Contents

1 Introduction 4
  1.1 Feature .................................................. 4
  1.2 Status .................................................. 5

2 Overview 6
  2.1 Installation ............................................. 6
    2.1.1 Architecture ..................................... 6
    2.1.2 Compute Nodes ................................... 7
    2.1.3 Enabling TORQUE as a Service ................... 8
  2.2 Basic Configuration .................................. 8
    2.2.1 Initialize/Configure TORQUE on the Server (pbs_server) 8
    2.2.2 Specify Compute Nodes ............................. 9
    2.2.3 Initialize/Configure TORQUE on the Each Compute Node 10
    2.2.4 Finalize Configurations ........................... 10
  2.3 Customizing the Install .............................. 10
    2.3.1 Server Config Overview ............................ 11
    2.3.2 Name Service Config ............................... 11
    2.3.3 Configuring Job Submission Hosts ................. 11
    2.3.4 Configuring TORQUE on a Multi-Homed Server ...... 11
    2.3.5 Specifying Non-Root Administrators ............... 12
  2.4 Testing ................................................. 12

3 Submitting and Managing Jobs 14
  3.1 Job Submission .......................................... 14
    3.1.1 Requesting Resources .............................. 14
    3.1.2 Requesting Generic Resources ..................... 15
    3.1.3 Requesting Floating Resources ................... 15
    3.1.4 Requesting Other Resources ....................... 16
    3.1.5 Requesting Other Resources ....................... 16
  3.2 Monitoring Jobs ......................................... 16
  3.3 Canceling Jobs .......................................... 16
  3.4 Job Preemption .......................................... 16
  3.5 Completed Jobs .......................................... 17

4 Managing Nodes 18
  4.1 Adding Nodes ........................................... 18
    4.1.1 Run-Time Node Changes ............................ 18
  4.2 Nodes Properties ....................................... 18
    4.2.1 Run-Time Node Changes ............................ 18
    4.2.2 Manual Node Changes ............................... 18
  4.3 Changing Node State .................................... 19
5 Setting Server Policies
5.1 Queue Configuration ........................................... 20
  5.1.1 Queue Attributes ........................................... 20
  5.1.2 Example Queue Configuration ................................. 20
  5.1.3 Setting a Default Queue ..................................... 20
  5.1.4 Mapping a Queue to a Subset of Resources ................. 21
  5.1.5 Creating a Routing Queue ................................... 21
  5.1.6 Server Parameters ......................................... 22

6 Configuring Data Management ........................................ 23
  6.1 SCP/RCP Setup ................................................ 23
    6.1.1 Generate SSH Key on Source Host .......................... 23
    6.1.2 Copy Public SSH Key to Each Destination Host .......... 23
    6.1.3 Configure the SSH Daemon on Each Destination Host .... 23
    6.1.4 Validating Correct SSH Configuration ..................... 24
    6.1.5 Enabling Bi-Directional SCP Access ....................... 24
    6.1.6 Compile TORQUE to Support SCP ........................... 24
  6.2 NFS and Other Networked Filesystems .......................... 24
    6.2.1 TORQUE Data Management ................................. 24
  6.3 File Stage-In/Stage-Out ........................................ 25

7 Interfacing with Message Passing .................................. 26
  7.1 MPI (Message Passing Interface) Support ........................ 26
    7.1.1 MPI (Message Passing Interface) Overview ................ 26
    7.1.2 MPICH .................................................... 26
    7.1.3 MPICH-VMI ............................................... 27
    7.1.4 LAM-MPI .................................................. 27
1 Introduction

TORQUE is an open source resource manager providing control over batch jobs and distributed compute nodes. It is a community effort based on the original PBS project and, with more than 1,200 patches, has incorporated significant advances in the areas of scalability, fault tolerance, and feature extensions contributed by NCSA, OSC, USC, the U.S. Dept of Energy, Sandia, PNNL, University of Buffalo, TeraGrid, and many other leading edge HPC organizations. It may be freely modified and redistributed subject to the constraints of the included license in download package.

TORQUE can integrate with other packages like Maui Scheduler and Moab Workload Manager to improve overall utilization, scheduling and administration on a cluster.

1.1 Feature

TORQUE provides enhancements over standard OpenPBS in the following areas:

- Fault Tolerance
  - Additional failure conditions checked/handled
  - Node health check script support
- Scheduling Interface
  - Extended query interface providing the scheduler with additional and more accurate information
  - Extended control interface allowing the scheduler increased control over job behavior and attributes
  - Allows the collection of statistics for completed jobs
- Scalability
  - Significantly improved server to MOM communication model
  - Ability to handle larger clusters (over 15 TF/2,500 processors)
  - Ability to handle larger jobs (over 2000 processors)
  - Ability to support larger server messages
- Usability
  - Extensive logging additions
  - More human readable logging (ie no more ’error 15038 on command 42’)

4
1.2 Status

TORQUE is freely available for download

http://www.clusterresources.com/downloads/torque

TORQUE users can subscribe to TORQUEs mailing list or view the archive for questions, comments or patches. Please send mail directly to help@supercluster.org if you have any patches to contribute or if you are aware of any issues in the distribution.
2 Overview

2.1 Installation

If you have not yet download the TORQUE, please go and download it here:

http://www.clusterresources.com/downloads/torque

Extract and build the distribution on the machine that will act as the TORQUE server - the machine that will monitor and control all compute nodes by running the pbs_server daemon. See the example below (where XXX stands for the latest distribution (e.g., -1.2.0p4):

```$ tar -xzvf torqueXXX.tar.gz
$ cd torqueXXX
$ ./configure
$ make
$ make install```

It’s better and more comfortable to set the PATH environment variable for fur-
ther references to TORQUE commands. The default installation directories for
the binaries are either /usr/local/bin and /usr/local/sbin.

In this document $(TORQUECFG) corresponds to where TORQUE stores its
configuration files. This defaults to /usr/spool/PBS.

TORQUE 2.0p2 and higher includes a standard spec file for building your
own rpms. Simply run:

```$ rpmbuild -ta torqueXXX.tar.gz```

to generate RPM packages. It is also possible to use the checkinstall program
to create your own RPM, tgz, or deb package.

2.1.1 Architecture

A TORQUE cluster consists of 1 headnode and many compute nodes. The
headnode runs the pbs_server daemon and the compute nodes run the pbs_mom
daemon. Client commands for submitting and managing jobs can be installed
on any host (including hosts that dont run pbs_server or pbs_mom.)

The headnode will also run a scheduler daemon. The scheduler interacts
with pbs_server to make local policy decisions for resource usage and allocate
nodes to jobs. A simple FIFO scheduler, and code to construct more advanced
schedulers are provided in the TORQUE source distribution, but most sites opt
for a packaged advanced scheduler like Maui.

Users submit jobs to pbs_server using the qsub command. When pbs_server receives a new job, it informs the scheduler. If and when the scheduler finds nodes for the job, it sends instructions to run the job with the nodelist to pbs_server. pbs_server sends the new job to the first node in the nodelist to launch the job. This node is designated as the execution host or Mother Superior. Other nodes in a job are called sister moms.

2.1.2 Compute Nodes

To install TORQUE on a compute node do the following on each machine:

- Create the self-extracting, distributable packages with make packages (See the INSTALL file for additional options and features of the distributable packages) and use the parallel shell command from your cluster management suite to copy and execute the package on all nodes ie: xCAT users might do:

```
prcp torque-package-linux-i686.sh main:/tmp/
psh main /tmp/torque-package-linux-i686.sh install
```

Optionally, distribute and install the clients package.

- Although optional, it is also possible to use the TORQUE server as a compute node and install a pbs_mom alongside the pbs_server daemon.

Here it is an example for compute node installation:

```
$ make packages
Building ./torque-package-clients-linux-i686.sh ...
Building ./torque-package-mom-linux-i686.sh ...
Building ./torque-package-server-linux-i686.sh ...
Building ./torque-package-gui-linux-i686.sh ...
Building ./torque-package-devel-linux-i686.sh ...
Done.
```

The package files are self-extracting packages that can be copied and executed on your production machines. Use help for options.
Both `pbs_iff` and `pbs_rcp` will be installed suid root.

### 2.1.3 Enabling TORQUE as a Service

An optional startup/shutdown service script is provided as an example of how to run TORQUE as an OS service that starts at bootup.

- There is a script in Appendix A. Note that, this script was written specifically for Redhat variants, and may require modification to work with other Linux/UNIX distributions.
- Place the file in `/etc/init.d/` directory.
- Make symbolic links (`S99torque` and `K15torque`, for example) in desired runtime levels (e.g. `/etc/rc.d/rc3.d/` on Redhat, etc.) This can be added to the self-extracting packages (See INSTALL for details.)

### 2.2 Basic Configuration

TORQUE only requires a few steps to get the system initially configured and running. First, the server needs to be initialized and configured. Second, each of the compute nodes needs to be specified. Finally, each of the compute nodes need to know where the `pbs_server` is located.

#### 2.2.1 Initialize/Configure TORQUE on the Server (pbs_server)

The first time `pbs_server` is run, it needs to be run with `-t create` to initialize the `serverdb`. This is a binary database that holds almost all configurations. Now run `qmgr`, which gives you a shell prompt to configure `pbs_server`. The first configurations should be setting the operators, creating an initial job queue, and enable scheduling. List the configuration with print server. There is also an online help within `qmgr` with the `help` command, and be sure to read over the `qmgr` manpage.
set server operators = root@headnode
set server operators += username@headnode
create queue batch
set queue batch queue_type = Execution
set queue batch started = True
set queue batch enabled = True
set server default_queue = batch
set server resources_default.nodes = 1
set server scheduling = True

If you experience problems, make sure that the most recent TORQUE executables are being executed, or that the executables are in your current PATH.

Proper server configuration can be verified by following the steps listed in Testing section.

2.2.2 Specify Compute Nodes

In order for the pbs_server to communicate with each of the pbs_mom, it needs to know which machines to contact. Each node which is to be a part of the batch system must be specified on a line in the server nodes file. This file is located at $TORQUECFG/server_priv/nodes. In most cases, it is sufficient to specify just the node name on a line as in the following example:

node001
node002
node003
node004

If the compute node has multiple processors, specify the number of processors with np=number of processors. For example, if node001 has 2 processors and node002 has 4:

node001 np=2
node002 np=4
...
2.2.3 Initialize/Configure TORQUE on the Each Compute Node

The minimal requirement for the MOM configuration, is that the hostname of the headnode must be in the $(TORQUECFG)/server_name file.

Additional configuration may be added to $(TORQUECFG)/mom_priv/config file on each node. See the pbs_mom manpage for a complete listing. For example,

```
$pbsserver headnode  # note: hostname running pbs_server
$logevent 255        # bitmap of which events to log
```

Since this file is identical for all compute nodes, it can be created on the head node and distributed in parallel to all systems.
Start the pbs_mom daemon on all nodes.

2.2.4 Finalize Configurations

After the serverdb is configured, the server_priv/nodes file is completed, and MOM has a minimal configuration, restart pbs_server:

```
qterm -t quick
pbs_server
```

Wait a few seconds, and pbsnodes -a should list all nodes in state free.

2.3 Customizing the Install

The TORQUE configure command takes a number of options. One key option:

By default, TORQUE uses rcp to copy data files. Using scp is highly recommended, use with-scp (see ssh setup for more information)

2.3.1 Server Config Overview

There are several steps to ensure that the server and the nodes are completely aware of each other and able to communicate directly. Some of this configuration takes place within TORQUE directly using the qmgr command. Other configuration settings are managed using the pbs_server nodes file, DNS files such as /etc/hosts and the /etc/hosts.equiv file.

2.3.2 Name Service Config

Each node, as well as the server, must be able to resolve the name of every node with which it will interact. This can be accomplished using /etc/hosts, DNS,
NIS, or other mechanisms. In the case of /etc/hosts, the file can be shared across systems in most cases.

A simple method of checking proper name service configuration is to verify that the server and the nodes can ping each other.

### 2.3.3 Configuring Job Submission Hosts

When jobs can be submitted from several different hosts, these hosts should be trusted via the R* commands (i.e., rsh, rcp, etc.). This can be enabled by adding the hosts to the /etc/hosts.equiv file of the machine executing the pbs_server daemon or using other R* command authorization methods. The exact specification can vary from OS to OS (see the man page for ruserok to find out how your OS validates remote users). In most cases, configuring this file is as simple as adding a line to your /etc/hosts.equiv as in the following:

```
# [+ | -] [hostname] [username]
mynode.myorganization.com
.....
```

Please note that when a hostname is specified, it must be the fully qualified domain name (FQDN) of the host. Job submission can be further secured using the server or queue acl_hosts and acl_host_enabled parameters.

If desired, all compute nodes can be enabled as job submit hosts without setting .rhosts or hosts.equiv by setting the ALLOWCOMPUTEHOSTSUBMIT parameter.

### 2.3.4 Configuring TORQUE on a Multi-Homed Server

If the pbs_server daemon is to be run on a multi-homed host (a host possessing multiple network interfaces), the interface to be used can be explicitly set using the SERVERHOST parameter.

### 2.3.5 Specifying Non-Root Administrators

By default, only root is allowed to start, configure and manage the pbs_server daemon. Additional trusted users can be authorized using the parameters managers and operators. To configure these parameters use the qmgr command as in the example below:
All manager and operator specifications must include a user name and either a fully qualified domain name or a host expression.

**Note:** To enable all users to be trusted as both operators and admins, place the + character (plus) on its own line in the file `server_priv/acl_svr/operators` and `server_priv/acl_svr/managers`.

### 2.4 Testing

The `pbs_server` daemon was started on the TORQUE server when the `torque.setup` file was executed or when it was manually configured. It must now be restarted so it can reload the updated configuration changes.

```plaintext
# shutdown server
$ qterm -t quick

# start server
$ pbs_server

# verify all queues are properly configured
$ qstat -q

# view additional server configuration
$ qmgr -c 'p s'

# verify all nodes are correctly reporting
$ pbsnodes -a

# submit a basic job
$ echo "sleep 30" | qsub

# verify jobs display
$ qstat
```

At this point, the job should be in the Q state and will not run. This is
because a scheduler isn't running yet. TORQUE can use its native scheduler by running `pbs_sched` or an advanced scheduler can be used. 5.1 Integrating Schedulers for TORQUE details setting up an advance scheduler.
3 Submitting and Managing Jobs

3.1 Job Submission

Job submission is accomplished using the `qsub` command. This command takes a number of command line arguments and integrates this into the specified PBS command file. The PBS command file may be specified as a filename on the `qsub` command line or may be entered via `STDIN`.

- The PBS command file does not need to be executable.
- The PBS command file may be piped into `qsub` (i.e., `cat pbs.cmd — qsub`)
- In the case of parallel jobs, the PBS command file is staged to, and executed on the first allocated compute node only. (use `pbsdsh` to run actions on multiple nodes)
- The command script is executed from the users home directory in all cases (the script may determine the submission directory by using the `$PBS_O_WORKDIR` environment variable)
- The command script will be executed using the default set of user environment variables unless the `-V` or `-v` flags are specified to include aspects of the job submission environment.

By default, job submission is allowed only on the TORQUE server host (host on which `pbs_server` is running). Enablement of job submission from other hosts is documented Advanced Configuring Job Submit Hosts.

3.1.1 Requesting Resources

Various resources can be requested at the time of job submission. A job can request a particular node, a particular node attribute, or even a number of nodes with particular attributes. Either native TORQUE resources, or external scheduler 13.3 RM Extensions resource extensions may be specified. The native TORQUE resources are listed in the table below:

*size format: The size format specifies the maximum amount in terms of bytes or words. It is expressed in the form `integer[suffix]`. The suffix is a multiplier defined in the following table (b means bytes (the default) and w means words). The size of a word is calculated on the execution server as its word size.

Example 1 (-l nodes)

- To ask for 12 nodes of any: `-l nodes=12`
- To ask for 2 server nodes and 14 other nodes (a total of 16): `-l nodes=2:server+ 14`
- The above consist of two `node_specs 2:server` and 14.
- To ask for (a) 1 node that is a server and has a hippi interface, (b) 10 nodes that are not servers, and 3 nodes that have a large amount of memory and have hippi: `-l nodes=server:hippi+10:noserver+3:bigmem:hippi`

- To ask for three nodes by name: `-l nodes=b2005+b1803+b1813`

- To ask for 2 processors on each of four nodes: `-l nodes=4:ppn=2`

- To ask for 4 processors on one node: `-l nodes=1:ppn=4`

- To ask for 2 processors on each of two blue nodes and three processors on one red node: `-l nodes=2:blue:ppn=2+red:ppn=3`

**Example 2**

```bash
$ qsub -l mem=200mb /home/user/script.sh
```

This job requests a node with 200 MB of available memory.

**Example 3**

```bash
$ qsub -l nodes=node01,mem=200mb /home/user/script.sh
```

This job will wait until `node01` is free with 200 MB of available memory.

### 3.1.2 Requesting Generic Resources

When generic resources have been assigned to nodes using the servers nodes file, these resources can be requested at the time of job submission using the `other` field.

**Example 4**

```bash
$ qsub -l other=matlab /home/user/script.sh
```

This job will run on any node that has the generic resource `matlab`.

**Note:** This can also be requested at the time of job submission using the `-W x=GRES:matlab` flag.

### 3.1.3 Requesting Floating Resources

When floating resources have been set up inside Moab, they can be requested in the same way as generic resources. Moab will automatically understand that these resources are floating and will schedule the job accordingly.

**Example 5**

```bash
$ qsub -l other=matlab /home/user/script.sh
```

This job will run on any node when there are enough floating resources available.

**Note:** This can also be requested at the time of job submission using the `-W x=GRES:matlab` flag.
3.1.4 Requesting Other Resources

Many other resources can be requested at the time of job submission using the Moab Workload Manager. See the Resource Manager Extensions page for a list of these supported requests and correct syntax.

3.1.5 Requesting Other Resources

When a batch job is started, a number of variables are introduced into the jobs environment which can be used by the batch script in making decisions, creating output files, etc. These variables are listed in the table below:

3.2 Monitoring Jobs

TORQUE allows users and administrators to monitor submitted jobs with the qstat command. If the command is run by just a user, it will output just that users jobs. For example,

$ qstat

<table>
<thead>
<tr>
<th>Job id</th>
<th>Name</th>
<th>User</th>
<th>Time</th>
<th>Use</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>4807</td>
<td>scatter</td>
<td>user01</td>
<td>12:56:34</td>
<td>R</td>
<td>batch</td>
</tr>
</tbody>
</table>

3.3 Canceling Jobs

TORQUE allows users and administrators to cancel submitted jobs with the qdel command. The job will be sent a TERM and KILL signals killing the running processes. When the top-level job script exits, the job will exit. Simply supply it the jobid to be canceled.

If a job is canceled by an operator or manager, an email notification will be sent to the user. Operators and managers may add a comment to this email with the -m option.

$ qstat

<table>
<thead>
<tr>
<th>Job id</th>
<th>Name</th>
<th>User</th>
<th>Time</th>
<th>Use</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>4807</td>
<td>scatter</td>
<td>user01</td>
<td>12:56:34</td>
<td>R</td>
<td>batch</td>
</tr>
</tbody>
</table>

$ qdel -m "hey! Stop abusing the NFS servers" 4807

3.4 Job Preemption

TORQUE supports job preemption by allowing authorized users to suspend and resume jobs. This is supported using one of two methods. If the node sup-
ports OS-level preemption, TORQUE will recognize that during the configure process and enable it. Otherwise, the MOM may be configured to launch a custom checkpoint script in order to support preempting a job. Using a custom checkpoint script requires that the job understand how to resume itself from a checkpoint after the preemption occurs.

Configuring a Checkpoint Script on a MOM  To configure the MOM to support a checkpoint script, the $checkpoint_script parameter must be set in the MOMs configuration file found in $TORQUEHOME/mom_priv/config. The checkpoint script should have execute permissions set. A typical config file might look like:

```
$pbsserver node06
$logevent 255
$restricted *.mycluster.org
$checkpoint_script /opt/moab/tools/mom-checkpoint.sh
```

The second thing that must be done to enable the checkpoint script is to change the value of MOM_CHECKPOINT to 1 in .../src/include/pbs_config.h. In some instances, MOM_CHECKPOINT may already be defined as 1. The new line should be:

```
#define MOM_CHECKPOINT 1
```

3.5 Completed Jobs
TORQUE provides the ability to report on the status of completed jobs for a configurable duration after the job has completed. This can be enabled by setting the keep_completed attribute on the job execution queue. This should be set the number of seconds that jobs should be held in the queue. Completed jobs will be reported in the C state and the exit status is seen in the exit_status job attribute.

By maintaining status information about completed (or canceled, failed...etc) jobs, administrators can better track failures and improve system performance. This also allows TORQUE to better track the status of jobs.

Note: keep_completed is a queue attribute. A future release will also provide a server attribute.
4 Managing Nodes

4.1 Adding Nodes

TORQUE can add and remove nodes either dynamically with qmgr or by manually editing the $TORQUEHOME/server_priv/nodes file (see Basic Configuration).

4.1.1 Run-Time Node Changes

TORQUE can dynamically add nodes with the qmgr command. For example, the following command will add node node003:

$ qmgr -c "create node node003"

Nodes can also be removed with a similar command:

$ qmgr -c "remove node node003"

Note: Typically, an admin will wish to change the state of a node instead of remove it (see Changing Node State).

Note: As of TORQUE 2.0, this capability was unreliable at best. It is highly recommended that node changes be followed by a restart of pbs_server, or just edit the nodes file manually and restart it.

4.2 Nodes Properties

TORQUE can associate properties with nodes to aid in identifying groups of nodes. It's typical for a site to conglomerate a heterogeneous sets of resources. To identify the different sets, properties can be given to each node in a set. For example, a group of nodes that have a higher speed network connection could have the property \texttt{ib}. TORQUE can set, update or remove properties either dynamically with qmgr or by manually editing the nodes file.

4.2.1 Run-Time Node Changes

TORQUE can dynamically change the properties of a node with the qmgr command. For example, to give node001 the properties of \texttt{bigmem} and \texttt{dualcore}:

> qmgr -c "set node node001 properties = bigmem"
> qmgr -c "set node node001 properties += dualcore"

To relinquish a stated property, use the -= operator.

4.2.2 Manual Node Changes

The properties of each node are enumerated in $TORQUEHOME/server_priv/nodes. The feature(s) must be in a space delimited list after the node name. For example, to give node001 the properties of \texttt{bigmem} and \texttt{dualcore} and node002 the properties of \texttt{bigmem} and \texttt{matlab}, edit the nodes file to contain:
node001 bigmem dualcore
dnode002 np=4 bigmem matlab

Note: For changes to the nodes file to be activated, pbs_server must be restarted.
Note: For a full description of this file, please see the PBS Administrator Guide

4.3 Changing Node State

A common task is to prevent jobs from running on a particular node by marking it offline with pbsnodes -o nodename. Once a node has been marked offline, the scheduler will no longer consider it available for new jobs. Simply use pbsnodes -c nodename when the node is returned to service.

Also useful is pbsnodes -l which lists all nodes with an interesting state such as down, unknown or offline. This provides a quick glance at nodes that might be having a problem.
See the pbsnodes manpage for details
5 Setting Server Policies

5.1 Queue Configuration

Under TORQUE, queue configuration is accomplished using the `qmgr` command. With this tool, the first step is to create the queue. This is accomplished using the `create queue` subcommand of `qmgr` as in the following example:

```
$ qmgr -c "create queue batch queue_type=execution"
```

Once created, the queue must be configured to be operational. At a minimum, this includes setting the options started and enabled. Further configuration is possible using any combination of the attributes listed in the section below.

For boolean attributes, T, t, 1, Y, and y are all synonymous with true, and F, f, 0, N, and n all mean false. For `queue_type`, E and R are synonymous with Execution and Routing. Queues can be deleted with the `delete` command in `qmgr`.

5.1.1 Queue Attributes

Resources may include one or more of the following: `arch, mem, nodes, ncpus, nodect, pvmem, and walltime`.

5.1.2 Example Queue Configuration

The following series of `qmgr` commands will create and configure a queue named `batch`:

```
$ qmgr -c "create queue batch queue_type=execution"
$ qmgr -c "set queue batch started=true"
$ qmgr -c "set queue batch enabled=true"
$ qmgr -c "set queue batch resources_default.nodes=1"
$ qmgr -c "set queue batch resources_default.walltime=3600"
```

This queue will accept new jobs and, if not explicitly specified in the job, will assign a nodecount of 1 and a walltime of 1 hour to each job.

5.1.3 Setting a Default Queue

By default, a job must explicitly specify which queue it is to run in. To change this behavior, the server parameter `default_queue` may be specified as in the following example:

```
$ qmgr -c "set server default_queue=batch"
```
5.1.4 Mapping a Queue to a Subset of Resources

TORQUE does not currently provide a simple mechanism for mapping queues to nodes. However, schedulers such as Maui can provide this functionality.

The simplest method is using `default_resources.neednodes` on an execution queue, setting it to a particular node attribute. Maui will use this information to ensure that jobs in that queue will be assigned nodes with that attribute. For example, say we have some nodes bought with money from the chemistry department, and some nodes paid by the biology department.

```
$TORQUECFG/server_priv/nodes:
node01 np=2 chem
node02 np=2 chem
node03 np=2 bio
node04 np=2 bio
```

```
qmgr:
set queue chem resources_default.neednodes=chem
set queue bio resources_default.neednodes=bio
```

*Note:* This example does not preclude other queues from accessing those nodes. One solution is to use some other generic attribute with all other nodes and queues.

More advanced configurations can be made with standing reservations and QOSes. (See TORQUE Queue to Node Mapping)

5.1.5 Creating a Routing Queue

A routing queue will steer a job to a destination queue based on job attributes and queue constraints. It is set up by creating a queue of `queue_typeRoute` with a `route_destinations` attribute set as in the example below.

```
> qmgr -c 'p q route'
create queue route
set queue route queue_type = Route
set queue route route_destinations = batch
set queue route route_destinations += fast
set queue route route_destinations += slow
set queue route enabled = True
set queue route started = True
```

Routing decisions will be based on the `resources_max, resources_min, acl_users, and acl_groups` attributes of the destination queues.

*Note:* If the error message `qsub: Job rejected by all possible destinations` is reported when submitting a job, it may be necessary to add queue location information, (i.e., in the routing queues `route_destinations` attribute, change `batch` to `batch@localhost`)
5.1.6 Server Parameters

Under Construction! See the `pbs_server_attributes` manpage.
6 Configuring Data Management

6.1 SCP/RCP Setup

To utilize scp based data management, TORQUE must be authorized to migrate data to any of the compute nodes. If this is not already enabled within the cluster, this can be achieved with the process described below. This process enables uni-directional access for a particular user from a source host to a destination host.

These directions were written using OpenSSH version 3.6 and may not transfer correctly to older versions.

6.1.1 Generate SSH Key on Source Host

On the source host as the transfer user, execute:

```bash
$ ssh-keygen -t rsa
```

This will prompt for a passphrase (optional) and create two files `id_rsa` and `id_rsa.pub` inside `~/.ssh/`.

6.1.2 Copy Public SSH Key to Each Destination Host

Transfer public key to each destination host as the transfer user:

```
$ scp ~/.ssh/id_rsa.pub //destHost//:~ (enter password)
```

Create an `authorized_keys` file on each destination host.

```
$ ssh //destHost// (enter password)
$ cat id_rsa.pub >> .ssh/authorized_keys
```

(If the `.ssh` directory does not exist, create it with 700 privileges (`mkdir .ssh; chmod 700 .ssh`))

```
$ chmod 600 .ssh/authorized_keys
$ rm id_rsa.pub
```

6.1.3 Configure the SSH Daemon on Each Destination Host

Some configuration of the ssh daemon may be required on the destination host. (Because this is not always the case, skip down to step 4 and test the changes made up to this point. If the tests fail, proceed with this step and then try testing again.) Typically, this is done by editing the `/etc/ssh/sshd_config` file (root access needed.) To verify correct configuration, see that the following attributes are set (not commented):

```
RSAAuthentication yes
PubkeyAuthentication yes
```

If configuration changes were required, the ssh daemon will need to be restarted (root access needed):

```
$ /etc/init.d/sshd restart
```
6.1.4 Validating Correct SSH Configuration

If all is properly configured, the following command issued on the source host should succeed and not prompt for a password:

```bash
> scp //destHost//:etc/motd /tmp
```

Note that if this is your first time accessing destination from source, it may ask you if you wish to add the fingerprint to a file of known hosts. If you type yes and this message should no longer appear and should not interfere with scp copying via TORQUE. Also, it is important that the full hostname appear in the `known_hosts` file. To do this, use the full hostname for `destHost`, as in `machine.domain.org` instead of just `machine`.

6.1.5 Enabling Bi-Directional SCP Access

The above steps allows source access to destination without prompting for a password. The reverse, however, is not true. Repeat the above steps, but this time using the destination as the source, etc. to enable bi-directional SCP access (i.e. source can send to destination and destination can send to source without password prompts.)

6.1.6 Compile TORQUE to Support SCP

TORQUE must be re-configured (and then rebuilt) to use SCP by passing in the `with-scp` flag to the configure script:

```bash
> ./configure --prefix=xxx --with-scp
> make
```

Troubleshooting  If, after following all of these steps, TORQUE is still having problems transferring data with scp set the `PBSDEBUG` environment variable and restart the `pbs_mom` for details about copying. Also check the MOM log files for more details.

6.2 NFS and Other Networked Filesystems

6.2.1 TORQUE Data Management

When a batch job starts, its stdin file (if specified) is copied from the submission directory on the remote submission host. This file is placed in the `$PBSMOMHOME` directory on the mother superior node (i.e., `/usr/spool/PBS/spool`). As the job runs, stdout and stderr files are generated and placed in this directory using the naming convention `$JOBID.OU` and `$JOBID.ER`.

When the job completes, the MOM copies the files into the directory from which the job was submitted. By default, this file copying will be accomplished using a remote copy facility such as `rcp` or `scp`.

If a shared filesystem such as NFS, DFS, or AFS is available, a site can specify
that the MOM should take advantage of this by specifying the $usecp directive inside the MOM config file (located in the $PBSMOMHOME/mom_priv directory) using the following format $usecp <HOST>:<SRCDIR> <DSTDIR> Where HOST can be specified with a leading wildcard (*) character. The example below demonstrates this directive:

```bash
# /home is NFS mounted on all hosts
$ usecp *:/home /home
# submission hosts in domain fte.com should map '/data' directory on submit host to
# '/usr/local/data' on compute host
$ usecp *.fte.com:/data /usr/local/data
```

If for any reason the MOM daemon is unable to copy the output or error files to the submission directory, these files are instead copied to the undelivered directory also located in $PBSMOMHOME.

### 6.3 File Stage-In/Stage-Out

Under Construction!!
7  Interfacing with Message Passing

7.1  MPI (Message Passing Interface) Support

7.1.1  MPI (Message Passing Interface) Overview

A message passing library is used by parallel jobs to augment communication between the tasks distributed across the cluster. TORQUE can run with any message passing library and provides limited integration with some MPI libraries.

7.1.2  MPICH

One of the most popular MPI libraries is MPICH available from Argonne National Lab. If using this release, you may want to consider also using the mpiexec tool for launching MPI applications. Support for mpiexec has been integrated into TORQUE.

MPIExec  mpiexec is a replacement program for the script mpirun, which is part of the mpich package. It is used to initialize a parallel job from within a PBS batch or interactive environment. Mpiexec uses the task manager library of PBS to spawn copies of the executable on the nodes in a PBS allocation.

Reasons to use mpiexec rather than a script (mpirun) or an external daemon (mpd):

- Starting tasks with the TM interface is much faster than invoking a separate rsh * once for each process.
- Resources used by the spawned processes are accounted correctly with mpiexec, and reported in the PBS logs, because all the processes of a parallel job remain under the control of PBS, unlike when using mpirun-like scripts.
- Tasks which exceed their assigned limits of CPU time, wallclock time, memory usage, or disk space are killed cleanly by PBS. It is quite hard for processes to escape control of the resource manager when using mpiexec. You can use mpiexec to enforce a security policy. If all jobs are forced to spawn using mpiexec and the PBS execution environment, it is not necessary to enable rsh or ssh access to the compute nodes in the cluster.

See the mpiexec homepage for more information.

When using mpich, some sites have issues with orphaned MPI child processes remaining on the system after the master MPI process has been terminated. To address this, TORQUE epilogue scripts can be created which properly clean up
the orphaned processes. Some sample scripts have been contributed and are accessible via the links in the table below.

<table>
<thead>
<tr>
<th>Contributing Site</th>
<th>Description</th>
<th>Soton Extracts of Perl Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1.3 MPICH-VMI</td>
<td>MPICH-VMI is a highly-optimized open-source message passing layer available from NCSA. Additional information can be found in the VMI tutorial.</td>
<td></td>
</tr>
<tr>
<td>7.1.4 LAM-MPI</td>
<td>(under construction)</td>
<td></td>
</tr>
</tbody>
</table>
8 Monitoring Resources

8.1 Resource Overview

A primary task of any resource manager is to monitor the state, health, configuration, and utilization of managed resources. TORQUE is specifically designed to monitor compute hosts for use in a batch environment. TORQUE is not designed to monitor non-compute host resources such as software licenses, networks, filesystems, etc, although these resources can be integrated into the cluster using some scheduling systems.

With regards to monitoring compute nodes, TORQUE reports about a number of attributes broken into 3 major categories, configuration, utilization, and state.

8.1.1 Configuration

Configuration includes both detected hardware configuration, and specified batch attributes.

8.1.2 Utilization

Utilization includes information regarding the amount of node resources currently available (in use) as well as information about who or what is consuming it.

8.1.3 State

State information includes administrative status, general node health information, and general usage status
9 Accounting

9.1 Accounting Records

TORQUE maintains accounting records for batch jobs in the directory

$TORQUEROOT/server_priv/accounting/<TIMESTAMP>

Where $TORQUEROOT defaults to /usr/spool/PBS and <TIMESTAMP> is in the form YYYYMMDD. These records include events, timestamps, and information on resources requested and utilized.

Records for four different event types are produced and are described in the table below.

A sample record in this file may look like the following:

06/06/2005 14:04:25;D;408.ign1.zeta2.org:requestor=guest@ign1.zeta2.org
06/06/2005 14:04:35;Q;409.ign1.zeta2.org;queue=batch
06/06/2005 14:04:44;Q;410.ign1.zeta2.org;queue=batch
06/06/2005 14:06:06;S;407.ign1.zeta2.org;user=guest group=guest jobname=STDIN queue=batch ctime=1118087915 qtime=1118087915 etime=1118087915 start=1118088366 exec_host=ign1.zeta2.org/0 Resource_List.neednodes=ign1.zeta2.org Resource_List.nodect=1 Resource_List.nodes=1 Resource_List.walltime=00:16:40
06/06/2005 14:07:17;D;407.ign1.zeta2.org:requestor=guest@ign1.zeta2.org
06/06/2005 14:07:17;E;407.ign1.zeta2.org;user=guest group=guest jobname=STDIN queue=batch ctime=1118087915 qtime=1118087915 etime=1118087915 start=1118088366 exec_host=ign1.zeta2.org/0 Resource_List.nodect=1 Resource_List.nodes=1 Resource_List.walltime=00:16:40 session=6365 end=1118088437 Exit_status=271 resources_used.cput=00:00:00 resources_used.mem=3068kb resources_used.vmem=16080kb resources_used.walltime=00:01:11