LAMMPS Guide

Last updated 9 February, 2023 • 4 min read

From the LAMMPS README file:

> *LAMMPS is a classical molecular dynamics simulation code designed to run efficiently on parallel computers. It was developed at Sandia National Laboratories, a US Department of Energy facility, with funding from the DOE. It is an open-source code, distributed freely under the terms of the GNU Public License (GPL).*

LAMMPS MPI and package install

LAMMPS has various options for building and installing different settings and packages. Due to LAMMPS having a lot of custom builds and various package options, we recommend users to install LAMMPS locally under their /home directories or team's /shared folder for ease of use and the current needs for the user or team. Reading the documentation pages for LAMMPS, we can compile LAMMPS with CMake or with regular make commands. We will use the GCC compiler in this example:

📘 Step 1: Create directory project and download LAMMPS:

```
# Create a directory for our LAMMPS project.

mkdir -pv ~/software/lammps
cd ~/software/lammps

# Download the latest Lammps Linux tarball file from their dc

Go to: https://www.lammps.org/download.html

# Copy the download link for the LAMMPS stable release and pa
wget pasteDownloadLinkforLammpsTarballHere

#Unpack the tarball and cd into the new lammps directory that
#This example will show the LAMMPS 23Jun2022 release

tar -xvf lammps-stable.tar.gz
cd lammps-23Jun2022

# Directory structure should look like the following:

~/software/lammps/lammps-23Jun2022



```

📘 Step 2: Edit LAMMPS makefile

```
#LAMMPS allows for CMake or regular make builds
#In this example we will show the regular make build and inst
#Going to build the MPI version of Lammps

cd lammps-23Jun2022/src/MAKE

#Make a copy of the Makefile.mpi and rename it to Makefile.or

cp Makefile.mpi Makefile.origmpi

#Copy the Makefile.omp file from the src/MAKE/OPTIONS folder

cp OPTIONS/Makefile.omp Makefile.mpi
vi Makefile.mpi
Press i to go into Insert mode in the VI editor

#Go down to the MPI Library section and update the paths to p

MPI_INC =     -I/gpfs/sharedfs1/admin/hpc2.0/apps/openmpi/4.1.
MPI_PATH =    -L/gpfs/sharedfs1/admin/hpc2.0/apps/openmpi/4.1.
MPI_LIB =     -lmpi -lpthread

#Most software installs do not use custom edited Makefiles ar
#If the build would need fftw3, the FFT library section would
#Now we need to add some Link flags at the top to have LAMMPS
#If the build would need blas or lapack libraries, they would

FC =          mpifort
CC =          mpicc
CXX =         mpicxx
CCFLAGS =     -g -O3 -fopenmp -std=c++11
SHFLAGS =     -fPIC
DEPFLAGS =    -M

LINK =        mpicxx
LINKFLAGS = -g -O3 -fopenmp -std=c++11 -L/gpfs/sharedfs1/admi
LIB = -lstdc++ -lgfortran
SIZE =        size

ARCHIVE =     ar
ARFLAGS =     -rc
SHLIBFLAGS =      -shared -rdynamic

#Once entered, save the Makefile with the following key strok

ESC key (to get out of Insert mode)
:wq!



```

Step 3: Build LAMMPS packages

```
#Go back one directory to the /src folder:

cd ..

#Should show the following path:

~/software/lammps/lammps-23Jun2022/src/

#Load the gcc/11.3.0 and openmpi/4.1.4 modules before buildir

module load gcc/11.3.0 openmpi/4.1.4

# Install needed LAMMPS packages for run with the following c

make yes-packagenamehere

# Here are some example Commands to install the openmp, qeq,

make yes-openmp

make yes-qeq

make yes-meam

make yes-reaxff

# After installing the needed packages, the following commanc

make ps

# Or if a list of installed packages is needed without showir

make pi
```

📘 Step 4: Build LAMMPS

```
# Build the LAMMPS MPI executable, the -j flag tells mpi to b

make -j4 mpi

# LAMMPS should start building the executable with the packag
# Once LAMMPS finishes building under the /src folder, LAMMPS
lmp_mpi

#Go back to the main lammps folder

cd ~/software/lammps/lammps-23Jun2022

#Create a /bin folder to move the lmp_mpi executable into

mkdir bin

#Copy the lmp_mpi exectuable over to this new /bin folder

rsync -a --progress ~/software/lammps/lammps-23Jun2022/src/lm

#Continue to the next step to create a module file for the ne
#If a module is not needed, the path can be specified in a su

~/software/lammps/lammps-23Jun2022/bin/lmp_mpi -i restOfLammp
```

📘 Step 5: Create a Module file for LAMMPS

We can create a module file for LAMMPS that so that we can conveniently load LAMMPS and it's dependencies. The name that you choose for your module file is important as that is what module uses to reference it. We will make our name different by adding the "-mine" suffix to help separate it from the system installed vasp.

```
mkdir -p ~/mod/lammps
cd ~/mod/lammps
vi 23Jun2022-mine
```

```
#%Module1.0

# Throw an error if any of these modules are loaded.
conflict lammps

# Load the particular GCC and openmpi versions we used for th
module load gcc/11.3.0
module load openmpi/4.1.4

# Modify the PATH to use our compiled LAMMPS.  Do not use a t
prepend-path PATH ~/software/lammps/lammps-23Jun2022/bin/
```

If you are interested, in learning about module files you can read `man modulefile`

Finally, make sure that `module` knows to look in your `~/mod` directory for your module files by setting the `MODULEPATH` environmental variable:

```
nano ~/.bashrc  # Add the lines below.
```

```
# My modules
source /etc/profile.d/modules.sh
MODULEPATH=${HOME}/mod:${MODULEPATH}
```

Reload your ~/.bashrc file in your current shell:

```
source ~/.bashrc
# Finally Now we can load and run our VASP module
module load lammps/23Jun2022-mine
which lmp_mpi
lmp_mpi -h
```

To install packages AFTER lammps is installed, go to the LAMMPS installer /src folder:

```
cd ~/software/lammps/lammps-23Jun2022/src
```

Then enter the following depending on which packages that would like to be installed:

6.2. Package details — LAMMPS documentation

Example Commands to install the qeq, meam, and reaxff LAMMPS packages:

```
make yes-qeq

make yes-meam

make yes-reaxff

```

After installing the needed packages, the following command can confirm if the packages installed:

```
make ps
```

Or if a list of installed packages is needed without showing the ones that are not installed, the following command can help:

```
make pi
```

Once installing the needed packages, lammps would need to be rebuilt from the previous build steps to invoke the changes.

```
make mpi
```

Once LAMMPS rebuilds with the new packages, the rebuilt lmp_mpi executable would need to be moved back to the /bin folder

```
#Copy the lmp_mpi exectuable over to this new /bin folder
rsync -a --progress ~/software/lammps/lammps-23Jun2022/src/lm

```

If the lammps/23Jun2022-mine module is currently loaded, it would need to be unloaded and reloaded to have the changes take into effect.

```
module unload lammps

module load lammps/23Jun2022-mine

```

Step 6: Running LAMMPS with MPI

The lmp_mpi MPI build above should automatically look for the openmpi/4.1.4 HPC module.

The openmpi/4.1.4 module would need to be loaded when the lmp_mpi executable is called.

Running LAMMPS with MPI would need specific MPI command line options to bypass issues when ignoring the old openib fabric and running on the new UCX framework.

These settings are automatically loaded when the openmpi/4.1.4 module is loaded in the environment.

This feature of the module allows the options to not be specified in the mpirun or mpiexec commands.

The following example will show how to run the LAMMPS MPI executable by spawning 4 MPI threads and run the specified *lammpsInputFileHere* input file and save the data output to a file called *OutPutFileName.txt*

```
mpirun -np 4 lmp_mpi -in lammpsInputFileHere > OutPutFileName.txt
```

📘 Step 7: The updated openmpi/4.1.4 modules

For extra information, the options that the **openmpi/4.1.4** and **openmpi/4.1.4-ics** modules set will be explained here (and are not needed when running the lmp_mpi command in a submission script):

```
mpirun --mca opal_warn_on_missing_libcuda 0 -mca pml ucx --mca btl
^vader,tcp,openib,uct -x UCX_NET_DEVICES=mlx5_0:1 -np 4
~/software/lammps/lammps-23Jun2022/bin/lmp_mpi -in
restOfLammpsCommandHere
```

Lets break down the mpi command line options:

The `--mca opal_warn_on_missing_libcuda 0` option disables the CUDA warning message

The `-mca pml ucx` tells mpi to run using the UCX framework for the point-to-point message layer

The `--mca btl ^vader,tcp,openib,uct` tells MPI to not run on the vader,tcp,openib, or uct frameworks

The `-x UCX_NET_DEVICES=mlx5_0:1` option tells MPI to target the Infiniband card mlx5_0 and connect to the card on port 1

The `-np 4` flag tells MPI to run on 4 MPI threads

The `~/software/lammps/lammps-23Jun2022/bin/lmp_mpi` path tells MPI to run the lmp_mpi executable under the given path, but if there is a local module loaded, the path to the lmp_mpi executable would not be needed. Instead the lmp_mpi command can replace the full path above.

`-np 4 lmp_mpi`

The `-i restOfLammpsCommandHere` indicates the command line options needed for LAMMPS, in this case the -i refers to the LAMMPS input file for LAMMPS to process.

Or a different way to pass the input file and generate an output file from the given input, can be the following command:

`lmp_mpi -in lammpsInputFileHere > OutPutFileName.txt`

Running LAMMPS

If possible, you should always first run your code on your local machine just to ensure that your code is correct. You can do it on a small dataset and a small configuration (single processor, etc.). This way you would be able to catch any errors not related to the cluster even before submitting your job.

Below we show a step-by-step example of how to run a simple LAMMPS simulation in the cluster. We have used one of the examples bundled with LAMMPS distribution, namely, `flow`.

Copy your code and data to the cluster

We are assuming that you are using the terminal to copy your data. If you are using a GUI client, you should be able to do it in a visual way.

Open a terminal to connect to the cluster and create a directory for the experiment.

```
mkdir lammpstest && cd lammpstest
```

The code from the example can be downloaded using the chunk of code below.

```
## cloning lammps project
git clone https://github.com/lammps/lammps.git
## moving the files from the "flow" example to our working di
mv lammps/examples/flow* .
## deleting the lammps project
rm -rf lammps
```

Now, your `lammpstest` directory should have the following files

```
ls -lt
total 40
-rw-r--r-- 1 netid domain users 5443 Nov 10 09:58 log.27Nov18
-rw-r--r-- 1 netid domain users 5444 Nov 10 09:58 log.27Nov18
-rw-r--r-- 1 netid domain users 5445 Nov 10 09:58 log.27Nov18
-rw-r--r-- 1 netid domain users 5441 Nov 10 09:58 log.27Nov18
-rw-r--r-- 1 netid domain users 1503 Nov 10 09:58 in.flow.poi
-rw-r--r-- 1 netid domain users 1505 Nov 10 09:58 in.flow.cou

```

SLURM script

`SLURM` is the scheduler program for our cluster. In the cluster, we need to create a simple script that would tell `SLURM` how to run your job. For details see the SLURM Guide.

You can either create this script in the terminal using any editor such as `nano`, or you can create it in your local machine and use the `scp` command to copy it into the cluster. We can put this script in the `lammpstest` directory, and it would contain the following lines:

```
$ cd ~/lammpstest
$ cat lammps_job.sh

#!/bin/bash
#SBATCH -N 1
#SBATCH -n 36
#SBATCH --constraint='skylake'
#SBATCH -o lammps_sim_out-%J.txt
#SBATCH -e lammps_sim_out-%J.txt
#SBATCH --mail-type=ALL
#SBATCH --mail-user=user@uconn.edu
module load openmpi/4.1.4
mpiexec lmp_mpi < in.flow.couette

```

This script is telling how many processors we need as well as which files the output (and errors) should be written to. Basically, the lines starting with `#SBATCH` provide the switches for the `sbatch` command, which submits a job to `SLURM`. Note that we have told `SLURM` to email us at every event for this job such as `begin / queued / end / error` etc.

The last line is the command that would be run as the job. It invokes the `lammps` module with the input `~/lammpstest/flow/in.flow.couette`.

If you are using `mpirun` in your submission scripts, it is recommended to use the following command syntax

```
mpirun lmp_mpi -in inputfile
```

The `lammps` documentation site mentioned this note if `mpirun` is being used with the `<` operator:

> *The redirection operator `<` will not always work when running in parallel with `mpirun`; for those systems the -in form is required.*

Submitting your job

Before you submit your job make sure that the `LAMMPS` module is loaded, as described in the first part of this guide. When you are ready, simply do the following:

```
sbatch lammps_job.sh
```

Checking output

When the job is done we would get email notifications. You can also check your job status using the `sjobs` command. We can check on the `lammps` output itself using `tail`:

```
cat lammps_sim_out-JOBID.txt
```

Note that you should replace `JOBID` with the id associated to the submitted job.