HPC Docs: Slurm vs Moab/Torque

hpcc.umd.edu/hpcc/help/slurm-vs-moab.html

Slurm vs Moab/Torque on Deepthought HPC clusters

Intro and Overview: What is a scheduler?

A high performance computing (HPC) cluster (hereafter abbreviated HPCC) like the Deepthought clusters consists of many compute nodes, but at the same time have many users submitting many jobs, often very large jobs. The HPCC needs a mechanism to distribute jobs across the nodes in a reasonable fashion; this is the task of a program called a *scheduler*. This is a complicated tasks: the various jobs can have various requirements (e.g. CPU, memory, diskspace, network transportation, etc.) as well as differing priorities. And because we want to enable large parallel jobs to run, the scheduler needs to be able to reserve nodes for larger jobs (i.e. if an user submits a job requiring 100 nodes, and only 90 nodes are currently free, the scheduler might need to keep other jobs off the 90 free nodes in order that the 100 node job might eventually run). The scheduler must also account for nodes which are down, or have insufficient resources for a particular job, etc. As such, a *resource manager* is also needed (which can either be integrated with the scheduler or run as a separate program). The scheduler will also need to interface with an *accounting system* (which also can be integrated into the scheduler) to handle the charging of allocations for time used on the cluster.

As an user, you interact with the scheduler and/or resource manager whenever you submit a job, or query on the status of your jobs or the cluster as a whole, or seek to manage your jobs.

The original Deepthought HPC cluster at the University of Maryland originally used the Maui scheduler for scheduling jobs, along with the Torque Resource Manager and the Gold Allocation Manager. In 2009, we migrated to the Moab scheduler, still keeping Torque as our resource manager and Gold for allocation management. Moab derived from Maui, and so the user interface was mostly unchanged during this migration.

In the summer of 2014, the Deepthought2 cluster was inaugurated. The scheduler for this cluster is a product called Slurm (see e.g. http://slurm.schedmd.com). This program has many features which can better support the needs of our HPCCs. The original Deepthought HPCC was also migrated to slurm when the Moab license expired. Slurm includes its own resource management and accounting system, so Torque and Gold are no longer used.

The following document is provided to help users make the transition from the Maui/Moab/Torque commands for interacting with the scheduler to the Slurm equivalents, by giving Slurm equivalents of some common Moab/Torque commands and by discussing wrapper scripts which make some Moab/Torque commands work with slurm.

Native slurm equivalents for common Moab/Torque commands

In this section, we show the Slurm equivalents of some common Moab/Torque commands. You are encouraged to use these commands and to *Slurm-ify* your job submission scripts, as this will bring you the most benefit from the new features that the Slurm scheduler supports. In most cases, the Slurm versions are both more flexible and more rational than the Moab/Torque counterparts. But if you are really wish to continue using the old commands , we do have wrapper scripts which will enable most basic Moab/Torque commands to continue to work more or less as before.

The showq command

In a Moab/Torque environment, users will issue the `showq` command to see the state of jobs running on the cluster. The basic equivalent in Slurm is the `squeue` command.

To see a list of *all* jobs on the cluster, using Moab/Torque, one would issue just the `showq` command. The Slurm equivalent would be the `squeue` command:

```
login-1: squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1243530 standard  test2.sh  payerle  R   18:47:23      2 compute-b18-[2-3]
           1244127 standard  slurm.sh    kevin  R    1:15:47      1 compute-b18-4
           1230562 standard  test1.sh  payerle PD       0:00      1 (Resources)
           1244242 standard  test1.sh  payerle PD       0:00      2 (Resources)
           1244095 standard  slurm2.sh   kevin PD       0:00      1 (ReqNodeNotAvail)
```

The main differences in the outputs are that:

- Slurm by default provides the partition (i.e. queue in Moab/Torque terminology), the name of the job, and the nodes the job is running on (or the reason the job is not running if not running).
- Slurm does not provide different sections for different run states. Instead, the run state is listed under the ST (STate column), with the following codes:
  - R for Running
  - PD for PenDing
  - TO for TimedOut
  - PR for PReempted
  - S for Suspended
  - CD for CompleteD
  - CA for CAncelled
  - F for FAILED
  - NF for jobs terminated due to Node Failure
- Jobs in Slurm will continue to be listed for a while after they have completed/failed, with appropriate status code above.
- Slurm by default lists the time used by the job, not remaining time.
- Slurm by default lists the number of *nodes* requested/used by the job, not the number of *processes/tasks/cores*.
- Slurm does not by default list the time remaining for the job or the time the job was submitted.

Note that slurm lists the nodes in an abbreviated form. And that as indicated, this is the **default** output format; Slurm is very user configurable and you can specify exactly what you want outputted with the `-o` option. There is also an `-l` option for more verbose output, and a `-s` option to show job step information.

Often you are only interested in your jobs, or the jobs or a specific user. You can add the option `-u USERNAME` to restrict the output to jobs owned by *USERNAME*. You can also list only those jobs in specific states with the `-t STATELIST` option, where *STATELIST* is a comma separated list of state codes (e.g. R, CD, etc) to display.

The Slurm `squeue` command does not provide the status summary line showing the nodes/processors on cluster and percent usage. The slurm `sinfo` can provide the equivalent information. By default, it does not provide core counts, but the output format is very flexible, and something like `sinfo -o "%10P %10t %10D %20C"` will show a list of partitions, and the node counts by state, along with the number of CPUs Allocated, Idle, Offline, and Total.

Slurm also includes the `sview` which provides a graphical view of the cluster status.

The pbsnodes command

In Moab/Torque, the `pbsnodes` command gives the state of all or specific nodes in the cluster. We also have a `nodeusage` command which provides this information in a condensed, one-line per node format.

Slurm also provides a number of methods to display the state of nodes. The closest equivalent to the `pbsnodes` command is probably `scontrol show node`. You can specify specific nodes, or a list of specific nodes to show information about:

```
login-2>  scontrol show node compute-b17-1
NodeName=compute-b17-1 Arch=x86_64 CoresPerSocket=10
   CPUAlloc=0 CPUErr=0 CPUTot=20 CPULoad=0.00 Features=(null)
   Gres=(null)
   NodeAddr=compute-b17-1 NodeHostName=compute-b17-1
   OS=Linux RealMemory=128000 AllocMem=0 Sockets=2 Boards=1
   State=IDLE ThreadsPerCore=1 TmpDisk=750000 Weight=1
   BootTime=2014-04-30T15:20:33 SlurmdStartTime=2014-05-02T08:37:08
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

In place of `nodeusage`, the Slurm command `sinfo -N` provides a similar summary.

The qstat and checkjob commands

The `qstat` command in Torque provides a one line status of all jobs in the cluster, or for specific jobs if job numbers provided. It also has a `-f` option to display more details in a multiline format. The Moab `checkjob` command also provides detailed information about a specific job.

The Slurm `squeue` command can provide the single line status of jobs. To specify a single job, use the `-j JOBNUMBER` option. It is discussed more above.

For more detailed job information, use the `scontrol show job JOBNUMBER` command, e.g.

```
login-2> scontrol show job 486
JobId=486 Name=test1.sh
   UserId=payerle(34676) GroupId=glue-staff(8675)
   Priority=33 Account=test QOS=normal
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:03:00 TimeMin=N/A
   SubmitTime=2014-05-06T11:20:20 EligibleTime=2014-05-06T11:20:20
   StartTime=Unknown EndTime=Unknown
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=standard AllocNode:Sid=pippin:31236
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=2 NumCPUs=8 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/export/home/pippin/payerle/slurm-tests/test1.sh
   WorkDir=/home/pippin/payerle/slurm-tests
```

The qsub command

Submitting jobs to the scheduling system is probably the most important part of the scheduling system, and because you need to be able to express the requirements of your job so that it can be properly scheduled, is also the most complicated. The Torque command is `qsub`; the Slurm equivalent is `sbatch`. But the complicated part is in the parameters to specify the job requirements.

Both Torque/Moab and Slurm allow you to specify arguments to `qsub`/`sbatch`, respectively, either on the command line or as comments in the script file. In Torque/Moab environments, the latter are in lines that begin `#PBS` . For Slurm, the lines should begin `#SBATCH` . In either case, they should occur before any executable line in the script.

The following table translates some of the more commonly used options for `qsub` to their `sbatch` equivalents:

qsub to sbatch translation

To specify the:	qsub option	sbatch option	Comments
Queue/partition	-q *QUEUENAME*	-p *QUEUENAME*	Torque "queues" are called "partitions" in slurm. **Note:** the partition/queue structure has been simplified, see below.
Account/allocation to be charged	-A *ACCOUNTNAME*	-A *ACCOUNTNAME*	Gold uses "allocations", Slurm uses "accounts"
Number of nodes/ cores requested	-l nodes=*NUMBERCORES*	-n *NUMBERCORES*	See below
	-l nodes=*NUMBERNODES*:*CORESPERNODE*	-N *NUMBERNODES* -n *NUMBERCORES*	
Wallclock limit	-l walltime=*TIMELIMIT*	-t *TIMELIMIT*	*TIMELIMIT* should have form of *HOURS:MINUTES:SECONDS*. Slurm supports some other time formats as well.
Memory requirements	-l mem=*MEMORY*mb	--mem=*MEMORY*	Moab: This is Total memory used by job Slurm: This is memory per node
	-l pmem=*MEMORY*mb	--mem-per-cpu=*MEMORY*	This is per CPU/core. *MEMORY* in MB

Stdout file	-o *FILENAME*	-o *FILENAME*	This will combine stdout/stderr on slurm if `-e` not given also
Stderr file	-e *FILENAME*	-e *FILENAME*	This will combine stderr/stdout on slurm if `-o` not given also
Combining stdout/stderr	-j oe	-o *OUTFILE* and no `-e`option	stdout and stderr merged to stdout/*OUTFILE*
	-j eo	-e *ERRFILE* and no `-o`option	stdout and stderr merged to stderr/*ERRFILE*
Email address	-M *EMAILADDR*	--mail-user=*EMAILADDR*	
Email options	-mb	--mail-type=BEGIN	Send email when job starts
	-me	--mail-type=END	Send email when job ends
	-mbe	--mail-type=BEGIN --mail-type=END	Send email when job starts and ends
Job name	-N *NAME*	--job-name=NAME	
Working directory	-d *DIR*	--workdir=*DIR*	

The partition/queue structure has been simplified with the transition to Slurm, and in most cases you will not need to specify a partition.

The node/cpu specifications with Moab/Torque are complicated and counter-intuitive. `-l nodes=N<` requests *N* **CORES**, and `-l nodes=N:ppn=M` requests *N* sets of *M* cores, but two or more sets could end up on the same node.

Slurm's node specification parameters are more straightforward, but as such do not cleanly map onto the Moab/Torque conventions. You can request a number of *tasks*, and by default Slurm assigns a distinct core for each task, with the `--ntasks=T` or `-n T` option. You can also request a number of nodes, with the `--nodes=N` or `-N N` option. In the node specification, *N* can either be a single number, e.g. `-N 4` requests 4 nodes, or a range, e.g. `-N 1-4` requests between 1 and 4 nodes (inclusive).

You can specify both a task and node count, in which case the scheduler will only assign you a set of nodes meeting both criteria, or you can specify either one alone, in which case only the single criterion is considered. Slurm also has many other options for specifying node/CPU/core requirements which are not matched by Moab/Torque.

Batch Job environment

In addition to specifying the environment, there is also the matter of the scheduler telling the job about what resources were allocated to it. This is generally done via environmental variables. This is harder to emulate, but some basic correlations are given below:

Moab/Torque to Slurm Environment Correlations

Function	Moab/Torque Variable	Slurm Variable	Comments
Job ID	$PBS_JOBID	$SLURM_JOBID	
Job Name	$PBS_JOBNAME	$SLURM_JOB_NAME	
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	
Node List	cat $PBS_NODEFILE	$SLURM_JOB_NODELIST	See below
Host submitted from	$PBS_O_HOST	$SLURM_SUBMIT_HOST	
Number of nodes allocated to job	$PBS_NUM_NODES	$SLURM_JOB_NUM_NODES	
Number of cores/node	$PBS_NUM_PPN	$SLURM_CPUS_ON_NODE	
Total number of cores for job???	$PBS_NP	$SLURM_NTASKS	Uncertain about these

Index to node running on relative to nodes assigned to job	$PBS_O_NODENUM	$SLURM_NODEID
Index to core running on within node	$PBS_O_VNODENUM	$SLURM_LOCALID
Index to task relative to job	$PBS_O_TASKNUM	$SLURM_PROCID + 1

In addition, Torque/Moab defines the variables `PBS_O_XXXX`, where *XXXX* is one of:

- `HOME`
- `LANG`
- `LOGNAME`
- `PATH`
- `MAIL`
- `SHELL`
- `TZ`

to the value of the corresponding *XXXX* environmental variable at the time the `qsub` command was run. Slurm does not set corresponding variables, but for our environment, these should generally be the same as the value of the *XXXX* variable at the time the job is running.
The list of nodes allocated to a job is handled quite differently between the two schedulers. Moab/Torque returns this in a file, with one line listing the hostname for every **core** assigned, and sets the variable `$PBS_NODEFILE` to the name of that file.

Slurm returns the actual list of nodes, in a compact notation with numeric ranges in square brackets, in the actual `$SLURM_JOB_NODELIST` environmental variable.

For OpenMPI users, the `mpirun` command should be able to handle either format, so you do not need to worry about this. For other users, this might be more of an issue. To assist with this, there is an utility `generate_pbs_nodefile` which will generate a Torque/Moab-compatible `$PBS_NODEFILE` file from the `$SLURM_JOB_NODELIST`. You would use it like (for Csh style shells)

```
#!/bin/tcsh
#PBS -l nodes=12

setenv PBS_NODEFILE `generate_pbs_nodefile`

#Rest of your code below
...
```

For bourne style shells, the equivalent would be

```
#!/bin/sh
#PBS -l nodes=12

PBS_NODEFILE=`generate_pbs_nodefile`
export PBS_NODEFILE

#Rest of your code below
...
```

If you need more than just the PBS_NODEFILE, a small script exists that attempts to set all the Torque/Moab variables above to the best values it can, including PBS_NODEFILE. To use it, you must source the script (Do NOT run as a program as it needs to set environmental variables for you) near the top of your script, for C-shell style scripts it would be:

```
#!/bin/tcsh
#PBS -l nodes=12

source /usr/local/slurm/scripts/slurm2pbs_environment.csh

#Rest of your code below
...
```

For bourne style shells, the equivalent would be

```
#!/bin/sh
#PBS -l nodes=12

. /usr/local/slurm/scripts/slurm2pbs_environment.sh

#Rest of your code below
...
```

The qdel command

Sometimes one needs to kill a job. Under Torque/Moab one would use the `qdel` or `canceljob` commands. The equivalent command under Slurm is `scancel`.

```
login-1> scancel -i 122488
Cancel job_id=122488 name=test1.sh partition=standard [y/n]? y
login-1>
```

The showstart command

The `showstart` command under Moab/Torque shows the scheduler's estimate of when a pending/idle job will start running. It is, of course, just the scheduler's best estimate, given current conditions, and the actual time a job starts might be earlier or later than that depending on factors such as the behavior of currently running jobs, the submission of new jobs, and hardware issues, etc.

This information is available under Slurm with the `squeue` command, although it is not shown by default. You need to specify the `%S` field in the output format option, e.g.

```
login-1> squeue -o "%.9i %.9P %.8j %.8u %.2t %.10M %.6D %S"
    JOBID PARTITION     NAME     USER ST       TIME  NODES START_TIME
      473  standard test1.sh  payerle PD       0:00      4 2014-05-08T12:44:34
      479  standard test1.sh    kevin PD       0:00      4 N/A
      489  standard tptest1.  payerle PD       0:00      2 N/A
```

The mybalance and gbalance commands

It is important to keep track of the available SUs (CPU-hours) in your accounts. Under Gold, you would due this with the `mybalance` and/or `gbalance` commands.

You can obtain the same information under Slurm with the `sbank` command. In particular:

```
login-1> sbank balance statement -u
User             Usage |          Account       Usage | Account Limit   Available (CPU hrs)
---------- ----------- + --------------- ----------- + ------------- -----------
payerle              5 |             TEST          99 |         1,000         901
payerle              1 |           TPTEST           1 |             2           1
```

Without the `-u`, it will show usage for other users in accounts which you belong to also. All numbers are in SU (CPU-hours); while the accounting system tracks things to the CPU-second, all output is rounded to the nearest CPU-hour (SU). The above shows user `payerle` as having access to two accounts:

- Account `TEST` with 1000 SU limit, of which he used 5 SU and others in the account used 94 SU (for a total of 99 SU used), leaving 901 SU available for use.
- Account `TPTEST` with a 2 SU limit, of which he used 1 SU leaving 1 SU available for use.

The qpeek command

There is a contributed script for Moab/Torque called `qpeek` that allows one to see the partial output of a job while it is still running. This is not needed on Slurm, as the standard output/error are written to the files you specified in real time, and these can just be viewed.

Wrappers for Moab commands

In order to make the transition from Moab/Torque to Slurm as minimally inconvenient as possible, we have installed a number of wrapper scripts for Slurm which are named after and behave much like the Moab/Torque commands you are used to. These are intended to smooth the transition, and we encourage you to over time start using native Slurm commands instead, as a major motivation for the migration of schedulers is that Slurm has many features that are just not present in Moab/Torque, and the wrapper scripts will not provide access to those features. Also, the wrapper scripts

are NOT perfect equivalents; they should be able to handle many basic cases, but not always perfectly, and some cases they just will not handle. Although we have no plans at this time to remove the wrapper scripts at any point in the future, our support policy is one of "If it works, great. If it doesn't work, use native Slurm commands."

In this section we discuss some of the wrapper scripts, and list any known quirks, etc. with them. For commands that display information to the user, we do not expect the format of the information to even closely match that of the original Moab/Torque commands, just that the same basic information is available. Those differences will be glossed over below unless there is a specific item of confusion, etc. in the reading the two outputs.

Wrappers exist for the following commands:

- `pbsnodes`
- `qalter`
- `qdel`
- `qhold`
- `qrerun`
- `qrls`
- `qstat`
- `qsub`
- `showq`

Most of the above are quite straight-forward. `showq` takes somewhat different arguments than the original Moab/Torque version; the main difference is that `-u` displays your jobs only, and does NOT take an username as argument. The `-U` *`USERNAME`* option is the equivalent of the orginal Moab/Torque `-u` option, displaying only jobs for the specified user.

The `qsub` wrapper is probably the most complex, and the one which exposes the differences between Moab/Torque and Slurm the most. Most of the basic options should work, and most resource requirements lists should be translated to roughly equivalent resource requirements in Slurm.

One major issue is that although the `qsub` tries to correctly translate your arguments to `qsub` (either on the command line or in `#PBS` comment lines) into the Slurm `sbatch` equivalents, it **does NOT** provide the equivalent environment to your script that the Moab/Torque versions do. In particular, Moab/Torque creates a file with the list of nodes assigned to your job and puts the path to the file in `$PBS_NODEFILE`. Slurm does not do that (Slurm provides essentially the same information, but by other means). This is discussed in some more detail elsewhere.

The main point is that if your script is relying on `$PBS_NODEFILE` or other environmental variables set by Moab/Torque, it probably won't be happy without some modifications. We have attempted to assist with some scripts that can be sourced at the top of your script file to try to mimic the Moab/Torque job script environment. To use in a C-shell style shell, you would do something like:

```
#!/bin/tcsh
#PBS -l nodes=12

source /usr/local/slurm/scripts/slurm2pbs_environment.csh

#Rest of your code below
...
```

For bourne style shells, the equivalent would be

```
#!/bin/sh
#PBS -l nodes=12

. /usr/local/slurm/scripts/slurm2pbs_environment.sh

#Rest of your code below
...
```

Back to Top