

Quick Guide for the Torque Cluster Manager

Introduction:

One of the main purposes of the Mars Cluster is to accommodate especially long-running programs. Users who run long jobs (which take hours or days to run) will need to run these jobs through the Torque Scheduler. Torque provides a method for handling these jobs on a first-come first-served basis. In this manner, all jobs will run more efficiently and finish faster since each is allowed to have all system resources for the duration of its run. **All Torque Jobs must be launched from the marsrv.** (Note: Only marsrv, mars1 through mars6 are online at the time of this document's creation.)

The Three Most Common Commands:

The basic commands provided by Torque for starting and stopping jobs and for manipulating jobs in queues are shown below. For complete manual of Torque command, type "man <cmd>" (e.g. "man qsub").

qsub - the basic command for running jobs

qstat - show currently running jobs/queued jobs

pbsnodes -a - gives the current status of all nodes

How to Run a Batch Job

The Torque Scheduler will not accept an R program or a C program directly. It is designed instead to accept a shell script - a .sh file - which itself runs the commands necessary to launch your program. Once your script is ready, you can submit it to Torque with the qsub command:

```
qsub your_script.sh
```

When this command is issued, you will be given a job number, we will call it **XX** in this example, which is used for tracking and manipulating your job. Standard out and standard error from the job will be copied to files in your current directory named *your_script.sh.oXX* and *your_script.sh.eXX* for your reference.

To Run a Job on Several Nodes

If your job requires more than one node in order to run, the number of nodes can be specified when issuing the “qsub -l” command. In the following example, the test1.sh script requires 4 nodes in order to run. Issue this command to request 4 nodes for this job.

```
% qsub -l nodes=4 ~/test1.sh
```

Once this command is issued, the system will verify that there are 4 machines ready for use. If there are, it will allocate 4 machines for this job and this job will run. If 4 machines are not yet available, the job will be put in the queue.

A user can issue the “qstat” command to track the status of the job. In the example below, the status of this job is shown as “C” for “Complete.” A job can also show a status of “Q” (Queued) or “R” (Running).

```
[curt@marssrv ~]$ qstat
Job id          Name          User          Time Use S Queue
-----
6.marssrv      test1.sh      curt          00:00:00 C batch
```

Note - The text of test1.sh is shown for your reference in the Appendix.

Appendix

I - Further Details Regarding the Basic Commands:

About qsub

The qsub command submits a sequence of commands to the batch server along with the parameters specifying job resource requirements. The parameters may be provided on the command line, from within the job script, or a combination of both. To facilitate optimal scheduling, you should specify as many resources as possible.

The syntax for the qsub command is:

qsub [option(s)] [script-file]

The options field typically contains one or more -l or -W switch each followed by a comma-separated list of parameters. Commonly used options for use with the -l switch are given in the following table.

| <i>-l option</i> | Default | Action |
|---------------------------------|---------|---|
| <i>walltime=HH:MM:SS</i> | | The length of time your job will need to run. Your job will end after the allocated walltime has expired whether it is finished or not, so choose this value carefully. Appropriate walltime should be chosen in order to prevent programs that either run out of control, or who never exit, from consuming all system resources. If this option is not specified, the system will use default walltime settings. |
| <i>mem=N[measure]</i> | 2 GB | The maximum amount of memory the job is expected to use. The measure may be in GB, MB, KB, B. |

| | | |
|-----------------------|---|---|
| <i>ppn=N</i> | | Specifies the number of CPUs per node. Use 1 because currently, every machine on the mars cluster has a single core. |
| <i>nodes=n</i> | 1 | Specifies the number of nodes required. |

Examples:

qsub -l nodes=4:ppn=1,mem=2gb,walltime=1:00:00 your_script.sh

(requests four nodes with 1 processor per node, 2gb required per node, and a maximum runtime of 1 hour)

qsub -l nodes=mars1+mars3+mars6 your_script.sh

(requests 3 specific nodes)

qsub -l nodes=mars1+4 your_script.sh

(requests mars1 and any other 4 nodes)

About qstat

The qstat command monitors the status of all jobs currently submitted to Torque on the mars cluster.

Example:

qstat show all jobs

qstat -a show all jobs, alternate format

About qdel

A queued job may be removed from a queue or a running job may be killed using the qdel command.

Example:

qdel 1234

(where '1234' is the job ID. The job ID can be obtained with the qstat command.)

II - The test1.sh script

This script is designed to run with Torque and requires multiple nodes to operate. The script, when run through Torque, will examine each of the nodes present (as determined by the `-l nodes` parameter) and return the name of each available node to a file named `cluster_nodes`. This script also cleans up old `test1.sh.o*` files that might be present in the current directory, leaving only the most current output.

```
#!/bin/csh
```

```
cd $PBS_O_WORKDIR
```

```
#direct the output to cluster_nodes
```

```
cat $PBS_NODEFILE > ./cluster_nodes
```

```
rm test1.sh.*
```