE_INSTALL_PREFIX=/usr/local/VTK-7.1.1
```

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Perform the compilation and installation.

```
make –j48
make install
```

----End

5.32.7 Running and Verification

Procedure

Step 1 Download **VTKExamples** and add it to the test directory.

```
git clone https://github.com/lorensen/VTKExamples.git
cd VTKExamples
```

Step 2 Execute the test.

```
cd build
cmake .. -DVTK_DIR:PATH=/usr/local/VTK-7.1.1 -DBUILD_TESTING:BOOL=ON
make
ctest
```

If the following information is displayed, the verification is successful.

```
-- VTKWikiExamples: PineRootConnectivity requires git lfs to be installed and will not be built
-- VTKWikiExamples: PineRootConnectivityA requires git lfs to be installed and will not be built
-- VTKWikiExamples: PineRootDecimation requires git lfs to be installed and will not be built
-- VTKWikiExamples: DecimateHawaii requires git lfs to be installed and will not be built
-- VTKWikiExamples: Not building Qt examples
-- VTKWikiExamples: git lfs is not installed. Some tests will not be created.
-- Configuring done
-- Generating done
-- Build files have been written to: /data/VTK/VTKExamples/build
```

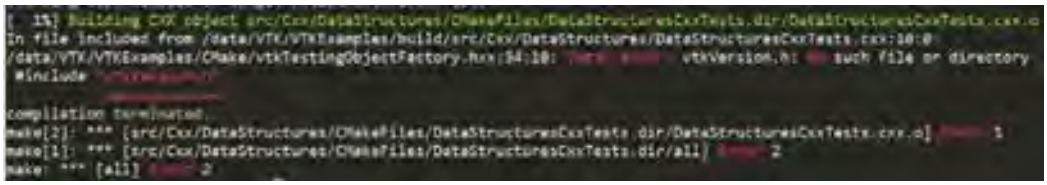
----End

5.32.8 Troubleshooting

5.32.8.1 Test Case Compilation Error: vtkVersion.h: No such file or directory

Symptom

The **vtkVersion.h: No such file or directory** error is reported when compiling test cases.



Root Cause

Some of the test cases are not contained in VTK 7.1.1.

Solution

Modify **VTKExamples/CMakeLists.txt** to mask the following modules:

```
add_subdirectory(src/Cxx/Animation)
add_subdirectory(src/Cxx/CompositeData)
if(VTK_USE_MYSQL)
  add_subdirectory(src/Cxx/Databases)
endif()
# add_subdirectory(src/Cxx/DataStructures)
if(${VTK_VERSION} VERSION_GREATER "6.10")
  add_subdirectory(src/Cxx/Developers)
endif()
#add_subdirectory(src/Cxx/Filtering)
#add_subdirectory(src/Cxx/GeometricObjects)
if(VTK_USE_INFOVIS)
  if(VTK_USE_BOOST)
    add_subdirectory(src/Cxx/Graphs)
  endif()
endif()
add_subdirectory(src/Cxx/Annotation)
#add_subdirectory(src/Cxx/ImageData)
add_subdirectory(src/Cxx/Geovis)
#add_subdirectory(src/Cxx/HyperTreeGrid)
add_subdirectory(src/Cxx/ImageProcessing)
#add_subdirectory(src/Cxx/Images)
#add_subdirectory(src/Cxx/ImplicitFunctions)
#add_subdirectory(src/Cxx/InfoVis)
#add_subdirectory(src/Cxx/Interaction)
#add_subdirectory(src/Cxx/IO)
add_subdirectory(src/Cxx/Lighting)
#add_subdirectory(src/Cxx/Math)
add_subdirectory(src/Cxx/Medical)
#add_subdirectory(src/Cxx/Meshes)
add_subdirectory(src/Cxx/Modelling)
if(VTK_USE_PARALLEL)
  add_subdirectory(src/Cxx/Parallel)
endif()
```

 Copyright © Huawei Technologies Co., Ltd.

```
#add_subdirectory(src/Cxx/Picking)
#add_subdirectory(src/Cxx/Plotting)
#add_subdirectory(src/Cxx/PolyData)
#add_subdirectory(src/Cxx/Points)
add_subdirectory(src/Cxx/RectilinearGrid)
#add_subdirectory(src/Cxx/Remote)
#add_subdirectory(src/Cxx/Rendering)
#add_subdirectory(src/Cxx/RenderMan)
add_subdirectory(src/Cxx/SimpleOperations)
#add_subdirectory(src/Cxx/Shaders)
#add_subdirectory(src/Cxx/StructuredGrid)
add_subdirectory(src/Cxx/StructuredPoints)
#add_subdirectory(src/Cxx/Texture)
#add_subdirectory(src/Cxx/UnstructuredGrid)
#add_subdirectory(src/Cxx/Utilities)
add_subdirectory(src/Cxx/Video)
#if(VTK_USE_VIEWS)
#  add_subdirectory(src/Cxx/Views)
#endif()
#add_subdirectory(src/Cxx/Visualization)
#add_subdirectory(src/Cxx/VisualizationAlgorithms)
#add_subdirectory(src/Cxx/VolumeRendering)
add_subdirectory(src/Cxx/Widgets)
```

5.32.9 Change History

Table 5-164

Date	Description
2021-08-21	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.

6 Change History

Date	Description
2023-06-07	This issue is the twenty-five official release. Added 58 porting guides: ● Meteorology (4) – grads 2.2.1 Porting Guide (CentOS 8.2) – wgrib 1.8.2 Porting Guide (Kylin V10) ● Education and research (9) – dugksFoam 1.0 Porting Guide (Kylin V10) – duns 4.7.1 Porting Guide (Kylin V10) ● Life sciences (34) – art-nextgen-simulation-tools 20160605 Porting Guide (Kylin V10) – bioconductor-appreci8r 1.14.0 Porting Guide (Kylin V10) ● Other (11) – crystal 1.7.2 Porting Guide (Kylin V10) – grib_api 1.21.0 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2023-03-02	This issue is the twenty-fourth official release. Added 51 porting guides: ● Meteorology (1) – gadget2 2.3.5 Porting Guide (Kylin V10) ● Manufacturing (3) – Kokkos 3.5.00 Porting Guide (Kylin V10) – Pencil Code 2021.02.20 Porting Guide (Kylin V10) ● Education and research (8) – Phono3py 2.3.2 Porting Guide (Kylin V10) – QDK 2.3.12 Porting Guide (Kylin V10) ● Life sciences (27) – MrBayes 3.2.7a Porting Guide (Kylin V10) – MSMC 1.1.0 Porting Guide (Kylin V10) ● Other (12) – Keras 2.9.0 Porting Guide (Kylin V10) – Lasagne 0.1 Porting Guide (Kylin V10)
2022-11-18	This issue is the twenty-third official release. Added 40 porting guides: Life sciences (40) ● DBG2OLC Porting Guide (openEuler 20.03) ● MAKER 2.31.11 Porting Guide (openEuler 20.03) ● Platanus 1.2.4 Porting Guide (openEuler 20.03)
2022-09-26	This issue is the twenty-second official release. Added 8 porting guides: ● Education and research (3) – ELK 8.4.30 Porting Guide (Kylin V10) – MOPAC 22.0.4 Porting Guide (Kylin V10) – QMCPACK 3.14.0 Porting Guide (Kylin V10) ● Other (5) – LIBPNG 1.6.37 Porting Guide (Kylin V10) – NCO 5.1.0 Porting Guide (Kylin V10) – PETSc 3.16 Porting Guide (Kylin V10)

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2022-06-16	This issue is the twenty-first official release. Added 39 porting guides: ● Meteorology (20) – CAMx 7.10 Porting Guide (openEuler 21.03) – CDO 1.9.8 Porting Guide (openEuler 20.03) – CESM 2.1.1 Porting Guide (openEuler 21.03) ● Manufacturing (2) – Code_Aster 14.6.0 Porting Guide (openEuler 21.03) – OpenFOAM 1906 Porting Guide (openEuler 21.03) ● Government HPC (15) – CP2K 7.1 Porting Guide (openEuler 21.03) – DL_POLY 1.10 Porting Guide (openEuler 21.03) – GROMACS 2019.3 Porting Guide (openEuler 21.03) ● Other (2) – BWA 0.7.17 Porting Guide (openEuler 21.03) – miniGhost 1.0 Porting Guide (openEuler 21.03)
2022-05-05	This issue is the twentieth official release. Added 47 porting guides: ● Meteorology (2) – ARPS 5.3.4 Porting Guide (CentOS 7.6) – MM5 V3 Porting Guide (CentOS 7.6) ● Manufacturing (2) – OpenLB 1.4-0 Porting Guide (GCC for openEuler) – OpenLB 1.4-0 Porting Guide (BiSheng Compiler) ● Government HPC (4) – LAMMPS 10 Feb 2021 Porting Guide (CentOS 8.2) – QE 6.8 Porting Guide (CentOS 8.2) – GROMACS 2019.3 Porting Guide (CentOS 8.2) ● Other (39) – ABSOLUTE 1.0.6 Porting Guide (CentOS 7.6) – GDS3D 1.9 Porting Guide (openEuler 20.03) – HOMER 4.11 Porting Guide (CentOS 8.2)

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2022-01-30	This issue is the nineteenth official release. • Modified the content of **4.38.2 Environment Requirements**, **4.38.3 Paths for Software Porting**, **4.38.4 Configuring the Compilation Environment**, and **4.38.6 Compiling and Installing MEGAHIT** in the *megahit 1.2.9 Porting Guide (CentOS 7.6)*. • Modified the content of **3.12.7 Running and Verifying Geant4** in the *Geant4 10.6 Porting Guide (CentOS 7.6)*.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2022-01-10	This issue is the eighteenth official release. • For the *Code_Aster 14.6.0 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **2.9.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **2.9.4 Configuring the Compilation Environment**. • For the *ESPResSo 4.1.4 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.15.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.15.4 Configuring the Compilation Environment**. • For the *MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)*, modified the description of setting up the self-developed MPI environment in **3.16.4 Configuring the Compilation Environment**. • For the *MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.17.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.17.4 Configuring the Compilation Environment**. • For the *Octopus 10.3 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.18.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.18.4 Configuring the Compilation Environment**. • For the *Psi4 1.3.2 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.19.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.19.4 Configuring the Compilation Environment**. • For the *SIESTA 4.0.2 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.20.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.20.4 Configuring the Compilation Environment**. • For the *Wannier90 3.1.0 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.21.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.21.4 Configuring the Compilation Environment**. • For the *WanT 2.6.1 Porting Guide (CentOS 7.6)*, modified the description of Hyper MPI installation paths in **3.22.3 Paths for Software Porting** and the description of setting up the self-developed MPI environment in **3.22.4 Configuring the Compilation Environment**.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2021-11-30	This issue is the seventeenth official release. Changed "Kunpeng GCC" to "GCC for openEuler" in the *CPMD 4.1 Porting Guide (CentOS 7.6)*, *ESPResSo 4.1.4 Porting Guide (CentOS 7.6)*, *MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)*, *MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6)*, *Octopus 10.3 Porting Guide (CentOS 7.6)*, *Psi4 1.3.2 Porting Guide (CentOS 7.6)*, *SIESTA 4.0.2 Porting Guide (CentOS 7.6)*, *Wannier90 3.1.0 Porting Guide (CentOS 7.6)*, *WanT 2.6.1 Porting Guide (CentOS 7.6)*, and *Code_Aster 14.6.0 Porting Guide (CentOS 7.6)*.
2021-09-15	This issue is the sixteenth official release. • Added a hyperlink to the *HPC Solution Basic Environment Setup Guide* in **3.9.3 Paths for Software Porting** of the *CP2K 7.1 Porting Guide (CentOS 7.6)*. • Modified the address for downloading test cases in **Software Requirements** of the *LAMMPS Jun 5 2019 Porting Guide (CentOS 7.6)*. • Modified the content of **Step 9** in the *NAMD 2.13 Porting Guide (CentOS 7.6)*.
2021-08-20	This issue is the fifteenth official release. Added the description of the host name **node1** and the process number **96** to **Step 7** of the *CESM 2.1.1 Porting Guide (CentOS 7.6)*.
2021-07-23	This issue is the fourteenth official release. • Modified the commands of **Step 2** in the *NEMO V3.6 Porting Guide (CentOS 7.6)*. • Deleted the **-lnetcdff** parameter from **Step 4** in the *SMOKE 4.7 Porting Guide (CentOS 7.6)*. • Modified the content of **Step 7** in the *OpenFOAM 1906 Porting Guide (CentOS 7.6)*.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2021-06-25	This issue is the thirteenth official release. Added 10 porting guides: • Manufacturing (1) Code_Aster 14.6.0 Porting Guide (CentOS 7.6) • Government HPC (9) – CPMD 4.1 Porting Guide (CentOS 7.6) – ESPResSo 4.1.4 Porting Guide (CentOS 7.6) – MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6) – MPB 1.11.1 and MEEP 1.17.1 Porting Guide (CentOS 7.6) – Octopus 10.3 Porting Guide (CentOS 7.6) – Psi4 1.3.2 Porting Guide (CentOS 7.6) – SIESTA 4.0.2 Porting Guide (CentOS 7.6) – Wannier90 3.1.0 Porting Guide (CentOS 7.6) – WanT 2.6.1 Porting Guide (CentOS 7.6)
2021-06-09	This issue is the twelfth official release. • Modified the content of **Step 7** in the *OpenFOAM 1906 Porting Guide (CentOS 7.6)*. • Modified the content of **Step 3** in the *Bowtie2 2.4.1 Porting Guide (CentOS 7.6)*. • Modified the content of **Step 4** in the *pblat 2.1 Porting Guide (CentOS 7.6)*.
2021-06-02	This issue is the eleventh official release. • Modified the content of **1.1.2 Environment Requirements** in the *WRF 3.8.1 Porting Guide (CentOS 7.6)*. • Modified the content of **3.9.4.5 Installing Libint** in the *CP2K 7.1 Porting Guide (CentOS 7.6)*.
2021-05-28	This issue is the tenth official release. Recategorized the "DNA sequencing" industry component into the "Other" sector.
2021-05-18	This issue is the ninth official release. Changed "Supercomputing" to "Government HPC."

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2021-04-29	This issue is the eighth official release. • Modified the content of **4.4.2 Environment Requirements**, **4.4.4.5 Installing Eigen**, and **4.4.6 Compiling and Installing Cufflinks** in the *Cufflinks 2.2.1 Porting Guide (CentOS 7.6)*. • Modified the content of **4.45.3 Paths for Software Porting**, **4.45.4.2 Installing Maker**, and **4.45.7 Running and Verifying SNAP** in the *SNAP 2013-11-29 Porting Guide (CentOS 7.6)*. • Modified the content of **1.8.2 Environment Requirements**, **1.8.4.2 Installing HDF5**, and **1.8.6 Compilation and Installation** in the *FVCOM 4.1 Porting Guide (CentOS 7.6)*.
2021-04-13	This issue is the seventh official release. Changed the solution name from "Kunpeng HPC Solution" to "Kunpeng BoostKit for HPC" in the **Progressive Knowledge**.
2020-12-20	This issue is the sixth official release. Added the porting guides of 23 components for DNA sequencing, 6 components for meteorology, and 1 component for supercomputing.
2020-09-20	This issue is the fifth official release. Added the porting guides of 3 components for education and scientific research, 5 components for DNA sequencing, 2 components for meteorology, and 1 component for manufacturing.
2020-08-20	This issue is the fourth official release. Updated the *WRF 3.8.1 Porting Guide (CentOS 7.6)* for meteorology, *Cufflinks 2.2.1 Porting Guide (CentOS 7.6)* for DNA sequencing, and *MiniFE 2.2.0 Porting Guide (CentOS 7.6)* for other industries.
2020-07-20	This issue is the third official release. Added the porting guides of 14 components for DNA sequencing and 1 component for manufacturing.
2020-04-20	This issue is the second official release. Added the porting guides of 14 components for industries such as education and scientific research, DNA sequencing, and meteorology.
2020-03-20	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.