TORQUE Resource Manager
Quick Start Guide Version

High Performance Computing Center

Ferdowsi University of Mashhad

http://www.um.ac.ir/hpcc

Jan. 2006

Contents

1 Introduction

TORQUE[1] 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 to improve overall utilization, scheduling and administration on a cluster.

In this document[2] we try to give an easy-to-use guide for installing and configuring TORQUE resource manager for your cluster. For more detailed information about TORQUE, you can refer to detailed version of this document.

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

[1]Most of material used in this document have gotten from www.clusterresources.org

[2]This document is generated by LaTeX

  - More human readable logging (ie no more 'error 15038 on command 42')

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 the TAR-ball package 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. To extract the install the package, 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 further references to TORQUE commands. The default installation directories for the binaries are either `/usr/local/bin` and `/usr/local/sbin`. To add the TORQUE directory to the `PATH` environment variable, do the following:

```
$ export PATH=/usr/local/bin;/usr/local/sbin;$PATH
```

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 `RPM`. 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 for debain distribution.

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.

```
$ cp torque-package-mom-linux-i686.sh /shared/storage
$ cp torque-package-clients-linux-i686.sh /shared/storage
$ dsh /shared/storage/torque-package-mom-linux-i686.sh --install
$ dsh /shared/storage/torque-package-clients-linux-i686.sh --install
```

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. This can save your time during running your cluster (you may need to reboot the whole cluster).

- 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:

```
$ qmgr

Qmgr: set server managers += josh@*.fsc.com
Qmgr: set server operators += josh@*.fsc.com
```

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.

```
# 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 is not running yet. TORQUE can use its native scheduler by running `pbs_sched` or an advanced scheduler can be used. To integrating Schedulers with TORQUE, see the detailed version of this document.

Appendix A

```
# Source function library.
if [ -f /etc/init.d/functions ] ; then
  . /etc/init.d/functions
elif [ -f /etc/rc.d/init.d/functions ] ; then
  . /etc/rc.d/init.d/functions
else
  exit 0
fi

# Read in the command arguments
case "$1" in
  start)
# Start TORQUE services first...

echo -n $"Starting TORQUE services: "
daemon /usr/local/torque/sbin/pbs_mom
daemon /usr/local/torque/sbin/pbs_server
echo

# Next start Moab scheduler...
echo -n $"Starting Moab scheduler: "
daemon /usr/local/moab/sbin/moab
echo
;;
  stop)
# Stop Moab first...

echo -n $"Shutting down Moab scheduler: "
killproc moab
echo

echo -n $"Shutting down TORQUE services: "
killproc pbs_server
echo
killproc pbs_mom
echo
;;
  restart)
$0 stop
$0 start
;;
  *)
echo $"Usage: moab {start|stop|restart}"
exit 1
```

```
esac

exit 0
```