Adaptive COMPUTING

Unify Context > Apply Policy > Act Intelligently

TORQUE

Administrator's Guide

Version 3.0.0

Adaptive COMPUTING

CLUSTER RESOURCES

Moab
Unified Intelligent Automation

TORQUE Admin Manual

version 3.0.0

Legal Notices

Copyright

Trademarks

Acknowledgments

TORQUE includes software developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and Veridian Information Solutions, Inc. Visit www.OpenPBS.org for OpenPBS software support, products and information. TORQUE is neither endorsed by nor affiliated with Altair Grid Solutions, Inc.

TORQUE Administrator Guide Overview

Advanced TORQUE Administration is a video tutorial of a session offered at Moab Con that offers further details on advanced TORQUE administration.

This collection of documentation for TORQUE resource manager is intended as a reference for both users and system administrators.

The 1.0 Overview section provides the details for installation and initialization, advanced configuration options, and (optional) qmgr options necessary to get the system up and running. System Testing is also covered.

The 2.0 Submitting and Managing Jobs section covers different actions applicable to jobs. The first section, 2.1 Job Submission, details how to submit a job and request resources (nodes, software licenses, and so forth) and provides several examples. Other actions include monitoring, canceling, preemption, and keeping completed jobs.

The 3.0 Managing Nodes section covers administrator tasks relating to nodes, which includes the following: adding nodes, changing node properties, and identifying state. Also an explanation of how to configure restricted user access to nodes is covered in section 3.4 Host Security.

The 4.0 Setting Server Policies section details server side configurations of queue and high availability.

The 5.0 Interfacing with a Scheduler section offers information about using the native scheduler versus an advanced scheduler.

The 6.0 Configuring Data Management section deals with issues of data management. For non-network file systems, the SCP/RCP Setup section details setting up SSH keys and nodes to automate transferring data. The NFS and Other Networked File Systems section covers configuration for these file systems. This chapter also addresses the use of File Stage-In/Stage-Out using the **stagein** and **stageout** directives of the qsub command.

The 7.0 Interfacing with Message Passing section offers details supporting MPI (Message Passing Interface).

The 8.0 Managing Resources section covers configuration, utilization, and states of resources.

The 9.0 Accounting section explains how jobs are tracked by TORQUE for accounting purposes.

The 10.0 Troubleshooting section is a troubleshooting guide that offers help with general problems; it includes an FAQ (Frequently Asked Questions) list and instructions for how to set up and use compute node checks and how to debug TORQUE.

The numerous appendices provide tables of commands, parameters, configuration options, error codes, the Quick Start Guide, and so forth.

- A. Commands Overview
- B. Server Parameters
- C. MOM Configuration
- D. Error Codes and Diagnostics
- E. Considerations Before Upgrading
- F. Large Cluster Considerations
- G. Prologue and Epilogue Scripts
- H. Running Multiple TORQUE Servers and Moms on the Same Node
- I. Security Overview
- J. Submit Filter (aka **qsub** Wrapper)
- K. torque.cfg File

Introduction

What is a Resource Manager?

While TORQUE has a built-in scheduler, **pbs_sched**, it is typically used solely as a *resource manager* with a scheduler making requests to it. Resources managers provide the low-level functionality to start, hold, cancel, and monitor jobs. Without these capabilities, a scheduler alone can not control jobs.

What are Batch Systems?

While TORQUE is flexible enough to handle scheduling a conference room, it is primarily used in batch systems. Batch systems are a collection of computers and other resources (networks, storage systems, license servers, and so forth) that operate under the notion that the whole is greater than the sum of the parts. Some batch systems consist of just a handful of machines running single-processor jobs, minimally managed by the users themselves. Other systems have thousands and thousands of machines executing users' jobs simultaneously while tracking software licenses and access to hardware equipment and storage systems.

Pooling resources in a batch system typically reduces technical administration of resources while offering a uniform view to users. Once configured properly, batch systems abstract away many of the details involved with running and managing jobs, allowing higher resource utilization. For example, users typically only need to specify the minimal constraints of a job and do not need to know the individual machine names of each host on which they are running. With this uniform abstracted view, batch systems can execute thousands and thousands of jobs simultaneously.

Batch systems are comprised of four different components: (1) Master Node, (2) Submit/Interactive Nodes, (3) Compute Nodes, and (4) Resources.

- **Master Node** - A batch system will have a master node where **pbs_server** runs. Depending on the needs of the systems, a master node may be dedicated to this task, or it may fulfill the roles of other components as well.

- **Submit/Interactive Nodes** - Submit or interactive nodes provide an entry point to the system for users to manage their workload. For these nodes, users are able to submit and track their jobs. Additionally, some sites have one or more nodes reserved for interactive use, such as testing and troubleshooting environment problems. These nodes have client commands (such as **qsub** and **qhold**).

- **Compute Nodes** - Compute nodes are the workhorses of the system. Their role is to execute submitted jobs. On each compute node, **pbs_mom** runs to start, kill, and manage submitted jobs. It communicates with **pbs_server** on the master node. Depending on the needs of the systems, a compute node may double as the master node (or more).

- **Resources** - Some systems are organized for the express purpose of managing a collection of resources beyond compute nodes. Resources can include high-speed networks, storage systems, license managers, and so forth. Availability of these resources is limited and needs to be managed intelligently to promote fairness and increased utilization.

Basic Job Flow

The life cycle of a job can be divided into four stages: (1) creation, (2) submission, (3) execution, and (4) finalization.

- **Creation** - Typically, a submit script is written to hold all of the parameters of a job. These parameters could include how long a job should run (**walltime**), what resources are necessary to run, and what to execute. The following is an example submit file:

```
#PBS -N localBlast
#PBS -S /bin/sh
#PBS -l nodes=1:ppn=2,walltime=240:00:00
#PBS -M user@my.organization.com
#PBS -m ea

source ~/.bashrc
cd $HOME/work/dir
sh myBlast.sh -i -v
```

This submit script specifies the name of the job (localBlast), what environment to use (`/bin/sh`), that it needs both processors on a single node (`nodes=1:ppn=2`), that it will run for at most 10 days, and that TORQUE should email user@my.organization.com when the job exits or aborts. Additionally, the user specifies where and what to execute.

- **Submission** - A job is submitted with the **qsub** command. Once submitted, the policies set by the administration and technical staff of the site dictate the priority of the job and therefore, when it will start executing.

- **Execution** - Jobs often spend most of their lifecycle executing. While a job is running, its status can be queried with **qstat**.

- **Finalization** - When a job completes, by default, the `stdout` and `stderr` files are copied to the directory where the job was submitted.

Glossary

The following are a few terms that appear in the documentation:

Epilogue

An optional script executed after a job completes. epilogue.user, epilogue.parallel and epilogue.precancel scripts also exist. See Appendix G: Prologue and Epilogue Scripts for more information.

Prologue

An optional script executed before a job starts. prologue.user and prologue.parallel scripts also exist. See Appendix G: Prologue and Epilogue Scripts for more information.

$TORQUE_HOME

The base directory for configuration directories. Defaults to /var/spool/torque (starting with version 2.1. Previously defaulted to /usr/spool/PBS)

1.1 TORQUE Installation

1.1.1 TORQUE Architecture

A TORQUE cluster consists of one head node and many compute nodes. The head node 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 not running pbs_server or pbs_mom).

The head node also runs 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, is provided in the TORQUE source distribution. Most TORQUE users choose to use a packaged, advanced scheduler such as Maui or Moab.

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

1.1.2 Installing TORQUE

Build the distribution on the machine that will act as the TORQUE server - the machine which monitors and controls all compute nodes by running the pbs_server daemon.

The built distribution package works only on compute nodes of a similar architecture. Nodes with different architecture must have the installation package built on them individually.

1. Download the TORQUE distribution file from http://clusterresources.com/downloads/torque. Source code can also be downloaded using Subversion from the repository at svn://clusterresources.com/torque/. Use the command `svn list svn://clusterresources.com/torque/` to display all branches.
2. Extract the packaged file and navigate to the unpackaged directory.

```
> tar -xzvf torque-2.3.4.tar.gz
> cd torque-2.3.4/
```

3. Configure the package.

   By default, **make install** installs all files in `/usr/local/bin`, `/usr/local/lib`, `/usr/local/sbin`, `/usr/local/include`, and `/usr/local/man` . You can specify an installation prefix other than `/usr/local` using `--prefix` as an argument to **./configure**, for example:

```
./configure --prefix=$HOME
```

   Verify you have environment variables configured so your system can find the shared libraries and binary files for TORQUE.

   To set the library path, add the directory where the TORQUE libraries will be installed. For example, if your TORQUE libraries are installed in `/opt/torque/lib`, execute the following:

```
> set LD_LIBRARY_PATH=$(LD_LIBRARY_PATH):/opt/torque/lib
> ldconfig
```

Cluster Resources recommends that the TORQUE administrator be root.

For information on customizing the build at configure time, see the configure options list.

```
> ./configure
```

4. Run **make** and **make install**.

TORQUE must be installed by a root user.

```
> make
> sudo make install
```

OSX 10.4 users need to change **#define __TDARWIN** in src/include/pbs_config.h to **#define __TDARWIN_8**.

After installation, verify you have the **PATH** environment variable configured to include/usr/local/bin/ and /usr/local/sbin/.

By default, **make install** creates a directory at /var/spool/torque. This directory is referred to as TORQUE_HOME. TORQUE_HOME has several sub-directories, including server_priv/, server_logs/, mom_priv/, mom_logs/, and other directories used in the configuration and running of TORQUE.

TORQUE 2.0p2 and later includes a torque.spec file for building your own RPMs. You can also use the checkinstall program to create your own RPM, tgz, or deb package.

While Adaptive Computing distributes the RPM files as part of the build, it does not support those files. Not every Linux distribution uses RPM. Adaptive Computing provides a single solution using **make** and **make install** that works across all Linux distributions and most UNIX systems. We recognize the RPM format provides many advantages for deployment but it is up to the indiviual site to repackage the TORQUE installation to match their individual needs.

1.1.3 Compute Nodes

Use the Cluster Resources tpackage system to create self-extracting tarballs which can be distributed and installed on compute nodes. The tpackages are customizable. See the INSTALL file for additional options and features.

To create tpackages

1. Configure and make as normal, and then run **make packages**.

```
> 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.
```

2. Copy the desired packages to a shared location.

```
> cp torque-package-mom-linux-i686.sh /shared/storage/
> cp torque-package-clients-linux-i686.sh /shared/storage/
```

3. Install the tpackages on the compute nodes.

Cluster Resources recommends that you use a remote shell, such as SSH, to install tpackages on remote systems. Set up shared SSH keys if you do not want to supply a password for each host.

The only required package for the compute nodes is mom-linux. Additional packages are recommended so you can use client commands and submit jobs from compute nodes.

The following is an example on how to copy and install mom-linux in a distributed fashion:

```
> for i in node01 node02 node03 node04 ; do scp torque-package-mom-
linux-i686.sh ${i}:/tmp/. ; done
>  for i in node01 node02 node03 node04 ; do scp torque-package-
clients-linux-i686.sh ${i}:/tmp/. ; done
> for i in node01 node02 node03 node04 ; do ssh ${i} /tmp/torque-
package-mom-linux-i686.sh --install ; done
> for i in node01 node02 node03 node04 ; do ssh ${i} /tmp/torque-
package-clients-linux-i686.sh --install ; done
```

Alternatively, you can use a tool like xCAT instead of dsh.

1. Copy the tpackage to the nodes.

   ```
   > prcp torque-package-linux-i686.sh noderange:/destinationdirectory/
   ```

2. Install the tpackage.

   ```
   > psh noderange /tmp/torque-package-linux-i686.sh --install
   ```

Alternatively, users with RPM-based Linux distributions can build RPMs from the source tarball in two ways.

- To use the default settings, use the **rpmbuild** command.

  ```
  >  rpmbuild -ta torque-2.3.4.tar.gz
  ```

- If configure options are required, untar and build as normal, and then use the **make rpms** command instead.

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

1.1.4 Enabling TORQUE as a service (optional)

The method for enabling TORQUE as a service is dependent on the Linux variant you are using. Startup scripts are provided in the `contrib/init.d/` directory of the source package.

- **Red Hat (as root)**

  ```
  > cp contrib/init.d/pbs_mom /etc/init.d/pbs_mom
  > chkconfig --add pbs_mom
  ```

- **SuSE (as root)**

  ```
  > cp contrib/init.d/suse.pbs_mom /etc/init.d/pbs_mom
  > insserv -d pbs_mom
  ```

- **Debian (as root)**

  ```
  > cp contrib/init.d/debian.pbs_mom /etc/init.d/pbs_mom
  > update-rc.d pbs_mom defaults
  ```

These options can be added to the self-extracting packages. For more details, see the `INSTALL` file.

See Also

- TORQUE Installation TroubleShooting

1.2 Initialize/Configure TORQUE on the Server (pbs_server)

The directory TORQUE_HOME/server_priv/ contains configuration and other information needed for **pbs_server**. One of the files in this directory is serverdb. The serverdb file contains configuration parameters for **pbs_server** and its queues. For **pbs_server** to run, serverdb must be initialized.

You can initialize serverdb in two different ways, but the recommended way is to use the ./torque.setup script:

1. Execute ./torque.setup from the build directory.
2. **pbs_server** -t create

Restart **pbs_server** after initializing serverdb.

```
> qterm
> pbs_server
```

1.2.0.1 ./torque.setup

The torque.setup script uses pbs_server -t create to initialize serverdb and then adds a user as a manager and operator of TORQUE and other commonly used attributes. The syntax is as follows:

- ./torque.setup *<username>*

```
> ./torque.setup ken
> qmgr -c 'p s'

#
# Create queues and set their attributes.
#
#
# Create and define queue batch
#
create queue batch
set queue batch queue_type = Execution
set queue batch resources_default.nodes = 1
set queue batch resources_default.walltime = 01:00:00
set queue batch enabled = True
set queue batch started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = kmn
set server managers = ken@kmn
set server operators = ken@kmn
set server default_queue = batch
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 6
set server mom_job_sync = True
```

1.2.0.2 pbs_server -t create

The -t create option instructs **pbs_server** to create the serverdb file and initialize it with a minimum configuration to run **pbs_server**. To see the configuration, use qmgr:

```
> pbs_server -t create
> qmgr -c 'p s'

#
# Set server attributes.
#
set server acl_hosts = kmn
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 6
```

A single queue named batch and a few needed server attribues are created.

1.2.1 Specify Compute Nodes

The environment variable *TORQUE_HOME* is where configuration files are stored. For TORQUE 2.1 and later, *TORQUE_HOME* is `/var/spool/torque/`. For earlier versions, *TORQUE_HOME* is `/usr/spool/PBS/`.

The **pbs_server** must recognize which systems on the network are its compute nodes. Specify each node on a line in the server's nodes file. This file is located at *TORQUE_HOME*`/server_priv/nodes`. In most cases, it is sufficient to specify just the names of the nodes on individual lines; however, various properties can be applied to each node.

Syntax of nodes file:

```
node-name[:ts] [np=] [gpus=] [properties]
```

The **[:ts]** option marks the node as timeshared. Timeshared nodes are listed by the server in the node status report, but the server does not allocate jobs to them.

The **[np=]** option specifies the number of virtual processors for a given node. The value can be less than, equal to, or greater than the number of physical processors on any given node.

The **[gpus=]** option specifies the number of GPUs for a given node. The value can be less than, equal to, or greater than the number of physical GPUs on any given node.

The node processor count can be automatically detected by the TORQUE server if **auto_node_np** is set to TRUE. This can be set using the command `qmgr -c set server auto_node_np = True`. Setting auto_node_np to TRUE overwrites the value of np set in `TORQUE_HOME/server_priv/nodes`.

The **[properties]** option allows you to specify arbitrary strings to identify the node. Property strings are alphanumeric characters only and must begin with an alphabetic character.

Comment lines are allowed in the nodes file if the first non-white space character is the pound sign (**#**).

The following example shows a possible node file listing.

`TORQUE_HOME/server_priv/nodes`:

```
# Nodes 001 and 003-005 are cluster nodes
#
node001 np=2 cluster01 rackNumber22
#
# node002 will be replaced soon
node002:ts waitingToBeReplaced
# node002 will be replaced soon
#
node003 np=4 cluster01 rackNumber24
node004  cluster01 rackNumber25
node005 np=2 cluster01 rackNumber26 RAM16GB
node006
node007 np=2
```

```
node008:ts np=4
...
```

1.2.2 Configure TORQUE on the Compute Nodes

If using TORQUE self extracting packages with default compute node configuration, no additional steps are required and you can skip this section.

If installing manually, or advanced compute node configuration is needed, edit the *TORQUE_HOME*/mom_priv/config file on each node. The recommended settings follow.

TORQUE_HOME/mom_priv/config:

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

This file is identical for all compute nodes and can be created on the head node and distributed in parallel to all systems.

1.2.3 Finalize Configurations

After configuring the serverdb and the server_priv/nodes files, and after ensuring minimal MOM configuration, restart the pbs_server on the server node and the pbs_mom on the compute nodes.

Compute Nodes:

```
> pbs_mom
```

Server Node:

```
> qterm -t quick
> pbs_server
```

After waiting several seconds, the pbsnodes -a command should list all nodes in state free.

See Also

- MOM Configuration Overview
- Advanced Configuration

1.3 Advanced Configuration

1.3.1 Customizing the Install

The TORQUE **configure** command has several options available. Listed below are some suggested options to use when running **./configure**.

- By default, TORQUE does not install the admin manuals. To enable this, use `--enable-docs`
- By default, TORQUE does not enable **syslog** usage. To enable this, use `--enable-syslog`

Full Configure Options List

Optional features:

Option	Description
--disable-clients	directs torque not to build and install the TORQUE client utilities such as qsub, qstat, qdel, etc.
--disable-dependency-tracking	directs TORQUE build system to only rebuild changed source files and not rebuild dependent files.
--disable-FEATURE	do not include FEATURE (same as --enable-FEATURE=no).
--disable-gcc-warnings	Disable gcc strictness and warnings. If using gcc, default is to error on any warning.
--disable-gui	do not include the GUI-clients.
--disable-libtool-lock	avoid locking (might break parallel builds).
--disable-mom	do not include the MOM daemon.
--disable-mom-checkspool	Don't check free space on spool directory and set an error.
--disable-posixmemlock	disable the moms use of mlockall. Some versions of OSs seem to have buggy POSIX MEMLOCK.
--disable-privports	disable the use of privileged ports for authentication. Some versions of OSX have a buggy bind() and cannot bind to privileged ports.
--disable-qsub-keep-override	do not allow the qsub -k flag to override -o -e.
--disable-rpp	By default, TORQUE uses RPP/UDP for resource queries from the PBS server to the MOMs. If disabled, TCP is used. This does not affect general node/job status messages, job launch/exit messages, inter-mom messages, etc.
--disable-server	do not include server and scheduler.
--disable-shell-pipe	give the job script file as standard input to the shell instead of passing its name via a pipe.
--disable-spool	if disabled, TORQUE will create output and error files directly in $HOME/.pbs_spool if it exists or in $HOME otherwise. By default, TORQUE will spool files in TORQUE_HOME/spool and copy them to the users home directory when the job completes.
--disable-	With HPUX and GCC, don't force usage of XOPEN and libxnet.

xopen-networking	
--enable-acct-x	enable adding x attributes to accounting log.
--enable-array	setting this under IRIX enables the SGI Origin 2000 parallel support. Normally autodetected from the /etc/config/array file.
--enable-autorun	allows jobs to run automatically as soon as they are queued if resources are available (available in TORQUE 2.3.1 and later).
--enable-blcr	enable BLCR support.
--enable-cpa	enable Cray's CPA support.
--enable-cpuset	enable Linux 2.6 kernel cpusets (in development).
--enable-debug	turn on the compilation of DEBUG code.
--enable-dependency-tracking	do not reject slow dependency extractors.
--enable-drmaa	build the DRMAA 1.0 library.
--enable-fast-install[=PKGS]	optimize for fast installation [default=yes].
--enable-FEATURE[=ARG]	include FEATURE [ARG=yes].
--enable-filesync	Open files with sync on each write operation. This has a negative impact on TORQUE performance. This is disabled by default.
--enable-force-nodefile	forces creation of nodefile regardless of job submission parameters. Not on by default.
--enable-geometry-requests	TORQUE is compiled to use procs_bitmap during job submission.
--enable-high-availability	enables enhanced high availability (high availability is enabled by default, but this option enables the enhanced version)
--enable-maintainer-mode	this is for the autoconf utility and tells autoconf to enable so called rebuild rules. See maintainer mode for more information.
--enable-maxdefault	turn on the RESOURCEMAXDEFAULT flag.  Versions of TORQUE earlier than 2.4.5 attempted to apply queue and server defaults to a job that didn't have defaults specified. If a setting still did not have a value after that, TORQUE applied the queue and server maximum values to a job (meaning, the maximum values for an applicable setting were applied to jobs that had no specified or default value). In TORQUE 2.4.5 and later, the queue and server maximum values are no longer used as a value for missing settings. To reenable this behavior in TORQUE 2.4.5 and later, use `--enable-maxdefault`.

--enable-nochildsignal	turn on the NO_SIGCHLD flag.
--enable-nodemask	enable nodemask-based scheduling on the Origin 2000.
--enable-pemask	enable pemask-based scheduling on the Cray T3e.
--enable-plock-daemons[=ARG]	enable daemons to lock themselves into memory: logical-or of 1 for pbs_server, 2 for pbs_scheduler, 4 for pbs_mom (no argument means 7 for all three).
--enable-quickcommit	turn on the QUICKCOMMIT flag.
--enable-shared[=PKGS]	build shared libraries [default=yes].
--enable-shell-use-argv	enable this to put the job script name on the command line that invokes the shell. Not on by default. Ignores --enable-shell-pipe setting.
--enable-sp2	build PBS for an IBM SP2.
--enable-srfs	enable support for SRFS on Cray.
--enable-static[=PKGS]	build static libraries [default=yes].
--enable-syslog	enable (default) the use of syslog for error reporting.
--enable-tcl-qstat	setting this builds qstat with Tcl interpreter features. This is enabled if Tcl is enabled.
--enable-unixsockets	enable the use of Unix Domain sockets for authentication.

Optional Packages:

Option	Description
--with-cpa-include=DIR	include path for cpalib.h.
--with-cpa-lib=DIR	lib path for libcpalib.
--with-debug	compile with debugging symbols.
--with-default-server=HOSTNAME	set the name of the computer that clients will access when no machine name is specified as part of the queue name. It defaults to the hostname of the machine on which PBS is being compiled.
--with-environ=PATH	set the path containing the environment variables for the daemons. For SP2 and AIX systems, suggested setting is to /etc/environment. Defaults to the file "pbs_environment" in the server-home. Relative paths are interpreted within the context of the server-home.
--with-gnu-ld	assume the C compiler uses GNU ld [default=no].
--with-maildomain=MAILDOMAIN	override the default domain for outgoing mail messages, i.e. "user@maildomain". The default maildomain is the hostname where the job was submitted from.
--with-modulefiles[=DIR]	use module files in specified directory [/etc/modulefiles].

--with-momlogdir	use this directory for MOM logs.
--with-momlogsuffix	use this suffix for MOM logs.
--without-PACKAGE	do not use PACKAGE (same as --with-PACKAGE=no).
--without-readline	do not include readline support (default: included if found).
--with-PACKAGE[=ARG]	use PACKAGE [ARG=yes].
--with-pam=DIR	Directory that holds the system PAM modules. Defaults to /lib(64)/security on Linux.
--with-pic	try to use only PIC/non-PIC objects [default=use both].
--with-qstatrc-file=FILE	set the name of the file that qstat will use if there is no ".qstatrc" file in the directory where it is being invoked. Relative path names will be evaluated relative to the server home directory (see above). If this option is not specified, the default name for this file will be set to "qstatrc" (no dot) in the server home directory.
--with-rcp	one of "scp", "rcp", "mom_rcp", or the fullpath of a remote file copy program. scp is the default if found, otherwise mom_rcp is used. Some rcp programs don't always exit with valid error codes in case of failure. mom_rcp is a copy of BSD rcp included with this source that has correct error codes, but it is also old, unmaintained, and doesn't have largefile support.
--with-sched=TYPE	sets the scheduler type. If TYPE is "c", the scheduler will be written in C. If TYPE is "tcl" the server will use a Tcl based scheduler. If TYPE is "basl", TORQUE will use the rule based scheduler. If TYPE is "no", then no scheduling is done. "c" is the default.
--with-sched-code=PATH	sets the name of the scheduler to use. This only applies to BASL schedulers and those written in the C language. For C schedulers this should be a directory name and for BASL schedulers a filename ending in ".basl". It will be interpreted relative to srctree/src/schedulers.SCHD_TYPE/samples. As an example, an appropriate BASL scheduler realtive path would be "nas.basl". The default scheduler code for "C" schedulers is "fifo".
--with-scp	In TORQUE 2.1 and later, SCP is the default remote copy protocol. See --with-rcp if a different protocol is desired.
--with-sendmail[=FILE]	sendmail executable to use.
--with-server-home=DIR	set the server home/spool directory for PBS use. Defaults to /var/spool/torque.
--with-server-name-file=FILE	set the file that will contain the name of the default server for clients to use. If this is not an absolute pathname, it will be evaluated relative to the server home directory that either defaults to /usr/spool/PBS or is set using the --with-server-home option to configure. If this option is not specified, the default name for this file will be set to "server_name".
--with-tags[=TAGS]	include additional configurations [automatic].
--with-tcl	directory containing tcl configuration (tclConfig.sh).
--with-tclatrsep=CHAR	set the Tcl attribute separator character this will default to "." if unspecified.
--with-tclinclude	directory containing the public Tcl header files.
--with-tclx	directory containing tclx configuration (tclxConfig.sh).

--with-tk	directory containing tk configuration (tkConfig.sh).
--with-tkinclude	directory containing the public Tk header files.
--with-tkx	directory containing tkx configuration (tkxConfig.sh).
--with-tmpdir=DIR	set the tmp directory that pbs_mom will use. Defaults to "/tmp". This is a Cray-specific feature.
--with-xauth=PATH	specify path to xauth program.

1.3.2 Server Configuration

1.3.2.1 Server Configuration 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.

1.3.2.2 Name Service Configuration

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.

1.3.2.3 Configuring Job Submission Hosts

Using RCmd Authentication

When jobs can be submitted from several different hosts, these hosts should be trusted via the R* commands (such as rsh and rcp). 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` file, as in the following:

`/etc/hosts.equiv`:

```
#[+ | -] [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.

Using the `submit_hosts` Server Parameter

Trusted submit host access may be directly specified without using RCmd authentication by setting the server submit_hosts parameter via qmgr as in the following example:

```
> qmgr -c 'set server submit_hosts = host1'
> qmgr -c 'set server submit_hosts += host2'
> qmgr -c 'set server submit_hosts += host3'
```

Use of **submit_hosts** is potentially subject to DNS spoofing and should not be used outside of controlled and trusted environments.

Allowing Job Submission from Compute Hosts

If preferred, all compute nodes can be enabled as job submit hosts without setting `.rhosts` or `hosts.equiv` by setting the allow_node_submit parameter to **true**.

1.3.2.4 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.

1.3.2.5 Architecture Specific Notes

1.3.2.5.1 Mac OS/X Specific Notes

With some versions of Mac OS/X, it is required to add the line `$restricted *.<DOMAIN>` to the **pbs_mom** configuration file. This is required to work around some socket bind bugs in the OS.

1.3.2.6 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 following example:

```
> 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.

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

1.3.2.7 Setting Up E-mail

Moab relies on e-mails from TORQUE about job events. To set up e-mail, do the following:

1. Use the `--with-sendmail` configure option at configure time. TORQUE needs to know where the email application is. If this option is not used, TORQUE tries to find the sendmail executable. If it isn't found, TORQUE cannot send e-mails.

   ```
   > ./configure --with-sendmail=<path_to_executable>
   ```

2. Set `mail_domain` in your server settings. If your domain is clusterresources.com, execute:

   ```
   > qmgr -c 'set server mail_domain=clusterresources.com'
   ```

3. (Optional) You can override the default mail_body_fmt and mail_subject_fmt values via qmgr:

   ```
   > qmgr -c 'set server mail_body_fmt=Job: %i \n Name: %j \n On host: %h
   \n \n %m \n \n %d'
   > qmgr -c 'set server mail_subject_fmt=Job %i - %r'
   ```

By default, users receive e-mails on job aborts. Each user can select which kind of e-mails to receive by using the qsub -m option when submitting the job. If you want to dictate when each user should receive e-mails, use a submit filter.

1.3.2.8 Using MUNGE Authentication

MUNGE is an authentication service that creates and validates user credentials. It was developed by Lawrence Livermore National Laboratoy (LLNL) to be highly scalable so it can be used in large environments such as

HPC clusters. To learn more about MUNGE and how to install it, see http://code.google.com/p/munge/

Configuring TORQUE to use MUNGE is a compile time operation. When you are building TORQUE use `–enable-munge-auth` as a command line option with `./configure`.

```
> ./configure –enable-munge-auth
```

You can use only one authorization method at a time. If `–enable-munge-auth` is configured, the privileged port ruserok method is disabled.

TORQUE does not link any part of the MUNGE library into its executables. It calls the MUNGE and UNMUNGE utilities which are part of the MUNGE daemon. The MUNGE daemon must be running on the server and all submission hosts. The TORQUE client utilities call MUNGE and then deliver the encrypted credential to pbs_server where the credential is then unmunged and the server verifies the user and host against the authorized users configured in serverdb.

Authorized users are added to serverdb using *qmgr* and the *authorized_users* server parameter. The syntax for *authorized_users* is *authorized_users=<user>@<host>*. To add an authorized user to the server you can use the following qmgr command:

```
> qmgr -c 'set server authorized_users=user1@hosta
> qmgr -c 'set server authorized_users+=user2@hosta
```

The previous example adds user1 and user2 from hosta to the list of authorized users on the server. Users can be removed from the list of authorized users by using the **-=** syntax as follows:

```
> qmgr  -c 'set server authorized_users-=user1@hosta
```

Users must be added with the <user>@<host> syntax. The user and the host portion can use the '*' wildcard to allow multiple names to be accepted with a single entry. A range of user or host names can be specified using a [a-b] syntax where *a* is the beginning of the range and *b* is the end.

```
> qmgr  -c 'set server authorized_users=user[1-10]@hosta
```

This allows user1 through user10 on hosta to run client commands on the server.

See Also

- Appendix B: Server Parameters

1.4 Manual Setup of Initial Server Configuration

Configuration of the pbs_server daemon is accomplished using the **qmgr** command. On a new system, the configuration database must be initialized using the command **pbs_server -t create**. Once this is done, the minimal configuration requires setting up the desired queue structure and enabling the scheduling interface.

The following example shows a simple one-queue configuration:

```
pbs_server -t create
qmgr -c "set server scheduling=true"
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"
qmgr -c "set server default_queue=batch"
```

These commands need to be executed by root.

In this example, the configuration database is initialized and the scheduling interface is activated (using 'scheduling=true'). This interface allows the scheduler to receive job and node events which allow it to be more responsive. The next step creates a queue and specifies the queue type. Within PBS, the queue must be declared an 'execution queue in order for it to run jobs. Additional configuration (i.e., setting the queue to started and enabled) allows the queue to *accept* job submissions, and *launch* queued jobs.

The next two lines are optional, setting default node and walltime attributes for a submitted job. These defaults will be picked up by a job if values are not explicitly set by the submitting user. The final line, default_queue=batch, is also a convenience line and indicates that a job should be placed in the batch queue unless explicitly assigned to another queue.

Additional information on configuration can be found in the admin manual and in the qmgr man page.

1.5 Server Node File Configuration

1.5.1 Basic Node Specification

For the **pbs_server** to communicate with each of the moms, it needs to know which machines to contact. Each node that 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 TORQUE_HOME/server_priv/nodes. In most cases, it is sufficient to specify just the node name on a line as in the following example:

server_priv/nodes:

```
node001
node002
node003
node004
```

1.5.2 Specifying Virtual Processor Count for a Node

By default each node has one virtual processor. Increase the number using the **np** attribute in the nodes file. The value of np can be equal to the number of physical cores on the node or it can be set to a value which represents available "execution slots" for the node. The value used is determined by the administrator based on hardware, system, and site criteria.

The following example shows how to set the np value in the nodes file. Note that node001 has two physical cores and the administrator wants the value of np to reflect that. However, node002 will be set up to handle multiple virtual processors without regard to the number of physical cores on the system.

server_priv/nodes:

```
node001 np=2
node002 np=12
...
```

1.5.3 Specifying GPU Count for a Node

TORQUE can track the number of GPUs on a node. The number of GPUs a node has is specified by the **gpus** attribute in the nodes file. The value of gpu can be equal to the number of physical GPUs on the node or it can be set to a value which represents available "execution slots" for the node. The value used is determined by the administrator based on hardware, system, and site criteria.

The following example shows how to set the GPU value in the nodes file. Note that node001 has one physical GPU and the administrator wants the value of gpus= to reflect that. However, node002 will be set up to handle multiple virtual GPUs without regard to the number of physical GPUs on the system.

server_priv/nodes:

```
node001 gpus=1
node002 gpus=4
...
```

1.5.4 Specifying Node Features (aka Node Properties)

Node features can be specified by placing one or more white space delimited strings on the line for the associated host as in the following example:

server_priv/nodes:

```
```

```
node001 np=2 fast ia64
node002 np=4 bigmem fast ia64 smp
...
```

These features can be used by users to request specific nodes when submitting jobs.

See Also

- Server Commands
- Moab Node Feature Overview

1.6 Testing Server Configuration

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.

```
# 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 - DO NOT RUN AS ROOT
> su - testuser
> echo "sleep 30" | qsub

# verify jobs display
> qstat
```

At this point, the job should be in the **Q** state and will not run because a scheduler is not running yet. TORQUE can use its native scheduler by running **pbs_sched** or an advanced scheduler (such as Moab Workload Manager). Section 5.1 Integrating Schedulers for TORQUE details setting up an advanced scheduler.

1.7 TORQUE on NUMA Systems

Starting in TORQUE version 3.0, TORQUE can be configured to take full advantage of Non-Uniform Memory Archtecture (NUMA) systems. The following instructions are a result of development on SGI Altix and UV hardware.

1.7.1 TORQUE NUMA Configuration

There are three steps to configure TORQUE to take advantage of NUMA architectures:

1. Configure TORQUE with `--enable-numa-support`.
2. Create the mom_priv/mom.layout file.
3. Configure server_priv/nodes.

1.7.2 Building TORQUE with NUMA Support

To turn on NUMA support for TORQUE the `-enable-numa-support` option must be used during the configure portion of the installation. In addition to any other configuration options, add the `-enable-num-support` option as indicated in the following example:

```
$ ./configure --enable-numa-support
```

1.7.2.1 Creating mom.layout

When TORQUE is enabled to run with NUMA support, there is only a single instance of pbs_mom (MOM) that is run on the system. However, TORQUE will report that there are multiple nodes running in the cluster. While pbs_mom and pbs_server both know there is only one instantiation of pbs_mom, they manage the cluster as if there were multiple separate MOM nodes.

The mom.layout file is a virtual mapping between the system hardware configuration and how the administrator wants TORQUE to view the system. Each line in the mom.layout file equates to a node in the cluster and is referred to as a NUMA node. To properly set up the mom.layout file, it is important to know how the hardware is configured. Use the topology command line utility and inspect the contents of /sys/devices/system/node. The hwloc library can also be used to create a custom discovery tool.

Typing `topology` on the command line of a NUMA system produces something similar to the following:

```
Partition number: 0
6 Blades
72 CPUs
378.43 Gb Memory Total

Blade         ID       asic  NASID         Memory
-------------------------------------------------
    0 r001i01b00  UVHub 1.0      0    67089152 kB
    1 r001i01b01  UVHub 1.0      2    67092480 kB
    2 r001i01b02  UVHub 1.0      4    67092480 kB
    3 r001i01b03  UVHub 1.0      6    67092480 kB
    4 r001i01b04  UVHub 1.0      8    67092480 kB
    5 r001i01b05  UVHub 1.0     10    67092480 kB

CPU      Blade PhysID CoreID APIC-ID Family Model Speed L1(KiB)
L2(KiB) L3(KiB)
-------------------------------------------------------------------
----------------
  0 r001i01b00     00     00       0      6    46  2666 32d/32i
256   18432
  1 r001i01b00     00     02       4      6    46  2666 32d/32i
256   18432
  2 r001i01b00     00     03       6      6    46  2666 32d/32i
```

```
256   18432
  3 r001i01b00     00     08      16      6    46  2666 32d/32i
256   18432
  4 r001i01b00     00     09      18      6    46  2666 32d/32i
256   18432
  5 r001i01b00     00     11      22      6    46  2666 32d/32i
```

From this partial output, note that this system has 72 CPUs on 6 blades. Each blade has 12 CPUs grouped into clusters of 6 CPUs. If the entire content of this command were printed you would see each Blade ID and the CPU ID assigned to each blade.

The topology command shows how the CPUs are distributed, but you likely also need to know where memory is located relative to CPUs, so go to /sys/devices/system/node. If you list the node directory you will see something similar to the following:

```
# ls -al
total 0
drwxr-xr-x 14 root root    0 Dec  3 12:14 .
drwxr-xr-x 14 root root    0 Dec  3 12:13 ..
-r--r--r--  1 root root 4096 Dec  3 14:58 has_cpu
-r--r--r--  1 root root 4096 Dec  3 14:58 has_normal_memory
drwxr-xr-x  2 root root    0 Dec  3 12:14 node0
drwxr-xr-x  2 root root    0 Dec  3 12:14 node1
drwxr-xr-x  2 root root    0 Dec  3 12:14 node10
drwxr-xr-x  2 root root    0 Dec  3 12:14 node11
drwxr-xr-x  2 root root    0 Dec  3 12:14 node2
drwxr-xr-x  2 root root    0 Dec  3 12:14 node3
drwxr-xr-x  2 root root    0 Dec  3 12:14 node4
drwxr-xr-x  2 root root    0 Dec  3 12:14 node5
drwxr-xr-x  2 root root    0 Dec  3 12:14 node6
drwxr-xr-x  2 root root    0 Dec  3 12:14 node7
drwxr-xr-x  2 root root    0 Dec  3 12:14 node8
drwxr-xr-x  2 root root    0 Dec  3 12:14 node9
-r--r--r--  1 root root 4096 Dec  3 14:58 online
-r--r--r--  1 root root 4096 Dec  3 14:58 possible
```

The directory entries node0, node1,...node11 represent groups of memory and CPUs local to each other. These groups are a node board, a grouping of resources that are close together. In most cases, a node board is made up of memory and processor cores. Each bank of memory is called a memory node by the operating system, and there are certain CPUs that can access that memory very rapidly. Note under the directory for node board node0 that there is an entry called `cpulist`. This contains the CPU IDs of all CPUs local to the memory in node board 0.

Now create the mom.layout file. The content of cpulist 0-5 indicating CPUs 0-5 are local to the memory of node board 0. The cpulist for node board 1 shows 6-11 indicating CPUs 6-11 are local to the memory of node board 1. Repeat this for all twelve node boards and create the following mom.layout file for the 72 CPU system.

```
cpus=0-5        mem=0
cpus=6-11       mem=1
cpus=12-17      mem=2
cpus=18-23      mem=3
cpus=24-29      mem=4
cpus=30-35      mem=5
cpus=36-41      mem=6
cpus=42-47      mem=7
cpus=48-53      mem=8
cpus=54-59      mem=9
cpus=60-65      mem=10
cpus=66-71      mem=11
```

- cpus= should be the index of the cpus for that nodeboard or entity, and these are the cpus that will be considered part of that numa node.

- mem= should be the index of the memory nodes that are associated with that node board or entity, and the memory from these will be considered part of that NUMA node.

Each line in the mom.layout file is reported as a node to pbs_server by the pbs_mom daemon.

The mom.layout file does not need to match the hardware layout exactly. It is possible to combine node boards and create larger NUMA nodes. The following example shows how to do this:

```
cpus=0-11       mem=0-1
```

The memory nodes can be combined the same as CPUs. The memory nodes combined must be contiguous. You cannot combine mem 0 and 2.

1.7.2.2 Configuring server_priv/nodes

The pbs_server requires awareness of how the MOM is reporting nodes since there is only one MOM daemon and multiple MOM nodes. So, configure the server_priv/nodes file with the num_numa_nodes and numa_node_str attributes. The attribute num_numa_nodes tells pbs_server how many numa nodes are reported by the MOM. Following is an example of how to configure the nodes file with num_numa_nodes:

```
        numa-10 np=72 num_numa_nodes=12
```

This line in the nodes file tells pbs_server there is a host named numa-10 and that it has 72 processors and 12 nodes. The pbs_server divides the value of np (72) by the value for num_numa_nodes (12) and determines there are 6 CPUs per NUMA node.

In this example, the NUMA system is uniform in its configuration of CPUs per node board, but a system does not need to be configured with the same number of CPUs per node board. For systems with non-uniform CPU distributions, use the attribute numa_node_str to let pbs_server know where CPUs are located in the cluster.

The following is an example of how to configure the server_priv/nodes file for non-uniformly distributed CPUs:

```
        Numa-11 numa_node_str=6,8,12
```

In this configuration, pbs_server knows it has three MOM nodes and the nodes have 6, 8, and 12 CPUs respectively. Note that the attribute np is not used. The np attribute is ignored because the number of CPUs per node is expressly given.

1.7.2.2.1 Enforcement of memory resource limits

TORQUE can better enforce memory limits with the use of the utility **memacctd**. The **memacctd** utility is provided by SGI on SuSe Linux Enterprise Edition (SLES). It is a daemon that caches memory footprints when it is queried. When configured to use the memory monitor, TORQUE queries **memacctd**. It is up to the user to make sure **memacctd** is installed. See the SGI memacctd man page for more information.

To configure TORQUE to use memacctd for memory enforcement do the following:

1. Start **memacctd** as instructed by SGI.
2. Reconfigure TORQUE with -enable-numa-mem-monitor. This will link in the necessary library when TORQUE is recompiled.
3. Recompile and reinstall TORQUE.
4. Restart all MOM nodes.
5. (optional) Alter the qsub filter to include a default memory limit for all jobs that are not submitted with memory limit.

1.8 TORQUE Multi-MOM

Starting in TORQUE version 3.0 users can run multiple MOMs on a single node. The initial reason to develop a multiple MOM capability was for testing purposes. A small cluster can be made to look larger since each MOM instance is treated as a separate node.

When running multiple MOMs on a node each MOM must have its own service and manager ports assigned. The default ports used by the MOM are 15002 and 15003. With the multi-mom alternate ports can be used without the need to change the default ports for pbs_server even when running a single instance of the MOM.

1.8.1 Configuration

There are three steps to setting up multi-mom capability:

1. Configure server_priv/nodes file
2. Edit /etc/hosts file
3. Start pbs_mom with multi-mom options

1.8.1.1 Configure server_priv/nodes

The attributes *mom_service_port* and *mom_manager_port* were added to the nodes file syntax to accommodate multiple MOMs on a single node. By default pbs_mom opens ports 15002 and 15003 for the service and management ports respectively. For multiple MOMs to run on the same IP address they need to have their own port values so they can be distinguished from each other. pbs_server learns about the port addresses of the different MOMs from entries in the server_priv/nodes file. The following is an example of a nodes file configured for multiple MOMs:

```
        hosta   np=2
        hosta-1 np=2 mom_service_port=30001 mom_manager_port=30002
        hosta-2 np=2 mom_service_port=31001 mom_manager_port=31002
        hosta-3 np=2 mom_service_port=32001 mom_manager_port=32002
```

Note that all entries have a unique host name and that all port values are also unique. The entry hosta does not have a mom_service_port or mom_manager_port given. If unspecified, then the MOM defaults to ports 15002 and 15003.

1.8.1.2 /ect/hosts file

Host names in the server_priv/nodes file must be resolvable. Creating an alias for each host enables the server to find the IP address for each MOM; the server uses the port values from the server_priv/nodes file to contact the correct MOM. An example /etc/hosts entry for the previous server_priv/nodes example might look like the following:

```
        192.65.73.10 hosta hosta-1 hosta-2 hosta-3
```

Even though the host name and all the aliases resolve to the same IP address, each MOM instance can still be distinguished from the others because of the unique port value assigned in the server_priv/nodes file.

1.8.1.3 Starting pbs_mom with multi-mom options

To start multiple instances of pbs_mom on the same node, use the following syntax:

```
        pbs_mom -m -M <port value of mom_service_port> -R <port value
of mom_manager_port>
```

Continuing based on the earlier example, if you want to create four MOMs on hosta, type the following at the command line:

```
```

```
# pbs_mom -m -M 30001 -R 30002
# pbs_mom -m -M 31001 -R 31002
# pbs_mom -m -M 32001 -R 32002
# pbs_mom
```

Notice that the last call to pbs_mom uses no arguments. By default pbs_mom opens on ports 15002 and 15003. No arguments are necessary because there are no conflicts.

1.8.2 Stopping pbs_mom in multi-mom mode

Terminate pbs_mom by using the momctl -s command. For any MOM using the default manager port 15003, the **momctl -s** command stops the mom. However, to terminate moms with a manager port value not equal to 15003, you must use the following syntax:

```
momctl -s -p
```

The -p option sends the terminating signal to the MOM manager port and the MOM is terminated.

2.1 Job Submission

Job submission is accomplished using the qsub command, which takes a number of command line arguments and integrates such 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 user's 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 in Configuring Job Submit Hosts.

Versions of TORQUE earlier than 2.4.5 attempted to apply queue and server defaults to a job that didn't have defaults specified. If a setting still did not have a value after that, TORQUE applied the queue and server maximum values to a job (meaning, the maximum values for an applicable setting were applied to jobs that had no specified or default value).

In TORQUE 2.4.5 and later, the queue and server maximum values are no longer used as a value for missing settings.

2.1.1 Multiple Jobs Submission

Sometimes users will want to submit large numbers of jobs based on the same job script. Rather than using a script to repeatedly call qsub, a feature known as job arrays now exists to allow the creation of multiple jobs with one qsub command. Additionally, this feature includes a new job naming convention that allows users to reference the entire set of jobs as a unit, or to reference one particular job from the set.

Job arrays are submitted through the **-t** option to qsub, or by using `#PBS -t` in your batch script. This option takes a comma-separated list consisting of either a single job ID number, or a pair of numbers separated by a dash. Each of these jobs created will use the same script and will be running in a nearly identical environment.

```
> qstat -t 0-4 job_script
1098[].hostname

> qstat
1098[0].hostname ...
1098[1].hostname ...
1098[2].hostname ...
1098[3].hostname ...
1098[4].hostname ...
```

Versions of TORQUE earlier than 2.3 had different semantics for the -t argument. In these versions, -t took a single integer number—a count of the number of jobs to be created.

Each 1098[x] job has an environment variable called PBS_ARRAYID, which is set to the value of the array index of the job, so 1098[0].hostname would have PBS_ARRAYID set to 0. This allows you to create job arrays where each job in the array performs slightly different actions based on the value of this variable, such as performing the same tasks on different input files. One other difference in the environment between jobs in the same array is the value of the PBS_JOBNAME variable.

```
# These two examples are equivalent in TORQUE 2.2
> qsub -t 0-99
> qsub -t 100

# You can also pass comma delimited lists of ids and ranges:
> qsub -t 0,10,20,30,40
> qsub -t 0-50,60,70,80
```

Running qstat displays a job summary, which provides an overview of the array's state. To see each job in the array, run `qstat -t`.

The qalter, qdel, qhold, and qrls commands can operate on arrays—either the entire array or a range of that array. Additionally, any job in the array may be accessed normally by using that job's ID, just as you would with any other job. For example, running the following command would run only the specified job:

```
qrun 1098[0].hostname
```

2.1.2 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 resource extensions may be specified. The native TORQUE resources are listed in the following table:

Resource	Format	Description
arch	string	Specifies the administrator defined system architecture required. This defaults to whatever the **PBS_MACH** string is set to in "local.mk".
cput	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by all processes in the job.
epilogue	string	Specifies a user owned epilogue script which will be run before the system epilogue and epilogue.user scripts at the completion of a job. The syntax is `epilogue=<file>`. The file can be designated with an absolute or relative path.
file	size*	The amount of total disk requested for the job. (Ignored on Unicos.)
host	string	Name of the host on which the job should be run. This resource is provided for use by the site's scheduling policy. The allowable values and effect on job placement is site dependent.
mem	size*	Maximum amount of physical memory used by the job. (Ignored on Darwin, Digital Unix, Free BSD, HPUX 11, IRIX, NetBSD, and SunOS. Also ignored on Linux if number of nodes is not 1. Not implemented on AIX and HPUX 10.)
nice	integer	Number between -20 (highest priority) and 19 (lowest priority). Adjust the process execution priority.
nodes	{<node_count> | <hostname>} [:ppn=<ppn>][:gpus=<gpu>][:<property>[:<property>]...] [+ ...]	Number and/or type of nodes to be reserved for exclusive use by the job. The value is one or more node_specs joined with the **+** (plus) character:

node_spec[+node_spec...]. Each node_spec is a number of nodes required of the type declared in the node_spec and a name of one or more properties desired for the nodes. The number, the name, and each property in the node_spec are separated by a **:** (colon). If no number is specified, one (1) is assumed.

The name of a node is its hostname. The properties of nodes are:

- **ppn=**# - Specify the number of virtual processors per node requested for this job.

  The number of virtual processors available on a node by default is 1, but it can be configured in the $TORQUE_HOME/server_priv/nodes file using the np attribute. The virtual processor can relate to a physical core on the node or it can be interpreted as an "execution slot" such as on sites that set the node np value greater than the number of physical cores (or hyper-thread contexts). The ppn value is a characteristic of the hardware, system, and site, and its value is to be determined by the administrator.

- **gpus=#** - Specify the number of GPUs per node requested for this job.

  The number of GPUs available on a node can be configured in the $TORQUE_HOME/server_priv/nodes file using the gpu attribute. The GPU value is a characteristic of the hardware, system, and site, and its value is to be determined by the administrator.

- **property** - A string assigned by the system administrator specifying a node's features. Check with your administrator as to the node names and properties available to you.

See Example 1 (-l nodes) for examples.

By default, the **node** resource is mapped to a virtual node (that is, directly to a processor, not a full physical compute node). This behavior can be changed within Maui or Moab by setting the JOBNODEMATCHPOLICY parameter. See Appendix F of the Moab Workload Manager Administrator's Guide for more

		information.
opsys	string	Specifies the administrator defined operating system as defined in the MOM configuration file.
other	string	Allows a user to specify site specific information. This resource is provided for use by the site's scheduling policy. The allowable values and effect on job placement is site dependent.
pcput	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by any single process in the job.
pmem	size*	Maximum amount of physical memory used by any single process of the job. (Ignored on Fujitsu. Not implemented on Digital Unix and HPUX.)
procs	procs=<integer>	*(Applicable in version 2.5.0 and later.)* The number of processors to be allocated to a job. The processors can come from one or more qualified node(s). Only one `procs` declaration may be used per submitted qsub command. `> qsub -l nodes=3+procs=2`
procs_bitmap	string	A string made up of 1's and 0's in reverse order of the processor cores requested. A **procs_bitmap=1110** means the job requests a node that has four available cores, but the job runs exclusively on cores two, three, and four. With this bitmap, core one is not used.
prologue	string	Specifies a user owned prologue script which will be run after the system prologue and prologue.user scripts at the beginning of a job. The syntax is `prologue=<file>`. The file can be designated with an absolute or relative path.
pvmem	size*	Maximum amount of virtual memory used by any single process in the job. (Ignored on Unicos.)
software	string	Allows a user to specify software required by the job. This is useful if certain software packages are only available on certain systems in the site. This resource is provided for use by the site's scheduling policy. The allowable values and effect on job placement is site dependent. (See Scheduler License Management in the Moab Workload Manager Administrator's Guide for more information.)
vmem	size*	Maximum amount of virtual memory used by all concurrent processes in the job. (Ignored on Unicos.)

walltime	seconds, or [[HH:]MM:]SS	Maximum amount of real time during which the job can be in the running state.

***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.

Suffix	Multiplier
b w	1
kb kw	1024
mb mw	1,048,576
gb gw	1,073,741,824
tb tw	1,099,511,627,776

Example 1 (qsub -l nodes)

Usage	Description
`> qsub -l nodes=12`	request 12 nodes of any type
`> qsub -l nodes=2:server+14`	request 2 "server" nodes and 14 other nodes (a total of 16) - this specifies two node_specs, "2:server" and "14"
`> qsub -l nodes=server:hippi+10:noserver+3:bigmem:hippi`	request (a) 1 node that is a "server" and has a "hippi" interface, (b) 10 nodes that are not servers, and (c) 3 nodes that have a large amount of memory an have hippi
`> qsub -l nodes=b2005+b1803+b1813`	request 3 specific nodes by hostname
`> qsub -l nodes=4:ppn=2`	request 2 processors on each of four nodes
`> qsub -l nodes=1:ppn=4`	request 4 processors on one node
`> qsub -l nodes=2:blue:ppn=2+red:ppn=3+b1014`	request 2 processors on each of two blue nodes, three processors on one red node, and the compute node "b1014"

Example 2

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

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

Example 3

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

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

2.1.3 Requesting Generic Resources

When **generic** resources have been assigned to nodes using the server's nodes file, these resources can be requested at the time of job submission using the ***other*** field. (See Consumable Generic Resources in the Moab Workload Manager Administrator's Guide for details on configuration within Moab).

Example 1

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

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

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

2.1.4 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. (See Floating Generic Resources in the Moab Workload Manager Administrator's Guide for details on configuration within Moab.)

Example 2

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

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

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

2.1.5 Requesting Other Resources

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

2.1.6 Exported Batch Environment Variables

When a batch job is started, a number of variables are introduced into the job's environment that can be used by the batch script in making decisions, creating output files, and so forth. These variables are listed in the following table:

Variable	Description
PBS_JOBNAME	user specified jobname
PBS_ARRAYID	zero-based value of job array index for this job (in version 2.2.0 and later)
PBS_O_WORKDIR	job's submission directory
PBS_ENVIRONMENT	N/A
PBS_TASKNUM	number of tasks requested
PBS_O_HOME	home directory of submitting user
PBS_MOMPORT	active port for MOM daemon
PBS_O_LOGNAME	name of submitting user
PBS_O_LANG	language variable for job
PBS_JOBCOOKIE	job cookie
PBS_NODENUM	node offset number
PBS_O_SHELL	script shell
PBS_O_JOBID	unique pbs job id
PBS_O_HOST	host on which job script is currently running
PBS_QUEUE	job queue
PBS_NODEFILE	file containing line delimited list on nodes allocated to the job
PBS_O_PATH	path variable used to locate executables within job script

2.1.7 Enabling Trusted Submit Hosts

By default, only the node running the pbs_server daemon is allowed to submit jobs. Additional nodes can be trusted as submit hosts by taking any of the following steps:

- Set the allow_node_submit server parameter.
  - Allows any host trusted as a compute host to also be trusted as a submit host.
- Set the submit_hosts server parameter (comma-delimited).
  - Allows specified hosts to be trusted as a submit host.
- Use .rhosts to enable *ruserok()* based authentication.

See Job Submission Host Advanced Config for more information.

If **allow_node_submit** is set, the parameter **allow_proxy_user** must be set to allow user proxying when submitting/running jobs.

2.1.8 Example Submit Scripts

The following is an example job test script:

```
#!/bin/sh
#
#This is an example script example.sh
#
#These commands set up the Grid Environment for your job:
#PBS -N ExampleJob
#PBS -l nodes=1,walltime=00:01:00
#PBS -q np_workq
#PBS -M YOURUNIQNAME@umich.edu
#PBS -m abe

#print the time and date
date

#wait 10 seconds
sleep 10

#print the time and date again
date
```

See Also

- Maui Documentation
- http://www.lunarc.lu.se/Support/SpecialTopics/ExampleScripts/MatlabScripts/MatlabScript
- http://www.clusters.umaine.edu/wiki/index.php/Example_Submission_Scripts
- qsub wrapper - Allow local checking and modification of submitted jobs

2.2 Monitoring Jobs

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

```
> qstat

Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - ----
-
4807             scatter          user01           12:56:34 R batch
...
```

2.3 Canceling Jobs

TORQUE allows users and administrators to cancel submitted jobs with the qdel command. The job will be sent TERM and KILL signals killing the running processes. When the top-level job script exits, the job will exit. The only parameter is the ID of the job 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
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - ----
-
4807             scatter          user01           12:56:34 R batch
...
$ qdel -m "hey! Stop abusing the NFS servers" 4807
$
```

2.4 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 supports 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 MOM's configuration file found in `TORQUE_HOME/mom_priv/config`. The checkpoint script should have execute permissions set. A typical configuration file might look as follows:

`mom_priv/config`:

```
$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 as follows:

`/src/include/pbs_config.h`:

```
#define MOM_CHECKPOINT 1
```

2.5 Keeping 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 to 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 allows TORQUE to better communicate with Moab Workload Manager and track the status of jobs. This gives Moab the ability to track specific failures and to schedule the workload around possible hazards. (See NODEFAILURERESERVETIME in Appendix F of the Moab Workload Manager Administrator's Guide for more information.)

2.6 Job Checkpoint and Restart

While TORQUE has had a job checkpoint and restart capability for many years, this was tied to machine specific features. Now TORQUE supports BLCR — an architecture independent package that provides for process checkpoint and restart.

The support for BLCR is only for serial jobs, not for any MPI type jobs.

Introduction to BLCR

BLCR is a kernel level package. It must be downloaded and installed from BLCR.

After building and making the package, it must be installed into the kernel with commands as follows. These can be installed into the file **/etc/modules** but all of the testing was done with explicit invocations of **modprobe**.

Installing BLCR into the kernel:

```
# /sbin/insmod /usr/local/lib/blcr/2.6.12-1.234/blcr_imports.ko
# /sbin/insmod /usr/local/lib/blcr/2.6.12-1.234/blcr_vmadump.ko
# /sbin/insmod /usr/local/lib/blcr/2.6.12-1.234/blcr.ko
```

The BLCR system provides four command line utilities: (1) **cr_checkpoint**, (2) **cr_info**, (3) **cr_restart**, and (4) **cr_run**.

For more information about BLCR, see the BLCR Administrator's Guide.

Configuration files and scripts

Configuring and Building TORQUE for BLCR:

```
> ./configure --enable-unixsockets=no --enable-blcr
> make
> sudo make install
```

The pbs_mom configuration file located in /var/spool/torque/mom_priv must be modified to identify the script names associated with invoking the BLCR commands. The following variables should be used in the configuration file when using BLCR checkpointing.

- $checkpoint_interval - How often periodic job checkpoints will be taken (minutes).
- $checkpoint_script - The name of the script file to execute to perform a job checkpoint.
- $restart_script - The name of the script file to execute to perform a job restart.
- $checkpoint_run_exe - The name of an executable program to be run when starting a checkpointable job (for BLCR, cr_run).

The following example shows the contents of the configuration file used for testing the BLCR feature in TORQUE.

The script files below must be executable by the user. Be sure to use **chmod** to set the permissions to 754.

Script file permissions:

```
# chmod 754 blcr*
# ls -l
total 20
-rwxr-xr-- 1 root root 2112 2008-03-11 13:14 blcr_checkpoint_script
-rwxr-xr-- 1 root root 1987 2008-03-11 13:14 blcr_restart_script
-rw-r--r-- 1 root root  215 2008-03-11 13:13 config
drwxr-x--x 2 root root 4096 2008-03-11 13:21 jobs
```

```
-rw-r--r-- 1 root root    7 2008-03-11 13:15 mom.lock
```

mom_priv/config:

```
$checkpoint_script  /var/spool/torque/mom_priv/blcr_checkpoint_script
$restart_script  /var/spool/torque/mom_priv/blcr_restart_script
$checkpoint_run_exe /usr/local/bin/cr_run
$pbsserver makua.cridomain
$loglevel 7
```

mom_priv/blcr_checkpoint_script:

```
#! /usr/bin/perl
###############################################################################

#
# Usage: checkpoint_script
#
# This script is invoked by pbs_mom to checkpoint a job.
#
###############################################################################

use strict;
use Sys::Syslog;

# Log levels:
# 0 = none -- no logging
# 1 = fail -- log only failures
# 2 = info -- log invocations
# 3 = debug -- log all subcommands
my $logLevel = 3;

logPrint(2, "Invoked: $0 " . join(' ', @ARGV) . "\n");

my ($sessionId, $jobId, $userId, $signalNum, $checkpointDir,
$checkpointName);
my $usage =
  "Usage: $0       \n";

# Note that depth is not used in this script but could control a
limit to the number of checkpoint
```

mom_priv/blcr_restart_script:

```
#! /usr/bin/perl
###############################################################################

#
# Usage: restart_script
#
# This script is invoked by pbs_mom to restart a job.
#
###############################################################################

use strict;
use Sys::Syslog;

# Log levels:
# 0 = none -- no logging
# 1 = fail -- log only failures
# 2 = info -- log invocations
# 3 = debug -- log all subcommands
my $logLevel = 3;

logPrint(2, "Invoked: $0 " . join(' ', @ARGV) . "\n");
```

```
my ($sessionId, $jobId, $userId, $checkpointDir, $restartName);
my $usage =
  "Usage: $0      \n";
if (@ARGV == 5)
{
    ($sessionId, $jobId, $userId, $checkpointDir, $restartName) =
      @ARGV;
```

Starting a checkpointable job

Not every job is checkpointable. A job for which checkpointing is desirable must be started with the **-c** command line option. This option takes a comma-separated list of arguments that are used to control checkpointing behavior. The list of valid options available in the 2.4 version of Torque is show below.

- **none** - No checkpointing (not highly useful, but included for completeness).
- **enabled** - Specify that checkpointing is allowed, but must be explicitly invoked by either the **qhold** or **qchkpt** commands.
- **shutdown** - Specify that checkpointing is to be done on a job at pbs_mom shutdown.
- **periodic** - Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the $checkpoint_interval option in the MOM configuration file, or by specifying an interval when the job is submitted.
- **interval=minutes** - Specify the checkpoint interval in minutes.
- **depth=number** - Specify a number (depth) of checkpoint images to be kept in the checkpoint directory.
- **dir=path** - Specify a checkpoint directory (default is /var/spool/torque/checkpoint).

Sample test program:

```
#include "stdio.h"
int main( int argc, char *argv[] )
{
int i;
        for (i=0; i<100; i++)
        {
                printf("i = %d\n", i);
                fflush(stdout);
                sleep(1);
        }
}
```

Instructions for building test program:

```
>  gcc -o test test.c
```

Sample test script:

```
#!/bin/bash

./test
```

Starting the test job:

```
>  qstat
>  qsub -c enabled,periodic,shutdown,interval=1 test.sh
77.jakaa.cridomain
>  qstat
Job id                    Name             User            Time Use S
Queue
------------------------- ---------------- --------------- ------
-- - -----
77.jakaa                  test.sh          jsmith                 0 Q
batch
>
```

If you have no scheduler running, you might need to start the job with qrun.

As this program runs, it writes its output to a file in /var/spool/torque/spool. This file can be observered with the command **tail -f**.

Checkpointing a job

Jobs are checkpointed by issuing a qhold command. This causes an image file representing the state of the process to be written to disk. The directory by default is `/var/spool/torque/checkpoint`.

This default can be altered at the queue level with the **qmgr** command. For example, the command **qmgr -c set queue batch checkpoint_dir=/tmp** would change the checkpoint directory to **/tmp** for the queue 'batch'.

The default directory can also be altered at job submission time with the **-c dir=/tmp** command line option.

The name of the checkpoint directory and the name of the checkpoint image file become attributes of the job and can be observed with the command **qstat -f**. Notice in the output the names **checkpoint_dir** and **checkpoint_name**. The variable **checkpoint_name** is set when the image file is created and will not exist if no checkpoint has been taken.

A job can also be checkpointed without stopping or holding the job with the command **qchkpt**.

Restarting a job in the Held state

The qrls command is used to restart the hibernated job. If you were using the **tail -f** command to watch the output file, you will see the test program start counting again.

It is possible to use the qalter command to change the name of the checkpoint file associated with a job. This could be useful if there were several job checkpoints and it restarting the job from an older image was specified.

Restarting a job in the Completed state

In this case, the job must be moved to the Queued state with the qrerun command. Then the job must go to the Run state either by action of the scheduler or if there is no scheduler, through using the qrun command.

Acceptance tests

A number of tests were made to verify the functioning of the BLCR implementation. See tests-2.4 for a description of the testing.

2.7 Job Exit Status

Once a job under TORQUE has completed, the exit_status attribute will contain the result code returned by the job script. This attribute can be seen by submitting a **qstat -f** command to show the entire set of information associated with a job. The exit_status field is found near the bottom of the set of output lines.

```
qstat -f (job failure example)
Job Id: 179.host
    Job_Name = STDIN
    Job_Owner = user@host
    job_state = C
    queue = batchq
    server = host
    Checkpoint = u
    ctime = Fri Aug 29 14:55:55 2008
    Error_Path = host:/opt/moab/STDIN.e179
    exec_host = node1/0
    Hold_Types = n
    Join_Path = n
    Keep_Files = n
    Mail_Points = a
    mtime = Fri Aug 29 14:55:55 2008
    Output_Path = host:/opt/moab/STDIN.o179
    Priority = 0
    qtime = Fri Aug 29 14:55:55 2008
    Rerunable = True
    Resource_List.ncpus = 2
    Resource_List.nodect = 1
    Resource_List.nodes = node1
    Variable_List = PBS_O_HOME=/home/user,PBS_O_LOGNAME=user,

PBS_O_PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:,P

  PBS_O_WORKDIR=/opt/moab,PBS_O_QUEUE=batchq
    sched_hint = Post job file processing error; job 179.host on
```

This code can be useful in diagnosing problems with jobs that may have unexpectedly terminated.

If TORQUE was unable to start the job, this field will contain a negative number produced by the pbs_mom.

Otherwise, if the job script was successfully started, the value in this field will be the return value of the script.

TORQUE Supplied Exit Codes

Name	Value	Description
JOB_EXEC_OK	0	job exec successful
JOB_EXEC_FAIL1	-1	job exec failed, before files, no retry
JOB_EXEC_FAIL2	-2	job exec failed, after files, no retry
JOB_EXEC_RETRY	-3	job execution failed, do retry
JOB_EXEC_INITABT	-4	job aborted on MOM initialization
JOB_EXEC_INITRST	-5	job aborted on MOM init, chkpt, no migrate
JOB_EXEC_INITRMG	-6	job aborted on MOM init, chkpt, ok migrate
JOB_EXEC_BADRESRT	-7	job restart failed
JOB_EXEC_CMDFAIL	-8	exec() of user command failed

Example of exit code from C program:

```
$ cat error.c

#include
#include


int
main(int argc, char *argv)
{
  exit(256+11);
}


$ gcc -o error error.c


$ echo ./error | qsub
180.xxx.yyy


$ qstat -f
Job Id: 180.xxx.yyy
    Job_Name = STDIN
    Job_Owner = test.xxx.yyy
    resources_used.cput = 00:00:00
    resources_used.mem = 0kb
    resources_used.vmem = 0kb
    resources_used.walltime = 00:00:00
    job_state = C
```

Notice that the C routine **exit** passes only the low order byte of its argument. In this case, 256+11 is really 267 but the resulting exit code is only 11 as seen in the output.

2.8 Service Jobs

TORQUE service jobs are a special kind of job that is treated differently by TORQUE than normal batch jobs. TORQUE service jobs are **not** related to Moab's dynamic service jobs. A TORQUE service job cannot dynamically grow and shrink in size over time.

Jobs are marked as service jobs at the time they are submitted to Moab or TORQUE. Just like a normal job, a script file is specified with the job. In a batch job, the contents of the script file are taken by TORQUE and executed on the compute nodes. For a service job, however, the script file is assumed to respond to certain command-line arguments. Instead of just executing the script, TORQUE will use these command-line arguments to start, stop, and check on the status of the job. Listed below are the three command-line arguments that must be supported by any script submitted as part of a TORQUE service job:

- 'start' - The script should take this argument and launch its service/workload. The script should remain executing/running until the service stops.
- 'stop' - The script should take this argument and stop the service/workload that was earlier started.
- 'status' - The script should take this argument and return, via standard out, either "running" if the service/workload is running as expected or "stopped" if the service is not running.

This feature was created with long-running services in mind. The command-line arguments should be familiar to users who interact with Unix services, as each of the service scripts found in /etc/init.d/ also accept and respond to the arguments as explained above.

For example, if a user wants to start the Apache 2 server on a compute node, they can use a TORQUE service job and specify a script which will start, stop, and check on the status of the "httpd" daemon--possibly by using the already present /etc/init.d/httpd script.

Moab Version Required

If you wish to submit service jobs only through TORQUE, no special version of Moab is required. If you wish to submit service jobs using Moab's msub, then Moab 5.4 is required.

Submitting Service Jobs

There is a new option to qsub, "-s" which can take either a 'y' or 'n' (yes or no, respectively). When "-s y" is present, then the job is marked as a service job.

```
qsub -l walltime=100:00:00,nodes=1 -s y service_job.py
```

The example above submits a job to TORQUE with a walltime of 100 hours, one node, and it is marked as a service job. The script "service_job.py" will be used to start, stop, and check the status of the service/workload started on the compute nodes.

Moab, as of version 5.4, is able to accept the "-s y" option when msub is used for submission. Moab will then pass this information to TORQUE when the job is migrated.

Submitting Service Jobs in MCM

Submitting a service job in MCM requires the latest Adaptive Computing Suite snapshot of MCM. It also requires MCM to be started with the "--future=2" option.

Once MCM is started, open the **Create Workload** window and verify **Show Advanced Options** is checked. Notice that there is a **Service** checkbox that can be selected in the **Flags/Options** area. Use this to specify the job is a service job.

Managing Service Jobs

Managing a service job is done much like any other job; only a few differences exist.

Examining the job with qstat -f will reveal that the job has the service = True attribute. Non-service jobs will not make any mention of the "service" attribute.

Canceling a service job is done with qdel, mjobctl -c, or through any of the GUI's as with any other job. TORQUE, however, cancels the job by calling the service script with the "stop" argument instead of killing it directly. This behavior also occurs if the job runs over its wallclock and TORQUE/Moab is configured to cancel the job.

If a service job completes when the script exits after calling it with "start," or if TORQUE invokes the script with "status" and does not get back "running," it will **not** be terminated by using the "stop" argument.

3.1 Adding Nodes

TORQUE can add and remove nodes either dynamically with **qmgr** or by manually editing the **TORQUE_HOME/server_priv/nodes** file. (See Initializing/Configure TORQUE on the server (pbs_server).

3.1.1 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"
```

The above command appends the $TORQUE_HOME/server_priv/nodes file with:

```
node003
```

Nodes can also be removed with a similar command:

```
> qmgr -c "delete node node003"
```

Typically, an administrator will want to change the state of a node instead of remove it. See Changing Node State.

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.

3.2 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 has a higher speed network connection could have the property "ib". TORQUE can set, update, or remove properties either dynamically with **qmgr** or by manually editing the **nodes** file.

3.2.1 Run-Time Node Changes

TORQUE can dynamically change the properties of a node with the **qmgr** command. For example, note the following to give `node001` the properties of `bigmem` and `dualcore`:

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

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

3.2.2 Manual Node Changes

The properties of each node are enumerated in **TORQUE_HOME/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 `bigmem` and `dualcore` and `node002` the properties of `bigmem` and `matlab`, edit the **nodes** file to contain the following:

`server_priv/nodes`:

```
node001 bigmem dualcore
node002 np=4 bigmem matlab
```

For changes to the **nodes** file to be activated, **pbs_server** must be restarted.

For a full description of this file, please see the **PBS Administrator Guide**.

See Also:

- 2.1 Job Submission for specifying nodes properties for submitted jobs.

3.3 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 migth be having a problem.

See the pbsnodes manpage for details.

3.4 Host Security

For systems requiring dedicated access to compute nodes (for example, users with sensitive data), TORQUE prologue and epilogue scripts provide a vehicle to leverage the authenication provided by linux-PAM modules. (See Appendix G Prologue and Epilogue Scripts for more information.)

To allow only users with running jobs (and root) to access compute nodes, do the following:

- Untar contrib/pam_authuser.tar.gz (found in the src tar ball).
- Compile pam_authuser.c with **make** and **make install** on every compute node.
- Edit /etc/system-auth as described in README.pam_authuser, again on every compute node.
- Either make a tarball of the epilogue* and prologue* scripts (to preserve the symbolic link) and untar it in the mom_priv directory, or just copy epilogue* and prologue* to mom_priv/.

The prologue* scripts are Perl scripts that add the user of the job to /etc/authuser. The epilogue* scripts then remove the first occurrence of the user from /etc/authuser. File locking is employed in all scripts to eliminate the chance of race conditions. There is also some commented code in the epilogue* scripts, which, if uncommented, kills all processes owned by the user (using pkill), provided that the user doesn't have another valid job on the same node.

prologue and epilogue scripts were added to the pam_authuser tarball in version 2.1 of TORQUE.

3.5 Linux Cpuset Support

3.5.1 Cpuset Overview

Linux kernel 2.6 Cpusets are logical, hierarchical groupings of CPUs and units of memory. Once created, individual processes can be placed within a cpuset. The processes will only be allowed to run/access the specified CPUs and memory. Cpusets are managed in a virtual file system mounted at /dev/cpuset. New cpusets are created by simply making new directories. Cpusets gain CPUs and memory units by simply writing the unit number to files within the cpuset.

3.5.2 Cpuset Support

When started, pbs_mom will create an initial top-level cpuset at /dev/cpuset/torque. This cpuset contains all CPUs and memory of the host machine. If this "torqueset" already exists, it will be left unchanged to allow the administrator to override the default behavior. All subsequent cpusets are created within the torqueset.

When a job is started, the jobset is created at /dev/cpuset/torque/$jobid and populated with the CPUs listed in the exec_host job attribute. Also created are individual tasksets for each CPU within the jobset. This happens before prologue, which allows it to be easily modified, and it happens on all nodes.

The top-level batch script process is executed in the jobset. Tasks launched through the TM interface (pbsdsh and PW's mpiexec) will be executed within the appropriate taskset.

On job exit, all tasksets and the jobset are deleted.

3.5.3 Cpuset Configuration

At the moment, there are no run-time configurations. The support is disabled by default at build-time. Run configure with `--enable-cpuset` if you would like to test the code.

If enabled and run on a machine without cpuset support, **pbs_mom** still executes the jobs, but it logs errors in the log file.

A run-time **pbs_mom** boolean needs to be created to enable/disable it.

On the Linux host, the virtual file system must be mounted:

```
mount -t cpuset none /dev/cpuset
```

3.5.4 Cpuset advantages / disadvantages

Presently, any job can request a single CPU and proceed to use everything available in the machine. This is occasionally done to circumvent policy, but most often is simply an error on the part of the user. Cpuset support will easily constrain the processes to not interfere with other jobs.

Jobs on larger NUMA systems may see a performance boost if jobs can be intelligently assigned to specific CPUs. Jobs may perform better if striped across physical processors, or contained within the fewest number of memory controllers.

TM tasks are constrained to a single core, thus a multi-threaded process could seriously suffer.

3.6 Scheduling Cores

In TORQUE 2.4 and later, users can request specific cores on a node at job submission by using geometry requests. To use this feature, users specify the procs_bitmap resource request of qsub -l at job submission.

3.6.1 Geometry Request Configuration

A Linux kernel of 2.6, or later, is required to use geometry requests. If the operating evnironment is suitable for geometry requests, configure TORQUE with the `--enable-geometry-requests` option.

```
> ./configure --prefix=/home/john/torque --enable-geometry-requests
```

TORQUE is configured to install to `/home/john/torque` and to enable the geometry requests feature.

The geometry request feature uses a subset of the cpusets feature. This means that `--enable-cpuset` must be configured to do jobs without a procs_bitmap request to use cpusets.

3.6.2 Geometry Request Usage

Once enabled, users can submit jobs with a geometry request by using the **procs_bitmap**=<*string*> resource request. **procs_bitmap** requires a numerical string made up of 1's and 0's. A 0 in the bitmap means the job can not run on the core that matches the 0's index in the bitmap. The index is in reverse order of the number of cores available. If a job is submitted with `procs_bitmap=1011`, then the job requests a node with four free cores, and uses only cores one, two, and four.

The geometry request feature requires a node that has all cores free. A job with a geometry request cannot run on a node that has cores that are busy, even if the node has more than enough cores available to run the job.

```
qsub -l procs_bitmap=0011 ossl.sh
```

The job ossl.sh is submitted with a geometry request of 0011.

In the above example, the submitted job can run only on a node that has four cores. When a suitable node is found, the job runs exclusively on cores one and two.

3.6.3 Geometry Request Considerations

As stated above, jobs with geometry requests require a node with all of its cores available. After the job starts running on the requested cores, the node cannot run other jobs, even if the node has enough free cores to meet the requirements of the other jobs. Once the geometry requesting job is done, the node is available to other jobs again.

3.7 Scheduling GPUs

In TORQUE 2.5.4 and later, users can request GPUs on a node at job submission by specifying a nodes= resource request using the qsub -l option. The number of GPUs a node has must be specified in the nodes file. The GPU is then reported in the output of pbsnodes:

```
napali
state = free
np = 2
ntype = cluster
status =
rectime=1288888871,varattr=,jobs=,state=free,netload=1606207294,gres=to
1805 2380 2428 1161 3174 3184 3191 3209 3228 3272 3333 20560
32371,uname=Linux napali 2.6.32-25-generic #45-Ubuntu SMP Sat Oct 16
19:52:42 UTC 2010 x86_64,opsys=linux
mom_service_port = 15002
mom_manager_port = 15003
gpus = 1
```

The $PBS_GPUFILE has been created to include GPU awareness. The GPU appears as a separate line in $PBS_GPUFILE and follows this syntax:

```
<hostname>-gpu<index>
```

If a job were submitted to run on a server called napali (the submit command would look something like: qsub test.sh -l nodes=1:ppn=2:gpus=1), the $PBS_GPUFILE would contain:

```
napali
napali
napali-gpu0
```

The first two lines signify the job has 2 ppn on napali, and the last line explains that napali has GPU index 0 (the first GPU) to execute on as well. It is left up to the job's owner to make sure that the job executes properly on the GPU. By default, TORQUE treats GPUs exactly the same as ppn (which corresponds to CPUs).

4.1 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** 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 what follows.

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.

4.1.1 Queue Attributes

acl_groups	
Format:	<GROUP>[@<HOST>][+<USER>[@<HOST>]]...
Default:	---
Description:	Specifies the list of groups which may submit jobs to the queue. If acl_group_enable is set to **true**, only users with a primary group listed in **acl_groups** may utilize the queue.  If the **PBSACLUSEGROUPLIST** variable is set in the **pbs_server** environment, acl_groups checks against all groups of which the job user is a member.

Example:	`> qmgr -c "set queue batch acl_groups=staff"`
	`> qmgr -c "set queue batch acl_groups+=ops@h2"`
	`> qmgr -c "set queue batch acl_groups+=staff@h3"`
	  Used in conjunction with acl_group_enable

acl_group_enable	
Format:	<BOOLEAN>
Default:	FALSE
Description:	If **TRUE**, constrains TORQUE to only allow jobs submitted from groups specified by the acl_groups parameter.
Example:	`qmgr -c "set queue batch acl_group_enable=true"`

acl_group_sloppy

Format:	<BOOLEAN>
Default:	FALSE
Description:	If **TRUE**, acl_groups will be checked against all groups of which the job user is a member.
Example:	---

acl_hosts	
Format:	<HOST>[+<HOST>]...
Default:	---
Description:	Specifies the list of hosts that may submit jobs to the queue.
Example:	`qmgr -c "set queue batch acl_hosts=h1+h2+h3"`  Used in conjunction with acl_host_enable.

acl_host_enable	
Format:	<BOOLEAN>
Default:	FALSE
Description:	If TRUE, constrains TORQUE to only allow jobs submitted from hosts specified by the **acl_hosts** parameter.
Example:	`qmgr -c "set queue batch acl_host_enable=true"`

acl_logic_or	
Format:	<BOOLEAN>
Default:	FALSE
Description:	If **TRUE**, user and group acls are logically OR'd together, meaning that either acl may be met to allow access. If false or unset, then both acls are AND'd, meaning that both acls must be satisfied.
Example:	`qmgr -c "set queue batch acl_logic_or=true"`

acl_users	
Format:	<USER>[@<HOST>][+<USER>[@<HOST>]]...
Default:	---
Description:	Specifies the list of users who may submit jobs to the queue. If acl_user_enable is set to **TRUE**, only users listed in **acl_users** may use the queue

Example:	`> qmgr -c "set queue batch acl_users=john"`
	`> qmgr -c "set queue batch acl_users+=steve@h2"`
	`> qmgr -c "set queue batch acl_users+=stevek@h3"`

	Used in conjunction with acl_user_enable.

acl_user_enable	
Format:	<BOOLEAN>
Default:	FALSE
Description:	If **TRUE**, constrains TORQUE to only allow jobs submitted from users specified by the acl_users parameter.
Example:	`qmgr -c "set queue batch acl_user_enable=true"`

disallowed_types	
Format:	<type>[+<type>]...
Default:	---
Description:	Specifies classes of jobs that are not allowed to be submitted to this queue. Valid types are interactive, batch, rerunable, nonrerunable, fault_tolerant (as of version 2.4.0 and later), fault_intolerant (as of version 2.4.0 and later), and job_array (as of version 2.4.1 and later).
Example:	`qmgr -c "set queue batch disallowed_types = interactive"` `qmgr -c "set queue batch disallowed_types += job_array"`

enabled	
Format:	<BOOLEAN>
Default:	FALSE
Description:	Specifies whether the queue accepts new job submissions.
Example:	`qmgr -c "set queue batch enabled=true"`

keep_completed	
Format:	<INTEGER>
Default:	0
Description:	Specifies the number of seconds jobs should be held in the Completed state after exiting.
Example:	`qmgr -c "set queue batch keep_completed=120"`

kill_delay	
Format:	<INTEGER>
Default:	2
Description:	Specifies the number of seconds between sending a SIGTERM and a SIGKILL to a job being cancelled.
Example:	`qmgr -c "set queue batch kill_delay=30"`

max_queuable	
Format:	<INTEGER>
Default:	unlimited
Description:	Specifies the maximum number of jobs allowed in the queue at any given time (includes idle, running, and blocked jobs).
Example:	`qmgr -c "set queue batch max_queuable=20"`

max_running	
Format:	<INTEGER>
Default:	unlimited
Description:	Specifies the maximum number of jobs in the queue allowed to run at any given time.
Example:	`qmgr -c "set queue batch max_running=20"`

max_user_queuable	
Format:	<INTEGER>
Default:	unlimited
Description:	Specifies the maximum number of jobs, per user, allowed in the queue at any given time (includes idle, running, and blocked jobs). Version 2.1.3 and greater.
Example:	`qmgr -c "set queue batch max_user_queuable=20"`

max_user_run	
Format:	<INTEGER>
Default:	unlimited
Description:	Specifies the maximum number of jobs, per user, in the queue allowed to run at any given time.
Example:	`qmgr -c "set queue batch max_user_run=10"`

priority	
Format:	<INTEGER>
Default:	0
Description:	Specifies the priority value associated with the queue.
Example:	`qmgr -c "set queue batch priority=20"`

queue_type	
Format:	one of **e**, **execution**, **r**, or route

Default:	---
Description:	Specifies the queue type.  This value must be explicitly set for all queues.
Example:	`qmgr -c "set queue batch queue_type=execution"`

resources_available	
Format:	<STRING>
Default:	---
Description:	Specifies to cumulative resources available to all jobs running in the queue.
Example:	`qmgr -c "set queue batch resources_available.nodect=20"`  **pbs_server** must be restarted for changes to take effect. Also, resources_available is constrained by the smallest of queue.resources_available and the server.resources_available.

resources_default	
Format:	<STRING>
Default:	N/A
Description:	Specifies default resource requirements for jobs submitted to the queue.
Example:	`qmgr -c "set queue batch resources_default.walltime=3600"`

resources_max	
Format:	<STRING>
Default:	N/A
Description:	Specifies the maximum resource limits for jobs submitted to the queue.
Example:	`qmgr -c "set queue batch resources_max.nodect=16"`

resources_min	
Format:	<STRING>
Default:	N/A
Description:	Specifies the minimum resource limits for jobs submitted to the queue.
Example:	`qmgr -c "set queue batch resources_min.nodect=2"`

route_destinations	
Format:	<queue>[@<host>][+<queue>[@<host>]]...
Default:	N/A
Description:	Specifies the potential destination queues for jobs submitted to the associated routing queue.  This attribute is only valid for routing queues.

Example:	`> qmgr -c "set queue route route_destinations=fast"`
	`> qmgr -c "set queue route route_destinations+=slow"`
	`> qmgr -c "set queue route route_destinations+=medium@hostname"`

started	
Format:	<BOOLEAN>
Default:	FALSE
Description:	Specifies whether jobs in the queue are allowed to execute.
Example:	`qmgr -c "set queue batch started=true"`

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

4.1.2 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.

4.1.3 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"
```

4.1.4 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 Moab and 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/Moab will use this information to ensure that jobs in that queue will be assigned nodes with that attribute. For example, suppose we have some nodes bought with money from the chemistry department, and some nodes paid by the biology department.

```
$TORQUE_HOME/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
```

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.

4.1.5 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_type Route with a route_destinations attribute set, as in the following example.

```
qmgr
# routing queue
create queue route
set queue route queue_type = Route
set queue route route_destinations = reg_64
set queue route route_destinations += reg_32
set queue route route_destinations += reg
set queue route enabled = True
set queue route started = True

# queue for jobs using 1-15 nodes
create queue reg
set queue reg queue_type = Execution
set queue reg resources_min.ncpus = 1
set queue reg resources_min.nodect = 1
set queue reg resources_default.ncpus = 1
set queue reg resources_default.nodes = 1
set queue reg enabled = True
set queue reg started = True

# queue for jobs using 16-31 nodes
create queue reg_32
set queue reg_32 queue_type = Execution
set queue reg_32 resources_min.ncpus = 31
set queue reg_32 resources_min.nodes = 16
set queue reg_32 resources_default.walltime = 12:00:00
set queue reg_32 enabled = True
set queue reg_32 started = True
```

In this example, the compute nodes are dual processors and default walltimes are set according to the number of processors/nodes of a job. Jobs with 32 nodes (63 processors) or more will be given a default walltime of 6 hours. Also, jobs with 16-31 nodes (31-62 processors) will be given a default walltime of 12 hours. All other jobs will have the server default walltime of 24 hours.

The ordering of the route_destinations is important. In a routing queue, a job is assigned to the first possible destination queue based on the resources_max, resources_min, acl_users, and acl_groups attributes. In the preceding example, the attributes of a single processor job would first be checked against the reg_64 queue, then the reg_32 queue, and finally the reg queue.

Adding the following settings to the earlier configuration elucidates the queue resource requirements:

```
qmgr
set queue reg resources_max.ncpus = 30
set queue reg resources_max.nodect = 15

set queue reg_16 resources_max.ncpus = 62
set queue reg_16 resources_max.ncpus = 31
```

The time of enforcement of server and queue defaults is important in this example. TORQUE applies server and queue defaults differently in job centric and queue centric modes. For job centric mode, TORQUE waits to apply the server and queue defaults until the job is assigned to its final execution queue. For queue centric mode, it enforces server defaults before it is placed in the routing queue. In either mode, queue defaults override the server defaults. TORQUE defaults to job centric mode. To set queue centric mode, set queue_centric_limits, as in what follows:

```
qmgr
set server queue_centric_limits = true
```

An artifact of job centric mode is that if a job does not have an attribute set, the server and routing queue defaults are not applied when queue resource limits are checked. Consequently, a job that requests 32 nodes (not ncpus=32) will not be checked against a min_resource.ncpus limit. Also, for the preceding example, a job without any attributes set will be placed in the reg_64 queue, since the server ncpus default will be applied after the job is assigned to an execution queue.

Routine queue defaults are NOT applied to job attributes in versions 2.1.0 and before.

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 queue's route_destinations attribute, change 'batch' to 'batch@localhost').

See Also

- Server Parameters
- qalter - command which can move jobs from one queue to another

4.2 Server High Availability

The option of running TORQUE in a redundant or high availability mode has been implemented. This means that there can be multiple instances of the server running and waiting to take over processing in the event that the currently running server fails.

The high availability feature is available in the 2.3 and later versions of TORQUE. TORQUE 2.4 included several enhancements to high availablilty.

Redundant Server Host Machines

High availability enables TORQUE to continue running even if pbs_server is brought down. This is done by running multiple copies of pbs_server which have their `torque/server_priv` directory mounted on a shared file system. The `torque/server_name` must include the host names of all nodes that run pbs_server. All MOM nodes also must include the host names of all nodes running pbs_server in their `torque/server_name` file. The syntax of the `torque/server_name` is a comma delimited list of host names.

`torque/server_name` example:

```
host1,host2,host3
```

All instances of **pbs_server** need to be started with the **--ha** command line option that allows the servers to run at the same time. Only the first server to start will complete the full startup. The second server to start will block very early in the startup when it tries to lock the file `torque/server_priv/server.lock`. When the second server cannot obtain the lock, it will spin in a loop and wait for the lock to clear. The sleep time between checks of the lock file is one second.

Notice that not only can the servers run on independent server hardware, there can also be multiple instances of the **pbs_server** running on the same machine. This was not possible before as the second one to start would always write an error and quit when it could not obtain the lock.

Because the file `server_priv/serverdb` is created in a way which is not compatible between hardware architectures, the machines that are running **pbs_server** in high-availability mode must be of similar architecture. For example, a 32-bit machine is unable to read the `server_priv/serverdb` file of a 64-bit machine. Therefore, when choosing hardware, verify all servers are of the same architecture.

Enhanced High Availability

The default high availability configuration of TORQUE 2.4 is backward compatible with version 2.3, but an enhanced high availability option is available with version 2.4. The enhanced version in 2.4 fixes some shortcomings in the default configuration and is more robust. The lock file mechanism used to trigger a fail-over in TORQUE 2.3 works correctly only if the primary pbs_server is taken down gracefully, and releases the lock on the file being used as the semaphore. If the server crashes, the lock stays in place and the backup server will not start unless the lock is manually removed by the administrator. With 2.4 enhanced high availability the reliance on the file system is bypassed with a much more reliable mechanism.

In order to use enhanced high availability with TORQUE 2.4, TORQUE must be configured using the `--enable-high-availability` option (in addition to all other configuration options you specify).

```
> ./configure --prefix=/usr/var/torque --enable-high-availability
```

In the above example, TORQUE installs to the `/usr/var/torque` directory and is configured to use the high availability features.

Once TORQUE has been compiled and installed, it is launched the same way as with TORQUE 2.3; start each instance of **pbs_server** with the **--ha** option.

In addition to the new fail-over mechanism, three server options have been added to help manage enhanced high availability in TORQUE 2.4. The server parameters are lock_file, lock_file_update_time, and lock_file_check_time.

The lock_file option allows the administrator to change the location of the lock file. The default location is `torque/server_priv`. If the lock_file option is used, the new location must be on the shared partition so all servers have access.

The lock_file_update_time and lock_file_check_time parameters are used by the servers to determine if the primary server is active. The primary pbs_server will update the lock file based on the lock_file_update_time (default value of 3 seconds). All backup pbs_servers will check the lock file as indicated by the lock_file_check_time parameter (default value of 9 seconds). The lock_file_update_time must be less than the lock_file_check_time. When a failure occurs, the backup pbs_server takes up to the lock_file_check_time value to take over.

```
> qmgr -c "set server lock_file_check_time=5"
```

In the above example, after the primary pbs_server goes down, the backup pbs_server takes up to 5 seconds to take over. It takes additional time for all MOMs to switch over to the new **pbs_server**.

The clock on the primary and redundant servers must be synchronized in order for high availability to work. Use a utility such as NTP to ensure your servers have a synchronized time.

Enhanced High Availability with Moab

When TORQUE is run with an external scheduler such as Moab, and the pbs_server is not running on the same host as Moab, pbs_server needs to know where to find the scheduler. To do this, use the following syntax (the port is required and the default is 15004):

```
> pbs_server --ha -l <moabhost:port>
```

If Moab is running in HA mode, add a -l option for each redundant server.

```
> pbs_server --ha -l <moabhost1:port> -l <moabhost2:port>
```

The root user of each Moab host must be added to the operators and managers lists of the server. This enables Moab to execute root level operations in TORQUE.

How Commands Select the Correct Server Host

The various commands that send messages to **pbs_server** usually have an option of specifying the server name on the command line, or if none is specified will use the default server name. The default server name comes either from the environment variable **PBS_DEFAULT** or from the file `torque/server_name`.

When a command is executed and no explicit server is mentioned, an attempt is made to connect to the first server name in the list of hosts from **PBS_DEFAULT** or `torque/server_name`. If this fails, the next server name is tried. If all servers in the list are unreachable, an error is returned and the command fails.

Note that there is a period of time after the failure of the current server during which the new server is starting up where it is unable to process commands. The new server must read the existing configuration and job information from the disk, so the length of time that commands cannot be received varies. Commands issued during this period of time might fail due to timeouts expiring.

Job Names

One aspect of this enhancement is in the construction of job names. Job names normally contain the name of the host machine where **pbs_server** is running. When job names are constructed, only the first name from the server specification list is used in building the job name.

Persistence of the pbs_server Process

The system adminstrator must ensure that **pbs_server** continues to run on the server nodes. This could be as simple as a **cron** job that counts the number of **pbs_server**'s in the process table and starts some more if needed.

High Availability of the NFS Server

One consideration of this implemention is that it depends on NFS file system also being redundant. NFS can be set up as a redundant service. See the following.

- Setting Up A Highly Available NFS Server
- Making NFS Work On Your Network
- Sourceforge Linux NFS FAQ
- NFS v4 main site

There are also other ways to set up a shared file system. See the following.

- Red Hat Global File System
- Data sharing with a GFS storage cluster

Example Setup of High Availability

1. The machines running **pbs_server** must have access to a shared `server_priv/` directory (usually an NFS share on a MoM).
2. All MoMs must have the same content in their `server_name` file. This can be done manually or via an NFS share. The `server_name` file contains a comma-delimited list of the hosts that run **pbs_server**.

```
# List of all servers running pbs_server
server1,server2
```

3. The machines running **pbs_server** must be listed in `acl_hosts`.

```
> qmgr -c "set server acl_hosts += server1"
> qmgr -c "set server acl_hosts += server2"
```

4. Start **pbs_server** with the **--ha** option.

```
[root@server1]$ pbs_server --ha

[root@server2]$ pbs_server --ha
```

5.1 Integrating Schedulers for TORQUE

Selecting the cluster scheduler is an important decision and significantly affects cluster utilization, responsiveness, availability, and intelligence. The default TORQUE scheduler, **pbs_sched**, is very basic and will provide poor utilization of your cluster's resources. Other options, such as Maui Scheduler or Moab Workload Manager, are highly recommended. If using Maui or Moab, refer to the Moab-PBS Integration Guide. If using **pbs_sched**, simply start the pbs_sched daemon.

If you are installing Moab Cluster Suite, TORQUE and Moab were configured at installation for interoperability and no further action is required.

6.1 SCP Setup

To use *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 the following:

```
>  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:

Easy Key Copy:

```
ssh-copy-id [-i [identity_file]] [user@]machine
```

Manual Steps to Copy Keys:

```
> 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 700 .ssh/authorized_keys
```

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 to step 4 and test the changes made 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:

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

Note that if this is your first time accessing *destination* from *source*, it may ask you if you want to add the fingerprint to a file of known hosts. If you specify yes, 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 preceding steps allow *source* access to *destination* without prompting for a password. The reverse, however, is not true. Repeat the 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

In TORQUE 2.1 and later, SCP is the default remote copy protocol. This step is only necessary for earlier versions.

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

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

If special scp flags are required in your local setup, these can be specified using the rcpcmd parameter.

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 file system 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 configuration file (located in the $PBSMOMHOME/mom_priv directory) using the following format **$usecp <HOST>:<SRCDIR> <DSTDIR>**

HOST can be specified with a leading wildcard ('*****') character. The following example demonstrates this directive:

```
mom_priv/config
# /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

File staging requirements are specified using the **stagein** and **stageout** directives of the qsub command. Stagein requests occur before the job starts execution, while stageout requests happen after a job completes.

On completion of the job, all staged-in and staged-out files are removed from the execution system. The **file_list** is in the form **local_file@hostname:remote_file[,...]** regardless of the direction of the copy. The name **local_file** is the name of the file on the system where the job executed. It may be an absolute path or relative to the home directory of the user. The name **remote_file** is the destination name on the host specified by hostname. The name may be absolute or relative to the user's home directory on the destination host. The use of wildcards in the file name is not recommended.

The file names map to a remote copy program (rcp/scp/cp, depending on configuration) called on the execution system in the following manner:

- For stagein: rcp/scp hostname:remote_file local_file
- For stageout: rcp/scp local_file hostname:remote_file

Examples

```
# stage /home/john/input_source.txt from node13.fsc to
/home/john/input_destination.txt on master compute node
> qsub -l nodes=1,walltime=100 -W
stagein=input_source.txt@node13.fsc:/home/john/input_destination.txt
```

```
# stage /home/bill/output_source.txt on master compute node to
/tmp/output_destination.txt on node15.fsc
> qsub -l nodes=1,walltime=100 -W
stageout=/tmp/output_source.txt@node15.fsc:/home/bill/output_destinatio
```

```
$ fortune >xxx;echo cat xxx|qsub -W stagein=xxx@`hostname`:xxx
199.myhost.mydomain
$ cat STDIN*199
Anyone who has had a bull by the tail knows five or six more things
than someone who hasn't.
                -- Mark Twain
```

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 Overview

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 task manager (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 that 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.

MPIExec Troubleshooting

Although problems with **mpiexec** are rare, if issues do occur, the following steps may be useful:

- determine current version using `mpiexec --version` and review the change log available on the MPI homepage to determine if the reported issue has already been corrected
- send email to the **mpiexec** mailing list at **mpiexec@osc.edu**
- browse the **mpiexec** user list archives for similar problems and resolutions
- read the FAQ contained in the **README** file and the mpiexec man pages contained within the **mpiexec** distribution
- increase the logging of **mpiexec** operation with **mpiexec --verbose** (reports messages to stderr)
- increase logging of the master and slave resource manager execution daemons associated with the job (with **TORQUE**, use `$loglevel` to 5 or higher in `$TORQUEROOT/mom_priv/config` and look for '**tm**' messages after associated `join job` messages).
- use **tracejob** (included with **TORQUE**) or **qtracejob** (included with OSC's **pbstools** package) to isolate failures within the cluster.
- if the message '`exec: Error: get_hosts: pbs_connect: Access from host not allowed, or unknown host`' appears, this indicates that **mpiexec** cannot communicate with the **pbs_server** daemon. In most cases, this indicates that the '`$TORQUEROOT/server_name`' file points to the wrong server or the node cannot resolve the server's name. The **qstat** command can be run on the node to test this.

General MPI Troubleshooting

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 that properly clean up the orphaned processes.

7.1.3 MPICH-VMI

MPICH-VMI is a highly-optimized open-source message passing layer available from NCSA. Additional information can be found in the VMI tutorial.

7.1.4 Open MPI

Open MPI is a new MPI implementation that combines technologies from multiple projects to create the best possible library. It supports the TM interface for intergration with TORQUE. More inforamtion is available in the FAQ.

8.1 Monitoring Resources

8.1.1 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, file systems, and so forth, although these resources can be integrated into the cluster using some scheduling systems.

With regard to monitoring compute nodes, TORQUE reports about a number of attributes broken into three major categories:

- configuration
- utilization
- state

8.1.1.1 Configuration

Configuration includes both detected hardware configuration and specified batch attributes.

Attribute	Description	Details
Architecture (arch)	operating system of the node	The value reported is a derivative of the operating system installed.
Node Features (properties)	arbitrary string attributes associated with the node	No node features are specified by default. If required, they are set using the nodes file located in the TORQUE_HOME/server_priv directory. They may specify any string and are most commonly used to allow users to request certain subsets of nodes when submitting jobs.
Local Disk (size)	configured local disk	By default, local disk space is not monitored. If the MOM configuration size parameter is set, TORQUE will report, in kilobytes, configured disk space within the specified directory.
Memory (physmem)	local memory/RAM	Local memory/RAM is monitored and reported in kilobytes.
Processors (ncpus/np)	real/virtual processors	The number of processors detected by TORQUE is reported via the **ncpus** attribute. However, for scheduling purposes, other factors are taken into account. In its default configuration, TORQUE operates in **dedicated** mode with each node possessing a single virtual processor. In dedicated mode, each job task will consume one virtual processor and TORQUE will accept workload on each node until all virtual processors on that node are in use. While the number of virtual processors per node defaults to 1, this may be configured using the nodes file located in the TORQUE_HOME/server_priv directory. An alternative to dedicated mode is *timeshared* mode. If TORQUE's *timeshared* mode is enabled, TORQUE will accept additional workload on each node until the node's **maxload** limit is reached.
Swap (totmem)	virtual memory/Swap	Virtual memory/Swap is monitored and reported in kilobytes.

8.1.1.2 Utilization

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

Attribute	Description	Details
Disk (size)	local disk availability	By default, local disk space is not monitored. If the MOM configuration size parameter is set, TORQUE will report configured and currently available disk space within the specified directory in kilobytes.
Memory (availmem)	real memory/RAM	Available real memory/RAM is monitored and reported in kilobytes.
Network (netload)	local network adapter usage	Reports total number of bytes transferred in or out by the network adapter.
Processor Utilization (loadave)	node's cpu load average	Reports the node's 1 minute bsd load average.

8.1.1.3 Node States

State information includes administrative status, general node health information, and general usage status.

Attribute	Description	Details
Idle Time (idletime)	time since local keyboard/mouse activity has been detected	Time in seconds since local keyboard/mouse activity has been detected.
State (state)	monitored/admin node state	A node can be in one or more of the following states: • **busy** - node is full and will not accept additional work • **down** - node is failing to report, is detecting local failures with node configuration or resources, or is marked down by an administrator • **free** - node is ready to accept additional work • **job-exclusive** - all available virtual processors are assigned to jobs • **job-sharing** - node has been allocated to run multiple shared jobs and will remain in this state until jobs are complete • **offline** - node has been instructed by an admin to no longer accept work • **reserve** - node has been reserved by the server • **time-shared** - node always allows multiple jobs to run concurrently • **unknown** - node has not been detected

9.1 Accounting Records

TORQUE maintains accounting records for batch jobs in the following directory:

$TORQUEROOT/server_priv/accounting/<*TIMESTAMP*>

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

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

Record Marker	Record Type	Description
D	delete	job has been deleted
E	exit	job has exited (either successfully or unsuccessfully)
Q	queue	job has been submitted/queued
S	start	an attempt to start the job has been made (if the job fails to properly start, it may have multiple job start records)

Accounting Variables

The following table offers accounting variable descriptions. Descriptions for accounting variables not indicated in the table, particularly those prefixed with Resources_List, are available at 2.1 Job Submission.

Variable	Description
ctime	time job was created
etime	time job was queued
qtime	time job became eligible to run
start	time job started to run

A sample record in this file can 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
```

10.1 Job Logging

New in TORQUE 2.5.3 is the ability to log job information for completed jobs. The information stored in the log file is the same information produced with the command qstat -f. The log file data is stored using an XML format. Data can be extracted from the log using the utility *showjobs* found in the `contrib/` directory of the TORQUE source tree. Custom scripts that can parse the XML data can also be used.

10.1.1 Job Log Location and Name

The job log is kept at `$TORQUE_HOME/job_logs`. The naming convention for the job log is the same as for the server log or MOM log. The log name is created from the current year/month/day . For example, if today's date is 26 October, 2010 the log file is named 20101026. A new log file is created each new day that data is written to the log.

10.1.2 Enabling Job Logs

There are five new server parameters used to enable job logging. These parameters control what information is stored in the log and manage the log files.

- **record_job_info** - This must be set to true in order for job logging to be enabled. If not set to true, the remaining server parameters are ignored.
- **record_job_script** - If set to *true*, this adds the contents of the script executed by a job to the log.
- **job_log_file_max_size** - This specifies a soft limit (in kilobytes) for the job log's maximum size. The file size is checked every five minutes and if the **current day** file size is greater than or equal to this value, it is rolled from <*filename*> to <*filename.1*> and a new empty log is opened. If the current day file size exceeds the maximum size a second time, the <*filename.1*> log file is rolled to <*filename.2*>, the current log is rolled to <*filename.1*>, and a new empty log is opened. Each new log causes all other logs to roll to an extension that is one greater than its current number. Any value less than 0 is ignored by pbs_server (meaning the log will not be rolled).
- **job_log_file_roll_depth** - This sets the maximum number of new log files that are kept in a day if the `job_log_file_max_size` parameter is set. For example, if the roll depth is set to 3, no file can roll higher than <*filename.3*>. If a file is already at the specified depth, such as <*filename.3*>, the file is deleted so it can be replaced by the incoming file roll, <*filename.2*>.
- **job_log_keep_days** - This maintains logs for the number of days designated. If set to 4, any log file older than 4 days old is deleted.

11.1 Troubleshooting

There are a few general strategies that can be followed to determine the cause of unexpected behavior. These are a few of the tools available to help determine where problems occur.

11.1.1 Host Resolution

The TORQUE server host must be able to perform both forward and reverse name lookup on itself and on all compute nodes. Likewise, each compute node must be able to perform forward and reverse name lookup on itself, the TORQUE server host, and all other compute nodes. In many cases, name resolution is handled by configuring the node's `/etc/hosts` file although **DNS** and **NIS** services may also be used. Commands such as **nslookup** or **dig** can be used to verify proper host resolution.

Invalid host resolution may exhibit itself with compute nodes reporting as down within the output of pbsnodes -a and with failure of the momctl -d 3 command.

11.1.2 Firewall Configuration

Be sure that, if you have firewalls running on the server or node machines, you allow connections on the appropriate ports for each machine. TORQUE **pbs_mom** daemons use UDP ports 1023 and below if privileged ports are configured (privileged ports is the default). The **pbs_server** and **pbs_mom** daemons use TCP and UDP ports 15001-15004 by default.

Firewall based issues are often associated with server to MOM communication failures and messages such as 'premature end of message' in the log files.

Also, the **tcpdump** program can be used to verify the correct network packets are being sent.

11.1.3 TORQUE Log Files

The **pbs_server** keeps a daily log of all activity in the `TORQUE_HOME/server_logs` directory. The **pbs_mom** also keeps a daily log of all activity in the `TORQUE_HOME/mom_logs/` directory. These logs contain information on communication between server and MOM as well as information on jobs as they enter the queue and as they are dispatched, run, and terminated. These logs can be very helpful in determining general job failures. For MOM logs, the verbosity of the logging can be adjusted by setting the loglevel parameter in the `mom_priv/config` file. For server logs, the verbosity of the logging can be adjusted by setting the server log_level attribute in qmgr.

For both **pbs_mom** and **pbs_server** daemons, the log verbosity level can also be adjusted by setting the environment variable **PBSLOGLEVEL** to a value between 0 and 7. Further, to dynamically change the log level of a running daemon, use the *SIGUSR1* and *SIGUSR2* signals to increase and decrease the active loglevel by one. Signals are sent to a process using the **kill** command. For example, **kill -USR1 `pgrep pbs_mom`** would raise the log level up by one. The current loglevel for **pbs_mom** can be displayed with the command **momctl -d3**.

11.1.4 Using tracejob to Locate Job Failures

Overview

The **tracejob** utility extracts job status and job events from accounting records, MOM log files, server log

files, and scheduler log files. Using it can help identify where, how, a why a job failed. This tool takes a job id as a parameter as well as arguments to specify which logs to search, how far into the past to search, and other conditions.

Syntax

```
tracejob [-a|s|l|m|q|v|z] [-c count] [-w size] [-p path] [ -n <DAYS>] [-f filter_type]
<JOBID>

  -p : path to PBS_SERVER_HOME
  -w : number of columns of your terminal
  -n : number of days in the past to look for job(s) [default 1]
  -f : filter out types of log entries, multiple -f's can be specified
       error, system, admin, job, job_usage, security, sched, debug,
       debug2, or absolute numeric hex equivalent
  -z : toggle filtering excessive messages
  -c : what message count is considered excessive
  -a : don't use accounting log files
  -s : don't use server log files
  -l : don't use scheduler log files
  -m : don't use MOM log files
  -q : quiet mode - hide all error messages
  -v : verbose mode - show more error messages
```

Example

```
> tracejob -n 10 1131

Job: 1131.icluster.org

03/02/2005 17:58:28  S    enqueuing into batch, state 1 hop 1
03/02/2005 17:58:28  S    Job Queued at request of dev@icluster.org,
owner =
                          dev@icluster.org, job name = STDIN, queue =
batch
03/02/2005 17:58:28  A    queue=batch
03/02/2005 17:58:41  S    Job Run at request of dev@icluster.org
03/02/2005 17:58:41  M    evaluating limits for job
03/02/2005 17:58:41  M    phase 2 of job launch successfully
completed
03/02/2005 17:58:41  M    saving task (TMomFinalizeJob3)
03/02/2005 17:58:41  M    job successfully started
03/02/2005 17:58:41  M    job 1131.koa.icluster.org reported
successful start on 1 node(s)
03/02/2005 17:58:41  A    user=dev group=dev jobname=STDIN
queue=batch ctime=1109811508
                          qtime=1109811508 etime=1109811508
start=1109811521
                          exec_host=icluster.org/0
Resource_List.neednodes=1 Resource_List.nodect=1
                          Resource_List.nodes=1
Resource_List.walltime=00:01:40
03/02/2005 18:02:11  M    walltime 210 exceeded limit 100
03/02/2005 18:02:11  M    kill_job
03/02/2005 18:02:11  M    kill job found a task to kill
```

The **tracejob** command operates by searching the pbs_server accounting records and the pbs_server, mom, and scheduler logs. To function properly, it must be run on a node and as a user which can access these files. By default, these files are all accessible by the user **root** and only available on the cluster management node. In particular, the files required by **tracejob** are located in the following directories:

- TORQUE_HOME/server_priv/accounting
- TORQUE_HOME/server_logs
- TORQUE_HOME/mom_logs
- TORQUE_HOME/sched_logs

tracejob may only be used on systems where these files are made available. Non-root users may be able to use this command if the permissions on these directories or files is changed appropriately.

11.1.5 Using GDB to Locate Failures

If either the **pbs_mom** or **pbs_server** fail unexpectedly (and the log files contain no information on the failure) gdb can be used to determine whether or not the program is crashing. To start **pbs_mom** or **pbs_server** under GDB export the environment variable **PBSDEBUG=yes** and start the program (i.e., gdb pbs_mom and then issue the **run** subcommand at the gdb prompt). GDB may run for some time until a failure occurs and at which point, a message will be printed to the screen and a gdb prompt again made available. If this occurs, use the gdb **where** subcommand to determine the exact location in the code. The information provided may be adequate to allow local diagnosis and correction. If not, this output may be sent to the mailing list or to help for further assistance. (for more information on submitting bugs or requests for help please see the Mailing List Instructions)


See the PBSCOREDUMP parameter for enabling creation of core files.

11.1.6 Other Diagnostic Options

When **PBSDEBUG** is set, some client commands will print additional diagnostic information.

```
$ export PBSDEBUG=yes
$ cmd
```

To debug different kinds of problems, it can be useful to see where in the code time is being spent. This is called profiling and there is a Linux utility **gprof** that will output a listing of routines and the amount of time spent in these routines. This does require that the code be compiled with special options to instrument the code and to produce a file, gmon.out, that will be written at the end of program execution.

The following listing shows how to build TORQUE with profiling enabled. Notice that the output file for pbs_mom will end up in the mom_priv directory because its startup code changes the default directory to this location.

```
# ./configure "CFLAGS=-pg -lgcov -fPIC"
# make -j5
# make install
# pbs_mom
... do some stuff for a while ...
# momctl -s
# cd /var/spool/torque/mom_priv
# gprof -b `which pbs_mom` gmon.out |less
#
```

Another way to see areas where a program is spending most of its time is with the valgrind program. The advantage of using valgrind is that the programs do not have to be specially compiled.

```
# valgrind --tool=callgrind pbs_mom
```

11.1.7 Stuck Jobs

If a job gets stuck in TORQUE, try these suggestions to resolve the issue.

- Use the qdel command to cancel the job.
- Force the MOM to send an obituary of the job ID to the server.

  ```
  > qsig -s 0 <JOBID>
  ```

- You can try clearing the stale jobs by using the momctl command on the compute nodes where the

jobs are still listed.

```
> momctl -c 58925 -h compute-5-20
```

- Setting the qmgr server setting mom_job_sync to *True* might help prevent jobs from hanging.

```
> qmgr -c "set server mom_job_sync = True"
```

To check and see if this is already set, use:

```
> qmgr -c "p s"
```

- If the suggestions above cannot remove the stuck job, you can try qdel -p. However, since the -p option purges all information generated by the job, this is not a recommended option unless the above suggestions fail to remove the stuck job.

```
> qdel -p <JOBID>
```

- The last suggestion for removing stuck jobs from compute nodes is to restart the pbs_mom.

For additional troubleshooting, run a tracejob on one of the stuck jobs. You can then create an online support ticket with the full server log for the time period displayed in the trace job.

11.1.8 Frequently Asked Questions (FAQ)

Cannot connect to server: error=15034

This error occurs in TORQUE clients (or their APIs) because TORQUE cannot find the server_name file and/or the **PBS_DEFAULT** environment variable is not set. The server_name file or **PBS_DEFAULT** variable indicate the pbs_server's hostname that the client tools should communicate with. The server_name file is usually located in TORQUE's local state directory. Make sure the file exists, has proper permissions, and that the version of TORQUE you are running was built with the proper directory settings. Alternatively you can set the **PBS_DEFAULT** environment variable. Restart TORQUE daemons if you make changes to these settings.

Deleting 'Stuck' Jobs

To manually delete a *stale* job which has no process, and for which the mother superior is still alive, sending a sig 0 with qsig will often cause MOM to realize the job is stale and issue the proper JobObit notice. Failing that, use momctl -c to forcefully cause MOM to purge the job. The following process should never be necessary:

- shut down the MOM on the mother superior node

- delete all files and directories related to the job from TORQUE_HOME/mom_priv/jobs
- restart the MOM on the mother superior node.

If the mother superior MOM has been lost and cannot be recovered (i.e, hardware or disk failure), a job running on that node can be purged from the output of qstat using the qdel -p command or can be removed manually using the following steps:

To remove job X:

1. Shutdown **pbs_server**.
```
> qterm
```
2. Remove job spool files.
```
> rm TORQUE_HOME/server_priv/jobs/X.SC
TORQUE_HOME/server_priv/jobs/X.JB
```
3. Restart **pbs_server**.
```
> pbs_server
```

Which user must run TORQUE?

TORQUE (**pbs_server** & **pbs_mom**) must be started by a user with root privileges.

Scheduler cannot run jobs - rc: 15003

For a scheduler, such as Moab or Maui, to control jobs with TORQUE, the scheduler needs to be run be a user in the server operators / managers list (see qmgr (set server operators / managers)). The default for the server operators / managers list is root@localhost. For TORQUE to be used in a grid setting with Silver, the scheduler needs to be run as root.

PBS_Server: pbsd_init, Unable to read server database

If this message is displayed upon starting **pbs_server** it means that the local database cannot be read. This can be for several reasons. The most likely is a version mismatch. Most versions of TORQUE can read each others' databases. However, there are a few incompatibilities between OpenPBS and TORQUE. Because of enhancements to TORQUE, it cannot read the job database of an OpenPBS server (job structure sizes have been altered to increase functionality). Also, a compiled in 32 bit mode cannot read a database generated by a 64 bit **pbs_server** and vice versa.

To reconstruct a database (excluding the job database), first print out the old data with this command:

```
%> qmgr -c "p s"
#
# Create queues and set their attributes.
#
#
# Create and define queue batch
#
create queue batch
set queue batch queue_type = Execution
set queue batch acl_host_enable = False
set queue batch resources_max.nodect = 6
set queue batch resources_default.nodes = 1
set queue batch resources_default.walltime = 01:00:00
```

```
set queue batch resources_available.nodect = 18
set queue batch enabled = True
set queue batch started = True
#
# Set server attributes.
#
set server scheduling = True
set server managers = griduser@oahu.icluster.org
set server managers += scott@*.icluster.org
set server managers += wightman@*.icluster.org
set server operators = griduser@oahu.icluster.org
set server operators += scott@*.icluster.org
set server operators += wightman@*.icluster.org
set server default_queue = batch
set server log_events = 511
set server mail from = adm
```

Copy this information somewhere. Restart **pbs_server** with the following command:

```
> pbs_server -t create
```

When it to prompts to overwrite the previous database enter 'y' then enter the data exported by the *qmgr* command with a command similar to the following:

```
> cat data | qmgr
```

Restart **pbs_server** without the flags:

```
> qterm
> pbs_server
```

This will reinitialize the database to the current version. Note that reinitializing the server database will reset the next jobid to 1.

---

qsub will not allow the submission of jobs requesting many processors

TORQUE's definition of a node is context sensitive and can appear inconsistent. The qsub '**-l nodes=<X>**' expression can at times indicate a request for **X** processors and other time be interpreted as a request for **X** nodes. While **qsub** allows multiple interpretations of the keyword *nodes*, aspects of the TORQUE server's logic are not so flexible. Consequently, if a job is using '-l nodes' to specify processor count and the requested number of processors exceeds the available number of physical nodes, the server daemon will reject the job.

To get around this issue, the server can be told it has an *inflated* number of nodes using the **resources_available** attribute. To take affect, this attribute should be set on both the server and the associated queue as in the example below. See resources_available for more information.

```
> qmgr
Qmgr: set server resources_available.nodect=2048
Qmgr: set queue batch resources_available.nodect=2048
```

The **pbs_server** daemon will need to be restarted before these changes will take affect.

---

qsub reports 'Bad UID for job execution'

```
[guest@login2]$ qsub test.job
```

```
qsub: Bad UID for job execution
```

Job submission hosts must be explicitly specified within TORQUE or enabled via RCmd security mechanisms in order to be trusted. In the example above, the host 'login2' is not configured to be trusted. This process is documented in Configuring Job Submission Hosts.

Why does my job keep bouncing from running to queued?

There are several reasons why a job will fail to start. Do you see any errors in the MOM logs? Be sure to increase the loglevel on MOM if you don't see anything. Also be sure TORQUE is configured with **--enable-syslog** and look in /var/log/messages (or wherever your syslog writes).

Also verify the following on all machines:

- DNS resolution works correctly with matching forward and reverse
- time is synchronized across the head and compute nodes
- user accounts exist on all compute nodes
- user home directories can be mounted on all compute nodes
- prologue scripts (if specified) exit with 0

If using a scheduler such as Moab or Maui, use a scheduler tool such as checkjob to identify job start issues.

How do I use PVM with TORQUE?

- Start the master pvmd on a compute node and then add the slaves
- mpiexec can be used to launch slaves using rsh or ssh (use export PVM_RSH=/usr/bin/ssh to use ssh)

Access can be managed by rsh/ssh without passwords between the batch nodes, but denying it from anywhere else, including the interactive nodes. This can be done with xinetd and sshd configuration (root is allowed to ssh everywhere). This way, the pvm daemons can be started and killed from the job script.

The problem is that this setup allows the users to bypass the batch system by writing a job script that uses rsh/ssh to launch processes on the batch nodes. If there are relatively few users and they can more or less be trusted, this setup can work.

My build fails attempting to use the TCL library

TORQUE builds can fail on TCL dependencies even if a version of TCL is available on the system. TCL is only utilized to support the **xpbsmon** client. If your site does not use this tool (most sites do not use **xpbsmon**), you can work around this failure by rerunning **configure** with the **--disable-gui** argument.

My job will not start, failing with the message 'cannot send job to mom, state=PRERUN'

If a node crashes or other major system failures occur, it is possible that a job may be *stuck* in a corrupt state on a compute node. TORQUE 2.2.0 and higher automatically handle this when the **mom_job_sync** parameter is set via qmgr (the default). For earlier versions of TORQUE, set this parameter and restart the **pbs_mom** daemon.

This error can also occur if not enough free space is available on the partition that holds TORQUE.

I want to submit and run jobs as root

While this can be a *very* bad idea from a security point of view, in some restricted environments this can be quite useful and can be enabled by setting the acl_roots parameter via qmgr command as in the following example:

```
qmgr
> qmgr -c 's s acl_roots+=root@*'
```

How do I determine what version of Torque I am using?

There are times when you want to find out what version of Torque you are using. An easy way to do this is to run the following command:

```
qmgr
> qmgr -c "p s" | grep pbs_ver
```

See Also

- PBSCOREDUMP parameter

11.2 Compute Node Health Check

11.2.1 Compute Node Health Check Overview

TORQUE provides the ability to perform health checks on each compute node. If these checks fail, a failure message can be associated with the node and routed to the scheduler. Schedulers (such as Moab) can forward this information to administrators by way of scheduler triggers, make it available through scheduler diagnostic commands, and automatically mark the node *down* until the issue is resolved. (See the RMMSGIGNORE parameter in Appendix F of the Moab Workload Manager Administrator's Guide for more information.)

11.2.2 Configuring MOM's to Launch a Health Check

The health check feature is configured via the **pbs_mom** config file using the parameters described below:

Parameter	Format	Default	Description
node_check_script	<STRING>	N/A	(required) specifies the fully qualified pathname of the health check script to run
node_check_interval	<INTEGER>	1	(optional) specifies the number of MOM intervals between health checks (by default, each MOM interval is 45 seconds long - this is controlled via the **DEFAULT_SERVER_STAT_UPDATES** #define located in `TORQUE_HOME/src/resmom/mom_main.c`)

11.2.3 Creating the Health Check Script

The health check script is executed directly by the pbs_mom daemon under the *root* user id. It must be accessible from the compute node and may be a script or compile executable program. It may make any needed system calls and execute any combination of system utilities but should not execute resource manager client commands. Also, as of TORQUE 1.0.1, the pbs_mom daemon blocks until the health check is completed and does not possess a built-in timeout. Consequently, it is advisable to keep the launch script execution time short and verify that the script will not block even under failure conditions.

If the script detects a failure, it should return the keyword '**ERROR**' to stdout followed by an error message. When a failure is detected, the **ERROR** keyword should be printed to stdout before any other data. The message (up to 1024 characters) immediately following the **ERROR** keyword must all be contained on the same line. The message is assigned to the node attribute 'message' of the associated node.

11.2.4 Adjusting Node State Based on the Health Check Output

If the health check reports an error, the node attribute 'message' is set to the error string returned. Cluster schedulers can be configured to adjust a given node's state based on this information. For example, by default, Moab sets a node's state to down if a node error message is detected and restores the state as soon as the failure disappears.

11.2.5 Example Health Check Script

As mentioned, the health check can be a shell script, PERL, Python, C-executable, or anything which can be executed from the command line capable of setting STDOUT. The example below demonstrates a very simple health check:

```
#!/bin/sh

/bin/mount | grep global

if [ $? != "0" ]
```

```
  then
    echo "ERROR cannot locate filesystem global"
  fi
```

11.3 Debugging

11.3.1 Debugging Facilities

TORQUE supports a number of diagnostic and debug options including the following:

- **PBSDEBUG** environment variable - If set to 'yes', this variable will prevent **pbs_server**, **pbs_mom**, and/or **pbs_sched** from backgrounding themselves allowing direct launch under a debugger. Also, some client commands will provide additional diagnostic information when this value is set.
- **PBSLOGLEVEL** environment variable - Can be set to any value between 0 and 7 and specifies the logging verbosity level (default = 0)
- **PBSCOREDUMP** environment variable - If set, it will cause the offending resource manager daemon to create a core file if a **SIGSEGV**, **SIGILL**, **SIGFPE**, **SIGSYS**, or **SIGTRAP** signal is received. The core dump will be placed in the daemon's home directory (`$PBSHOME/mom_priv` for **pbs_mom**).
- **NDEBUG** #define - if set at build time, will cause additional low-level logging information to be output to stdout for **pbs_server** and **pbs_mom** daemons.
- **tracejob** reporting tool - can be used to collect and report logging and accounting information for specific jobs

11.3.2 TORQUE Error Codes

Error Code Name	Number	Description
PBSE_NONE	15000	No error
PBSE_UNKJOBID	15001	Unknown job identifier
PBSE_NOATTR	15002	Undefined attribute
PBSE_ATTRRO	15003	Attempt to set READ ONLY attribute
PBSE_IVALREQ	15004	Invalid request
PBSE_UNKREQ	15005	Unknown batch request
PBSE_TOOMANY	15006	Too many submit retries
PBSE_PERM	15007	No permission
PBSE_BADHOST	15008	Access from host not allowed
PBSE_JOBEXIST	15009	Job already exists
PBSE_SYSTEM	15010	System error occurred
PBSE_INTERNAL	15011	Internal server error occurred
PBSE_REGROUTE	15012	Parent job of dependent in rte queue
PBSE_UNKSIG	15013	Unknown signal name
PBSE_BADATVAL	15014	Bad attribute value
PBSE_MODATRRUN	15015	Cannot modify attribute in run state
PBSE_BADSTATE	15016	Request invalid for job state
PBSE_UNKQUE	15018	Unknown queue name
PBSE_BADCRED	15019	Invalid credential in request
PBSE_EXPIRED	15020	Expired credential in request

PBSE_QUNOENB	15021	Queue not enabled
PBSE_QACESS	15022	No access permission for queue
PBSE_BADUSER	15023	Bad user - no password entry
PBSE_HOPCOUNT	15024	Max hop count exceeded
PBSE_QUEEXIST	15025	Queue already exists
PBSE_ATTRTYPE	15026	Incompatible queue attribute type
PBSE_QUEBUSY	15027	Queue busy (not empty)
PBSE_QUENBIG	15028	Queue name too long
PBSE_NOSUP	15029	Feature/function not supported
PBSE_QUENOEN	15030	Cannot enable queue,needs add def
PBSE_PROTOCOL	15031	Protocol (ASN.1) error
PBSE_BADATLST	15032	Bad attribute list structure
PBSE_NOCONNECTS	15033	No free connections
PBSE_NOSERVER	15034	No server to connect to
PBSE_UNKRESC	15035	Unknown resource
PBSE_EXCQRESC	15036	Job exceeds queue resource limits
PBSE_QUENODFLT	15037	No default queue defined
PBSE_NORERUN	15038	Job not rerunnable
PBSE_ROUTEREJ	15039	Route rejected by all destinations
PBSE_ROUTEEXPD	15040	Time in route queue expired
PBSE_MOMREJECT	15041	Request to MOM failed
PBSE_BADSCRIPT	15042	(qsub) Cannot access script file
PBSE_STAGEIN	15043	Stage-In of files failed
PBSE_RESCUNAV	15044	Resources temporarily unavailable
PBSE_BADGRP	15045	Bad group specified
PBSE_MAXQUED	15046	Max number of jobs in queue
PBSE_CKPBSY	15047	Checkpoint busy, may be retries
PBSE_EXLIMIT	15048	Limit exceeds allowable
PBSE_BADACCT	15049	Bad account attribute value
PBSE_ALRDYEXIT	15050	Job already in exit state
PBSE_NOCOPYFILE	15051	Job files not copied
PBSE_CLEANEDOUT	15052	Unknown job id after clean init
PBSE_NOSYNCMSTR	15053	No master in sync set
PBSE_BADDEPEND	15054	Invalid dependency

PBSE_DUPLIST	15055	Duplicate entry in list
PBSE_DISPROTO	15056	Bad DIS based request protocol
PBSE_EXECTHERE	15057	Cannot execute there
PBSE_SISREJECT	15058	Sister rejected
PBSE_SISCOMM	15059	Sister could not communicate
PBSE_SVRDOWN	15060	Requirement rejected -server shutting down
PBSE_CKPSHORT	15061	Not all tasks could checkpoint
PBSE_UNKNODE	15062	Named node is not in the list
PBSE_UNKNODEATR	15063	Node-attribute not recognized
PBSE_NONODES	15064	Server has no node list
PBSE_NODENBIG	15065	Node name is too big
PBSE_NODEEXIST	15066	Node name already exists
PBSE_BADNDATVAL	15067	Bad node-attribute value
PBSE_MUTUALEX	15068	State values are mutually exclusive
PBSE_GMODERR	15069	Error(s) during global modification of nodes
PBSE_NORELYMOM	15070	Could not contact Mom
PBSE_NOTSNODE	15071	No time-shared nodes

See Also

- Troubleshooting Guide

Appendix A: Commands Overview

A.1 Client Commands

Command	Description
momctl	manage/diagnose MOM (node execution) daemon
pbsdsh	launch tasks within a parallel job
pbsnodes	view/modify batch status of compute nodes
qalter	modify queued batch jobs
qchkpt	checkpoint batch jobs
qdel	delete/cancel batch jobs
qhold	hold batch jobs
qmgr	manage policies and other batch configuration
qrerun	rerun a batch job
qrls	release batch job holds
qrun	start a batch job
qsig	send a signal to a batch job
qstat	view queues and jobs
qsub	submit jobs
qterm	shutdown pbs server daemon
tracejob	trace job actions and states recorded in TORQUE logs

A.2 Binary Executables

Command	Description
pbs_iff	interprocess authentication service
pbs_mom	start MOM (node execution) daemon
pbs_server	start server daemon
pbs_track	tell pbs_mom to track a new process

See Also

- MOM Configuration
- Server Parameters

momctl

(PBS Mom Control)

Synopsis

```
momctl -c { <JOBID> | all }
momctl -C
momctl -d { <INTEGER> | <JOBID> }
momctl -f <FILE>
momctl -h <HOST>[,<HOST>]...
momctl -p <PORT_NUMBER>
momctl -q <ATTRIBUTE>
momctl -r { <FILE> | LOCAL:<FILE> }
momctl -s
```

Overview

The **momctl** command allows remote shutdown, reconfiguration, diagnostics, and querying of the pbs_mom daemon.

Format

-c — Clear	
Format:	{ <JOBID> | **all** }
Default:	---
Description:	Clear stale job information
Example:	`momctl -h node1 -c 15406`

-C — Cycle	
Format:	---
Default:	---
Description:	Cycle pbs_mom(s)
Example:	`momctl -h node1 -C` Cycle pbs_mom on node1

-d — Diagnose	
Format:	{ <INTEGER> | <JOBID> }
Default:	0
Description:	Diagnose mom(s) See the Diagnose Detail table below for more information.
Example:	`momctl -h node1 -d 2` Print level 2 and lower diagnose information for the MOM on node1

-f — Host File	

Format:	<FILE>
Default:	---
Description:	A file contain only comma or whitespace (space, tab, or new line) delimited hostnames
Example:	`momctl -f hosts.txt -d` Print diagnose information for the moms running on the hosts specified in hosts.txt

-h — Host List	
Format:	<HOST>[,<HOST>]...
Default:	localhost
Description:	A comma separated list of hosts
Example:	`momctl -h node1,node2,node3 -d` Print diagnose information for the moms running on node1, node2 and node3

-p — Port	
Format:	<PORT_NUMBER>
Default:	TORQUE's default port number
Description:	The port number for the specified mom(s)
Example:	`momctl -p 5455 -h node1 -d` Request diagnose information over port 5455 on node1

-q — Query	
Format:	<ATTRIBUTE>
Default:	---
Description:	Query <ATTRIBUTE> on specified MOM (where <ATTRIBUTE> is a property listed by pbsnodes -a)
Example:	`momctl -q physmem` Print the amount of physmem on localhost

-r — Reconfigure	
Format:	{ <FILE> | **LOCAL:**<FILE> }
Default:	---
Description:	Reconfigure mom(s) with remote or local config file, <FILE>. This does not work if $remote_reconfig is not set to true when the MOM is started.
Example:	`momctl -r /home/user1/new.config -h node1` Reconfigure MOM on node1 with /home/user1/new.config on node1

-s — Shutdown	

Format:	
Default:	---
Description:	Shutdown pbs_mom
Example:	`momctl -s` Terminates pbs_mom process on localhost

Query Attributes

- **arch** — node hardware architecture
- **availmem** — available RAM
- **loadave** — 1 minute load average
- **ncpus** — number of CPUs available on the system
- **netload** — total number of bytes transferred over all network interfaces
- **nsessions** — number of sessions active
- **nusers** — number of users active
- **physmem** — configured RAM
- **sessions** — list of active sessions
- **totmem** — configured RAM plus configured swap

Diagnose Detail

Level	Description
0	Display the following information: • Local hostname • Expected server hostname • Execution version • MOM home directory • MOM config file version (if specified) • Duration MOM has been executing • Duration since last request from **pbs_server** daemon • Duration since last request to **pbs_server** daemon • RM failure messages (if any) • Log verbosity level • Local job list
1	All information for level 0 plus the following: • Interval between updates sent to server • Number of initialization messages sent to **pbs_server** daemon • Number of initialization messages received from **pbs_server** daemon • Prolog/epilog alarm time • List of trusted clients
2	All information from level 1 plus the following: • PID • Event alarm status

Example 1: MOM Diagnostics

```
> momctl -d 1

Host: nsrc/nsrc.fllcl.com   Server: 10.10.10.113   Version:
torque_1.1.0p4
HomeDirectory:          /usr/spool/PBS/mom_priv
ConfigVersion:          147
MOM active:             7390 seconds
Last Msg From Server:   7389 seconds (CLUSTER_ADDRS)
Server Update Interval: 20 seconds
Server Update Interval: 20 seconds
Init Msgs Received:     0 hellos/1 cluster-addrs
Init Msgs Sent:         1 hellos
LOGLEVEL:               0 (use SIGUSR1/SIGUSR2 to adjust)
Prolog Alarm Time:      300 seconds
Trusted Client List:    12.14.213.113,127.0.0.1
JobList:                NONE

diagnostics complete
```

Example 2: System Shutdown

```
> momctl -s -f /opt/clusterhostfile

shutdown request successful on node001
shutdown request successful on node002
shutdown request successful on node003
shutdown request successful on node004
shutdown request successful on node005
shutdown request successful on node006
```

pbsdsh

distribute tasks to nodes under pbs

Synopsis

```
pbsdsh [-c copies] [-o] [-s] [-u] [-v] program [args]
pbsdsh [-n node] [-o] [-s] [-u] [-v] program [args]
pbsdsh [-h nodename] [-o] [-v] program [args]
```

Description

Executes (spawns) a normal Unix program on one or more nodes under control of the Portable Batch System, PBS. Pbsdsh uses the Task Manager API (see tm_spawn(3)) to distribute the program on the allocated nodes.

When run without the -c or the -n option, pbsdsh will spawn the program on all nodes allocated to the PBS job. The spawns take place concurrently - all execute at (about) the same time.

Users will find the PBS_TASKNUM, PBS_NODENUM, and the PBS_VNODENUM environmental variables useful. They contain the TM task id, the node identifier, and the cpu (virtual node) identifier.

Options

-c *copies*
: The program is spawned on the first Copies nodes allocated. This option is mutual exclusive with -n.

-h *hostname*
: The program is spawned on the node specified.

-n *node*
: The program is spawned on one node which is the n-th node allocated. This option is mutual exclusive with -c.

-o
: Capture stdout of the spawned program. Normally stdout goes to the job's output.

-s
: If this option is given, the program is run in turn on each node, one after the other.

-u
: The program is run once on each node (unique). This ignores the number of allocated processors on a given node.

-v
: Verbose output about error conditions and task exit status is produced.

Operands

The first operand, program, is the program to execute.

Additional operands are passed as arguments to the program.

Standard Error

The pbsdsh command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the command, the exit status will be a value of zero.

If the pbsdsh command fails to process any operand, or fails to contact the MOM daemon on the localhost the command exits with a value greater than zero.

See Also

qsub(1B), tm_spawn(3B)

pbsnodes

pbs node manipulation

Synopsis

```
pbsnodes [-{a|x}] [-q] [-s server] [node|:property]
pbsnodes -l [-q] [-s server] [state] [nodename|:property ...]
pbsnodes [-{c|d|o|r}] [-q] [-s server] [-n -l] [-N "note"] [node|:property]
```

Description

The pbsnodes command is used to mark nodes down, free or offline. It can also be used to list nodes and their state. Node information is obtained by sending a request to the PBS job server. Sets of nodes can be operated on at once by specifying a node property prefixed by a colon. See Node States for more information on node states.

Nodes do not exist in a single state, but actually have a set of states. For example, a node can be simultaneously "busy" and "offline". The "free" state is the absense of all other states and so is never combined with other states.

In order to execute pbsnodes with other than the -a or -l options, the user must have PBS Manager or Operator privilege.

Options

-a
: All attributes of a node or all nodes are listed. This is the default if no flag is given.

-x
: Same as -a, but the output has an XML-like format.

-c
: Clear OFFLINE from listed nodes.

-d
: Print MOM diagnosis on the listed nodes. Not yet implemented. Use momctl instead.

-o
: Add the OFFLINE state. This is different from being marked DOWN. OFFLINE prevents new jobs from running on the specified nodes. This gives the administrator a tool to hold a node out of service without changing anything else. The OFFLINE state will never be set or cleared automatically by pbs_server; it is purely for the manager or operator.

-p
: Purge the node record from pbs_server. Not yet implemented.

-r
: Reset the listed nodes by clearing OFFLINE and adding DOWN state. pbs_server will ping the node and, if they communicate correctly, free the node.

-l
: List node names and their state. If no state is specified, only nodes in the DOWN, OFFLINE, or UNKNOWN states are listed. Specifying a state string acts as an output filter. Valid state strings are "active", "all", "busy", "down", "free", "offline", "unknown", and "up".

  Using **all** displays all nodes and their attributes.

  Using **active** displays all nodes which are job-exclusive, job-sharing, or busy.

  Using **up** displays all nodes in an "up state". Up states include job-execlusive, job-sharing, reserve, free, busy and time-shared.

  All other strings display the nodes which are currently in the state indicated by the string.

-N
: Specify a "note" attribute. This allows an administrator to add an arbitrary annotation to the listed nodes. To clear a note, use -N "" or -N n.

-n

Show the "note" attribute for nodes that are DOWN, OFFLINE, or UNKNOWN. This option requires -l.

-q

Supress all error messages.

-s

Specify the PBS server's hostname or IP address.

See Also

pbs_server(8B) and the PBS External Reference Specification

qalter

alter batch job

Synopsis

```
qalter [-a date_time][-A account_string][-c interval][-e path_name]
       [-h hold_list][-j join_list][-k keep_list][-l resource_list]
       [-m mail_options][-M mail_list][-N name][-o path_name]
       [-p priority][-q ][-r y|n][-S path_name_list][-u user_list]
       [-v variable_list][-W additional_attributes]
       [-t array_range]
       job_identifier ...
```

Description

The qalter command modifies the attributes of the job or jobs specified by job_identifier on the command line. Only those attributes listed as options on the command will be modified. If any of the specified attributes cannot be modified for a job for any reason, none of that job's attributes will be modified.

The qalter command accomplishes the modifications by sending a Modify Job batch request to the batch server which owns each job.

Options

-a *date_time*
Replaces the time at which the job becomes eligible for execution. The date_time argument syntax is:

`[[[[CC]YY]MM]DD]hhmm[.SS].`

If the month, MM, is not specified, it will default to the current month if the specified day DD, is in the future. Otherwise, the month will be set to next month. Likewise, if the day, DD, is not specified, it will default to today if the time hhmm is in the future. Otherwise, the day will be set to tomorrow.

This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-A *account_string*
Replaces the the account string associated with the job. This attribute cannot be altered once the job has begun execution.

-c *checkpoint_interval*
Replaces the the interval at which the job will be checkpointed. If the job executes upon a host which does not support checkpointing, this option will be ignored.

The interval argument is specified as:

n
No checkpointing is to be performed

s
Checkpointing is to be performed only when the server executing the job is shutdown.

c
Checkpointing is to be performed at the default minimum cpu time for the queue from which the job is executing.

c=minutes
Checkpointing is performed at intervals of the specified amount of time in minutes. Minutes are the number of minutes of CPU time used, not necessarily clock time. This value must be greater than zero. If the number is less than the default checkpoint time, the default time will be used.

This attribute can be altered once the job has begun execution, but the new value does not take effect unless the job is rerun.

-e *path_name*

Replaces the path to be used for the standard error stream of the batch job. The path argument is of the form:

```
[hostname:]path_name
```

where hostname is the name of a host to which the file will be returned and path_name is the path name on that host in the syntax recognized by POSIX 1003.1. The argument will be interpreted as follows:

path_name
: Where path_name is not an absolute path name, then the qalter command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component.

hostname:path_name
: Where path_name is not an absolute path name, then the qalter command will not expand the path name. The execution server will expand it relative to the home directory of the user on the system specified by hostname.

This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-h *hold_list*

Updates the the types of holds on the job. The hold_list argument is a string of one or more of the following characters:

u
: Add the USER type hold.

s
: Add the SYSTEM type hold if the user has the appropriate level of privilege. (Typically reserved to the batch administrator.)

o
: Add the OTHER (or OPERATOR) type hold if the user has the appropriate level of privilege. (Typically reserved to the batch administrator and batch operator.)

n
: Set to none and clear the hold types which could be applied with the users level of privilege.

Repetition of characters is permitted, but "n" may not appear in the same option argument with the other three characters. This attribute can be altered once the job has begun execution, but the hold will not take effect unless the job is rerun.

-j *join*

Declares which standard streams of the job will be merged together. The join argument value may be the characters "oe" and "eo", or the single character "n".

A argument value of oe directs that the standard output and standard error streams of the job will be merged, intermixed, and returned as the standard output. A argument value of eo directs that the standard output and standard error streams of the job will be merged, intermixed, and returned as the standard error.

A value of n directs that the two streams will be two separate files. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-k *keep*

Defines which if either of standard output or standard error of the job will be retained on the execution host. If set for a stream, this option overrides the path name for that stream.

The argument is either the single letter "e", "o", or "n", or one or more of the letters "e" and "o" combined in either order.

n
No streams are to be retained.

e
The standard error stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:
job_name.esequence
where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier.

o
The standard output stream is to be retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:
job_name.osequence
where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier.

eo
Both the standard output and standard error streams will be retained.

oe
Both the standard output and standard error streams will be retained.

This attribute cannot be altered once the job has begun execution.

-l *resource_list*
Modifies the list of resources that are required by the job. The Resource_List argument is in the following syntax:

resource_name[=[value]][,resource_name[=[value]],...]

If a requested modification to a resource would exceed the resource limits for jobs in the current queue, the server will reject the request.

If the job is running, only certain resources can be altered. Which resources can be altered in the run state is system dependent. A user may only lower the limit for those resources.

-m *mail_options*
Replaces the set of conditions under which the execution server will send a mail message about the job. The mail_options argument is a string which consists of the single character "n", or one or more of the characters "a", "b", and "e".

If the character "n" is specified, no mail will be sent.

For the letters "a", "b", and "e":

a
mail is sent when the job is aborted by the batch system.

b
mail is sent when the job begins execution.

e
mail is sent when the job ends.

-M *user_list*
Replaces the list of users to whom mail is sent by the execution server when it sends mail about the job.

The user_list argument is of the form:

user[@host][,user[@host],...]

-N *name*

Renames the job. The name specified may be up to and including 15 characters in length. It must consist of printable, non white space characters with the first character alphabetic.

-o *path*

Replaces the path to be used for the standard output stream of the batch job. The path argument is of the form:

[hostname:]path_name

where hostname is the name of a host to which the file will be returned and path_name is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:

path_name

Where path_name is not an absolute path name, then the qalter command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component.

hostname:path_name

Where path_name is not an absolute path name, then the qalter command will not expand the path name. The execution server will expand it relative to the home directory of the user on the system specified by hostname.

This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-p *priority*

Replaces the priority of the job. The priority argument must be a integer between -1024 and +1023 inclusive.

This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-r *[y|n]*

Declares whether the job is rerunable. See the qrerun command. The option argument c is a single character. PBS recognizes the following characters: y and n.

If the argument is "y", the job is marked rerunable. If the argument is "n", the job is marked as not rerunable.

-S *path*

Declares the shell that interprets the job script.

The option argument path_list is in the form:

```
path[@host][,path[@host],...]
```

Only one path may be specified for any host named. Only one path may be specified without the corresponding host name. The path selected will be the one with the host name that matched the name of the execution host. If no matching host is found, then the path specified (without a host) will be selected.

If the -S option is not specified, the option argument is the null string, or no entry from the path_list is selected, the execution will use the login shell of the user on the execution host.

This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.

-t *array_range*

The *array_range* argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimted list. Examples: `-t 1-100` or `-t 1,10,50-100`

If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

An optional **slot limit** can be specified to limit the amount of jobs that can run concurrently in the job array. The default value is unlimited. The slot limit must be the last thing specified in the *array_request* and is delimited from the array by a percent sign (%).

```
qalter weatherSimulationArray[] -t %20
```

Here, the array `weatherSimulationArray[]` is configured to allow a maximum of 20 concurrently running jobs.

Slot limits can be applied at job submit time with qsub, or can be set in a global server parameter policy with max_slot_limit

-u *user_list*

Replaces the user name under which the job is to run on the execution system.

The user_list argument is of the form:

```
user[@host][,user[@host],...]
```

Only one user name may be given for per specified host. Only one of the user specifications may be supplied without the corresponding host specification. That user name will be used for execution on any host not named in the argument list.

This attribute cannot be altered once the job has begun execution.

-W *additional_attributes*

The -W option allows for the modification of additional job attributes.

Note if white space occurs anywhere within the option argument string or the equal sign, "=", occurs within an attribute_value string, then the string must be enclosed with either single or double quote marks.

PBS currently supports the following attributes within the -W option.

depend=dependency_list

Redefines the dependencies between this and other jobs. The dependency_list is in the form:

```
type[:argument[:argument...][,type:argument...]
```

The argument is either a numeric count or a PBS job id accord ing to type. If argument is a count, it must be greater than 0. If it is a job id and is not fully specified in the form: seq_number.server.name, it will be expanded according to the default server rules. If argument is null (the preceding colon need not be specified), the dependency of the corresponding type is cleared (unset).

synccount:count

This job is the first in a set of jobs to be executed at the same time. Count is the number of additional jobs in the set.

syncwith:jobid

This job is an additional member of a set of jobs to be executed at the same time. In the above and following dependency types, jobid is the job identifier of the first job in the set.

after:jobid [:jobid...]

This job may be scheduled for execution at any point after jobs jobid have started execution.

afterok:jobid [:jobid...]

This job may be scheduled for execution only after jobs jobid have terminated with no errors. See the csh warning under "Extended Description".

afternotok:jobid [:jobid...]

This job may be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning under "Extended Description".

afterany:jobid [:jobid...]

This job may be scheduled for execution after jobs jobid have terminated, with or without

errors.

on:count
: This job may be scheduled for execution after count dependencies on other jobs have been satisfied. This dependency is used in conjunction with any of the 'before' dependencies shown below. If job A has on:2, it will wait for two jobs with 'before' dependencies on job A to be fulfilled before running.

before:jobid [:jobid...]
: When this job has begun execution, then jobs jobid... may begin.

beforeok:jobid [:jobid...]
: If this job terminates execution without errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforenotok:jobid [:jobid...]
: If this job terminates execution with errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforeany:jobid [:jobid...]
: When this job terminates execution, jobs jobid... may begin.

If any of the before forms are used, the job referenced by jobid must have been submitted with a dependency type of on.

If any of the before forms are used, the jobs referenced by jobid must have the same owner as the job being altered. Otherwise, the dependency will not take effect.

Error processing of the existence, state, or condition of the job specified to qalter is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the job will be deleted by the server. Mail will be sent to the job submitter stating the error.

group_list=g_list
: Alters the group name under which the job is to run on the execution system.

The g_list argument is of the form:

```
group[@host][,group[@host],...]
```

Only one group name may be given per specified host. Only one of the group specifications may be supplied without the corresponding host specification. That group name will used for execution on any host not named in the argument list.

stagein=file_list
stageout=file_list
: Alters which files are staged (copied) in before job start or staged out after the job completes execution. The file_list is in the form:

```
local_file@hostname:remote_file[,...]
```

The name local_file is the name on the system where the job executes. It may be an absolute path or a path relative to the home directory of the user. The name remote_file is the destination name on the host specified by hostname. The name may be absolute or relative to the users home directory on the destination host.

Operands

The qalter command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Standard Error

Any error condition, either in processing the options or the operands, or any error received in reply to the batch requests will result in an error message being written to standard error.

Exit Status

Upon successful processing of all the operands presented to the qalter command, the exit status will be a value of zero.

If the qalter command fails to process any operand, the command exits with a value greater than zero.

See Also

Batch Environment Services , *qdel* , *qhold* , *qmove* , *qrls* , *qsub* , *touch*

Copyright

qchkpt

checkpoint pbs batch jobs

Synopsis

```
qchkpt <JOBID>[ <JOBID>] ...
```

Description

The qchkpt command requests that the PBS Mom generate a checkpoint file for a running job.

This is an extension to POSIX.2d.

The qchkpt command sends a Chkpt Job batch request to the server as described in the general section.

Options

None

Operands

The qchkpt command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qchkpt 3233     request a checkpoint for job 3233
```

Standard Error

The qchkpt command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qchkpt command, the exit status will be a value of zero.

If the qchkpt command fails to process any operand, the command exits with a value greater than zero.

See Also

```
qhold(1B), qrls(1B), qalter(1B), qsub(1B), pbs_alterjob(3B), pbs_holdjob(3B),
  pbs_rlsjob(3B), pbs_job_attributes(7B), pbs_resources_unicos8(7B)
```

qdel

(delete job)

Synopsis

```
qdel [{-m <message>|-p|-W <delay>|-t <array_range>}] <JOBID>[ <JOBID>]... | 'all' | 'ALL'
```

Description

The qdel command deletes jobs in the order in which their job identifiers are presented to the command. A job is deleted by sending a Delete Job batch request to the batch server that owns the job. A job that has been deleted is no longer subject to management by batch services.

A batch job may be deleted by its owner, the batch operator, or the batch administrator.

A batch job being deleted by a server will be sent a SIGTERM signal following by a SIGKILL signal. The time delay between the two signals is an attribute of the execution queue from which the job was run (set table by the administrator). This delay may be overridden by the -W option.

See the PBS ERS section 3.1.3.3, "Delete Job Request", for more information.

Options

-w *delay*
: Specify the wait delay between the sending of the SIGTERM and SIGKILL signals. The argument is the length of time in seconds of the delay.

-p *purge*
: Forcibly purge the job from the server. This should only be used if a running job will not exit because its allocated nodes are unreachable. The admin should make every attempt at resolving the problem on the nodes. If a job's mother superior recovers after purging the job, any epilogue scripts may still run. This option is only available to a batch operator or the batch administrator.

-m *message*
: Specify a comment to be included in the email. The argument message specifies the comment to send. This option is only available to a batch operator or the batch administrator.

-t *array_range*
: The *array_range* argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimted list. Examples: `-t 1-100` or `-t 1,10,50-100`

  If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

Operands

The qdel command accepts one or more job_identifier operands of the form:

```
  sequence_number[.server_name][@server]
```

or

```
  all
```

Examples

```
> qdel 1324
> qdel 1324-3    To delete one job of a job array
> qdel all       To delete all jobs (Version 2.3.0 and later)
```

Standard Error

The qdel command will write a diagnostic messages to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qdel command, the exit status will be a value of zero.

If the qdel command fails to process any operand, the command exits with a value greater than zero.

See Also

qsub(1B), qsig(1B), and pbs_deljob(3B)

qhold

(hold job)

Synopsis

```
qhold [{-h <HOLD LIST>|-t <array_range>}] <JOBID>[ <JOBID>] ...
```

Description

The qhold command requests that the server place one or more holds on a job. A job that has a hold is not eligible for execution. There are three supported holds: USER, OTHER (also known as operator), and SYSTEM.

A user may place a USER hold upon any job the user owns. An "operator", who is a user with "operator privilege," may place ether an USER or an OTHER hold on any job. The batch administrator may place any hold on any job.

If no -h option is given, the USER hold will be applied to the jobs described by the job_identifier operand list.

If the job identified by job_identifier is in the queued, held, or waiting states, then the hold type is added to the job. The job is then placed into held state if it resides in an execution queue.

If the job is in running state, then the following additional action is taken to interrupt the execution of the job. If checkpoint / restart is supported by the host system, requesting a hold on a running job will (1) cause the job to be checkpointed, (2) the resources assigned to the job will be released, and (3) the job is placed in the held state in the execution queue.

If checkpoint / restart is not supported, qhold will only set the the requested hold attribute. This will have no effect unless the job is rerun with the qrerun command.

Options

-h *hold_list*

The hold_list argument is a string consisting of one or more of the letters "u", "o", or "s" in any combination. The hold type associated with each letter is:

- u - USER
- o - OTHER
- s - SYSTEM

-t *array_range*

The *array_range* argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimted list. Examples: `-t 1-100` or `-t 1,10,50-100`

If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

Operands

The qhold command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qhold -h u 3233     place user hold on job 3233
```

Standard Error

The qhold command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qhold command, the exit status will be a value of zero.

If the qhold command fails to process any operand, the command exits with a value greater than zero.

See Also

qrls(1B), qalter(1B), qsub(1B), pbs_alterjob(3B), pbs_holdjob(3B), pbs_rlsjob(3B), pbs_job_attributes(7B), pbs_resources_unicos8(7B)

qmgr

(PBS Queue Manager)

pbs batch system manager

Synopsis

qmgr [-a] [-c command] [-e] [-n] [-z] [server...]

Description

The **qmgr** command provides an administrator interface to query and configure batch system parameters.

The command reads directives from standard input. The syntax of each directive is checked and the appropriate request is sent to the batch server or servers.

The list or print subcommands of qmgr can be executed by general users. Creating or deleting a queue requries PBS Manager privilege. Setting or unsetting server or queue attributes requires PBS Operator or Manager privilege. **NOTE**: by default the user **root** is the only PBS Operator and Manager. To allow other users to be privileged, the server attributes **operators** and **managers** will need to be set (i.e., as root, issue 'qmgr -c 'set server managers += <USER1>@<HOST>'). See PBS Access Config for more information.

If **qmgr** is invoked without the *-c* option and standard output is connected to a terminal, qmgr will write a prompt to standard output and read a directive from standard input.

Commands can be abbreviated to their minimum unambiguous form. A command is terminated by a new line character or a semicolon, ";", character. Multiple commands may be entered on a single line. A command may extend across lines by escaping the new line character with a back-slash "\".

Comments begin with the # character and continue to end of the line. Comments and blank lines are ignored by qmgr.

Options

-a
: Abort **qmgr** on any syntax errors or any requests rejected by a server.

-c *command*
: Execute a single command and exit **qmgr.**

-e
: Echo all commands to standard output.

-n
: No commands are executed, syntax checking only is performed.

-z
: No errors are written to standard error.

Operands

The *server* operands identify the name of the batch server to which the administrator requests are sent. Each *server* conforms to the following syntax:
host_name[:port]
where **host_name** is the network name of the host on which the server is running and **port** is the port number to which to connect. If **port** is not specified, the default port number is used.

If *server* is not specified, the administrator requests are sent to the local server.

Standard Input

The **qmgr** command reads standard input for directives until end of file is reached, or the *exit* or *quit* directive is read.

Standard Output

If Standard Output is connected to a terminal, a command prompt will be written to standard output when qmgr is ready to read a directive.

If the *-e* option is specified, **qmgr** will echo the directives read from standard input to standard output.

Standard Error

If the *-z* option is not specified, the qmgr command will write a diagnostic message to standard error for each error occurrence.

Directive Syntax

A qmgr directive is one of the following forms:
```
command server [names] [attr OP value[,attr OP value,...]]
command queue [names] [attr OP value[,attr OP value,...]]
command node [names] [attr OP value[,attr OP value,...]]
```
where "command" is the command to perform on a object. Commands are:

active
: sets the active objects. If the active objects are specified, and the name is not given in a qmgr cmd the active object names will be used.

create
: is to create a new object, applies to queues and nodes.

delete
: is to destroy an existing object, applies to queues and nodes.

set
: is to define or alter attribute values of the object.

unset
: is to clear the value of attributes of the object. Note, this form does not accept an OP and value, only the attribute name.

list
: is to list the current attributes and associated values of the object.

print
: is to print all the queue and server attributes in a format that will be usable as input to the qmgr command.

names
: is a list of one or more names of specific objects The name list is in the form:
  [name][@server][,queue_name[@server]...]
with no intervening white space. The name of an object is declared when the object is first created. If the name is @server, then all the objects of specified type at the server will be effected.

attr
: specifies the name of an attribute of the object which is to be set or modified. If the attribute is one which consist of a set of resources, then the attribute is specified in the form:
  attribute_name**.**resource_name

OP
: operation to be performed with the attribute and its value:

  =
  : set the value of the attribute. If the attribute has a existing value, the current value is replaced with the new value.

  +=
  : increase the current value of the attribute by the amount in the new value.

  -=
  : decrease the current value of the attribute by the amount in the new value.

value
: the value to assign to an attribute. If the value includes white space, commas or other special characters, such as the # character, the value string must be inclosed in quote marks (").

The following are examples of qmgr directives:

```
create queue fast priority=10,queue_type=e,enabled =
```

```
true,max_running=0
set queue fast max_running +=2
create queue little
set queue little resources_max.mem=8mw,resources_max.cput=10
unset queue fast max_running
set node state = "down,offline"
active server s1,s2,s3
list queue @server1
set queue max_running = 10      - uses active queues
```

Exit Status

Upon successful processing of all the operands presented to the qmgr command, the exit status will be a value of zero.

If the qmgr command fails to process any operand, the command exits with a value greater than zero.

See Also

pbs_server (8B), pbs_queue_attributes (7B), pbs_server_attributes (7B), qstart (8B), qstop (8B), qenable (8B), qdisable (8), and the PBS External Reference Specification

qrerun

(rerun a batch job)

Synopsis

```
qrerun [{-f}] <JOBID>[ <JOBID>] ...
```

Description

The **qrerun** command directs that the specified jobs are to be rerun if possible. To rerun a job is to terminate the session leader of the job and return the job to the queued state in the execution queue in which the job currently resides.

If a job is marked as not rerunable then the rerun request will fail for that job. If the mini-server running the job is down, or it rejects the request, the Rerun Job batch request will return a failure unless **-f** is used.

Using **-f** violates IEEE Batch Processing Services Standard and should be handled with great care. It should only be used under exceptional circumstances. The best practice is to fix the problem mini-server host and let qrerun run normally. The nodes may need manual cleaning. See the **-r** option on the **qsub** and **qalter** commands.

Options

-f

Force a rerun on a job.

```
qrerun -f 15406
```

Operands

The qrerun command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Standard Error

The qrerun command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qrerun command, the exit status will be a value of zero.

If the qrerun command fails to process any operand, the command exits with a value greater than zero.

Examples

```
> qrerun 3233
```

Job 3233 will be re-run.

See Also

qsub(1B), qalter(1B), pbs_alterjob(3B), pbs_rerunjob(3B)

qrls

(release hold on pbs batch jobs)

Synopsis

```
qrls [{-h <HOLD LIST>|-t <array_range>}] <JOBID>[ <JOBID>] ...
```

Description

The **qrls** command removes or releases holds which exist on batch jobs.

A job may have one or more types of holds which make the job ineligible for execution. The types of holds are USER, OTHER, and SYSTEM. The different types of holds may require that the user issuing the qrls command have special privileges. A user may always remove a USER hold on their own jobs, but only privileged users can remove OTHER or SYSTEM holds. An attempt to release a hold for which the user does not have the correct privilege is an error and no holds will be released for that job.

If no -h option is specified, the USER hold will be released.

If the job has no execution_time pending, the job will change to the queued state. If an execution_time is still pending, the job will change to the waiting state.

Options

-h *hold_list*

Defines the types of hold to be released from the jobs. The hold_list option argument is a string consisting of one or more of the letters "u", "o", and "s" in any combination. The hold type associated with each letter is:

- u - USER
- o - OTHER
- s - SYSTEM

-t *array_range*

The *array_range* argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimted list. Examples: `-t 1-100` or `-t 1,10,50-100`

If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

Operands

The qrls command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qrls -h u 3233     release user hold on job 3233
```

Standard Error

The qrls command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qrls command, the exit status will be a value

of zero.

If the qrls command fails to process any operand, the command exits with a value greater than zero.

See Also

qsub(1B), qalter(1B), qhold(1B), pbs_alterjob(3B), pbs_holdjob(3B), and pbs_rlsjob(3B).

qrun

(run a batch job)

Synopsis

```
qrun [{-H <HOST>|-a}] <JOBID>[ <JOBID>] ...
```

Overview

The **qrun** command runs a job.

Format

-H	
Format:	<STRING> Host Identifier
Default:	---
Description:	specifies the host within the cluster on which the job(s) are to be run. The host argument is the name of a host that is a member of the cluster of hosts managed by the server. If the option is not specified, the server will select the "worst possible" host on which to execute the job.
Example:	`qrun -H hostname 15406`

-a	
Format:	---
Default:	---
Description:	run the job(s) asynchronously.
Example:	`qrun -a 15406`

Command Details

The **qrun** command is used to force a batch server to initiate the execution of a batch job. The job is run regardless of scheduling position or resource requirements.

In order to execute qrun, the user must have PBS Operation or Manager privileges.

Examples

```
> qrun 3233
```

Run job 3233.

qsig

(signal a job)

Synopsis

```
qsig [{-s <SIGNAL>}] <JOBID>[ <JOBID>] ...
     [-a]
```

Description

The qsig command requests that a signal be sent to executing batch jobs. The signal is sent to the session leader of the job. If the -s option is not specified, SIGTERM is sent. The request to signal a batch job will be rejected if:

- The user is not authorized to signal the job.
- The job is not in the running state.
- The requested signal is not supported by the system upon which the job is executing.

The qsig command sends a Signal Job batch request to the server which owns the job.

Options

-s *signal*

Declares which signal is sent to the job.

The signal argument is either a signal name, e.g. SIGKILL, the signal name without the SIG prefix, e.g. KILL, or a unsigned signal number, e.g. 9. The signal name SIGNULL is allowed; the server will send the signal 0 to the job which will have no effect on the job, but will cause an obituary to be sent if the job is no longer executing. Not all signal names will be recognized by qsig. If it doesnt recognize the signal name, try issuing the signal number instead.

Two special signal names, "suspend" and "resume", are used to suspend and resume jobs. Cray systems use the Cray-specific suspend()/resume() calls.

On non-Cray system, suspend causes a SIGTSTP to be sent to all processes in the job's top task, wait 5 seconds, and then send a SIGSTOP to all processes in all tasks on all nodes in the job. This differs from TORQUE 2.0.0 which did not have the ability to propogate signals to sister nodes. Resume sends a SIGCONT to all processes in all tasks on all nodes.

When suspended, a job continues to occupy system resources but is not executing and is not charged for walltime. The job will be listed in the "S" state. Manager or operator privilege is required to suspend or resume a job.

Note that interactive jobs may not resume properly because the top-level shell will background the suspended child process.

-a *asynchronously*

Makes the command run asynchronously.

Operands

The qsig command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qsig -s SIGKILL 3233     send a SIGKILL to job 3233
> qsig -s KILL 3233        send a SIGKILL to job 3233
> qsig -s 9 3233           send a SIGKILL to job 3233
```

Standard Error

The qsig command will write a diagnostic messages to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qsig command, the exit status will be a value of zero.

If the qsig command fails to process any operand, the command exits with a value greater than zero.

See Also

qsub(1B), pbs_sigjob(3B), pbs_resources_*(7B) where * is system type, and the PBS ERS.

qstat

show status of pbs batch jobs

Synopsis

```
qstat [-f [-1]][-W site_specific] [job_identifier... | destination...] [time]
qstat   [-a|-i|-r|-e]  [-n  [-1]]  [-s] [-G|-M]  [-R]  [-u  user_list]
       [job_identifier... |  destination...]
qstat -Q [-f [-1]][-W site_specific] [destination...]
qstat -q [-G|-M] [destination...]
qstat -B [-f [-1]][-W site_specific] [server_name...]
qstat -t
```

Description

The qstat command is used to request the status of jobs, queues, or a batch server. The requested status is written to standard out.

When requesting job status, synopsis format 1 or 2, qstat will output information about each job_identifier or all jobs at each destination. Jobs for which the user does not have status privilege are not displayed.

When requesting queue or server status, synopsis format 3 through 5, qstat will output information about each destination.

Options

-f
: Specifies that a full status display be written to standard out. The [time] value is the amount of walltime, in seconds, remaining for the job. [time] does not account for walltime multipliers.

-a
: All jobs are displayed in the alternative format, see the Standard Output section. If the operand is a destination id, all jobs at that destination are displayed. If the operand is a job id, information about that job is displayed.

-e
: If the operand is a job id or not specified, only jobs in executable queues are displayed. Setting the PBS_QSTAT_EXECONLY environment variable will also enable this option.

-i
: Job status is displayed in the alternative format. For a destination id operand, status for jobs at that destination which are not running are displayed. This includes jobs which are queued, held or waiting. If an operand is a job id, status for that job is displayed regardless of its state.

-r
: If an operand is a job id, status for that job is displayed. For a destination id operand, status for jobs at that destination which are running are displayed, this includes jobs which are suspended.

-n
: In addition to the basic information, nodes allocated to a job are listed.

-1
: In combination with -n, the -1 option puts all of the nodes on the same line as the job ID. In combination with -f, attributes are not folded to fit in a terminal window. This is intended to ease the parsing of the qstat output.

-s
: In addition to the basic information, any comment provided by the batch administrator or scheduler is shown.

-G
: Show size information in giga-bytes.

-M
: Show size information, disk or memory in mega-words. A word is considered to be 8 bytes.

-R
: In addition to other information, disk reservation information is shown. Not applicable to all systems.

-t

Normal qstat output displays a summary of the array instead of the entire array, job for job. qstat -t expands the output to display the entire array. Note that arrays are now named with brackets following the array name; for example:

dbeer@napali:~/dev/torque/array_changes$ echo sleep 20 | qsub -t 0-299 189[].napali

Individual jobs in the array are now also noted using square brackets instead of dashes; for example, here is part of the output of qstat -t for the preceding array:

189[299].napali STDIN[299] dbeer 0 Q batch

-u

Job status is displayed in the alternative format. If an operand is a job id, status for that job is displayed. For a destination id operand, status for jobs at that destination which are owned by the user(s) listed in user_list are displayed. The syntax of the user_list is:

user_name[@host][,user_name[@host],...]

Host names may be wild carded on the left end, e.g. "*.nasa.gov". User_name without a "@host" is equivalent to "user_name@*", that is at any host.

-Q

Specifies that the request is for queue status and that the operands are destination identifiers.

-q

Specifies that the request is for queue status which should be shown in the alternative format.

-B

Specifies that the request is for batch server status and that the operands are the names of servers.

Operands

If neither the -Q nor the -B option is given, the operands on the qstat command must be either job identifiers or destinations identifiers.

If the operand is a job identifier, it must be in the following form:

sequence_number[.server_name][@server]

where sequence_number.server_name is the job identifier assigned at submittal time, see qsub. If the .server_name is omitted, the name of the default server will be used. If @server is supplied, the request will be for the job identifier currently at that Server.

If the operand is a destination identifier, it is one of the following three forms:

- queue
- @server
- queue@server

If queue is specified, the request is for status of all jobs in that queue at the default server. If the @server form is given, the request is for status of all jobs at that server. If a full destination identifier, queue@server, is given, the request is for status of all jobs in the named queue at the named server.

If the -Q option is given, the operands are destination identifiers as specified above. If queue is specified, the status of that queue at the default server will be given. If queue@server is specified, the status of the named queue at the named server will be given. If @server is specified, the status of all queues at the named server will be given. If no destination is specified, the status of all queues at the default server will be given.

If the -B option is given, the operand is the name of a server.

Standard Output

Displaying Job Status

If job status is being displayed in the default format and the -f option is not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the job identifier assigned by PBS.
- the job name given by the submitter.
- the job owner.
- he CPU time used.
- the job state:
  - C
    Job is completed after having run
  - E
    Job is exiting after having run.
  - H
    Job is held.
  - Q
    job is queued, eligible to run or routed.
  - R
    job is running.
  - T
    job is being moved to new location.
  - W
    job is waiting for its execution time (-a option) to be reached.
  - S
    (Unicos only) job is suspended.
- the queue in which the job resides.

If job status is being displayed and the -f option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See the configuration section for details. If Tcl is not being used, full display for each job consists of the header line:

Job Id: job identifier

Followed by one line per job attribute of the form:

attribute_name = value

If any of the options -a, -i, -r, -u, -n, -s, -G or -M are provided, the alternative display format for jobs is used. The following items are displayed on a single line, in the specified order, separated by white space:

- the job identifier assigned by PBS.
- the job owner.
- The queue in which the job currently resides.
- The job name given by the submitter.
- The session id (if the job is running).
- The number of nodes requested by the job.
- The number of cpus or tasks requested by the job.
- The amount of memory requested by the job.
- Either the cpu time, if specified, or wall time requested by the job, (hh:mm).
- The jobs current state.
- The amount of cpu time or wall time used by the job (hh:mm).

If the -R option is provided, the line contains:

- the job identifier assigned by PBS.
- the job owner.
- The queue in which the job currently resides.
- The number of nodes requested by the job.
- The number of cpus or tasks requested by the job.
- The amount of memory requested by the job.
- Either the cpu time or wall time requested by the job.
- The jobs current state.
- The amount of cpu time or wall time used by the job.
- The amount of SRFS space requested on the big file system.
- The amount of SRFS space requested on the fast file system.
- The amount of space requested on the parallel I/O file system.

The last three fields may not contain useful information at all sites or on all systems.

Displaying Queue Status

If queue status is being displayed and the -f option was not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the queue name.
- the maximum number of jobs that may be run in the queue concurrently.
- the total number of jobs in the queue.
- the enable or disabled status of the queue.
- the started or stopped status of the queue.
- for each job state, the name of the state and the number of jobs in the queue in that state.
- the type of queue, execution or routing.

If queue status is being displayed and the -f option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See the configuration section for details. If Tcl is not being used, the full display for each queue consists of the header line:

    Queue: queue_name

Followed by one line per queue attribute of the form:

    attribute_name = value

If the -q option is specified, queue information is displayed in the alternative format: The following information is displayed on a single line:

- the queue name.
- the maximum amount of memory a job in the queue may request.
- the maximum amount of cpu time a job in the queue may request.
- the maximum amount of wall time a job in the queue may request.
- the maximum amount of nodes a job in the queue may request.
- the number of jobs in the queue in the running state.
- the number of jobs in the queue in the queued state.
- the maximum number (limit) of jobs that may be run in the queue concurrently.
- the state of the queue given by a pair of letters:
    - either the letter E if the queue is Enabled or D if Disabled
      and
    - either the letter R if the queue is Running (started) or S if Stopped.

Displaying Server Status

If batch server status is being displayed and the -f option is not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the server name.
- the maximum number of jobs that the server may run concurrently.
- the total number of jobs currently managed by the server.
- the status of the server.
- for each job state, the name of the state and the number of jobs in the server in that state.

If server status is being displayed and the -f option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See the configuration section for details. If Tcl is not being used, the full display for the server consist of the header line:

    Server: server name

Followed by one line per server attribute of the form:

    attribute_name = value

Standard Error

The qstat command will write a diagnostic message to standard error for each error occurrence.

Configuration

If qstat is compiled with an option to include a Tcl interpreter, using the -f flag to get a full display causes a check to be made for a script file to use to output the requested information. The first location checked is $HOME/.qstatrc. If this does not exist, the next location checked is administrator configured. If one of these is found, a Tcl interpreter is started and the script file is passed to it along with three global variables. The command line arguments are split into two variable named flags and operands . The status information is passed in a variable named objects . All of these variables are Tcl lists. The flags list contains the name of the command (usually "qstat") as its first element. Any other elements are command line option flags with any options they use, presented in the order given on the command line. They are broken up individually so that if two flags are given together on the command line, they are separated in the list. For example, if the user typed:

qstat -QfWbigdisplay

the flags list would contain

qstat -Q -f -W bigdisplay

The operands list contains all other command line arguments following the flags. There will always be at least one element in operands because if no operands are typed by the user, the default destination or server name is used. The objects list contains all the information retrieved from the server(s) so the Tcl interpreter can run once to format the entire output. This list has the same number of elements as the operands list. Each element is another list with two elements.

The first element is a string giving the type of objects to be found in the second. The string can take the values "server", "queue", "job" or "error".

The second element will be a list in which each element is a single batch status object of the type given by the string discussed above. In the case of "error", the list will be empty. Each object is again a list. The first element is the name of the object. The second is a list of attributes.

The third element will be the object text.

All three of these object elements correspond with fields in the structure batch_status which is described in detail for each type of object by the man pages for pbs_statjob(3), pbs_statque(3), and pbs_statserver(3). Each attribute in the second element list whose elements correspond with the attrl structure. Each will be a list with two elements. The first will be the attribute name and the second will be the attribute value.

Exit Status

Upon successful processing of all the operands presented to the qstat command, the exit status will be a value of zero.

If the qstat command fails to process any operand, the command exits with a value greater than zero.

See Also:

- qalter(1B)
- qsub(1B)
- **pbs_alterjob(3B)**
- **pbs_statjob(3B)**
- **pbs_statque(3B)**
- **pbs_statserver(3B)**
- **pbs_submit(3B)**
- **pbs_job_attributes(7B)**
- **pbs_queue_attributes(7B)**
- **pbs_server_attributes(7B)**

- **qmgr** query_other_jobs parameter (allow non-admin users to see other users' jobs
- **pbs_resources_*(7B)** where * is system type
- **PBS ERS**

qsub

submit pbs job

Synopsis

```
qsub [-a date_time] [-A account_string] [-b secs] [-c checkpoint_options]
       [-C directive_prefix] [-d path] [-D path] [-e path] [-f] [-h]
       [-I ] [-j  join ] [-k  keep ] [-l resource_list ]
       [-m  mail_options] [-M  user_list] [-N  name] [-o path]
       [-p  priority] [-P user[:group]] [-q  destination] [-r c] [-S  path_list]
       [-t  array_request] [-u  user_list]
       [-v  variable_list] [-V ] [-W additional_attributes] [-X] [-z] [script]
```

Description

To create a job is to submit an executable script to a batch server. The batch server will be the default server unless the -q option is specified. The command parses a script prior to the actual script execution; it does not execute a script itself. All script-writing rules remain in effect, including the "#!" at the head of the file. See discussion of PBS_DEFAULT under Environment Variables below. Typically, the script is a shell script which will be executed by a command shell such as sh or csh.

Options on the qsub command allow the specification of attributes which affect the behavior of the job.

The qsub command will pass certain environment variables in the Variable_List attribute of the job. These variables will be available to the job. The value for the following variables will be taken from the environment of the qsub command: HOME, LANG, LOGNAME, PATH, MAIL, SHELL, and TZ. These values will be assigned to a new name which is the current name prefixed with the string "PBS_O_". For example, the job will have access to an environment variable named PBS_O_HOME which have the value of the variable HOME in the qsub command environment.

In addition to the above, the following environment variables will be available to the batch job.

PBS_O_HOST
: the name of the host upon which the qsub command is running.

PBS_SERVER
: the hostname of the pbs_server which qsub submits the job to.

PBS_O_QUEUE
: the name of the original queue to which the job was submitted.

PBS_O_WORKDIR
: the absolute path of the current working directory of the qsub command.

PBS_ARRAYID
: each member of a job array is assigned a unique identifier (see -t).

PBS_ENVIRONMENT
: set to PBS_BATCH to indicate the job is a batch job, or to PBS_INTERACTIVE to indicate the job is a PBS interactive job, see -I option.

PBS_JOBID
: the job identifier assigned to the job by the batch system. It can be used in the stdout and stderr paths. TORQUE replaces $PBS_JOBID with the job's jobid (for example, `#PBS -o /tmp/$PBS_JOBID.output`).

PBS_JOBNAME
: the job name supplied by the user.

PBS_NODEFILE
: the name of the file contain the list of nodes assigned to the job (for parallel and cluster systems).

PBS_QUEUE
: the name of the queue from which the job is executed.

Options

-a *date_time*
: Declares the time after which the job is eligible for execution.

  The date_time argument is in the form:

[[[[CC]YY]MM]DD]hhmm[.SS]

Where CC is the first two digits of the year (the century), YY is the second two digits of the year, MM is the two digits for the month, DD is the day of the month, hh is the hour, mm is the minute, and the optional SS is the seconds.

If the month, MM, is not specified, it will default to the current month if the specified day DD, is in the future. Otherwise, the month will be set to next month. Likewise, if the day, DD, is not specified, it will default to today if the time hhmm is in the future. Otherwise, the day will be set to tomorrow. For example, if you submit a job at 11:15am with a time of -a 1110, the job will be eligible to run at 11:10am tomorrow.

-A *account_string*
Defines the account string associated with the job. The account_string is an undefined string of characters and is interpreted by the server which executes the job. See section 2.7.1 of the PBS ERS.

-b *seconds*
Defines the maximum number of seconds qsub will block attempting to contact pbs_server. If pbs_server is down, or for a variety of communication failures, qsub will continually retry connecting to pbs_server for job submission. This value overrides the CLIENTRETRY parameter in torque.cfg. This is a non-portable TORQUE extension. Portability-minded users can use the PBS_CLIENTRETRY environmental variable. A negative value is interpreted as infinity. The default is 0.

-c *checkpoint_options*
Defines the options that will apply to the job. If the job executes upon a host which does not support checkpoint, these options will be ignored.

Valid checkpoint options are:

none
No checkpointing is to be performed.
enabled
Specify that checkpointing is allowed but must be explicitly invoked by either the qhold or qchkpt commands.
shutdown
Specify that checkpointing is to be done on a job at pbs_mom shutdown.
periodic
Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the $checkpoint_interval option in the MOM config file or by specifying an interval when the job is submitted
interval=minutes
Checkpointing is to be performed at an interval of minutes, which is the integer number of minutes of wall time used by the job. This value must be greater than zero.
depth=number
Specify a number (depth) of checkpoint images to be kept in the checkpoint directory.
dir=path
Specify a checkpoint directory (default is /var/spool/torque/checkpoint).

-C *directive_prefix*
Defines the prefix that declares a directive to the qsub command within the script file. See the paragraph on script directives in the Extended Description section.

If the -C option is presented with a directive_prefix argument that is the null string, qsub will not scan the script file for directives.

-d *path*
Defines the working directory path to be used for the job. If the -d option is not specified, the default working directory is the home directory. This option sets the environment variable PBS_O_INITDIR.

-D *path*
Defines the root directory to be used for the job. This option sets the environment variable PBS_O_ROOTDIR.

-e *path*

Defines the path to be used for the standard error stream of the batch job. The path argument is of the form:

[hostname:]path_name

Where hostname is the name of a host to which the file will be returned and path_name is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:

path_name
: Where path_name is not an absolute path name, then the qsub command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component.

hostname:path_name
: Where path_name is not an absolute path name, then the qsub command will not expand the path name relative to the current working directory of the command. On delivery of the standard error, the path name will be expanded relative to the users home directory on the hostname system.

path_name
: Where path_name specifies an absolute path name, then the qsub will supply the name of the host on which it is executing for the hostname.

hostname:path_name
: Where path_name specifies an absolute path name, the path will be used as specified.

If the -e option is not specified, the default file name for the standard error stream will be used. The default name has the following form:

job_name.esequence_number

where job_name is the name of the job, see -N option, and sequence_number is the job number assigned when the job is submitted.

-f

Job is made fault tolerant. Jobs running on multiple nodes are periodically polled by mother superior. If one of the nodes fails to report, the job is canceled by mother superior and a failure is reported. If a job is fault tolerant, it will not be canceled based on failed polling (no matter how many nodes fail to report). This may be desirable if transient network failures are causing large jobs not to complete, where ignoring one failed polling attempt can be corrected at the next polling attempt.

If TORQUE is compiled with PBS_NO_POSIX_VIOLATION (there is no config option for this), you have to use `-W fault_tolerant=true` to mark the job as fault tolerant.

-h

Specifies that a user hold be applied to the job at submission time.

-I

Declares that the job is to be run "interactively". The job will be queued and scheduled as any PBS batch job, but when executed, the standard input, output, and error streams of the job are connected through qsub to the terminal session in which qsub is running. Interactive jobs are forced to not rerunable. See the "Extended Description" paragraph for addition information of interactive jobs.

-j *join*

Declares if the standard error stream of the job will be merged with the standard output stream of the job.

An option argument value of oe directs that the two streams will be merged, intermixed, as standard output. An option argument value of eo directs that the two streams will be merged, intermixed, as standard error.

If the join argument is n or the option is not specified, the two streams will be two separate files.

-k *keep*

Defines which (if either) of standard output or standard error will be retained on the execution host. If set for a stream, this option overrides the path name for that stream. If not set, neither stream is retained on the execution host.

The argument is either the single letter "e" or "o", or the letters "e" and "o" combined in either order. Or the argument is the letter "n".

e
: The standard error stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:
  job_name.esequence
  where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier.

o
: The standard output stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:
  job_name.osequence
  where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier.

eo
: Both the standard output and standard error streams will be retained.

oe
: Both the standard output and standard error streams will be retained.

n
: Neither stream is retained.

-l *resource_list*

Defines the resources that are required by the job and establishes a limit to the amount of resource that can be consumed. If not set for a generally available resource, such as CPU time, the limit is infinite. The resource_list argument is of the form:

```
resource_name[=[value]][,resource_name[=[value]],...]
```

-m *mail_options*

Defines the set of conditions under which the execution server will send a mail message about the job. The mail_options argument is a string which consists of either the single character "n", or one or more of the characters "a", "b", and "e".

If the character "n" is specified, no normal mail is sent. Mail for job cancels and other events outside of normal job processing are still sent.

For the letters "a", "b", and "e":

a
: mail is sent when the job is aborted by the batch system.

b
: mail is sent when the job begins execution.

e
: mail is sent when the job terminates.

If the -m option is not specified, mail will be sent if the job is aborted.

-M *user_list*

Declares the list of users to whom mail is sent by the execution server when it sends mail about the job.

The user_list argument is of the form:

```
user[@host][,user[@host],...]
```

If unset, the list defaults to the submitting user at the qsub host, i.e. the job owner.

-N *name*

Declares a name for the job. The name specified may be up to and including 15 characters in length. It must consist of printable, non white space characters with the first character alphabetic.

If the -N option is not specified, the job name will be the base name of the job script file specified on the command line. If no script file name was specified and the script was read from the standard input, then the job name will be set to STDIN.

-o *path*

Defines the path to be used for the standard output stream of the batch job. The path argument is of the form:

[hostname:]path_name

where hostname is the name of a host to which the file will be returned and path_name is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:

path_name
: Where path_name is not an absolute path name, then the qsub command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component.

hostname:path_name
: Where path_name is not an absolute path name, then the qsub command will not expand the path name relative to the current working directory of the command. On delivery of the standard output, the path name will be expanded relative to the users home directory on the hostname system.

path_name
: Where path_name specifies an absolute path name, then the qsub will supply the name of the host on which it is executing for the hostname.

hostname:path_name
: Where path_name specifies an absolute path name, the path will be used as specified.

If the -o option is not specified, the default file name for the standard output stream will be used. The default name has the following form:

job_name.osequence_number

where job_name is the name of the job, see -N option, and sequence_number is the job number assigned when the job is submitted.

-p *priority*

Defines the priority of the job. The priority argument must be a integer between -1024 and +1023 inclusive. The default is no priority which is equivalent to a priority of zero.

-P *user[:group]*

Allows a root user to submit a job as another user. TORQUE treats proxy jobs as though the jobs were submitted by the supplied username. This feature is available in TORQUE 2.4.7 and later, however, TORQUE 2.4.7 does not have the ability to supply the *[:group]* option. The *[:group]* option is available in TORQUE 2.4.8 and later.

-q *destination*

Defines the destination of the job. The destination names a queue, a server, or a queue at a server.

The qsub command will submit the script to the server defined by the destination argument. If the destination is a routing queue, the job may be routed by the server to a new destination.

If the -q option is not specified, the qsub command will submit the script to the default server. See PBS_DEFAULT under the Environment Variables section on this man page and the PBS ERS section 2.7.4, "Default Server".

If the -q option is specified, it is in one of the following three forms:

- queue
- @server
- queue@server

If the destination argument names a queue and does not name a server, the job will be submitted to the named queue at the default server.

If the destination argument names a server and does not name a queue, the job will be submitted to the default queue at the named server.

If the destination argument names both a queue and a server, the job will be submitted to the named queue at the named server.

-r *y|n*

Declares whether the job is rerunable. See the qrerun command. The option argument is a single character, either y or n.dd

If the argument is "y", the job is rerunable. If the argument is "n", the job is not rerunable. The default value is y, rerunable.

-S *path_list*

Declares the shell that interprets the job script.

The option argument path_list is in the form:

path[@host][,path[@host],...]

Only one path may be specified for any host named. Only one path may be specified without the corresponding host name. The path selected will be the one with the host name that matched the name of the execution host. If no matching host is found, then the path specified without a host will be selected, if present.

If the -S option is not specified, the option argument is the null string, or no entry from the path_list is selected, the execution will use the users login shell on the execution host.

-t *array_request*

Specifies the task ids of a job array. Single task arrays are allowed.

The array_request argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimted list. Examples: `-t 1-100` or `-t 1,10,50-100`.

An optional **slot limit** can be specified to limit the amount of jobs that can run concurrently in the job array. The default value is unlimited. The slot limit must be the last thing specified in the *array_request* and is delimited from the array by a percent sign (%).

```
qsub script.sh -t 0-299%5
```

This sets the slot limit to 5. Only 5 jobs from this array can run at the same time.

You can use qalter to modify slot limits on an array. The server parameter max_slot_limit can be used to set a global slot limit policy.

-u *user_list*

Defines the user name under which the job is to run on the execution system.

The user_list argument is of the form:

user[@host][,user[@host],...]

Only one user name may be given per specified host. Only one of the user specifications may be supplied without the corresponding host specification. That user name will used for execution on any host not named in the argument list. If unset, the user list defaults to the user who is running qsub.

-v *variable_list*

Expands the list of environment variables that are exported to the job.

In addition to the variables described in the "Description" section above, variable_list names environment variables from the qsub command environment which are made available to the job when it executes. The variable_list is a comma separated list of strings of the form variable or variable=value. These variables and their values are passed to the job.

-V

Declares that all environment variables in the qsub commands environment are to be exported to the batch job.

-W *additional_attributes*

The -W option allows for the specification of additional job attributes. The general syntax of the -W is in the form:

-W attr_name=attr_value[,attr_name=attr_value...]

Note if white space occurs anywhere within the option argument string or the equal sign, "=", occurs within an attribute_value string, then the string must be enclosed with either single or double quote marks.

PBS currently supports the following attributes within the -W option.

depend=dependency_list

Defines the dependency between this and other jobs. The dependency_list is in the form:

type[:argument[:argument...][,type:argument...]

The argument is either a numeric count or a PBS job id according to type. If argument is a count, it must be greater than 0. If it is a job id and not fully specified in the form seq_number.server.name, it will be expanded according to the default server rules which apply to job IDs on most commands. If argument is null (the preceding colon need not be specified), the dependency of the corresponding type is cleared (unset). These are the valid dependencies:

synccount:count
: This job is the first in a set of jobs to be executed at the same time. Count is the number of additional jobs in the set.

syncwith:jobid
: This job is an additional member of a set of jobs to be executed at the same time. In the above and following dependency types, jobid is the job identifier of the first job in the set.

after:jobid[:jobid...]
: This job may be scheduled for execution at any point after jobs jobid have started execution.

afterok:jobid[:jobid...]
: This job may be scheduled for execution only after jobs jobid have terminated with no errors. See the csh warning under "Extended Description".

afternotok:jobid[:jobid...]
: This job may be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning under "Extended Description".

afterany:jobid[:jobid...]
: This job may be scheduled for execution after jobs jobid have terminated, with or without errors.

on:count
: This job may be scheduled for execution after count dependencies on other jobs have been satisfied. This form is used in conjunction with one of the before forms, see below.

before:jobid[:jobid...]
: When this job has begun execution, then jobs jobid... may begin.

beforeok:jobid[:jobid...]
: If this job terminates execution without errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforenotok:jobid[:jobid...]
: If this job terminates execution with errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforeany:jobid[:jobid...]
: When this job terminates execution, jobs jobid... may begin.

If any of the before forms are used, the jobs referenced by jobid must have been submitted with a dependency type of on.

If any of the before forms are used, the jobs referenced by jobid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.

Array dependencies make a job depend on an array or part of an array. If no count is given, then the entire array is assumed. Array dependency examples are here.

afterstartarray:arrayid[count]
: After this many jobs have started from arrayid, this job may start.

afterokarray:arrayid[count]
: This job may be scheduled for execution only after jobs in arrayid have terminated with no errors.

afternotokarray:arrayid[count]
: This job may be scheduled for execution only after jobs in arrayid have terminated with errors.

afteranyarray:arrayid[count]
: This job may be scheduled for execution after jobs in arrayid have terminated, with or without errors.

beforestartarray:arrayid[count]
: Before this many jobs have started from arrayid, this job may start.

beforeokarray:arrayid[count]
: If this job terminates execution without errors, then jobs in arrayid may begin.

beforenotokarray:arrayid[count]
: If this job terminates execution with errors, then jobs in arrayid may begin.

beforeanyarray:arrayid[count]
: When this job terminates execution, jobs in arrayid may begin.

If any of the before forms are used, the jobs referenced by arrayid must have been submitted with a dependency type of on. If any of the before forms are used, the jobs referenced by arrayid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.

Error processing of the existence, state, or condition of he job on which the newly submitted job is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the new job will be deleted by the server. Mail will be sent to the job submitter stating the error.

Dependency examples:

- qsub -W depend=afterok:123.big.iron.com /tmp/script
- qsub -W depend=before:234.hunk1.com:235.hunk1.com
- /tmp/script
- qsub script.sh -W depend=afterokarray:427[]
  This assumes every job in array 427 has to finish successfully for the dependency to be satisfied.
- qsub script.sh -W depend=afterokarray:427[][5]
  This means that 5 of the jobs in array 427 have to successfully finish in order for the dependency to be satisfied.

group_list=g_list
: Defines the group name under which the job is to run on the execution system. The g_list argument is of the form:
  group[@host][,group[@host],...]

Only one group name may be given per specified host. Only one of the group specifications may be supplied without the corresponding host specification. That group name will used for execution on any host not named in the argument list. If not set, the group_list defaults to the primary group of the user under which the job will be run.

interactive=true
: If the interactive attribute is specified, the job is an interactive job. The -I option is a alternative method of specifying this attribute.

stagein=file_list
stageout=file_list
: Specifies which files are staged (copied) in before job start or staged out after the job completes execution. On completion of the job, all staged-in and staged-out files are

removed from the execution system. The file_list is in the form:

local_file@hostname:remote_file[,...]

regardless of the direction of the copy. The name local_file is the name of the file on the system where the job executed. It may be an absolute path or relative to the home directory of the user. The name remote_file is the destination name on the host specified by hostname. The name may be absolute or relative to the users home directory on the destination host. The use of wildcards in the file name is not recommended. The file names map to a remote copy program (rcp) call on the execution system in the follow manner:

- For stagein: rcp hostname:remote_file local_file
- For stageout: rcp local_file hostname:remote_file

Data staging examples:

```
-W stagein=/tmp/input.txt@headnode:/home/user/input.txt
-W stageout=/tmp/output.txt@headnode:/home/user/output.txt
```

If TORQUE has been compiled with wordexp support, then variables can be used in the specified paths. Currently only $PBS_JOBID, $HOME, and $TMPDIR are supported for stagein.

umask=XXX
: Sets umask used to create stdout and stderr spool files in pbs_mom spool directory. Values starting with 0 are treated as octal values, otherwise the value is treated as a decimal umask value.

-X
: Enables X11 forwarding. The DISPLAY environment variable must be set.

-z
: Directs that the qsub command is not to write the job identifier assigned to the job to the commands standard output.

Operands

The qsub command accepts a script operand that is the path to the script of the job. If the path is relative, it will be expanded relative to the working directory of the qsub command.

If the script operand is not provided or the operand is the single character "-", the qsub command reads the script from standard input. When the script is being read from Standard Input, qsub will copy the file to a temporary file. This temporary file is passed to the library interface routine pbs_submit. The temporary file is removed by qsub after pbs_submit returns or upon the receipt of a signal which would cause qsub to terminate.

Standard Input

The qsub command reads the script for the job from standard input if the script operand is missing or is the single character "-".

Input Files

The script file is read by the qsub command. Qsub acts upon any directives found in the script.

When the job is created, a copy of the script file is made and that copy cannot be modified.

Standard Output

Unless the -z option is set, the job identifier assigned to the job will be written to standard output if the job is successfully created.

Standard Error

The qsub command will write a diagnostic message to standard error for each error occurrence.

Environment Variables

The values of some or all of the variables in the qsub commands environment are exported with the job, see the -v and -V options.

The environment variable PBS_DEFAULT defines the name of the default server. Typically, it corresponds to the system name of the host on which the server is running. If PBS_DEFAULT is not set, the default is defined by an administrator established file.

The environment variable PBS_DPREFIX determines the prefix string which identifies directives in the script.

The environment variable PBS_CLIENTRETRY defines the maximum number of seconds qsub will block. See the -b option above. Despite the name, currently qsub is the only client that supports this option.

TORQUE.cfg

The torque.cfg file, located in PBS_SERVER_HOME (/var/spool/torque by default) controls the behavior of the qsub command. This file contains a list of parameters and values separated by whitespace

QSUBSLEEP takes an integer operand which specifies time to sleep when running qsub command. Used to prevent users from overwhelming the scheduler.

SUBMITFILTER specifies the path to the submit filter used to pre-process job submission. The default path is libexecdir/qsub_filter, which falls back to /usr/local/sbin/torque_submitfilter for backwards compatibility. This torque.cfg parameter overrides this default.

SERVERHOST

QSUBHOST

QSUBSENDUID

XAUTHPATH

CLIENTRETRY

VALIDATEGROUP

DEFAULTCKPT

VALIDATEPATH

RERUNNABLEBYDEFAULT

For example:

- QSUBSLEEP 2
- RERUNNABLEBYDEFAULT false

Extended Description

Script Processing:

A job script may consist of PBS directives, comments and executable statements. A PBS directive provides a way of specifying job attributes in addition to the command line options. For example:

```
:
#PBS -N Job_name
#PBS -l walltime=10:30,mem=320kb
#PBS -m be
#
step1 arg1 arg2
```

step2 arg3 arg4

The qsub command scans the lines of the script file for directives. An initial line in the script that begins with the characters "#!" or the character ":" will be ignored and scanning will start with the next line. Scanning will continue until the first executable line, that is a line that is not blank, not a directive line, nor a line whose first non white space character is "#". If directives occur on subsequent lines, they will be ignored.

A line in the script file will be processed as a directive to qsub if and only if the string of characters starting with the first non white space character on the line and of the same length as the directive prefix matches the directive prefix.

The remainder of the directive line consists of the options to qsub in the same syntax as they appear on the command line. The option character is to be preceded with the "-" character.

If an option is present in both a directive and on the command line, that option and its argument, if any, will be ignored in the directive. The command line takes precedence.

If an option is present in a directive and not on the command line, that option and its argument, if any, will be processed as if it had occurred on the command line.

The directive prefix string will be determined in order of preference from:

The value of the -C option argument if the option is specified on the command line.

The value of the environment variable PBS_DPREFIX if it is defined.

The four character string #PBS.

If the -C option is found in a directive in the script file, it will be ignored.

User Authorization:

When the user submits a job from a system other than the one on which the PBS Server is running, the name under which the job is to be executed is selected according to the rules listed under the -u option. The user submitting the job must be authorized to run the job under the execution user name. This authorization is provided if:

- The host on which qsub is run is trusted by the execution host (see /etc/hosts.equiv)
- The execution user has an .rhosts file naming the submitting user on the submitting host.

C-Shell .logout File:

The following warning applies for users of the c-shell, csh. If the job is executed under the csh and a .logout file exists in the home directory in which the job executes, the exit status of the job is that of the .logout script, not the job script. This may impact any inter-job dependencies. To preserve the job exit status, either remove the .logout file or place the following line as the first line in the .logout file

set EXITVAL = $status

and the following line as the last executable line in .logout

exit $EXITVAL

Interactive Jobs:

If the -I option is specified on the command line or in a script directive, or if the "interactive" job attribute declared true via the -W option, -W interactive=true, either on the command line or in a script directive, the job is an interactive job. The script will be processed for directives, but will not be included with the job. When the job begins execution, all input to the job is from the terminal session in which qsub is running.

When an interactive job is submitted, the qsub command will not terminate when the job is submitted. Qsub will remain running until the job terminates, is aborted, or the user interrupts qsub with an SIGINT (the control-C key). If qsub is interrupted prior to job start, it will query if the user wishes to exit. If the user response "yes", qsub exits and the job is aborted.

One the interactive job has started execution, input to and output from the job pass through qsub. Keyboard generated interrupts are passed to the job. Lines entered that begin with the tilde (~) character and contain special sequences are escaped by qsub. The recognized escape sequences are:

~.
: Qsub terminates execution. The batch job is also terminated.

~susp
: Suspend the qsub program if running under the C shell. "susp" is the suspend character, usually CNTL-Z.

~asusp
: Suspend the input half of qsub (terminal to job), but allow output to continue to be displayed. Only works under the C shell. "asusp" is the auxiliary suspend character, usually CNTL-Y.

Exit Status

Upon successful processing, the qsub exit status will be a value of zero.

If the qsub command fails, the command exits with a value greater than zero.

See Also

qalter(1B), qdel(1B), qhold(1B), qmove(1B), qmsg(1B), qrerun(1B), qrls(1B), qselect(1B), qsig(1B), qstat(1B), pbs_connect(3B), pbs_job_attributes(7B), pbs_queue_attributes(7B), pbs_resources_irix5(7B), pbs_resources_sp2(7B), pbs_resources_sunos4(7B), pbs_resources_unicos8(7B), pbs_server_attributes(7B), and pbs_server(8B)

qterm

terminate processing by a pbs batch server

Synopsis

qterm [-t type] [server...]

Description

The qterm command terminates a batch server. When a server receives a terminate command, the server will go into the *Terminating* state. No new jobs will be allowed to be started into execution nor enqueued into the server. The impact on jobs currently being run by the server depends

In order to execute qterm, the user must have PBS Operation or Manager privileges.

Options

-t *type*
Specifies the type of shut down. The types are:

- immediate - All running jobs are to immediately stop execution. If checkpointing is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If checkpoint is not supported or the job cannot be checkpointed, running jobs are requeued if the rerunable attribute is true. Otherwise, jobs are killed.
- delay - If checkpointing is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If a job cannot be checkpointed, but can be rerun, the job is terminated and requeued. Otherwise, running jobs are allowed to continue to run. Note, the operator or administrator may use the qrerun and qdel commands to remove running jobs.
- quick - This is the default action if the -t option is not specified. This option is used when you wish that running jobs be left running when the server shuts down. The server will cleanly shutdown and can be restarted when desired. Upon restart of the server, jobs that continue to run are shown as running; jobs that terminated during the server's absence will be placed into the exiting state.

Operands

The server operand specifies which servers are to shutdown. If no servers are given, then the default server will be terminated.

Standard Error

The qterm command will write a diagnostic message to standard error for each error occurrence.

Exit Status

Upon successful processing of all the operands presented to the qterm command, the exit status will be a value of zero.

If the qterm command fails to process any operand, the command exits with a value greater than zero.

See Also

pbs_server(8B), qmgr(8B), pbs_resources_aix4(7B), pbs_resources_irix5(7B), pbs_resources_sp2(7B), pbs_resources_sunos4(7B), and pbs_resources_unicos8(7B)

pbs_mom

start a pbs batch execution mini-server

Synopsis

```
pbs_mom [-C chkdirectory] [-c config] [-d directory] [-h hostname]
        [-L logfile] [-M MOMport] [-R RPPport] [-p|-r] [-x]
```

Description

The pbs_mom command is located within the TORQUE_HOME directory and starts the operation of a batch Machine Oriented Mini-server (MOM) on the execution host. To insure that the pbs_mom command is not runnable by the general user community, the server will only execute if its real and effective uid is zero.

The first function of pbs_mom is to place jobs into execution as directed by the server, establish resource usage limits, monitor the job's usage, and notify the server when the job completes. If they exist, pbs_mom will execute a prologue script before executing a job and an epilogue script after executing the job.

The second function of pbs_mom is to respond to resource monitor requests. This was done by a separate process in previous versions of PBS but has now been combined into one process. It provides information about the status of running jobs, memory available etc.

The last function of pbs_mom is to respond to task manager requests. This involves communicating with running tasks over a tcp socket as well as communicating with other MOMs within a job (a.k.a. a "sisterhood").

pbs_mom will record a diagnostic message in a log file for any error occurrence. The log files are maintained in the mom_logs directory below the home directory of the server. If the log file cannot be opened, the diagnostic message is written to the system console.

Options

Flag	Name	Description
-a	alarm	Used to specify the alarm timeout in seconds for computing a resource. Every time a resource request is processed, an alarm is set for the given amount of time. If the request has not completed before the given time, an alarm signal is generated. The default is 5 seconds.
-C	chkdirectory	Specifies The path of the directory used to hold checkpoint files. [Currently this is only valid on Cray systems.] The default directory is TORQUE_HOME/spool/checkpoint, see the -d option. The directory specified with the -C option must be owned by root and accessible (rwx) only by root to protect the security of the checkpoint files.
-c	config	Specify a alternative configuration file, see description below. If this is a relative file name it will be relative to TORQUE_HOME/mom_priv, see the -d option. If the specified file cannot be opened, pbs_mom will abort. If the -c option is not supplied, pbs_mom will attempt to open the default configuration file "config" in TORQUE_HOME/mom_priv. If this file is not present, pbs_mom will log the fact and continue.
-d	directory	Specifies the path of the directory which is the home of the server's working files, TORQUE_HOME. This option is typically used along with -M when debugging MOM. The default directory is given by $PBS_SERVER_HOME which is typically /usr/spool/PBS.
-h	hostname	Set MOM's hostname. This can be useful on multi-homed networks.
-L	logfile	Specify an absolute path name for use as the log file. If not specified, MOM will open a file named for the current date in the TORQUE_HOME/mom_logs directory, see the -d option.

-M	port	Specifies the port number on which the mini-server (MOM) will listen for batch requests.
-p	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. On any restart of MOM, the new mini-server will not be the parent of any running jobs, MOM has lost control of her offspring (not a new situation for a mother). With the -p option, MOM will allow the jobs to continue to run and monitor them indirectly via polling. This flag is redundant in that this is the default behavior when starting the server. The -p option is mutually exclusive with the -r and -q options.
-q	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. With the -q option, MOM will allow the processes belonging to jobs to continue to run, but will not attempt to monitor them. The -q option is mutually exclusive with the -p and -r options.
-R	port	Specifies the port number on which the mini-server (MOM) will listen for resource monitor requests, task manager requests and inter-MOM messages. Both a UDP and a TCP port of this number will be used.
-r	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. With the -r option, MOM will kill any processes belonging to jobs, mark the jobs as terminated, and notify the batch server which owns the job. The -r option is mutually exclusive with the -p and -q options. Normally the mini-server is started from the system boot file without the -p or the -r option. The mini-server will make no attempt to signal the former session of any job which may have been running when the mini-server terminated. It is assumed that on reboot, all processes have been killed. If the -r option is used following a reboot, process IDs (pids) may be reused and MOM may kill a process that is not a batch session.
-x	n/a	Disables the check for privileged port resource monitor connections. This is used mainly for testing since the privileged port is the only mechanism used to prevent any ordinary user from connecting.

Configuration File

The configuration file may be specified on the command line at program start with the -c flag. The use of this file is to provide several types of run time information to pbs_mom: static resource names and values, external resources provided by a program to be run on request via a shell escape, and values to pass to internal set up functions at initialization (and re-initialization).

Each item type is on a single line with the component parts separated by white space. If the line starts with a hash mark (pound sign, #), the line is considered to be a comment and is skipped.

Static Resources

For static resource names and values, the configuration file contains a list of resource names/values pairs, one pair per line and separated by white space. An example of static resource names and values could be the number of tape drives of different types and could be specified by:

- tape3480 4
- tape3420 2
- tapedat 1
- tape8mm 1

Shell Commands

If the first character of the value is an exclamation mark (!), the entire rest of the line is saved to be

executed through the services of the system(3) standard library routine.

The shell escape provides a means for the resource monitor to yield arbitrary information to the scheduler. Parameter substitution is done such that the value of any qualifier sent with the query, as explained below, replaces a token with a percent sign (%) followed by the name of the qualifier. For example, here is a configuration file line which gives a resource name of "escape":

```
escape !echo %xxx %yyy
```

If a query for "escape" is sent with no qualifiers, the command executed would be `echo %xxx %yyy`. If one qualifier is sent, `escape[xxx=hi there]`, the command executed would be `echo hi there %yyy`. If two qualifiers are sent, `escape[xxx=hi][yyy=there]`, the command executed would be `echo hi there`. If a qualifier is sent with no matching token in the command line, `escape[zzz=snafu]`, an error is reported.

size[fs=<FS>]

Specifies that the available and configured disk space in the <FS> filesystem is to be reported to the pbs_server and scheduler. To request disk space on a per job basis, specify the file resource as in, `qsub -l nodes=1,file=1000kb`. For example, the available and configured disk space in the /localscratch filesystem will be reported:

```
size[fs=/localscratch]
```

Initialization Value

An initialization value directive has a name which starts with a dollar sign ($) and must be known to the MOM via an internal table. The entries in this table now are:

- pbsclient

  Causes a host name to be added to the list of hosts which will be allowed to connect to theMOM as long as they are using a privilaged port for the purposes of resource monitor requests. For example, here are two configuration file lines which will allow the hosts "fred" and "wilma" to connect:

  ```
  $pbsclient fred
  $pbsclient wilma
  ```

  Two host names are always allowed to connect to pbs_mom "localhost" and the name returned to pbs_mom by the system call gethostname(). These names need not be specified in the configuration file. The hosts listed as "clients" can issue Resource Manager (RM) requests. Other MOM nodes and servers do not need to be listed as clients.

- restricted

  Causes a host name to be added to the list of hosts which will be allowed to connect to the MOM without needing to use a privilaged port. These names allow for wildcard matching. For example, here is a configuration file line which will allow queries from any host from the domain "ibm.com".

  ```
  $restricted *.ibm.com
  ```

  The restriction which applies to these connections is that only internal queries may be made. No resources from a config file will be found. This is to prevent any shell commands from being run by a non-root process. This parameter is generally not required except for some versions of OSX.

- logevent

  Sets the mask that determines which event types are logged by pbs_mom. For example:

  ```
  $logevent 0x1fff $logevent 255
  ```

  The first example would set the log event mask to 0x1ff (511) which enables logging of all events including debug events. The second example would set the mask to 0x0ff (255) which enables all events except debug events.

- cputmult

  Sets a factor used to adjust cpu time used by a job. This is provided to allow adjustment of time charged and limits enforced where the job might run on systems with different cpu performance.

If the MOM's system is faster than the reference system, set cputmult to a decimal value greater than 1.0. If the MOM's system is slower, set cputmult to a value between 1.0 and 0.0. For example:

```
$cputmult 1.5 $cputmult 0.75
```

- usecp
  Specifies which directories should be staged with cp instead of rcp/scp. If a shared filesystem is available on all hosts in a cluster, this directive is used to make these filesystems known to the MOM. For example, if /home is NFS mounted on all nodes in a cluster:

  ```
  $usecp *:/home /home
  ```

- wallmult
  Sets a factor to adjust wall time usage by to job to a common reference system. The factor is used for walltime calculations and limits in the same way that cputmult is used for cpu time.

- configversion
  Specifies the version of the config file data, a string.

- check_poll_time
  Specifies the MOM interval in seconds. The MOM checks each job for updated resource usages, exited processes, over-limit conditions, etc., once per interval. This value should be equal or lower to pbs_server's job_stat_rate. High values result in stale information reported to pbs_server. Low values result in increased system usage by the MOM. Default is 45 seconds.

- down_on_error
  Causes the MOM to report itself as state "down" to pbs_server in the event of a failed health check. This feature is experimental. See health check below.

- ideal_load
  Ideal processor load. Represents a low water mark for the load average. A node that is currently busy will consider itself free after falling below ideal_load.

- loglevel
  Specifies the verbosity of logging with higher numbers specifying more verbose logging. Values may range between 0 and 7.

- log_file_max_size
  If this is set to a value > 0, then pbs_mom will roll the current log file to log-file-name.1 when its size is greater than or equal to the value of log_file_max_size. This value is interpreted as kilobytes.

- log_file_roll_depth
  If this is set to a value >=1 and log_file_max_size is set, then pbs_mom will allow logs to be rolled up to the specified number of logs. At every roll, the oldest log will be the one to be deleted to make room for rolling. Pbs_mom will continue rolling the log files to log-file-name.log_file_roll_depth.

- max_load
  Maximum processor load. Nodes over this load average are considered busy (see ideal_load above).

- enablemomrestart
  Enables automatic restarts of the MOM. If enabled, the MOM will check if its binary has been updated and restart itself at a safe point when no jobs are running; thus making upgrades easier. The check is made by comparing the mtime of the pbs_mom executable. Command-line args, the process name, and the PATH env variable are preserved across restarts. It is recommended that this not be enabled in the config file, but enabled when desired with momctl (see RESOURCES for more information.)

- node_check_script
  Specifies the fully qualified pathname of the health check script to run (see health check for more information).

node_check_interval

Specifies when to run the MOM health check. The check can be either periodic, event-driven, or both. The value starts with an integer specifying the number of MOM intervals between subsequent executions of the specified health check. After the integer is an optional comma-separated list of event names. Currently supported are "jobstart" and "jobend". This value defaults to 1 with no events indicating the check is run every MOM interval. (see health check for more information)

```
$node_check_interval 0,Disabled
$node_check_interval 0,jobstartOnly
$node_check_interval 10,jobstart,jobend
```

- prologalarm

  Specifies maximum duration (in seconds) which the MOM will wait for the job prolog or job job epilog to complete. This parameter defaults to 300 seconds (5 minutes).

- rcpcmd

  Specify the full path and argument to be used for remote file copies. This overrides the compile-time default found in configure. This must contain 2 words: the full path to the command and the options. The copy command must be able to recursively copy files to the remote host and accept arguments of the form "user@host:files." For example:

  ```
  $rcpcmd /usr/bin/rcp -rp
  $rcpcmd /usr/bin/scp -rpB
  ```

- remote_reconfig

  Enables the ability to remotely reconfigure pbs_mom with a new config file. Default is disabled. This parameter accepts various forms of true, yes, and 1.

- timeout

  Specifies the number of seconds before TCP messages will time out. TCP messages include job obituaries, and TM requests if RPP is disabled. Default is 60 seconds.

- tmpdir

  Sets the directory basename for a per-job temporary directory. Before job launch, the MOM will append the jobid to the tmpdir basename and create the directory. After the job exit, the MOM will recursively delete it. The env variable TMPDIR will be set for all prolog/epilog scripts, the job script, and TM tasks.

  Directory creation and removal is done as the job owner and group, so the owner must have write permission to create the directory. If the directory already exists and is owned by the job owner, it will not be deleted after the job. If the directory already exists and is NOT owned by the job owner, the job start will be rejected.

- status_update_time

  Specifies (in seconds) how often theMOM updates its status information to pbs_server. This value should correlate with the server's scheduling interval and its "node_check_rate" attribute. High values for "status_update_time" cause pbs_server to report stale information, while low values increase the load of pbs_server and the network. Default is 45 seconds.

- varattr

  This is similar to a shell escape above, but includes a TTL. The command will only be run every TTL seconds. A TTL of -1 will cause the command to be executed only once. A TTL of 0 will cause the command to be run every time varattr is requested. This parameter may be used multiple times, but all output will be grouped into a single "varattr" attribute in the request and status output. If the command has no output, the name will be skipped in the output.

  ```
  $varattrseta
  $varattrsetb
  ```

- xauthpath

  Specifies the path to the xauth binary to enable X11 fowarding.

- ignvmem

  If set to true, then pbs_mom will ignore vmem/pvmem limit enforcement.

- ignwalltime

If set to true, then pbs_mom will ignore walltime limit enforcement.

- mom_host
  Sets the local hostname as used by pbs_mom.

Resources

Resource Manager queries can be made with momctl -q options to retrieve and set pbs_mom options. Any configured static resource may be retrieved with a request of the same name. These are resource requests not otherwise documented in the PBS ERS.

- cycle
  Forces an immediate MOM cycle.
- status_update_time
  Retrieve or set the $status_update_time parameter.
- check_poll_time
  Retrieve or set the $check_poll_time parameter.
- configversion
  Retrieve the config version.
- jobstartblocktime
  Retrieve or set the $jobstartblocktime parameter.
- enablemomrestart
  Retrieve or set the $enablemomrestart parameter.
- loglevel
  Retrieve or set the $loglevel parameter.
- down_on_error
  Retrieve or set the EXPERIMENTAL $down_on_error parameter.
- diag0 - diag4
  Retrieves various diagnostic information.
- rcpcmd
  Retrieve or set the $rcpcmd parameter.
- version
  Retrieves the pbs_mom version.

Health Check

The health check script is executed directly by the pbs_mom daemon under the root user id. It must be accessible from the compute node and may be a script or compiled executable program. It may make any needed system calls and execute any combination of system utilities but should not execute resource manager client commands. Also, as of TORQUE 1.0.1, the pbs_mom daemon blocks until the health check is completed and does not possess a built-in timeout. Consequently, it is advisable to keep the launch script execution time short and verify that the script will not block even under failure conditions.

If the script detects a failure, it should return the keyword Error to stdout followed by an error message. The message (up to 256 characters) immediately following the Error string will be assigned to the node attribute message of the associated node.

If the script detects a failure when run from "jobstart", then the job will be rejected. This should probably only be used with advanced schedulers like Moab so that the job can be routed to another node.

TORQUE currently ignores Error messages by default, but advanced schedulers like Moab can be configured to react appropriately.

If the experimental $down_on_error MOM setting is enabled, the MOM will set itself to state down and report

to pbs_server, and pbs_server will report the node as "down". Additionally, the experimental "down_on_error" server attribute can be enabled which has the same effect but moves the decision to pbs_server. It is redundant to have MOM's $down_on_error and pbs_servers down_on_error features enabled. See "down_on_error" in pbs_server_attributes(7B).

Files

- $PBS_SERVER_HOME/server_name
  Contains the hostname running pbs_server.
- $PBS_SERVER_HOME/mom_priv
  The default directory for configuration files, typically (/usr/spool/pbs)/mom_priv.
- $PBS_SERVER_HOME/mom_logs
  Directory for log files recorded by the server.
- $PBS_SERVER_HOME/mom_priv/prologue
  The administrative script to be run before job execution.
- $PBS_SERVER_HOME/mom_priv/epilogue
  The administrative script to be run after job execution.

Signal Handling

pbs_mom handles the following signals:

- SIGHUP
  Causes pbs_mom to re-read its configuration file, close and reopen the log file, and reinitialize resource structures.
- SIGALRM
  Results in a log file entry. The signal is used to limit the time taken by certain children processes, such as the prologue and epilogue.
- SIGINT and SIGTERM
  Results in pbs_mom exiting without terminating any running jobs. This is the action for the following signals as well: SIGXCPU, SIGXFSZ, SIGCPULIM, and SIGSHUTDN.
- SIGUSR1, SIGUSR2
  Causes the MOM to increase and decrease logging levels, respectively.
- SIGPIPE, SIGINFO
  Are ignored.
- SIGBUS, SIGFPE, SIGILL, SIGTRAP, and SIGSYS
  Cause a core dump if the PBSCOREDUMP environmental variable is defined.

All other signals have their default behavior installed.

Exit Status

If the pbs_mom command fails to begin operation, the server exits with a value greater than zero.

See Also

- pbs_server(8B)
- pbs_scheduler_basl(8B)
- pbs_scheduler_tcl(8B)
- the PBS External Reference Specification
- the PBS Administrators Guide.

pbs_server

(PBS Server)

pbs batch system manager

Synopsis

```
pbs_server [-a active] [-d config_path] [-p port] [-A acctfile]
           [-L logfile] [-M mom_port] [-R momRPP_port] [-S scheduler_port]
           [-h hostname] [-t type] [--ha]
```

Description

The pbs_server command starts the operation of a batch server on the local host. Typically, this command will be in a local boot file such as /etc/rc.local . If the batch server is already in execution, pbs_server will exit with an error. To insure that the pbs_server command is not runnable by the general user community, the server will only execute if its real and effective uid is zero.

The server will record a diagnostic message in a log file for any error occurrence. The log files are maintained in the server_logs directory below the home directory of the server. If the log file cannot be opened, the diagnostic message is written to the system console.

Options

-A *acctfile*
: Specifies an absolute path name of the file to use as the accounting file. If not specified, the file name will be the current date in the PBS_HOME/server_priv/accounting directory.

-a *active*
: Specifies if scheduling is active or not. This sets the server attribute scheduling. If the option argument is "true" ("True", "t", "T", or "1"), the server is active and the PBS job scheduler will be called. If the argument is "false" ("False", "f", "F", or "0), the server is idle, and the scheduler will not be called and no jobs will be run. If this option is not specified, the server will retain the prior value of the scheduling attribute.

-d *config_path*
: Specifies the path of the directory which is home to the servers configuration files, PBS_HOME. A host may have multiple servers. Each server must have a different configuration directory. The default configuration directory is given by the symbol $PBS_SERVER_HOME which is typically var/spool/torque.

-h *hostname*
: Causes the server to start under a different hostname as obtained from gethostname(2). Useful for servers with multiple network interfaces to support connections from clients over an interface that has a hostname assigned that differs from the one that is returned by gethost name(2).

--ha (high availablilty)
: Starts server in high availablility mode.

-L *logfile*
: Specifies an absolute path name of the file to use as the log file. If not specified, the file will be the current date in the PBS_HOME/server_logs directory, see the -d option.

-M *mom_port*
: Specifies the host name and/or port number on which the server should connect the job executor, MOM. The option argument, mom_conn, is one of the forms: host_name, [:]port_number, or host_name:port_number. If host_name not specified, the local host is assumed. If port_number is not specified, the default port is assumed. See the -M option for pbs_mom(8).

-p *port*
: Specifies the port number on which the server will listen for batch requests. If multiple servers are running on a single host, each must have its own unique port number. This option is for use in testing with multiple batch systems on a single host.

-R *mom_RPPport*
Specifies the port number on which the the server should query the up/down status of theMOM. See the -R option for pbs_mom(8).

-S *scheduler_port*
Specifies the port number to which the server should connect when contacting the scheduler. The argument scheduler_conn is of the same syntax as under the -M option.

-t *type*
Specifies the impact on jobs which were in execution, running, when the server shut down. If the running job is not rerunnable or restartable from a checkpoint image, the job is aborted. If the job is rerunnable or restartable, then the actions described below are taken. When the type argument is:

hot
All jobs are requeued except non-rerunnable jobs that were executing. Any rerunnable job which was executing when the server went down will be run immediately. This returns the server to the same state as when it went down. After those jobs are restarted, then normal scheduling takes place for all remaining queued jobs.

If a job cannot be restarted immediately because of a missing resource, such as a node being down, the server will attempt to restart it periodically for upto 5 minutes. After that period, the server will revert to a normal state, as if warm started, and will no longer attempt to restart any remaining jobs which were running prior to the shutdown.

warm
All rerunnable jobs which were running when the server went down are requeued. All other jobs are maintained. New selections are made for which jobs are placed into execution. Warm is the default if -t is not specified.

cold
All jobs are deleted. Positive confirmation is required before this direction is accepted.

create
The server will discard any existing configuration files, queues and jobs, and initialize configuration files to the default values. The server is idled.

Files

TORQUE_HOME/server_priv
default directory for configuration files, typically /usr/spool/pbs/server_priv

TORQUE_HOME/server_logs
directory for log files recorded by the server

Signal Handling

On receipt of the following signals, the server performs the defined action:

SIGHUP
The current server log and accounting log are closed and reopened. This allows for the prior log to be renamed and a new log started from the time of the signal.

SIGINT
Causes an orderly shutdown of pbs_server.

SIGUSR1, SIGUSR2
Causes server to increase and decrease logging levels, respectively.

SIGTERM
Causes an orderly shutdown of pbs_server.

SIGSHUTDN

On systems (Unicos) where SIGSHUTDN is defined, it also causes an orderly shutdown of the server.

SIGPIPE
This signal is ignored.

All other signals have their default behavior installed.

Exit Status

If the server command fails to begin batch operation, the server exits with a value greater than zero.

See Also

qsub (1B), pbs_connect(3B), pbs_mom(8B), pbs_sched_basl(8B), pbs_sched_tcl(8B), pbsnodes(8B), qdisable(8B), qenable(8B), qmgr(1B), qrun(8B), qstart(8B), qstop(8B), qterm(8B), and the PBS External Reference Specification.

pbs_track

starts a new process and informs pbs_mom to start tracking it

Synopsis

```
pbs_track -j <JOBID> [-b] <executable> [args]
```

Description

The pbs_track command tells a pbs_mom daemon to monitor the lifecycle and resource usage of the process that it launches using exec(). The pbs_mom is told about this new process via the Task Manager API, using tm_adopt(). The process must also be associated with a job that already exists on the pbs_mom.

By default, pbs_track will send its PID to TORQUE via tm_adopt(). It will then perform an exec(), causing <executable> to run with the supplied arguments. pbs_track will not return until the launched process has completed because it becomes the launched process.

This command can be considered related to the pbsdsh command which uses the tm_spawn() API call. The pbsdsh command asks a pbs_mom to launch and track a new process on behalf of a job. When it is not desirable or possible for the pbs_mom to spawn processes for a job, pbs_track can be used to allow an external entity to launch a process and include it as part of a job.

This command improves integration with TORQUE and SGI's MPT MPI implementation.

Options

- -j <JOBID>
  Job ID the new process should be associated with.
- -b
  Instead of having pbs_track send its PID to TORQUE, it will fork() first, send the child PID to TORQUE, and then execute from the forked child. This essentially "backgrounds" pbs_track so that it will return after the new process is launched.

Operands

The pbs_track command accepts a path to a program/executable (<executable>) and, optionally, one or more arguments to pass to that program.

Exit Status

Because the pbs_track command becomes a new process (if used without -b), its exit status will match that of the new process. If the -b option is used, the exit status will be zero if no errors occurred before launching the new process.

If pbs_track fails, whether due to a bad argument or other error, the exit status will be set to a non-zero value.

See Also

pbsdsh(1B), tm_spawn(3B)

Appendix B: Server Parameters

TORQUE server parameters are specified using the qmgr command. The **set** subcommand is used to modify the **server** object. For example:

```
> qmgr -c 'set server default_queue=batch'
```

Parameters

acl_hosts	
Format:	<HOST>[,<HOST>]... **or** <HOST>[range] **or** <HOST*> where the asterisk (*) can appear anywhere in the host name
Default:	(only the host running pbs_server may submit jobs)

Description:	Specifies a list of hosts from which jobs may be submitted. Hosts in the server nodes file located at $TORQUE/server_priv/nodes cannot be added to the list using the acl_hosts parameter. To submit batch or interactive jobs (See Configuring Job Submit Hosts) through hosts that are specified in the server nodes file, use the submit_hosts parameter.
	
	`Qmgr: set queue batch acl_hosts = "hostA,hostB"`
	`Qmgr: set queue batch acl_hosts += "hostE,hostF,hostG"`
	
	In version 2.5 and later, the wildcard (*) character can appear anywhere in the host name, and ranges are supported; these specifications also work for managers and operators.
	
	`Qmgr: set server acl_hosts = "galaxy*.tom.org"`
	`Qmgr: set server acl_hosts += "galaxy[0-50].tom.org"`
	`Qmgr: set server managers+=tom@galaxy[0-50].tom.org`

acl_host_enable	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Specifies if the **acl_hosts** value is enabled

acl_logic_or	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Specifies if user and group queue ACL's should be logically AND'd or logically OR'd

acl_roots	
Format:	<username>@<domain>
Default:	**---**
Description:	Specifies which root users are allowed to submit and run jobs.

allow_node_submit	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Specifies if users can submit jobs directly from any trusted compute host directly or from within batch jobs (See Configuring Job Submit Hosts)

allow_proxy_user	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Specifies users can proxy from one user to another. Proxy requests will be either validated by *ruserok()* or by the scheduler. (See Job Submission Configuration.)

auto_node_np	
Format:	<BOOLEAN>
Default:	**DISABLED**
Description:	Automatically configures a node's np (number of processors) value based on the ncpus value from the status update. Requires full manager privilege to set or alter.

checkpoint_defaults	
Format:	<STRING>
Default:	---
Description:	Specifies for a queue the default checkpoint values for a job that does not have checkpointing specified. The checkpoint_defaults parameter only takes effect on execution queues. `set queue batch checkpoint_defaults="enabled, periodic, interval=5"`

clone_batch_delay	
Format:	<INTEGER>
Default:	1
Description:	Specifies the delay (in seconds) between clone batches.

clone_batch_size	
Format:	<INTEGER>
Default:	256
Description:	Job arrays are created in batches of size *X*. *X* jobs are created, and after the clone_batch_delay, *X* more are created. This repeats until all are created.

default_queue	
Format:	<STRING>

Default:	---
Description:	Indicates the queue to assign to a job if no queue is explicitly specified by the submitter

disable_server_id_check	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Makes it so the user for the job doesn't have to exist on the server. The user must still exist on all the compute nodes or the job will fail when it tries to execute.  If you have **disable_server_id_check** set to TRUE, a user could request a group to which they do not belong. Setting VALIDATEGROUP to TRUE in the torque.cfg file prevents such a scenario.

job_log_file_max_size	
Format:	<INTEGER>
Default:	---
Description:	This specifies a soft limit (in kilobytes) for the job log's maximum size. The file size is checked every five minutes and if the **current day** file size is greater than or equal to this value, it is rolled from <*filename*> to <*filename.1*> and a new empty log is opened. If the current day file size exceeds the maximum size a second time, the <*filename.1*> log file is rolled to <*filename.2*>, the current log is rolled to <*filename.1*>, and a new empty log is opened. Each new log causes all other logs to roll to an extension that is one greater than its current number. Any value less than 0 is ignored by pbs_server (meaning the log will not be rolled).

job_log_file_roll_depth	
Format:	<INTEGER>
Default:	---
Description:	This sets the maximum number of new log files that are kept in a day if the job_log_file_max_size parameter is set. For example, if the roll depth is set to 3, no file can roll higher than <*filename.3*>. If a file is already at the specified depth, such as <*filename.3*>, the file is deleted so it can be replaced by the incoming file roll, <*filename.2*>.

job_log_keep_days	
Format:	<INTEGER>
Default:	---
Description:	This maintains logs for the number of days designated. If set to 4, any log file older than 4 days old is deleted.

job_nanny	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Enables the experimental "job deletion nanny" feature. All job cancels will create a repeating task that will resend KILL signals if the initial job cancel failed. Further job cancels will be rejected with the message "job cancel in progress." This is useful for temporary failures with a

	job's execution node during a job delete request.

job_stat_rate	
Format:	<INTEGER>
Default:	45 (set to 30 in TORQUE 1.2.0p5 and earlier)
Description:	Specifies the maximum age of MOM level job data which is allowed when servicing a qstat request. If data is older than this value, the pbs_server daemon will contact the MOMs with stale data to request an update.For large systems, this value should be increased to 5 minutes or higher.

job_stat_timeout	
Format:	<INTEGER>
Default:	---
Description:	Specifies the pbs_server to pbs_mom TCP socket timeout in seconds that is used when the pbs_server sends a job start to the pbs_mom. It is useful when the MOM has extra overhead involved in starting jobs. If not specified, then the tcp_timeout value is used. `set server job_start_timeout=15`

lock_file	
Format:	<STRING>
Default:	`torque/server_priv/server.lock`
Description:	Specifies the name and location of the lock file used to determine which high availability server should be active. If a full path is specified, it is used verbatim by TORQUE. If a relative path is specified, TORQUE will prefix it with `torque/server_priv`.

lock_file_update_time	
Format:	<INTEGER>
Default:	3
Description:	Specifies how often (in seconds) the thread will update the lockfile.

lock_file_check_time	
Format:	<INTEGER>
Default:	9
Description:	Specifies how often (in seconds) a high availability server will check to see if it should become active.

log_events	
Format:	Bitmap
Default:	---

Description: By default, all events are logged. However, you can customize things so that only certain events show up in the log file. These are the bitmaps for the different kinds of logs:

```
#define PBSEVENT_ERROR 0x0001 /* internal errors */
#define PBSEVENT_SYSTEM 0x0002 /* system (server) events */
#define PBSEVENT_ADMIN 0x0004 /* admin events */
#define PBSEVENT_JOB 0x0008 /* job related events */
#define PBSEVENT_JOB_USAGE 0x0010 /* End of Job accounting */
#define PBSEVENT_SECURITY 0x0020 /* security violation events */
#define PBSEVENT_SCHED 0x0040 /* scheduler events */
#define PBSEVENT_DEBUG 0x0080 /* common debug messages */
#define PBSEVENT_DEBUG2 0x0100 /* less needed debug messages */
#define PBSEVENT_FORCE 0x8000 /* set to force a message */
```

If you want to log only error, system, and job information, set log_events to 11:

```
set server log_events = 11
```

log_file_max_size	
Format:	<INTEGER>
Default:	0
Description:	Specifies a soft limit, in kilobytes, for the server's log file. The filesize is checked every 5 minutes, and if the **current day** filesize is greater than or equal to this value then it will be rolled from X to X.1 and a new empty log will be opened. Any value 0 will be ignored by pbs_server (the log will not be rolled).

log_file_roll_depth	
Format:	<INTEGER>
Default:	1
Description:	This parameter controlls how deep the **current day** log files will be rolled, if log_file_max_size is set, before they are deleted.

log_keep_days	
Format:	<INTEGER>
Default:	0
Description:	This specifies how long (in days) a server or MOM log should be kept.

log_level	
Format:	<INTEGER>
Default:	0
Description:	Specifies the pbs_server logging verbosity. Maximum value is 7.

mail_body_fmt	
Format:	A printf-like format string

Default:	PBS Job Id: %i Job Name: %j Exec host: %h %m %d
Description:	Override the default format for the body of outgoing mail messages. A number of printf-like format specifiers and escape sequences can be used: \n new line \t tab \\ backslash \' single quote \" double quote %d details concerning the message %h PBS host name %i PBS job identifier %j PBS job name %m long reason for message %r short reason for message %% a single %

mail_domain	
Format:	<STRING>
Default:	---
Description:	Override the default domain for outgoing mail messages. If set, emails will be addressed to <user>@<hostdomain>. If unset, the job's Job_Owner attribute will be used. If set to never, TORQUE will never send emails.

mail_subject_fmt	
Format:	A printf-like format string
Default:	PBS JOB %i
Description:	Override the default format for the subject of outgoing mail messages. A number of printf-like format specifiers and escape sequences can be used: \n new line \t tab \\ backslash \' single quote \" double quote %d details concerning the message

	%h PBS host name %i PBS job identifier %j PBS job name %m long reason for message %r short reason for message %% a single %

managers	
Format:	<user>@<host.sub.domain>[,<user>@<host.sub.domain>...]
Default:	root@localhost
Description:	List of users granted batch administrator privileges. The host, sub-domain, or domain name may be wildcarded by the use of an asterisk character (*). Requires full manager privilege to set or alter.

max_job_array_size	
Format:	<INTEGER>
Default:	Unlimited
Description:	Sets the maximum number of jobs that can be in a single job array.

max_slot_limit	
Format:	<INTEGER>
Default:	Unlimited

Description:	This is the maximum number of jobs that can run concurrently in any job array. Slot limits can be applied at submission time with qsub, or it can be modifed with qalter.
	`qmgr -c 'set server max_slot_limit=10'`
	No array can request a slot limit greater than 10. Any array that does not request a slot limit receives a slot limit of 10. Using the example above, slot requests greater than 10 are rejected with the message:
	
	`Requested slot limit is too large, limit is 10.`

mom_job_sync	
Format:	<BOOLEAN>
Default:	**TRUE**
Description:	Specifies that the pbs_server will synchronize its view of the job queue and resource allocation with compute nodes as they come online. If a job exists on a compute node in a pre-execution or corrupt state, it will be automatically cleaned up and purged. (enabled by default in TORQUE 2.2.0 and higher)

np_default	
Format:	<INTEGER>
Default:	---
Description:	Allows the administrator to unify the number of processors (np) on all nodes. The value can be dynamically changed. A value of 0 tells pbs_server to use the value of np found in the nodes file. The maximum value is 32767.

operators	
Format:	<user>@<host.sub.domain>[,<user>@<host.sub.domain>...]
Default:	root@localhost
Description:	List of users granted batch operator privileges. Requires full manager privilege to set or alter.

node_check_rate	
Format:	<INTEGER>
Default:	600
Description:	Specifies the minimum duration (in seconds) that a node can be unresponsive to server queries before being marked down by the pbs_server daemon

node_pack	
Format:	<BOOLEAN>
Default:	---
Description:	Controls how multiple processor nodes are allocated to jobs. If this attribute is set to true, jobs will be assigned to the multiple processor nodes with the fewest free processors. This packs jobs into the fewest possible nodes leaving multiple processor nodes free for jobs which need many processors on a node. If set to false, jobs will be scattered across nodes reducing conflicts over memory between jobs. If unset, the jobs are packed on nodes in the order that the nodes are declared to the server (in the nodes file). Default value: unset - assigned to nodes as nodes in order that were declared.

node_ping_rate	
Format:	<INTEGER>
Default:	300
Description:	Specifies the maximum interval (in seconds) between successive *pings* sent from the **pbs_server** daemon to the **pbs_mom** daemon to determine node/daemon health.

poll_jobs	
Format:	<BOOLEAN>
Default:	**TRUE** (FALSE in TORQUE 1.2.0p5 and earlier)
Description:	If set to **TRUE**, pbs_server will poll job info from MOMs over time and will not block on handling requests which require this job information. If set to **FALSE**, no polling will occur and if requested job information is stale, pbs_server may block while it attempts to update this information. For large systems, this value should be set to **TRUE**.

query_other_jobs	
Format:	<BOOLEAN>
Default:	**FALSE**
Description:	Specifies whether or not non-admin users may view jobs they do not own

record_job_info	
Format:	<BOOLEAN>
Default:	FALSE
Description:	This must be set to true in order for job logging to be enabled.

record_job_script	
Format:	<BOOLEAN>
Default:	FALSE
Description:	If set to *TRUE*, this adds the contents of the script executed by a job to the log.

resources_available	
Format:	<STRING>
Default:	---
Description:	Allows overriding of detected resource quantity limits (see queue resources_available). **pbs_server** must be restarted for changes to take effect. Also, resources_available is constrained by the smallest of queue.resources_available and the server.resources_available.

submit_hosts	
Format:	"<HOSTNAME>[,<HOSTNAME>]..."
Default:	---
Description:	Indicates which hosts included in the server nodes file located at $TORQUE/server_priv/nodes can submit batch or interactive jobs (See Configuring Job Submit Hosts). For more information on adding hosts that are not included in the first nodes file, see the acl_hosts parameter.

tcp_timeout	
Format:	<INTEGER>
Default:	8
Description:	Specifies the **pbs_server** to **pbs_mom** TCP socket timeout in seconds. (see Considerations for Large Clusters)

Appendix C: Node Manager (MOM) Configuration

Under TORQUE, MOM configuration is accomplished using the `mom_priv/config` file located in the PBS directory on each execution server.

C.1 Parameters

arch	
Format:	<STRING>
Description:	specifies the architecture of the local machine. This information is used by the scheduler only.
Example:	`arch ia64`

$alias_server_name	
Format:	<STRING>
Description:	*(Applicable in version 2.5.0 and later.)* Allows the MOM to accept an additional pbs_server host name as a trusted address. This feature was added to overcome a problem with UDP and RPP where alias IP addresses are used on a server. With alias IP addresses a UDP packet can be sent to the alias address but the UDP reply packet will come back on the primary IP address. RPP matches addresses from its connection table to incoming packets. If the addresses do not match an entry in the RPP table, the packet is dropped. This feature allows an additional address for the server to be added to the table so legitimate packets are not dropped.
Example:	`$alias_server_name node01`

$clienthost	
Format:	<STRING>
Description:	specifies the machine running **pbs_server** This parameter is deprecated, use pbsserver
Example:	`$clienthost node01.teracluster.org`

$check_poll_time	
Format:	<STRING>
Description:	amount of time between checking running jobs, polling jobs, and trying to resend obituariess for jobs that haven't sent successfully. Default is 45 seconds.
Example:	`$check_poll_time 90`

$configversion	
Format:	<STRING>

Description:	specifies the version of the config file data
Example:	`$configversion 113`

$cputmult	
Format:	<FLOAT>
Description:	cpu time multiplier.  If set to 0.0, MOM level cputime enforcement is disabled.
Example:	`$cputmult 2.2`

$ideal_load	
Format:	<FLOAT>
Description:	ideal processor load
Example:	`$ideal_load 4.0`

$igncput	
Format:	<BOOLEAN>
Description:	ignores limit violations pertaining to cpu time. Default is false.
Example:	`$igncput true`

$ignmem	
Format:	<BOOLEAN>
Description:	ignores limit violations pertaining to physical memory. Default is false.
Example:	`$ignmem true`

$ignvmem	
Format:	<BOOLEAN>
Description:	ignores limit violations pertaining to virtual memory. Default is false.
Example:	`$ignvmem true`

$ignwalltime	
Format:	<BOOLEAN>
Description:	ignore walltime (do not enable mom based walltime limit enforcement)
Example:	`$ignwalltime true`

$job_output_file_umask	
Format:	<STRING>
Description:	uses the specified umask when creating job output and error files. Values can be specified in base 8, 10, or 16; leading 0 implies octal and leading 0x or 0X hexadecimal. A value of

	"userdefault" will use the user's default umask. This parameter is in version 2.3.0 and later.
Example:	`$job_output_file_umask 027`

$log_directory	
Format:	<STRING>
Description:	changes the log directory. Default is `TORQUE_HOME/mom_logs/`. `TORQUE_HOME` default is `/var/spool/torque/` but can be changed in the `./configure` script. The value is a string and should be the full path to the desired mom log directory.
Example:	`$log_directory /opt/torque/mom_logs/`

$log_file_suffix	
Format:	<STRING>
Description:	optional suffix to append to log file names. If %h is the suffix, **pbs_mom** appends the hostname for where the log files are stored if it knows it, otherwise it will append the hostname where the mom is running.
Example:	$log_file_suffix %h = 20100223.mybox $log_file_suffix foo = 20100223.foo

$logevent	
Format:	<STRING>
Description:	specifies a bitmap for event types to log
Example:	`$logevent 255`

$loglevel	
Format:	<INTEGER>
Description:	specifies the verbosity of logging with higher numbers specifying more verbose logging. Values may range between 0 and 7.
Example:	`$loglevel 4`

$log_file_max_size	
Format:	<INTEGER>
Description:	Soft limit for log file size in kilobytes. Checked every 5 minutes. If the log file is found to be greater than or equal to log_file_max_size the current log file will be moved from X to X.1 and a new empty file will be opened.
Example:	`$log_file_max_size = 100`

$log_file_roll_depth	
Format:	<INTEGER>
Description:	specifies how many times a log fill will be rolled before it is deleted.
Example:	`$log_file_roll_depth = 7`

$log_keep_days	
Format:	<INTEGER>
Description:	Specifies how many days to keep log files. pbs_mom deletes log files older than the specified number of days. If not specified, pbs_mom won't delete log files based on their age.
Example:	`$log_keep_days 10`

$max_load	
Format:	<FLOAT>
Description:	maximum processor load
Example:	`$max_load 4.0`

$memory_pressure_duration	
Format:	<INTEGER>
Description:	Memory pressure duration sets a limit to the number of times the value of memory_pressure_threshold can be exceeded before a process is terminated. For more information see $memory_pressure_threshold.
Example:	`$memory_pressure_duration 5`

$memory_pressure_threshold	
Format:	<INTEGER>
Description:	See the cpuset man page for more information concerning memory pressure. The memory_pressure of a cpuset provides a simple per-cpuset running average of the rate that the processes in a cpuset are attempting to free up in-use memory on the nodes of the cpuset to satisfy additional memory requests. The memory_pressure_threshold is an integer number used to compare against the reclaim rate provided by the memory_pressure file. If the threshold is exceeded and memory_pressure_duration is set, then the process terminates after exceeding the threshold by the number of times set in memory_pressure_duration. If memory_pressure duration is not set, then a warning is logged and the process continues.
Example:	`$memory_pressure_threshold 1000`

$node_check_script	
Format:	<STRING>
Description:	specifies the fully qualified pathname of the health check script to run. (see Health Check for more information)
Example:	`$node_check_script /opt/batch_tools/nodecheck.pl`

$node_check_interval	
Format:	<INTEGER>
Description:	specifies the number of MOM intervals between subsequent executions of the specified health check. This value default to 1 indicating the check is run every mom interval (see Health Check for more information).

	$node_check_interval has two special strings that can be set: • jobstart - makes the node health script run when a job is started. • jobend - makes the node health script run after each job has completed on a node.
Example:	`$node_check_interval 5`

$nodefile_suffix	
Format:	<STRING>
Description:	Specifies the suffix to append to a host names to denote the data channel network adapter in a multihomed compute node.
Example:	`$nodefile_suffix i` With the suffix of '`i`' and the control channel adapter with the name `node01`, the data channel would have a hostname of `node01i`.

$nospool_dir_list	
Format:	<STRING>
Description:	If this is configured, the job's output is spooled in the working directory of the job or the specified output directory. Specify the list in full paths, delimited by commas. If the job's working directory (or specified output directory) is in one of the paths in the list (or a subdirectory of one of the paths in the list), the job is spooled directly to the output location. `$nospool_dir_list *` is accepted. The user that submits the job must have write permission on the folder where the job is written, and read permission on the folder where the file is spooled. Alternatively, you can use the $spool_as_final_name parameter to force the job to spool directly to the final output. This should generally be used only when the job can run on the same machine as where the output file goes, or if there is a shared filesystem. If not, this parameter can slow down the system or fail to create the output file.
Example:	`$nospool_dir_list /home/mike/jobs/,/var/tmp/spool/`

opsys	
Format:	<STRING>
Description:	specifies the operating system of the local machine. This information is used by the scheduler only.
Example:	`opsys RHEL3`

$pbsclient	
Format:	<STRING>
Description:	specifies machines which the mom daemon will trust to run resource manager commands via momctl. This may include machines where monitors, schedulers, or admins require the use of this command.)

Example:	`$pbsclient node01.teracluster.org`

$pbsserver	
Format:	<STRING>
Description:	specifies the machine running **pbs_server**  This parameter replaces the deprecated parameter clienthost.
Example:	`$pbsserver node01.teracluster.org`

$prologalarm	
Format:	<INTEGER>
Description:	Specifies maximum duration (in seconds) which the mom will wait for the job prologue or job epilogue to complete. This parameter defaults to 300 seconds (5 minutes).
Example:	`$prologalarm 60`

$rcpcmd	
Format:	<STRING>
Description:	specifies the full path and optional additional command line args to use to perform remote copies
Example:	`mom_priv/config`: `$rcpcmd /usr/local/bin/scp -i /etc/sshauth.dat`

$remote_reconfig	
Format:	<STRING>
Description:	Enables the ability to remotely reconfigure pbs_mom with a new config file. Default is disabled. This parameter accepts various forms of true, yes, and 1. For more information on how to reconfigure MOMs, see momctl -r.
Example:	`$remote_reconfig true`

$restricted	
Format:	<STRING>
Description:	Specifies hosts which can be trusted to access mom services as non-root. By default, no hosts are trusted to access mom services as non-root.
Example:	`$restricted *.teracluster.org`

size[fs=<FS>]	
Format:	N/A
Description:	Specifies that the available and configured disk space in the <FS> filesystem is to be reported to the pbs_server and scheduler. To request disk space on a per job basis, specify the `file` resource as in 'qsub -l

	nodes=1,file=1000kb'.
	Unlike most mom config options, the **size** parameter is not preceded by a '**$**' character.
Example:	`size[fs=/localscratch]` the available and configured disk space in the `/localscratch` filesystem will be reported.

$source_login_batch	
Format:	<STRING>
Description:	Specifies whether or not mom will source the /etc/profile, etc. type files for **batch** jobs. Parameter accepts various forms of true, false, yes, no, 1 and 0. Default is True. This parameter is in version 2.3.1 and later.
Example:	`$source_login_batch False` mom will bypass the sourcing of /etc/profile, etc. type files.

$source_login_interactive	
Format:	<STRING>
Description:	Specifies whether or not mom will source the /etc/profile, etc. type files for **interactive** jobs. Parameter accepts various forms of true, false, yes, no, 1 and 0. Default is True. This parameter is in version 2.3.1 and later.
Example:	`$source_login_interactive False` mom will bypass the sourcing of /etc/profile, etc. type files.

$spool_as_final_name	
Format:	<BOOLEAN>
Description:	This will spool the job under the final name that the output and error files will receive, instead of having an intermediate file and then copying the result to the final file when the job has completed. This allows users easier access to the file if they want to watch the jobs output as it runs.
Example:	`$spool_as_final_name true`

$status_update_time	
Format:	<INTEGER>
Description:	Specifies the number of seconds between subsequent mom-to-server update reports. Default is 45 seconds
Example:	status_update_time: `$status_update_time 120` mom will send server update reports every 120 seconds.

$timeout	
Format:	<INTEGER>
Description:	Specifies the number of seconds before mom-to-mom messages will timeout if RPP is disabled. Default is 60 seconds
Example:	`$timeout 120` mom-to-mom communication will allow up to 120 seconds before timing out.

$tmpdir	
Format:	<STRING>
Description:	Specifies a directory to create job-specific scratch space (see Creating Per-Job Temporary Directories
Example:	`$tmpdir /localscratch`

$usecp	
Format:	<HOST>:<SRCDIR> <DSTDIR>
Description:	Specifies which directories should be staged (see TORQUE Data Management)
Example:	`$usecp *.fte.com:/data /usr/local/data`

varattr	
Format:	<INTEGER> <STRING>
Description:	Provides a way to keep track of dynamic attributes on nodes. <INTEGER> is how many seconds should go by between calls to the script to update the dynamic values. If set to -1, the script is read only one time. <STRING> is the script path. This script should check for whatever dynamic attributes are desired, and then output lines in this format: name=value Include any arguments after the script's full path. These features are visible in the output of pbsnodes -a: varattr=Matlab=7.1;Octave=1.0.
Example:	`varattr 25 /usr/local/scripts/nodeProperties.pl arg1 arg2 arg3`

$wallmult	
Format:	<FLOAT>
Description:	Sets a factor to adjust walltime usage by multiplying a default job time to a common reference system. It modifies real walltime on a per-MOM basis (MOM configuration parameters). The factor is used for walltime calculations and limits in the same way that cputmult is used for cpu time.

	If set to 0.0, MOM level walltime enforcement is disabled.
Example:	`$wallmult 2.2`

C.2 Node Features and Generic Consumable Resource Specification

Node features (a.k.a. **node properties**) are *opaque* labels which can be applied to a node. They are not consumable and cannot be associated with a value. (use generic resources described below for these purposes). Node features are configured within the global nodes file on the pbs_server head node and are not specified on a per node basis. This file can be used to specify an arbitrary number of node features.

Additionally, per node *consumable* generic resources may be specified using the format '<ATTR> <VAL>' with no leading dollar ('**$**') character. When specified, this information is routed to the scheduler and can be used in scheduling decisions. For example, to indicate that a given host has two tape drives and one node-locked matlab license available for batch jobs, the following could be specified:

mom_priv/config:

```
$clienthost 241.13.153.7

tape    2
matlab  1
```

Dynamic Consumable Generic Resources

Dynamic consumable resource information can be routed in by specifying a value preceded by a *exclamation point* (i.e., '!') as in the example below. If the resource value is configured in this manner, the specified file will be periodically executed to load the effective resource value. (See section 2.5.3 of the 'PBS Administrator Guide' for more information)

mom_priv/config:

```
$clienthost 241.13.153.7

tape    !/opt/rm/gettapecount.pl
matlab  !/opt/tools/getlicensecount.pl
```

C.3 Command-line Arguments

Below is a table of pbs_mom command-line startup flags.

Flag	Description
a <integer>	Alarm time in seconds.
c <file>	Config file path.
C <directory>	Checkpoint path.
d <directory>	Home directory.
L <file>	Logfile.
M <integer>	MOM port to listen on.
p	Perform 'poll' based job recovery on restart (jobs persist until associated processes terminate).

P	On restart, deletes all jobs that were running on MOM (Available in 2.4.X and later).
q	On restart, requeues all jobs that were running on MOM (Available in 2.4.X and later).
r	On restart, kills all processes associated with jobs that were running on MOM, and then requeues the jobs.
R <integer>	MOM 'RM' port to listen on.
S <integer>	pbs_server port to connect to.
v	Display version information and exit.
x	Disable use of privileged port.
?	Show usage information and exit.

*For more details on these command-line options, see the pbs_mom man page.

See Also

- Server Commands
- Setting up Prolog and Epilog Scripts

Appendix D: Diagnostics and Error Codes

D.1 TORQUE Diagnostics

TORQUE has a diagnostic script to assist you in giving TORQUE Support the files they need to support issues. It should be run by a user that has access to run all TORQUE commands and access to all TORQUE directories (this is usually root).

The script (`contrib/diag/tdiag.sh`) is available in TORQUE 2.3.8, TORQUE 2.4.3, and later. The script grabs the nodefile, server and MOM logfiles, and captures the output of **`qmgr -c 'p s'`**. These are put in a tarfile.

The script also has the following options (this can be shown in the command line by entering `./tdiag.sh -h`):

```
USAGE: ./torque_diag [-d DATE] [-h] [-o OUTPUT_FILE] [-t TORQUE_HOME]
```

`DATE` should be in the format `YYYYmmdd`. For example, `20091130` would be the date for November 30th, 2009. If no date is specified, today's date is used. `OUTPUT_FILE` is the optional name of the output file. The default output file is `torque_diag<today's_date>.tar.gz`. `TORQUE_HOME` should be the path to your TORQUE directory. If no directory is specified, `/var/spool/torque` is the default.

D.2 TORQUE Error Codes

Error Code Name	Number	Description
PBSE_NONE	15000	No error
PBSE_UNKJOBID	15001	Unknown job identifier
PBSE_NOATTR	15002	Undefined attribute
PBSE_ATTRRO	15003	Attempt to set READ ONLY attribute
PBSE_IVALREQ	15004	Invalid request
PBSE_UNKREQ	15005	Unknown batch request
PBSE_TOOMANY	15006	Too many submit retries
PBSE_PERM	15007	No permission
PBSE_BADHOST	15008	Access from host not allowed
PBSE_JOBEXIST	15009	Job already exists
PBSE_SYSTEM	15010	System error occurred
PBSE_INTERNAL	15011	Internal server error occurred
PBSE_REGROUTE	15012	Parent job of dependent in rte queue
PBSE_UNKSIG	15013	Unknown signal name
PBSE_BADATVAL	15014	Bad attribute value
PBSE_MODATRRUN	15015	Cannot modify attribute in run state
PBSE_BADSTATE	15016	Request invalid for job state
PBSE_UNKQUE	15018	Unknown queue name
PBSE_BADCRED	15019	Invalid credential in request
PBSE_EXPIRED	15020	Expired credential in request

PBSE_QUNOENB	15021	Queue not enabled
PBSE_QACESS	15022	No access permission for queue
PBSE_BADUSER	15023	Bad user - no password entry
PBSE_HOPCOUNT	15024	Max hop count exceeded
PBSE_QUEEXIST	15025	Queue already exists
PBSE_ATTRTYPE	15026	Incompatible queue attribute type
PBSE_QUEBUSY	15027	Queue busy (not empty)
PBSE_QUENBIG	15028	Queue name too long
PBSE_NOSUP	15029	Feature/function not supported
PBSE_QUENOEN	15030	Cannot enable queue, needs add def
PBSE_PROTOCOL	15031	Protocol (ASN.1) error
PBSE_BADATLST	15032	Bad attribute list structure
PBSE_NOCONNECTS	15033	No free connections
PBSE_NOSERVER	15034	No server to connect to
PBSE_UNKRESC	15035	Unknown resource
PBSE_EXCQRESC	15036	Job exceeds queue resource limits
PBSE_QUENODFLT	15037	No default queue defined
PBSE_NORERUN	15038	Job not rerunnable
PBSE_ROUTEREJ	15039	Route rejected by all destinations
PBSE_ROUTEEXPD	15040	Time in route queue expired
PBSE_MOMREJECT	15041	Request to the MOM failed
PBSE_BADSCRIPT	15042	(qsub) cannot access script file
PBSE_STAGEIN	15043	Stage In of files failed
PBSE_RESCUNAV	15044	Resources temporarily unavailable
PBSE_BADGRP	15045	Bad group specified
PBSE_MAXQUED	15046	Max number of jobs in queue
PBSE_CKPBSY	15047	Checkpoint busy, may be retries
PBSE_EXLIMIT	15048	Limit exceeds allowable
PBSE_BADACCT	15049	Bad account attribute value
PBSE_ALRDYEXIT	15050	Job already in exit state
PBSE_NOCOPYFILE	15051	Job files not copied
PBSE_CLEANEDOUT	15052	Unknown job id after clean init
PBSE_NOSYNCMSTR	15053	No master in Sync Set

PBSE_BADDEPEND	15054	Invalid dependency
PBSE_DUPLIST	15055	Duplicate entry in List
PBSE_DISPROTO	15056	Bad DIS based request protocol
PBSE_EXECTHERE	15057	Cannot execute there
PBSE_SISREJECT	15058	Sister rejected
PBSE_SISCOMM	15059	Sister could not communicate
PBSE_SVRDOWN	15060	Requirement rejected -server shutting down
PBSE_CKPSHORT	15061	Not all tasks could checkpoint
PBSE_UNKNODE	15062	Named node is not in the list
PBSE_UNKNODEATR	15063	Node-attribute not recognized
PBSE_NONODES	15064	Server has no node list
PBSE_NODENBIG	15065	Node name is too big
PBSE_NODEEXIST	15066	Node name already exists
PBSE_BADNDATVAL	15067	Bad node-attribute value
PBSE_MUTUALEX	15068	State values are mutually exclusive
PBSE_GMODERR	15069	Error(s) during global modification of nodes
PBSE_NORELYMOM	15070	Could not contact the MOM
PBSE_NOTSNODE	15071	No time-shared nodes

Appendix E: Considerations Before Upgrading

TORQUE is flexible in regards to how it can be upgraded. In most cases, a TORQUE *shutdown* followed by a **configure**, **make**, **make install** procedure as documented in the TORQUE Administrator's Guide is all that is required. This process will preserve existing configuration and in most cases, existing workload.

A few considerations are included below:

- If upgrading from OpenPBS, PBSPro, or TORQUE 1.0.3 or earlier, queued jobs whether active or idle will be lost. In such situations, job queues should be completely drained of all jobs.
- If not using the **pbs_mom** -r or -p flag, running jobs may be lost. In such cases, running jobs should be allowed to completed or should be requeued before upgrading TORQUE.
- **pbs_mom** and **pbs_server** daemons of differing versions may be run together. However, not all combinations have been tested and unexpected failures may occur.

Upgrade Steps

1. Build new release (do not install)
2. Stop all TORQUE daemons - See qterm and momctl -s
3. Install new TORQUE - use **make install**
4. Start all TORQUE daemons

E.1 Rolling Upgrade

the **enablemomrestart** option causes a MOM to check if its binary has been updated and will restart itself at a safe point when no jobs are running, making upgrades easier. This can be enabled in the MOM config file, but it is recommended to enable it with momctl.

1. Prepare the new version MOM package
2. Install the MOM package on the compute nodes
3. Run **momctl -q enablemomrestart=1 -h :ALL**

Appendix F: Large Cluster Considerations

F.1 Communication Overview

TORQUE has enhanced much of the communication found in the original OpenPBS project. This has resulted in a number of key advantages:

- Support for larger clusters
- Support for more jobs
- Support for larger jobs
- Support for larger messages

In most cases, enhancements made apply to all systems and no tuning is required. However, some changes have been made configurable to allow site specific modification. The configurable communication parameters are:

- node_check_rate
- node_ping_rate
- tcp_timeout

F.2 Scalability Guidelines

In very large clusters (in excess of 1,000 nodes), it may be advisable to additionally tune a number of communication layer timeouts. By default, PBS MOM daemons will timeout on inter-MOM messages after 60 seconds. In TORQUE 1.1.0p5 and higher, this can be adjusted by setting the **timeout** parameter in the mom_priv/config file. If 15059 errors (cannot receive message from sisters) are seen in the MOM logs, it may be necessary to increase this value.

Client-to-PBS server and MOM-to-PBS server communication timeouts are specified via the tcp_timeout server option using the qmgr command.

On some systems, **ulimit** values may prevent large jobs from running. In particular, the open file descriptor limit (i.e., `ulimit -n`) should be set to at least the maximum job size in procs + 20. Further, there may be value in setting the `fs.file-max` in `sysctl.conf` to a high value, such as:

`/etc/sysctl.conf`:

```
fs.file-max = 65536
```

F.3 End User Command Caching

F.3.1 qstat

In a large system, users may tend to place excessive load on the system by manual or automated use of resource manager end user client commands. A simple way of reducing this load is through the use of client command wrappers which cache data. The example script below will cache the output of the command '`qstat -f`' for 60 seconds and report this info to end users.

```
#!/bin/sh

# USAGE: qstat $@

CMDPATH=/usr/local/bin/qstat
CACHETIME=60
TMPFILE=/tmp/qstat.f.tmp

if [ "$1" != "-f" ] ; then
  #echo "direct check (arg1=$1) "
```

```
  $CMDPATH $1 $2 $3 $4
  exit $?
fi

if [ -n "$2" ] ; then
   #echo "direct check (arg2=$2)"
   $CMDPATH $1 $2 $3 $4
   exit $?
fi

if [ -f $TMPFILE ] ; then
  TMPFILEMTIME=`stat -c %Z $TMPFILE`
else
  TMPFILEMTIME=0
fi

NOW=`date +%s`

AGE=$(($NOW - $TMPFILEMTIME))
```

The above script can easily be modified to cache any command and any combination of arguments by changing one or more of the following attributes:

- script name
- value of $CMDPATH
- value of $CACHETIME
- value of $TMPFILE

For example, to cache the command `pbsnodes -a`, make the following changes:

- move original `pbsnodes` command to `pbsnodes.orig`
- save the script as 'pbsnodes'
- change $CMDPATH to `pbsnodes.orig`
- change $TMPFILE to `/tmp/pbsnodes.a.tmp`

F.5 Other Considerations

F.5.1 job_stat_rate

In a large system, there may be many users, many jobs, and many requests for information. To speed up response time for users and for programs using the API the job_stat_rate can be used to tweak when the pbs_server daemon will query MOMs for job information. By increasing this number, a system will not be constantly querying job information and causing other commands to block.

F.5.2 poll_jobs

The poll_jobs parameter allows a site to configure how the pbs_server daemon will poll for job information. When set to **TRUE**, the pbs_server will poll job information in the background and not block on user requests. When set to **FALSE**, the pbs_server may block on user requests when it has stale job information data. Large clusters should set this parameter to **TRUE**.

F.5.3 Internal Settings

On large, slow, and/or heavily loaded systems, it may be desirable to increase the **pbs_tcp_timeout** setting used by the **pbs_mom** daemon in MOM-to-MOM communication. This setting defaults to 20 seconds and requires rebuilding code to adjust. For client-server based communication, this attribute can be set using the qmgr command. For MOM-to-MOM communication, a source code modification is required. To make this change, edit the `$TORQUEBUILDDIR/src/lib/Libifl/tcp_dis.c` file and set **pbs_tcp_timeout** to the desired maximum number of seconds allowed for a MOM-to-MOM request to be serviced.

A system may be heavily loaded if it reports multiple 'End of File from addr' or 'Premature end of message' failures in the **pbs_mom** or **pbs_server** logs.

F.5.4 Scheduler Settings

If using Moab, there are a number of parameters which can be set on the schedululer which may improve TORQUE performance. In an environment containing a large number of short-running jobs, the JOBAGGREGATIONTIME parameter (see Appendix F of the Moab Workload Manager Administrator's Guide) can be set to reduce the number of workload and resource queries performed by the scheduler when an event based interface is enabled. If the **pbs_server** daemon is heavily loaded and PBS API timeout errors (ie. 'Premature end of message') are reported within the scheduler, the **TIMEOUT** attribute of the RMCFG parameter (see Appendix F of the Moab Workload Manager Administrator's Guide) may be set with a value of between 30 and 90 seconds.

F.5.5 File System

TORQUE can be configured to disable file system blocking until data is physically written to the disk by using the `--disable-filesync` argument with **configure**. While having filesync enabled is more reliable, it may lead to server delays for sites with either a larger number of nodes, or a large number of jobs. Filesync is enabled by default.

F.5.6 Network ARP cache

For networks with more than 512 nodes it is mandatory to increase the kernel's internal ARP cache size. For a network of ~1000 nodes, we use these values in **/etc/sysctl.conf** on all nodes and servers:

```
/etc/sysctl.conf
# Don't allow the arp table to become bigger than this
net.ipv4.neigh.default.gc_thresh3 = 4096
# Tell the gc when to become aggressive with arp table cleaning.
# Adjust this based on size of the LAN.
net.ipv4.neigh.default.gc_thresh2 = 2048
# Adjust where the gc will leave arp table alone
net.ipv4.neigh.default.gc_thresh1 = 1024
# Adjust to arp table gc to clean-up more often
net.ipv4.neigh.default.gc_interval = 3600
# ARP cache entry timeout
net.ipv4.neigh.default.gc_stale_time = 3600
```

Use sysctl -p to reload this file.

The ARP cache size on other Unixes can presumably be modified in a similar way.

An alternative approach is to have a static **/etc/ethers** file with all hostnames and MAC addresses and load this by **arp -f /etc/ethers**. However, maintaining this approach is quite cumbersome when nodes get new MAC addresses (due to repairs, for example).

Appendix G: Prologue & Epilogue Scripts

TORQUE provides administrators the ability to run scripts before and/or after each job executes. With such a script, a site can prepare systems, perform node health checks, prepend and append text to output and error log files, cleanup systems, and so forth.

The following table shows which MOM runs which script. All scripts must be in the TORQUE_HOME/mom_priv/ directory and be available on *every* compute node. *Mother Superior*, as referenced in the following table, is the pbs_mom on the first node allocated, and the term *Sisters* refers to pbs_moms, although note that a Mother Superior is also a sister node.

Script	Execution Location	Executed as	Execution Directory	File Permissions
prologue **epilogue** **prologue.<name>** **epilogue.<name>**	Mother Superior	root	TORQUE_HOME/mom_priv/	readable and executable by root and NOT writable by anyone besides root (e.g., `-r-x------`)
prologue.user **epilogue.user**		user		readable and executable by root and other (e.g., `-r-x---r-x`)
prologue.parallel **epilogue.parallel***	Sisters	root		readable and executable by root and NOT writable by anyone besides root (e.g., `-r-x------`)
epilogue.precancel	Mother Superior This script is run after a job cancel request is received from pbs_server and before a kill signal is sent to the job process.			

* available in Version 2.1

The epilogue and prologue scripts are controlled by the system administrator. However, beginning in TORQUE version 2.4 a user epilogue and prologue script can be used on a per job basis. See G.2 Per Job Prologue and Epilogue Scripts for more information.

G.1 Script Environment

The prolog and epilog scripts can be very simple. On most systems, the script must declare the execution shell using the **#!<SHELL>** syntax (i.e., '#!/bin/sh'). In addition, the script may want to process context sensitive arguments passed by TORQUE to the script.

Prolog Environment

The following arguments are passed to the prologue, prologue.user, and prologue.parallel scripts:

Argument	Description
argv[1]	job id

argv[2]	job execution user name
argv[3]	job execution group name
argv[4]	job name (TORQUE 1.2.0p4 and higher only)
argv[5]	list of requested resource limits (TORQUE 1.2.0p4 and higher only)
argv[6]	job execution queue (TORQUE 1.2.0p4 and higher only)
argv[7]	job account (TORQUE 1.2.0p4 and higher only)

Epilog Environment

TORQUE supplies the following arguments to the epilogue, epilogue.user, epilogue.precancel, and epilogue.parallel scripts:

Argument	Description
argv[1]	job id
argv[2]	job execution user name
argv[3]	job execution group name
argv[4]	job name
argv[5]	session id
argv[6]	list of requested resource limits
argv[7]	list of resources used by job
argv[8]	job execution queue
argv[9]	job account
argv[10]	job exit code

The **epilogue.precancel** script is run after a job cancel request is received by the MOM and before any signals are sent to job processes. If this script exists, it is run whether the canceled job was active or idle.

For all scripts, the environment passed to the script is empty. Also, standard input for both scripts is connected to a system dependent file. Currently, for all systems this is **/dev/null**. Except for the epilogue scripts of an interactive job, the standard output and error, are connected to input and error files associated with the job. For an interactive job, since the pseudo terminal connection is released after the job completes, the standard input and error point to **/dev/null**.

G.2 Per Job Prologue and Epilogue Scripts

TORQUE now supports per job prologue and epilogue scripts when using the qsub -l option. The syntax is: `qsub -l prologue=<prologue_script_path> epilogue=<epilogue_script_path> <script>`. The path can be either relative (from the directory where the job is submitted) or absolute. The files must be owned by the user with at least execute and write privileges, and the permissions must not be writeable by group or other.

TORQUE_HOME/mom_priv/:

```
-r-x------ 1 usertom usertom 24 2009-11-09 16:11 prologue_script.sh
-r-x------ 1 usertom usertom 24 2009-11-09 16:11 epilogue_script.sh
```

Example:

```
$ qsub -l prologue=/home/usertom/dev/prologue_script.sh
epilogue=/home/usertom/dev/epilogue_script.sh job14.pl
```

This job submission executes the prologue script first. When the prologue script is complete, job14.pl runs. When job14.pl completes, the epilogue script is executed.

G.3 Prologue and Epilogue Scripts Time Out

TORQUE takes preventative measures against prologue and epilogue scripts by placing an alarm around the scripts execution. By default, TORQUE sets the alarm to go off after 5 minutes of execution. If the script exceeds this time, it will be terminated and the node will be marked down. This timeout can be adjusted by setting the prologalarm parameter in the `mom_priv/config` file.

While TORQUE is executing the `epilogue`, `epilogue.user`, or `epilogue.precancel` scripts, the job will be in the **E** (exiting) state.

G.4 Prologue Error Processing

If the prologue script executes successfully, it should exit with a zero status. Otherwise, the script should return the appropriate error code as defined in the table below. The **pbs_mom** will report the script's exit status to pbs_server which will in turn take the associated action. The following table describes each exit code for the prologue scripts and the action taken.

Error	Description	Action
-4	The script timed out	Job will be requeued
-3	The wait(2) call returned an error	Job will be requeued
-2	Input file could not be opened	Job will be requeued
-1	Permission error (script is not owned by root, or is writable by others)	Job will be requeued
0	Successful completion	Job will run
1	Abort exit code	Job will be aborted
>1	other	Job will be requeued

Example 1

Following are example prologue and epilogue scripts that write the arguments passed to them in the job's standard out file:

prologue	

Script	`#!/bin/sh`
	
	`echo "Prologue Args:"`
	`echo "Job ID: $1"`
	`echo "User ID: $2"`
	`echo "Group ID: $3"`
	`echo ""`
	
	`exit 0`

stdout	
	
	`Prologue Args:`
	`Job ID: 13724.node01`
	`User ID: user1`
	`Group ID: user1`

epilogue	

Script	`#!/bin/sh`
	
	`echo "Epilogue Args:"`
	`echo "Job ID: $1"`
	`echo "User ID: $2"`
	`echo "Group ID: $3"`
	`echo "Job Name: $4"`
	`echo "Session ID: $5"`
	`echo "Resource List: $6"`
	`echo "Resources Used: $7"`
	`echo "Queue Name: $8"`
	`echo "Account String: $9"`
	`echo ""`
	
	`exit 0`

stdout	Epilogue Args: Job ID: 13724.node01 User ID: user1 Group ID: user1 Job Name: script.sh Session ID: 28244 Resource List: neednodes=node01,nodes=1,walltime=00:01:00 Resources Used: cput=00:00:00,mem=0kb,vmem=0kb,walltime=00:00:07 Queue Name: batch Account String:

Example 2

The Ohio Supercomputer Center contributed the following scripts:
"prologue creates a unique temporary directory on each node assigned to a job before the job begins to run, and epilogue deletes that directory after the job completes.

Having a separate temporary directory on each node is probably not as good as having a good, high performance parallel filesystem.

prologue

```
#!/bin/sh
# Create TMPDIR on all the nodes
# Copyright 1999, 2000, 2001 Ohio Supercomputer Center
# prologue gets 3 arguments:
# 1 -- jobid
# 2 -- userid
# 3 -- grpid
#
jobid=$1
user=$2
group=$3
nodefile=/var/spool/pbs/aux/$jobid
if [ -r $nodefile ] ; then
    nodes=$(sort $nodefile | uniq)
else
    nodes=localhost
fi
tmp=/tmp/pbstmp.$jobid
for i in $nodes ; do
```

```
    ssh $i mkdir -m 700 $tmp \&\& chown $user.$group $tmp
done
exit 0
```

epilogue

```
#!/bin/sh
# Clear out TMPDIR
# Copyright 1999, 2000, 2001 Ohio Supercomputer Center
# epilogue gets 9 arguments:
# 1 -- jobid
# 2 -- userid
# 3 -- grpid
# 4 -- job name
# 5 -- sessionid
# 6 -- resource limits
# 7 -- resources used
# 8 -- queue
# 9 -- account
#
jobid=$1
nodefile=/var/spool/pbs/aux/$jobid
if [ -r $nodefile ] ; then
    nodes=$(sort $nodefile | uniq)
else
    nodes=localhost
fi
tmp=/tmp/pbstmp.$jobid
for i in $nodes ; do
    ssh $i rm -rf $tmp
done
exit 0
```

Prologue, prologue.user and prologue.parallel scripts can have dramatic effects on job scheduling if written improperly.

Appendix H: Running Multiple TORQUE Servers and MOMs on the Same Node

TORQUE can be configured to allow multiple servers and MOMs to run on the same node. This example will show how to configure, compile and install two different TORQUE servers and moms on the same node.

H.1 Configuring the First TORQUE

```
./configure --with-server-home=/usr/spool/PBS1 --
bindir=/usr/spool/PBS1/bin --sbindir=/usr/spool/PBS1/sbin
```

Then *make* and *make install* will place the first TORQUE into "/usr/spool/PBS1" with the executables in their corresponding directories.

H.2 Configuring the Second TORQUE

```
./configure --with-server-home=/usr/spool/PBS2 --
bindir=/usr/spool/PBS2/bin --sbindir=/usr/spool/PBS2/sbin
```

Then *make* and *make install* will place the second TORQUE into "/usr/spool/PBS2" with the executables in their corresponding directories.

H.3 Bringing the First TORQUE server online

Each command (including *pbs_server* and *pbs_mom* takes parameters indicating which servers and ports to connect to or listen on (when appropriate). Each of these is documented in their corresponding man pages (configure with --enable-docs).

In this example the first TORQUE server will accept batch requests on port 35000, communicate with the MOMs on port 35001, and communicate via RPP on port 35002. The first TORQUE MOM will try to connect to the server on port 35000, it will listen for requests from the server on port 35001 and will communicate via RPP on port 35002 (Each of these command arguments is discussed in further details on the corresponding man page. In particular, `-t create` is only used the first time a server is run.).

```
> pbs_server -p 35000 -M 35001 -R 35002 -t create
> pbs_mom -S 35000 -M 35001 -R 35002
```

Afterwards, when using a client command to make a batch request it is necessary to specify the servername and serverport (35000):

```
> pbsnodes -a -s node01:35000
```

Submitting jobs can be accomplished using the -q option ([queue][@host[:port]]):

```
> qsub -q @node01:35000 /tmp/script.pbs
```

H.4 Bringing the Second TORQUE Server Online

In this example the second TORQUE server will accept batch requests on port 36000, communicate with the MOMS on port 36002, and communicate via RPP on port 36002. The second TORQUE MOM will try to connect to the server on port 36000, it will listen for requests from the server on port 36001 and will communicate via RPP on port 36002.

```
> pbs_server -p 36000 -M 36001 -R 36002 -t create
> pbs_mom -S 36000 -M 36001 -R 36002
```

Afterward, when using a client command to make a batch request it is necessary to specify the servername

and serverport (36002):

```
> pbsnodes -a -s node01:36000
> qsub -q @node01:36000 /tmp/script.pbs
```

Appendix I: Security Overview

I.1 SUID Usage

TORQUE uses setuid (SUID) permissions in a single location so as to validate the identity of a user request. This is accomplished using the **pbs_iff** tool which is SUID root and performs the following actions:

- parse specified server hostname and port
- connect to specified server port using reserved/privileged port
- determine UID of executing user
- report UID and socket port info of caller to server
- verify response from server

I.2 /etc/hosts Usage

In systems where security is a major concern, please be aware that some security experts consider adding the compute nodes to the /etc/hosts file to be more secure than using ACL lists.

Appendix J: Job Submission Filter (aka 'qsub Wrapper')

When a *submit filter* exists, TORQUE will send the command file (or contents of STDIN if piped to qsub) to that script/executable and allow it to evaluate the submitted request based on specific site policies. The resulting file is then handed back to qsub and processing continues. Submit filters can check user jobs for correctness based on site policies. They can also modify user jobs as they are submitted. Some examples of what a submit filter might evaluate and check for are:

- Memory Request - Verifiy that the job requests memory and rejects if it does not.
- Job event notifications - Check if the job does one of the following and rejects it if it:
  - explicitly requests no notification.
  - requests notifications but does not provide an email address.
- Walltime specified - Verify that the walltime is specified.
- Global Walltime Limit - Verify that the walltime is below the global max walltime.
- Test Walltime Limit - If the job is a test job, this check rejects the job it if it requests a walltime longer than the testing maximum.

The script below reads the original submission request from STDIN and shows how you could insert parameters into a job submit request:

```
#!/bin/sh

# add default memory constraints to all requests
# that did not specify it in user's script or on command line

echo "#PBS -l mem=16MB"

while read i
  do
    echo $i
  done
```

Command line arguments passed to qsub are passed as arguments to the submit filter (filter won't see them in STDIN) in the same order and may be used as needed. It should be noted that as of TORQUE 2.2.0 extended attributes are not passed to the filter. Exit status of -1 will cause qsub to reject the submission with a message stating that it failed due to administrative policies.

The *submit filter* must be executable, must be available on each of the nodes where users may submit jobs, and by default, must be located at `${libexecdir}/qsub_filter` (for version 2.1 and older: `/usr/local/sbin/torque_submitfilter`). At run time, if the file does not exist at this new preferred path then qsub will fall back to the old hard-coded path. The submit filter location can be customized by setting the **SUBMITFILTER** parameter inside the torque.cfg file, as in the following example:

`torque.cfg`:

```
SUBMITFILTER /opt/torque/submit.pl
...
```

*Initial development courtesy of Oak Ridge National Laboratories

Appendix K: torque.cfg Configuration File

The `torque.cfg` file should be placed in the TORQUE home directory (i.e., `/var/spool/torque`). Below is a list of `torque.cfg` parameters:

CLIENTRETRY	
Format:	<INT>
Default:	0
Description:	Seconds between retry attempts to talk to pbs_server

DEFAULTCKPT	
Format:	<STRING>
Default:	None
Description:	Default value for job's checkpoint attribute

FAULT_TOLERANT_BY_DEFAULT	
Format:	<BOOLEAN>
Default:	FALSE
Description:	Sets all jobs to fault tolerant by default. See qsub -f for more information on fault tolerance.

QSUBHOST	
Format:	<HOSTNAME>
Default:	None

Description:	Specify the hostname where pbs_mom will find qsub for interactive jobs.

QSUBSENDUID	
Format:	N/A
Default:	None
Description:	Integer for jobs's PBS_O_UID variable. Specifying the parameter name anywhere in the config file enables the feature. Removing the parameter name disables the feature.

QSUBSLEEP	
Format:	<INT>
Default:	0
Description:	Specifies time to sleep when running qsub command, used to prevent users from overwhelming the scheduler.

RERUNNABLEBYDEFAULT	
Format:	<BOOLEAN>
Default:	TRUE
Description:	Specifies if a job is rerunnable by default. Setting this to false causes the rerunnable attribute value to be false unless the users specifies otherwise with the qsub -r option.. New in TORQUE 2.4.

SERVERHOST	
Format:	<STRING>
Default:	localhost
Description:	If set, the server will open socket connections and communicate with client commands and other services using the specified network interface. (useful with multi-homed hosts, i.e., nodes with multiple network adapters)

SUBMITFILTER	
Format:	<STRING>
Default:	${libexecdir}/qsub_filter (for version 2.1 and older: /usr/local/sbin/torque_submitfilter)
Description:	Specifies the location of the submit filter used to pre-process job submission.

VALIDATEGROUP	
Format:	<BOOLEAN>
Default:	False
Description:	Validate submit user's group on qsub commands. For TORQUE builds released after 2/8/2011, **VALIDATEGROUP** also checks any groups requested in group_list at the submit host. Set **VALIDATEGROUP** to TRUE if you set disable_server_id_check to TRUE.

VALIDATEPATH	
Format:	<BOOLEAN>
Default:	TRUE
Description:	Validate local existence of '-d' working directory

Example:

torque.cfg:

```
QSUBSLEEP               2
SERVERHOST              orion15
```

Appendix L: TORQUE Quick Start Guide

L.1 Initial Installation

- Download the TORQUE distribution file from http://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:

```
> tar -xzvf torque.tar.gz
> cd torque
> ./configure
> make
> make install
```

OSX 10.4 users need to change the *#define __TDARWIN* in `src/include/pbs_config.h` to *#define __TDARWIN_8*.

After installation, verify you have **PATH** environment variables configured for `/usr/local/bin/` and `/usr/local/sbin/`. Client commands are installed to `/usr/local/bin` and server binaries are installed to `/usr/local/sbin.<`

In this document TORQUE_HOME corresponds to where TORQUE stores its configuration files. The default is `/var/spool/torque`.

L.2 Initialize/Configure TORQUE on the Server (pbs_server)

- Once installation on the TORQUE server is complete, configure the pbs_server daemon by executing the command `torque.setup <USER>` found packaged with the distribution source code, where `<USER>` is a username that will act as the TORQUE admin. This script will set up a basic batch queue to get you started. If you experience problems, make sure that the most recent TORQUE executables are being executed, or that the executables are in your current PATH.
- If doing this step manually, be certain to run the command '**pbs_server -t create**' to create the new batch database. If this step is not taken, the **pbs_server** daemon will be unable to start.
- Proper server configuration can be verified by following the steps listed in Section 1.4 Testing

L.3 Install TORQUE on the Compute Nodes

To configure a compute node do the following on each machine (see page 19, Section 3.2.1 of PBS Administrator's Manual for full details):

- 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.

L.4 Configure TORQUE on the Compute Nodes

- For each compute host, the MOM daemon must be configured to trust the **pbs_server** daemon. In TORQUE 2.0.0p4 and earlier, this is done by creating the `TORQUE_HOME/mom_priv/config` file and setting the $pbsserver parameter. In TORQUE 2.0.0p5 and later, this can also be done by creating the `TORQUE_HOME/server_name` file and placing the server hostname inside.
- Additional config parameters may be added to `TORQUE_HOME/mom_priv/config` (See the MOM Config page for details.)

L.5 Configure Data Management on the Compute Nodes

Data management allows jobs' data to be staged in/out or to and from the server and compute nodes.

- For shared filesystems (i.e., NFS, DFS, AFS, etc.) use the **$usecp** parameter in the `mom_priv/config` files to specify how to map a user's home directory.
  (Example: `$usecp gridmaster.tmx.com:/home /home`)
- For local, non-shared filesystems, rcp or scp must be configured to allow direct copy without prompting for passwords (key authentication, etc.)

L.6 Update TORQUE Server Configuration

- On the TORQUE server, append the list of newly configured compute nodes to the `TORQUE_HOME/server_priv/nodes` file:

```
server_priv/nodes
computenode001.cluster.org
computenode002.cluster.org
computenode003.cluster.org
```

L.7 Start the pbs_mom Daemons on Compute Nodes

- Next start the **pbs_mom** daemon on each compute node by running the `pbs_mom` executable.

L.8 Verifying Correct TORQUE Installation

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 # shutdown server

# 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 will not start because there is no scheduler running. The scheduler is enabled in the next step below.

L.9 Enabling the Scheduler

Selecting the cluster scheduler is an important decision and significantly affects cluster utilization, responsiveness, availability, and intelligence. The default TORQUE scheduler, **pbs_sched**, is very basic and will provide poor utilization of your cluster's resources. Other options, such as Maui Scheduler or Moab Workload Manager are highly recommended. If using Maui/Moab, refer to the Moab-PBS Integration Guide. If using **pbs_sched**, start this daemon now.

If you are installing ClusterSuite, TORQUE and Moab were configured at installation for interoperability and no further action is required.

L.10 Startup/Shutdown Service Script for TORQUE/Moab (OPTIONAL)

Optional startup/shutdown service scripts are provided as an example of how to run TORQUE as an OS service that starts at bootup. The scripts are located in the `contrib/init.d/` directory of the TORQUE tarball you downloaded. In order to use the script you must:

- Determine which init.d script suits your platform the best.
- Modify the script to point to TORQUE's install location. This should only be necessary if you used a non-default install location for TORQUE (by using the `--prefix` option of `./configure`).
- Place the script in the `/etc/init.d/` directory.
- Use a tool like chkconfig to activate the start-up scripts or make symbolic links (S99moab and K15moab, for example) in desired runtimes (/etc/rc.d/rc3.d/ on Redhat, etc.).

See Also:

- Advanced Server Configuration

TORQUE Resource Manager Change Log

Legend

- **TORQUE 3.0**
  - TORQUE 3.0.0
- **TORQUE 2.5**
  - TORQUE 2.5.5
  - TORQUE 2.5.4
  - TORQUE 2.5.3
  - TORQUE 2.5.2
  - TORQUE 2.5.1
  - TORQUE 2.5.0
- **TORQUE 2.4**
  - TORQUE 2.4.12
  - TORQUE 2.4.11
  - TORQUE 2.4.10
  - TORQUE 2.4.9
  - TORQUE 2.4.8
  - TORQUE 2.4.7
  - TORQUE 2.4.6
  - TORQUE 2.4.5
  - TORQUE 2.4.4
  - TORQUE 2.4.3
  - TORQUE 2.4.2
  - TORQUE 2.4.1
  - TORQUE 2.4.0
- **TORQUE 2.3**
  - TORQUE 2.3.12 - This is the last offical release of TORQUE 2.3.
  - TORQUE 2.3.11
  - TORQUE 2.3.10
  - TORQUE 2.3.9
  - TORQUE 2.3.8
  - TORQUE 2.3.7
  - TORQUE 2.3.6
  - TORQUE 2.3.5
  - TORQUE 2.3.4
  - TORQUE 2.3.3
  - TORQUE 2.3.2
  - TORQUE 2.3.1
  - TORQUE 2.3.0
- **TORQUE 2.2**
  - TORQUE 2.2.0
- **TORQUE 2.1**
  - TORQUE 2.1.2
  - TORQUE 2.1.1
  - TORQUE 2.1.0p0
- **TORQUE 2.0**
  - TORQUE 2.0.0p6
  - TORQUE 2.0.0p5
  - TORQUE 2.0.0p4
  - TORQUE 2.0.0p3
  - TORQUE 2.0.0p2
  - TORQUE 2.0.0p1
  - TORQUE 2.0.0p0
- **TORQUE 1.2**

Legend

c - crash
b - bug fix
e - enhancement
f - new feature
n - note

TORQUE 3.0

3.0.0

- e - serverdb is now stored as xml; this is no longer configurable.
- f - Added --enable-numa-support for supporting NUMA-type architectures. We have tested this build on UV and Altix machines. The server treats the MOM as a node with several special NUMA nodes embedded, and the pbs_mom reports on these numa nodes instead of itself as a whole.
- f - For NUMA configurations, pbs_mom creates cpusets for memory as well as cpus.
- e - Adapted the task manager interface to interact properly with NUMA systems, including tm_adopt.
- e - Addeded autogen.sh go make life easier in a Makefile.in-less world.
- e - Modified buildutils/pbs_mkdirs.in to create server_priv/nodes file at install time. The file only shows examples and a link to the TORQUE documentation.
- f - Added ATTR_node_exclusive to allow a job to have a node exclusively.
- f - Added --enable-memacct to use an extra protocol in order to accurately track jobs that exceed their memory limits and kill them
- e - When ATTR_node_exclusive is set, reserve the entire node (or entire NUMA node if applicable) in the cpuset.
- n - Changed the protocol versions for all client-to-server, mom-to-server, and mom-to-mom protocols from 1 to 2. The changes to the protocol in this version of TORQUE will make it incompatible with previous versions.
- e - When a select statement is used, tally up the memory requests and mark the total in the resource list. This allows memory enforcement for NUMA jobs, but doesn't affect others as memory isn't enforced for multinode jobs.
- e - Add an asynchronous option to qdel.
- b - Do not reply when an asynchronous reply has already been sent.

- e - Make the mem, vmem, and cput usage available on a per-mom basis using momctl -d2. (Dr. Bernd Kallies)
- e - Move the memory monitor functionality to linux/mom_mach.c in order to store the more accurate statistics for usage, and still use it for applying limits. (Dr. Bernd Kallies)
- e - When pbs_mom is compiled to use cpusets, instead of looking at all processes, only examine the ones in cpuset task files. For busy machines (especially large systems like UVs) this can exponentially reduce job monitoring/harvesting times. (Dr. Bernd Kallies)
- e - When cpusets are configured and memory pressure enabled, add the ability to check memory pressure for a job. Using $memory_pressure_threshold and $memory_pressure_duration in the mom's config, the admin sets a threshold at which a job becomes a problem. If duration is set, the job will be killed if it exceeds the threshold for the configured number of checks. If duration isn't set, then an error is logged. (Dr. Bernd Kallies)
- e - Change pbs_track to look for the executable in the existing path so it doesn't always need a complete path. (Dr. Bernd Kallies)
- e - Report sessions on a per NUMA node basis when NUMA is enabled. (Dr. Bernd Kallies)
- b - Merged revision 4325 from 2.5-fixes. Fixed a problem where the -m n (request no mail on qsub) was not always being recongnized.
- e - Merged buildutils/torque.spec.in from 2.4-fixes. Refactored torque spec file to comply with established RPM best practices, including the following:
  - - Standard installation locations based on RPM macro configuration (e.g., %{_prefix})
  - - Latest upstream RPM conditional build semantics with fallbacks for older versions of RPM (e.g., RHEL4)
  - - Initial set of optional features (GUI, PAM, syslog, SCP) with more planned
  - - Basic working configuration automatically generated at install-time
  - - Reduce the number of unnecessary subpackages by consolidating where it makes sense and using existing RPM features (e.g., --excludedocs)

TORQUE 2.5

2.5.5

- b - change so gpus get written back to nodes file
- e - make it so that even if an array request has multiple consecutive '%' the slot limit will be set correctly

- b - Fixed bug in job_log_open where the global variable logpath was freed instead of joblogpath.
- b - Fixed memory leak in function procs_requested.
- b - Validated incoming data for escape_xml to prevent a seg-fault with incoming null pointers
- e - Added submit_host and init_work_dir as job attributes. These two values are now displayed with a qstat -f. The submit_host is the name of the host from where the job was submitted. init_work_dir is the working directory as in PBS_O_WORKDIR.
- e - change so blcr checkpoint jobs can restart on different node. Use configure --enable-blcr to allow.
- b - remove the use of a GNU specific function, and fix an error for solaris builds
- b - Updated PBS_License.txt to remove the implication that the software is not freely redistributable.
- b - remove the $PBS_GPUFILE when job is done on mom
- b - fix a race condition when issuing a qrerun followed by a qdel that caused the job to be queued instead of deleted sometimes.
- e - Implemented Bugzilla Bug 110. If a host in the nodes file cannot be resolved at startup the server will try once every 5 minutes until the node will resolve and it will add it to the nodes list.
- e - Added a "create" method to pbs_server init.d script so a serverdb file can be created if it does not exist at startup time. This is an enhancement in reference to Bugzilla bug 90.
- e - Add code to verify the group list as well when VALIDATEGROUPS is set in torque.cfg (backported from 3.0.1)
- b - Fix a bug where if geometry requests are enabled and cpusets are enabled, the cpuset wasn't deleted unless a geometry request was made. (backported from 3.0.1)
- b - Fix a race condition when starting pbs_mom with the -q option. exitstatus was getting overwritten and as a result jobs would not always be requeued to pbs_server but were being deleted instead. (backported from 3.0.1)
- e - Add a configure option --with-tcp-retry-limit to prevent potential 4+ hour hangs on pbs_server. We recommend --with-tcp-retry-limit=2 (backported from 3.0.1)
- b - preserve the order on array strings in TORQUE, like the route_destinations for a routing queue (backported from 3.0.1)
- b - fix bugzilla #111, multi-line environment variables causing errors in TORQUE. (backported from 3.0.1)
- b - allow apostrophes in Mail_Users attributes, as apostrophes are rare but legal email characters (backported from 3.0.1)
- b - Fixed a problem in parse_node_token where the local static variable pt would be advanced past the end of the line input if there is no newline character at the end of the nodes file.

2.5.4

- f - added the ability to track gpus. Users set gpus=X in the nodes file for relevant node, and then request gpus in the nodes request: -l nodes=X[:ppn=Y][:gpus=Z]. The gpus appear in $PBS_GPUFILE, a new environment variable, in the form: <hostname>-gpu<index> and in a new job attribute exec_gpus: <hostname>-gpu/<index>[+<hostname>-gpu/<index>...]
- b - clean up job MOM checkpoint directory on checkpoint failure
- e - Bugzilla bug 91. Check the status before the service is actually started. (Steve Traylen - CERN)
- e - Bugzilla bug 89. Only touch lock/subsys files if service actually starts. (Steve Traylen - CERN)
- c - when using job_force_cancel_time, fix a crash in rare cases
- e - add server parameter moab_array_compatible. When set to true, this parameter places a limit hold on jobs past the slot limit. Once one of the unheld jobs completes or is deleted, one of the held jobs is freed.
- b - fix a potential memory corruption for walltime remaining for jobs (Vikentsi Lapa)

- b - fix potential buffer overrun in pbs_sched (Bugzilla #98, patch from Stephen Usher @ University of Oxford)
- e - check if a process still exists before killing it and sleeping. This speeds up the time for killing a task exponentially, although this will show mostly for SMP/NUMA systems, but it will help everywhere. (Dr. Bernd Kallies)
- b - Fixed a problem where the -m n (request no mail on qsub) was not always being recongnized.
- b - Added patch for bug 101 by Martin Siegert. A null string was causing a segfault in pbs_server when record_jobinfo called into attr_to_string.
- b - Submitted patch from Vikentsi Lapa for bug 104. This patch adds the global variable pbsuser and sets it to the user id of the current user. This was needed for cygwin because the code had hard coded the value of 0 for root for seteuid. In the case of cygwin root cannot be used.
- b - Fix for reque failures on mom. Forked pbs_mom would silently segfault and job was left in Exiting state.
- b - prevent the nodes file from being overwritten when running make packages
- b - change so "mom_checkpoint_job_has_checkpoint" and "execing command" log messages do not always get logged

2.5.3

- b - stop reporting errors on success when modifying array ranges
- b - don't try to set the user id multiple times
- b - added some retrying to get connection and changed some log messages when doing a pbs_alterjob after a checkpoint
- c - fix segfault in tracejob. It wasn't malloc'ing space for the null terminator
- e - add the variables PBS_NUM_NODES and PBS_NUM_PPN to the job environment (TRQ-6)
- e - be able to append to the job's variable_list through the API (TRQ-5)
- e - Added support for munge authentication. This is an alternative for the default ruserok remote authentication and pbs_iff. This is a compile time option. The configure option to use is --enable-munge-auth. Ken Nielson (TRQ-7) September 15, 2010.
- b - fix the dependency hold for arrays. They were accidentally cleared before (RT 8593)
- e - add a logging statement if sendto fails at any points in rpp_send_out
- b - Applied patch submitted by Will Nolan to fix bug 76. "blocking read does not time out using signal handler"
- e - Added functionality that allows the values for the server parameter authorized_users to use wild cards for both the user and host portion.
- c - corrected a segfault when display_job_server_suffix is set to false and job_suffix_alias was unset.
- b - Bugzilla bug 84. Security bug on the way checkpoint is being handled. (Robin R. - Miami Univ. of Ohio)
- e - Now saving serverdb as an xml file instead of a byte-dump, thus allowing canned installations without qmgr scripts, as well as more portability. Able to upgrade automatically from 2.1, 2.3, and 2.4
- e - serverdb as xml is now optional, and it has to be configured with --enable-server-xml.

Each setting (normal and xml-enabled) can load the other format

- e - Created the ability to log all jobs to a file. The new file is located under $TORQUE_HOME/job_logs. The file follows the same naming format as server_logs and mom_logs. The name is derived from the current date. This log file is optional. It can be activated using a new server parameter record_job_info. By default this is false. If set to true it will begin recording every job record when the job is purged.
- b - fix to cleanup job files on MOM after a BLCR job is checkpointed and held
- b - make the tcp reading buffer able to grow dynamically to read larger values in order to avoid "invalid protocol" messages
- e - change so checkpoint files are transfered as the user, not as root.
- f - Added configure option --with-servchkptdir which allows specifying path for server's checkpoint files
- b - could not set the server HA parameters lock_file_update_time and lock_file_check_time previously. Fixed.
- e - Added new server parameter record_job_script. This works with record_job_info. These are both boolean values and default to false. record_job_info must be true in order for record_job_script to be enabled. If both values are enabled the entire content of the job script will be recorded to the job log file.
- e - qpeek now has the options --ssh, --rsh, --spool, --host, -o, and -e. Can now output both the STDOUT and STDERR files. Eliminated numlines, which didn't work.
- e - Added the server parameters job_log_file_max_size, job_log_file_roll_depth and job_log_keep_days to help manage job log files.
- b - fix to prevent a possible segfault when using checkpointing.

2.5.3

- e - Allow the nodes file to use the syntax node[0-100] in the name to create identical nodes with names node0, node1, ..., node100. (also node[000-100] => node000, node001, ... node100)
- b - fix support of the 'procs' functionality for qsub.
- b - remove square brackets [] from job and default stdout/stderr filenames for job arrays (fixes conflict with some non-bash shells)
- n - fix build system so README.array_changes is included in tar.gz file made with "make dist"
- n - fix build system so contrib/pbsweb-lite-0.95.tar.gz, contrib/qpool.gz and

contrib/README.pbstools are included the the tar.gz file made with "make dist"

- c - fixed crash when moving the job to a different queue (bugzilla 73)
- e - Modified buildutils/pbs_mkdirs.in to create server_priv/nodes file at install time. The file only shows examples and a link to the TORQUE documentation. This enhancement was first committed to trunk.
- c - fix pbs_server crash from invalid qsub -t argument
- b - fix so blcr checkpoint jobs work correctly when put on hold
- b - fixed bugzilla #75 where pbs_server would segfault with a double free when calling qalter on a running job or job array.
- e - Changed free_br back to its original form and modifed copy_batchrequest to make a copy of the rq_extend element which will be freed in free_br.
- b - fix condition where job array "template" may not get cleaned up properly after a server restart
- b - fix to get new pagg ID and add additional CSA records when restarting from checkpoint
- e - added documentation for pbs_alterjob_async(), pbs_checkpointjob(), pbs_fbserver(), pbs_get_server_list() and pbs_sigjobasync().
- b - Commited patch from Eygene Ryanbinkin to fix bug 61. /dev/null would under some circumstances have its permissions modified when jobs exited on a compute node.
- b - only clear the MOM state when actually running the health check script
- e - allow input of walltime in the format of [DD]:HH:MM:SS
- b - Fix so BLCR checkpoint files get copied to server on qchkpt and periodic checkpoints

2.5.1

- b - modified Makefile.in and Makefile.am at root to include contrib/AddPrivileges

2.5.0

- b - Updated URLs in README.torque file at root of build.
- b - Updated URLs in INSTALL file at root of build.
- e - Added new server config option alias_server_name. This option allows the MOM to add an additional server name to be added to the list of trusted addresses. The point of this is to be able to handle alias ip addresses. UDP requests that come into an aliased ip address are returned through the primary ip address in TORQUE. Because the address of the reply packet from the server is not the same address the MOM sent its HELLO1 request, the MOM drops the packet and the MOM cannot be added to the server.
- b - When the server parameter auto_node_np is set to true it is suppose to set the number of processors of a node to the value returned by the MOM in the ncpus value as returned in pbsnodes. If the configured processor value is less thanncpus the value is adjusted but if it is greater the value was not adjusted. This fix enables pbs_server to adjust processor values down as well as up.
- e - Changed qsub to allow for a -l nodes=x+procs=y syntax.
- b - Made a fix to qmgr.c in is_attr. When checking node names against attribute keywords is_attr used strncmp and limited the length of the compare to the length of the keyword. So node names like stateless were tagged as an error. (replaced strncmp with strcmp)
- e - Enabled TORQUE to be able to parse the -l procs=x node spec. Previously TORQUE simply recored the value of x for procs in Resources_List. It now takes that value and allocates x processors packed on any available node. (Ken Nielson Adaptive Computing. June 17, 2010)

f - added full support (server-scheduler-mom) for Cygwin (UIIP NAS of Belarus, uiip.bas-net.by)
- f - architecture and build system changes to support Cygwin (Igor Ilyenko, UIIP Minsk)
- b - fixed EINPROGRESS in net_client.c. This signal appears every time of connecting and requires individual processing. The old erroneous processing brought to big network delay, especially on Cygwin.
- e - improved signal processing after connecting in client_to_svr and added own implementation of bindresvport for OS which lack it (Igor Ilyenko, UIIP Minsk)
- f - created permission checking of Windows (Cygwin) users, using mkpasswd, mkgroup and own functions IamRoot, IamUser (Yauheni Charniauski, UIIP Minsk)
- f - created permission checking of submited jobs (Vikentsi Lapa, UIIP Minsk)
- f - Added the --disable-daemons configure option for start server-sched-mom as Windows services, cygrunsrv.exe goes its into background independently.
- e - Adapted output of Cygwin's diagnostic information (Yauheni Charniauski, UIIP Minsk)
- b - Changed pbsd_main to call daemonize_server early only if high_availability_mode is set.
- e - removed the very old A_ macros (patch provided by Simon Toth, CESNET z.s.p.o.)
- e - added new qmgr server attributes (clone_batch_size, clone_batch_delay) for controlling job cloning (Bugzilla #4)
- e - added new qmgr attribute (checkpoint_defaults) for setting default checkpoint values on Execution queues (Bugzilla #1)
- b - Merged revision 3268 from 2.4-fixes. removed block of code that broke pbs_statjob for requested attributes
- e - print a more informative error if pbs_iff isn't found when trying to authenticate a client
- n - 01/18/2010. Merged 2.4.5 revisions 3268-3375.
- e - added qmgr server attribute job_start_timeout, specifies timeout to be used for sending job to mom. If not set, tcp_timeout is used.
- e - added -DUSESAVEDRESOURCES code that uses servers saved resources used for accounting end record instead of current resources used for jobs that stopped running while MOM was not up.
- e - TORQUE job arrays now use arrays to hold the job pointers and not linked lists (allows constant lookup).
- f - Allow users to delete a range of jobs from the job array (qdel -t)
- f - Added a slot limit to the job arrays - this restricts the number of jobs that can concurrently run from one job array.
- f - added support for holding ranges of jobs from an array with a single qhold (using the -t option).
- f - now ranges of jobs in an array can be modified through qalter (using the -t option).
- f - jobs can now depend on arrays using these dependencies: afterstartarray, afterokarray, afternotokarray, afteranyarray
- f - added support for using qrls on arrays with the -t option
- e - complte overhaul of job array submission code
- f - by default show only a single entry in qstat output for the whole array (qstat -t expands the job array)
- f - server parameter max_job_array_size limits the number of jobs allowed in an array
- b - job arrays can no longer circumvent max_user_queuable
- b - job arrays can no longer circumvent max_queuable
- f - added server parameter max_slot_limit to restrict slot limits
- e - changed array names from jobid-index to jobid[index] for consistency
- n - TORQUE 2.5.0 released on 19-07-10

TORQUE 2.4

2.4.12

- b - Bugzilla bug 84. Security bug on the way checkpoint is being handled. (Robin R. - Miami Univ. of Ohio, back-ported from 2.5.3)
- b - make the tcp reading buffer able to grow dynamically to read larger values in order to avoid "invalid protocol" messages (backported from 2.5.3)
- b - could not set the server HA parameters lock_file_update_time and lock_file_check_time

previously. Fixed. (backported from 2.5.3)
- e - qpeek now has the options --ssh, --rsh, --spool, --host, -o, and -e. Can now output both the STDOUT and STDERR files. Eliminated numlines, which didn't work. (backported from 2.5.3)
- b - Modified the pbs_server startup routine to skip unknown hosts in the nodes file instead of terminating the server startup.
- b - fix to prevent a possible segfault when using checkpointing (back-ported from 2.5.3).
- b - fix to cleanup job files on MOM after a BLCR job is checkpointed and held (back-ported from 2.5.3)
- c - when using job_force_cancel_time, fix a crash in rare cases (backported from 2.5.4)
- b - fix a potential memory corruption for walltime remaining for jobs (Vikentsi Lapa, backported from 2.5.4)
- b - fix potential buffer overrun in pbs_sched (Bugzilla #98, patch from Stephen Usher @ University of Oxford, backported from 2.5.4)
- e - check if a process still exists before killing it and sleeping. This speeds up the time for killing a task exponentially, although this will show mostly for SMP/NUMA systems, but it will help everywhere. (backported from 2.5.4) (Dr. Bernd Kallies)
- e - Refactored torque spec file to comply with established RPM best practices, including the following:
  - Standard installation locations based on RPM macro configuration (e.g., %{_prefix})
  - Latest upstream RPM conditional build semantics with fallbacks for older versions of RPM (e.g., RHEL4)
  - Initial set of optional features (GUI, PAM, syslog, SCP) with more planned
  - Basic working configuration automatically generated at install-time
  - Reduce the number of unnecessary subpackages by consolidating where it makes sense and using existing RPM features (e.g., --excludedocs).
- b - Merged revision 4325 from 2.5-fixes. Fixed a problem where the -m n (request no mail on qsub) was not always being recongnized.
- b - Fix for reque failures on mom. Forked pbs_mom would silently segfault and job was left in Exiting state. (backported from 2.5.4)
- b - prevent the nodes file from being overwritten when running make packages

2.4.11

- b - changed type cast for calloc of ioenv from sizeof(char) to sizof(char *) in pbsdsh.c. This fixes bug 79.
- e - allow input of walltime in the format of [DD]:HH:MM:SS (backported from 2.5.2)
- b - only clear the MOM state when actually running the health check script (backported from 2.5.3)
- b - don't try to set the user id multiple times - (backported from 2.5.3)
- c - fix segfault in tracejob. It wasn't malloc'ing space for the null terminator (back-ported from 2.5.3)
- e - add the variables PBS_NUM_NODES and PBS_NUM_PPN to the job environment (backported from 2.5.3, TRQ-6)
- e - be able to append to the job's variable_list through the API (backported from 2.5.3, TRQ-5)
- b - Added patch to fix bug 76, "blocking read does not time out using signal handler.

2.4.10

- b - fix to get new pagg ID and add additional CSA records when restarting from checkpoint (backported from 2.5.2)
- e - added documentation for pbs_alterjob_async(), pbs_checkpointjob(), pbs_fbserver(),

pbs_get_server_list() and pbs_sigjobasync(). (backported from 2.5.2)

- b - fix for bug 61. The fix takes care of a problem where pbs_mom under some situations will change the mode and permissions of /dev/null.

2.4.9

- b - Backed out enhancement for preempted jobs. This was SVN revision 3784. This patch cased qrun to hang and not return when executing jobs.
- e - Commited changes that changed how preempted jobs are killed. This change uses a SIGTERM followed by a kill_delay SIGKILL to give preemted jobs time to checkpoint before terminating.
- e - Patch to correctly log attempts to create a cpuset as debug messages. The function im_request() in src/resmom/mom_comm.c logs the message:
  pbs_mom: LOG_ERROR::im_request, about to create cpuset for job 55100.blah
  as an error rather than a debug message (as used in src/resmom/start_exec.c).
  The fix is to replace:
  log_err(-1, id, log_buffer);
  with:
  log_ext(-1, id, log_buffer, LOG_INFO);
- b - Modified fix in qmgr.c in is_attr to check for the '.' character on resource attributes such as resources_available.nodect. The attribute is striped of the '.' and the element and just the attribute name is used to compare for a valid attribute.
- b - Made a fix to qmgr.c in is_attr. When checking node names against attribute keywords is_attr used strncmp and limited the length of the compare to the length of the keyword. So node names like stateless were tagged as an error. I replaced strncmp with strcmp. This fix was added to trunk first. Version 2.5.0
- b - Bugzilla bug 57. Check return value of malloc for tracejob for Linux (Chris Samuel - Univ. of Melbourne)
- b - fix so "gres" config gets displayed by pbsnodes
- b - use QSUBHOST as the default host for output files when no host is specified. (RT 7678)
- e - allow users to use cpusets and geometry requests at the same time by specifying both at configure time.
- b - Bugzilla bug 55. Check return value of malloc for pbs_mom for Linux (Chris Samuel - Univ. of Melbourne)
- e - added server parameter job_force_cancel_time. When configured to X seconds, a job that is still there X seconds after a qdel will be purged. Useful for freeing nodes from a job when one node goes down midjob.
- b - fixed gcc warnings reported by Skip Montanaro
- e - added RPT_BAVAIL define that allows pbs_mom to report f_bavail instead of f_bfree on Linux systems
- b - no longer consider -t and -T the same in qsub
- e - make PBS_O_WORKDIR accessible in the environment for prolog scripts
- e - Bugzilla 59. Applied patch to allow '=' for qdel -m. (Chris Samuel - Univ. of Melbourne)
- b - properly escape characters (&"'<>) in XML output)
- b - ignore port when checking host in svr_get_privilege()
- b - restore ability to parse -W x=geometry:{...,...}
- e - from Simon Toth: If no available amount is specified for a resource and the max limit is set, the requirement should be checked against the maximum only (for scheduler, bugzilla 23).
- b - check return values from fwrite in cpuset.c to avoid warnings
- e - expand acl host checking to allow * in the middle of hostnames, not just at the beginning. Also allow ranges like a[10-15] to mean a10, a11, ..., a15.

2.4.8

- e - Bugzilla bug 22. HIGH_PRECISION_FAIRSHARE for fifo scheduling.

- c - no longer sigabrt with "running" jobs not in an execution queue. log an error.
- b - fixed kill_delay. In order to fix and not change behavior, added the parameter $kill_delay to mom's config. Activate by setting to true
- b - commented out a 'fi' left uncommented in contrib/init/pbs_server
- e - mapped 'qsub -P user:group' to qsub -P user -W group_list=group
- e - added -DQSUBHOSTNAME to allow qsub to determine PBS_O_HOST
- b - fixed segfault for when TORQUE thinks there's a nanny but there isn't
- b - reverted to old behavior where interactive scripts are checked for directives and not run without a parameter.
- e - setting a queue's resource_max.nodes now actually restricts things, although so far it only limits based on the number of nodes (i.e. not ppn)
- f - added QSUBSENDGROUPLIST to qsub. This allows the server to know the correct group name when disable_server_id_check is set to true and the user doesn't exist on the server.
- e - Bugzilla bug 54. Patch submitted by Bas van der Vlies to make pbs_mkdirs more robust, provide a help function and new option -C <chk_tree_location>
- n - TORQUE 2.4.8 released on 29-04-10

2.4.7

- b - fixed a bug for when a resource_list has been set, but isn't completely initialized, causing a segfault
- b - stop counting down walltime remaining after a job is completed
- b - correctly display the number for tasks as used in TORQUE in qstat -a output
- b - no longer ignoring fread return values in Linux cpuset code (gcc 4.3.3)
- b - fixed a bug where job was added to obit retry list multiple times, causing a segfault
- b - Fix for Bugzilla bug 43. "configure ignores with-modulefiles=no"
- b - no longer try to decide when to start with -t create in init.d scripts, -t creates should be done manually by the user
- b - no longer let qsub determine the PBS_O_HOST. This work is done on the server and the server code accounts for the connection interface as well as aliasing. Code to set PBS_O_HOST on the server is already there, but now will be active.
- f - added -P to qsub. When submitting a job as root, the root user may add -P <username> to submit the job as the proxy user specified by <usermname>
- n - 2.4.7 released on 29-03-10

2.4.6

- f - added an asynchronous option for qsig, specified with -a.
- b - fix to cleanup job that is left in running state after a MOM restart
- e - qsub's -W can now parse attributes with quoted lists, for example: qsub script -W attr="foo,foo1,foo2,foo3" will set foo,foo1,foo2,foo3 as attr's value.
- e - qstat -f now includes an extra field "Walltime Remaining" that tells the remaining walltime in seconds. This field is does not account for weighted walltime.
- b - fixed erroneous display of walltime when start time hasn't been set
- b - fixed possible segfault when finding remaining walltime (if the job's resources haven't been defined)
- e - altered the display of walltime remaining so that the xml produced by qstat -f stays consistent. Also updated the man page.
- b - split Cray job library and CSA functionality since CSA is dependant on job library but job library is not dependant on CSA
- f - added two server parameters: display_job_server_suffix and job_suffix_alias. The first defaults to true and is whether or not jobs should be appended by .server_name. The second defaults to NULL, but if it is defined it will be appended at the end of the jobid, i.e. jobid.job_suffix_alias.
- f - added -l option to qstat so that it will display a server name and an alias if both are used. If these aren't used, -l has no effect.
- b - fixed an off by one error for a case in get_correct_jobname

- e - altered the display_job_server_suffix parameter and the job_suffix_alias parameter so that they don't interfere with FQDN.
- b - fixed open_std_file to setegid as well, this caused a problem with epilogue.user scripts.
- n - 2.4.6 officially released on 02/24/2010

2.4.5

- b - epilogue.user scripts were being run with prologue argments. Fixed bug in run_pelog() to include PE_EPILOGUSER so epilogue arguments get passed to eplilogue.user script.
- b - Ticket 6665. pbs_mom and job recovery. Fixed a bug where the -q option would terminate running processes as well as requeue jobs. This made the -q option the same as the -r option for pbs_mom. -q will now only reque jobs and will not attempt to kill running processes. I also added a -P option to start pbs_mom. This is similar to the -p option except the -P option will only delete any left over jobs from the queue and will not attempt to adopt and running processes.
- e - Modified man page for pbs_mom. Added new -P option plus edited -p, -q and -r options to hopefully make them more understandable.
- n - 01/15/2010 created snapshot torque-2.4.5-snap201001151416.tar.gz.
- b - now checks secondary groups (as well as primary) for creating a file when spooling. Before it wouldn't create the spool file if a user had permission through a secondary group.
- b - fixed a file descriptor error with high availability. Before it was possible to try to regain a file descriptor which was never held, now this is fixed.
- e - Added the function prepare_child_tasks_for_delete() to src/resmom/solaris5 /mom_mach.c. This is required for new -P functionality.
- b - updated to a newer gcc and fixed warnings related to disregarded return values. Logging was added.
- b - Modified code specific to Solaris5 platform. Changes were made so TORQUE would successfully compile on sun system. Tcl still does not successfully compile. configure needs to be done with the --disable-gui option. 01/21/2010.
- e - Moved the function prepare_child_tasks_for_delete() from the mom_mach.c files for Linux and Solaris5. This routine is not platform dependent.
  No other platform had the function yet. 01/22/2010
- e - Commited changes that will allow TORQUE to compile on the Solaris5 platform with gcc-warnings enabled.
- b - No longer overwrites the user's environment when spoolasfinalname is set. Now the environment is handled correctly.
- b - No longer will segfault if pbs_mom restarts in a bad state (user environment not initialized)
- e - added qmgr server attribute job_start_timeout, specifies timeout to be used for sending

job to mom. If not set, tcp_timeout is used.

- e - added -DUSESAVEDRESOURCES code that uses servers saved resources used for accounting end record instead of current resources used for jobs that stopped running while MOM was not up.
- e - Changing MAXNOTDEFAULT behavior. Now, by default, max is not default and max can be configured as default with --enable-maxdefault.
- n - TORQUE 2.4.5 released on 02/02/10 (Groundhog day!)

2.4.4

- b - fixed contrib/init.d/pbs_mom so that it doesn't overwrite $args defined in /etc/sysconfig/pbs_mom
- b - when spool_as_final_name is configured for the mom, no longer send email messages about not being able to copy the spool file
- b - when spool_as_final_name is configured for the mom, correctly substitue job environment variables
- f - added logging for email events, allows the admin to check if emails are being sent correctly
- b - Made a fix to svr_get_privilege(). On some architectures a non-root user name would be set to null after the line " host_no_port[num_host_chars] = 0;" because num_host_chars was = 1024 which was the size of hot_no_port. The null termination needed to happen at 1023. There were other problems with this function so code was added to validate the incoming variables before they were used. The symptom of this bug was that non-root managers and operators could not perform operations where they should have had rights.
- b - Missed a format statement in an sprintf statement for the bug fix above.
- b - Fixed a way that a file descriptor (for the server lockfile) could be used without initialization. RT 6756

2.4.3

- b - fix PBSD_authenticate so it correctly splits PATH with : instead of ; (bugzilla #33)
- e - Refactored tcp_dis function calls. With the removal of the global variable dis_buffer the seperation of dis calls was no longer needed. the tcp_dis function calls have been removed and all calls go to the dis functions whether using tcp or rpp.
- b - pbs_mom now sets resource limits for tasks started with tm_spawn (Chris Samuel, VPAC)
- c - fix assumption about size of unsocname.sun_path in Libnet/net_server.c
- b - Fix for Bugzilla bug 34. "torque 2.4.X breaks OSC's mpiexec". fix in src/server src/server/stat_job.c revision 3268.
- b - Fix for Bugzilla bug 35 - printing the wrong pid (normal mode) and not printing any pid for high availability mode.
- f - added a diagnostic script (contrib/diag/tdiag.sh). This script grabs the log files for the server and the mom, records the output of qmgr -c 'p s' and the nodefile, and creates a tarfile containing these.
- b - Changed momctl -s to use exit(EXIT_FAILURE) instead of return(-1) if a mom is not running.
- b - Fix for Bugzilla bug 36. "qsub crashes with long dependency list".
- b - Fix for Bugzilla bug 41. "tracejob creates a file in the local directory".

2.4.2

- b - Changed predicate in pbsd_main.c for the two locations where daemonize_server is called to check for the value of high_availability_mode to determine when to put the server process in the background.
- b - Added pbs_error_db.h to src/include/Makefile.am and src/include/Makefile.in. pbs_error_db.h now needed for install.
- e - Modified pbs_get_server_list so the $TORQUE_HOME/server_name file will work with a comma delimited string or a list of server names separated by a new line.
- b - fix tracejob so it handles multiple server and MOM logs for the same day
- f - Added a new server parameter np_default. This allows the administrator to change the number of processors to a unified value dynamically for the entire cluster.
- e - high availability enhanced so that the server spawns a separate thread to update the "lock" on the lockfile. Thread update and check time are both setable parameters in qmgr.
- b - close empty ACL files

2.4.1

- e - added a prologue and epilogue option to the list of resources for qsub -l which allows a per job prologue or epilogue script. The syntax for the new option is qsub -l prologue=<prologue script>, epilogue=<epilogue script>
- f - added a "-w" option to qsub to override the working directory
- e - changes needed to allow relocatable checkpoint jobs. Job checkpoint files are now under the control of the server.
- c - check filename for NULL to prevent crash
- b - changed so we don't try to copy a local file when the destination is a directory and the file is already in that directory
- f - changes to allow TORQUE to operate without pbs_iff (merged from 2.3)
- e - made logging functions rentrant safe by using localtime_r instead of localtime() (merged from 2.3)
- e - Merged in more logging and NOSIGCHLDMOM capability from Yahoo branch
- e - merged in new log_ext() function to allow more fine grained syslog events, you can now specify severity level. Also added more logging statements
- b - fixed a bug where CPU time was not being added up properly in all cases (fix for Linux only)
- c - fixed a few memory errors due to some uninitialized memory being allocated (ported from 2.3 R2493)
- e - added code to allow compilers to override CLONE_BATCH_SIZE at configure time (allows

for finer grained control on how arrays are created) (ported from Yahoo R2461)
- e - added code which prefixes the severity tag on all log_ext() and log_err() messages (ported from Yahoo R2358)
- f - added code from 2.3-extreme that allows TORQUE to handle more than 1024 sockets. Also, increased the size of TORQUE's internal socket handle table to avoid running out of handles under busy conditions.
- e - TORQUE can now handle server names larger than 64 bytes (now set to 1024, which should be larger than the max for hostnames)
- e - added qmgr option accounting_keep_days, specifies how long to keep accounting files.
- e - changed MOM config varattr so invoked script returns the varattr name and value(s)
- e - improved the performance of pbs_server when submitting large numbers of jobs with dependencies defined
- e - added new parameter "log_keep_days" to both pbs_server and pbs_mom. Specifies how long to keep log files before they are automatically removed
- e - added qmgr server attribute lock_file, specifies where server lock file is located
- b - change so we use default file name for output / error file when just a directory is specified on qsub / qalter -e -o options
- e - modified to allow retention of completed jobs across server shutdown
- e - added job_must_report qmgr configuration which says the job must be reported to scheduler. Added job attribute "reported". Added PURGECOMP functionality which allows scheduler to confirm jobs are reported. Also added -c option to qdel. Used to clean up unreported jobs.
- b - Fix so interactive jobs run when using $job_output_file_umask userdefault
- f - Allow adding extra End accounting record for a running job that is rerun. Provides usage data. Enabled by CFLAGS=-DRERUNUSAGE.
- b - Fix to use queue/server resources_defaults to validate mppnodect against resources_max when mppwidth or mppnppn are not specified for job
- f - merged in new dynamic array struct and functions to implement a new (and more efficient) way of loading jobs at startup--should help by 2 orders of magnitude!
- f - changed TORQUE_MAXCONNECTTIMEOUT to be a global variable that is now changed by the MOM to be smaller than the pbs_server and is also configurable on the MOM ($max_conn_timeout_micro_sec)
- e - change so queued jobs that get deleted go to complete and get displayed in qstat based on keep_completed
- b - Changes to improve the qstat -x XML output and documentation
- b - Change so BATCH_PARTITION_ID does not pass through to child jobs
- c - fix to prevent segfault on pbs_server -t cold
- b - fix so find_resc_entry still works after setting server extra_resc
- c - keep pbs_server from trying to free empty attrlist after recieving bad request (Michael Meier, University of Erlangen-Nurnberg) (merged from 2.3.8)
- f - new fifo scheduler config option. ignore_queue: queue_name allows the scheduler to be instructed to ignore up to 16 queues on the server (Simon Toth, CESNET z.s.p.o.)
- e - add administrator customizable email notifications (see manpage for pbs_server_attributes) - (Roland Haas, Georgia Tech)
- e - moving jobs can now trigger a scheduling iteration (merged from 2.3.8)
- e - created a utility module that is shared between both server and MOM but does NOT get placed in the libtorque library
- e - allow the user to request a specific processor geometry for their job using a bitmap, and then bind their jobs to those processors using cpusets.
- b - fix how qsub sets PBS_O_HOST and PBS_SERVER (Eirikur Hjartarson, deCODE genetics) (merged from 2.3.8)
- b - fix to prevent some jobs from getting deleted on startup.
- f - add qpool.gz to contrib directory
- e - improve how error constants and text messages are represented (Simon Toth, CESNET z.s.p.o)
- f - new boolean queue attribute "is_transit" that allows jobs to exceede server resource limits (queue limits are respected). This allows routing queues to route jobs that would be rejected for exceeding local resources even when the job won't be run locally. (Simon Toth, CESNET z.s.p.o)

- e - add support for "job_array" as a type for queue disallowed_types attribute
- e - added pbs_mom config option ignmem to ignore mem/pmem limit enforcement
- e - added pbs_mom config option igncput to ignore pcput limit enforcement

2.4.0

- f - added a "-q" option to pbs_mom which does *not* perform the default -p behavior
- e - made "pbs_mom -p" the default option when starting pbs_mom
- e - added -q to qalter to allow quicker response to modify requests
- f - added basic qhold support for job arrays
- b - clear out ji_destin in obit_reply
- f - add qchkpt command
- e - renamed job.h to pbs_job.h
- b - fix logic error in checkpoint interval test
- f - add RERUNNABLEBYDEFAULT parameter to torque.cfg. allows admin to change the default value of the job rerunnable attribute from true to false
- e - added preliminary Comprehensive System Accounting (CSA) functionality for Linux. Configure option --enable-csa will cause workload management records to be written if CSA is installed and wkmg is turned on.
- b - changes to allow post_checkpoint() to run when checkpoint is completed, not when it has just started. Also corrected issue when checkpoint fails while trying to put job on hold.
- b - update server immediately with changed checkpoint name and time attributes after successful checkpoint.
- e - Changes so checkpoint jobs failing after restarted are put on hold or requeued
- e - Added checkpoint_restart_status job attribute used for restart status
- b - Updated manpages for qsub and qterm to reflect changed checkpointing options.
- b - reject a qchkpt request if checkpointing is not enabled for the job
- b - Mom should not send checkpoint name and time to server unless checkpoint was successful
- b - fix so that running jobs that have a hold type and that fail on checkpoint restart get deleted when qdel is used
- b - fix so we reset start_time, if needed, when restarting a checkpointed job
- f - added experimental fault_tolerant job attribute (set to true by passing -f to qsub) this attribute indicates that a job can survive the loss of a sister MOM also added corresponding fault_tolerant and fault_intolerant types to the "disallowed_types" queue attribute
- b - fixes for pbs_moms updating of comment and checkpoint name and time
- e - change so we can reject hold requests on running jobs that do not have checkpoint enabled if system was configured with --enable-blcr
- e - change to qsub so only the host name can be specified on the -e/-o options
- e - added -w option to qsub that allows setting of PBS_O_WORKDIR

TORQUE 2.3

2.3.12 - This is the last offical release of TORQUE 2.3.

- b - Applied patch submitted for bug 61. pbs_mom changing /dev/null mode and perms

2.3.11

- b - no longer ignoring fread return values in Linux cpuset code (gcc 4.3.3)
- b - fixed segfault for when TORQUE thinks there's a nanny but there isn't
- b - Bugzilla bug 57. Check return value of malloc for tracejob for Linux (Chris Samuel - Univ. of Melbourne)
- b - fix so "gres" config gets displayed by pbsnodes
- b - Bugzilla bug 55. Check return value of malloc for pbs_mom for Linux (Chris Samuel - Univ. of Melbourne)
- b - no longer consider -t and -T the same in qsub
- c - very rare read of a potentially NULL pointer

- b - properly escape characters (&"'<>) in XML output) b - ignore port when checking host in svr_get_privilege()

2.3.10

- b - Fixed a bug in run_pelog (src/resmom/prolog.c) where epilogue.user was given the argument list for prologue scripts and not epilogue scripts. Ticket 6296.
- b - Fixed pbs_mom's default restart behavior. On a restart the MOM is suppose to terminate jobs that were in a running state while the MOM was up and report them to the batch server where the job will be reset to a queued state. But it should not try and kill any of the running processes that were associated with the job. Prior to this fix the MOM would try and kill running processes associated with any running jobs.
- n - 01/15/2010 snapshot torque-2.3.10-snap.201001151340.tar.gz created.
- b - Made changes to source files and configure.ac to enable TORQUE to compile on Solaris5 platform with gcc-warnings enabled. Currently TORQUE must be compiled with the --disable-gui option because X11 support on Solaris is not working with the current TORQUE build scripts.
- e - added qmgr server attribute job_start_timeout, specifies timeout to be used for sending job to mom. If not set, tcp_timeout is used.
- n - 2.3.10 released on 02/02/10 (Groundhog Day!

2.3.9

- b - Made a fix to svr_get_privilege(). On some architectures a non-root user name would be set to null after the line " host_no_port[num_host_chars] = 0;" because num_host_chars was = 1024 which was the size of hot_no_port. The null termination needed to happen at 1023. There were other problems with this function so code was added to validate the incoming variables before they were used. The symptom of this bug was that non-root managers and operators could not perform operations where they should have had rights.

2.3.8

- c - keep pbs_server from trying to free empty attrlist after recieving bad request (Michael Meier, University of Erlangen-Nurnberg)
- e - moving jobs can now trigger a scheduling iteration
- b - fix how qsub sets PBS_O_HOST and PBS_SERVER (Eirikur Hjartarson, deCODE genetics)
- f - add qpool.gz to contrib directory
- b - fix return value of cpuset_delete() for Linux (Chris Samuel - VPAC)
- e - Set PBS_MAXUSER to 32 from 16 in order to accomodate systems that use a 32 bit user name.(Ken Nielson Cluster Resources)
- c - modified acct_job in server/accounting.c to dynamically allocate memory
- to accomodate strings larger than PBS_ACCT_MAX_RCD. (Ken Nielson Cluster Resources)
- e - all the user to turn off credential lifetimes so they don't have to lose iterations while credentials are renewed.
- e - added OS independent resending of failed job obits (from D Beer), also removed OS specific CACHEOBITFAILURES code.
- b - fix so after* dependencies are handled correctly for exiting / completed jobs

2.3.7

- b - fixed a bug where Unix domain socket communication was failing when "--disable-privports" was used.
- e - add job exit status as 10th argument to the epilogue script
- b - fix truncated output in qmgr (peter h IPSec+jan n NANCO)
- b - change so set_jobexid() gets called if JOB_ATR_egroup is not set
- e - pbs_mom sisters can now tolerate an explicit group ID instead of only a
- valid group name. This helps TORQUE be more robust to group lookup failures.

2.3.6

- e - in Linux, a pbs_mom will now "kill" a job's task, even if that task can no longer be found in the OS processor table. This prevents jobs from getting "stuck" when the PID vanishes in some rare cases.
- e - forward-ported change from 2.1-fixes (r2581) (b - reissue job obit even if no processes are found)
- b - change back to not sending status updates until we get cluster addr message from server, also only try to send hello when the server stream is down.
- b - change pbs_server so log_file_max_size of zero behavior matches documentation
- e - added periodic logging of version and loglevel to help in support
- e - added pbs_mom config option ignvmem to ignore vmem/pvmem limit enforcement
- b - change to correct strtoks that accidentally got changed in astyle formatting

2.3.5

- e - added new init.d scripts for Debian/Ubuntu systems
- b - fixed regression in 2.3.4 release which incorrectly changed soname for libtorque
- b - fixed a bug where TORQUE's exponential backoff for sending messages to the MOM could overflow

2.3.4

- b - fixed a bug with RPM spec files due to new pbs_track executable
- b - fixed a bug with "max_report" where jobs not in the Q state were not always being reported to scheduler
- b - fixed bug with new Unix socket communication when more than one TORQUE instance is running on the same host
- c - fixed a few memory errors due to a spurious comma and some uninitialized memory being allocated
- b - fixed a bug preventing multiple TORQUE servers and TORQUE MOMs from operating properly all from the same host
- f - enabled 'qsub -T' to specify "job type." Currently this will allow a per job prolog/epilog
- f - added a new '-E' option to qstat which allows command-line users to pass "extend" strings via the API
- f - added new max_report queue attribute which will limit the number of Idle jobs, per queue, that TORQUE reports to the scheduler
- e - enhanced logging when a hostname cannot be looked up in DNS
- e - PBS_NET_MAX_CONNECTIONS can now be defined at compile time (via CFLAGS)
- e - modified source code so that all .c and .h files now conform more closely to the new CRI format style
- c - fixed segfault when loading job files of an older/incompatible version
- b - fixed a bug where if attempt to send job to a pbs_mom failed due to timeout, the job would indefinitely remain the in 'R' state
- b - fixed a bug where CPU time was not being added up properly in all cases (fix for Linux only)
- e - pbs_track now allows passing of - and -- options to the a.out argument
- b - qsub now properly interprets -W umask=0XXX as octal umask
- e - allow $HOME to be specified for path

- e - added --disable-qsub-keep-override to allow the qsub -k flag to not override -o -e.
- e - updated with security patches for setuid, setgid, setgroups
- b - fixed correct_ct() in svr_jobfunc.c so we don't crash if we hit COMPLETED job
- b - fixed problem where momctl -d 0 showed ConfigVersion twice
- e - if a .JB file gets upgraded pbs_server will back up the original
- b - removed qhold / qrls -h n option since there is no code to support it
- b - set job state and substate correctly when job has a hold attribute and is being rerun
- e - fixed several compiler error and warnings for AIX 5.2 systems

2.3.3

- b - fixed bug where pbs_mom would sometimes not connect properly with pbs_server after network failures
- b - changed so run_pelog opens correct stdout/stderr when join is used
- b - corrected pbs_server man page for SIGUSR1 and SIGUSR2
- f - added new pbs_track command which may be used to launch an external process and a pbs_mom will then track the resource usage of that process and attach it to a specified job (experimental) (special thanks to David Singleton and David Houlder from APAC)
- e - added alternate method for sending cluster addresses to MOM (ALT_CLSTR_ADDR)

2.3.2

- e - added --disable-posixmemlock to force MOM not to use POSIX MEMLOCK.
- b - fix potential buffer overrun in qsub
- b - keep pbs_mom, pbs_server, pbs_sched from closing sockets opened by nss_ldap (SGI)
- e - added PBS_VERSION environment variable
- e - added --enable-acct-x to allow adding of x attributes to accounting log
- b - fix net_server.h build error
- b - fixed code that was causing jobs to fail due to "neednodes" errors when Moab/Maui was the scheduler

2.3.1

- b - fixed a bug where torque would fail to start if there was no LF in nodes file
- b - fixed a bug where TORQUE would ignore the "pbs_asyrunjob" API extension string when starting jobs in asynchronous mode
- b - fixed memory leak in free_br for PBS_BATCH_MvJobFile case
- e - torque can now compile on Linux and OS X with NDEBUG defined
- f - when using qsub it is now possible to specify both -k and -o/-e (before -o/-e did not behave as expected if -k was also used)
- e - changed pbs_server to have "-l" option. Specifies a host/port that event messages will be sent to. Event messages are the same as what the scheduler currently receives.
- e - added --enable-autorun to allow qsub jobs to automatically try to run if there are any nodes available.
- e - added --enable-quickcommit to allow qsub to combine the ready to commit and commit phases into 1 network transmission.
- e - added --enable-nochildsignal to allow pbs_server to use inline checking for SIGCHLD instead of using the signal handler.
- e - change qsub so '-v var=' will look in environment for value. If value is not found set it to "".
- b - fixed mom_server code's HELLO initiation retry control to reduce occurrence of pbs_server incorrectly marking node as unknown/down
- b - fix qdel of entire job arrays for non operator/managers
- b - fix so we continue to process exiting jobs for other servers
- e - added source_login_batch and source_login_interactive to MOM config. This allows us to bypass the sourcing of /etc/profile, etc. type files.
- b - fixed pbs_server segmentation fault when job_array submissions are rejected before ji_arraystruct was initialized
- e - add some casts to fix some compiler warnings with gcc-4.1 on i386 when -D_FILE_OFFSET_BITS=64 is set
- e - added --enable-maxnotdefault to allow not using resources_max as defaults.
- b - fixed file descriptor leak with Linux cpusets (VPAC)
- b - added new values to TJobAttr so we don't have mismatch with job.h values. Added some comments also.
- b - reset ji_momhandle so we cannot have more than one pjob for obit_reply to find.
- e - change qdel to accept 'ALL' as well as 'all'
- b - changed order of searching so we find most recent jobs first. Prevents finding old leftover job when pids rollover. Also some CACHEOBITFAILURES updates.
- b - handle case where MOM replies with an unknown job error to a stat request from the server
- b - allow qalter to modify HELD jobs if BLCR is not enabled
- b - change to update errpath/outpath attributes when -e -o are used with qsub
- e - added string output for errnos, etc.

2.3.0

- b - fixed a bug where TORQUE would ignore the "pbs_asyrunjob" API extension string when starting jobs in asynchronous mode
- e - redesign how torque.spec is built
- e - added -a to qrun to allow asynchronous job start
- e - allow qrerun on completed jobs
- e - allow qdel to delete all jobs
- e - make qdel -m functionality match the documentation
- b - prevent runaway hellos being sent to server when mom's node is removed from the server's node list
- e - local client connections use a Unix domain socket, bypassing inet and pbs_iff
- f - Linux 2.6 cpuset support (in development)
- e - new job array submission syntax
- b - fixed SIGUSR1 / SIGUSR2 to correctly change the log level

f - health check script can now be run at job start and end
- e - tm tasks are now stored in a single .TK file rather than eat lots of inodes
- f - new "extra_resc" server attribute
- b - "pbs_version" attr is now correctly read-only
- e - increase max size of .JB and .SC file names
- e - new "sched_version" server attribute
- f - new printserverdb tool
- e - pbs_server/pbs_mom hostname arg is now -H, -h is help
- e - added $umask to pbs_mom config, used for generated output files.
- e - minor pbsnodes overhaul
- b - fixed memory leak in pbs_server

TORQUE 2.2

2.2.0

- e - improve RPP logging for corruption issues
- f - dynamic resources
- b - correct run-time symbol in pam module on RHEL4
- f - allow manager to set "next job number" vi hidden qmgr attribute next_job_number
- b - some minor hpux11 build fixes (PACCAR)
- e - allow pam_pbssimpleauth to be built on OSX and Solaris
- b - fix bug with log roll and automatic log filenames
- e - use mlockall() in pbs_mom if _POSIX_MEMLOCK
- f - consumable resource "tokens" support (Harte-Hanks)
- b - networking fixes for HPUX, fixes pbs_iff (PACCAR)
- e - fix "list_head" symbol clash on Solaris 10
- f - Linux 2.6 cpuset support
- b - compile error with size_fs() on digitalunix
- e - build process sets default submit filter path to ${libexecdir}/qsub_filter
  - - we fall back to /usr/local/sbin/torque_submitfilter to maintain compatibility
- e - allow long job names when not using -N
- e - pbs_server will now print build details with --about

TORQUE 2.1

2.1.2

- b - fix momctl queries with multiple hosts
- b - don't fail make install if --without-sched
- b - correct MOM compile error with atol()
- f - qsub will now retry connecting to pbs_server (see manpage)
- f - X11 forwarding for single-node, interactive jobs with qsub -X

- f - new pam_pbssimpleauth PAM module, requires --with-pam=DIR
- e - add logging for node state adjustment
- f - correctly track node state and allocation based for suspended jobs
- e - entries can always be deleted from manager ACL, even if ACL contains host(s) that no longer exist
- e - more informative error message when modifying manager ACL
- f - all queue create, set, and unset operations now set a queue mtime
- f - added support for log rolling to libtorque
- f - pbs_server and pbs_mom have two new attributes log_file_max_size, log_file_roll_depth
- e - support installing client libs and cmds on unsupported OSes (like cygwin)
- b - fix subnode allocation with pbs_sched
- b - fix node allocation with suspend-resume
- b - fix stale job-exclusive state when restarting pbs_server
- b - don't fall over when duplicate subnodes are assigned after suspend-resume
- b - handle suspended jobs correctly when restarting pbs_server
- b - allow long host lists in runjob request
- b - fix truncated XML output in qstat and pbsnodes
- b - typo broke compile on irix6array and unicos8
- e - momctl now skips down nodes when selecting by property
- f - added submit_args job attribute

2.1.1

- c - fix mom_sync_job code that crashes pbs_server (USC)
- b - checking disk space in $PBS_SERVER_HOME was mistakenly disabled (USC)
- e - node's np now accessible in qmgr (USC)
- f - add ":ALL" as a special node selection when stat'ing nodes (USC)
- f - momctl can now use :property node selection (USC)
- f - send cluster addrs to all nodes when a node is created in qmgr (USC)
  - - new nodes are marked offline
  - - all nodes get new cluster ipaddr list
  - - new nodes are cleared of offline bit
- f - set a node's np from the status' ncpus (only if ncpus > np) (USC)
  - - controlled by new server attribute "auto_node_np"
- c - fix possible pbs_server crash when nodes are deleted in qmgr (USC)
- e - avoid dup streams with nodes for quicker pbs_server startup (USC)
- b - configure program prefix/suffix will now work correctly (USC)
- b - handle shared libs in tpackages (USC)
- f - qstat's -1 option can now be used with -f for easier parsing (USC)
- b - fix broken TM on OSX (USC)
- f - add "version" and "configversion" RM requests (USC)
- b - in pbs-config --libs, don't print rpath if libdir is in the sys dlsearch path (USC)
- e - don't reject job submits if nodes are temporarily down (USC)
- e - if MOM can't resolve $pbsserver at startup, try again later (USC)
  - - $pbsclient still suffers this problem
- c - fix nd_addrs usage in bad_node_warning() after deleting nodes (MSIC)
- b - enable build of xpbsmom on darwin systems (JAX)
- e - run-time config of MOM's rcp cmd (see pbs_mom(8)) (USC)
- e - momctl can now accept query strings with spaces, multiple -q opts (USC)
- b - fix linking order for single-pass linkers like IRIX (ncifcrf)
- b - fix MOM compile on solaris with statfs (USC)

- b - memory corruption on job exit causing cpu0 to be allocated more than once (USC)
- e - add increased verbosity to tracejob and added '-q' commandline option
- e - support larger values in qstat output (might break scripts!) (USC)
- e - make qterm server shutdown faster (USC)

2.1.0p0

- fixed job tracking with SMP job suspend/resume (MSIC)
- modify pbs_mom to enforce memory limits for serial jobs (GaTech)
- - Linux only
- enable 'never' qmgr maildomain value to disable user mail
- enable qsub reporting of job rejection reason
- add suspend/resume diagnostics and logging
- prevent stale job handler from destroying suspended jobs
- prevent rapid hello from MOM from doing DOS on pbs_server
- add diagnostics for why node not considered available
- add caching of local serverhost addr lookup
- enable job centric vs queue centric queue limit parameter
- brand new autoconf+automake+libtool build system (USC)
- automatic MOM restarts for easier upgrades (USC)
- new server attributes: acl_group_sloppy, acl_logic_or, keep_completed, kill_delay
- new server attributes: server_name, allow_node_submit, submit_hosts
- torque.cfg no longer used by pbs_server
- pbsdsh and TM enhancements (USC)
- - tm_spawn() returns an error if execution fails
- - capture TM stdout with -o
- - run on unique nodes with -u
- - run on a given hostname with -h

- largefile support in staging code and when removing $TMPDIR (USC)
- use bindresvport() instead of looping over calls to bind() (USC)
- fix qsub "out of memory" for large resource requests (SANDIA)
- pbsnodes default arg is now '-a' (USC)
- new ":property" node selection when node stat and manager set (pbsnodes) (USC)
- fix race with new jobs reporting wrong walltime (USC)
- sister moms weren't setting job state to "running" (USC)
- don't reject jobs if requested nodes is too large node_pack=T (USC)
- add epilogue.parallel and epilogue.user.parallel (SARA)
- add $PBS_NODENUM, $PBS_MSHOST, and $PBS_NODEFILE to pelogs (USC)
- add more flexible --with-rcp='scp|rcp|mom_rcp' instead of --with-scp (USC)
- build/install a single libtorque.so (USC)
- nodes are no longer checked against server host acl list (USC)
- Tcl's buildindex now supports a 3rd arg for "destdir" to aid fakeroot installs (USC)
- fixed dynamic node destroy qmgr option
- install rm.h (USC)
- printjob now prints saved TM info (USC)
- make MOM restarts with running jobs more reliable (USC)
- fix return check in pbs_rescquery fixing segfault in pbs_sched (USC)
- add README.pbstools to contrib directory
- workaround buggy recvfrom() in Tru64 (USC)
- attempt to handle socklen_t portably (USC)
- fix infinite loop in is_stat_get() triggered by network congestion (USC)
- job suspend/resume enhancements (see qsig manpage) (USC)
- support higher file descriptors in TM by using poll() instead of select() (USC)
- immediate job delete feedback to interactive queued jobs (USC)
- move qmgr manpage from section 8 to section 1
- add SuSE initscripts to contrib/init.d/
- fix ctrl-c race while starting interactive jobs (USC)
- fix memory corruption when tm_spawn() is interrupted (USC)

TORQUE 2.0

2.0.0p6

- fix segfault in new "acl_group_sloppy" code if a group doesn't exist (USC)
- configure defaults changed to enable syslog, enable docs, and disable filesync (USC)
- pelog now correctly restores previous alarm handler (Sandia)
- misc fixes with syscalls returns, sign-mismatches, and mem corruption (USC)
- prevent MOM from killing herself on new job race condition - Linux only (USC)
- remove job delete nanny earlier to not interrupt long stageouts (USC)
- display C state later when using keep_completed (USC)
- add 'printtracking' command in src/tools (USC)
- stop overriding the user with name resolution on qsub's -o/-e args (USC)

2.0.0p5

- reorganize ji_newt structure to eliminate 64 bit data packing issues
- enable '--disable-spool' configure directive
- enable stdout/stderr stageout to search through $HOME and $HOME/.pbs_spool
- fixes to qsub's env handling for newlines and commas (UMU)
- fixes to at_arst encoding and decoding for newlines and commas (USC)
- use -p with rcp/scp (USC)
- several fixes around .pbs_spool usage (USC)
- don't create "kept" stdout/err files ugo+rw (avoid insane umask) (USC)
- qsub -V shouldn't clobber qsub's environ (USC)
- don't prevent connects to "down" nodes that are still talking (USC)
- allow file globs to work correctly under --enable-wordexp (USC)
- enable secondary group checking when evaluating queue acl_group attribute
- - enable the new queue parameter "acl_group_sloppy"
- sol10 build system fixes (USC)
- fixed node manager buffer overflow (UMU)
- fix "pbs_version" server attribute (USC)
- torque.spec updates (USC)
- remove the leading space on the node session attribute on darwin (USC)
- prevent SEGV if config file is missing/corrupt
- "keep_completed" execution queue attribute
- several misc code fixes (UMU)

2.0.0p4

- fix up socklen_t issues
- fixed epilog to report total job resource utilization
- improved RPM spec (USC)
- modified qterm to drop hung connections to bad nodes
- enhance HPUX operation

2.0.0p3

- fixed dynamic gres loading in pbs_mom (CRI)
- added torque.spec (rpmbuild -tb should work) (USC)
- new 'packages' make target (see INSTALL) (USC)
- added '-1' qstat option to display node info (UMICH)
- various fixes in file staging and copying (USC)
- - reenable stageout of directories
- - fix confusing email messages on failed stageout
- - child processes can't use MOM's logging, must use syslog
- fix overflow in RM netload (USC)
- don't check walltime on sister nodes, only on MS (ANU)
- kill_task wasn't being declared properly for all mach types (USC)

- don't unnecessarily link with libelf and libdl (USC)
- fix compile warnings with qsort/bsearch on bsd/darwin (USC)
- fix --disable-filesync to actually work (USC)
- added prolog diagnostics to 'momctl -d' output (CRI)
- added logging for job file management (CRI)
- added MOM parameter $ignwalltime (CRI)
- added $PBS_VNODENUM to job/TM env (USC)
- fix self-referencing job deps (USC)
- Use --enable-wordexp to enable variables in data staging (USC)
- $PBS_HOME/server_name is now used by MOM _iff $pbsserver isn't used_ (USC)
- Fix TRU64 compile issues (NCIFCRF)
- Expand job limits up to ULONG_MAX (NCIFCRF)
- user-supplied TMPDIR no longer treated specially (USC)
- remtree() now deals with symlinks correctly (USC)
- enable configurable mail domain (Sandia)
- configure now handles darwin8 (USC)
- configure now handles --with-scp=path and --without-scp correctly (USC)

2.0.0p2

- fix check_pwd() memory leak (USC)

2.0.0p1

- fix mpiexec stdout regression from 2.0.0p0 (USC)
- add 'qdel -m' support to enable annotating job cancellation (CRI)
- add MOM diagnostics for prolog failures and timeouts (CRI)
- interactive jobs cannot be rerunable (USC)
- be sure nodefile is removed when job is purged (USC)
- don't run epilogue multiple times when multiple jobs exit at once (USC)
- fix clearjob MOM request (momctl -c) (USC)
- fix detection of local output files with localhost or /dev/null (USC)
- new qstat/qselect -e option to only select jobs in exec queues (USC)
- $clienthost and $headnode removed, $pbsclient and $pbsserver added (USC)
- $PBS_HOME/server_name is now added to MOM's server list (USC)
- resmom transient TMPDIR (USC)
- add joblist to MOM's status and add server "mom_job_sync" (USC)
- export PBS_SCHED_HINT to pelogues if set in the job (USC)
- don't build or install pbs_rcp if --enable-scp (USC)
- set user hold on submitted jobs with invalid deps (USC)
- add initial multi-server support for HA (CRI)
- Altix cpuset enhancements (CSIRO)
- enhanced momctl to diagnose and report on connectivity issues (CRI)
- added hostname resolution diagnostics and logging (CRI)

- fixed 'first node down' rpp failure (USC)
- improved qsub response time

2.0.0p0

- torque patches for RCP and resmom (UCHSC)
- enhanced DIS logging
- improved start-up to support quick startup with down nodes
- fixed corrupt job/node/queue API reporting
- fixed tracejob for large jobs (Sandia)
- changed qdel to only send one SIGTERM at MOM level
- fixed doc build by adding AIX 5 resources docs
- added prerun timeout change (RENTEC)
- added code to handle select() EBADF - 9
- disabled MOM quota feature by default, enabled with -DTENABLEQUOTA
- cleanup MOM child error messages (USC)
- fix makedepend-sh for gcc-3.4 and higher (DTU)
- don't fallback to mom_rcp if configured to use scp (USC)

TORQUE 1.2

1.2.0p6

- enabled arch MOM config (CRI)
- fixed qrun based default scheduling to ignore down nodes (USC)
- disable unsetting of key/integer server parameters (USC)
- allow FC4 support - quota struct fix (USC)
- add fix for out of memory failure (USC)
- add file recovery failure messages (USC)
- add direct support for external scheduler extensions
- add passwd file corruption check
- add job cancel nanny patch (USC)
- recursively remove job dependencies if children can never be satisfied (USC)
- make poll_jobs the default behavior with a restat time of 45 seconds
- added 'shell-use-arg' patch (OSC)
- improved API timeout disconnect feature

- added improved rapid start up
- reworked mom-server state management (USC)
- - removed 'unknown' state
- - improved pbsnodes 'offline' management
- - fixed 'momctl -C' which actually _prevented_ an update
- - fixed incorrect math on 'tmpTime'
- - added 'polltime' to the math on 'tmpTime'
- - consolidated node state changes to new 'update_node_state()'
- - tightened up the "node state machine"
- - changed mom's state to follow the documented state guidelines
- - correctly handle "down" from mom
- - moved server stream handling out of 'is_update_stat()' to new
- 'init_server_stream()'
- - refactored the top of the main loop to tighten up state changes
- - fixed interval counting on the health check script
- - forced health check script if update state is forced
- - don't spam the server with updates on startup
- - required new addr list after connections are dropped
- - removed duplicate state updates because of broken multi-server support
- - send "down" if internal_state is down (aix's query_adp() can do this)
- - removed ferror() check on fread() because fread() randomly fails on initial
- MOM startup.
- - send "down" if health check returns "ERROR"
- - send "down" if disk space check fails.

1.2.0p5

- make '-t quick' default behavior for qterm
- added '-p' flag to qdel to enable forced job purge (USC)
- fixed server resources_available n-1 issue
- added further Altix CPUSet support (NCSA)
- added local checkpoint script support for Linux
- fixed 'premature end of message warning'
- clarify job deleted mail message (SDSC)
- fixed AIX 5.3 support in configure (WestGrid)
- fixed crash when qrun issued on job with incomplete requeue
- added support for >= 4GB memory usage (GMX)
- log job execution limits failures
- added more detailed error messages for missing user shell on mom
- fixed qsub env overflow issue

1.2.0p4

- extended job prolog to include jobname, resource, queue, and account info (MAINE)
- added support for Darwin 8/OS X 10.4 (MAINE)
- fixed suspend/resume for MPI jobs (NORWAY)
- added support for epilog.precancel to enable local job cancellation handling
- fixed build for case insensitive filesystems
- fixed relative path based Makefiles for xpbsmom
- added support for gcc 4.0
- added PBSDEBUG support to client commands to allow more verbose diagnostics of client failures
- added ALLOWCOMPUTEHOSTSUBMIT option to torque.cfg
- fixed dynamic pbs_server loglevel support
- added mom-server rpp socket diagnostics
- added support for multi-homed hosts w/SERVERHOST parameter in torque.cfg
- added support for static linking w/PBSBINDIR
- added availmem/totmem support to Darwin systems (MAINE)
- added netload support to Darwin systems (MAINE)

1.2.0p3

- enable multiple server to MOM communication
- fixed node reject message overwrite issue
- enable pre-start node health check (BOEING)
- fixed pid scanning for RHEL3 (VPAC)
- added improved vmem/mem limit enforcement and reporting (UMU)
- added submit filter return code processing to qsub

1.2.0p2

- enhance network failure messages
- fixed tracejob tool to only match correct jobs (WESTGRID)
- modified reporting of Linux availmem and totmem to allow larger file sizes
- fixed pbs_demux for OSF/TRU64 systems to stop orphaned demux processes
- added dynamic pbs_server loglevel specification
- added intelligent MOM job stat sync'ing for improved scalability (USC/CRI)
- added MOM state sync patch for dup join (USC)
- added spool dir space check (MAINE)

1.2.0p1

- add default DEFAULTMAILDOMAIN configure option
- improve configure options to use pbs environment (USC)
- use openpty() based tty management by default
- enable default resource manager extensions
- make MOM config parameters case insensitive
- added jobstartblocktime MOM parameter
- added bulk read in pbs_disconnect() (USC)
- added support for solaris 5
- added support for program args in pbsdsh (USC)
- added improved task recovery (USC)

1.2.0p0

- fixed MOM state update behavior (USC/Poland)
- fixed set_globid() crash
- added support for > 2GB file size job requirements

- updated config.guess to 2003 release
- general patch to initialize all function variables (USC)
- added patch for serial job TJE leakage (USC)
- add "hw.memsize" based physmem MOM query for darwin (Maine)
- add configure option (--disable-filesync) to speed up job submission
- set PBS mail precedence to bulk to avoid vactaion responses (VPAC)
- added multiple changes to address gcc warnings (USC)
- enabled auto-sizing of 'qstat -Q' columns
- purge DOS EOL characters from submit scripts

TORQUE 1.1

1.1.0p6

- added failure logging for various MOM job launch failures (USC)
- allow qsub '-d' relative path qsub specification
- enabled $restricted parameter w/in FIFO to allow used of non-privileged ports (SAIC)
- checked job launch status code for retry decisions
- added nodect resource_available checking to FIFO
- disabled client port binding by default for darwin systems (use --enable-darwinbind to re-enable)
- - workaround for darwin bind and pclose OS bugs
- fixed interactive job terminal control for MAC (NCIFCRF)
- added support for MAC MOM-level cpu usage tracking (Maine)
- fixed __P warning (USC)
- added support for server level resources_avail override of job nodect limits (VPAC)
- modify MOM copy files and delete file requests to handle NFS root issues (USC/CRI)
- enhance port retry code to support mac socket behavior
- clean up file/socket descriptors before execing prolog/epilog
- enable dynamic cpu set management (ORNL)
- enable array services support for memory management (ORNL)
- add server command logging to diagnostics
- fix Linux setrlimit persistance on failures

1.1.0p5

- added loglevel as MOM config parameter
- distributed job start sequence into multiple routines
- force node state/subnode state offline stat synchronization (NCSA)
- fixed N-1 cpu allocation issue (no sanity checking in set_nodes)
- enhance job start failure logging
- added continued port checking if connect fails (rentec)
- added case insensitive host authentication checks
- added support for submitfilter command line args
- added support for relocatable submitfilter via torque.cfg
- fixed offline status cleared when server restarted (USC)
- updated PBSTop to 4.05 (USC)
- fixed PServiceType array to correctly report service messages
- fixed pbs_server crash from job dependencies
- prevent MOM from truncating lock file when MOM is already running
- tcp timeout added as config option

1.1.0p4

- added 15004 error logging
- added use of openpty() call for locating pseudo terminals (SNL)
- add diagnostic reporting of config and executable version info
- add support for config push
- add support for MOM config version parameters
- log node offline/online and up/down state changes in pbs_server logs
- add MOM fork logging and home directory check
- add timeout checking in rpp socket handling
- added buffer overflow prevention routines
- added lockfile logging
- supported protected env variables with qstat

1.1.0p3

- added support for node specification w/pbsnodes -a
- added hstfile support to momctl
- added chroot (-D) support (SRCE)
- added MOM chdir pjob check (SRCE)
- fixed MOM HELLO initialization procedure
- added momctl diagnostic/admin command (shutdown, reconfig, query, diagnose)
- added MOM job abort bailout to prevent infinite loops
- added network reinitialization when socket failure detected
- added mom-to-scheduler reporting when existing job detected
- added MOM state machine failure logging

1.1.0p2

- add support for disk size reporting via pbs_mom
- fixed netload initialization
- fixed orphans on MOM fork failure
- updated to pbstop v 3.9 (USC)
- fixed buffer overflow issue in net_server.c
- added pestat package to contrib (ANU)
- added parameter checking to cpy_stage() (NCSA)
- added -x (xml output) support for 'qstat -f' and 'pbsnodes -a'
- added SSS xml library (SSS)
- updated user-project mapping enforcement (ANL)
- fix bogus 'cannot find submitfilter' message for interactive jobs
- fix incorrect job allocation issue for interactive jobs (NCSA)
- prevent failure with invalid 'servername' specification (NCSA)
- provide more meaningful 'post processing error' messages (NCSA)
- check for corrupt jobs in server database and remove them immediately
- enable SIGUSR1/SIGUSR2 pbs_mom dynamic loglevel adjustment
- profiling enhancements
- use local directory variable in scan_non_child_tasks() to prevent race condition (VPAC)
- added AIX 5 odm support for realmem reporting (VPAC)

1.1.0p1

- added pbstop to contrib (USC)
- added OSC mpiexec patch (OSC)
- confirmed OSC mom-restart patch (OSC)
- fix pbsd_init purge job tracking
- allow tracking of completed jobs (w/TORQUEKEEPCOMPLETED env)
- added support for MAC OS 10

- added qsub wrapper support
- added '-d' qsub command line flag for specifying working directory
- fixed numerous spelling issues in pbs docs
- enable logical or'ing of user and group ACL's
- allow large memory sizes for physmem under solaris (USC)
- fixed qsub SEGV on bad '-o' specification
- add null checking on ap->value
- fixed physmem() routine for tru64 systems to load compute node physical memory
- added netload tracking

1.1.0p0

- fixed Linux swap space checking
- fixed AIX5 resmom ODM memory leak
- handle split var/etc directories for default server check (CHPC)
- add pbs_check utility
- added TERAGRID nospool log bounds checking
- add code to force host domains to lower case
- verified integration of OSC prologue-environment.patch (export Resource_List.nodes in an environment variable for prologue)
- verified integration of OSC no-munge-server-name.patch (do not install over existing server_name)
- verified integration of OSC docfix.patch (fix minor manpage type)

TORQUE 1.0

1.0.1p6

- add messaging to report remote data staging failures to pbs_server
- added tcp_timeout server parameter
- add routine to mark hung nodes as down
- add torque.setup initialization script
- track okclient status
- fixed INDIANA ji_grpcache MOM crash
- fixed pbs_mom PBSLOGLEVEL/PBSDEBUG support
- fixed pbs_mom usage
- added rentec patch to MOM 'sessions' output
- fixed pbs_server --help option
- added OSC patch to allow jobs to survive MOM shutdown
- added patch to support server level node comments
- added support for reporting of node static resources via sss interface
- added support for tracking available physical memory for IRIX/Linux systems
- added support for per node probes to dynamically report local state of arbitrary value
- fixed qsub -c (checkpoint) usage

1.0.1p5

- add SuSE 9.0 support
- add Linux 2.4 meminfo support
- add support for inline comments in mom_priv/conf

- allow support for upto 100 million unique jobs
- add pbs_resources_all documentation
- fix kill_task references
- add contrib/pam_authuser

1.0.1p4

- fixed multi-line readline buffer overflow
- extended TORQUE documentation
- fixed node health check management

1.0.1p3

- added support for pbs_server health check and routing to scheduler
- added support for specification of more than one clienthost parameter
- added PW unused-tcp-interrupt patch
- added PW mom-file-descriptor-leak patch
- added PW prologue-bounce patch
- added PW mlockall patch (release mlock for MOM children)
- added support for job names up to 256 chars in length
- added PW errno-fix patch

1.0.1p2

- added support for macintosh (darwin)
- fixed qsub 'usage' message to correctly represent '-j',
- '-k', '-m', and '-q' support
- add support for 'PBSAPITIMEOUT' env variable
- fixed MOM dec/hp/linux physmem probes to support 64 bit
- fixed MOM dec/hp/linux availmem probes to support 64 bit
- fixed MOM dec/hp/linux totmem probes to support 64 bit
- fixed MOM dec/hp/linux disk_fs probes to support 64 bit
- removed pbs server request to bogus probe
- added support for node 'message' attribute to report internal
- failures to server/scheduler
- corrected potential buffer overflow situations
- improved logging replacing 'unknown' error with real error message
- enlarged internal tcp message buffer to support 2000 proc systems
- fixed enc_attr return code checking

1.0.1p1

- **NOTE:** See TORQUE distribution CHANGELOG file

1.0.1p0

- **NOTE:** See TORQUE distribution CHANGELOG file

See Also

- TORQUE Installation TroubleShooting