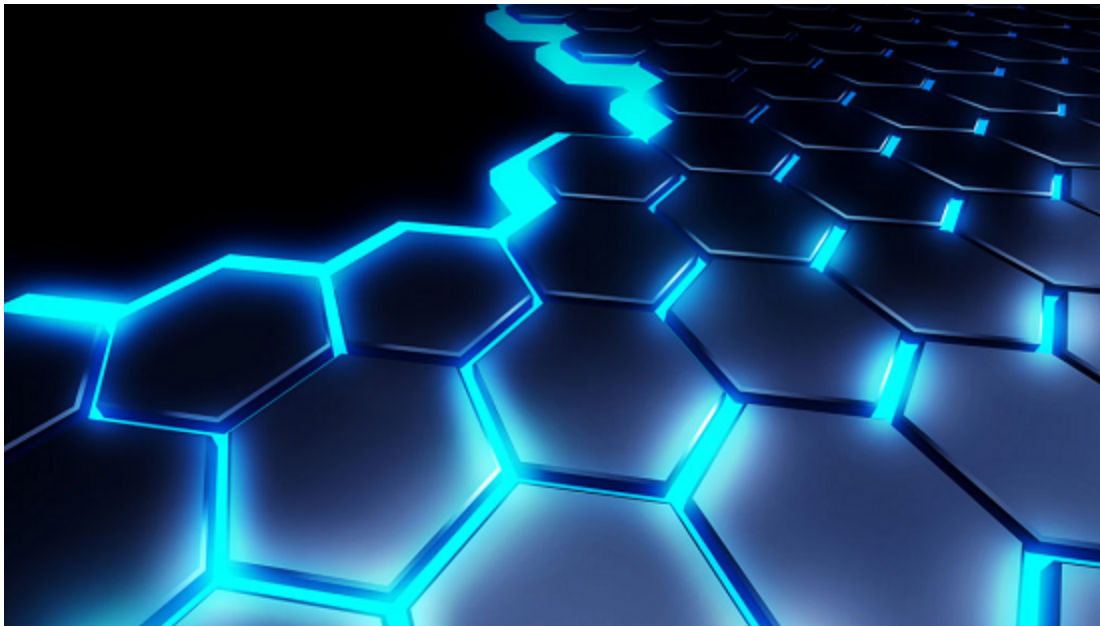


# Torque Resource Manager

## Administrator Guide 7.1.0

March 2025



## Legal Notices

© 2013, 2025 Adaptive Computing Enterprises, Inc. All rights reserved.

This documentation and related software are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Distribution of this document for commercial purposes in either hard or soft copy form is strictly prohibited without prior written consent from Adaptive Computing Enterprises, Inc.

This documentation and related software may provide access to or information about content, products, and services from third-parties. Adaptive Computing is not responsible for and expressly disclaims all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Adaptive Computing. Adaptive Computing will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Adaptive Computing.

Adaptive Computing, Cluster Resources, Moab, Moab Workload Manager, Moab Viewpoint, Moab Cluster Manager, Moab Cluster Suite, Moab Grid Scheduler, Moab Grid Suite, Moab Access Portal, and other Adaptive Computing products are either registered trademarks or trademarks of Adaptive Computing Enterprises, Inc. The Adaptive Computing logo and the Cluster Resources logo are trademarks of Adaptive Computing Enterprises, Inc. All other company and product names may be trademarks of their respective companies.

The information contained herein is subject to change without notice and is not warranted to be error free. If you find any errors, please report them to us in writing.

Adaptive Computing Enterprises, Inc.  
1100 5th Avenue South, Suite #201  
Naples, FL 34102  
+1 (239) 330-6093  
[www.adaptivecomputing.com](http://www.adaptivecomputing.com)

# Contents

<b>Chapter 1: Introduction</b>	<b>12</b>
1.1 Torque Resource Manager Administrator Guide Overview	12
1.2 Getting Started	13
1.2.1 What is a Resource Manager?	14
1.2.2 What are Batch Systems?	14
1.2.3 Basic Job Flow	15
<b>Chapter 2: Installation and Configuration</b>	<b>17</b>
2.1 Torque Installation Overview	17
2.1.1 Basic Server Configuration	18
2.1.2 Torque Architecture	20
2.1.3 Installing Torque Resource Manager	20
2.1.4 Compute Nodes	27
2.1.5 Enabling Torque as a Service	29
2.2 Initializing/Configuring Torque on the Server (pbs_server)	29
2.2.1 Specifying Compute Nodes	30
2.2.2 Configuring Torque on Compute Nodes	31
2.2.3 Configuring Ports	32
2.2.4 Configuring trqauthd for Client Commands	36
2.2.5 Finalizing Configurations	37
2.3 Advanced Configuration	37
2.3.1 Customizing the Install	38
2.3.2 Server Configuration	49
2.3.3 Setting Up the MOM Hierarchy (Optional)	54
2.3.4 Opening Ports in a Firewall	57
2.3.5 Port Reference	58
2.4 Manual Setup of Initial Server Configuration	63
2.5 Server Node File Configuration	64
2.5.1 Basic Node Specification	65
2.5.2 Specifying Virtual Processor Count for a Node	65
2.5.3 Specifying GPU Count for a Node	66
2.5.4 Specifying Node Features (Node Properties)	66
2.6 Testing Server Configuration	67
2.7 Configuring Torque for NUMA Systems	68
2.7.1 Torque NUMA-Aware Configuration	69
2.7.2 Torque NUMA-Support Configuration	70
2.8 Torque Multi-MOM	75

2.8.1 Multi-MOM Configuration .....	76
2.8.2 Stopping pbs_mom in Multi-MOM Mode .....	77
2.9 Supporting MIG Devices in Torque .....	77
2.9.1 Requirements .....	78
2.9.2 Functionality .....	78
2.9.3 Limitations .....	78
<b>Chapter 3: Submitting and Managing Jobs .....</b>	<b>79</b>
3.1 Job Submission .....	79
3.1.1 Multiple Job Submission .....	81
3.1.2 Managing Multi-Node Jobs .....	82
3.1.3 Requesting Resources .....	83
3.1.4 Requesting NUMA-Aware Resources .....	92
3.1.5 Requesting Generic Resources .....	93
3.1.6 Requesting Floating Resources .....	94
3.1.7 Requesting Other Resources .....	94
3.1.8 Exported Batch Environment Variables .....	94
3.1.9 Enabling Trusted Submit Hosts .....	95
3.1.10 Example Submit Scripts .....	96
3.2 Monitoring Jobs .....	96
3.3 Canceling Jobs .....	98
3.4 Job Preemption .....	99
3.5 Keeping Completed Jobs .....	99
3.6 Job Checkpoint and Restart .....	100
3.6.1 Introduction to BLCR .....	101
3.6.2 Configuration Files and Scripts .....	101
3.6.3 Starting a Checkpointable Job .....	105
3.6.4 Checkpointing a Job .....	107
3.6.5 Restarting a Job .....	107
3.6.6 Acceptance Tests .....	108
3.7 Job Exit Status .....	108
3.8 Torque Process Tracking .....	111
3.8.1 Default Process Tracking .....	111
3.8.2 Task Manager API .....	111
3.8.3 Process Tracking with cgroups/cpusets .....	112
3.9 Large Job Arrays .....	113
<b>Chapter 4: Managing Nodes .....</b>	<b>114</b>
4.1 Adding Nodes .....	114
4.2 Node Properties .....	116
4.2.1 Run-Time Node Changes .....	116
4.2.2 Manual Node Changes .....	117

4.2.3 Adding Memory to a Node .....	117
4.3 Changing Node State .....	118
4.3.1 Marking Jobs Offline .....	118
4.3.2 Listing Node States .....	118
4.3.3 Node Recovery .....	118
4.4 Changing Node Power States .....	118
4.5 Host Security .....	121
4.5.1 Enabling PAM with Torque .....	121
4.5.2 Using PAM Exception Instructions .....	122
4.5.3 Legacy Torque PAM Configuration .....	123
4.6 Linux cpuset Support .....	123
4.6.1 cpuset Overview .....	124
4.6.2 cpuset Support .....	124
4.6.3 Configuring cpuset .....	125
4.6.4 cpuset Advantages/Disadvantages .....	125
4.7 Scheduling Cores .....	125
4.7.1 Geometry Request Configuration .....	126
4.7.2 Geometry Request Usage .....	126
4.7.3 Geometry Request Considerations .....	127
4.8 Scheduling Accelerator Hardware .....	127
4.9 Node Resource Plug-In .....	127
4.9.1 Plug-In Implementation Recommendations .....	128
4.9.2 Building the Plug-In .....	128
4.9.3 Testing the Plug-In .....	128
4.9.4 Enabling the Plug-In .....	130
<b>Chapter 5: Setting Server Policies .....</b>	<b>132</b>
5.1 Queue Configuration .....	132
5.1.1 Example Queue Configuration .....	133
5.1.2 Setting Queue Resource Controls .....	133
5.1.3 Setting a Default Queue .....	134
5.1.4 Mapping a Queue to Subset of Resources .....	134
5.1.5 Creating a Routing Queue .....	135
5.2 Server High Availability .....	137
5.2.1 Redundant Server Host Machines .....	137
5.2.2 Enabling High Availability .....	138
5.2.3 Enhanced High Availability with Moab .....	139
5.2.4 How Commands Select the Correct Server Host .....	139
5.2.5 Job Names .....	140
5.2.6 Persistence of the pbs_server Process .....	140
5.2.7 High Availability of the NFS Server .....	140
5.2.8 Installing Torque in High Availability Mode .....	141

5.2.9 Installing Torque in High Availability Mode on Headless Nodes .....	145
5.2.10 Example Setup of High Availability .....	150
5.3 Setting min_threads and max_threads .....	150
<b>Chapter 6: Integrating Schedulers for Torque .....</b>	<b>152</b>
<b>Chapter 7: Configuring Data Management .....</b>	<b>153</b>
7.1 SCP Setup .....	153
7.1.1 Generating SSH Key on Source Host .....	154
7.1.2 Copying Public SSH Key to Each Destination Host .....	154
7.1.3 Configuring the SSH Daemon on Each Destination Host .....	154
7.1.4 Validating Correct SSH Configuration .....	155
7.1.5 Enabling Bi-Directional SCP Access .....	155
7.1.6 Troubleshooting .....	155
7.2 NFS and Other Networked Filesystems .....	155
7.3 File stage-in/stage-out .....	156
<b>Chapter 8: MPI (Message Passing Interface) Support .....</b>	<b>158</b>
8.1 MPICH .....	158
8.1.1 MPIExec Overview .....	158
8.1.2 MPIExec Troubleshooting .....	159
8.1.3 General MPI Troubleshooting .....	160
8.2 Open MPI .....	160
<b>Chapter 9: Resources .....</b>	<b>163</b>
9.1 About Resources .....	163
9.2 Configuration .....	163
9.3 Utilization .....	164
9.4 Node States .....	165
<b>Chapter 10: Accounting Records .....</b>	<b>166</b>
10.1 Location and Contents .....	166
10.2 Record Types .....	166
10.3 Accounting Variables .....	167
10.4 Fields .....	168
<b>Chapter 11: Job Logging .....</b>	<b>171</b>
11.1 Job Log Location and Name .....	171
11.2 Enabling Job Logs .....	171
<b>Chapter 12: NUMA and Torque .....</b>	<b>173</b>
12.1 Supported NUMA Systems .....	173
12.2 NUMA-Aware Systems .....	173
12.2.1 About NUMA-Aware Systems .....	174

12.2.2 Installation and Configuration .....	175
12.2.3 Job Resource Requests .....	175
12.2.4 Job Monitoring .....	176
12.2.5 Moab/Torque NUMA Configuration .....	176
12.2.6 Considerations when Upgrading Versions or Changing Hardware .....	176
12.3 NUMA Tutorials .....	176
12.3.1 NUMA Primer .....	176
12.3.2 How NUMA Places Jobs .....	186
12.3.3 NUMA Discovery and Persistence .....	188
12.4 -L NUMA Resource Request .....	191
12.4.1 Syntax .....	191
12.4.2 Allocation Options .....	191
12.5 pbsnodes with NUMA-Awareness .....	200
12.6 NUMA-Support Systems .....	201
12.6.1 About NUMA-Supported Systems .....	202
12.6.2 Torque Installation and Configuration .....	202
12.6.3 Moab/Torque NUMA Configuration .....	202
<b>Chapter 13: Troubleshooting .....</b>	<b>203</b>
13.1 Automatic Queue and Job Recovery .....	203
13.2 Host Resolution .....	204
13.3 Firewall Configuration .....	204
13.4 Torque Log Files .....	204
13.4.1 pbs_server and pbs_mom Log Files .....	204
13.4.2 trqauthd Log Files .....	205
13.5 Using tracejob to Locate Job Failures .....	206
13.5.1 Overview .....	206
13.5.2 Syntax .....	206
13.5.3 Example .....	206
13.6 Using GDB to Locate Job Failures .....	208
13.7 Other Diagnostic Options .....	209
13.8 Stuck Jobs .....	209
13.9 Frequently Asked Questions .....	210
13.9.1 Cannot connect to server: error=15034 .....	211
13.9.2 Deleting 'stuck' jobs .....	211
13.9.3 Which user must run Torque? .....	212
13.9.4 Scheduler cannot run jobs - rc: 15003 .....	212
13.9.5 PBS_Server: pbsd_init, Unable to read server database .....	212
13.9.6 qsub will not allow the submission of jobs requesting many processors .....	213
13.9.7 qsub reports 'Bad UID for job execution' .....	214
13.9.8 Why does my job keep bouncing from running to queued? .....	214
13.9.9 How do I use PVM with Torque? .....	215

13.9.10 My build fails attempting to use the TCL library .....	215
13.9.11 My job will not start, failing with the message 'cannot send job to mom, state=PRERUN' ....	216
13.9.12 How do I determine what version of Torque I am using? .....	216
13.9.13 How do I resolve autogen.sh errors that contain "error: possibly undefined macro: AC_MSG_ERROR"?	216
13.9.14 Why are there so many error messages in the client logs (trqauthd logs) when I don't notice client commands failing? .....	216
13.10 Compute Node Health Check .....	216
13.10.1 Configuring MOMs to Launch a Health Check .....	217
13.10.2 Creating the Health Check Script .....	218
13.10.3 Adjusting Node State Based on the Health Check Output .....	218
13.10.4 Example Health Check Script .....	218
13.11 Debugging .....	219
13.11.1 Diagnostic and Debug Options .....	219
13.11.2 Torque Error Codes .....	220
<b>Appendix A: Commands Overview .....</b>	<b>226</b>
A.1 Torque Services .....	227
A.2 Client Commands .....	227
A.3 momctl .....	228
A.4 pbs_mom .....	236
A.5 pbs_server .....	242
A.6 pbs_track .....	246
A.7 pbsdsh .....	248
A.8 pbsnodes .....	250
A.9 qalter .....	253
A.10 qchkpt .....	263
A.11 qdel .....	264
A.12 qgpumode .....	267
A.13 qgpureset .....	268
A.14 qhold .....	269
A.15 qmgr .....	271
A.16 qmove .....	275
A.17 qorder .....	276
A.18 qrerun .....	278
A.19 qrls .....	279
A.20 qrun .....	281
A.21 qsig .....	283
A.22 qstat .....	285
A.23 qsub .....	293
A.24 qterm .....	314
A.25 trqauthd .....	316



<b>Appendix B: Server Parameters</b>	<b>319</b>
<b>Appendix C: Node Manager (MOM) Configuration</b>	<b>347</b>
C.1 MOM Parameters	347
C.2 Node Features and Generic Consumable Resource Specification	370
<b>Appendix D: Diagnostics and Error Codes</b>	<b>371</b>
<b>Appendix E: Preparing to Upgrade</b>	<b>379</b>
E.1 Considerations Before Upgrading	379
E.2 To Upgrade	379
E.3 Rolling Upgrade	380
<b>Appendix F: Large Cluster Considerations</b>	<b>382</b>
F.1 Scalability Guidelines	382
F.2 End-User Command Caching	383
F.3 Moab and Torque Configuration for Large Clusters	385
F.4 Starting Torque in Large Environments	386
F.5 Other Considerations	386
F.5.1 job_stat_rate	387
F.5.2 poll_jobs	387
F.5.3 Scheduler Settings	387
F.5.4 File System	387
F.5.5 Network ARP Cache	387
<b>Appendix G: Prologue and Epilogue Scripts</b>	<b>389</b>
G.1 MOM Prologue and Epilogue Scripts	389
G.2 Script Order of Execution	391
G.3 Script Environment	392
G.3.1 Prologue Environment	392
G.3.2 Epilogue Environment	393
G.3.3 Environment Variables	394
G.3.4 Standard Input	395
G.4 Per Job Prologue and Epilogue Scripts	395
G.5 Prologue and Epilogue Scripts Time Out	396
G.6 Prologue Error Processing	396
<b>Appendix H: Running Multiple Torque Servers and MOMs on the Same Node</b>	<b>400</b>
H.1 Configuring Multiple Servers to Run on the Same Node	400
H.2 Configuring the First Torque	400
H.3 Configuring the Second Torque	400
H.4 Bringing the First Torque Server Online	400
H.5 Bringing the Second Torque Server Online	401

<b>Appendix I: Security Overview</b>	<b>402</b>
<b>Appendix J: Job Submission Filter (qsub Wrapper)</b>	<b>403</b>
<b>Appendix K: torque.cfg Configuration File</b>	<b>405</b>
<b>Appendix L: Torque Quick Start Guide</b>	<b>411</b>
L.1 Initial Installation	411
L.2 Initialize/Configure Torque on the Server (pbs_server)	412
L.3 Install Torque on the Compute Nodes	412
L.4 Configure Torque on the Compute Nodes	413
L.5 Configure Data Management on the Compute Nodes	413
L.6 Update Torque Server Configuration	413
L.7 Start the pbs_mom Daemons on Compute Nodes	414
L.8 Verify Correct Torque Installation	414
L.9 Enable the Scheduler	414
L.10 (Optional) Startup/Shutdown Service Script for Torque/Moab	415
<b>Appendix M: BLCR Acceptance Tests</b>	<b>416</b>
M.1 Test Environment	416
M.2 Test 1 - Basic Operation	416
M.2.1 Introduction	417
M.2.2 Test Steps	417
M.2.3 Possible Failures	417
M.2.4 Successful Results	417
M.3 Test 2 - Persistence of Checkpoint Images	419
M.3.1 Introduction	419
M.3.2 Test Steps	419
M.3.3 Possible Failures	420
M.3.4 Successful Results	420
M.4 Test 3 - Restart After Checkpoint	420
M.4.1 Introduction	420
M.4.2 Test Steps	420
M.4.3 Successful Results	420
M.5 Test 4 - Multiple Checkpoint/Restart	420
M.5.1 Introduction	420
M.5.2 Test Steps	421
M.5.3 Successful Results	421
M.6 Test 5 - Periodic Checkpoint	421
M.6.1 Introduction	421
M.6.2 Test Steps	421
M.6.3 Successful Results	421
M.7 Test 6 - Restart from Previous Image	422

M.7.1 Introduction .....	422
M.7.2 Test Steps .....	422
M.7.3 Successful Results .....	422
<b>Appendix N: Queue Attributes .....</b>	<b>423</b>
N.1 Queue Attribute Reference .....	423
N.2 Attributes .....	423
N.3 Assigning Queue Resource Limits .....	433

## Chapter 1: Introduction

### Welcome to the *Torque Resource Manager Administrator Guide 7.1.0*

This guide is intended as a reference for system administrators.

In this chapter:

- [1.1 Torque Resource Manager Administrator Guide Overview](#)
- [1.2 Getting Started](#)

## 1.1 Torque Resource Manager Administrator Guide Overview

[Chapter 1: Introduction](#) provides basic introduction information to help you get started using Torque.

[Chapter 2: Installation and Configuration](#) provides the details for installation and initialization, advanced configuration options, and (optional) `qmgr` option necessary to get the system up and running. System testing is also covered.

[Chapter 3: Submitting and Managing Jobs](#) covers different actions applicable to jobs. The first section 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.

[Chapter 4: Managing Nodes](#) covers administrator tasks relating to nodes, which include 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 [Host Security](#).

[Chapter 5: Setting Server Policies](#) details server-side configurations of queue and high availability.

[Chapter 6: Integrating Schedulers for Torque](#) offers information about using the native scheduler versus an advanced scheduler.

[Chapter 7: Configuring Data Management](#) deals with issues of data management. For non-network file systems, [SCP Setup](#) details setting up SSH keys and nodes to automate transferring data. [NFS and Other Networked Filesystems](#) covers configuration for these file systems. This chapter also addresses the use of file staging using the `stagein` and `stageout` directives of the `qsub` command.

[Chapter 8: MPI \(Message Passing Interface\) Support](#) offers details supporting MPI.

[Chapter 9: Resources](#) covers configuration, utilization, and states of resources.

[Chapter 10: Accounting Records](#) explains how jobs are tracked by Torque for accounting purposes.

[Chapter 11: Job Logging](#) explains how to enable job logs that contain information for completed jobs.

[Chapter 12: NUMA and Torque](#) provides a centralized location for information on configuring Torque for NUMA systems.

[Chapter 13: Troubleshooting](#) is a guide that offers help with general problems. It includes FAQ and instructions for how to set up and use compute node checks. It also explains how to debug Torque.

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

- [Appendix A: Commands Overview](#)
- [Appendix B: Server Parameters](#)
- [Appendix C: Node Manager \(MOM\) Configuration](#)
- [Appendix D: Diagnostics and Error Codes](#)
- [Appendix E: Preparing to Upgrade](#)
- [Appendix F: Large Cluster Considerations](#)
- [Appendix G: Prologue and Epilogue Scripts](#)
- [Appendix H: Running Multiple Torque Servers and MOMs on the Same Node](#)
- [Appendix I: Security Overview](#)
- [Appendix J: Job Submission Filter \(qsub Wrapper\)](#)
- [Appendix K: torque.cfg Configuration File](#)
- [Appendix L: Torque Quick Start Guide](#)
- [Appendix M: BLCR Acceptance Tests](#)
- [Appendix N: Queue Attributes](#)

## 1.2 Getting Started

This section contains some basic information to help you get started using Torque.

In this section:

[1.2.1 What is a Resource Manager?](#)

[1.2.2 What are Batch Systems?](#)

[1.2.3 Basic Job Flow](#)

## 1.2.1 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. Resource managers provide the low-level functionality to start, hold, cancel, and monitor jobs. Without these capabilities, a scheduler alone cannot control jobs.

## 1.2.2 What are Batch Systems?

While Torque is flexible enough to handle scheduling a conference room, it is primarily used in batch systems. Torque is based on a job scheduler called Portable Batch System (PBS). 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.

Component	Description
<b>Master Node</b>	A batch system will have a master node where <i>pbs_server</i> 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.

Component	Description
<b>Submit/Interactive Nodes</b>	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 <i>qsub</i> and <i>qhold</i> ).
<b>Compute Nodes</b>	Compute nodes are the workhorses of the system. Their role is to execute submitted jobs. On each compute node, <i>pbs_mom</i> runs to start, kill, and manage submitted jobs. It communicates with <i>pbs_server</i> on the master node. Depending on the needs of the systems, a compute node may double as the master node (or more).
<b>Resources</b>	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.

### 1.2.3 Basic Job Flow

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

Stage	Description
<b>Creation</b>	<p>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 (<i>walltime</i>), what resources are necessary to run, and what to execute. The following is an example submit file:</p> <pre>#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</pre> <p>This submit script specifies the name of the job (<i>localBlast</i>), what environment to use (<i>/bin/sh</i>), that it needs both processors on a single node (<i>nodes=1:ppn=2</i>), 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.</p>

Stage	Description
<b>Submission</b>	A job is submitted with the <code>qsub</code> 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.
<b>Execution</b>	Jobs often spend most of their lifecycle executing. While a job is running, its status can be queried with <code>qstat</code> .
<b>Finalization</b>	When a job completes, by default, the <code>stdout</code> and <code>stderr</code> files are copied to the directory where the job was submitted.



## Chapter 2: Installation and Configuration

This chapter contains some basic information about Torque, including how to install and configure it on your system.

In this chapter:

- [2.1 Torque Installation Overview](#)
- [2.2 Initializing/Configuring Torque on the Server \(pbs\\_server\)](#)
- [2.3 Advanced Configuration](#)
- [2.4 Manual Setup of Initial Server Configuration](#)
- [2.5 Server Node File Configuration](#)
- [2.6 Testing Server Configuration](#)
- [2.7 Configuring Torque for NUMA Systems](#)
- [2.8 Torque Multi-MOM](#)
- [2.9 Supporting MIG Devices in Torque](#)

### 2.1 Torque Installation Overview

This section contains information about Torque architecture and explains how to install Torque. It also describes how to install Torque packages on compute nodes and how to enable Torque as a service.

In this section:

- [2.1.1 Basic Server Configuration](#)
- [2.1.2 Torque Architecture](#)
- [2.1.3 Installing Torque Resource Manager](#)
- [2.1.4 Compute Nodes](#)
- [2.1.5 Enabling Torque as a Service](#)

---

#### Related Topics

- [Chapter 13: Troubleshooting](#)

## 2.1.1 Basic Server Configuration

In this topic:

[2.1.1.A Server Configuration File \(serverdb\)](#)

[2.1.1.B ./torque.setup](#)

[2.1.1.C pbs\\_server -t create](#)

[2.1.1.D Setting Up the Environment for pbs\\_server and pbs\\_mom](#)

### 2.1.1.A Server Configuration File (serverdb)

The server configuration is maintained in a file named `serverdb`, located in `TORQUE_HOME/server_priv`. The `serverdb` file contains all parameters pertaining to the operation of Torque plus all of the queues that are in the configuration. 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:

- As root, execute `./torque.setup` from the build directory (see [./torque.setup](#)).
- Use `pbs_server -t create` (see [-t](#)).

Restart `pbs_server` after initializing `serverdb`:

```
> qterm
> systemctl start pbs_server.service
```

### 2.1.1.B ./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 node_check_rate = 150
set server tcp_timeout = 6
set server mom_job_sync = True
set server keep_completed = 300

```

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

### 2.1.1.C 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`:

```
> pbs_server -t create
```

To see the configuration and verify that Torque is configured correctly, use `qmgr`:

```

> qmgr -c 'p s'

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

```

### 2.1.1.D Setting Up the Environment for pbs\_server and pbs\_mom

The `pbs_environment` file (default location: `TORQUE_HOME/pbs_environment`) will be sourced by `pbs_mom` and `pbs_server` when they are launched. If there are environment variables that should be set for `pbs_server` and/or `pbs_mom`, they can be placed in this file.

A `pbs_environment` file with a non-default name and/or location can be specified before compilation with the `--with-environ=PATH` configuration option. See [Table 2-2: Optional Packages](#) for more information. To determine whether a non-default `pbs_environment` file is in use, run `pbs_server --about`.

**i** The `pbs_environment` file should not be confused with the `PBS_ENVIRONMENT` job environment variable.

## Related Topics

- [2.3 Advanced Configuration](#)
- [13.11 Debugging](#)
- [Appendix C: Node Manager \(MOM\) Configuration](#)

## 2.1.2 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*.

## 2.1.3 Installing Torque Resource Manager

This topic contains instructions on how to install and start Torque Resource Manager (Torque).

In this topic:

[2.1.3.A Requirements](#)

[2.1.3.B Open Necessary Ports](#)

- [2.1.3.C Install Dependencies, Packages, or Clients](#)
- [2.1.3.D Install Torque Server](#)
- [2.1.3.E Install Torque MOMs](#)
- [2.1.3.F Install Torque Clients](#)
- [2.1.3.G Configure Data Management](#)

## 2.1.3.A Requirements

### Supported Operating Systems

- Red Hat Linux 7, 8
- SUSE Linux 12, 15
- Ubuntu 18.04, 20.04, 22.04

**Note:** Moab 10.x / Torque 7.x are required to use CentOS 8, RHEL 8, and Ubuntu 18.04 and later.

### Software Requirements

- libxml2-devel package (package name may vary)
- openssl-devel package (package name may vary)
- Tcl/Tk version 8 or later if you plan to build the GUI portion of Torque, or use a Tcl-based scheduler
- cpusets and cgroups

cgroups are supported and cpusets are handled by the cgroup cpuset subsystem.



We recommend that you use `--enable-cgroups` instead of `--enable-cpuset`. `--enable-cpuset` is deprecated and no new features will be added to it.

- boost version: 1.41 or later
- libcgroup version: Red Hat-based systems must use libcgroup version 0.40.rc1-16.el6 or later; SUSE-based systems need to use a comparative libcgroup version.
- libhwloc version: 1.9.1 is the minimum supported, however NVIDIA K80 requires libhwloc 1.11.0. Instructions for installing hwloc are provided as part of the Torque Resource Manager install or upgrade instructions.

- If using NVIDIA/NVML configuration flags:
  - NVIDIA driver version  $\geq$  450.80.02
  - CUDA driver  $\geq$  11.0
- If you build Torque from source, the following additional software is required:
  - gcc
  - gcc-c++
  - posix-compatible version of make
  - libtool 1.5.22 or later
  - boost-devel 1.36.0 or later

**i** Red Hat-based systems come packaged with 1.53.0. If needed, use the `--with-boost-path=DIR` option to change the packaged boost version. See [2.3.1 Customizing the Install](#) for more information.

### 2.1.3.B Open Necessary Ports

Torque requires certain ports to be open for essential communication.

If your site is running firewall software on its hosts, you will need to configure the firewall to allow connections to the necessary ports.

Location	Port	Function	When Needed
Torque Server Host	15001	Torque Client and MOM communication to Torque Server	Always
Torque MOM Host (Compute Nodes)	15002	Torque Server communication to Torque MOMs	Always
Torque MOM Host (Compute Nodes)	15003	Torque MOM communication to other Torque MOMs	Always

If using the MOM hierarchy, documented in [2.3.3 Setting Up the MOM Hierarchy \(Optional\)](#), you must also open port 15003 from the server to the nodes.

See also:

- [2.2.3 Configuring Ports](#) for more information on how to configure the ports that Torque uses for communication.

- [2.3.4 Opening Ports in a Firewall](#) for general instructions and an example of how to open ports in the firewall.

### 2.1.3.C Install Dependencies, Packages, or Clients

#### Install Packages

On the Torque Server Host, use the following commands to install the `libxml2-devel`, `openssl-devel`, and `boost-devel` packages.

- Red Hat-based systems:

```
[root]# yum install libtool openssl-devel libxml2-devel boost-devel gcc gcc-c++
```

- SUSE-based systems:

```
[root]# zypper install libopenssl-devel libtool libxml2-devel boost-devel gcc gcc-c++
make gmake postfix
```

#### Install hwloc

On the Torque Server Host, each Torque MOM Host, and each Torque Client Host, use the following commands to install the `hwloc` development package:

- Red Hat-based systems:

```
[root]# yum install hwloc-devel
```

- SUSE-based systems:

```
[root]# zypper install hwloc-devel
```

- Ubuntu-based systems:

```
[root]# apt install libhwloc-dev
```

### 2.1.3.D Install Torque Server

**i** You *must* complete the tasks to install the dependencies, packages, or clients before installing Torque Server. See [2.1.3.C Install Dependencies, Packages, or Clients](#).

If your configuration uses firewalls, you *must also* open the necessary ports before installing the Torque Server. See [2.1.3.B Open Necessary Ports](#).

On the Torque Server Host, do the following.

1. Download the latest Torque build from [Adaptive Computing Torque Downloads](#):

```
[root]# tar -xzvf torque-7.1.0.tar.gz
[root]# cd torque-7.1.0/
```

2. Determine which `./configure` command options you need to add, based on your system configuration.

At a minimum, you add:

`--enable-cgroups`

`--with-hwloc-path=/usr/local` (see [2.1.3.A Requirements](#) for more information)

**i** These instructions assume you are using `cgroups`. When `cgroups` are supported, `cpusets` are handled by the `cgroup` `cpuset` subsystem. If you are not using `cgroups`, use `--enable-cpusets` instead.

**i** For SUSE-based systems only. If `--enable-gui` is part of your configuration, do the following:

```
$ cd /usr/lib64
$ ln -s libXext.so.6.4.0 libXext.so
$ ln -s libXss.so.1 libXss.so
```

When finished, `cd` back to your install directory.

See [2.3.1 Customizing the Install](#) for more information on which options are available to customize the `./configure` command.

3. Run each of the following commands in order:

```
[root]# ./configure --enable-cgroups --with-hwloc-path=/usr/local # add any other
specified options
[root]# make
[root]# make install
```

4. Source the appropriate profile file to add `/usr/local/bin` and `/usr/local/sbin` to your path:

```
[root]# . /etc/profile.d/torque.sh
```

5. Initialize `serverdb` by executing the `torque.setup` script:

```
[root]# ./torque.setup root
```

6. Add nodes to the `/var/spool/torque/server_priv/nodes` file. See [2.2.1 Specifying Compute Nodes](#) for information on syntax and options for specifying compute nodes.



## 7. Configure pbs\_server to start automatically at system boot, and then start the daemon:

```
[root]# qterm
[root]# systemctl enable pbs_server.service
[root]# systemctl start pbs_server.service
```

### 2.1.3.E Install Torque MOMs

In most installations, you will install a Torque MOM on each of your compute nodes.

**i** See [2.2.1 Specifying Compute Nodes](#) or [2.2.2 Configuring Torque on Compute Nodes](#) for more information.

Do the following.

1. On the Torque Server Host, do the following.
  - a. Create the self-extracting packages that are copied and executed on your nodes:

```
[root]# make packages
Building ./torque-package-clients-linux-x86_64.sh ...
Building ./torque-package-mom-linux-x86_64.sh ...
Building ./torque-package-server-linux-x86_64.sh ...
Building ./torque-package-gui-linux-x86_64.sh ...
Building ./torque-package-devel-linux-x86_64.sh ...
Done.

The package files are self-extracting packages that can be copied and executed
on your production machines. Use --help for options.
```

- b. Copy the self-extracting MOM packages to each Torque MOM Host.

We recommend that you use a remote shell, such as SSH, to install packages on remote systems. Set up shared SSH keys if you do not want to supply a password for each Torque MOM Host.

```
[root]# scp torque-package-mom-linux-x86_64.sh <mom-node>:
```

- c. Copy the pbs\_mom startup script to each Torque MOM Host:

```
[root]# scp contrib/systemd/pbs_mom.service <mom-node>:/usr/lib/systemd/system/
```

2. On *each* Torque MOM Host, do the following.

- a. Install cgroup-tools.
    - Red Hat-based systems:

```
[root]# yum install libcgroup-tools
```

- SUSE-based systems:

```
[root]# zypper install libcgroup-tools
```

b. Install cgroup-tools.

- Red Hat-based systems:

```
[root]# yum install libcgroup-tools
```

- SUSE-based systems:

```
[root]# zypper install libcgroup-tools
```

c. Install the self-extracting MOM package:

```
[root]# ./torque-package-mom-linux-x86_64.sh --install
```

d. (Optional.) If you expect your jobs to require more than the default 12 MB of stack space, increase the stack limit by editing the `LimitSTACK` setting in `/usr/lib/systemd/system/pbs_mom.service`:

```
LimitSTACK=infinity
```

e. Configure `pbs_mom` to start at system boot, and then start the daemon:

```
[root]# systemctl enable pbs_mom.service
[root]# systemctl start pbs_mom.service
```

### 2.1.3.F Install Torque Clients

If you want to have the Torque client commands installed on hosts other than the Torque Server Host (such as the compute nodes or separate login nodes), do the following.

1. On the Torque Server Host, do the following.
  - a. Copy the self-extracting client package to *each* Torque Client Host:

**i** We recommend that you use a remote shell, such as SSH, to install packages on remote systems. Set up shared SSH keys if you do not want to supply a password for each Torque Client Host.

```
[root]# scp torque-package-clients-linux-x86_64.sh <torque-client-host>:
```

b. Copy the `trqauthd` startup script to *each* Torque Client Host:

```
[root]# scp contrib/systemd/trqauthd.service <torque-client-host>:/usr/lib/systemd/system/
```

## 2. On *each* Torque Client Host, do the following.

### a. Install the self-extracting client package:

```
[root]# ./torque-package-clients-linux-x86_64.sh --install
```

### b. Enable and start the trqauthd service:

```
[root]# systemctl enable trqauthd.service
[root]# systemctl start trqauthd.service
```

## 2.1.3.G Configure Data Management

When a batch job completes, stdout and stderr files are generated and placed in the spool directory on the master Torque MOM Host for the job instead of the submit host. You can configure the Torque batch environment to copy the stdout and stderr files back to the submit host. See [Chapter 7: Configuring Data Management](#) for more information.

## 2.1.4 Compute Nodes

Use the Adaptive Computing Torque package system to create self-extracting tarballs that can be distributed and installed on compute nodes. The Torque packages are customizable. See the `INSTALL` file for additional options and features.

**i** If you installed Torque using the RPMs, you must install and configure your nodes manually by modifying the `/var/spool/torque/mom_priv/config` file of each one. This file is identical for all compute nodes and can be created on the head node and distributed in parallel to all systems.

```
[root]# vi /var/spool/torque/mom_priv/config

$pbsserver      headnode      # hostname running pbs server
$logevent       225           # bitmap of which events to log

[root]# service pbs_mom restart
```

## To Create Torque Packages

### 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 Torque packages on the compute nodes.

We recommend that you use a remote shell, such as SSH, to install Torque packages on remote systems. Set up shared SSH keys if you do not want to supply a password for each host.

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

The following is an example of 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
> for i in node01 node02 node03 node04 ; do ssh ${i} ldconfig ; done
```

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

## 1. Copy the Torque package to the nodes:

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

## 2. Install the Torque package:

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

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

## 2.1.5 Enabling Torque as a Service

**i** Enabling Torque as a service is optional. In order to run Torque as a service, you must enable `trqauthd`. See [2.2.4 Configuring trqauthd for Client Commands](#).

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/` or `contrib/systemd/` directory of the source package. To enable Torque as a service, run the following *as root* on the host for the appropriate Torque daemon:

```
> cp contrib/systemd/pbs_mom.service /usr/lib/systemd/pbs_server.service
> systemctl enable pbs_mom.service
> cp contrib/systemd/pbs_server.service /usr/lib/systemd/pbs_server.service
> systemctl enable pbs_server.service
```

**i** You will need to customize these scripts to match your system.

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

### Related Topics

- [2.2.4 Configuring trqauthd for Client Commands](#)

## 2.2 Initializing/Configuring Torque on the Server (pbs\_server)

The Torque server (`pbs_server`) contains all the information about a cluster. It knows about all of the MOM nodes in the cluster based on the information in the `TORQUE_HOME/server_priv/nodes` file (see [2.2.2 Configuring Torque on Compute Nodes](#)). It also maintains the status of each MOM node through updates from the MOMs in the cluster (see [pbsnodes](#)). All jobs are submitted via `qsub` to the server, which maintains a master database of all jobs and their states.

Schedulers such as Moab Workload Manager receive job, queue, and node information from `pbs_server` and submit all jobs to be run to `pbs_server`.

In this section:

- 2.2.1 Specifying Compute Nodes
- 2.2.2 Configuring Torque on Compute Nodes
- 2.2.3 Configuring Ports
- 2.2.4 Configuring `trqauthd` for Client Commands
- 2.2.5 Finalizing Configurations

## 2.2.1 Specifying Compute Nodes

The environment variable `TORQUE_HOME` holds the directory path where configuration files are stored. If you used the default locations during installation, you do not need to specify the `TORQUE_HOME` environment variable.

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.

**i** Only a root user can access the `server_priv` directory.

Syntax of nodes file:

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

- The `node-name` must match the hostname on the node itself, including whether it is fully qualified or shortened.

**i** You can specify a compute node's hostname by starting each `pbs_mom` with the `-H hostname` option, or by adding a line for `$mom_host` in `TORQUE_HOME/mom_priv/config` on the `pbs_mom` host(s). (You can run `hostname -f` to obtain a node's hostname.)

- 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 this 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 enables 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
...

```

## 2.2.2 Configuring Torque on Compute Nodes

If you are 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`:

```
$logevent      1039      # 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.

## 2.2.3 Configuring Ports

This topic provides information on configuring and managing the ports Torque uses for communication.

In this topic:

[2.2.3.A Configuring Torque Communication Ports](#)

[2.2.3.B Changing Default Ports](#)

### 2.2.3.A Configuring Torque Communication Ports

You can optionally configure the various ports that Torque uses for communication. Most ports can be configured multiple ways. Instructions for configuring each of the ports is provided below.

**i** If you are running PBS Professional on the same system, be aware that it uses the same environment variables and `/etc/services` entries.

### Configuring the `pbs_server` Listening Port

To configure the port the `pbs_server` listens on, follow *any* of these steps:

- Set an environment variable called `PBS_BATCH_SERVICE_PORT` to the port desired.
- Edit the `/etc/services` file and set `pbs port_num/tcp`.
- Start `pbs_server` with the `-p` option:
  - Edit `/etc/systemconfig/pbs_server`:
 

```
PBS_ARGS="-p"
```
  - Start the service:
 

```
systemctl start pbs_server.service
```
- Edit the `$PBS_HOME/server_name` file and change `server_name` to `server_name:<port_num>`.
- Start `pbs_server` with the `-H` option:
  - Edit `/etc/systemconfig/pbs_server`:
 

```
PBS_ARGS="-H"
```



- Start the service:

```
systemctl start pbs_server.service
```

## Configuring the pbs\_mom Listening Port

To configure the port the pbs\_mom listens on, follow *any* of these steps:

- Set an environment variable called PBS\_MOM\_SERVICE\_PORT to the port desired.
- Edit the /etc/services file and set pbs\_mom port\_num/tcp.
- Start pbs\_mom with the -M option:
  - Edit /etc/systemconfig/pbs\_mom:

```
PBS_ARGS="-M"
```

- Start the service:

```
systemctl start pbs_mom.service
```

- Edit the pbs\_server nodes file to add mom\_service\_port=port\_num.

## Configuring the Port pbs\_server Uses to Communicate with pbs\_mom

To configure the port the pbs\_server uses to communicate with pbs\_mom, follow *any* of these steps:

- Set an environment variable called PBS\_MOM\_SERVICE\_PORT to the port desired.
- Edit the /etc/services file and set pbs\_mom port\_num/tcp.
- Start pbs\_mom with the -M option:
  - Edit /etc/systemconfig/pbs\_mom:

```
PBS_ARGS="-M"
```

- Start the service:

```
systemctl start pbs_mom.service
```

## Configuring the Port pbs\_mom Uses to Communicate with pbs\_server

To configure the port the pbs\_mom uses to communicate with pbs\_server, follow *any* of these steps:

- Set an environment variable called PBS\_BATCH\_SERVICE\_PORT to the port desired.
- Edit the /etc/services file and set pbs port\_num/tcp.

- Start `pbs_mom` with the `-S` option:

- Edit `/etc/systemconfig/pbs_mom`:

```
PBS_ARGS="-S"
```

- Start the service:

```
systemctl start pbs_mom.service
```

- Edit the `nodes` file entry for that list: add `mom_service_port=port_num`.

## Configuring the Port Client Commands Use to Communicate with `pbs_server`

To configure the port client commands use to communicate with `pbs_server`, follow *any* of these steps:

- Edit the `/etc/services` file and set `pbs port_num/tcp`.
- Edit the `$PBS_HOME/server_name` file and change `server_name` to `server_name:<port_num>`

## Configuring the Port `trqauthd` Uses to Communicate with `pbs_server`

To configure the port `trqauthd` uses to communicate with `pbs_server`, edit the `$PBS_HOME/server_name` file and change `server_name` to `server_name:<port_num>`.

### 2.2.3.B Changing Default Ports

This section provides examples of changing the default ports (using non-standard ports).

#### MOM Service Port

The MOM service port is the port number on which MOMs are listening. This example shows how to change the default MOM service port (15002) to port 30001.

Do the following.

1. On the server, for the `server_priv/nodes` file, change the node entry:

```
nodename np=4 mom_service_port=30001
```

2. On the MOM, start `pbs_mom` with the `-M` option:

- Edit `/etc/systemconfig/pbs_mom`:

```
PBS_ARGS="-M"
```

- Start the service:

```
systemctl start pbs_mom.service
```

## Default Port on the Server

Do the following.

1. Set the  $\$(TORQUE\_HOME)/server\_name$  file:

```
hostname:newport
numa3.ac:45001
```

2. On the MOM, start `pbs_mom` with the `-S` option:

- Edit `/etc/systemconfig/pbs_mom`:

```
PBS_ARGS="-S"
```

- Start the service:

```
systemctl start pbs_mom.service
```

## MOM Manager Port

The MOM manager port tell MOMs which ports on which other MOMs are listening for MOM-to-MOM communication. This example shows how to change the default MOM manager port (15003) to port 30002.

Do the following.

1. On the server nodes file:

```
nodename np=4 mom_manager_port=30002
```

2. On the MOM, start `pbs_mom` with the `-R` option:

- Edit `/etc/systemconfig/pbs_mom`:

```
PBS_ARGS="-R"
```

- Start the service:

```
systemctl start pbs_mom.service
```

---

## Related Topics

- [A.5 pbs\\_server](#)
- [A.4 pbs\\_mom](#)

- [2.2.4 Configuring trqauthd for Client Commands](#)
- [Appendix A: Commands Overview](#)

## 2.2.4 Configuring trqauthd for Client Commands

`trqauthd` is a daemon used by Torque client utilities to authorize user connections to `pbs_server`. Once started, it remains resident. Torque client utilities then communicate with `trqauthd` on port 15005 on the loopback interface. It is multi-threaded and can handle large volumes of simultaneous requests.

### Running trqauthd

`trqauthd` must be run as root. It must also be running on any host where Torque client commands will execute.

By default, `trqauthd` is installed to `/usr/local/bin`.

**i** If you run `trqauthd` before starting `pbs_server`, you will receive a warning that no servers are available. To avoid this message, start `pbs_server` before running `trqauthd`.

`trqauthd` can be invoked directly from the command line or by the use of scripts that are located in the Torque source tree.

**Note:** The `systemd` scripts are located in the `contrib/systemd` directory.

There are two scripts for `trqauthd`:

Script	Description
<b>suse.trqauthd</b>	Used only for SUSE-based systems.
<b>trqauthd</b>	An example for other package managers (Red Hat, Scientific, CentOS, and Fedora are some common examples).

**i** You should edit these scripts to be sure they will work for your site.

Inside each of the scripts are the variables `PBS_DAEMON` and `PBS_HOME`. These two variables should be updated to match your Torque installation. `PBS_DAEMON` needs to point to the location of `trqauthd`. `PBS_HOME` needs to match your Torque installation.

Do the following.

1. Choose the script that matches your dist system and copy it to `/etc/init.d`. If needed, rename it to `trqauthd`.
2. Restart the service:

```
systemctl restart trqauthd.service
```

**i** If you receive an error that says "Could not open socket in trq\_simple\_connect. error 97", check your `/etc/hosts` file for multiple entries of a single host name pointing to the same IP address. Delete the duplicate(s), save the file, and launch `trqauthd` again.

## 2.2.5 Finalizing 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:

```
> systemctl start pbs_mom.service
```

- Server Node:

```
> systemctl restart pbs_server.service
```

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

## 2.3 Advanced Configuration

This section contains information about how you can customize the installation and configure the server to ensure that the server and nodes are communicating correctly.

In this section:

- [2.3.1 Customizing the Install](#)
- [2.3.2 Server Configuration](#)
- [2.3.3 Setting Up the MOM Hierarchy \(Optional\)](#)
- [2.3.4 Opening Ports in a Firewall](#)

### 2.3.5 Port Reference

## Related Topics

- [Appendix B: Server Parameters](#)

## 2.3.1 Customizing the Install

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

By default, only children MOM processes use `syslog`. To enable `syslog` for all of Torque, use `--enable-syslog`.


*Table 2-1: Optional Features*

Option	Description
<code>--disable-clients</code>	Directs Torque not to build and install the Torque client utilities such as <code>qsub</code> , <code>qstat</code> , <code>qdel</code> , etc.
<code>--disable-FEATURE</code>	Do not include <code>FEATURE</code> (same as <code>--enable-FEATURE=no</code> ).
<code>--disable-libtool-lock</code>	Avoid locking (might break parallel builds).
<code>--disable-mom</code>	Do not include the MOM daemon.
<code>--disable-mom-checkspool</code>	Don't check free space on spool directory and set an error.
<code>--disable-posixmemlock</code>	Disable the MOM's use of <code>mlockall</code> . Some versions of OSs seem to have buggy POSIX MEMLOCK.


Option	Description
<b>--disable-privports</b>	Disable the use of privileged ports for authentication. Some versions of OSX have a buggy <code>bind()</code> and cannot bind to privileged ports.
<b>--disable-qsub-keep-override</b>	Do not allow the <code>qsub -k</code> flag to override <code>-o -e</code> .
<b>--disable-server</b>	Do not include server and scheduler.
<b>--disable-shell-pipe</b>	Give the job script file as standard input to the shell instead of passing its name via a pipe.
<b>--disable-spool</b>	If disabled, Torque will create output and error files directly in <code>TORQUE_HOME/.pbs_spool</code> if it exists or in <code>TORQUE_HOME</code> otherwise. By default, Torque will spool files in <code>TORQUE_HOME/spool</code> and copy them to the users home directory when the job completes.
<b>--disable-xopen-networking</b>	With HPUX and GCC, don't force usage of <code>XOPEN</code> and <code>libxnet</code> .
<b>--enable-acct-x</b>	Enable adding x attributes to accounting log.
<b>--enable-array</b>	Setting this under IRIX enables the SGI Origin 2000 parallel support. Normally autodetected from the <code>/etc/config/array</code> file.


Option	Description
<b>--enable-autorun</b>	Turn on the AUTORUN_JOBS flag. When enabled, Torque runs the jobs as soon as they are submitted (destroys Moab compatibly). This option is not supported.
<b>--enable-blcr</b>	Enable BLCR support.
<b>--enable-cgroups</b>	<p>Enable cgroups. When cgroups are enabled, cpusets are handled by the cgroup cpuset subsystem. Requires --with-hwloc-path.</p> <div> <p><b>i</b> If you are building with cgroups enabled, you must have Boost version 1.41 or later.</p> <p><b>i</b> --enable-cgroups is not compatible with --enable-geometry-requests.</p> </div>
<b>--enable-cpuset</b>	Enable Linux 2.6 kernel cpusets.



Option	Description
	<div>  We recommend that you use <code>--enable-cgroups</code> instead of <code>--enable-cpuset</code>. <code>--enable-cpuset</code> is deprecated and no new features will be added to it. <code>--enable-cgroups</code> and <code>--enable-cpuset</code> are mutually exclusive. </div>
<code>--enable-debug</code>	Prints debug information to the console for <code>pbs_server</code> and <code>pbs_mom</code> while they are running. This is different than <code>--with-debug</code> , which will compile with debugging symbols.
<code>--enable-dependency-tracking</code>	Do not reject slow dependency extractors.
<code>--enable-fast-install[=PKGS]</code>	Optimize for fast installation [default=yes].
<code>--enable-FEATURE[=ARG]</code>	Include FEATURE [ARG=yes].
<code>--enable-filesync</code>	Open files with sync on each write operation. This has a negative impact on Torque performance. This is disabled by default.
<code>--enable-force-nofile</code>	Forces creation of nofile regardless of job submission parameters. Not on by default.


Option	Description
<code>--enable-gcc-warnings</code>	Enable gcc strictness and warnings. If using gcc, default is to error on any warning.
<code>--enable-geometry-requests</code>	<p>Torque is compiled to use <a href="#">procs_bitmap</a> during job submission.</p> <div> <p><b>i</b> When using <code>--enable-geometry-requests</code>, do not disable cpusets. Torque looks at the cpuset when killing jobs.</p> <p><b>i</b> <code>--enable-geometry-requests</code> is not compatible with <code>--enable-cgroups</code>.</p> </div>
<code>--enable-gui</code>	Include the GUI-clients.
<code>--enable-maintainer-mode</code>	This is for the <code>autoconf</code> utility and tells <code>autoconf</code> to enable so called rebuild rules. See <a href="#">maintainer mode</a> for more information.
<code>--enable-maxdefault</code>	Turn on the <code>RESOURCEMAXDEFAULT</code> flag.

Option	Description
	<div data-bbox="917 310 1250 1182">  Earlier versions of Torque 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). Now, the queue and server maximum values are no longer used as a value for missing settings. To re-enable this behavior, use <code>--enable-maxdefault</code>. </div>
<b><code>--enable-nochildsignal</code></b>	Turn on the <code>NO_SIGCHLD</code> flag.
<b><code>--enable-nodemask</code></b>	Enable nodemask-based scheduling on the Origin 2000.
<b><code>--enable-nvidia-gpus</code></b>	Include support for NVIDIA GPUs. See 'Scheduling GPUs' in the Accelerators chapter of the <i>Moab Workload Manager Administrator Guide</i> for setup details and options.
<b><code>--enable-plock-daemons [=ARG]</code></b>	Enable daemons to lock themselves into memory:

Option	Description
	logical-or of 1 for pbs_server, 2 for pbs_scheduler, 4 for pbs_mom (no argument means 7 for all three).
<b>--enable-pmix</b>	<p>Moms must be built with -enable-pmix to support jobs using PMIx. Binaries built with PMIx will also support jobs that are not using PMIx. However, if the moms are not built with-enable-pmix, the PMIx jobs will fail right after being launched.</p> <div>  The PMIx 1.0 API is supported except for registering and deregistering events. </div>
<b>--enable-quickcommit</b>	Turn on the QUICKCOMMIT flag. When enabled, adds a check to make sure the job is in an expected state and does some bookkeeping for array jobs. This option is not supported.
<b>--enable-shared[=PKGS]</b>	Build shared libraries [default=yes].
<b>--enable-shell-use-argv</b>	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.
<b>--enable-sp2</b>	Build PBS for an IBM SP2.


Option	Description
<code>--enable-static[=PKGS]</code>	Build static libraries [default=yes].
<code>--enable-syslog</code>	Enable (default) the use of <code>syslog</code> for error reporting.
<code>--enable-tcl-qstat</code>	Setting this builds <code>qstat</code> with Tcl interpreter features. This is enabled if Tcl is enabled.
<code>--enable-unixsockets</code>	Enable the use of UNIX Domain sockets for authentication.

Table 2-2: Optional Packages

Option	Description
<code>--with-blcr=DIR</code>	BLCR installation prefix.
<code>--with-blcr-include=DIR</code>	Include path for <code>libcr.h</code> .
<code>--with-blcr-lib=DIR</code>	Lib path for <code>libcr</code> .
<code>--with-blcr-bin=DIR</code>	Bin path for BLCR utilities.
<code>--with-boost-path=DIR</code>	<div>  Boost version 1.36.0 or later is supported. Red Hat-based systems come packaged with 1.53.0. </div> <p>Set the path to the Boost header files to be used during make. This option does not require Boost to be built or installed.</p> <p>The <code>--with-boost-path</code> value must be a directory containing a subdirectory called <code>boost</code> that contains the <code>boost.hpp</code> files.</p> <p>For example, if downloading the boost 1.55.0 source tarball to the adaptive user's home directory:</p> <pre>[adaptive]\$ cd ~ [adaptive]\$ wget https://sourceforge.net/projects/boost/files/boost/1.55.0/boost_1_55_0.tar.gz/download</pre>

Option	Description
	<pre>[adaptive]\$ tar xzf boost_1_55_0.tar.gz [adaptive]\$ ls boost_1_55_0 boost boost-build.jam ...</pre> <p>In this case use <code>--with-boost-path=/home/adaptive/boost_1_55_0</code> during configure. Another example would be to use an installed version of Boost. If the installed Boost header files exist in <code>/usr/include/boost/*.hpp</code>, use <code>--with-boost-path=/usr/include</code>.</p>
<code>--with-cpa-include=DIR</code>	Include path for <code>cpalib.h</code> .
<code>--with-cpa-lib=DIR</code>	Lib path for <code>libcpalib</code> .
<code>--with-debug=no</code>	Do not compile with debugging symbols.
<code>--with-default-server=HOSTNAME</code>	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.
<code>--with-environ=PATH</code>	Set the path containing the environment variables for the daemons. For SP2 and AIX systems, suggested setting it to <code>/etc/environment</code> . Defaults to the file <code>pbs_environment</code> in the server home. Relative paths are interpreted within the context of the server home.
<code>--with-gnu-ld</code>	Assume the C compiler uses GNU <code>ld</code> [default=no].
<code>--with-hwloc-path</code>	<p>Path for <code>hwloc</code> include and library files. Example:</p> <pre>./configure --with-hwloc-path=/usr/local/hwloc-1.9</pre> <p><i>Specifies that the include files are in <code>/usr/local/hwloc-1.9/include</code> and the libraries are in <code>/usr/local/hwloc-1.9/lib</code>.</i></p>
<code>--with-maildomain=MAILDOMAIN</code>	Override the default domain for outgoing mail messages (i.e., <code>user@maildomain</code> ). The default <code>maildomain</code> is the <code>hostname</code> where the job was submitted from.

Option	Description
<b>--with-modulefiles</b> <b>[=DIR]</b>	Use module files in specified directory [/etc/modulefiles].
<b>--with-momlogdir</b>	Use this directory for MOM logs.
<b>--with-momlogsuffix</b>	Use this suffix for MOM logs.
<b>--with-nvml-</b> <b>include=DIR</b>	Include path for <code>nvml.h</code> . See 'Scheduling GPUs' in the Accelerators chapter of the <i>Moab Workload Manager Administrator Guide</i> for setup details and options.
<b>--with-nvml-lib=DIR</b>	Library path for <code>libnvidia-ml.so</code> . See 'Scheduling GPUs' in the Accelerators chapter of the <i>Moab Workload Manager Administrator Guide</i> for setup details and options.
<b>--without-PACKAGE</b>	Do not use PACKAGE (same as <code>--with-PACKAGE=no</code> ).
<b>--without-readline</b>	Do not include readline support (default: included if found).
<b>--with-PACKAGE</b> <b>[=ARG]</b>	Use PACKAGE [ARG=yes].
<b>--with-pam=DIR</b>	Directory that holds the system PAM modules. Defaults to <code>/lib(64)/security</code> on Linux.
<b>--with-pic</b>	Try to use only PIC/non-PIC objects [default=use both].
<b>--with-qstatrc-</b> <b>file=FILE</b>	Set the name of the file that <code>qstat</code> will use if there is no <code>.qstatrc</code> 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 <code>qstatrc</code> (no dot) in the server home directory.
<b>--with-rcp</b>	One of <code>scp</code> , <code>rcp</code> , <code>mom_rcp</code> , or the full path of a remote file copy program. <code>scp</code> is the default if found; otherwise, <code>mom_rcp</code> is used. Some <code>rcp</code> programs don't always exit with valid error codes in case of failure. <code>mom_rcp</code> is a copy of BSD <code>rcp</code> included with this source that has correct error codes, but it is also old, unmaintained, and doesn't have large file support.
<b>--with-reserved-</b> <b>port-start=PORT</b>	Set the lower bound of the reserved port range that Torque will used when opening a reserved port. <code>PORT</code> must be between 144

Option	Description
	<p>and 823, inclusive.</p> <div>  <p>Setting this parameter reduces the number of privileged ports available to the system. This could affect performance, because it limits the number of concurrent reserved ports <code>pbs_server</code> can open.</p> </div>
<b>--with-sched=TYPE</b>	Sets the scheduler type. If TYPE is <code>c</code> , the scheduler will be written in C. If TYPE is <code>tcl</code> the server will use a Tcl based scheduler. If TYPE is <code>basl</code> , Torque will use the rule based scheduler. If TYPE is <code>no</code> , then no scheduling is done. <code>c</code> is the default.
<b>--with-sched-code=PATH</b>	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 <code>.basl</code> . It will be interpreted relative to <code>src/tree/src/schedulers.SCHD_TYPE/samples</code> . As an example, an appropriate BASL scheduler relative path would be <code>nas.basl</code> . The default scheduler code for 'C' schedulers is <code>fifo</code> .
<b>--with-scp</b>	SCP is the default remote copy protocol. See <b>--with-rcp</b> above if a different protocol is desired.
<b>--with-sendmail</b> <b>[=PATH_TO_EXECUTABLE]</b>	Sendmail executable to use. If <code>=PATH_TO_EXECUTABLE</code> is not specified or if <b>--with-sendmail</b> is not used at all, configure will attempt to find sendmail.
<b>--with-server-home=DIR</b>	Set the server <code>home/spool</code> directory for PBS use. Defaults to <code>/var/spool/torque</code> .
<b>--with-server-name-file=FILE</b>	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 <code>/var/spool/torque</code> or is set using the <b>--with-server-home</b> option to configure. If this option is not specified, the default name for this file will be set to <code>server_name</code> .
<b>--with-tcl</b>	Directory containing tcl configuration ( <code>tclConfig.sh</code> ).
<b>--with-tclattrsep=CHAR</b>	Set the Tcl attribute separator character this will default to <code>'</code> if unspecified.



Option	Description
<code>--with-tclinclude</code>	Directory containing the public Tcl header files.
<code>--with-tclx</code>	Directory containing tclx configuration ( <code>tclxConfig.sh</code> ).
<code>--with-tk</code>	Directory containing tk configuration ( <code>tkConfig.sh</code> ).
<code>--with-tkinclude</code>	Directory containing the public Tk header files.
<code>--with-tkx</code>	Directory containing tkx configuration ( <code>tkxConfig.sh</code> ).
<code>--with-xauth=PATH</code>	Specify path to xauth program.

### 2.3.1.A HAVE\_WORDEXP

`Wordxp()` performs a shell-like expansion, including environment variables. By default, `HAVE_WORDEXP` is set to 1 in `src/pbs_config.h`. If set to 1, will limit the characters that can be used in a job name to those allowed for a file in the current environment, such as BASH. If set to 0, any valid character for the file system can be used.

If a user would like to disable this feature by setting `HAVE_WORDEXP` to 0 in `src/include/pbs_config.h`, it is important to note that the error and the output file names will not expand environment variables, including `$PBS_JOBID`. The other important consideration is that characters that BASH dislikes, such as `()`, will not be allowed in the output and error file names for jobs by default.

---

#### Related Topics

- [2.3.2 Server Configuration](#)

## 2.3.2 Server Configuration

This topic contains information and instructions to configure your server.

In this topic:

[2.3.2.A Server Configuration Overview](#)

[2.3.2.B Name Service Configuration](#)

[2.3.2.C Configuring Job Submission Hosts](#)

[2.3.2.D Configuring Torque on a Multi-Homed Server](#)

[2.3.2.E Architecture Specific Notes](#)

[2.3.2.F Specifying Non-Root Administrators](#)

[2.3.2.G Setting Up Email](#)

[2.3.2.H Using MUNGE Authentication](#)

### 2.3.2.A 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.

### 2.3.2.B 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.

### 2.3.2.C 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
.....
```

Either of the hostname or username fields can be replaced with a wildcard symbol (+). The (+) can be used as a stand-alone wildcard but not connected to a username or hostname (e.g., +node01 or +user01). However, a (-) can be used in that manner to specifically exclude a user.



Following the Linux man page instructions for hosts.equiv may result in a failure. You cannot precede the user or hostname with a (+). To clarify, node1 +user1 will not work and user1 will not be able to submit jobs.

For example, the following lines will not work or will not have the desired effect:

```
+node02 user1
node02 +user1
```

These lines will work:

```
node03 +
+ jsmith
node04 -tjones
```

The most restrictive rules must precede more permissive rules. For example, to restrict user tsmith but allow all others, follow this format:

```
node01 -tsmith
node01 +
```

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 (for details, see [Appendix N: Queue Attributes](#)).

## Using the 'submit\_hosts' Service Parameter

Trusted submit host access can 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`.

### 2.3.2.D Configuring Torque on a Multi-Homed Server

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

### 2.3.2.E Architecture 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.

### 2.3.2.F 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.

**i** 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.

### 2.3.2.G Setting Up Email

Moab relies on emails from Torque about job events. To set up email, do the following.

1. Specify the location of the `sendmail` executable. You can do this using the `sendmail_path` server attribute:

```
qmgr -c 'set server sendmail_path = <path_to_executable>'
```

If this server option is not set, you can set a default location during the build:

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

If a location for the sendmail executable is not specified, Torque will attempt to find it when you run configure. If you installed Torque using RPMs from Adaptive Computing, the default path will be `/usr/sbin/sendmail`.

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 emails on job aborts. Each user can select which kind of emails to receive by using the `qsub -m` option when submitting the job. If you want to dictate when each user should receive emails, use a submit filter [for details, see the appendix [Job Submission Filter \(qsub Wrapper\)](#)].

### 2.3.2.H Using MUNGE Authentication

**i** The same version of MUNGE must be installed on all of your Torque Hosts (Server, Client, MOM).

MUNGE is an authentication service that creates and validates user credentials. It was developed by Lawrence Livermore National Laboratory (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 [MUNGE Uid 'N' Gid Emporium](#).

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 MUNGED 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* 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.

---

## Related Topics

- [2.3.3 Setting Up the MOM Hierarchy \(Optional\)](#)

## 2.3.3 Setting Up the MOM Hierarchy (Optional)

**i** Mom hierarchy is designed for large systems to configure how information is passed directly to the *pbs\_server*.

The MOM hierarchy enables you to override the compute nodes' default behavior of reporting status updates directly to the *pbs\_server*. Instead, you configure compute nodes so that each node sends its status update information to another compute node. The compute nodes pass the information up a tree or hierarchy until eventually the information reaches a node that will pass the information directly to *pbs\_server*. This can significantly reduce network traffic and ease the load on the *pbs\_server* in a large system.

**i** We recommend approximately 25 nodes per path. Numbers larger than this may reduce the system performance.

In this topic:

[2.3.3.A MOM Hierarchy Example](#)

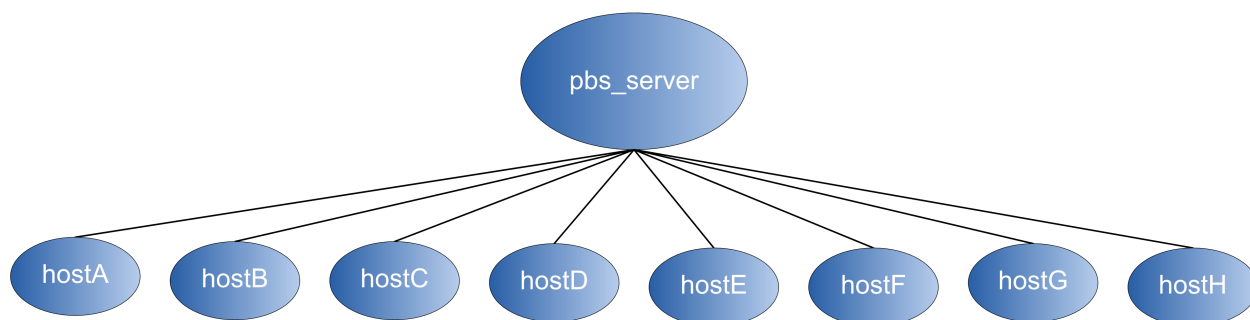
[2.3.3.B Setting Up the MOM Hierarchy](#)

[2.3.3.C Putting the MOM Hierarchy on the MOMs](#)

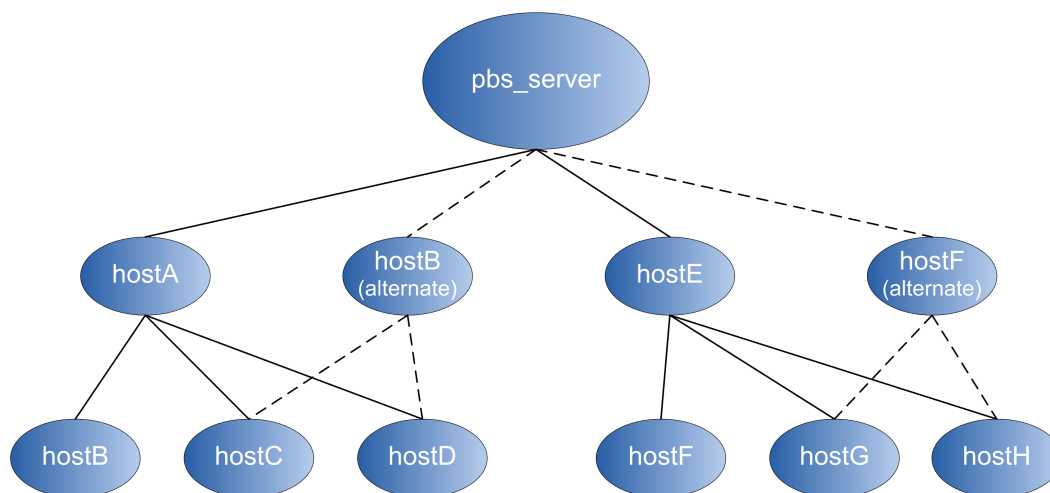
### 2.3.3.A MOM Hierarchy Example

The following example illustrates how information is passed to the *pbs\_server* without and with mom\_hierarchy:

Without mom\_hierarchy



With mom\_hierarchy



**i** The dotted lines indicates an alternate path if the hierarchy-designated node goes down.

The following is the `mom_hierarchy_file` for the `mom_hierarchy` example:

```
<path>
  <level>hostA,hostB</level>
  <level>hostB,hostC,hostD</level>
</path>
<path>
  <level>hostE,hostF</level>
  <level>hostE,hostF,hostG</level>
</path>
```

### 2.3.3.B Setting Up the MOM Hierarchy

The name of the file that contains the configuration information is named `mom_hierarchy`. By default, it is located in the `/var/spool/torque/server_priv` directory. The file uses syntax similar to XML:

```
<path>
  <level>comma-separated node list</level>
  <level>comma-separated node list</level>
  ...
</path>
...
```

The `<path></path>` tag pair identifies a group of compute nodes. The `<level></level>` tag pair contains a comma-separated list of compute node names listed by their hostnames. Multiple paths can be defined with multiple levels within each path.

Within a `<path></path>` tag pair, the levels define the hierarchy. All nodes in the top level communicate directly with the server. All nodes in lower levels communicate to the first available node in the level directly above it. If the first node in the upper level goes down, the nodes in the subordinate level will then communicate to the next node in the upper level. If no nodes are available in an upper level then the node will communicate directly to the server.

**i** When setting up the MOM hierarchy, you must open port 15003 for communication from `pbs_server` to `pbs_mom`.

If an upper level node has gone down and then becomes available, the lower level nodes will eventually find that the node is available and start sending their updates to that node.



**i** If you want to specify MOMs on a different port than the default, you must list the node in the form: `hostname:mom_manager_port`. For example:

```
<path>
  <level>hostname:mom_manager_port,... </level>
  ...
</path>
...
```

### 2.3.3.C Putting the MOM Hierarchy on the MOMs

You can put the MOM hierarchy file directly on the MOMs. The default location is `/var/spool/torque/mom_priv/mom_hierarchy`. This way, the *pbs\_server* doesn't have to send the hierarchy to all the MOMs during each *pbs\_server* startup. The hierarchy file still has to exist on the *pbs\_server* and if the file versions conflict, the *pbs\_server* version overwrites the local MOM file. When using a global file system accessible from both the MOMs and the *pbs\_server*, it is recommended that the hierarchy file be symbolically linked to the MOMs.

Once the hierarchy file exists on the MOMs, start *pbs\_server* with the `-n` option, which tells *pbs\_server* to not send the hierarchy file on startup. Instead, *pbs\_server* waits until a MOM requests it.

## 2.3.4 Opening Ports in a Firewall

If your site is running firewall software on its hosts, you will need to configure the firewall to allow connections to the products in your installation.

This topic provides an example and general instructions for how to open ports in your firewall. See [2.3.5 Port Reference](#) for the actual port numbers for the various products.

In this topic:

[2.3.4.A Red Hat-Based Systems](#)

[2.3.4.B SUSE-Based Systems](#)

### 2.3.4.A Red Hat-Based Systems

Red Hat-based systems use `firewalld` as the default firewall software. If you use different firewall software, refer to your firewall documentation for opening ports in your firewall.

The following is an example of adding port 1234 when using `firewalld`:

```
[root]# firewall-cmd --add-port=1234/tcp --permanent
[root]# firewall-cmd --reload
```

### 2.3.4.B SUSE-Based Systems

SUSE-based systems use SuSEfirewall2 as the default firewall software. If you use different firewall software, refer to your firewall documentation for opening ports in your firewall.

The following is an example of adding port 1234 when using SuSEfirewall2:

```
[root]# vi /etc/sysconfig/SuSEfirewall2
FW_SERVICES_EXT_TCP="1234"
[root]# systemctl restart SuSEfirewall2
```

### 2.3.5 Port Reference

The following table contains the port numbers for the various products in the Moab HPC Suite.

#### Adaptive Computing Local RPM Repository

Location	Port	Function	When Needed
Deployment Host	80 443	Adaptive Computing Local RPM repository	The duration of the install when using RPM installation method.

#### RLM Server

Location	Port	Function	When Needed
RLM Server Host	5053	RLM Server Port	Always
RLM Server Host	5054	RLM Web Interface Port	Always
RLM Server Host	57889	Remote Visualization Port	If Remote Visualization is part of your configuration.

Location	Port	Function	When Needed
RLM Server Host	5135	ISV adaptiveco Port (for the Adaptive license-enabled products)	For Moab Workload Manager <i>and</i> if Nitro is part of your configuration.

## Torque Resource Manager

Location	Port	Function	When Needed
Torque Server Host	15001	Torque Client and MOM communication to Torque Server	Always
Torque MOM Host (Compute Nodes)	15002	Torque Server communication to Torque MOMs	Always
Torque MOM Host (Compute Nodes)	15003	Torque MOM communication to other Torque MOMs	Always

## Moab Workload Manager

Location	Port	Function	When Needed
Moab Server Host	42559	Moab Server Port	If you intend to run client commands on a host different from the Moab Server Host <i>or</i> if you will be using Moab in a grid

## Moab Accounting Manager

Location	Port	Function	When Needed
MAM Server Host	7112	MAM Server Port	If you will be installing the MAM Server on a different host from where you installed the Moab Server <i>or</i> you will be installing the MAM Clients on other hosts
MAM GUI Host	443	HTTPS Port	If using the MAM GUI
MAM Web Services Host	443	HTTPS Port	If using MAM Web Services
MAM Database Host	5432	MAM	If you will be installing the MAM

Location	Port	Function	When Needed
		PostgreSQL Server Port	Database on a different host from the MAM Server

## Moab Web Services

Location	Port	Function	When Needed
MWS Server Host	8080	Tomcat Server Port	Always
MWS Database Host	27017	MWS MongoDB Server Port	If you will be installing the MWS Database on a different host from the MWS Server

## Moab Insight

Location	Port	Function	When Needed
Insight Server Host	5568	Insight Server Port	Always
Moab MongoDB Database Host	27017	Moab MongoDB Server Port	Always
Moab Server Host	5574	Moab Data Port	Always
Moab Server Host	5575	Moab Reliability Port	Always

## Moab Viewpoint

Location	Port	Function	When Needed
Viewpoint Server Host	8081	Viewpoint Web Server Port	Always
Moab Server Host	8443	Viewpoint File Manager Port	Always
Viewpoint Database Host	5432	Viewpoint PostgreSQL Database Port	If you will be installing the Viewpoint Database on a different host from the Viewpoint Server

## Remote Visualization

Location	Port	Function	When Needed
Remote Visualization Server Host (also known as the Gateway Server)	3443	FastX Web Server Port	Always
Remote Visualization Session Server Host (Torque MOM Host)	Add ports as required, for example, TCP: 3443, 6000-6005, 16001, 35091 UDP: 117	Session Server Ports	Ports 16001 and 35091 are <i>only</i> needed when using gnome

## Nitro

**i** The listed ports are for configurations that have only one coordinator. If multiple coordinators are run on a single compute host, then sets of ports (range of 4) must be opened for the number of expected simultaneous coordinators.

Location	Port	Function	When Needed
Compute Hosts (Nitro Coordinator)	47000	Coordinator/Worker communication	Always
Compute Hosts (Nitro Coordinator)	47001	Coordinator PUB/SUB channel - publishes status information	Always
Compute Hosts (Nitro Coordinator)	47002	Reserved for future functionality	
Compute Hosts (Nitro Coordinator)	47003	API communication channel	Always

## Nitro Web Services

Location	Port	Function	When Needed
Nitro Web	9443	Tornado Web	Always

Location	Port	Function	When Needed
Services Host		Port	
Nitro Web Services Host	47100	ZMQ Port	Always
Nitro Web Services Database Host	27017	Nitro Web Services MongoDB Server Port	If you will be installing the Nitro Web Services Database on a different host from Nitro Web Services

## Reporting

Suggested Host	Service	Port	Function	When Needed
Reporting Master	HDFS name node	8020	HDFS communication	Always
Reporting Master	HDFS name node	50070	HDFS web interface	Always
Reporting Master	Spark Master	6066, 7077	Spark communication	Always
Reporting Master	Spark Master	8082	Spark Master web interface	Always
Reporting Master	Apache Kafka	9092	Kafka communication	Always
Reporting Master	Apache Zookeeper	2181	Zookeeper communication with Kafka and Drill	Always
Insight Server	Apache Drill	8047	Drill HTTP interface	Always
Reporting Worker	HDFS data node	50075, 50010, 50020	HDFS communication	Always
Reporting Worker	Spark Worker	4040	Spark communication	Always

Suggested Host	Service	Port	Function	When Needed
Reporting Worker	Spark worker	8083	Spark worker web interface	Always
MWS Host	Tomcat	8080	Reporting Web Services HTTP interface	Always
MWS Host	MongoDB	27017	MongoDB communication	Always

## 2.4 Manual Setup of Initial Server Configuration

On a new installation of Torque, the server database must be initialized using the command `pbs_server -t create`. This command creates a file in `$TORQUE_HOME/server_priv` named `serverdb`, which contains the server configuration information.

The following output from `qmgr` shows the base configuration created by the command `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 node_check_rate = 150
set server tcp_timeout = 6
```

This is a bare minimum configuration and it is not very useful. By using `qmgr`, the server configuration can be modified to set up Torque to do useful work. The following `qmgr` commands will create a queue and enable the server to accept and run jobs. These commands must be executed by root.

```
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"
```

**i** When Torque reports a new queue to Moab a class of the same name is automatically applied to all nodes.

In this example, the configuration database is initialized and the scheduling interface is activated using (`scheduling=true`). This option allows the scheduler to receive job and node events, which allow it to be more responsive (see the server parameter [scheduling](#) for more information). The next command 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](#) main page.

---

## Related Topics

- [2.1 Torque Installation Overview](#)

## 2.5 Server Node File Configuration

This section contains information about configuring server node files. It explains how to specify node virtual processor counts and GPU counts, as well as how to specify node features or properties.

In this section:

- [2.5.1 Basic Node Specification](#)
- [2.5.2 Specifying Virtual Processor Count for a Node](#)
- [2.5.3 Specifying GPU Count for a Node](#)
- [2.5.4 Specifying Node Features \(Node Properties\)](#)

---

## Related Topics

- [2.1 Torque Installation Overview](#)
- [Appendix B: Server Parameters](#)
- 'Node Features/Node Properties' in the *Moab Workload Manager Administrator Guide*



## 2.5.1 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
```

**i** The `server nodes` file also displays the parameters applied to the node. See [4.1 Adding Nodes](#) for more information on the parameters.

## 2.5.2 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 that represents available 'execution slots' for the node. The value used is determined by the admin based on hardware, system, and site criteria.

The following example shows how to set the `np` value in the `nodes` file. In this example, we are assuming that `node001` and `node002` have four physical cores. The admin wants the value of `np` for `node001` to reflect that it has four cores. 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=4
node002 np=12
...
```

## 2.5.3 Specifying GPU Count for a Node

**i** This section describes a rudimentary method for configuring GPUs manually. Administrators can configure the MOMs to automatically detect the number of NVIDIA GPUs and get detailed GPU reporting on each node (the recommended method). Combining this with cgroups will also prevent unauthorized access to resources. See 'Scheduling GPUs' in the *Moab Workload Manager Administrator Guide* for details on this automated method.

When using this method, `pbs_server` automatically appends `gpus=<count>` to the end of the line in `TORQUE_HOME/server_priv/nodes` for any node with a GPU, overriding any such manual configuration.

To manually set the number of GPUs on a node, use the `gpus` attribute in the nodes file. The value of GPUs is determined by the admin based on hardware, system, and site criteria.

The following example shows how to set the GPU value in the nodes file. In the example, we assume `node01` and `node002` each have two physical GPUs. The admin wants the value of `node001` to reflect the physical GPUs available on that system and adds `gpus=2` to the nodes file entry for `node001`. 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=2
node002 gpus=4
...
```

## 2.5.4 Specifying Node Features (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. For example:

```
qsub -l nodes=1:bigmem+1:fast job.sh
```

This job submission will look for a node with the bigmem feature (node002) and a node with the fast feature (either node001 or node002).

## 2.6 Testing Server Configuration

If you have initialized Torque using the `torque.setup` script or started Torque using `pbs_server -t create` and `pbs_server` is still running, terminate the server by calling `qterm`. Next, start `pbs_server` again without the `-t create` arguments. Follow the script below to verify your server configuration. The output for the examples below is based on the nodes file example in [2.4 Manual Setup of Initial Server Configuration](#) and [2.5.4 Specifying Node Features \(Node Properties\)](#).

```
# verify all queues are properly configured
> qstat -q

server:kmn

Queue      Memory      CPU Time      Walltime      Node      Run      Que      Lm      State
-----
batch      --           --           --           --           0       0       --      ER
                                0       0

# view additional server configuration
> 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 = user1@kmn
set server operators = user1@kmn
set server default_queue = batch
set server log_events = 511
set server mail_from = adm
set server node_check_rate = 150
set server tcp_timeout = 300
set server job_stat_rate = 45
set server poll_jobs = True
set server mom_job_sync = True
set server keep_completed = 300
set server next_job_number = 0
```

```
# verify all nodes are correctly reporting
> pbsnodes -a
node001
  state=free
  np=2
  properties=bigmem,fast,ia64,smp
  ntype=cluster

status=rectime=1328810402,varattr=,jobs=,state=free,netload=6814326158,gres=,loadave=0
.21,ncpus=6,physmem=8193724kb,
availmem=13922548kb,totmem=16581304kb,idletime=3,nusers=3,nsessions=18,sessions=1876
1120 1912 1926 1937 1951 2019 2057 28399 2126 2140 2323 5419 17948 19356 27726 22254
29569,uname=Linux kmn 2.6.38-11-generic #48-Ubuntu SMP Fri Jul 29 19:02:55 UTC 2025
x86_64,opsys=linux
  mom_service_port = 15002
  mom_manager_port = 15003
  gpus = 0
# submit a basic job - DO NOT RUN AS ROOT
> su - testuser
> echo "sleep 30" | qsub

# verify jobs display
> qstat
```

Job id	Name	User	Time Use	S	Queue
0.kmn	STDIN	knielson	0	Q	batch

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). See [Chapter 6: Integrating Schedulers for Torque](#) for details on setting up an advanced scheduler.

## Related Topics

- [2.1 Torque Installation Overview](#)

## 2.7 Configuring Torque for NUMA Systems

Torque supports these two types of Non-Uniform Memory Architecture (NUMA) systems:

- **NUMA-Aware** – Supports multi-req jobs and jobs that span hosts. Requires the `--enable-cgroups` configuration command to support cgroups. See [2.7.1 Torque NUMA-Aware Configuration](#) for instructions and additional information.
- **NUMA-Support** – *Only* for large-scale SLES systems (SGI Altix and UV hardware). Requires the `--enable-numa-support` configuration command. See [2.7.2 Torque NUMA-Support Configuration](#) for instructions and additional information.



Torque cannot be configured for both systems at the same.

In this section:

- [2.7.1 Torque NUMA-Aware Configuration](#)
- [2.7.2 Torque NUMA-Support Configuration](#)

## 2.7.1 Torque NUMA-Aware Configuration

This topic provides instructions for enabling NUMA-aware, including cgroups. For instructions on NUMA-support configurations, see [2.7.2 Torque NUMA-Support Configuration](#). This topic assumes you have a basic understanding of cgroups. See [Red Hat Resource Management Guide](#) or [cgroups on kernel.org](#) for basic information on cgroups.

In this topic:

- [2.7.1.A About cgroups](#)
- [2.7.1.B Prerequisites](#)
- [2.7.1.C Installation Instructions](#)
- [2.7.1.D Multiple cgroup Directory Configuration](#)

### 2.7.1.A About cgroups

Torque uses cgroups to better manage CPU and memory accounting, memory enforcement, cpuset management, and binding jobs to devices such as MICs and GPUs.

 Be aware of the following:

- cgroups is incompatible with (and supersedes) cpuset support (`--enable-cpuset` and `--enable-geometry-requests`). Configuring with `--enable-cgroups` overrides these other options.
- If you are building with cgroups enabled, you must have boost version 1.41 or later.
- The `pbs_mom` daemon is the binary that interacts cgroups, but both the server and the MOM must be built with `--enable-cgroups` to understand all of the new structures.

### 2.7.1.B Prerequisites

1. Install the prerequisites found in [2.1.3 Installing Torque Resource Manager](#).

### 2.7.1.C Installation Instructions

Do the following.

1. Install the libcgroup package.

**i** Red Hat-based Systems must use libcgroup version 0.40.rc1-16.el6 or later; SUSE-based systems need to use a comparative libcgroup version.

- Red Hat-based systems:

```
yum install libcgroup-tools
```

- SUSE-based systems:

```
zypper install libcgroup-tools
```

2. Enable Torque to access cgroups:

```
$ ./configure --enable-cgroups --with-hwloc-path=/usr/local
```

### 2.7.1.D Multiple cgroup Directory Configuration

If your system has more than one cgroup directory configured, you must create the `trq-cgroup-paths` file in the `$TORQUE_HOME` directory. This file has a list of the cgroup subsystems and the mount points for each subsystem in the syntax of `<subsystem> <mount point>`.

All five subsystems used by `pbs_mom` must be in the `trq-cgroup-paths` file. In the example that follows, a directory exists at `/cgroup` with subdirectories for each subsystem. Torque uses this file first to configure where it will look for cgroups.

```
cpuset    /cgroup/cpuset
cpuacct   /cgroup/cpuacct
cpu       /cgroup/cpu
memory    /cgroup/memory
devices   /cgroup/devices
```

## 2.7.2 Torque NUMA-Support Configuration

**i** This topic provides instructions for enabling NUMA-support on large-scale SLES systems using SGI Altix and UV hardware. For instructions on enabling NUMA-aware, see [2.7.1 Torque NUMA-Aware Configuration](#).

Perform the following steps:

1. [Configure Torque for NUMA-Support](#)
2. [Create the mom.layout File](#)
3. [Configure the server\\_priv/nodes File](#)
4. [Limit Memory Resources \(Optional\)](#)

### 2.7.2.A Configure Torque for 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-numa-support` option as indicated in the following example:

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

 Don't use MOM hierarchy with NUMA.

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 instance 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 admin wants Torque to view the system. Each line in *mom.layout* equates to a node in the cluster and is referred to as a NUMA node.

### 2.7.2.B Create the mom.layout File

This section provides instructions to create the *mom.layout* file.

Do *one* of the following:

- [Automatically Create mom.layout \(Recommended\)](#)
- [Manually Create mom.layout](#)

#### Automatically Create mom.layout (Recommended)

A perl script named *mom\_gencfg* is provided in the *contrib/* directory that generates the *mom.layout* file for you. The script can be customized by setting a few variables in it.

To automatically create the `mom.layout` file, do the following:

1. Verify `hwloc` library and corresponding `hwloc-devel` package are installed. See [2.1.3 Installing Torque Resource Manager](#) for more information.
2. Install `Sys::Hwloc` from CPAN.
3. Verify `$PBS_HOME` is set to the proper value.
4. Update the variables in the 'Config Definitions' section of the script. Especially update `firstNodeId` and `nodesPerBoard` if desired. The `firstNodeId` variable should be set above 0 if you have a root cpuset that you want to exclude and the `nodesPerBoard` variable is the number of NUMA nodes per board. Each node is defined in `/sys/devices/system/node`, in a subdirectory `node<node index>`.
5. Back up your current file in case a variable is set incorrectly or neglected.
6. Run the script:

```
$ ./mom_gencfg
```

## Manually Create mom.layout

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	256	18432
6	r001i01b00	01	00	32	6	46	2666	32d/32i	256	18432



```

 7 r001i01b00    01    02    36    6    46  2666 32d/32i    256  18432
 8 r001i01b00    01    03    38    6    46  2666 32d/32i    256  18432
 9 r001i01b00    01    08    48    6    46  2666 32d/32i    256  18432
10 r001i01b00    01    09    50    6    46  2666 32d/32i    256  18432
11 r001i01b00    01    11    54    6    46  2666 32d/32i    256  18432
12 r001i01b01    02    00    64    6    46  2666 32d/32i    256  18432
13 r001i01b01    02    02    68    6    46  2666 32d/32i    256  18432
14 r001i01b01    02    03    70    6    46  2666 32d/32i    256  18432

```

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 are local to the memory of node board 0, and the memory and cpus for that node are specified in the layout file by saying `nodes=0`. The `cpulist` for node board 1 shows 6-11 and memory node index 1. To specify this, simply write `nodes=1`. Repeat this for all twelve node boards and create the following `mom.layout` file for the 72 CPU system.

```
nodes=0
nodes=1
nodes=2
nodes=3
nodes=4
nodes=5
nodes=6
nodes=7
nodes=8
nodes=9
nodes=10
nodes=11
```

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:

```
nodes=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.

### 2.7.2.C Configure the `server_priv/nodes` File

The *pbs\_server* requires awareness of how the MOM is reporting nodes since there is only one MOM daemon and multiple MOM nodes.

You need to configure the `server_priv/nodes` file with the `num_node_boards` and `numa_board_str` attributes. The attribute `num_node_boards` tells *pbs\_server* how many numa nodes are reported by the MOM.

The following is an example of how to configure the nodes file with `num_node_boards`:

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

In this example, 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_node_boards` (12) and determines there are 6 CPUs per NUMA node.

The previous example showed that the NUMA system is uniform in its configuration of CPUs per node board. However, 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_board_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_board_str=6,8,12
```

In this example, *pbs\_server* knows it has 3 MOM nodes and the nodes have 6, 8, and 12 CPUs respectively. Notice that the attribute *np* is not used. The *np* attribute is ignored because the number of CPUs per node is expressly given.

### 2.7.2.D Limit Memory Resources (Optional)

Torque can better enforce memory limits with the use of the *memacctd* utility. The *memacctd* utility is a daemon that caches memory footprints when it is queried. When configured to use the memory monitor, Torque queries *memacctd*.

**i** The *memacctd* utility is provided by SGI for SLES systems only. It is up to the user to make sure *memacctd* is installed.

To configure Torque to use *memacctd* for memory enforcement, do the following:

1. Start *memacctd* as instructed by SGI.
2. Reconfigure Torque with `--enable-memacct`. This will link in the necessary library when Torque is recompiled.
3. Recompile and reinstall Torque.
4. Restart all MOM nodes.

**i** You use the *qsub* filter to include a default memory limit for all jobs that are not submitted with *memory limit*.

## 2.8 Torque Multi-MOM

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.

In this section:

## 2.8.1 Multi-MOM Configuration

### 2.8.2 Stopping `pbs_mom` in Multi-MOM Mode

## 2.8.1 Multi-MOM Configuration

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

1. [Configure `server\_priv/nodes`](#)
2. [Edit the `/etc/hosts` File](#)
3. [Start `pbs\_mom` with Multi-MOM Options](#)

### 2.8.1.A 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.

### 2.8.1.B Edit the `/etc/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.

### 2.8.1.C Start `pbs_mom` with Multi-MOM Options

To start multiple instances of `pbs_mom` on the same node, use the following syntax (see [A.4 pbs\\_mom](#) for details):

```
pbs_mom -m -M <port value of MOM_service_port> -R <port value of MOM_manager_port> -A
<name of MOM alias>
```

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 -A hosta-1
# pbs_mom -m -M 31001 -R 31002 -A hosta-2
# pbs_mom -m -M 32001 -R 32002 -A hosta-3
# 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.

### 2.8.2 Stopping `pbs_mom` in Multi-MOM Mode

Terminate `pbs_mom` by using the `momctl -s` command (for details, see [momctl](#)). 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 <port value of MOM_manager_port>
```

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

## 2.9 Supporting MIG Devices in Torque

This section describes how to use the MIG features on a MIG capable NVIDIA GPU.

In this section:

### 2.9.1 Requirements

### 2.9.2 Functionality

### 2.9.3 Limitations

## 2.9.1 Requirements

The configuration required for Multi-Instance GPU (MIG) capabilities include `hwloc` installed along with the following configuration option flags:

- `--enable-cgroups`
- `--with-hwloc-path`
- `--enable-nvidia-gpus`
- `--with-nvml-include`
- `--with-nvml-lib`

## 2.9.2 Functionality

The partitioned MIG instances will be enumerated as real GPU instances. Job submission resource request limitations are subject to current NVIDIA/CUDA driver limitations described below. A job resource request is performed identically to requesting a normal GPU resource.

Torque recognizes MIG instances as real GPUs, therefore, the `PBS_MOM` service must be stopped on a node when enabling or disabling MIG mode on a MIG capable GPU and then restarted. The `PBS_MOM` service must also be stopped and restarted whenever changing the configuration of MIG partitions on a node.

## 2.9.3 Limitations

Due to current NVIDIA driver limitations, no GPU to GPU P2P (either PCIe or NVLink) is supported. With CUDA 11, only enumeration of a single MIG instance is supported. See [MIG User Guide](#) for more information.

This means that when MIG instances are enabled on a GPU, only one GPU should be requested per job. This may change in future CUDA/NVIDIA driver versions.

## Chapter 3: Submitting and Managing Jobs

This section contains information about how you can submit and manage jobs with Torque.

In this chapter:

- [3.1 Job Submission](#)
- [3.2 Monitoring Jobs](#)
- [3.3 Canceling Jobs](#)
- [3.4 Job Preemption](#)
- [3.5 Keeping Completed Jobs](#)
- [3.6 Job Checkpoint and Restart](#)
- [3.7 Job Exit Status](#)
- [3.8 Torque Process Tracking](#)
- [3.9 Large Job Arrays](#)

### 3.1 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 can be specified as a filename on the `qsub` command line or can be entered via STDIN:

- The PBS command file does not need to be executable.
- The PBS command file can 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.
- PBS directives should be declared first in the job script:

```
#PBS -S /bin/bash
#PBS -m abe
#PBS -M <yourEmail@company.com>
echo sleep 300
```

This is an example of properly declared PBS directives.

```
#PBS -S /bin/bash
SOMEVARIABLE=42
#PBS -m abe
#PBS -M <yourEmail@company.com>
echo sleep 300
```

This is an example of improperly declared PBS directives. PBS directives below "SOMEVARIABLE=42" are ignored.

**i** 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 [2.3.2 Server Configuration](#).

**i** Earlier versions of Torque 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).

Now, the queue and server maximum values are no longer used as a value for missing settings.

In this section:

- [3.1.1 Multiple Job Submission](#)
- [3.1.2 Managing Multi-Node Jobs](#)
- [3.1.3 Requesting Resources](#)
- [3.1.4 Requesting NUMA-Aware Resources](#)
- [3.1.5 Requesting Generic Resources](#)
- [3.1.6 Requesting Floating Resources](#)
- [3.1.7 Requesting Other Resources](#)
- [3.1.8 Exported Batch Environment Variables](#)
- [3.1.9 Enabling Trusted Submit Hosts](#)
- [3.1.10 Example Submit Scripts](#)



## Related Topics

- [Maui Documentation](#)
- [Appendix J: Job Submission Filter \(qsub Wrapper\)](#) – Allow local checking and modification of submitted job

### 3.1.1 Multiple Job Submission

Sometimes users 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 enables the creation of multiple jobs with one *qsub* command. Additionally, this feature includes a job naming convention that enables users to reference the entire set of jobs as a unit, or to reference one particular job from the set.

In this topic:

[3.1.1.A Submitting Job Arrays](#)

[3.1.1.B Slot Limit](#)

#### 3.1.1.A Submitting Job Arrays

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.

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

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

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 enables 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
```

```

> 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 can 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
```

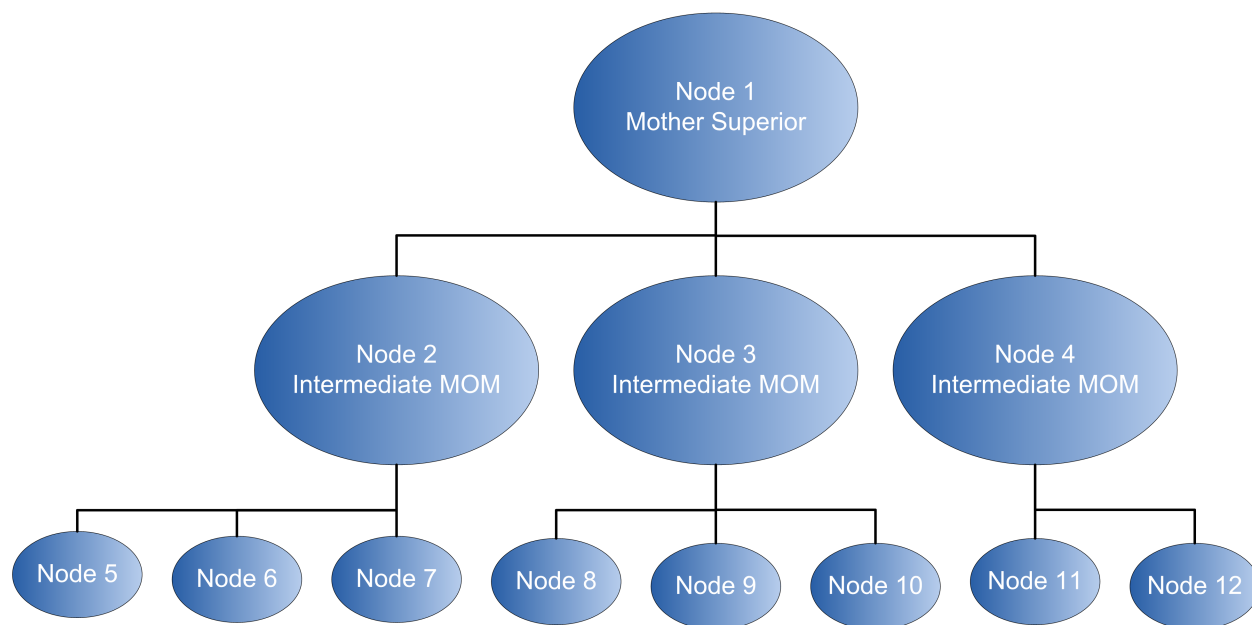
### 3.1.1.B Slot Limit

The slot limit is a way for admins to limit the number of jobs from a job array that can be eligible for scheduling at the same time. When a slot limit is used, Torque puts a hold on all jobs in the array that exceed the slot limit. When an eligible job in the array completes, Torque removes the hold flag from the next job in the array. Slot limits can be declared globally with the *max\_slot\_limit* parameter, or on a per-job basis with *qsub -t*.

## 3.1.2 Managing Multi-Node Jobs

By default, when a multi-node job runs, the Mother Superior manages the job across all the sister nodes by communicating with each of them and updating *pbs\_server*. Each of the sister nodes sends its updates and stdout and stderr directly to the Mother Superior. When you run an extremely large job using hundreds or thousands of nodes, you may want to reduce the amount of network traffic sent from the sisters to the Mother Superior by specifying a job radix. Job radix sets a maximum number of nodes with which the Mother Superior and resulting intermediate MOMs communicate and is specified using the *-W* option for *qsub*.

For example, if you submit a smaller, 12-node job and specify *job\_radix=3*, Mother Superior and each resulting intermediate MOM is only allowed to receive communication from 3 subordinate nodes.

*Image 3-1: Job radix example***12-Node Job With a Job Radix of 3**`qsub -l nodes=12 -W job_radix=3 job.sh`

The Mother Superior picks three sister nodes with which to communicate the job information. Each of those nodes (intermediate MOMs) receives a list of all sister nodes that will be subordinate to it. They each contact up to three nodes and pass the job information on to those nodes. This pattern continues until the bottom level is reached. All communication is now passed across this new hierarchy. The stdout and stderr data is aggregated and sent up the tree until it reaches the Mother Superior, where it is saved and copied to the `.o` and `.e` files.

**i** Job radix is meant for extremely large jobs only. It is a tunable parameter and should be adjusted according to local conditions in order to produce the best results.

### 3.1.3 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 (with the `-l <resource>` syntax) or external scheduler resource extensions (with `-W x=`) can be specified.

`qsub -l` supports:

- All the native Torque resources. See [3.1.3.A Native Torque Resources](#) for a list of resources.
- Some Moab scheduler job extensions (for legacy support). See [3.1.3.D Moab Job Extensions](#) for a list of resources.

**i** For Moab resource extensions, `qsub -W x=` is recommended instead of `qsub -l`. See 'Resource Manager Extensions' in the *Moab Workload Manager Administrator Guide* for a complete list of scheduler-only job extensions.

In this topic:

- [3.1.3.A Native Torque Resources](#)
- [3.1.3.B Interpreting Resource Requests](#)
- [3.1.3.C Interpreting Node Requests](#)
- [3.1.3.D Moab Job Extensions](#)


### 3.1.3.A Native Torque Resources

The native Torque resources are listed in the following table:

Resource	Format	Description
<b>arch</b>	string	Specifies the administrator defined system architecture required. This defaults to whatever the <code>PBS_MACH</code> string is set to in <code>local.mk</code> .
<b>cput</b>	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by all processes in the job on all requested processors.
<b>cpuclock</b>	string	<p>Specify the CPU clock frequency for each node requested for this job. A <code>cpuclock</code> request applies to every processor on every node in the request. Specifying varying CPU frequencies for different nodes or different processors on nodes in a single job request is not supported.</p> <p>Not all processors support all possible frequencies or ACPI states. If the requested frequency is not supported by the CPU, the nearest frequency is used.</p> <p>The clock frequency can be specified via:</p> <ul style="list-style-type: none"> <li>• a number that indicates the clock frequency (with or without the SI unit suffix).</li> </ul> <pre>qsub -l cpuclock=1800,nodes=2 script.sh</pre>

Resource	Format	Description
		<div> <pre>qsub -l cpuclock=1800mhz,nodes=2 script.sh</pre> </div> <div> <p><i>This job requests 2 nodes and specifies their CPU frequencies should be set to 1800 MHz.</i></p> </div> <ul style="list-style-type: none"> <li>• a Linux power governor policy name. The governor names are: <ul style="list-style-type: none"> <li>◦ <b>performance</b>: This governor instructs Linux to operate each logical processor at its maximum clock frequency. This setting consumes the most power and workload executes at the fastest possible speed.</li> <li>◦ <b>powersave</b>: This governor instructs Linux to operate each logical processor at its minimum clock frequency. This setting executes workload at the slowest possible speed. This setting does not necessarily consume the least amount of power since applications execute slower, and may actually consume more energy because of the additional time needed to complete the workload's execution.</li> <li>◦ <b>ondemand</b>: This governor dynamically switches the logical processor's clock frequency to the maximum value when system load is high and to the minimum value when the system load is low. This setting causes workload to execute at the fastest possible speed or the slowest possible speed, depending on OS load. The system switches between consuming the most power and the least power.</li> </ul> </li> </ul> <div> <p><b>i</b> The power saving benefits of <b>ondemand</b> might be non-existent due to frequency switching latency if the system load causes clock frequency changes too often.</p> <p>This has been true for older processors since changing the clock frequency required putting the processor into the C3 'sleep' state, changing its clock frequency, and then waking it up, all of which required a significant amount of time.</p> <p>Newer processors, such as the Intel Xeon E5-2600 Sandy Bridge processors, can change clock frequency dynamically and much faster.</p> </div> <ul style="list-style-type: none"> <li>◦ <b>conservative</b>: This governor operates like the <b>ondemand</b> governor but is more conservative in switching between frequencies. It switches more gradually and uses all possible clock frequencies. This governor can switch to an intermediate clock</li> </ul>

Resource	Format	Description
		<p>frequency if it seems appropriate to the system load and usage, which the <code>ondemand</code> governor does not do.</p> <pre>qsub -l cpuclock=<del>performance</del>,nodes=2 script.sh</pre> <p><i>This job requests 2 nodes and specifies their CPU frequencies should be set to the performance power governor policy.</i></p> <ul style="list-style-type: none"> <li>an ACPI performance state (or P-state) with or without the P prefix. P-states are a special range of values (0-15) that map to specific frequencies. Not all processors support all 16 states, however, they all start at P0. P0 sets the CPU clock frequency to the highest performance state, which runs at the maximum frequency. P15 sets the CPU clock frequency to the lowest performance state, which runs at the lowest frequency.</li> </ul> <pre>qsub -l cpuclock=<b>3</b>,nodes=2 script.sh qsub -l cpuclock=<del>p3</del>,nodes=2 script.sh</pre> <p><i>This job requests 2 nodes and specifies their CPU frequencies should be set to a performance state of 3.</i></p> <p>When reviewing job or node properties when <code>cpuclock</code> was used, be mindful of unit conversion. The OS reports frequency in Hz, not MHz or GHz.</p>
<b>epilogue</b>	string	Specifies a user owned epilogue script, which will be run before the system epilogue and <code>epilogue.user</code> scripts at the completion of a job. The syntax is <code>epilogue=&lt;file&gt;</code> . The file can be designated with an absolute or relative path. For more information, see <a href="#">Appendix G: Prologue and Epilogue Scripts</a> .
<b>feature</b>	string	Specifies a property or feature for the job. Feature corresponds to Torque node properties and Moab features.
		<pre>qsub script.sh -l procs=10,feature=bigmem</pre>
<b>file</b>	<a href="#">size*</a>	Sets <code>RLIMIT_FSIZE</code> for each process launched through the TM interface. See 'FILEREQUESTISJOBCENTRIC' in the <i>Moab Workload Manager Administrator Guide</i> for information on how Moab schedules.
<b>host</b>	string	Name of the host on which the job should be run. This resource is provided for use by the site's scheduling policy.

Resource	Format	Description
		The allowable values and effect on job placement is site dependent.
<b>mem</b>	size*	<p>Maximum amount of physical memory used by the job. Ignored on Darwin, Digital UNIX, Free BSD, HPUX 11, IRIX, NetBSD, and SunOS. Not implemented on AIX and HPUX 10.</p> <p>The <code>mem</code> resource will only work for single-node jobs. If your job requires multiple nodes, use <code>pmem</code> instead.</p>
<b>ncpus</b>	integer	<p>The number of processors in one task where a task cannot span nodes.</p> <div>  You cannot request both <code>ncpus</code> and <code>nodes</code> in the same job. </div>
<b>nice</b>	integer	Number between -20 (highest priority) and 19 (lowest priority). Adjust the process execution priority.
<b>nodes</b>	{<node_count>   <hostname>} [:ppn=<ppn>] [:gpus=<gpu>] [:<property> [:<property>]...] [+ ...]	<p>Number and/or type of nodes to be reserved for exclusive use by the job. The value is one or more <code>node_specs</code> joined with the + (plus) character: <code>node_spec[+node_spec...]</code>. Each <code>node_spec</code> is a number of nodes required of the type declared in the <code>node_spec</code> and a name of one or more properties desired for the nodes. The number, the name, and each property in the <code>node_spec</code> are separated by a : (colon). If no number is specified, one (1) is assumed. The name of a node is its hostname.</p> <p>The properties of nodes are:</p> <ul style="list-style-type: none"> <li><code>ppn=#</code> - 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 <code>TORQUE_HOME/server_priv/nodes</code> file using the <code>np</code> attribute (see <a href="#">2.5 Server Node File Configuration</a>). 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 <code>np</code> value greater than the number of physical cores (or hyper-thread contexts). The <code>ppn</code> value is a characteristic of the hardware, system, and site, and its value is to be determined by the admin.</li> <li><code>gpus=#</code> - Specify the number of GPUs per node requested for this job. The number of GPUs available on a node can be configured in the <code>TORQUE_HOME/server_priv/nodes</code> file using the <code>gpu</code> attribute (see <a href="#">2.5 Server</a></li> </ul>

Resource	Format	Description
		<p><a href="#">Node File Configuration</a>). The GPU value is a characteristic of the hardware, system, and site, and its value is to be determined by the admin.</p> <ul style="list-style-type: none"> <li>property - A string assigned by the system admin specifying a node's features. Check with your admin as to the node names and properties available to you.</li> </ul> <div> <p><b>i</b> Torque does not have a TPN (tasks per node) property. You can specify TPN in Moab Workload Manager with Torque as your resource manager, but Torque does not recognize the property when it is submitted directly to it via <i>qsub</i>.</p> </div> <p>See <a href="#">Example 3-2: qsub -l nodes</a> for examples.</p> <div> <p><b>i</b> 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 'Moab Parameters' in the <i>Moab Workload Manager Administrator Guide</i> for more information.</p> </div> <div> <p><b>i</b> All nodes in Torque have their own name as a property. You can request a specific node by using its name in the nodes request. Multiple nodes can be requested this way by using '+' as a delimiter. For example:</p> <pre>qsub -l nodes=node01:ppn=3+node02:ppn=6</pre> </div> <p>See the HOSTLIST RM extension in the <i>Moab Workload Manager Administrator Guide</i> for more information.</p>
<b>opsys</b>	string	Specifies the administrator defined operating system as defined in the MOM configuration file.
<b>other</b>	string	<p>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.</p> <div> <p><b>i</b> This does not work for <i>msub</i> using Moab and Maui.</p> </div>
<b>pcput</b>	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by any single process in the job.
<b>pmem</b>	size*	Maximum amount of physical memory used by any single



Resource	Format	Description
		process of the job. Ignored on Fujitsu. Not implemented on Digital UNIX and HPUX.
<b>procs</b>	procs=<integer>	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 can be used per submitted <a href="#">qsub</a> command.  > qsub -l nodes=3 -1 procs=2
<b>procs_bitmap</b>	string	A string made up of 1s and 0s in reverse order of the processor cores requested. A <code>procs_bitmap=1110</code> 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. For more information, see <a href="#">4.7 Scheduling Cores</a> .
<b>prologue</b>	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 <code>prologue=&lt;file&gt;</code> . The file can be designated with an absolute or relative path. For more information, see <a href="#">Prologue and Epilogue Scripts</a> .
<b>pvmem</b>	<a href="#">size</a> *	Maximum amount of virtual memory used by any single process in the job. (Ignored on Unicos.)
<b>size</b>	integer	For Torque, this resource has no meaning. It is passed on to the scheduler for interpretation. See the <a href="#">note</a> at the end of this table about the supported format.
<b>software</b>	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 'License Management' in the <i>Moab Workload Manager Administrator Guide</i> for more information.
<b>vmem</b>	<a href="#">size</a> *	Maximum amount of virtual memory used by all concurrent processes in the job. (Ignored on Unicos.)
<b>walltime</b>	seconds, or [[HH:]MM:]SS	Maximum amount of real time during which the job can be in the running state.

\*size

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

### 3.1.3.B Interpreting Resource Requests

The table below shows how various requests are interpreted in the `qsub -l` syntax and corresponding cgroup usage.

Memory parameters (`mem`, `pmem`, `vmem`, `pvmem`) may specify units (examples: `mem=1024mb`, `mem=320kb`, `mem=1gb`). Recognized units are kb (kilobytes), mb (megabytes), gb (gigabytes), tb (terabyte), pb (petabytes), and eb (exabyte). If units are not specified, mb (megabytes) is assumed.

*Example 3-1: Interpreting `qsub -l` requests*

Usage	Description
<code>node=X:ppn=Y</code>	Creates <i>X</i> tasks that will use <i>Y</i> lprocs per task.
<code>procs=X</code>	Creates <i>X</i> tasks that will use 1 lproc each.
<code>ncpus=X</code>	Creates 1 task that will use <i>X</i> lprocs.
<code>mem=X</code>	The entire job will use <i>X</i> memory, divided evenly among the tasks.*
<code>pmem=X</code>	Each task will use <i>X</i> memory. In translation, <code>-l nodes=1:ppn=4,pmem=1gb</code> will use 4 GB of memory.*
<code>vmem=X</code>	The entire job will use <i>X</i> swap, divided evenly among the tasks. If legacy_

Usage	Description
	vmem is set to true in the server, then the entire specified value will be given per host.**
pvmem=X	Each task will use X swap. In translation, -l nodes=1:ppn=4,pvmem=1gb will use 4 GB of swap.**

\*If both mem and pmem are specified, the less restrictive of the two will be used as the limit for the job. For example, `qsub job.sh -l nodes=2:ppn=2,mem=4gb,pmem=1gb` will apply the mem requested instead of pmem, because it will allow 2 GB per task (4 GB/2 tasks) instead of 1 GB per task.

\*\*If both vmem and pvmem are specified, the less restrictive of the two will be used as the limit for the job. For example, `qsub job.sh -l nodes=2:ppn=2,vmem=4gb,pvmem=1gb` will apply pvmem instead of vmem, because it will allow 2 GB swap per task (1 GB \* 2 ppn) instead of .5 GB per task (1 GB/2 tasks).

### 3.1.3.C Interpreting Node Requests

The table below shows how various `qsub -l nodes` requests are interpreted.

*Example 3-2: qsub -l nodes*

Usage	Description
<code>&gt; qsub -l nodes=12</code>	Request 12 nodes of any type.
<code>&gt; qsub -l nodes=2:server+14</code>	Request 2 'server' nodes and 14 other nodes (a total of 16) - this specifies two node_specs, '2:server' and '14'.
<code>&gt; qsub -l nodes=server:hippi+10:noserver+3:bigmem:hippi</code>	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 and have hippy.
<code>&gt; qsub -l nodes=b2005+b1803+b1813</code>	Request 3 specific nodes by hostname.
<code>&gt; qsub -l nodes=4:ppn=2</code>	Request 2 processors on each of four nodes.
<code>&gt; qsub -l nodes=1:ppn=4</code>	Request 4 processors on one node.
<code>&gt; qsub -l nodes=2:blue:ppn=2+red:ppn=3+b1014</code>	Request 2 processors on each of two blue nodes, three processors on one red node, and the compute node 'b1014'.

**Example 3-3:**

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

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

**Example 3-4:**

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

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

### 3.1.3.D Moab Job Extensions

`qsub -l` also supports some Moab resource extension values listed below, but be advised that this usage has been deprecated. The ones that currently work will remain for the purpose of legacy support, but additional ones will not be added. Instead, we recommend transitioning to the `-W x=<resource>` syntax mentioned at the top of this page.

advres	image	procs	subnode_list
cpuclock	jgroup	procs_bitmap	taskdistpolicy
deadline	jobflags	prologue	template
depend	latency	qos	termsig
ddisk	loglevel	queuejob	termtime
dmem	minprocspeed	reqattr	tid
energy_used	minpreempttime	retrycount	tpn
epilogue	minwclimit	retrycc	trig
feature	naccesspolicy	rmttype	trl
flags	nallocpolicy	select	var
gattr	nodeset	sid	vcores
geometry	opsys	signal	wcrequeue
gmetric	os	stagein	
gres	partition	spriority	
hostlist	pref	subnode	

### 3.1.4 Requesting NUMA-Aware Resources

 This topic only applies for NUMA-aware systems.

Various NUMA resources can be requested at the time of job submission.

The `qsub -L` option enables admins the ability to place jobs at the 'task' or 'OS process' level to get maximum efficiency out of the available hardware. In addition, multiple, non-symmetric resource requests can be made for the same job using the `-L` job submission syntax. See section [12.4 -L NUMA Resource Request](#) for a complete list of `-L` values. For example:

```
qsub -L tasks=4:lprocs=2:usecores:memory=500mb -L tasks=8:lprocs=4:memory=2gb
```

Creates two requests. The first will create 4 tasks with two logical processors per task and 500 MB of memory per task. The logical processors will be placed on cores. The second request calls for 8 tasks with 4 logical processors per task and 2 GB of memory per task. The logical processors can be placed on cores or threads since the default placement is `allowthreads`.

**i** The queue attribute `resources_default` has several options that are not compatible with the `qsub -L` syntax. If a queue has any of the following `resources_default` options set (again, without a corresponding `req_information_default` setting), the job will be rejected from the queue:

`nodes, size, mppwidth, mem, hostlist, ncpus, procs, pvmem, pmem, vmem, reqattr, software, geometry, opsys, tpn, and trl.`

**i** See [5.1.2 Setting Queue Resource Controls](#) for more information about setting queue resource requirements and the use of `-l` and `-L` job submission syntaxes.

### 3.1.5 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 'Managing Consumable Generic Resources' in the *Moab Workload Manager Administrator Guide* for details on configuration within Moab.

#### Example 3-5: Generic

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

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

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

### 3.1.6 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 'Managing Shared Cluster Resources (Floating Resources)' in the *Moab Workload Manager Administrator Guide* for details on configuration within Moab.

#### Example 3-6: Floating

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

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

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

### 3.1.7 Requesting Other Resources

Many other resources can be requested at the time of job submission using Moab Workload Manager [via the `qsub -W x=syntax` (or `qsub -l` in limited cases), or via `msub -l` or `msub -W x=`]. See 'Resource Manager Extensions' in the *Moab Workload Manager Administrator Guide* for a list of these supported requests and correct syntax.

### 3.1.8 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
<b>PBS_ARRAYID</b>	Zero-based value of job array index for this job
<b>PBS_ENVIRONMENT</b>	Set to <code>PBS_BATCH</code> to indicate the job is a batch job, or to <code>PBS_INTERACTIVE</code> to indicate the job is a PBS interactive job (see <code>-I</code> option)
<b>PBS_GPUFILE</b>	Line-delimited list of GPUs allocated to the job located in <code>TORQUE_HOME/aux/jobidgpu</code> . Each line follows the following format: <code>&lt;host&gt;-gpu&lt;number&gt;</code> For example, <code>myhost-gpu1</code> .

Variable	Description
<b>PBS_JOBCOOKIE</b>	Job cookie
<b>PBS_JOBID</b>	Unique pbs job id
<b>PBS_JOBNAME</b>	User specified jobname
<b>PBS_MOMPORT</b>	Active port for MOM daemon
<b>PBS_NODEFILE</b>	File containing line delimited list of nodes allocated to the job
<b>PBS_NODENUM</b>	Node offset number
<b>PBS_NP</b>	Number of execution slots (cores) for the job
<b>PBS_NUM_NODES</b>	Number of nodes allocated to the job
<b>PBS_NUM_PPN</b>	Number of procs per node allocated to the job
<b>PBS_O_HOME</b>	Home directory of submitting user
<b>PBS_O_HOST</b>	Host on which job script is currently running
<b>PBS_O_LANG</b>	Language variable for job
<b>PBS_O_LOGNAME</b>	Name of submitting user
<b>PBS_O_PATH</b>	Path variable used to locate executables within job script
<b>PBS_O_SHELL</b>	Script shell
<b>PBS_O_WORKDIR</b>	Job's submission directory
<b>PBS_QUEUE</b>	Job queue
<b>PBS_TASKNUM</b>	Number of tasks requested

### 3.1.9 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 (see the section [Allowing Job Submission from Compute Hosts](#)). Allows any host trusted as a compute host to also be trusted as a submit host.
- Set the `submit_hosts` server parameter (see the section [Using the 'submit\\_hosts' Service Parameter](#)). Allows specified hosts to be trusted as a submit host.
- Use `.rhosts` to enable `ruserok()` based authentication (see the section [Using RCmd Authentication](#)).

See [2.3.2.C Configuring Job Submission Hosts](#) for more information.

**i** When you enable `allow_node_submit`, you must also enable the `allow_proxy_user` parameter to allow user proxying when submitting and running jobs.

### 3.1.10 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
```

## 3.2 Monitoring Jobs

Torque enables users and admins 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
...					

## Monitoring NUMA Job Task Placement

When using NUMA, job resources are tracked per task. To support this `qstat -f` produces a new category of information that begins with the `req_information` keyword. Following each `req_information` keyword is another keyword giving information about how the job was allocated. See [12.4 -L NUMA Resource Request](#) for available allocation keywords.

When the job has completed, the output will also include the per-task resident memory used and per-task CPU time used.

**i** Timing issues may prevent the `resources_uses.mem` value from accurately reporting the maximum amount of memory used, particularly if the logging level is set above 0.

The following is a sample `qstat -f` completed job output. You will see that `req_information.task_usage.0.task.0.cpu_list` gives the cores to which the job is bound for the `cpuset`. The same for `mem_list`. The keywords `memory_used` and `cput_used` report the per task resident memory used and CPU time used respectively.

```
Job Id: 832.pv-knielson-dt
Job_Name = bigmem.sh
Job_Owner = knielson@pv-knielson-dt
resources_used.cput = 00:00:00
resources_used.energy_used = 0
resources_used.mem = 3628kb
resources_used.vmem = 31688kb
resources_used.walltime = 00:00:00
job_state = C
queue = second
server = pv-knielson-dt
Checkpoint = u
ctime = Tue Jul 28 23:23:15 2025
Error_Path = pv-knielson-dt:/home/knielson/jobs/bigmem.sh.e832
exec_host = pv-knielson-dt/0-3
Hold_Types = n
Join_Path = n
Keep_Files = n
Mail_Points = a
mtime = Tue Jul 28 23:23:18 2025
Output_Path = pv-knielson-dt:/home/knielson/jobs/bigmem.sh.o832
Priority = 0
qtime = Tue Jul 28 23:23:15 2025
Rerunable = True
Resource_List.walltime = 00:05:00
session_id = 2708
substate = 59
Variable_List = PBS_O_QUEUE=routeme,PBS_O_HOME=/home/knielson,
PBS_O_LOGNAME=knielson,
PBS_O_PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b
```

```

in:/usr/games:/usr/local/games,PBS_O_SHELL=/bin/bash,PBS_O_LANG=en_US,
PBS_O_WORKDIR=/home/knielson/jobs,PBS_O_HOST=pv-knielson-dt,
PBS_O_SERVER=pv-knielson-dt
euser = knielson
egroup = company
hashname = 832.pv-knielson-dt
queue_rank = 391
queue_type = E
etime = Tue Jul 28 23:23:15 2025
exit_status = 0
submit_args = -L tasks=2:lprocs=2 ../scripts/bigmem.sh
start_time = Tue Jul 28 23:23:18 2025
start_count = 1
fault_tolerant = False
comp_time = Tue Jul 28 23:23:18 2025
job_radix = 0
total_runtime = 0.093262
submit_host = pv-knielson-dt
req_information.task_count.0 = 2
req_information.lprocs.0 = 2
req_information.thread_usage_policy.0 = allowthreads
req_information.hostlist.0 = pv-knielson-dt:ppn=4
req_information.task_usage.0.task.0.cpu_list = 2,6
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.memory_used = 258048
req_information.task_usage.0.task.0.cput_used = 18
req_information.task_usage.0.task.0.cores = 0
req_information.task_usage.0.task.0.threads = 0
req_information.task_usage.0.task.0.host =
req_information.task_usage.0.task.1.cpu_list = 3,7
req_information.task_usage.0.task.1.mem_list = 0
req_information.task_usage.0.task.1.memory_used = 258048
req_information.task_usage.0.task.1.cput_used = 18
req_information.task_usage.0.task.1.cores = 0
req_information.task_usage.0.task.1.threads = 2
req_information.task_usage.0.task.1.host = pv-knielson-dt

```

### 3.3 Canceling Jobs

Torque enables users and admins 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 can 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
$

```

## 3.4 Job Preemption

Torque supports job preemption by allowing authorized users to suspend and resume jobs. This is supported using one of two methods. If the node supports OS-level preemption, Torque will recognize that during the configure process and enable it. Otherwise, the MOM can 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
```

## 3.5 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 attribute on the job execution queue or the `keep_completed` parameter on the server. This should be set to the number of seconds that jobs should be held in the queue. If you set `keep_completed` on the job execution queue, completed jobs will be reported in the `C` state and the exit status is seen in the `exit_status` job attribute.

**i** If the Mother Superior and Torque server are on the same server, expect the following behavior:

- When `keep_completed` is set, the job spool files will be deleted when the specified time arrives and Torque purges the job from memory.
- When `keep_completed` is not set, Torque deletes the job spool files upon job completion.
- If you manually purge a job (`qdel -p`) before the job completes or time runs out, Torque will never delete the spool files.

By maintaining status information about completed (or canceled, failed, etc.) jobs, admins can better track failures and improve system performance. This enables 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 'Moab Parameters' in the *Moab Workload Manager Administrator Guide* for more information.

## 3.6 Job Checkpoint and Restart

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

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

In this section:

- [3.6.1 Introduction to BLCR](#)
- [3.6.2 Configuration Files and Scripts](#)
- [3.6.3 Starting a Checkpointable Job](#)
- [3.6.4 Checkpointing a Job](#)
- [3.6.5 Restarting a Job](#)
- [3.6.6 Acceptance Tests](#)

### 3.6.1 Introduction to BLCR

BLCR (Berkeley Lab Checkpoint/Restart) 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:

- `cr_checkpoint`
- `cr_info`
- `cr_restart`
- `cr_run`

For more information about BLCR, see the [BLCR Administrator's Guide](#).

### 3.6.2 Configuration Files and Scripts

Configuring and Building Torque for BLCR:

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

Depending on where BLCR is installed you may also need to use the following configure options to specify BLCR paths:

Option	Description
<code>--with-blcr-include=DIR</code>	include path for <code>libcr.h</code>
<code>--with-blcr-lib=DIR</code>	lib path for <code>libcr</code>
<code>--with-blcr-bin=DIR</code>	bin path for BLCR utilities

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:

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

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

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

#### Example 3-7: Script file permissions

```
# chmod 754 blcr*
# ls -l
total 20
-rwxr-xr-- 1 root root 2112 2025-03-11 13:14 blcr_checkpoint_script
-rwxr-xr-- 1 root root 1987 2025-03-11 13:14 blcr_restart_script
-rw-r--r-- 1 root root 215 2025-03-11 13:13 config
drwxr-x--x 2 root root 4096 2025-03-11 13:21 jobs
-rw-r--r-- 1 root root 7 2025-03-11 13:15 mom.lock
```

#### Example 3-8: `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
```

#### Example 3-9: `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
# image files that are preserved on the disk.
#
# Note also that a request was made to identify whether this script was invoked by the
# job's
# owner or by a system administrator. While this information is known to pbs_server,
# it
# is not propagated to pbs_mom and thus it is not possible to pass this to the script.

# Therefore, a workaround is to invoke qmgr and attempt to set a trivial variable.
# This will fail if the invoker is not a manager.

if (@ARGV == 7)
{
    ($sessionId, $jobId, $userId, $checkpointDir, $checkpointName, $signalNum $depth)
    =
        @ARGV;
}
else { logDie(1, $usage); }

# Change to the checkpoint directory where we want the checkpoint to be created
chdir $checkpointDir
    or logDie(1, "Unable to cd to checkpoint dir ($checkpointDir): $!\n")
    if $logLevel;

my $cmd = "cr_checkpoint";
$cmd .= " --signal $signalNum" if $signalNum;
$cmd .= " --tree $sessionId";
$cmd .= " --file $checkpointName";
my $output = `$cmd 2>&1`;
my $rc = $? >> 8;
logDie(1, "Subcommand ($cmd) failed with rc=$rc:\n$output")
    if $rc && $logLevel >= 1;
logPrint(3, "Subcommand ($cmd) yielded rc=$rc:\n$output")
    if $logLevel >= 3;
exit 0;

#####
# logPrint($message)
# Write a message (to syslog) and die
#####
sub logPrint
{

```

```

    my ($level, $message) = @_ ;
    my @severity = ('none', 'warning', 'info', 'debug');

    return if $level > $logLevel;

    openlog('checkpoint_script', '', 'user');
    syslog($severity[$level], $message);
    closelog();
}

#####
# logDie($message)
# Write a message (to syslog) and die
#####
sub logDie
{
    my ($level, $message) = @_ ;
    logPrint($level, $message);
    die($message);
}

```

**Example 3-10:** *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;
}
else { logDie(1, $usage); }

# Change to the checkpoint directory where we want the checkpoint to be created
chdir $checkpointDir
    or logDie(1, "Unable to cd to checkpoint dir ($checkpointDir): $!\n")
    if $logLevel;

```



```

my $cmd = "cr_restart";
$cmd .= " $restartName";
my $output = `$cmd 2>&1`;
my $rc = $? >> 8;
logDie(1, "Subcommand ($cmd) failed with rc=$rc:\n$output")
    if $rc && $logLevel >= 1;
logPrint(3, "Subcommand ($cmd) yielded rc=$rc:\n$output")
    if $logLevel >= 3;
exit 0;

#####
# logPrint($message)
# Write a message (to syslog) and die
#####
sub logPrint
{
    my ($level, $message) = @_ ;
    my @severity = ('none', 'warning', 'info', 'debug');

    return if $level > $logLevel;
    openlog('restart_script', '', 'user');
    syslog($severity[$level], $message);
    closelog();
}

#####
# logDie($message)
# Write a message (to syslog) and die
#####
sub logDie
{
    my ($level, $message) = @_ ;

    logPrint($level, $message);
    die($message);
}

```

### 3.6.3 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 is shown below:

Option	Description
<b>none</b>	No checkpointing (not highly useful, but included for completeness).
<b>enabled</b>	Specify that checkpointing is allowed, but must be explicitly invoked by either the <code>qhold</code> or <code>qchkpt</code> commands.

Option	Description
<b>shutdown</b>	Specify that checkpointing is to be done on a job at pbs_mom shutdown.
<b>periodic</b>	Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the <code>\$checkpoint_interval</code> option in the MOM configuration file, or by specifying an interval when the job is submitted.
<b>interval=minutes</b>	Specify the checkpoint interval in minutes.
<b>depth=number</b>	Specify a number (depth) of checkpoint images to be kept in the checkpoint directory.
<b>dir=path</b>	Specify a checkpoint directory (default is <code>/var/spool/torque/checkpoint</code> ).

*Example 3-11: 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);
    }
}
```

*Example 3-12: Instructions for building test program*

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

*Example 3-13: Sample test script*

```
#!/bin/bash ./test
```

*Example 3-14: 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 observed with the command `tail -f`.

### 3.6.4 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`.

### 3.6.5 Restarting a Job

#### 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 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.

## 3.6.6 Acceptance Tests

A number of tests were made to verify the functioning of the BLCR implementation. See the appendix [BLCR Acceptance Tests](#) for a description of the testing.

## 3.7 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.

*Example 3-15: `qstat -f` (job failure)*

```
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 2025
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 2025
Output_Path = host:/opt/moab/STDIN.o179
Priority = 0
qtime = Fri Aug 29 14:55:55 2025
Rerunnable = True Resource_List.ncpus = 2
Resource_List.nodecnt = 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:,PBS_O_
SHELL=/bin/bash,PBS_O_HOST=host,
PBS_O_WORKDIR=/opt/moab,PBS_O_QUEUE=batchq
    sched_hint = Post job file processing error; job 179.host on host node1/0Ba
d UID for job execution REJHOST=pala.cridomain MSG=cannot find user 'user' in
password file
    etime = Fri Aug 29 14:55:55 2025
    exit_status = -1
```

 The value of `Resource_List.*` is the amount of resources requested.

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.

*Example 3-16: Torque supplied exit codes*

Name	Value	Description
<b>JOB_EXEC_OK</b>	0	Job execution successful
<b>JOB_EXEC_FAIL1</b>	-1	Job execution failed, before files, no retry
<b>JOB_EXEC_FAIL2</b>	-2	Job execution failed, after files, no retry
<b>JOB_EXEC_RETRY</b>	-3	Job execution failed, do retry
<b>JOB_EXEC_INITABT</b>	-4	Job aborted on MOM initialization
<b>JOB_EXEC_INITRST</b>	-5	Job aborted on MOM init, chkpt, no migrate
<b>JOB_EXEC_INITRMG</b>	-6	Job aborted on MOM init, chkpt, ok migrate
<b>JOB_EXEC_BADRESRT</b>	-7	Job restart failed
<b>JOB_EXEC_CMDFAIL</b>	-8	Exec() of user command failed
<b>JOB_EXEC_STDOUTFAIL</b>	-9	Could not create/open stdout stderr files
<b>JOB_EXEC_OVERLIMIT_MEM</b>	-10	Job exceeded a memory limit
<b>JOB_EXEC_OVERLIMIT_WT</b>	-11	Job exceeded a walltime limit
<b>JOB_EXEC_OVERLIMIT_CPU</b>	-12	Job exceeded a CPU time limit
<b>JOB_EXEC_RETRY_CGROUP</b>	-13	Could not create the job's cgroups
<b>JOB_EXEC_RETRY_PROLOGUE</b>	-14	Prologue failed

*Example 3-17: 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
  queue = batch
  server = xxx.yyy
  Checkpoint = u
  ctime = Wed Apr 30 11:29:37 2025
  Error_Path = xxx.yyy:/home/test/STDIN.e180
  exec_host = node01/0
  Hold_Types = n
  Join_Path = n
  Keep_Files = n
  Mail_Points = a
  mtime = Wed Apr 30 11:29:37 2025
  Output_Path = xxx.yyy:/home/test/STDIN.o180
  Priority = 0
  qtime = Wed Apr 30 11:29:37 2025
  Rerunnable = True
  Resource_List.needsnodes = 1
  Resource_List.nodect = 1
  Resource_List.nodes = 1
  Resource_List.walltime = 01:00:00
  session_id = 14107
  substate = 59
  Variable_List = PBS_O_HOME=/home/test,PBS_O_LANG=en_US.UTF-8,
    PBS_O_LOGNAME=test,
    PBS_O_PATH=/usr/local/perltests/bin:/home/test/bin:/usr/local/s
    bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games,
    PBS_O_SHELL=/bin/bash,PBS_SERVER=xxx.yyy,
    PBS_O_HOST=xxx.yyy,PBS_O_WORKDIR=/home/test,
    PBS_O_QUEUE=batch
  euser = test
  egroup = test
  hashname = 180.xxx.yyy
  queue_rank = 8
  queue_type = E
  comment = Job started on Wed Apr 30 at 11:29
  etime = Wed Apr 30 11:29:37 2025
  exit_status = 11

```

```
start_time = Wed Apr 30 11:29:37 2025
start_count = 1
```

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.

## 3.8 Torque Process Tracking

This section describes how Torque tracks the lifecycle and resource usage of processes and how to use the Task Manager (TM) API to inform Torque of processes to be tracked. See [Chapter 8: MPI \(Message Passing Interface\) Support](#) for more details on the TM API.

In this section:

- [3.8.1 Default Process Tracking](#)
- [3.8.2 Task Manager API](#)
- [3.8.3 Process Tracking with cgroups/cpusets](#)

### 3.8.1 Default Process Tracking

By default, Torque tracks all processes that it launches, as well as child processes that share the same session ID as processes launched by Torque.

When a job is launched, the master process for the job is a child of the `pbs_mom` daemon on the mother superior node. If that process forks, the child will share a session ID with the master process, and Torque automatically tracks the process.

### 3.8.2 Task Manager API

If a job uses the Task Manager (TM) API to launch a process, then that process will also be automatically tracked along with its children. Most, if not all, MPI libraries can be built to interact with Torque. When properly configured to do so, MPI libraries either launch processes in the job through Torque, or inform Torque that it has launched a new process that should be part of the job.

#### Launching Through Torque

The TM API provides the function `tm_spawn()`. If this function is invoked by an MPI library or some other program, it will send the executable path or name with all of its arguments and environment to the local `pbs_mom`, along with instructions for where the process should be launched and some data for identifying and tracking that process. The

local MOM will then launch the process if it is local, or send the information to a remote MOM to launch the process if it should be launched on another host that is part of the job.

The `pbsdsh` command that comes with Torque uses `tm_spawn()` to launch processes that will be part of the job. If you are doing simple proof-of-concept work, `pbsdsh` is a built-in launcher that offers some simple options for launching processes within a Torque job.

## Informing Torque of Other Processes

Another option available for making a process part of a job is the `tm_adopt()` function. Some MPI implementations have their own launching mechanism for starting processes—whether remote or local—and use this instead of the one provided by Torque. To accommodate this behavior, the `tm_adopt()` function can be used to inform the MOM that it should track the process as part of the job.

**i** The `tm_adopt()` function must be called on the host where the process has been launched.

The `pbs_track()` command can be used to launch a process that will be adopted by a specified Torque job, or it can be used to inform the local MOM that an existing process should be adopted by a specified Torque job. In either case, the specified Torque job must be currently executing on the local MOM.

### 3.8.3 Process Tracking with cgroups/cpusets

With cgroups, Torque generally follows its default approach to tracking processes, but instead of following session IDs, any process that is part of the job's cgroup is considered part of the job. Generally speaking, processes launched by processes within a cgroup inherit their parent's cgroup, but this part is managed by the operating system. In the case of processes that are launched or adopted by the TM API, the mom daemon will add these processes to the job's cgroup.

If a process is launched in some way that is exterior to Torque and avoids the cgroup, then it will not be restricted by the job's cgroup. The only way to guarantee that jobs are properly restricted is to ensure that process launchers (usually MPI implementations) are properly configured to either launch through Torque or inform Torque of the processes that they launch.

---

## Related Topics

- [A.6 pbs\\_track](#)



## 3.9 Large Job Arrays

### Behavior for Job Arrays Submitted with `qsub`

When a job array is submitted using `qsub`, Torque will create all of the subjobs in its queues, and then it reports the new jobs to Moab the next time it performs a cluster query. When Moab discovers the new jobs, it will create all of the job files, place the jobs in the appropriate queue, and perform other job submission tasks.

With small job arrays, this does not present any problems, but if a very large job array is submitted via `qsub` however, the impact on Moab is significant. The cluster query will require longer to complete, and scheduling will stop while Moab processes the new data and jobs. Sites have even restarted Moab thinking it had crashed, when more than likely it was just very busy, which extended the scheduling iteration and made it unresponsive.

For arrays larger than 500 subjobs, we recommend that `msub` be used for job submission instead of `qsub`. This puts Moab in control of managing the job array right from submission, and with the proper settings, Moab can handle any size array very well. Refer to section 'Minimizing Impact of Very Large Array Jobs' in the *Moab Workload Manager Administrator Guide* for more details.

We also recommend that the setting `set server moab_array_compatible = True` be used if job arrays are expected to be submitted.

## Chapter 4: Managing Nodes

This chapter contains information about adding and configuring compute nodes. It explains how to work with host security for systems that require dedicated access to compute nodes. It also contains information about scheduling specific cores on a node at job submission.

In this chapter:

- [4.1 Adding Nodes](#)
- [4.2 Node Properties](#)
- [4.3 Changing Node State](#)
- [4.4 Changing Node Power States](#)
- [4.5 Host Security](#)
- [4.6 Linux cpuset Support](#)
- [4.7 Scheduling Cores](#)
- [4.8 Scheduling Accelerator Hardware](#)
- [4.9 Node Resource Plug-In](#)

### 4.1 Adding Nodes

Torque can add and remove nodes either dynamically with [qmgr](#) or by manually editing the `TORQUE_HOME/server_priv/nodes` file. See [2.2 Initializing/Configuring Torque on the Server \(pbs\\_server\)](#).

**i** Be aware of the following:

- Nodes cannot be added or deleted dynamically if there is a `mom_hierarchy` file in the `server_priv` directory.
- When you make changes to nodes by directly editing the nodes file, you must restart `pbs_server` for those changes to take effect. Changes made using `qmgr` do not require a restart.
- When you make changes to a node's IP address, you must clear the `pbs_server` cache. Either restart `pbs_server` or delete the changed node and then re-add it.
- Before a newly added node is set to a free state, the cluster must be informed that the new node is valid and they can trust it for running jobs. Once this is done, the node will automatically transition to free.
- Adding or changing a hostname on a node requires a `pbs_server` restart in order to add the new hostname as a node.

## 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[,node004,node005...] [np=n] [, [TTL=YYYY-MM-DDThh:mm:ssZ], [acl=user:user1[:user2:user3...]], [requestid=n]]'
```

The optional parameters are used as follows:

- `np` – Number of virtual processors.
- `TTL` – (Time to Live) Specifies the time in UTC format that the node is supposed to be retired by Moab. Moab will not schedule any jobs on a node after its time to live has passed.
- `acl` – (Access control list) Can be used to control which users have access to the node in Moab.



Except for temporary nodes and/or the simplest of cluster configurations, We recommend avoiding the use of the `acl` parameter, as this can lead to confusion about the root cause of jobs being unable to run. Use Moab reservations with user ACLs instead.

- `requestid` – An ID that can be used to track the request that created the node.

You can alter node parameters by following these examples:

```
qmgr -c 'set node node003 np=6'
qmgr -c 'set node node003 TTL=2025-12-31T23:59:59Z'
qmgr -c 'set node node003 requestid=23234'
qmgr -c 'set node node003 acl="user:user10:user11:user12"'
```

```
qmgr -c 'set node node003 acl=""'
```

**i** Torque does not use the `TTL`, `acl`, and `requestid` parameters. Information for those parameters are simply passed to Moab.

**!** The `set node` subcommand of `qmgr` supports the `+=` and `-=` syntax, but has known problems when used to alter the `acl` parameter. Do not use it for this. Instead, simply reset the full user list, as shown in the above example.

The `create node` and `set node` command examples above would append the following line(s) to the bottom of the `TORQUE_HOME/server_priv/nodes` file:

```
node003 np=6 TTL=2025-12-31T23:59:59Z acl=user1:user2:user3 requestid=3210
node004 ...
```

Nodes can also be removed with a similar command:

```
> qmgr -c 'delete node node003[,node004,node005...]'
```

## 4.2 Node Properties

Torque can associate properties with nodes to aid in identifying groups of nodes. It's typical for a site to conglomerate a heterogeneous set 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.

In this section:

[4.2.1 Run-Time Node Changes](#)

[4.2.2 Manual Node Changes](#)

[4.2.3 Adding Memory to a Node](#)

### 4.2.1 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.

## 4.2.2 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:

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

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

**i** For a full description of this file, see [2.5 Server Node File Configuration](#).

## 4.2.3 Adding Memory to a Node

Torque caches information about each node, such as the amount of memory a node has. If you add memory to a node, `pbs_server` may not recognize the additional memory. To force Torque to update a node's configuration, do the following:

1. Stop `pbs_server`.
2. Remove the entry for the node from the `nodes` file (`TORQUE_HOME/server_priv/nodes`).
3. Remove the file with the name corresponding to the modified node from the `TORQUE_HOME/server_priv/node_usage` directory.
4. Start `pbs_server`.
5. Add entry for the node back into the `nodes` file.
6. Restart `pbs_server`.

---

### Related Topics

- [3.1 Job Submission](#)

## 4.3 Changing Node State

In this section:

[4.3.1 Marking Jobs Offline](#)

[4.3.2 Listing Node States](#)

[4.3.3 Node Recovery](#)

### 4.3.1 Marking Jobs Offline

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.

### 4.3.2 Listing Node States

Also useful is `pbsnodes -l`, which lists all nodes with an interesting state, such as down, unknown, or offline. This provides a quick glance at nodes that might be having a problem (see [A.8 pbsnodes](#) for details.)

### 4.3.3 Node Recovery

When a MOM gets behind on processing requests, `pbs_server` has a failsafe to allow for node recovery in processing the request backlog. After three failures without having two consecutive successes in servicing a request, `pbs_server` will mark the MOM as offline for five minutes to allow the MOM extra time to process the backlog before it resumes its normal activity. If the MOM has two consecutive successes in responding to network requests before the timeout, then it will come back earlier.

## 4.4 Changing Node Power States

The `pbsnodes -m` command can modify the power state of nodes. Node cannot go from one low-power state to another low-power state. They must be brought up to the Running state and then moved to the new low-power state. The supported power states are:

State	Description
<b>Running</b>	<ul style="list-style-type: none"> <li>Physical machine is actively working</li> <li>Power conservation is on a per-device basis</li> <li>Processor power consumption controlled by P-states</li> </ul>
<b>Standby</b>	<ul style="list-style-type: none"> <li>System appears off</li> <li>Processor halted (OS executes a 'halt' instruction)</li> <li>Processor maintains CPU and system cache state</li> <li>RAM refreshed to maintain memory state</li> <li>Machine in low-power mode</li> <li>Requires interrupt to exit state</li> <li>Lowest-latency sleep state - has no effect on software</li> </ul>
<b>Suspend</b>	<ul style="list-style-type: none"> <li>System appears off</li> <li>Processor and support chipset have no power</li> <li>OS maintains CPU, system cache, and support chipset state in memory</li> <li>RAM in slow refresh</li> <li>Machine in lowest-power state</li> <li>Usually requires specific interrupt (keyboard, mouse) to exit state</li> <li>Third lowest-latency sleep state - system must restore power to processor and support chipset</li> </ul>
<b>Hibernate</b>	<ul style="list-style-type: none"> <li>System is off</li> <li>Physical machine state and memory saved to disk</li> <li>Requires restoration of power and machine state to exit state</li> <li>Second highest-latency sleep state - system performs faster boot using saved machine state and copy of memory</li> </ul>
<b>Shutdown</b>	<ul style="list-style-type: none"> <li>Equivalent to <code>shutdown now</code> command as root</li> </ul>

In order to wake nodes and bring them up to a running state:

- the nodes must support, and be configured to use, Wake-on-LAN (WOL).
- the `pbsnodes` command must report the node's MAC address correctly.

## To Configure Nodes to use Wake-on-LAN

1. Enable WOL in the BIOS for each node. If needed, contact your hardware manufacturer for details.
2. Use the `ethtool` command to determine what types of WOL packets your hardware supports. Torque uses the `g` packet. If the `g` packet is not listed, you cannot use WOL with Torque.

```
[root]# ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full

    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 100Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 2
    Transceiver: internal
    Auto-negotiation: on
    MDI-X: off
    Supports Wake-on: pumbg
    Wake-on: p
    Current message level: 0x00000007 (7)
                           drv probe link

    Link detected: yes
```

This Ethernet interface supports the *g* WOL packet, but is currently set to use the *p* packet.

3. If your Ethernet interface supports the *g* packet, but is configured for a different packet, use `ethtool -s <interface> wol g` to configure it to use *g*:

```
[root]# ethtool -s eth0 wol g
[root]# ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full

    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 100Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 2
    Transceiver: internal
    Auto-negotiation: on
    MDI-X: off
    Supports Wake-on: pumbg
    Wake-on: g
    Current message level: 0x00000007 (7)
                           drv probe link

    Link detected: yes
```

Now the power state of your nodes can be modified and they can be woken up from power-saving states.



## Related Topics

- [A.8 pbsnodes](#)

## 4.5 Host Security

In this section:

- [4.5.1 Enabling PAM with Torque](#)
- [4.5.2 Using PAM Exception Instructions](#)
- [4.5.3 Legacy Torque PAM Configuration](#)

### 4.5.1 Enabling PAM with Torque

Torque is able to take advantage of the authentication services provided through Pluggable Authentication Modules (PAM) to help admins manage access to compute nodes by users. The PAM module available in Torque is located in the PAM security directory. This module, when used in conjunction with other PAM modules, restricts access to the compute node unless the user has a job currently running on the node. The following configurations are examples only. For more information about PAM, see the [PAM \(Pluggable Authentication Modules\) documentation](#) from LinuxDocs.

**i** Security Enhanced Linux (SELinux) must either be disabled or configured to properly work with PAM.

To enable Torque PAM configure Torque using the `--with-pam` option. Using `--with-pam` is sufficient but if your PAM security modules are not in the default `/lib/security` or `/lib64/security` directory, you can specify the location using `--with-pam=<DIR>` where `<DIR>` is the directory where you want the modules to be installed. When Torque is installed the files `pam_pbssimpleauth.la` and `pam_pbssimpleauth.so` appear in `/lib/security`, `/lib64/security`, or the directory designated on the configuration line.

PAM is very flexible and policies vary greatly from one site to another. The following example restricts users trying to access a node using SSH. Administrators need to assess their own installations and decide how to apply the Torque PAM restrictions.

In this example, after installing Torque with PAM enabled, you would add the following two lines to `/etc/pam.d/ssh`:

```
account required pam_pbssimpleauth.so
```

```
account required pam_access.so
```

In `/etc/security/access.conf` make sure all users who access the compute node are added to the configuration. This is an example that allows the users `root`, `george`, `allen`, and `michael` access:

```
 -:ALL EXCEPT root george allen michael torque:ALL
```

With this configuration, if user `george` has a job currently running on the compute node, `george` can use `ssh` to login to the node. If there are currently no jobs running, `george` is disconnected when attempting to login.

Torque PAM is good at keeping users out who do not have jobs running on a compute node. However, it does not have the ability to force a user to log out once they are in. To accomplish this use epilogue or prologue scripts to force users off the system.

## 4.5.2 Using PAM Exception Instructions

PAM exception instructions enable you to configure exceptions to access restrictions. For example, users may be restricted from logging into nodes on which they do not have a running job *unless* they are a member of a group permitted to bypass that restriction.

To configure a bypass group, do the following.

1. Create the bypass group:

```
# groupadd torque-pam-bypass
```

2. Add users to the bypass group:

```
# usermod -G torque-pam-bypass jsmith
```

3. Configure group membership on remote hosts:

```
# pdsh -w ibm[03,04,06,07,15] "usermod -G torque-pam-bypass jsmith"
```

4. Edit `/etc/security/access.conf` to add the group exception *at the end of the file*:

```
# vim /etc/security/access.conf

# -:ALL EXCEPT root rmckay testuser torque:ALL
# --- PAM exception workaround
# -:ALL EXCEPT (torque-pam-bypass):ALL
```

5. Edit `/etc/pam.d/sshd` to configure PAM to allow users to login using SSH only when they have a job running:

```
# vim /etc/pam.d/sshd
```

```
# PAM exception method to allow a non-root pam_access group
account    sufficient    pam_access.so
account    required      pam_nologin.so
account    required      pam_pbssimpleauth.so
```

### 4.5.3 Legacy Torque PAM Configuration

There is an alternative PAM configuration for Torque that can be found in the tarball 'contrib/pam\_authuser.tar.gz'. Adaptive Computing does not currently support this method but the instructions are given here for those who are currently using it and for those who want to use it.

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 authentication 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

1. Untar `contrib/pam_authuser.tar.gz` (found in the `src` tarball).
2. Compile `pam_authuser.c` with `make` and `make install` on every compute node.
3. Edit `/etc/system-auth` as described in `README.pam_authuser`, again on every compute node.
4. 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.

## 4.6 Linux cpuset Support

In this section:

- [4.6.1 cpuset Overview](#)
- [4.6.2 cpuset Support](#)
- [4.6.3 Configuring cpuset](#)
- [4.6.4 cpuset Advantages/Disadvantages](#)

## 4.6.1 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.

**i** cgroups support encompasses and expands upon the cpusets functionality described in this section. See [2.7.1 Torque NUMA-Aware Configuration](#) for details on configuring cgroup support, and for full documentation.

## 4.6.2 cpuset Support

**i** All nodes using cpusets must have the `hwloc` library and corresponding `hwloc-devel` package installed. See [2.1.3 Installing Torque Resource Manager](#) for more information.

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 admin 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.

### 4.6.3 Configuring cpuset

To configure cpuset, do the following.

1. As root, mount the virtual filesystem for cpusets:

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

**i** Do this for each MOM that is to use cpusets.

2. Because cpuset usage is a build-time option in Torque, you must add `--enable-cpuset` to your configure options:

```
./configure --enable-cpuset
```

3. Use this configuration for the MOMs across your system.

### 4.6.4 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, therefore a multi-threaded process could seriously suffer.

---

#### Related Topics

- [4.7.1 Geometry Request Configuration](#)

## 4.7 Scheduling Cores

You can request specific cores on a node at job submission by using geometry requests. To use this feature, specify the `procs_bitmap` resource request of `qsub-l` (see [A.23 qsub](#)) at job submission.

**i** cgroups is incompatible with (and supersedes) cpuset support (`--enable-cpuset` and `--enable-geometry-requests`). Configuring with `--enable-cgroups` overrides these other options. See section [12.2 NUMA-Aware Systems](#) for more information about cgroups and job resource requests.

In this section:

- [4.7.1 Geometry Request Configuration](#)
- [4.7.2 Geometry Request Usage](#)
- [4.7.3 Geometry Request Considerations](#)

## 4.7.1 Geometry Request Configuration

A Linux kernel of 2.6 or later is required to use geometry requests, because this feature uses Linux cpusets in its implementation. In order to use this feature, the cpuset directory has to be mounted. For more information on how to mount the cpuset directory, see [4.6 Linux cpuset Support](#). If the operating environment 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.

**i** The geometry request feature uses a subset of the cpusets feature. When you configure Torque using `--enable-cpuset` and `--enable-geometry-requests` at the same time, and use `-l procs_bitmap=X`, the job will get the requested cpuset. Otherwise, the job is treated as if only `--enable-cpuset` was configured.

**i** cgroups is incompatible with (and supersedes) cpuset support (`--enable-cpuset` and `--enable-geometry-requests`). Configuring with `--enable-cgroups` overrides these other options.

## 4.7.2 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 1s and 0s. A 0 in the bitmap means the job cannot run on the core that matches the 0s 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.

**i** 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.

### 4.7.3 Geometry Request Considerations

As previously stated, 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.

## 4.8 Scheduling Accelerator Hardware

Torque works with accelerators (such as NVIDIA GPUs and Intel MICs) and can collect and report metrics from them or submit workload to them. This feature requires the use of the Moab scheduler. See 'Accelerators' in the *Moab Workload Manager Administrator Guide* for information on configuring accelerators in Torque.

## 4.9 Node Resource Plug-In

There is now an API for creating a resource plug-in to allow the reporting of custom varattrs, generic resources, generic metrics, and node features. Additionally, jobs can be made to report custom resources through the same plug-in. The purpose of this plug-in is to allow some resource integration to happen outside of the normal code release cycle and without having to be part of the main codebase for Torque. This should allow specific sites to implement things that are not of general interest, as well as provide a tight integration option for resources that vary widely based on hardware.

Torque's resource plug-in capability provides an API through which a Torque plug-in can add arbitrary generic resources, generic metrics, varattrs, and features to a node. Additionally, Torque plug-in can add arbitrary resource usage per job.

The API can be found in `trq_plugin_api.h`. To implement a plug-in, you must implement all of the API functions, even if the function does nothing. An implementation that does nothing can be found in `contrib/resource_plugin.cpp`. If you want, you can simply add the desired functionality to this file, build the library, and link it to the MOM at build time.

In this section:

[4.9.1 Plug-In Implementation Recommendations](#)

[4.9.2 Building the Plug-In](#)

[4.9.3 Testing the Plug-In](#)

[4.9.4 Enabling the Plug-In](#)

## 4.9.1 Plug-In Implementation Recommendations

Your plug-in must execute very quickly in order to avoid causing problems for the `pbs_mom` daemon. The node resource portion of the plug-in has a 5 second time limit, and the job resource usage portion has a 3 second time limit. The node resource portion executes each time the MOM sends a status to `pbs_server`, and the job resource usage portion executes once per job at the same time interval. The node resource and job resource portions block `pbs_mom` while they are executing, so they should execute in a short, deterministic amount of time.

Remember, you are responsible for plug-ins, so design well and test thoroughly.

## 4.9.2 Building the Plug-In

If you do not change the name of the `.cpp` file and want to build it, execute the following:

```
export TRQ_HEADER_LOCATION=/usr/local/include/
g++ -fPIC -I $TRQ_HEADER_LOCATION resource_plugin.cpp -shared -o libresource_plugin.so
```

**Note:** Change `TRQ_HEADER_LOCATION` if you configured torque with the `--prefix` option.

## 4.9.3 Testing the Plug-In

**Note:** You assume all responsibility for any plug-ins. This document is intended to assist you in testing the plug-ins, but this list of suggested tests may not be comprehensive. We do



not assume responsibility if these suggested tests do not cover everything.

In this topic:

[4.9.3.A Testing Basic Functionality](#)

[4.9.3.B Testing for Memory Leaks](#)

### 4.9.3.A Testing Basic Functionality

Once you've implemented and built your library, you can begin testing. For your convenience, a simple test driver can be found in `plugin_driver.cpp`. You can build this executable and link it against your library as shown below in order to manually verify the output:

```
export PLUGIN_LIB_PATH=/usr/local/lib
g++ plugin_driver.cpp -I $TRQ_HEADER_LOCATION -L $PLUGIN_LIB_PATH -lresource_plugin -o driver
```

You can then execute the driver and manually inspect the output:

```
./driver
```

**Note:** Change `PLUGIN_LIB_PATH` if you have installed the plug-in somewhere other than `/usr/local/lib`.

To illustrate output, a simple plug-in that reports:

- 2 broams of stormlight used, ignoring the random process ID found by the driver
- 1024 hbmemb for GRES
- temperature of 75.2 for GMETRICS
- octave = 3.2.4 for VARATTRS
- haswell for features

will have the output:

```
$ ./driver
Your plugin reported the following for the random pid 7976:
stormlight = 2broams
Your plugin reports the following for this host:
  GRES:
    hbmemb = 1024

  GMETRICS:
    temperature = 75.20

  VARATTRS:
    octave = 3.2.4

  FEATURES: haswell
```

### 4.9.3.B Testing for Memory Leaks

In order to prevent your compute nodes from being compromised for speed or even going down due to out-of-memory conditions, you should run your plug-in under valgrind to test that it is correctly managing memory.

Assuming you are executing the driver from the 'Testing Basic Functionality' section, you can run:

```
valgrind --tool=memcheck --leak-check=full --log-file=plugin_valgrind_output.txt
./driver
```

If you are not familiar with valgrind, a good primer can be found at [The Valgrind Quick Start Guide](#).

We recommend that you fix all errors reported by valgrind.

### 4.9.4 Enabling the Plug-In

Once you've implemented, built, and thoroughly tested your plug-in (remember that our suggestions may not address everything), you will want to enable it in Torque. Your plug-in can be linked in at build time:

```
./configure <your other options> --with-resource-plugin=<path to your resource plugin>
```

**Note:** You will want to make sure that the path you specify is in `$LD_LIBRARY_PATH`, or can otherwise be found by `pbs_mom` when you start the daemon.

Once you build, you can then start the new MOM and be able to observe the plug-in's output using `pbsnodes`, `qstat -f`, and in the accounting file.

Sample `pbsnodes` output:

```
<normal output>
gres:hbmem = 20
gmetric:temperature = 76.20
varattr:octave = 3.2.4
features = haswell
```

The keywords at the front let Moab know what each line means, so it can use them accordingly.

Sample accounting file entry:

```
<normal entry until resources used> resources_used.cput=0
resources_used.energy_used=0 resources_used.mem=452kb
resources_used.vmem=22372kb resources_used.walltime=00:05:00
resources_used.stormlight=2broams
```

Your plug-in resources reported will appear in the form:

```
resources_used.<name you supplied>=<value you supplied>
```

The above example includes the arbitrary resource stormlight and the value of 2broams.

Sample *qstat -f* output:

```
<normal qstat -f output>  
resources_used.stormlight = 2broams
```

The *resources used* reported by the plug-in will appear at the end of the *qstat -f* output in the same format as in the accounting file.

## Chapter 5: Setting Server Policies

This section explains how to set up and configure your queue. This section also explains how to set up Torque to run in high availability mode.

In this chapter:

- [5.1 Queue Configuration](#)
- [5.2 Server High Availability](#)
- [5.3 Setting `min\_threads` and `max\_threads`](#)

### 5.1 Queue Configuration

To initially define a queue, use the `create` subcommand of `qmgr`:

```
> 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`.

```
> qmgr -c "set queue batch started=true"
> qmgr -c "set queue batch enabled=true"
```

Further configuration is possible using any combination of the following attributes.

Boolean attributes *T*, *t*, *1*, *Y*, and *y* are all synonymous with 'TRUE,' and *F*, *f*, *0*, *N*, and *n* all mean 'FALSE.' *E* and *R* are synonymous with 'Execution' and 'Routing' (respectively).

In this section:

- [5.1.1 Example Queue Configuration](#)
- [5.1.2 Setting Queue Resource Controls](#)
- [5.1.3 Setting a Default Queue](#)
- [5.1.4 Mapping a Queue to Subset of Resources](#)
- [5.1.5 Creating a Routing Queue](#)

## Related Topics

- [Appendix B: Server Parameters](#)
- [Appendix N: Queue Attributes](#)
- [A.9 qalter](#) - command that can move jobs from one queue to another

### 5.1.1 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.

**i** See [5.1.2 Setting Queue Resource Controls](#) for more information about setting queue resource requirements and the use of `-l` and `-L` job submission syntaxes.

### 5.1.2 Setting Queue Resource Controls

#### Setting Queue Resource Controls with Resource Request Syntax 2.0

Using the `-L` syntax, you can set default, minimum, and maximum values for lprocs, memory, swap, disk, sockets, numanode, cores and threads with resource request 2.0.

These can be set in the general format:

```
qmgr -c "set queue <queue_name> req_information_[default|min|max].
[lprocs|memory|swap|disk|sockets|numanode|core|thread]"
```

#### Example 5-1: Jobs using `-L` syntax

```
qmgr -c "set queue q1 req_information_default.lprocs=2"
qmgr -c "set queue q1 req_information_minimum.memory=2gb"
qmgr -c "set queue q1 req_information_maximum.core=10"
```



Regarding queue defaults and the newer `-L` NUMA-aware syntax: with *only* the newer `req_information_default.<attribute>` configured on a queue, the queue will only accept submissions with the `-L` syntax. The same holds true for `resources_default.<attribute>` and `-l` submissions. Setting both on a queue (as in [Example 5-2](#) below) enables the queue to accept job submissions with either syntax.

#### Example 5-2: Jobs using `-L` or `-l` syntax

This example shows how to enable a queue to be able to accept *both* kinds of jobs and still be able to enforce defaults:

```
qmgr -c "create queue batch"
qmgr -c "set queue batch queue_type = Execution"
qmgr -c "set queue batch resources_default.mem = 3gb"
qmgr -c "set queue batch enabled = True"
qmgr -c "set queue batch started = True"
qmgr -c "set queue batch req_information_default.memory = 3gb"
```

In this example, jobs submitted that explicitly use the `-L` syntax will have the `req_information_default.memory` setting applied. If the job does *not* explicitly use this syntax, then the `resources_default.mem` setting will be applied.

### 5.1.3 Setting a Default Queue

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

```
qmgr -c "set server default_queue=batch"
```

### 5.1.4 Mapping a Queue to 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
```

**i** 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.

## 5.1.5 Creating a Routing Queue

A routing queue will steer a job to a destination queue based on job attributes and queue constraints. It is set up by creating a queue of `queue_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

# queue for jobs using 32+ nodes
create queue reg_64
set queue reg_64 queue_type = Execution
set queue reg_64 resources_min.ncpus = 63
set queue reg_64 resources_min.nodes = 32
set queue reg_64 resources_default.walltime = 06:00:00
```

```

set queue reg_64 enabled = True
set queue reg_64 started = True

# have all jobs go through the routing queue
set server default_queue = route
set server resources_default.ncpus = 1
set server resources_default.walltime = 24:00:00
...

```

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.



You should not use a Torque routing queue in conjunction with Moab remap classes to route jobs to queues/nodes. You should select only one of the two methods.

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.nodect = 31

```

Torque waits to apply the server and queue defaults until the job is assigned to its final execution queue. Queue defaults override the server defaults. 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.



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'`).



## Related Topics

- [Appendix N: Queue Attributes](#)

## 5.2 Server High Availability

Torque can run in a redundant or high availability mode. This means that there can be multiple instances of the server running and waiting to take over processing in the event that the primary server fails.

In this section:

- [5.2.1 Redundant Server Host Machines](#)
- [5.2.2 Enabling High Availability](#)
- [5.2.3 Enhanced High Availability with Moab](#)
- [5.2.4 How Commands Select the Correct Server Host](#)
- [5.2.5 Job Names](#)
- [5.2.6 Persistence of the pbs\\_server Process](#)
- [5.2.7 High Availability of the NFS Server](#)
- [5.2.8 Installing Torque in High Availability Mode](#)
- [5.2.9 Installing Torque in High Availability Mode on Headless Nodes](#)
- [5.2.10 Example Setup of High Availability](#)

### 5.2.1 Redundant Server Host Machines

High availability enables Moab to continue running even if `pbs_server` is brought down. This is done by running multiple copies of `pbs_server`, each of which has its `TORQUE_HOME/server_priv` directory mounted on a shared file system.

**i** Do not use symlinks when sharing the Torque home directory or `server_priv` directories. A workaround for this is to use `mount --rbind /path/to/share /var/spool/torque`. Also, it is highly recommended that you only share the `server_priv` directory and not the entire `TORQUE_HOME` directory.

The `server_name` file (for *all* servers and compute nodes) must include the names of every `pbs_server` host. The syntax of the `server_name` file is a comma-delimited list of host names. For example:

```
host1,host2,host3
```

**i** When configuring high availability, do not use `$pbsserver` in the `pbs_mom` configuration file (`TORQUE_HOME/mom_priv/config` on the compute nodes) to specify the server host names. You must use the `TORQUE_HOME/server_name` file.

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. Each subsequent server to start will block very early upon startup when it tries to lock the `TORQUE_HOME/server_priv/server.lock` file. When the 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 any server starting after the first would always write an error and quit when it could not obtain the lock.

## 5.2.2 Enabling High Availability

To use high availability, you must start each instance of `pbs_server` with the `--ha` option.

Three server options help manage high availability. The server parameters are `lock_file`, `lock_file_update_time`, and `lock_file_check_time`.

The `lock_file` option enables the admin to change the location of the lock file (default location: `TORQUE_HOME/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`.

**i** 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.

**i** Do not use anything but a simple NFS fileshare that is not used by anything else (i.e., only Moab can use the fileshare).

**i** Do not use a general-purpose NAS, parallel file system, or company-wide shared infrastructure to set up Moab high availability using 'native' high availability.

### 5.2.3 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 `-l` option as demonstrated in the example below (the port is required and the default is 15004):

Set the `PBS_ARGS` environment variable in the `/etc/sysconfig/pbs_server` file. Set `PBS_ARGS=-l <moabhost:port>` where `moabhost` is the name of the alternate server node and `port` is the port on which Moab on the alternate server node is listening (default 15004).

If Moab is running in HA mode, set the `-l` option for each redundant server:

Set the `PBS_ARGS` environment variable in the `/etc/sysconfig/pbs_server` file to `PBS_ARGS=-l <moabhost:port> -l <moabhost2:port>`.

If `pbs_server` and Moab run on the same host, use the `--ha` option as demonstrated in the example below:

Set the `PBS_ARGS` environment variable in the `/etc/sysconfig/pbs_server` file to `PBS_ARGS=--ha`.

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.

### 5.2.4 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 `PBS_DEFAULT` environment variable or from the `TORQUE_HOME/server_name`.

**i** `PBS_DEFAULT` overrides the value in the `server_name` file.

Whenever a Torque client command executes with no explicit server mentioned, it attempts to connect to the first server name listed in `PBS_DEFAULT` or `TORQUE_HOME/server_name`. If this fails, the command tries the next server name. 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.

### 5.2.5 Job Names

Job names normally contain the name of the host machine where `pbs_server` is running. When job names are constructed, only the server name in `$PBS_DEFAULT` or the first name from the server specification list, `TORQUE_HOME/server_name`, is used in building the job name.

### 5.2.6 Persistence of the `pbs_server` Process

The system administrator 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_servers` in the process table and starts some more if needed.

### 5.2.7 High Availability of the NFS Server

**i** Before installing a specific NFS HA solution contact Adaptive Computing Support for a detailed discussion on NFS HA type and implementation path.

One consideration of this implementation 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](#)
- [Sourceforge Linux NFS FAQ](#)
- [CentOS Documentation Home](#)
- [SUSE Linux Enterprise Administration Guide: Sharing File Systems with NFS](#)

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

- [Red Hat Enterprise Linux: Chapter 1. High Availability Add-On Overview](#)

## 5.2.8 Installing Torque in High Availability Mode

The following procedure demonstrates a Torque installation in high availability (HA) mode.

### Requirements

- gcc (GCC) 4.1.2
- BASH shell
- Servers configured the following way:
  - 2 main servers with identical architecture:
    - `server1` — Primary server running Torque with a shared file system (this example uses NFS)
    - `server2` — Secondary server running with Torque with a shared file system (this example uses NFS)
  - `fileServer` — Shared file system (this example uses NFS)
  - Compute nodes

### To Install Torque in HA Mode

1. Stop all firewalls or update your firewall to allow traffic from Torque services:

```
> systemctl stop firewalld
> systemctl disable firewalld
```

If you are unable to stop the firewall due to infrastructure restriction, open the following ports:

- 15001[tcp,udp]
- 15002[tcp,udp]
- 15003[tcp,udp]

2. Disable SELinux:

```
> vi /etc/sysconfig/selinux
SELINUX=disabled
```

3. Update your main `~/.bashrc` profile to ensure you are always referencing the applications to be installed on all servers:

```
Torque
```

```
export TORQUE_HOME=/var/spool/torque

# Library Path

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${TORQUE_HOME}/lib

# Update system paths
export PATH=${TORQUE_HOME}/bin:${TORQUE_HOME}/sbin:${PATH}
```

4. Verify `server1` and `server2` are resolvable via either DNS or looking for an entry in the `/etc/hosts` file.
5. Configure the NFS Mounts by following these steps.

- a. Create mount point folders on `fileServer`:

```
fileServer# mkdir -m 0755 /var/spool/torque
fileServer# mkdir -m 0750 /var/spool/torque/server_priv
```

- b. Update `/etc/exports` on `fileServer` (the IP addresses should be that of `server2`):

```
/var/spool/torque/server_priv 192.168.0.0/255.255.255.0(rw,sync,no_root_squash)
```

- c. Update the list of NFS exported file systems:

```
fileServer# exportfs -r
```

6. If the NFS daemons are not already running on `fileServer`, start them:

```
> systemctl restart rpcbind.service
> systemctl start nfs-server.service
> systemctl start nfs-lock.service
> systemctl start nfs-idmap.service
```

7. Mount the exported file systems on `server1` by following these steps.

- a. Create the directory reference and mount them:

```
server1# mkdir /var/spool/torque/server_priv
```

Repeat this process for `server2`.

- b. Update `/etc/fstab` on `server1` to ensure that NFS mount is performed on startup:

```
fileServer:/var/spool/torque/server_priv /var/spool/torque/server_priv nfs
rsize= 8192,wsiz=8192,timeo=14,intr
```

Repeat this step for `server2`.

8. Install Torque by following these steps.

- a. Download and extract the latest Torque build from [Adaptive Computing Product Download](#).
- b. Navigate to the Torque directory and compile Torque on server1:

```
server1# configure
server1# make
server1# make install
server1# make packages
```

- c. If the installation directory is shared on both head nodes, then run `make install` on server1:

```
server1# make install
```

If the installation directory is not shared, repeat step 8a-b (downloading and installing Torque) on server2.

9. Start `trqauthd`:

```
server1# systemctl start trqauthd
```

10. Configure Torque for HA.

- a. List the host names of all nodes that run `pbs_server` in the `TORQUE_HOME/server_name` file. You must also include the host names of all nodes running `pbs_server` in the `TORQUE_HOME/server_name` file of each MOM node. The syntax of `TORQUE_HOME/server_name` is a comma-delimited list of host names.

```
server1,server2
```

- b. Create a simple queue configuration for Torque job queues on server1:

```
server1# pbs_server -t create
server1# qmgr -c "set server scheduling=true"
server1# qmgr -c "create queue batch queue_type=execution"
server1# qmgr -c "set queue batch started=true"
server1# qmgr -c "set queue batch enabled=true"
server1# qmgr -c "set queue batch resources_default.nodes=1"
server1# qmgr -c "set queue batch resources_default.walltime=3600"
server1# qmgr -c "set server default_queue=batch"
```



Because `server_priv/*` is a shared drive, you do not need to repeat this step on server2.

- c. Add the root users of Torque to the Torque configuration as an operator and manager:

```
server1# qmgr -c "set server managers += root@server1"
```

```
server1# qmgr -c "set server managers += root@server2"
server1# qmgr -c "set server operators += root@server1"
server1# qmgr -c "set server operators += root@server2"
```

**i** Because `server_priv/*` is a shared drive, you do not need to repeat this step on Server 2.

- d. You must update the lock file mechanism for Torque in order to determine which server is the primary. To do so, use the `lock_file_update_time` and `lock_file_check_time` parameters. The primary `pbs_server` will update the lock file based on the specified `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.

```
server1# qmgr -c "set server lock_file_check_time=5"
server1# qmgr -c "set server lock_file_update_time=3"
```

**i** Because `server_priv/*` is a shared drive, you do not need to repeat this step on server2.

- e. List the servers running `pbs_server` in the `Torqueacl_hosts` file:

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

**i** Because `server_priv/*` is a shared drive, you do not need to repeat this step on server2.

- f. Restart the running `pbs_server` in HA mode:

```
systemctl stop pbs_server
```

- g. Start the `pbs_server` on the secondary server:

```
systemctl start pbs_server
```

11. Check the status of Torque in HA mode:

```
server1# qmgr -c "p s"
server2# qmgr -c "p s"
```

The commands above return all settings from the active Torque server from either node.



- a. Stop one of the `pbs_servers` to verify that the secondary server picks up the request:

```
systemctl stop pbs_server
```

- b. Stop the `pbs_server` on `server2` and restart `pbs_server` on `server1` to verify that both nodes can handle a request from the other.

## 12. Install a `pbs_mom` on the compute nodes.

- a. Copy the install scripts to the compute nodes and install.
- b. Navigate to the shared source directory of Torque and run the following:

```
node1 torque-package-mom-linux-x86_64.sh --install
node2 torque-package-clients-linux-x86_64.sh --install
```

Repeat this for each compute node. Verify that the `/var/spool/torque/server_name` file shows all your compute nodes.

- c. On `server1` or `server2`, configure the nodes file to identify all available MOMs. To do so, edit the `/var/spool/torque/server_priv/nodes` file.

```
node1 np=2
node2 np=2
```

 Change the `np` flag to reflect number of available processors on that node.

- d. Recycle the `pbs_servers` to verify that they pick up the MOM configuration:

```
systemctl stop pbs_server
```

- e. Start the `pbs_mom` on each execution node:

```
systemctl start pbs_mom
```

## 5.2.9 Installing Torque in High Availability Mode on Headless Nodes

The following procedure demonstrates a Torque installation in high availability (HA) mode on nodes with no local hard drive.

### Requirements

- gcc (GCC) 4.1.2
- BASH shell
- Servers (these cannot be two VMs on the same hypervisor) configured the following way:

- 2 main servers with identical architecture
  - `server1` — Primary server running Torque with a file system share (this example uses NFS)
  - `server2` — Secondary server running with Torque with a file system share (this example uses NFS)
- Compute nodes
- `fileServer` — A shared file system server (this example uses NFS)

## To Install Torque in HA Mode on a Node with no Local Hard Drive

1. Stop all firewalls or update your firewall to allow traffic from Torque services:

```
> systemctl stop firewalld
> systemctl disable firewalld
```

If you are unable to stop the firewall due to infrastructure restriction, open the following ports:

- 15001[tcp,udp]
- 15002[tcp,udp]
- 15003[tcp,udp]

2. Disable SELinux:

```
> vi /etc/sysconfig/selinux

SELINUX=disabled
```

3. Update your main `~/ .bashrc` profile to ensure you are always referencing the applications to be installed on all servers:

```
# Torque
export TORQUE_HOME=/var/spool/torque

# Library Path

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${TORQUE_HOME}/lib

# Update system paths
export PATH=${TORQUE_HOME}/bin:${TORQUE_HOME}/sbin:${PATH}
```

4. Verify `server1` and `server2` are resolvable via either DNS or looking for an entry in the `/etc/hosts` file.
5. Configure the NFS Mounts by following these steps.

- a. Create mount point folders on fileServer:

```
fileServer# mkdir -m 0755 /var/spool/torque
```

- b. Update /etc/exports on fileServer. The IP addresses should be that of server2.

```
/var/spool/torque/ 192.168.0.0/255.255.255.0(rw,sync,no_root_squash)
```

- c. Update the list of NFS exported file systems:

```
fileServer# exportfs -r
```

6. If the NFS daemons are not already running on fileServer, start them:

```
> systemctl restart rpcbind.service
> systemctl start nfs-server.service
> systemctl start nfs-lock.service
> systemctl start nfs-idmap.service
```

7. Mount the exported file systems on server1 by following these steps.

- a. Create the directory reference and mount them:

```
server1# mkdir /var/spool/torque
```

Repeat this process for server2.

- b. Update /etc/fstab on server1 to ensure that NFS mount is performed on startup:

```
fileServer:/var/spool/torque/server_priv /var/spool/torque/server_priv nfs
rsize= 8192,wsiz=8192,timeo=14,intr
```

Repeat this step for server2.

8. Install Torque by following these steps.

- a. Download and extract the latest Torque build from [Adaptive Computing Torque Downloads](#).

- b. Navigate to the Torque directory and compile Torque with the HA flag on server1:

```
server1# configure --prefix=/var/spool/torque
server1# make
server1# make install
server1# make packages
```

- c. If the installation directory is shared on both head nodes, then run make install on server1:

```
server1# make install
```

If the installation directory is not shared, repeat step 8a-b (downloading and installing Torque) on server2.

9. Start trqauthd:

```
server1# systemctl start trqauthd
```

10. Configure Torque for HA.

- a. List the host names of all nodes that run `pbs_server` in the `TORQUE_HOME/server_name` file. You must also include the host names of all nodes running `pbs_server` in the `TORQUE_HOME/server_name` file of each MOM node. The syntax of `TORQUE_HOME/server_name` is a comma-delimited list of host names.

```
server1,server2
```

- b. Create a simple queue configuration for Torque job queues on server1:

```
server1# pbs_server -t create
server1# qmgr -c "set server scheduling=true"
server1# qmgr -c "create queue batch queue_type=execution"
server1# qmgr -c "set queue batch started=true"
server1# qmgr -c "set queue batch enabled=true"
server1# qmgr -c "set queue batch resources_default.nodes=1"
server1# qmgr -c "set queue batch resources_default.walltime=3600"
server1# qmgr -c "set server default_queue=batch"
```

**i** Because `TORQUE_HOME` is a shared drive, you do not need to repeat this step on server2.

- c. Add the root users of Torque to the Torque configuration as an operator and manager:

```
server1# qmgr -c "set server managers += root@server1"
server1# qmgr -c "set server managers += root@server2"
server1# qmgr -c "set server operators += root@server1"
server1# qmgr -c "set server operators += root@server2"
```

**i** Because `TORQUE_HOME` is a shared drive, you do not need to repeat this step on server2.

- d. You must update the lock file mechanism for Torque in order to determine which server is the primary. To do so, use the `lock_file_update_time` and `lock_file_check_time` parameters. The primary `pbs_server` will update the lock file based on the specified `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.

```
server1# qmgr -c "set server lock_file_check_time=5"
server1# qmgr -c "set server lock_file_update_time=3"
```

**i** Because `TORQUE_HOME` is a shared drive, you do not need to repeat this step on `server2`.

- e. List the servers running `pbs_server` in the `Torqueacl_hosts` file:

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

**i** Because `TORQUE_HOME` is a shared drive, you do not need to repeat this step on `server2`.

- f. Restart the running `pbs_server` in HA mode:

```
systemctl stop pbs_server
```

- g. Start the `pbs_server` on the secondary server:

```
systemctl start pbs_server
```

11. Check the status of Torque in HA mode:

```
server1# qmgr -c "p s"
server2# qmgr -c "p s"
```

The commands above return all settings from the active Torque server from either node.

- a. Stop one of the `pbs_servers` to verify that the secondary server picks up the request:

```
systemctl stop pbs_server
```

- b. Stop the `pbs_server` on `server2` and restart `pbs_server` on `server1` to verify that both nodes can handle a request from the other.

12. Install a `pbs_mom` on the compute nodes.

- a. On `server1` or `server2`, configure the `nodes` file to identify all available MOMs. To do so, edit the `/var/spool/torque/server_priv/nodes` file.

```
node1 np=2
node2 np=2
```

**i** Change the `np` flag to reflect number of available processors on that node.

- b. Recycle the `pbs_servers` to verify that they pick up the MOM configuration:

```
systemctl stop pbs_server
```

**i** You can specify command line arguments for `pbs_server` using the `PBS_ARGS` environment variable in the `/etc/sysconfig/pbs_server` file. Set `PBS_ARGS=--ha -l <host>:<port>` where `<host>` is the name of the alternate server node and `<port>` is the port on which `pbs_server` on the alternate server node is listening (default 15004).

- c. Start the `pbs_mom` on each execution node:

```
systemctl start pbs_mom
```

## 5.2.10 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:

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

4. Start `pbs_server`:

```
[root@server1]$ systemctl start pbs_server
[root@server2]$ systemctl start pbs_server
```

## 5.3 Setting `min_threads` and `max_threads`

There are two threadpools in Torque, one for background tasks and one for incoming requests from the MOMs and through the API (client commands, Moab, and so forth). The `min_threads` and `max_threads` parameters control the number of total threads used for both, not for each individually. The incoming requests' threadpool has three-quarters of

`min_threads` for its minimum, and three-quarters of `max_threads` for its maximum, with the background pool receiving the other one-quarter.

Additionally, `pbs_server` no longer allows incoming requests to pile up indefinitely. When the threadpool is too busy for incoming requests, it indicates such, returning `PBSE_SERVER_BUSY` with the accompanying message that "Pbs Server is currently too busy to service this request. Please retry this request." The threshold for this message, if the request is from a manager, is that at least two threads be available in the threadpool. If the request comes from a non-manager, 5% of the threadpool must be available for the request to be serviced. Note that availability is calculated based on the maximum threads and not based on the current number of threads allocated.

If an undesirably large number of requests are given a busy response, one option is to increase the number of maximum threads for the threadpool. If the load on the server is already very high, then this is probably not going to help, but if the CPU load is lower, then it may help. Remember that by default the threadpool shrinks down once the extra threads are no longer needed. This is controlled via the `thread_idle_seconds` server parameter.

**i** Any change in the `min_threads`, `max_threads`, or `thread_idle_seconds` parameters requires a restart of `pbs_server` to take effect.

## Chapter 6: 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 you are using Maui or Moab, see 'Moab-Torque Integration Guide' in the *Moab Workload Manager Administrator Guide*. If using `pbs_sched`, simply start the `pbs_sched` daemon.

**i** If you are installing Moab Cluster Manager, Torque and Moab were configured at installation for interoperability and no further action is required.



## Chapter 7: Configuring Data Management

This chapter provides instructions to configure Torque for data management purposes. For example, if you want to copy stdout and stderr files back to the submit host.

In this chapter:

[7.1 SCP Setup](#)

[7.2 NFS and Other Networked Filesystems](#)

[7.3 File stage-in/stage-out](#)

### 7.1 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.

To set up Torque for SCP, follow the directions in each of the topics in this section.

In this section:

[7.1.1 Generating SSH Key on Source Host](#)

[7.1.2 Copying Public SSH Key to Each Destination Host](#)

[7.1.3 Configuring the SSH Daemon on Each Destination Host](#)

[7.1.4 Validating Correct SSH Configuration](#)

[7.1.5 Enabling Bi-Directional SCP Access](#)

[7.1.6 Troubleshooting](#)

### 7.1.1 Generating 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/`.

### 7.1.2 Copying 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
```

### 7.1.3 Configuring 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, see [7.1.4 Validating Correct SSH Configuration](#) 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
```

### 7.1.4 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
```

**i** 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`.

### 7.1.5 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).

### 7.1.6 Troubleshooting

If, after following all of the instructions in this section ([7.1 SCP Setup](#)), 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.

## 7.2 NFS and Other Networked Filesystems

When a batch job starts, its `stdin` file (if specified) is copied from the submission directory on the remote submission host to the `TORQUE_HOME/spool` directory (e.g., `/var/spool/torque/spool` on the execution host (or 'mother superior')).

When the job completes, the MOM copies the files back to the submission directory on the remote submit host. The file copy happens using a remote copy facility such as `rcp` (default) 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 `TORQUE_HOME/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 cannot copy the output or error files to the submission directory, it copies them to `TORQUE_HOME/undelivered` on that host.

## 7.3 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_destination.txt
```

```
$ 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
```

## Chapter 8: MPI (Message Passing Interface) Support

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.

In this chapter:

[8.1 MPICH](#)

[8.2 Open MPI](#)

### 8.1 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.

In this section:

[8.1.1 MPIExec Overview](#)

[8.1.2 MPIExec Troubleshooting](#)

[8.1.3 General MPI Troubleshooting](#)

#### 8.1.1 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.

For more information, see the [mpiexec](#) homepage.

## 8.1.2 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](mailto: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.

### 8.1.3 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 (see [Appendix G: Prologue and Epilogue Scripts](#)).

#### Related Topics

- [3.8 Torque Process Tracking](#)

## 8.2 Open MPI

**Open MPI** is an implementation that combines technologies from multiple projects to create the best possible library. It supports the TM interface for integration with Torque. More information is available in the [Open MPI FAQ](#).

### TM Aware

To make use of Moab's TM interface, MPI must be configured to be TM aware.

Use these guidelines:

1. If you have installed from source, you need to use `./configure --with-tm` when you configure and make `openmpi`.
2. Run `mpirun` *without* the `-machinefile`. Moab will copy down the environment PATH and Library path down to each sister MOM. If `-machinefile` is used, `mpirun` will bypass the TM interface.

#### Example 8-1: Without TM Aware

```
[jbooth@support-mpil ~]$ /usr/lib64/openmpi/bin/mpirun -np 4 -machinefile $PBS_
NODEFILE echo.sh
=====
support-mpil
=====
/usr/lib64/openmpi/bin:/usr/lib64/openmpi/bin:/usr/lib64/qt-
3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/lib64/openmpi/lib:/usr/lib64/openmpi/lib

=====
support-mpil
=====
/usr/lib64/openmpi/bin:/usr/lib64/openmpi/bin:/usr/lib64/qt-
```



```

3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/lib64/openmpi/lib:/usr/lib64/openmpi/lib

=====
support-mpi2
=====
/usr/lib64/openmpi/bin:/usr/lib64/openmpi/bin:/usr/lib64/qt-
3.3/bin:/usr/local/bin:/bin:/usr/bin

/usr/lib64/openmpi/lib:/usr/lib64/openmpi/lib:

=====
support-mpi2
=====
/usr/lib64/openmpi/bin:/usr/lib64/openmpi/bin:/usr/lib64/qt-
3.3/bin:/usr/local/bin:/bin:/usr/bin

/usr/lib64/openmpi/lib:/usr/lib64/openmpi/lib:

```

The paths, /opt/moab/bin and /opt/moab/sbin, were not passed down to the sister MOMs.

### Example 8-2: With TM Aware

```

[jbooth@support-mpi1 ~]$ /usr/local/bin/mpirun -np 4 echo.sh
=====
support-mpi1
=====
/usr/local/bin:/usr/local/bin:/usr/lib64/qt-
3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/local/lib:/usr/local/lib

=====
support-mpi1
=====
/usr/local/bin:/usr/local/bin:/usr/lib64/qt-
3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/local/lib:/usr/local/lib

=====
support-mpi2
=====
/usr/local/bin:/usr/local/bin:/usr/local/bin:/usr/lib64/qt-
3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/local/lib:/usr/local/lib:/usr/local/lib

=====
support-mpi2
=====
/usr/local/bin:/usr/local/bin:/usr/local/bin:/usr/lib64/qt-
3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/opt/mo
ab/bin:/opt/moab/sbin:/home/jbooth/bin

/usr/local/lib:/usr/local/lib:/usr/local/lib

```

The paths, /opt/moab/bin and /opt/moab/sbin, were passed down to the sister MOMs.

---

### Related Topics

- [3.8 Torque Process Tracking](#)

## Chapter 9: Resources

In this chapter:

[9.1 About Resources](#)

[9.2 Configuration](#)

[9.3 Utilization](#)

[9.4 Node States](#)

### 9.1 About Resources

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
- Node States

### 9.2 Configuration

Configuration includes both detected hardware configuration and specified batch attributes.

Attribute	Description	Details
<b>Architecture (arch)</b>	operating system of the node	The value reported is a derivative of the operating system installed.
<b>Node Features (properties)</b>	arbitrary string attributes associated with the node	No node features are specified by default. If required, they are set using the <code>nodes</code> file located in the <code>TORQUE_HOME/server_priv</code> directory. They can specify any string and are most commonly used to allow users to request

Attribute	Description	Details
		certain subsets of nodes when submitting jobs.
<b>Local Disk (size)</b>	configured local disk	By default, local disk space is not monitored. If the MOM configuration <code>size[fs=&lt;FS&gt;]</code> parameter is set, Torque will report, in kilobytes, configured disk space within the specified directory.
<b>Memory (physmem)</b>	local memory/RAM	Local memory/RAM is monitored and reported in kilobytes.
<b>Processors (ncpus/np)</b>	real/virtual processors	The number of processors detected by Torque is reported via the <code>ncpus</code> 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 can be configured using the nodes file located in the <code>TORQUE_HOME/server_priv</code> 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 <code>maxload</code> limit is reached.
<b>Swap (totmem)</b>	virtual memory/Swap	Virtual memory/Swap is monitored and reported in kilobytes.

## 9.3 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
<b>Disk (size)</b>	local disk availability	By default, local disk space is not monitored. If the MOM configuration <code>size[fs=&lt;FS&gt;]</code> parameter is set, Torque will report configured and currently available disk space within the specified directory in kilobytes.
<b>Memory (availmem)</b>	real memory/RAM	Available real memory/RAM is monitored and reported in kilobytes.

Attribute	Description	Details
<b>Network (netload)</b>	local network adapter usage	Reports total number of bytes transferred in or out by the network adapter.
<b>Processor Utilization (loadave)</b>	node's cpu load average	Reports the node's 1 minute BSD load average.

## 9.4 Node States

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

Attribute	Description	Details
<b>Idle Time (idletime)</b>	time since local keyboard/mouse activity has been detected	Time in seconds since local keyboard/mouse activity has been detected.
<b>State (state)</b>	monitored/admin node state	<p>A node can be in one or more of the following states:</p> <ul style="list-style-type: none"> <li>• <i>busy</i> - node is full and will not accept additional work</li> <li>• <i>down</i> - node is failing to report, is detecting local failures with node</li> <li>• <i>free</i> - node is ready to accept additional work</li> <li>• <i>job-exclusive</i> - all available virtual processors are assigned to jobs</li> <li>• <i>job-sharing</i> - node has been allocated to run multiple shared jobs and will remain in this state until jobs are complete</li> <li>• <i>offline</i> - node has been instructed by an admin to no longer accept work</li> <li>• <i>reserve</i> - node has been reserved by the server</li> <li>• <i>time-shared</i> - node always allows multiple jobs to run concurrently</li> <li>• <i>unknown</i> - node has not been detected</li> </ul>

## Chapter 10: Accounting Records

In this chapter:

[10.1 Location and Contents](#)

[10.2 Record Types](#)

[10.3 Accounting Variables](#)

[10.4 Fields](#)

### 10.1 Location and Contents

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

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

`$TORQUEROOT` defaults to `/var/spool/torque` and `<TIMESTAMP>` is in the format: `YYYYMMDD`.

These records include events, time stamps, and information on resources requested and used.

For sites running Moab, and using the Torque accounting records, we recommend that Moab is configured to sync the job IDs. If the IDs are not synced, the Moab job ID will not be known from these accounting files. The Moab job ID can be obtained from the Moab event logs (in `$MOABHOMEDIR/stats`), but that requires extra effort. For additional information, see the instructions ‘Synchronizing Job IDs in Torque and Moab’ in the *Moab Workload Manager Administrator Guide*, section ‘Resource Manager Configuration’.

### 10.2 Record Types

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

Record Marker	Record Type	Description
<b>A</b>	abort	Job has been aborted by the server
<b>C</b>	checkpoint	Job has been checkpointed and held
<b>D</b>	delete	Job has been deleted

Record Marker	Record Type	Description
<b>E</b>	exit	Job has exited (either successfully or unsuccessfully)
<b>Q</b>	queue	Job has been submitted/queued
<b>R</b>	rerun	Attempt to rerun the job has been made
<b>S</b>	start	Attempt to start the job has been made (if the job fails to properly start, it may have multiple job start records)
<b>T</b>	restart	Attempt to restart the job (from checkpoint) has been made (if the job fails to properly start, it may have multiple job start records)

## 10.3 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 [Job Submission](#).

**i** Jobs submitted with the `-L` request syntax will have the `-L` submission recorded in the accounting log.

Variable	Description
<b>ctime</b>	Time job was created
<b>etime</b>	Time job became eligible to run
<b>qtime</b>	Time job was queued
<b>start</b>	Time job started to run

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

```
08/26/2025 17:07:44;Q;11923.napali;queue=batch
08/26/2025 17:07:50;S;11923.napali;user=dbeer group=company jobname=STDIN queue=batch
ctime=1409094464 qtime=1409094464 etime=1409094464 start=1409094470 owner=dbeer@napali
exec_host=napali/0+napali/1+napali/2+napali/3+napali/4+napali/5+torque-devtest-
03/0+torque-devtest-03/1+torque-devtest-03/2+torque-devtest-03/3+torque-devtest-
03/4+torque-devtest-03/5 Resource_List.nodes=2:ppn=6 Resource_List.nodect=2
Resource_List.nodes=2:ppn=6
08/26/2025 17:08:04;E;11923.napali;user=dbeer group=company jobname=STDIN queue=batch
ctime=1409094464 qtime=1409094464 etime=1409094464 start=1409094470 owner=dbeer@napali
```

```
exec_host=napali/0+napali/1+napali/2+napali/3+napali/4+napali/5+torque-devtest-
03/0+torque-devtest-03/1+torque-devtest-03/2+torque-devtest-03/3+torque-devtest-
03/4+torque-devtest-03/5 Resource_List.needsnodes=2:ppn=6 Resource_List.nodect=2
Resource_List.nodes=2:ppn=6 session=11352 total_execution_slots=12 unique_node_count=2
end=1409094484 Exit_status=265 resources_used.cput=00:00:00 resources_used.mem=82700kb
resources_used.vmem=208960kb resources_used.walltime=00:00:14 Error_Path=/dev/pts/11
Output_Path=/dev/pts/11
```

**i** The value of `Resource_List.*` is the amount of resources requested, and the value of `resources_used.*` is the amount of resources actually used.

**i** `total_execution_slots` and `unique_node_count` display additional information regarding the job resource usage.

## 10.4 Fields

### COMMON FIELDS

The COMMON\_FIELDS are fields that are used by both Start and Exit accounting logs.

Field	Description
<b>user</b>	USER
<b>group</b>	GROUP
<b>account</b>	ACCOUNT (optional)
<b>jobname</b>	JOBNAME
<b>queue</b>	QUEUE (optional)
<b>ctime</b>	CTIME
<b>qtime</b>	QTIME
<b>etime</b>	ETIME
<b>start_count</b>	NUMBER_OF_TIMES_STARTED
<b>start</b>	EXECUTION_START_TIME



Field	Description
<b>owner</b>	JOB_OWNER
<b>exec_host</b>	EXEC_HOST (optional)
<b>login_node</b>	LOGIN_NODE_ID (optional)
<b>RESOURCE_LIST</b>	JOB_ATR_resource (optional)
<b>Resource_Request_2.0</b>	(optional)
<b>x</b>	X_ATTRIBUTES (optional)

## Extra Fields

### Abort (A)

No extra fields.

### Checkpoint (C)

No extra fields (can say either 'Checkpointed and held' or just 'Checkpointed').

### Delete (D)

Field	Description
<b>Delete</b>	USER@HOST

### Exit (E)

Field	Description
<b>session</b>	SESSION_ID
<b>alt_id</b>	ALT_ID (optional)
<b>total_execution_slots</b>	TOTAL_EXECUTION_SLOTS (optional)
<b>unique_node_count</b>	UNIQUE_NODE_COUNT (optional)
<b>walltime</b>	WALLTIME (optional)

Field	Description
<b>end</b>	TIME_OF_COMPLETION
<b>Exit_status</b>	EXIT_STATUS

### Queue (Q)

Field	Description
<b>Queue</b>	QUEUE_NAME

### Rerun (R)

No extra fields.

### Start (S)

No extra fields.

### Restart (T)

No extra fields.

## Chapter 11: Job Logging

Torque has 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.

In this chapter:

[11.1 Job Log Location and Name](#)

[11.2 Enabling Job Logs](#)

### 11.1 Job Log Location and Name

When job logging is enabled (see [11.2 Enabling Job Logs](#)), 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, 2025 the log file is named 20251026.

A new log file is created each new day that data is written to the log.

### 11.2 Enabling Job Logs

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

Parameter	Description
<b>record_job_info</b>	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. Changing this parameter requires a restart of <code>pbs_server</code> to take effect.
<b>record_job_script</b>	If set to true, this adds the contents of the script executed by a job to the log.
<b>job_log_</b>	This specifies a soft limit (in kilobytes) for the job log's maximum size. The file

Parameter	Description
<b>file_max_size</b>	size is checked every five minutes and if the <i>current day</i> file size is greater than or equal to this value, it is rolled from <i>&lt;filename&gt;</i> to <i>&lt;filename.1&gt;</i> and a new empty log is opened. If the current day file size exceeds the maximum size a second time, the <i>&lt;filename.1&gt;</i> log file is rolled to <i>&lt;filename.2&gt;</i> , the current log is rolled to <i>&lt;filename.1&gt;</i> , 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).
<b>job_log_file_roll_depth</b>	This sets the maximum number of new log files that are kept in a day if the <a href="#">job_log_file_max_size</a> parameter is set. For example, if the roll depth is set to 3, no file can roll higher than <i>&lt;filename.3&gt;</i> . If a file is already at the specified depth, such as <i>&lt;filename.3&gt;</i> , the file is deleted so it can be replaced by the incoming file roll, <i>&lt;filename.2&gt;</i> .
<b>job_log_keep_days</b>	This maintains logs for the number of days designated. If set to 4, any log file older than 4 days old is deleted.

## Chapter 12: NUMA and Torque

Torque supports two types of Non-Uniform Memory Architecture (NUMA) systems. This chapter serves as a central information repository for the various configuration settings involved when using either NUMA system configuration.



Torque cannot be configured for both NUMA types simultaneously.

In this chapter:

- [12.1 Supported NUMA Systems](#)
- [12.2 NUMA-Aware Systems](#)
- [12.3 NUMA Tutorials](#)
- [12.4 -L NUMA Resource Request](#)
- [12.5 pbsnodes with NUMA-Awareness](#)
- [12.6 NUMA-Support Systems](#)

### 12.1 Supported NUMA Systems

Torque supports these two NUMA system configurations:

- NUMA-Aware – This configuration supports multi-req jobs and jobs that span hosts. Moab version 9.0 or later is required.
- NUMA-Support – This configuration supports only a single instance for pbs\_mom that is treated as if there were multiple nodes running in the cluster. This configuration is only for large-scale SLES systems using SGI Altix and UV hardware.

### 12.2 NUMA-Aware Systems

This section serves as a central information repository for NUMA-aware systems, and provides basic information and contains links to the various NUMA-aware topics found throughout the documentation.



Support for NUMA-aware systems is available only with Moab Workload Manager 9.0 or later.

In this section:

[12.2.1 About NUMA-Aware Systems](#)

[12.2.2 Installation and Configuration](#)

[12.2.3 Job Resource Requests](#)

[12.2.4 Job Monitoring](#)

[12.2.5 Moab/Torque NUMA Configuration](#)

[12.2.6 Considerations when Upgrading Versions or Changing Hardware](#)

## 12.2.1 About NUMA-Aware Systems

The NUMA-aware architecture is a hardware design that separates its cores into multiple clusters where each cluster has its own local memory region and still allows cores from one cluster to access all memory in the system. However, if a processor needs to use memory that is not its own memory region, it will take longer to access that (remote) memory. For applications where performance is crucial, preventing the need to access memory from other clusters is critical.

Torque uses cgroups to better manage cpu and memory accounting, memory enforcement, cpuset management, and binding jobs to devices such as MICs and GPUs. Torque will try to place jobs that request GPUs or MICs on NUMA nodes next to the GPU or MIC device to be used.

PCIe devices are similar to cores in that these devices will be closer to the memory of one NUMA node than another. Examples of PCIe devices are GPUs, NICs, disks, etc.

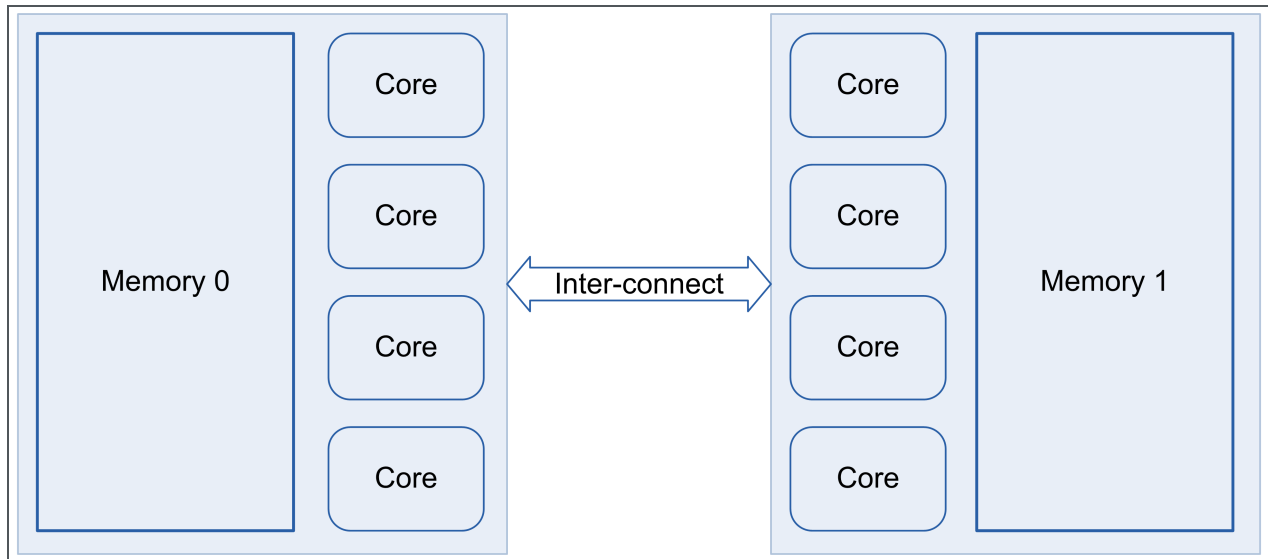
The resources of a processor chip have a hierarchy. The largest unit is a socket. A socket can contain one or more NUMA nodes with its cores and memory. A NUMA node will contain a set of cores and threads and memory that is local to the NUMA node. A core can have 0 or more threads.

- A socket refers to the physical location where a processor package plugs into a motherboard. The processor that plugs into the motherboard is also known as a socket. The socket can contain one or more NUMA nodes.
- A core is an individual execution unit within a processor that can independently execute a software execution thread and maintains its execution state separate from the execution state of any other cores within a processor.
- A thread refers to a hardware-based thread execution capability. For example, the Intel Xeon 7560 has eight cores, each of which has hardware that can effectively execute two software execution threads simultaneously, yielding 16 threads.

The following image is a simple depiction of a NUMA-aware architecture. In this example, the system has two NUMA nodes with four cores per NUMA node. The cores in each NUMA

node have access to their own memory region but they can also access the memory region of the other NUMA node through the inter-connect.

**i** If the cores from NUMA chip 0 need to get memory from NUMA chip 1 there will be a greater latency to fetch the memory.



## 12.2.2 Installation and Configuration

Once Torque is first installed, you need to perform configuration steps. See [2.7.1 Torque NUMA-Aware Configuration](#).

## 12.2.3 Job Resource Requests

NUMA-aware resources can be requested at the time of job submission using the `qsub/msub -L` parameter. In addition, the `req_information_max` and `req_information_min` queue attributes let you specify the maximum and minimum resource limits allowed for jobs submitted to a queue.

**i** Jobs requesting resources with `-L` can be run via `qrun` without a hostlist.

See also:

- [3.1.4 Requesting NUMA-Aware Resources](#)
- [Appendix N: Queue Attributes](#)

## 12.2.4 Job Monitoring

When using NUMA-aware, job resources are tracked per task. `qstat -f` produces a new category of information that begins with the 'req\_information' keyword. Following each 'req\_information' keyword is another keyword giving information about how the job was allocated. When the job has completed, the output will also include the per task resident memory used and per task cpu time used. See the section [Monitoring NUMA Job Task Placement](#)

## 12.2.5 Moab/Torque NUMA Configuration

Moab does not require special configuration to support this NUMA-aware system. However, there are a few Moab-specific things that would be helpful to know and understand. See 'Using NUMA with Moab' in the *Moab Workload Manager Administrator Guide*.

## 12.2.6 Considerations when Upgrading Versions or Changing Hardware

After upgrading server software or updating the hardware for a compute node, you should start the `pbs_mom` daemon with the `-f` flag to force the server to recognize the new configuration. See documentation for the [A.4 pbs\\_mom -f](#) flag.

# 12.3 NUMA Tutorials

This section contains links to tutorials and other documentation useful in understanding NUMA-Aware systems.

In this section:

[12.3.1 NUMA Primer](#)

[12.3.2 How NUMA Places Jobs](#)

[12.3.3 NUMA Discovery and Persistence](#)

## 12.3.1 NUMA Primer

Resource request syntax gives users the ability to request resources on a per task basis, have multiple asymmetric resource requests in the same job, and control where jobs



execute on cores and memory within NUMA hardware.

Control groups (cgroups) provide the ability to partition sets of tasks and their children into hierarchical groups with specialized behavior; cgroups are used to manage each job. In RHEL 7, the default directory for cgroups became `/sys/fs/cgroup`. The following examples use this standard.

**i** If you are building with cgroups enabled, you must have boost version 1.41 or later.

In this topic:

[12.3.1.A Torque cgroup Hierarchy](#)

[12.3.1.B cpuset Subsystem](#)

[12.3.1.C cpuacct Subsystem](#)

[12.3.1.D memory Subsystem](#)

[12.3.1.E Resource Request 2.0](#)

### 12.3.1.A Torque cgroup Hierarchy

Torque only uses the `cpu`, `devices`, `cpuacct`, `cpuset`, and `memory` subsystems. When `pbs_mom` is initialized it creates a sub-directory named `torque` in each of the five subsystem directories. When a job is started on the MOM a directory that is the full job ID is created under each `torque` directory. The following is an `'ls -al'` submission command example from the `cpuset/torque` hierarchy:

```
total 0
drwxr-xr-x 3 root root 0 Aug 28 13:36 .
drwxr-xr-x 4 root root 0 Aug 28 13:35 ..
drwx----- 4 root root 0 Aug 31 10:20 1301.hosta
-rw-r--r-- 1 root root 0 Aug 28 13:35 cgroup.clone_children
--w--w--w- 1 root root 0 Aug 28 13:35 cgroup.event_control
-rw-r--r-- 1 root root 0 Aug 28 13:35 cgroup.procs
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.cpu_exclusive
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.cpus
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.mem_exclusive
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.mem_hardwall
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.memory_migrate
-r--r--r-- 1 root root 0 Aug 28 13:35 cpuset.memory_pressure
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.memory_spread_page
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.memory_spread_slab
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.mems
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.sched_load_balance
-rw-r--r-- 1 root root 0 Aug 28 13:35 cpuset.sched_relax_domain_level
-rw-r--r-- 1 root root 0 Aug 28 13:35 notify_on_release
-rw-r--r-- 1 root root 0 Aug 28 13:35 tasks
```

Line 4 shows that the subdirectory is '1301.hosta'. This is the cpuset cgroup for job '1301.hosta'. If you were to issue an `ls` command on the '1301.hosta' subdirectory in this example, you would see the following:

```
total 0
drwx----- 4 root root 0 Aug 31 10:24 .
drwxr-xr-x 3 root root 0 Aug 31 10:22 ..
-rw-r--r-- 1 root root 0 Aug 31 10:24 cgroup.clone_children
--w--w--w- 1 root root 0 Aug 31 10:24 cgroup.event_control
-rw-r--r-- 1 root root 0 Aug 31 10:24 cgroup.procs
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.cpu_exclusive
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.cpus
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.mem_exclusive
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.mem_hardwall
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.memory_migrate
-r--r--r-- 1 root root 0 Aug 31 10:24 cpuset.memory_pressure
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.memory_spread_page
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.memory_spread_slab
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.mems
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.sched_load_balance
-rw-r--r-- 1 root root 0 Aug 31 10:24 cpuset.sched_relax_domain_level
-rw-r--r-- 1 root root 0 Aug 31 10:24 notify_on_release
drwx----- 2 root root 0 Aug 31 10:24 R0.t0
drwx----- 2 root root 0 Aug 31 10:24 R0.t1
-rw-r--r-- 1 root root 0 Aug 31 10:24 tasks
-rw-r--r-- 1 root root 0 Aug 28 13:35 tasks
```

For this job the `-L` resource request was:

```
qsub -L tasks=2:lprocs=2
```

This job has a single request and two tasks. The `R0` represents request 0 and the `t0` and `t1` represent the two tasks. In this case, cpuset information would be set for each task in their respective subdirectories. The `cpu`, `cpuacct`, `memory` and `devices` subsystems also utilize the same subdirectory structure.

### 12.3.1.B cpuset Subsystem

The Linux cpuset functionality was integrated into cgroups so that when Torque is configured with the `--enable-cgroups` option, cpuset functionality is also included. When jobs are submitted using the `-L` resource request syntax, Torque allocates a cpu set and memory set for each task in the job request. Examples of how cpusets and memory sets are allocated will be shown in the examples at the end of this primer.

### 12.3.1.C cpuacct Subsystem

The cpuacct subsystem keeps track of cpu time used for a cgroup. Torque now uses the cpuacct data to calculate cpu time used for a job. Also when using the `-L` resource request, cpu time per task is also recorded. Another advantage of cgroups is that the accounting information of a job does not disappear when the job process exits. So if `pbs_mom` goes

down for any reason while running jobs the cpu time and memory used can still be tracked when pbs\_mom is restarted.

### 12.3.1.D memory Subsystem

The memory subsystem keeps track of the maximum memory used by a cgroup and also can be used to limit the maximum amount of resident memory a task can use or the maximum amount of swap a task can use. The -L resource request syntax has a memory and a swap option.

Following are examples of how to request memory restrictions with the -L resource request:

```
qsub -L tasks=2:memory=300mb
```

Two tasks are created. The memory=300mb option restricts each task to a maximum of 300 megabytes of resident memory. If a task exceeds 300 MB, then the excess memory is sent to swap.

```
qsub -L tasks=2:swap=1Gb
```

Two tasks are created. The swap limit for each task is set to 1 GB.

**i** In order to be able to set swap and memory limits the Linux kernel must be built using the options CONFIG\_MEMCG=y, CONFIG\_MEMCG\_SWAP=y and CONFIG\_MEMCG\_SWAP\_ENABLED=y. For Red Hat 7-based systems, these options are set by default.

For SUSE 12-based systems, you will also need to modify the /etc/default/grub file:

1. Edit /etc/default.grub.
2. Add the following inside of the GRUB\_CMDLINE\_LINUX\_DEFAULT variable:

```
cgroup_enable=memory swapaccount=1
```

3. Run the following:

```
root# update-bootloader --refresh
```

4. Reboot your machine.

### 12.3.1.E Resource Request 2.0

Following are several different types of -L resource requests. The examples show how to use the syntax to be able to have resources allocated that can best fit your job needs.

## Single Resource Request with Two Tasks and Default Settings

```
qsub -L tasks=2:lprocs=1
```

After this job runs, the summarized qstat -f output is shown:

```
Job Id: 1306.hosta
Job_Name = bigmem.sh
Job_Owner = knielson@hosta
resources_used.cput = 00:00:01
resources_used.energy_used = 0
resources_used.mem = 1956984kb
resources_used.vmem = 2672652kb
resources_used.walltime = 00:00:10
job_state = C
. . . .
exit_status = 0
submit_args = -L tasks=2:lprocs=1 ../scripts/bigmem.sh
. . . .
req_information.task_count.0 = 2
req_information.lprocs.0 = 1
req_information.thread_usage_policy.0 = allowthreads
req_information.hostlist.0 = hosta:ppn=2
req_information.task_usage.0.task.0.cpu_list = 0
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.memory_used = 976816kb
req_information.task_usage.0.task.0.cores = 0
req_information.task_usage.0.task.0.threads = 1
req_information.task_usage.0.task.0.host = hosta
req_information.task_usage.0.task.1.cpu_list = 4
req_information.task_usage.0.task.1.mem_list = 0
req_information.task_usage.0.task.1.memory_used = 976752kb
req_information.task_usage.0.task.1.cores = 0
req_information.task_usage.0.task.1.threads = 1
req_information.task_usage.0.task.1.host = hosta
```

In this job, 1 second of cpu time was used. 1956984kb of resident memory was used, but with the new -L syntax there is a new set of information that starts with req\_information. This is the per task information of the job.

Output	Description
req_information.task_count.0 = 2	Two tasks are requested for this resource request; named tasks 0 and 1 respectively.
req_information.lprocs.0 = 1	One logical processor is requested per task. The lprocs value becomes the number of processing units per task allocated in the cpuset.
req_information.thread_usage_policy.0 = allowthreads	The processing unit allocation policy for the task. allowthreads is the user-specified default policy. allowthreads uses the first available core or thread. Processing units allocated in the cpuset are adjacent to each other unless other processors are also allocated.

Output	Description
<code>req_information.hostlist.0 = hosta:ppn=2</code>	On hostname <i>hosta</i> , two processing units are necessary. A single resource request can run on more than one host.
<code>req_information.task_usage.0.task.0.cpu_list = 0</code>	The <code>task_usage</code> keyword refers to the per task resource usage. 0 is the processing unit assigned to this task. In <code>req_information.task_usage.0.cpu_list.1</code> , the processing unit assigned is 4. This particular hardware is a 4 core system with hyper threading. So the core numbering is (0,4), (1,5), (2,6) and (3,7). Because the <code>thread_usage_policy</code> is <code>allowthreads</code> , the first two processing units are taken by default.
<code>req_information.task_usage.0.task0.mem_list = 0</code>	Memory location 0 is allocated to this task.
<code>req_information.task_usage.0.task0.memory_used = 976816kb</code>	The amount of resident memory used at task 0 is 976816kb.
<code>req_information.task_usage.0.task0.cores = 0</code>	This is the number of cores used by the task. In this case no cores were used because the <code>allowthreads</code> policy uses only threads and not discrete cores.
<code>req_information.task_usage.0.task0.host = hosta</code>	The task was run on hostname <i>hosta</i> .

**i** The information for `req_information.task_usage.0.task.1` as opposed to `task.0`, means that the information displayed is referring to what was performed on task 1, rather than task 0.

## Multiple lprocs

```
qsub -L tasks=1:lprocs=2
```

Two logical processors are specified with one task. The output of this job is as follows:

```
req_information.task_count.0 = 1
req_information.lprocs.0 = 2
req_information.thread_usage_policy.0 = allowthreads
req_information.hostlist.0 = hosta:ppn=2
req_information.task_usage.0.task.0.cpu_list = 0,4
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 0
```

```
req_information.task_usage.0.task.0.threads = 2
req_information.task_usage.0.task.0.host = hosta
```

The req\_information for this syntax shows a cpu\_list with two processing units. 0 and 4 are the first two processing units available so they are in the cpu\_list. Notice that now there are two threads running.

usecores

The following example shows how a request to use cores changes the cpu\_list allocation:

```
qsub -L tasks=1:lprocs=4:usecores

req_information.task_count.0 = 1
req_information.lprocs.0 = 4
req_information.thread_usage_policy.0 = usecores
req_information.hostlist.0 = hosta:ppn=4
req_information.task_usage.0.task.0.cpu_list = 0-3
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 4
req_information.task_usage.0.task.0.threads = 8
req_information.task_usage.0.task.0.host = hosta
```

Output	Description
req_information.task_usage.0.task.0.cores = 4	Four cores are used for task 0.
req_information.task_usage.0.task.0.threads = 8	When a core is requested, any threads for that core are no longer available for use by another task. In this case, each core has two threads. As a result, when one core is used two threads are also used. In this case, 8 threads are used in total.

usethreads

```
qsub -L tasks=1:lprocs=4:usethreads
```

The output of this job is as follows:

```
req_information.task_count.0 = 1
req_information.lprocs.0 = 4
req_information.thread_usage_policy.0 = usecores
req_information.hostlist.0 = hosta:ppn=4
req_information.task_usage.0.task.0.cpu_list = 0,4,1,5
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 0
req_information.task_usage.0.task.0.threads = 4
req_information.task_usage.0.task.0.host = hosta
```

Requesting usethreads gives adjacent processing units 0,4,1,5 and uses only 4 threads as indicated by req\_information.task\_usage.0.task.0.threads = 4.

Multiple Resource Requests

The -L resource requests makes it easier to request asymmetric resources for a single job. For example, you might have a job that needs several processors on a host to do work but only one or two processors on another host. The -L syntax easily accommodates this.

```
qsub -L tasks=2:lprocs=6:usecores -L tasks=1:lprocs=1:place=socket
```

```
req_information.task_count.0 = 2
req_information.lprocs.0 = 6
req_information.thread_usage_policy.0 = usecores
req_information.hostlist.0 = hostb:ppn=12
req_information.task_usage.0.task.0.cpu_list = 0-5
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 6
req_information.task_usage.0.task.0.threads = 12
req_information.task_usage.0.task.0.host = hostb
req_information.task_usage.0.task.1.cpu_list = 6-11
req_information.task_usage.0.task.1.mem_list = 1
req_information.task_usage.0.task.1.cores = 6
req_information.task_usage.0.task.1.threads = 12
req_information.task_usage.0.task.1.host = hostb
req_information.task_count.1 = 1
req_information.lprocs.1 = 1
req_information.socket.1 = 1
req_information.thread_usage_policy.1 = allowthreads
req_information.hostlist.1 = hostb:ppn=1
req_information.task_usage.1.task.0.cpu_list = 0
req_information.task_usage.1.task.0.mem_list = 0
req_information.task_usage.1.task.0.cores = 1
req_information.task_usage.1.task.0.threads = 1
req_information.task_usage.1.task.0.host = hostb
```

Output	Description
req_information.task_count.1 = 1	Only one task on request 1.

Output	Description
<code>req_information.socket.1 = 1</code>	One socket is requested and then allocated for use.

## place Directives

The `place` directive takes one of five arguments: `node`, `socket`, `numanode`, `core`, and `thread`. The `node`, `core`, and `thread` arguments do not take an assignment, however `socket` and `numanode` can be assigned a number value requesting the number of sockets or numanodes per task. The use of `'place=core'` or `'place=thread'` is the equivalent of using the `uscores` or `usethreads` syntax.

When processes share the same memory cache and are run on adjacent cores or threads, the likelihood of swapping out a cache line is high. When memory needs to be fetched from primary memory instead of the cache processing execution times are increased and become less predictable. In these examples, Torque disables linearly allocating cores. To help ensure best performance by avoiding the sharing of caches between processors, cores are spread as far apart as possible.

The following examples show the results of each directive:

```
place=socket
```

If a socket is not given a number, it defaults to the number 1:

```
qsub -L tasks=2:lprocs=2:place=socket
```

This request allocates two tasks with two logical processors each. Each task is placed on its own socket.

```
req_information.task_count.0 = 2
req_information.lprocs.0 = 2
req_information.socket.0 = 1
req_information.thread_usage_policy.0 = allowthreads
req_information.hostlist.0 = hosta:ppn=4
req_information.task_usage.0.task.0.cpu_list = 0,3
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 2
req_information.task_usage.0.task.0.threads = 4
req_information.task_usage.0.task.0.host = hosta
req_information.task_usage.0.task.1.cpu_list = 6,9
req_information.task_usage.0.task.1.mem_list = 1
req_information.task_usage.0.task.1.cores = 2
req_information.task_usage.0.task.1.threads = 4
req_information.task_usage.0.task.1.host = hosta
cpuset_string = hosta:0,3,6,9
memset_string = hosta:0-1
```



For the last example, the job was run on a dual socket host with 12 cores. Each core has two threads for a total of 24 processing units. Each socket has 6 cores and 12 threads. The cores for socket 0 are numbered 0, 1, 2, 3, 4, 5. The cores for socket 1 are numbered 6, 7, 8, 9, 10, 11. Task.0 uses cores 0 and 3 and task.1 uses cores 6 and 9.

```
place=numanode=2
```

```
qsub -L tasks=2:lprocs=2:place=numanode=2
```

This request allocates two numanodes, one for each task:

```
req_information.task_count.0 = 2
req_information.lprocs.0 = 2
req_information.numanode.0 = 2
req_information.thread_usage_policy.0 = allowthreads
req_information.hostlist.0 = hostb:ppn=2
req_information.task_usage.0.task.0.cpu_list = 0, 3
req_information.task_usage.0.task.0.mem_list = 0
req_information.task_usage.0.task.0.cores = 0
req_information.task_usage.0.task.0.threads = 0
req_information.task_usage.0.task.0.host = hostb
req_information.task_usage.0.task.1.cpu_list = 6, 9
req_information.task_usage.0.task.1.mem_list = 1
req_information.task_usage.0.task.1.cores = 2
req_information.task_usage.0.task.1.threads = 4
req_information.task_usage.0.task.1.host = hostb
```

## pbsnodes and Dedicated Resources

When a resource is requested (core, numanode, socket, etc.), the entire resource is no longer available for other jobs to use, and enters a dedicated state. pbsnodes tracks total sockets, numanodes, cores and threads per node. pbsnodes also tracks dedicated sockets, numanodes, cores, and threads.

Following is an example of node output in pbsnodes:

```
state = free
power_state = Running
np = 12
ntype = cluster
status =
rectime=1441054213,macaddr=78:e3:b5:0a:c0:58,cpuclock=Fixed,varattr=,jobs=,state=free...
mom_service_port = 15002
mom_manager_port = 15003
total_sockets = 2
total_numa_nodes = 2
total_cores = 12
total_threads = 12
dedicated_sockets = 0
dedicated_numa_nodes = 0
dedicated_cores = 0
dedicated_threads = 0
```

This node has a total of 2 sockets, 2 numanodes, 12 cores, and 12 threads. The number of dedicated sockets, numanodes, cores, and threads are set to 0 indicating there are currently no jobs running on this nodes. If a job is run with a syntax of:

```
qsub -L tasks=2:lprocs=2
```

the number of dedicated threads becomes four.

Once a job is completed, dedicated\_threads returns to 0.

### 12.3.2 How NUMA Places Jobs

This topic discusses how jobs are placed on their specific NUMA resources.

**i** In this topic, placing is defined as determining where on the node the job will go.

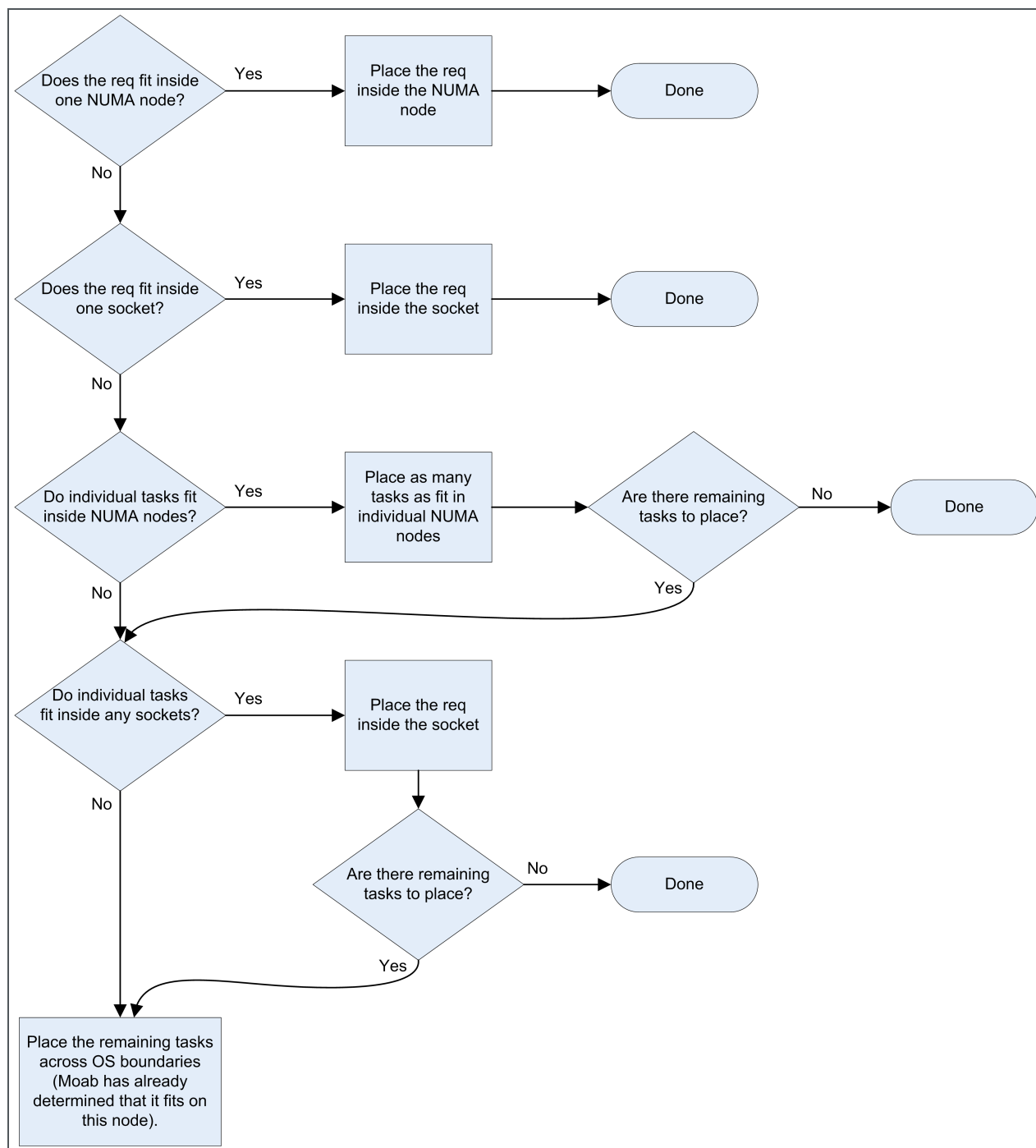
Moab determines where to place a job on a node and pbs\_server places that job inside the node. pbs\_server decides where to place the job based on the parameters specified by the job itself and optimal locality.

#### The Placement Algorithm

Whenever possible, jobs are given placement preference next to the GPUs that they will use. If the job's tasks' memory and CPUs are available on the socket that has the GPUs, that job will be placed at that location.

**i** This placement fails if either there is no socket that contains all of the required resources, or if jobs are and the GPUs are on socket - but all of the cores are used by another job.

The following diagram shows the decision making process for each request not using the 'place=' syntax:



For jobs using the 'place=' syntax, the decision making algorithm is much simpler. When the place is specified, it will become reserved and the job will be placed at that location.



For a job to occupy the user-requested place option, that option must be completely available for use.

The following is an example of a job submitted using the -L option:

```
-L tasks=1:lprocs=2:sockets=2
```

This job placed on two sockets with one core reserved per socket.

**i** The use of the word 'core' is intentional. If a 'place=socket' or 'place=numanode' is requested and the lprocs request is less than the number of cores inside the socket or NUMA node, then the job is given only cores.

Once pbs\_server has determined where each task should run, that decision is stored in attributes on the job itself. The complete\_req attribute stores where each task is allocated, and the mom reads that information to create appropriate cgroups for the job and for the entire task. This information is available to the user via qstat.

### 12.3.3 NUMA Discovery and Persistence

In this topic:

[12.3.3.A Initial Discovery](#)

[12.3.3.B Job Placement Decisions](#)

[12.3.3.C Persistence Across Restarts](#)

#### 12.3.3.A Initial Discovery

First, The mom performs the initial discovery of the host's layout, including the number of sockets, numa nodes, pci devices, cores, and threads. This is done using the hwloc library. Next, the mom sends that information to pbs\_server, which notes it. Last, pbs\_server writes files with the node's layout information locally. Following restarts, node information is gathered from these files.

#### 12.3.3.B Job Placement Decisions

Job placement decisions are done by pbs\_server so that it immediately knows which NUMA resources have been used by which job. As a result, the second a job starts or finishes the information for available numa resources is updated and accurate. This information is then communicated to the mom daemon. For more information on how jobs are placed, see [12.3.2 How NUMA Places Jobs](#).

### 12.3.3.C Persistence Across Restarts

To maintain correct usage information `pbs_server` writes files to a new directory in `/server_priv/node_usage`. The files are written in JSON format. The following is a representation of what these files look like:

```
# Simple Node
"node" :
{
  "socket" :
  {
    "os_index" : 0,
    "numanode" :
    {
      "os_index" : 0,
      "cores" : "0-5",
      "threads" : "",
      "mem" : 16435852
    }
  }
}

# More Complicated Node
"node" :
{
  "socket" :
  {
    "os_index" : 0,
    "numanode" :
    {
      "os_index" : 0,
      "cores" : "0-7",
      "threads" : "",
      "mem" : 16775316
    },
    "numanode" :
    {
      "os_index" : 1,
      "cores" : "8-15",
      "threads" : "",
      "mem" : 16777216
    }
  },
  "socket" :
  {
    "os_index" : 1,
    "numanode" :
    {
      "os_index" : 2,
      "cores" : "16-23",
      "threads" : "",
      "mem" : 8388608
    },
    "numanode" :
    {
      "os_index" : 3,
      "cores" : "24-31",
      "threads" : "",
      "mem" : 8388608
    }
  }
}
```

```
}
}
```

When jobs are present, an allocation object will also be there to record what resources are being used by the job. The allocation object will be beneath the numanode object, so it is possible to have more than one per job. An example of an allocation object is shown below:

```
# Allocation object

"allocation" :
{
  "cores_only" : 0,
  "cpus" : "0",
  "exclusive" : 6,
  "jobid" : "39.roshar",
  "mem" : "0"
}
```

An example of a complete node usage file is shown below:

```
# Node usage

{
  "node" :
  [
    {
      "socket" :
      {
        "numanodes" :
        [
          {
            "numanode" :
            {
              "allocations" :
              [
                {
                  "allocation" :
                  {
                    "cores_only" : 0,
                    "cpus" : "0",
                    "exclusive" : 6,
                    "jobid" : "39.roshar",
                    "mem" : "0",
                  }
                }
              ],
              "cores" : "0-7",
              "mem" : "16325348",
              "os_index" : 0,
              "threads" : ""
            }
          ]
        },
        "os_index" : 0
      }
    ]
  }
}
```

## 12.4 -L NUMA Resource Request

This section provides the -L option syntax and a description of the allocation options. The -L option is available in the qsub and msub commands to allow admins the ability to place jobs at the 'task' or 'OS process' level to get maximum efficiency out of the available hardware. Using the -L option requires a basic knowledge of the topologies of the available hardware where jobs will run. You will need to know how many cores, numanodes, sockets, etc., are available on the hosts within the cluster. The -L syntax is designed to allow for a wide variety of requests. However, if requests do not match the available hardware, you may have unexpected results.

In addition, multiple, non-symmetric resource requests can be made for the same job using the -L job submission syntax. For example, the following command creates two requests:

```
qsub -L tasks=4:lprocs=2:usecores:memory=500mb -L tasks=8:lprocs=4:memory=2gb
```

The first request creates 4 tasks with two logical processors and 500 MB of memory per task. The logical processors are placed on cores. The second request calls for 8 tasks with 4 logical processors and 2 GB of memory per task. Logical processors can be placed on cores or threads since the default placement is allowthreads.

In this section:

[12.4.1 Syntax](#)

[12.4.2 Allocation Options](#)




### 12.4.1 Syntax

```
-L tasks=#[:lprocs=#|all]
[:{usecores|usethreads|allowthreads}]
[:place={socket|numanode|core|thread}[=#]{node}][:memory=#][:swap=#][:maxtpn=#]
[:gpus=#[:<mode>]][:mics=#][:gres=<gres>][:feature=<feature>]
[[[:{cpt|cgroup_per_task}]][:{cph|cgroup_per_host}]]
```

### 12.4.2 Allocation Options

The following table identifies the various allocation options you can specify per task.



**i** **tasks=#** specifies the number of job tasks for which the resource request is to be applied. It is the only required element for the -L resource request. The remainder of the -L syntax allocates resources per task.


Value	Description
cpt, cgroup_ per_task, cph, cgroup_ per_host	<p>Specifies whether cgroups are created per-task or per-host. If submitting using <code>msub</code>, this information is passed through to Torque; there is no affect to Moab operations.</p> <div data-bbox="406 420 1421 604">  This option lets you specify how cgroups are created during job submission. This option can be used to override the Torque <code>cgroup_per_task</code> server parameter. If this option is not specified, the server parameter value is used. See the server parameter <a href="#">cgroup_per_task</a> for more information.         </div> <ul style="list-style-type: none"> <li>• <b>:cpt, :cgroup_per_task</b> – Job request will have one cgroup created per task; all the processes on that host will be placed in the <i>first</i> task's cgroup.</li> <li>• <b>:cph, :cgroup_per_host</b> – Job request will have one cgroup created per host; this is similar to pre-6.0 cpuset implementations.</li> </ul> <div data-bbox="406 808 1421 1134">  Some MPI implementations only launch one process through the TM API, and then fork each subsequent process that should be launched on that host. If the job is set to have one cgroup per task, this means that all of the processes on that host will be placed in the <i>first</i> task's cgroup. Confirm that the <code>cgroup_per_task</code> Torque server parameter is set to <code>FALSE</code> (default) or specify <code>:cph</code> or <code>:cgroup_per_host</code> at job submission.  If you know that your MPI will communicate each process launch to the mom individually, then set the <code>cgroup_per_task</code> Torque server parameter is set to <code>TRUE</code> or specify <code>:cpt</code> or <code>:cgroup_per_task</code> at job submission.         </div>
feature	<p>Specifies one or more node feature names used to qualify compute nodes for task resources (i.e., a compute node must have all (&amp;) and/or ( ) of the specified feature names assigned or the compute node's resources are ineligible for allocation to a job task).</p> <div data-bbox="406 1323 1421 1417"> <pre>:feature=bigmem :feature='bmem&amp;fio' :feature='bmem fio'</pre> </div>
gpus	<p>Specifies the quantity of GPU accelerators to allocate to a task, which requires placement at the locality-level to which an accelerator is connected or higher. &lt;MODE&gt; can be <code>exclusive_process</code>, <code>exclusive_thread</code>, or <code>reseterr</code>.</p> <div data-bbox="406 1564 1421 1717">  If you are using CUDA 8 or newer, the default of <code>exclusive_thread</code> is no longer supported. If the server specifies an <code>exclusive_thread</code> setting, the MOM will substitute an <code>exclusive_process</code> mode setting. We recommend that you set the default to <code>exclusive_process</code>.         </div> <p>The task resource request must specify placement at the numanode- (AMD only), socket-, or node-level. <code>place=core</code> and <code>place=thread</code> are invalid placement options when a task requests a PCIe-based accelerator device, since allowing</p>





Value	Description
	<p>other tasks to use cores and threads on the same NUMA chip or socket as the task with the PCIe device(s) would violate the consistent job execution time principle since these other tasks would likely interfere with the data transfers between the task's logical processors and its allocated accelerator(s).</p> <pre>:gpus=1</pre> <p><i>Allocates one GPU per task.</i></p> <pre>:gpus=2:exclusive_process:reseterr</pre> <p><i>Allocates two GPUs per task with exclusive access by process and resets error counters.</i></p>
gres	<p>Specifies the quantity of a specific generic resource <code>&lt;gres&gt;</code> to allocate to a task. If a quantity is not given, it defaults to one.</p> <p><b>i</b> Specify multiple GRES by separating them with commas and enclosing all the GRES names, their quantities, and the commas within single quotation marks.</p> <pre>:gres=matlab=1</pre> <p><i>Allocates one Matlab license per task.</i></p> <pre>:gres='dvd,blu=2'</pre> <p><i>Allocates one DVD drive and two Blu-ray drives per task, represented by the "dvd" and "blu" generic resource names, respectively.</i></p> <p><b>i</b> When scheduling, if a generic resource is node-locked, only compute nodes with the generic resource are eligible for allocation to a job task. If a generic resource is floating, it does not qualify or disqualify compute node resources from allocation to a job task.</p>
lprocs	<p>Specifies the quantity of 'logical processors' required by a single task to which it will be pinned by its control-group (cgroup).</p> <p><b>i</b> The 'place' value specifies the total number of physical cores/threads to which a single task has exclusive access. The <code>lprocs=</code> keyword indicates the <i>actual</i> number of cores/threads to which the task has exclusive access for the task's cgroup to pin to the task.</p> <ul style="list-style-type: none"> <li>When <code>:lprocs</code> is specified, and nothing is specified for <code>#</code>, the default is 1.</li> <li>When <code>:lprocs=all</code> is specified, all cores or threads in any compute node/server's available resource locality placement specified by the 'place' option is eligible for task placement (the user has not specified a quantity,</li> </ul>

Value	Description
	<p>other than "give me all logical processors within the resource locality or localities"), which allows a user application to take whatever it can get and adapt to whatever it receives, which cannot exceed one node.</p> <pre data-bbox="414 401 703 426">qsub -L tasks=1:lprocs=4</pre> <p><i>One task is created, which allocates four logical processors to the task. When the job is executed, the pbs_mom where the job is running will create a cpuset with four processors in the set. Torque will make a best effort to allocate the four processors next to each other but the placement is not guaranteed.</i></p> <pre data-bbox="414 648 857 674">qsub -L tasks=1:lprocs=all:place=node</pre> <p><i>Places one task on a single node, and places all processing units in the cpuset of the task. The "lprocs=all" parameter specifies that the task will use all cores and/or threads available on the resource level requested.</i></p>
maxtpn	<p>Specifies the maximum tasks per node; where '#' is the maximum tasks allocated per physical compute node. This restricts a task type to no more than '#' tasks per compute node and allows it to share a node with other task types or jobs. For example, a communication-intensive task may share a compute node with computation-intensive tasks.</p> <p><b>i</b> The number of nodes and tasks per node will not be known until the job is run.</p> <pre data-bbox="414 1167 703 1192">qsub -L tasks=7:maxtpn=4</pre> <p><i>Allocates seven tasks but a maximum of four tasks can run on a single node.</i></p>
memory	<p><b>i</b> 'memory' is roughly equivalent to the mem request for the qsub/msub -l resource request. However, with the -L qsub syntax, cgroups monitors the job memory usage and puts a ceiling on resident memory for each task of the job.</p> <p>Specifies the maximum resident memory allocated per task. Allowable suffixes are kb (kilobytes), mb (megabytes), gb (gigabytes), tb (terabyte), pb (petabytes), and eb (exabyte). If a suffix is not provided by the user, kb (kilobytes) is default. Either whole or decimal numbers are allowed.</p> <p><b>i</b> If a task uses more resident memory than specified the excess memory is moved to swap.</p> <pre data-bbox="414 1766 954 1791">qsub -L tasks=4:lprocs=2:usecores:memory=.5gb</pre>

Value	Description
	<p><i>Allocates four tasks with two logical processors each. Each task is given a limit of .5 GB of resident memory.</i></p> <pre>qsub -L tasks=2:memory=3500</pre> <p><i>Allocates two tasks with 3500 kb (the suffix was not specified so kilobytes is assumed).</i></p>
mics	<p>Specifies the quantity of Intel MIC accelerators to allocate to a task, which requires placement at the locality-level to which a MIC is connected or higher. The task resource request must specify placement at the NUMA chip- (makes sense for AMD only), socket-, or node-level. place=core and place=thread are invalid placement options when a task requests a PCIe-based accelerator device since allowing other tasks to use cores and threads on the same NUMA chip or socket as the task with the PCIe device(s) would violate the consistent job execution time principle since these other tasks would likely interfere with the data transfers between the task's logical processors and its allocated accelerator (s).</p> <p> Allocating resources for MICs operates in the exact same manner as for GPUs. See <a href="#">gpus</a>.</p> <pre>:mics=1</pre> <p><i>Allocates one MIC per task.</i></p> <pre>:mics=2</pre> <p><i>Allocates two MICs per task.</i></p>
place	<p>Specifies placement of a single task on the hardware. Specifically, this designates what hardware resource locality level and identifies the quantity of locality-level resources. Placement at a specific locality level is always exclusive, meaning a job task has exclusive use of all logical processor and physical memory resources at the specified level of resource locality, even if it does not use them.</p> <p><b>Options:</b></p> <p> If an option is not specified, the usecores, usethreads, and allowthreads parameters are used.</p> <ul style="list-style-type: none"> <li><b>socket[#]</b> – Refers to a socket within a compute node/server and specifies that each task is placed at the socket level with exclusive use of all logical processors and memory resources of the socket(s) allocated to a task. If a count is not specified, the default setting is 1.</li> </ul> <pre>qsub -L tasks=2:lprocs=4:place=socket</pre>

Value	Description
	<div data-bbox="516 300 1369 506" style="border: 1px solid black; padding: 10px; margin-bottom: 10px;"> <p><i>Two tasks are allocated with four logical processors each. Each task is placed on a socket where it will have exclusive access to all of the cores and memory of the socket. Although the socket can have more cores/threads than four, only four cores/threads will be bound in a cpuset per task per socket as indicated by "lprocs=4".</i></p> </div> <ul style="list-style-type: none"> <li> <b>numanode[=#]</b> – Refers to the numanode within a socket and specifies that each task is placed at the NUMA node level within a socket with exclusive use of all logical processor and memory resources of the NUMA node(s) allocated to the task. If a count is not given, the default value is 1. If a socket does not contain multiple numanodes, by default the socket contains one numanode.            To illustrate the locality level to which this option refers, the following examples are provided:            First, a Haswell-based Intel Xeon v3 processor with 10 or more cores is divided internally into two separate 'nodes', each with an equal quantity of cores and its own local memory (referred to as a 'numanode' in this topic).            Second, an AMD Opteron 6xxx processor is a 'multi-chip module' that contains two separate physical silicon chips each with its own local memory (referred to as a 'numanode' in this topic).            In both of the previous examples, a core in one 'node' of the processor can access its own local memory faster than it can access the remote memory of the other 'node' in the processor, which results in NUMA behavior.           <div data-bbox="462 1113 1421 1159" style="border: 1px dashed black; padding: 5px; margin: 10px 0;"> <pre>qsub -L tasks=2:lprocs=4:place=numanode</pre> </div> <div data-bbox="516 1169 1369 1278" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p><i>Places a single task on a single numanode and the task has exclusive use of all the logical processors and memory of the numanode.</i></p> </div> <div data-bbox="462 1291 1421 1337" style="border: 1px dashed black; padding: 5px; margin: 10px 0;"> <pre>qsub -L tasks=2:lprocs=4:place=numanode=2</pre> </div> <div data-bbox="516 1348 1369 1394" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p><i>Allocates two tasks with each task getting two numanodes each.</i></p> </div> </li> <li> <b>core[=#]</b> – Refers to a core within a numanode or socket and specifies each task is placed at the core level within the numanode or socket and has exclusive use of all logical processor and memory resources of the core(s) allocated to the task. Whether a core has SMT/hyper-threading enabled or not is irrelevant to this locality level. If a number of cores is not specified, it will default to the number of lprocs specified.           <div data-bbox="462 1608 1421 1738" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p> The number of cores specified must be greater than or equal to the number of lprocs available; otherwise, the job submission will be rejected.</p> </div> <div data-bbox="462 1749 1421 1795" style="border: 1px dashed black; padding: 5px; margin: 10px 0;"> <pre>qsub -L tasks=2:place=core=2</pre> </div> </li> </ul>

Value	Description
	<p><i>Two tasks with one logical processor each will be placed on two cores per task.</i></p> <pre>qsub -L tasks=2:lprocs=2:place=core</pre> <p><i>Two tasks are allocated with two logical processors per task. Each logical process will be assigned to one core each (two cores total, the same as the number of lprocs). Torque will attempt to place the logical processors on non-adjacent cores.</i></p> <ul style="list-style-type: none"> <li>• <b>thread[#]</b> – Specifies each task is placed at the thread level within a core and has exclusive use of all logical processor and memory resources of the thread(s) allocated to a task. This affinity level refers to threads within a core and is applicable only to nodes with SMT or hyper-threading enabled. If a node does not have SMT or hyper-threading enabled, Moab will consider the node ineligible when allocating resources for a task. If a specific number of threads is not specified, it will default to the number of lprocs specified.</li> </ul> <pre>qsub -L tasks=2:lprocs=4:place=thread</pre> <p><i>Allocates two tasks, each with four logical processors, which can be bound to any thread. Torque will make a best effort to bind the threads on the same nomanode but placement is not guaranteed. Because the number of threads is not specified, Torque will place the number of lprocs requested.</i></p> <ul style="list-style-type: none"> <li>• <b>node</b> – Specifies that each task is placed at the node level and has exclusive use of all the resources of the node(s) allocated to a task. This locality level usually refers to a physical compute node, blade, or server within a cluster.</li> </ul> <pre>qsub -L tasks=2:lprocs=all:place=node</pre> <p><i>Two tasks are allocated with one task per node, where the task has exclusive access to all the resources on the node. The "lprocs=all" specification directs Torque to create a cpuset with all of the processing units on the node. The "place=node" specification also claims all of the memory for the node/server.</i></p>
swap	<p>Specifies the maximum allocated resident memory <i>and</i> swap space allowed per task.</p> <p>Allowable suffixes are kb (kilobytes), mb (megabytes), gb (gigabytes), tb (terabyte), pb (petabytes), and eb (exabyte). If a suffix is not given, kb (kilobytes) is assumed. Either whole or decimal numbers are allowed.</p> <p>If a task exceeds the specified limit, the task will be killed; the associated job will be terminated.</p>

Value	Description
	<div>  If the swap limit is unable to be set, the job will still be allowed to run. All other cgroup-related failures will cause the job to be rejected. </div> <p>When requesting swap, it is not required that you give a value for the <code>:memory</code> option:</p> <ul style="list-style-type: none"> <li>If using <code>:swap</code> <i>without</i> a specified <code>:memory</code> value, Torque will supply a memory value up to the value of <code>:swap</code>; but not larger than available physical memory. <div> <pre>qsub -L tasks=4:lprocs=2:swap=4.5gb</pre> <div> <i>Allocates four tasks with two logical processors each. Each task is given a combined limit of 4.5 GB of resident memory and swap space. If a task exceeds the limit, the task is terminated.</i> </div> </div> </li> <li>If using <code>:swap</code> <i>with</i> a specified <code>:memory</code> value, Torque will only supply resident memory up to the <code>:memory</code> value. The rest of the swap can <i>only</i> be supplied from the swap space. <div>  The <code>:memory</code> value <i>must</i> be smaller than or equal to the <code>:swap</code> value. <div> <pre>qsub -L tasks=2:memory=3.5gb:swap=5gb</pre> <div> <i>Allocates two tasks and each task has up to 3.5 GB of resident memory and a maximum of 5 GB of swap. If a task exceed 3.5 GB of resident memory, the excess will be moved to the swap space. However, if the task exceed 5 GB of total swap, the task and job will be terminated.</i> </div> </div> </div> </li> </ul>
tasks	<p>Specifies the quantity of job tasks for which the resource request describes the resources needed by a single task:</p> <ul style="list-style-type: none"> <li><b>Distributed memory systems</b> - A single task must run within a single compute node/server (i.e., the task's resources must all come from the same compute node/server).</li> <li><b>Shared memory systems</b> - A single task can run on multiple compute nodes (i.e., the task's resources may come from multiple compute nodes).</li> </ul> <p>This option is <i>required</i> for task-based resource allocation and placement.</p> <div> <pre>qsub -L tasks=4</pre> <div> <i>Creates four tasks, each with one logical process. The tasks can be run on a core or thread (default allowthreads).</i> </div> </div>
usecores, usethreads,	The usecores, usethreads, and allowthreads parameters are used to indicate

Value	Description
allow threads	<p>whether the cgroup pins cores, threads, or either to a task, respectively. If no logical processor definition is given, the default is allowthreads for backward-compatible Moab scheduler and Torque resource manager behavior.</p> <p>In this context, 'cores' means an AMD Opteron core, a hyperthread-disabled Intel Xeon core, or thread 0 <i>and only thread 0</i> of a hyperthread-enabled Intel Xeon core. The term 'threads' refers to a hyperthread-enabled Intel Xeon thread. Likewise, 'either' refers to an AMD Opteron core, a hyperthread-disabled Intel Xeon core, or any thread of a hyperthread-enabled Intel Xeon.</p> <ul style="list-style-type: none"> <li>• <b>:usecores</b> – Denotes that the logical processor definition for a task resource request is a physical core. This means if a core has hyper-threading enabled, the task will use only thread 0 of the core. <div> <pre>qsub -L tasks=2:lprocs=2:usecores</pre> <p><i>Two tasks are allocated with two logical processors per task. The usecores parameter indicates the processor types must be a core or thread 0 of a hyper-threaded core.</i></p> </div> </li> <li>• <b>:usethreads</b> – Specifies the logical processor definition for a task resource request is a hardware-based thread or virtual core. <div> <pre>qsub -L tasks=2:lprocs=2:usethreads</pre> <p><i>Two tasks are allocated with two logical processors per task. The usethreads parameter indicates that any type of hardware-based thread or virtual core can be used.</i></p> </div> </li> <li>• <b>:allowthreads</b> – Specifies that the logical processor definition for a task resource request can be either a physical core (e.g., AMD Opteron), or hardware-based thread of a core (hyperthread-enabled Intel Xeon). <div> <pre>qsub -L tasks=2:lprocs=2:allowthreads</pre> <p><i>Two tasks are allocated with two logical processors per task. The allowthreads parameter indicates hardware threads or cores can be used.</i></p> </div> </li> </ul>

## Related Topics

- [A.23 qsub](#)
- [3.1.4 Requesting NUMA-Aware Resources](#)

## 12.5 pbsnodes with NUMA-Awareness

When Torque is configured with NUMA-awareness and configured with `--enable-cgroups`, the number of total *and* the number of available sockets, numachips (numa nodes), cores, and threads are returned when the status of nodes are queried by Moab (a call is made to `pbsnodes`).

The example output that follows shows a node with two sockets, four numachips, 16 cores and 32 threads. In this example, no jobs are currently running on this node; therefore, the available resources are the same as the total resources.

```
torque-devtest-01
  state = free
  power_state = Running
  np = 16
  ntype = cluster
  status =
rectime=1412732948,macaddr=00:26:6c:f4:66:a0,cpuclock=Fixed,varattr=,jobs=,state=free,
netload=17080856592,gres=,loadave=10.74,ncpus=16,physmem=49416100kb,availmem=50056608k
b,totmem=51480476kb,idletime=29,nusers=2,nsessions=3,sessions=8665
8671 1994,uname=Linux torque-devtest-01 2.6.32-358.el6.x86_64 #1 SMP
Fri Feb 22 00:31:26 UTC 2025 x86_64,opsys=linux
  mom_service_port = 15002
  mom_manager_port = 15003
  total_sockets = 2
  total_numachips = 4
  total_cores = 16
  total_threads = 32
  available_sockets = 2
  available_numachips = 4
  available_cores = 16
  available_threads = 32
```

However, if a job requesting only a single core was started on this node, the `pbsnodes` output will look like this:

```
torque-devtest-01
  state = free
  power_state = Running
  np = 16
  ntype = cluster
  jobs = 0/112.torque-devtest-01
  status =
rectime=1412732948,macaddr=00:26:6c:f4:66:a0,cpuclock=Fixed,varattr=,jobs=,state=free,
netload=17080856592,gres=,loadave=10.74,ncpus=16,physmem=49416100kb,availmem=50056608k
b,totmem=51480476kb,idletime=29,nusers=2,nsessions=3,sessions=8665
8671 1994,uname=Linux torque-devtest-01 2.6.32-358.el6.x86_64 #1 SMP
Fri Feb 22 00:31:26 UTC 2025 x86_64,opsys=linux
  mom_service_port = 15002
  mom_manager_port = 15003
  total_sockets = 2
  total_numachips = 4
  total_cores = 16
  total_threads = 32
  available_sockets = 1
  available_numachips = 3
  available_cores = 15
  available_threads = 30
```



Looking at the output for this example, you will see that even though only one core was requested, the available sockets, numachip, cores and threads were all reduced. This is because the NUMA architecture is hierarchical: socket contains one or more numachips; a numachip contains two or more cores; cores contain one or more threads (one thread in the case of non-threaded cores). In order for a resource to be available, the entire resource must be free. When a job uses one core, the use of that core consumes part of the associated socket, and numa chip resources. As a result, the affected socket and numachip cannot be used when subsequent jobs request sockets and numachips as resources. Also, because the job asked for one core, the number of threads for that core are consumed. As a result, the number of threads available on the machine is reduced by the number of threads in the core.

As another example, suppose a user makes a job request and they want to use a socket. The pbsnodes output will look like this:

```
torque-devtest-01
  state = free
  power state = Running
  np = 16
  ntype = cluster
  jobs = 113.torque-devtest-01
  status =
rectime=1412732948,macaddr=00:26:6c:f4:66:a0,cpuclock=Fixed,varattr=,jobs=,state=free,
netload=17080856592,gres=,loadave=10.74,ncpus=16,physmem=49416100kb,availmem=50056608k
b,totmem=51480476kb,idletime=29,nusers=2,nsessions=3,sessions=8665
8671 1994,uname=Linux torque-devtest-01 2.6.32-358.el6.x86_64 #1 SMP
Fri Feb 22 00:31:26 UTC 2025 x86_64,opsys=linux
  mom_service_port = 15002
  mom_manager_port = 15003
  total_sockets = 2
  total_numachips = 4
  total_cores = 16
  total_threads = 32
  available_sockets = 1
  available_numachips = 2
  available_cores = 8
  available_threads = 16
```

Looking at the output in this example, you will see that not only are the available sockets reduced to one, but all of the numachips, cores, and threads associated with the socket are no longer available. In other words, by requesting a job placement of 'socket' all of the resources of the socket are reserved and are no longer available to other jobs.

## 12.6 NUMA-Support Systems

This section serves as a central information repository for NUMA-support systems, and provides basic information and contains links to the various NUMA-aware topics found throughout the documentation.

**i** Support for NUMA-support systems is available only on large-scale SLES systems using SGI Altix and UV hardware.

In this section:

[12.6.1 About NUMA-Supported Systems](#)

[12.6.2 Torque Installation and Configuration](#)

[12.6.3 Moab/Torque NUMA Configuration](#)

## 12.6.1 About NUMA-Supported Systems

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 instance 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 admin wants Torque to view the system. Each line in *mom.layout* equates to a node in the cluster and is referred to as a NUMA node.

## 12.6.2 Torque Installation and Configuration

To enable Torque for NUMA-support, you will need to add the `--enable-numa-support` option during the configure portion of the installation. You will also need create the *mom.layout* file and configure the *server\_priv/nodes* file.

**i** With SGI Altix systems, each node must be configured on its own partition, so Moab does not schedule across Altix systems. Non-Altix nodes must be on a different partition than Altix systems.

See [2.7.2 Torque NUMA-Support Configuration](#) for more information.

## 12.6.3 Moab/Torque NUMA Configuration

Moab *requires* additional configuration to enable NUMA-support. See 'Moab-NUMA-Support Integration Guide' in the *Moab Workload Manager Administrator Guide*.

## Chapter 13: 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.

**Note:** If you currently have a support services contract and encounter an installation problem that you can't resolve, submit an [online support case](#), and a technical support specialist will contact you.

In this chapter:

- [13.1 Automatic Queue and Job Recovery](#)
- [13.2 Host Resolution](#)
- [13.3 Firewall Configuration](#)
- [13.4 Torque Log Files](#)
- [13.5 Using tracejob to Locate Job Failures](#)
- [13.6 Using GDB to Locate Job Failures](#)
- [13.7 Other Diagnostic Options](#)
- [13.8 Stuck Jobs](#)
- [13.9 Frequently Asked Questions](#)
- [13.10 Compute Node Health Check](#)
- [13.11 Debugging](#)

### 13.1 Automatic Queue and Job Recovery

When `pbs_server` restarts and recovers a job but cannot find that job's queue, it will create a new queue with the original name, but with a `ghost_queue` attribute (as seen in `qmgr`) and then add the job to that queue. This will happen for each queue the server does not recognize. Ghost queues will not accept new jobs, but will allow the jobs in it to run and be in a running state. If users attempt to submit any new jobs to these queues, the user will get an error stating that the queue had an error on recovery, and is in a ghost state. Once the admin reviews and corrects the queue's settings, the admin can remove the ghost setting and then the queue will function normally. See the attribute [ghost\\_queue](#) for more information.

## 13.2 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 can also be used. Commands such as `nslookup` or `dig` can be used to verify proper host resolution.

**i** Invalid host resolution may exhibit itself with compute nodes reporting as down within the output of `pbsnodes -a` and with failure of the `momctl -d3` command.

## 13.3 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.

## 13.4 Torque Log Files

In this section:

[13.4.1 pbs\\_server and pbs\\_mom Log Files](#)

[13.4.2 trqauthd Log Files](#)

### 13.4.1 pbs\_server and pbs\_mom 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`.

## 13.4.2 trqauthd Log Files

`trqauthd` logs its events in the `TORQUE_HOME/client_logs` directory. It names the log files in the format `<YYYYMMDD>`, creating a new log daily as events occur.

**i** You might see some peculiar behavior if you mount the `client_logs` directory for shared access via network-attached storage.

When `trqauthd` first gets access on a particular day, it writes an 'open' message to the day's log file. It also writes a 'close' message to the last log file it accessed prior to that, which is usually the previous day's log file, but not always. For example, if it is Monday and no client commands were executed over the weekend, `trqauthd` writes the 'close' message to Friday's file.

Since the various `trqauthd` binaries on the submit hosts (and potentially, the compute nodes) each write an 'open' and 'close' message on the first access of a new day, you'll see multiple (seemingly random) accesses when you have a shared log.

The `trqauthd` records the following events along with the date and time of the occurrence:

- When `trqauthd` successfully starts. It logs the event with the IP address and port.
- When a user successfully authenticates with `trqauthd`.
- When a user fails to authenticate with `trqauthd`.
- When `trqauthd` encounters any unexpected errors.

*Example 13-1: trqauthd logging sample*

```
2025-10-05 15:05:51.8404 Log opened
```

```
2025-10-05 15:05:51.8405 Torque authd daemon started and listening on IP:port
101.0.1.0:12345
2025-10-10 14:48:05.5688 User hfrye at IP:port abc:12345 logged in
```

## 13.5 Using tracejob to Locate Job Failures

In this section:

[13.5.1 Overview](#)

[13.5.2 Syntax](#)

[13.5.3 Example](#)

### 13.5.1 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.

### 13.5.2 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
```

### 13.5.3 Example

```
> tracejob -n 10 1131

Job: 1131.icluster.org
```

```

03/02/2025 17:58:28 S enqueueing into batch, state 1 hop 1
03/02/2025 17:58:28 S Job Queued at request of dev@icluster.org, owner =
dev@icluster.org, job name = STDIN, queue = batch
03/02/2025 17:58:28 A queue=batch
03/02/2025 17:58:41 S Job Run at request of dev@icluster.org
03/02/2025 17:58:41 M evaluating limits for job
03/02/2025 17:58:41 M phase 2 of job launch successfully completed
03/02/2025 17:58:41 M saving task (TMomFinalizeJob3)
03/02/2025 17:58:41 M job successfully started
03/02/2025 17:58:41 M job 1131.koa.icluster.org reported successful start on 1 node
(s)
03/02/2025 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.needsnodes=1 Resource_
List.nodect=1
Resource_List.nodes=1 Resource_List.walltime=00:01:40
03/02/2025 18:02:11 M walltime 210 exceeded limit 100
03/02/2025 18:02:11 M kill_job
03/02/2025 18:02:11 M kill_job found a task to kill
03/02/2025 18:02:11 M sending signal 15 to task
03/02/2025 18:02:11 M kill_task: killing pid 14060 task 1 with sig 15
03/02/2025 18:02:11 M kill_task: killing pid 14061 task 1 with sig 15
03/02/2025 18:02:11 M kill_task: killing pid 14063 task 1 with sig 15
03/02/2025 18:02:11 M kill_job done
03/02/2025 18:04:11 M kill_job
03/02/2025 18:04:11 M kill_job found a task to kill
03/02/2025 18:04:11 M sending signal 15 to task
03/02/2025 18:06:27 M kill_job
03/02/2025 18:06:27 M kill_job done
03/02/2025 18:06:27 M performing job clean-up
03/02/2025 18:06:27 A user=dev group=dev jobname=STDIN queue=batch ctime=1109811508

qtime=1109811508 etime=1109811508 start=1109811521
exec_host=icluster.org/0 Resource_List.needsnodes=1 Resource_
List.nodect=1
Resource_List.nodes=1 Resource_List.walltime=00:01:40
session=14060

end=1109811987 Exit_status=265 resources_used.cput=00:00:00
resources_used.mem=3544kb resources_used.vmem=10632kb
resources_used.walltime=00:07:46

...

```

- i** 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 that 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`

- i** `tracejob` can 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 are changed appropriately.

- i** The value of `Resource_List.*` is the amount of resources requested, and the value of `resources_used.*` is the amount of resources actually used.

## 13.6 Using GDB to Locate Job Failures

If either the `pbs_mom` or `pbs_server` fail unexpectedly (and the log files contain no information on the failure), GDB (GNU Debugger) 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 can be sent to the mailing list or to [help](#) for further assistance.

- i** See the `PBSCOREDUMP` parameter for enabling creation of core files (see [13.11 Debugging](#)).



## 13.7 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
```

## 13.8 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.

## 13.9 Frequently Asked Questions

In this section:

- 13.9.1 Cannot connect to server: error=15034
- 13.9.2 Deleting 'stuck' jobs
- 13.9.3 Which user must run Torque?
- 13.9.4 Scheduler cannot run jobs - rc: 15003
- 13.9.5 PBS\_Server: pbsd\_init, Unable to read server database
- 13.9.6 qsub will not allow the submission of jobs requesting many processors
- 13.9.7 qsub reports 'Bad UID for job execution'
- 13.9.8 Why does my job keep bouncing from running to queued?
- 13.9.9 How do I use PVM with Torque?
- 13.9.10 My build fails attempting to use the TCL library
- 13.9.11 My job will not start, failing with the message 'cannot send job to mom, state=PRERUN'
- 13.9.12 How do I determine what version of Torque I am using?

13.9.13 How do I resolve `autogen.sh` errors that contain "error: possibly undefined macro: `AC_MSG_ERROR`"?

13.9.14 Why are there so many error messages in the client logs (`trqauthd` logs) when I don't notice client commands failing?

## 13.9.1 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.

## 13.9.2 Deleting 'stuck' jobs

To manually delete a 'stale' job that 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. Shut down `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
```

## 13.9.3 Which user must run Torque?

Torque (pbs\_server & pbs\_mom) must be started by a user with root privileges.

## 13.9.4 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 by a user in the server operators/managers list (see [qmgr](#)). The default for the server operators/managers list is `root@localhost`. For Torque to be used in a grid setting with Moab, the scheduler needs to be run as `root`.

## 13.9.5 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 other's 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)

1. 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
set server resources_available.nodect = 80
set server node_ping_rate = 300
set server node_check_rate = 600
set server tcp_timeout = 6
```

2. Copy this information somewhere.

3. Restart `pbs_server`:

```
> pbs_server -t create
```

4. When you are prompted to overwrite the previous database, enter `y`, then enter the data exported by the `qmgr` command as in this example:

```
> cat data | qmgr
```

5. Restart `pbs_server` without the flags:

```
> qterm
> pbs_server
```

This will reinitialize the database to the current version.

 Reinitializing the server database will reset the next jobid to 1

### 13.9.6 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 effect, this attribute should be set on both the server and the associated queue as in the example below. See the attribute [resources\\_available](#) for more information.

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

**i** The `pbs_server` daemon will need to be restarted before these changes will take effect.

### 13.9.7 qsub reports 'Bad UID for job execution'

Submitting a job may fail with an error similar to the following:

```
[guest@login2]$ qsub test.job
qsub: submit error (Bad UID for job execution MSG=ruserok failed validating
guest/guest from login2)
```

This usually means that the host you are submitting the job from is not registered as a trusted submission host within Torque. In the example above, the host `login2` is not configured to be trusted.

To check what hosts are trusted as submission hosts, run the following on the Torque server host:

```
[root@torque-server-host]# qmgr -c "print server" | grep submit_hosts
```

If you do not see the host that you submitted the job from, you can add it by doing the following:

```
[root@torque-server-host]# qmgr -c "set server submit_hosts += login2"
```

For more information see [2.3.2.C Configuring Job Submission Hosts](#).

This error may also occur when using an identity and credential management system, such as Centrify and the identity management system has cached user credentials. To resolve this issue, flush the credential cache (for example, using Centrify's `adflush` command).

### 13.9.8 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.

### 13.9.9 How do I use PVM with Torque?

Start the master `pvm` 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`).

**i** 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.

### 13.9.10 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.

### 13.9.11 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 automatically handles this when the `mom_job_sync` parameter is set via `qmgr` (the default).

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

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

Run either of the following commands:

```
qstat --version
```

```
pbs_server --about
```

### 13.9.13 How do I resolve `autogen.sh` errors that contain "error: possibly undefined macro: AC\_MSG\_ERROR"?

Verify the `pkg-config` package is installed.

### 13.9.14 Why are there so many error messages in the client logs (`trqauthd` logs) when I don't notice client commands failing?

If a client makes a connection to the server and the `trqauthd` connection for that client command is authorized *before* the client's connection, the `trqauthd` connection is rejected. The connection is retried, but if all retry attempts are rejected, `trqauthd` logs a message indicating a failure. Some client commands then open a new connection to the server and try again. The client command fails only if all its retries fail.

## 13.10 Compute Node Health Check

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 admins 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 'Moab Parameters' in the *Moab Workload Manager Administrator Guide* for more information.

Additionally, Michael Jennings at LBNL has authored an open-source bash node health check script project. It offers an easy way to perform some of the most common node health checking tasks, such as verifying network and filesystem functionality. More information is available on the [project's page](#).

In this section:

[13.10.1 Configuring MOMs to Launch a Health Check](#)

[13.10.2 Creating the Health Check Script](#)

[13.10.3 Adjusting Node State Based on the Health Check Output](#)

[13.10.4 Example Health Check Script](#)

## 13.10.1 Configuring MOMs to Launch a Health Check

The health check feature is configured via the `mom_priv/config` file using the parameters described below:

Parameter	Format	Default	Description
<b>\$node_check_script</b>	<STRING>	N/A	(Required) Specifies the fully qualified pathname of the health check script to run.
<b>\$node_check_interval</b>	<INTEGER>	1	<p>(Optional) Specifies the number of MOM intervals between health checks (by default, each MOM interval is 45 seconds long - this is controlled via the <code>\$status_update_time</code> node parameter. The integer may be followed by a list of event names (<code>jobstart</code> and <code>jobend</code> are currently supported). See <code>pbs_mom</code> for more information.</p> <div> <p><b>i</b> The node health check can be configured to run before the prologue script by including the 'jobstart' option. However, the job environment variables are not in the health check at that point.</p> </div>

### 13.10.2 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 can be a script or compile executable program. It can make any needed system calls and execute any combination of system utilities but should not execute resource manager client commands. Also, 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.

By default, the script looks for the `EVENT :` keyword to indicate successes. 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 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.

**i** In order for the node health check script to log a positive run, it is necessary to include the keyword `EVENT :` at the beginning of the message your script returns. Failure to do so may result in unexpected outcomes.

**i** Both the `ERROR` and `EVENT :` keywords are case insensitive.

### 13.10.3 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. The node health script continues to run at the configured interval (see [13.10.1 Configuring MOMs to Launch a Health Check](#) for more information), and if it does not generate the error message again during one of its later executions, Moab picks that up at the beginning of its next iteration and restores the node to an online state.

### 13.10.4 Example Health Check Script

As mentioned, the health check can be a shell script, PERL, Python, C-executable, or anything that 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
```

## 13.11 Debugging

In this section:

[13.11.1 Diagnostic and Debug Options](#)

[13.11.2 Torque Error Codes](#)

### 13.11.1 Diagnostic and Debug Options

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 *pbs\_mom* or *pbs\_server* daemon to create a core file if a *SIGSEGV*, *SIGILL*, *SIGFPE*, *SIGSYS*, or *SIGTRAP* signal is received.

**i** To enable core dump file creation in systems using *systemd*, add this line to the *trqauthd.service*, *pbs\_mom.service*, and *pbs\_server.service* unit files (in */usr/lib/systemd/system/*):

```
LimitCORE=infinity
```

Core dumps will be placed in the daemons' home directories as shown in the table below.

Daemon	Path
trqauthd	/var/spool/torque

Daemon	Path
pbs_server	/var/spool/torque/server_priv
pbs_mom	/var/spool/torque/mom_priv

- *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 (see [13.5 Using tracejob to Locate Job Failures](#)) for more information.

**i** *PBSLOGLEVEL* and *PBSCOREDUMP* must be added to the `PBSHOME/pbs_environment` file, not just the current environment. To set these variables, add a line to the `pbs_environment` file as either 'variable=value' or just 'variable'. In the case of 'variable=value', the environment variable is set up as the value specified. In the case of 'variable', the environment variable is set based upon its value in the current environment.

### 13.11.2 Torque Error Codes

Error Code Name	Number	Description
<b>PBSE_FLOOR</b>	15000	No error
<b>PBSE_UNKJOBID</b>	15001	Unknown job identifier
<b>PBSE_NOATTR</b>	15002	Undefined attribute
<b>PBSE_ATTRRO</b>	15003	Attempt to set READ ONLY attribute
<b>PBSE_IVALREQ</b>	15004	Invalid request
<b>PBSE_UNKREQ</b>	15005	Unknown batch request
<b>PBSE_TOOMANY</b>	15006	Too many submit retries
<b>PBSE_PERM</b>	15007	No permission
<b>PBSE_MUNGE_NOT_FOUND</b>	15009	"munge" executable not found; unable to authenticate

Error Code Name	Number	Description
<b>PBSE_BADHOST</b>	15010	Access from host not allowed
<b>PBSE_JOBEXIST</b>	15011	Job already exists
<b>PBSE_SYSTEM</b>	15012	System error occurred
<b>PBSE_INTERNAL</b>	15013	Internal server error occurred
<b>PBSE_REGROUTE</b>	15014	Parent job of dependent in rte queue
<b>PBSE_UNKSIG</b>	15015	Unknown signal name
<b>PBSE_BADATVAL</b>	15016	Bad attribute value
<b>PBSE_MODATTRRUN</b>	15017	Cannot modify attribute in run state
<b>PBSE_BADSTATE</b>	15018	Request invalid for job state
<b>PBSE_UNKQUE</b>	15020	Unknown queue name
<b>PBSE_BADCRED</b>	15021	Invalid credential in request
<b>PBSE_EXPIRED</b>	15022	Expired credential in request
<b>PBSE_QUNOENB</b>	15023	Queue not enabled
<b>PBSE_QACCESS</b>	15024	No access permission for queue
<b>PBSE_BADUSER</b>	15025	Bad user - no password entry
<b>PBSE_HOPCOUNT</b>	15026	Max hop count exceeded
<b>PBSE_QUEEXIST</b>	15027	Queue already exists
<b>PBSE_ATTRTYPE</b>	15028	Incompatible queue attribute type
<b>PBSE_QUEBUSY</b>	15029	Queue busy (not empty)
<b>PBSE_QUENBIG</b>	15030	Queue name too long

Error Code Name	Number	Description
<b>PBSE_NOSUP</b>	15031	Feature/function not supported
<b>PBSE_QUENOEN</b>	15032	Cannot enable queue, needs add def
<b>PBSE_PROTOCOL</b>	15033	Protocol (ASN.1) error
<b>PBSE_BADATLST</b>	15034	Bad attribute list structure
<b>PBSE_NOCONNECTS</b>	15035	No free connections
<b>PBSE_NOSERVER</b>	15036	No server to connect to
<b>PBSE_UNKRESC</b>	15037	Unknown resource
<b>PBSE_EXCQRESC</b>	15038	Job exceeds queue resource limits
<b>PBSE_QUENODFLT</b>	15039	No default queue defined
<b>PBSE_NORERUN</b>	15040	Job not rerunnable
<b>PBSE_ROUTEREJ</b>	15041	Route rejected by all destinations
<b>PBSE_ROUTEEXPD</b>	15042	Time in route queue expired
<b>PBSE_MOMREJECT</b>	15043	Request to MOM failed
<b>PBSE_BADSCRIPT</b>	15044	(qsub) Cannot access script file
<b>PBSE_STAGEIN</b>	15045	Stage-In of files failed
<b>PBSE_RESCUNAV</b>	15046	Resources temporarily unavailable
<b>PBSE_BADGRP</b>	15047	Bad group specified
<b>PBSE_MAXQUED</b>	15048	Max number of jobs in queue
<b>PBSE_CKPBSY</b>	15049	Checkpoint busy, may be retries
<b>PBSE_EXLIMIT</b>	15050	Limit exceeds allowable

Error Code Name	Number	Description
<b>PBSE_BADACCT</b>	15051	Bad account attribute value
<b>PBSE_ALRDYEXIT</b>	15052	Job already in exit state
<b>PBSE_NOCOPYFILE</b>	15053	Job files not copied
<b>PBSE_CLEANEOUT</b>	15054	Unknown job ID after clean init
<b>PBSE_NOSYNCMSTR</b>	15055	No master in sync set
<b>PBSE_BADDEPEND</b>	15056	Invalid dependency
<b>PBSE_DUPLIST</b>	15057	Duplicate entry in list
<b>PBSE_DISPROTO</b>	15058	Bad DIS based request protocol
<b>PBSE_EXECTHERE</b>	15059	Cannot execute there
<b>PBSE_SISREJECT</b>	15060	Sister rejected
<b>PBSE_SISCOMM</b>	15061	Sister could not communicate
<b>PBSE_SVRDOWN</b>	15062	Requirement rejected -server shutting down
<b>PBSE_CKPSHORT</b>	15063	Not all tasks could checkpoint
<b>PBSE_UNKNODE</b>	15064	Named node is not in the list
<b>PBSE_UNKNODEATR</b>	15065	Node-attribute not recognized
<b>PBSE_NONODES</b>	15066	Server has no node list
<b>PBSE_NODENBIG</b>	15067	Node name is too big
<b>PBSE_NODEEXIST</b>	15068	Node name already exists
<b>PBSE_BADNDATVAL</b>	15069	Bad node-attribute value
<b>PBSE_MUTUALEX</b>	15070	State values are mutually exclusive

Error Code Name	Number	Description
<b>PBSE_GMODERR</b>	15071	Error(s) during global modification of nodes
<b>PBSE_NORELYMOM</b>	15072	Could not contact MOM
<b>PBSE_NOTSNODE</b>	15073	No time-shared nodes
<b>PBSE_JOBTYPE</b>	15074	Wrong job type
<b>PBSE_BADACLHOST</b>	15075	Bad ACL entry in host list
<b>PBSE_MAXUSERQUED</b>	15076	Maximum number of jobs already in queue for user
<b>PBSE_BADDISALLOWTYPE</b>	15077	Bad type in "disallowed_types" list
<b>PBSE_NOINTERACTIVE</b>	15078	Interactive jobs not allowed in queue
<b>PBSE_NOBATCH</b>	15079	Batch jobs not allowed in queue
<b>PBSE_NORERUNABLE</b>	15080	Rerunable jobs not allowed in queue
<b>PBSE_NONONRERUNABLE</b>	15081	Non-rerunable jobs not allowed in queue
<b>PBSE_UNKARRAYID</b>	15082	Unknown array ID
<b>PBSE_BAD_ARRAY_REQ</b>	15083	Bad job array request
<b>PBSE_TIMEOUT</b>	15084	Time out
<b>PBSE_JOBNOTFOUND</b>	15085	Job not found
<b>PBSE_NOFAULTTOLERANT</b>	15086	Fault tolerant jobs not allowed in queue
<b>PBSE_NOFAULTINTOLERANT</b>	15087	Only fault tolerant jobs allowed in queue
<b>PBSE_NOJOBARRAYS</b>	15088	Job arrays not allowed in queue
<b>PBSE_RELAYED_TO_MOM</b>	15089	Request was relayed to a MOM
<b>PBSE_MEM_MALLOC</b>	15090	Failed to allocate memory for memmgr



Error Code Name	Number	Description
<b>PBSE_MUTEX</b>	15091	Failed to allocate controlling mutex (lock/unlock)
<b>PBSE_TRHEADATTR</b>	15092	Failed to set thread attributes
<b>PBSE_THREAD</b>	15093	Failed to create thread
<b>PBSE_SELECT</b>	15094	Failed to select socket
<b>PBSE_SOCKET_FAULT</b>	15095	Failed to get connection to socket
<b>PBSE_SOCKET_WRITE</b>	15096	Failed to write data to socket
<b>PBSE_SOCKET_READ</b>	15097	Failed to read data from socket
<b>PBSE_SOCKET_CLOSE</b>	15098	Socket closed
<b>PBSE_SOCKET_LISTEN</b>	15099	Failed to listen in on socket
<b>PBSE_AUTH_INVALID</b>	15100	Invalid auth type in request
<b>PBSE_NOT_IMPLEMENTED</b>	15101	Functionality not yet implemented
<b>PBSE_QUENOTAVAILABLE</b>	15102	Queue is not available

## Appendix A: Commands Overview

In this appendix:

- [A.1 Torque Services](#)
- [A.2 Client Commands](#)
- [A.3 momctl](#)
- [A.4 pbs\\_mom](#)
- [A.5 pbs\\_server](#)
- [A.6 pbs\\_track](#)
- [A.7 pbsdsh](#)
- [A.8 pbsnodes](#)
- [A.9 qalter](#)
- [A.10 qchkpt](#)
- [A.11 qdel](#)
- [A.12 qgpumode](#)
- [A.13 qgpureset](#)
- [A.14 qhold](#)
- [A.15 qmgr](#)
- [A.16 qmove](#)
- [A.17 qorder](#)
- [A.18 qrerun](#)
- [A.19 qrls](#)
- [A.20 qrun](#)
- [A.21 qsig](#)
- [A.22 qstat](#)
- [A.23 qsub](#)
- [A.24 qterm](#)
- [A.25 trqauthd](#)

---

### Related Topics

- [Appendix B: Server Parameters](#)
- [Appendix C: Node Manager \(MOM\) Configuration](#)

## A.1 Torque Services

Command	Description
<code>pbs_mom</code>	PBS batch execution mini-server. Runs on each Torque compute node.
<code>pbs_server</code>	Batch system manager daemon. Runs on the Torque master node.
<code>pbs_track</code>	Process launcher. Starts a new process and tells pbs_mom to track its lifecycle and resource usage.
<code>trqauthd</code>	Torque authorization daemon.

## A.2 Client Commands

Command	Description
<code>momctl</code>	Manage/diagnose MOM (node execution) daemon
<code>pbsdsh</code>	Launch tasks within a parallel job
<code>pbsnodes</code>	View/modify batch status of compute nodes
<code>qalter</code>	Modify queued batch jobs
<code>qchkpt</code>	Checkpoint batch jobs
<code>qdel</code>	Delete/cancel batch jobs
<code>qgpumode</code>	Specifies new mode for GPU
<code>qgpureset</code>	Reset the GPU
<code>qhold</code>	Hold batch jobs
<code>qmgr</code>	Manage policies and other batch configuration
<code>qmove</code>	Move batch jobs

Command	Description
<b>qorder</b>	Exchange order of two batch jobs in any queue
<b>qrerun</b>	Rerun a batch job
<b>qrls</b>	Release batch job holds
<b>qrun</b>	Start a batch job
<b>qsig</b>	Send a signal to a batch job
<b>qstat</b>	View queues and jobs
<b>qsub</b>	Submit jobs
<b>qterm</b>	Shutdown pbs server daemon
<b>tracejob</b>	Trace job actions and states recorded in Torque logs (see <a href="#">13.5 Using tracejob to Locate Job Failures</a> )

## A.3 momctl

*(PBS MOM Control)*

### A.3.1 Synopsis

```

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

```

### A.3.2 Overview

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

### A.3.3 Format

-c — Clear	
Format	{ <JOBID>   <i>all</i> }
Default	---
Description	Makes the MOM unaware of the job's existence. It does not clean up any processes associated with the job.
Example	<pre>momctl -c 2 -h node1</pre>

-C — Cycle	
Format	---
Default	---
Description	Cycle <code>pbs_mom</code> (force the MOM to send a status update to <code>pbs_server</code> ).
Example	<pre>momctl -C -h node1</pre> <p><i>Cycle pbs_mom on node1.</i></p>

-d — Diagnose	
Format	{ <INTEGER>   <JOBID> }
Default	0
Description	Diagnose MOM(s). For more details, see <a href="#">A.3.6 Diagnose Detail</a> below.
Example	<pre>momctl -d 2 -h node1</pre> <p><i>Print level 2 and lower diagnostic information for the MOM on node1.</i></p>

-f — Host File	
<b>Format</b>	<FILE>
<b>Default</b>	---
<b>Description</b>	A file containing only comma or whitespace (space, tab, or new line) delimited hostnames.
<b>Example</b>	<pre>momctl -f hosts.txt -d 0</pre> <p><i>Print diagnose information for the MOMs running on the hosts specified in <code>hosts.txt</code>.</i></p>

-h — Host List	
<b>Format</b>	<HOST>[,<HOST>]...
<b>Default</b>	localhost
<b>Description</b>	A comma-separated list of hosts.
<b>Example</b>	<pre>momctl -h node1,node2,node3 -d 0</pre> <p><i>Print diagnose information for the MOMs running on node1, node2, and node3.</i></p>

-l — Layout	
<b>Format</b>	---
<b>Default</b>	---
<b>Description</b>	Display the layout of the machine as it is understood by Torque.

-l — Layout	
Example	<pre>[root@c04a numa]# momctl -l Machine (50329748KB) Socket 0 (33552532KB)   Chip 0 (16775316KB)     Core 0 (1 threads)     Core 1 (1 threads)     Core 2 (1 threads)     Core 3 (1 threads)     Core 4 (1 threads)     Core 5 (1 threads)     Core 6 (1 threads)     Core 7 (1 threads)   Chip 1 (16777216KB)     Core 8 (1 threads)     Core 9 (1 threads)     Core 10 (1 threads)     Core 11 (1 threads)     Core 12 (1 threads)     Core 13 (1 threads)     Core 14 (1 threads)     Core 15 (1 threads) Socket 1 (16777216KB)   Chip 2 (8388608KB)     Core 16 (1 threads)     Core 17 (1 threads)     Core 18 (1 threads)     Core 19 (1 threads)     Core 20 (1 threads)     Core 21 (1 threads)     Core 22 (1 threads)     Core 23 (1 threads)   Chip 3 (8388608KB)     Core 24 (1 threads)     Core 25 (1 threads)     Core 26 (1 threads)     Core 27 (1 threads)     Core 28 (1 threads)     Core 29 (1 threads)     Core 30 (1 threads)     Core 31 (1 threads)</pre>

-p — Port	
Format	<PORT_NUMBER>
Default	Torque's default port number.
Description	The port number for the specified MOM(s).
Example	<pre>momctl -p 5455 -h node1 -d 0</pre> <div>Request diagnose information over port 5455 on node1.</div>

<b>-q — Query</b>	
<b>Format</b>	<ATTRIBUTE>
<b>Default</b>	---
<b>Description</b>	Query or set <ATTRIBUTE>, where <ATTRIBUTE> is a property listed by <code>pbsnodes -a</code> (see <a href="#">A.3.4 Query Attributes</a> for a list of attributes). Can also be used to query or set <code>pbs_mom</code> options (see <a href="#">A.3.5 Resources</a> ).
<b>Example</b>	<pre>momctl -q physmem</pre> <p><i>Print the amount of physmem on localhost.</i></p> <pre>momctl -h node2 -q loglevel=7</pre> <p><i>Change the current MOM logging on node2 to level 7.</i></p>
<b>-r — Reconfigure</b>	
<b>Format</b>	{ <FILE>   LOCAL:<FILE> }
<b>Default</b>	---
<b>Description</b>	Reconfigure MOM(s) with remote or local config file, <FILE>. This does not work if <code>\$remote_reconfig</code> is not set to true when the MOM is started.
<b>Example</b>	<pre>momctl -r /home/user1/new.config -h node1</pre> <p><i>Reconfigure MOM on node1 with /home/user1/new.config on node1.</i></p>
<b>-s — Shutdown</b>	
<b>Format</b>	---
<b>Default</b>	---
<b>Description</b>	Have the MOM shut itself down gracefully (this includes reporting to server so that <code>pbsnodes</code> marks the node down).
<b>Example</b>	<pre>momctl -s</pre> <p><i>Shut down the pbs_mom process on localhost.</i></p>



<b>-u — Update</b>	
<b>Format</b>	---
<b>Default</b>	---
<b>Description</b>	Update the hardware configuration on pbs_server with a layout from the MOM.
<b>Example</b>	<pre>momctl -u</pre> <div>Update pbs_server hardware configuration.</div>

### A.3.4 Query Attributes

Attribute	Description
<b>arch</b>	Node hardware architecture
<b>availmem</b>	Available RAM
<b>loadave</b>	1 minute load average
<b>ncpus</b>	Number of CPUs available on the system
<b>netload</b>	Total number of bytes transferred over all network interfaces
<b>nsessions</b>	Number of sessions active
<b>nusers</b>	Number of users active
<b>physmem</b>	Configured RAM
<b>sessions</b>	List of active sessions
<b>totmem</b>	Configured RAM plus configured swap

### A.3.5 Resources

Resource Manager queries can be made with *momctl -q* options to retrieve and set *pbs\_mom* options. Any configured static resource can be retrieved with a request of the same name. These are resource requests not otherwise documented in the PBS ERS.

Request	Description
<b>cycle</b>	Forces an immediate MOM cycle.
<b>status_update_time</b>	Retrieve or set the <code>\$status_update_time</code> parameter.
<b>check_poll_time</b>	Retrieve or set the <code>\$check_poll_time</code> parameter.
<b>configversion</b>	Retrieve the config version.
<b>jobstartblocktime</b>	Retrieve or set the <code>\$jobstartblocktime</code> parameter.
<b>enablemomrestart</b>	Retrieve or set the <code>\$enablemomrestart</code> parameter.
<b>loglevel</b>	Retrieve or set the <code>\$loglevel</code> parameter.
<b>down_on_error</b>	Retrieve or set the <code>\$down_on_error</code> parameter.
<b>diag0 - diag4</b>	Retrieves varied diagnostic information.
<b>rcpcmd</b>	Retrieve or set the <code>\$rcpcmd</code> parameter.
<b>version</b>	Retrieves the <code>pbs_mom</code> version.

### A.3.6 Diagnose Detail

Level	Description
<b>0</b>	<p>Display the following information:</p> <ul style="list-style-type: none"> <li>• Local hostname</li> <li>• Expected server hostname</li> <li>• Execution version</li> <li>• MOM home directory</li> <li>• MOM config file version (if specified)</li> <li>• Duration MOM has been executing</li> <li>• Duration since last request from <code>pbs_server</code> daemon</li> <li>• Duration since last request to <code>pbs_server</code> daemon</li> <li>• RM failure messages (if any)</li> <li>• Log verbosity level</li> </ul>

Level	Description
	<ul style="list-style-type: none"> <li>Local job list</li> </ul>
<b>1</b>	<p>All information for level 0 plus the following:</p> <ul style="list-style-type: none"> <li>Interval between updates sent to server</li> <li>Number of initialization messages sent to pbs_server daemon</li> <li>Number of initialization messages received from pbs_server daemon</li> <li>Prologue/epilogue alarm time</li> <li>List of trusted clients</li> </ul>
<b>2</b>	<p>All information from level 1 plus the following:</p> <ul style="list-style-type: none"> <li>PID</li> <li>Event alarm status</li> </ul>
<b>3</b>	<p>All information from level 2 plus the following:</p> <ul style="list-style-type: none"> <li>syslog enabled</li> </ul>

**Example A-1: MOM diagnostics**

```
[root@node01]# momctl -dl
Host: node01/node01   Version: x.x.x   PID: 30404
Server[0]: torque-server (10.2.15.70:15001)
  Last Msg From Server: 1275 seconds (StatusJob)
  Last Msg To Server: 42 seconds
HomeDirectory: /var/spool/torque/mom_priv
stdout/stderr spool directory: '/var/spool/torque/spool/' (15518495 blocks available)
MOM active: 260257 seconds
Check Poll Time: 45 seconds
Server Update Interval: 45 seconds
LogLevel: 7 (use SIGUSR1/SIGUSR2 to adjust)
Communication Model: TCP
MemLocked: TRUE (mlock)
TCP Timeout: 300 seconds
Trusted Client List:
10.2.15.3:15003,10.2.15.5:15003,10.2.15.6:15003,10.2.15.70:0,10.2.15.204:15003,127.0.0
.1:0
Copy Command: /bin/scp -rpB
NOTE: no local jobs detected

diagnostics complete
```

**Example A-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
```

## A.4 pbs\_mom

Start a pbs batch execution mini-server.

### A.4.1 Synopsis

```
pbs_mom [-a alarm] [-A alias] [-c config] [-C chkdir] [-d directory] [-f] [-F] [-h help] [-H hostname] [-L logfile] [-M MOMport] [-p|-r] [-P purge] [-R RMPport] [-S serverport] [-v] [-w] [-x]
```

### A.4.2 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 ensure 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.

### A.4.3 Options

Flag	Name	Description
<b>-a</b>	alarm	Specifies 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.
<b>-A</b>	alias	Specifies this multimom's alias name. The alias name needs to be the same name used in the <code>mom.hierarchy</code> file. It is only needed when running multiple MOMs on the same machine. For more information, see <a href="#">2.8 Torque Multi-MOM</a> .
<b>-c</b>	config	Specifies an alternative configuration file, see description below. If this is a relative file name, it will be relative to <code>TORQUE_HOME/mom_priv</code> , (see the <b>-d</b> option). If the specified file cannot be opened, <code>pbs_mom</code> will abort. If the <b>-c</b> option is not supplied, <code>pbs_mom</code> will attempt to open the default configuration file, <code>TORQUE_HOME/mom_priv/config</code> . If this file is not present, <code>pbs_mom</code> will log the fact and continue.
<b>-C</b>	chkdirectory	Specifies the path of the directory used to hold checkpoint files. The default directory is <code>TORQUE_HOME/spool/checkpoint</code> (see the <b>-d</b> option). The directory specified with the <b>-C</b> option must be owned by root and accessible ( <code>rwX</code> ) only by root to protect the security of the checkpoint files.
<b>-d</b>	directory	Specifies the path of the directory that is the home of the server's working files, <code>TORQUE_HOME</code> . This option is typically used along with <b>-M</b> when debugging MOM. The default directory is given by <code>TORQUE_HOME</code> , which is typically <code>/var/spool/torque</code>
<b>-f</b>	force_update	Forces the server to accept an update of the hardware on the node. Should be used the first time <code>pbs_mom</code> is run after a hardware update on a node.
<b>-F</b>	fork	Do not fork. <div> This option is useful when running under <code>systemd</code> (Red Hat 7-based or SUSE 12-based systems).</div>
<b>-h</b>	help	Displays the help/usage message.
<b>-H</b>	hostname	Sets the MOM's hostname. This can be useful on multi-homed networks.

Flag	Name	Description
<b>-L</b>	logfile	Specifies 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 <code>TORQUE_HOME/mom_logs</code> directory (see the <b>-d</b> option).
<b>-M</b>	port	Specifies the port number on which the mini-server (MOM) will listen for batch requests.
<b>-p</b>	poll	Specifies the impact on jobs that 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 <b>-p</b> 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 <b>-p</b> option is mutually exclusive with the <b>-r</b> and <b>-q</b> options.
<b>-P</b>	purge	Specifies the impact on jobs that were in execution when the mini-server shut down. With the <b>-P</b> option, it is assumed that either the entire system has been restarted or the MOM has been down so long that it can no longer guarantee that the <code>pid</code> of any running process is the same as the recorded job process <code>pid</code> of a recovering job. Unlike the <b>-p</b> option, no attempt is made to try and preserve or recover running jobs. All jobs are terminated and removed from the queue.
<b>-q</b>	n/a	Specifies the impact on jobs that were in execution when the mini-server shut down. With the <b>-q</b> option, MOM will allow the processes belonging to jobs to continue to run, but will not attempt to monitor them. The <b>-q</b> option is mutually exclusive with the <b>-p</b> and <b>-r</b> options.
<b>-r</b>	n/a	<p>Specifies the impact on jobs that were in execution when the mini-server shut down. With the <b>-r</b> option, MOM will kill any processes belonging to jobs, mark the jobs as terminated, and notify the batch server that owns the job. The <b>-r</b> option is mutually exclusive with the <b>-p</b> and <b>-q</b> options.</p> <p>Normally the mini-server is started from the system boot file without the <b>-p</b> or the <b>-r</b> option. The mini-server will make no attempt to signal the former session of any job that may have been running when the mini-server terminated. It is assumed that on reboot, all processes have been killed. If the <b>-r</b> option is used following a reboot, process IDs (<code>pids</code>) may be reused and MOM may kill a process that is not a batch session.</p>
<b>-R</b>	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.

Flag	Name	Description
<b>-S</b>	server port	pbs_server port to connect to.
<b>-v</b>	version	Displays version information and exits.
<b>-w</b>	wait_for_server	When started with -w, pbs_moms wait until they get their MOM hierarchy file from <a href="#">pbs_server</a> to send their first update, or until 10 minutes pass. This reduces network traffic on startup and can bring up clusters faster.
<b>-x</b>	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.

### A.4.4 Configuration File

The configuration file, located at `mom_priv/config` by default, can 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).

See [C.1 MOM Parameters](#) for a full list of `pbs_mom` parameters.

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 or printers of different types and could be specified by:

```
tape3480    4
tape3420    2
tapedat     1
hpm527dn    2
epsonc20590 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 that 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.

## A.4.5 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 can be a script or compiled executable program. It can make any needed system calls and execute any combination of system utilities but should not execute resource manager client commands. Also, 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 `ERROR` keyword to `stdout` followed by an error message. The message (up to 1024 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. You can use this behavior with an advanced scheduler, such as Moab Workload Manager, to cause the job to be routed to another node. Torque currently ignores Error messages by default, but you can configure an advanced scheduler to react appropriately.

If the `$down_on_error` MOM setting is enabled, the MOM will set itself to state down and report to `pbs_server`. Additionally, the `$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_server`'s `down_on_error` features enabled. Also see [down\\_on\\_error](#) (in [Appendix B: Server Parameters](#)).

See [13.10.2 Creating the Health Check Script](#) for more information.



## A.4.6 Files

File	Description
<b>TORQUE_HOME/server_name</b>	File containing the <code>pbs_server</code> hostname
<b>TORQUE_HOME/mom_priv</b>	Directory for configuration files ( <code>/var/spool/torque/mom_priv</code> by default)
<b>TORQUE_HOME/mom_logs</b>	Directory for log files recorded by the server
<b>TORQUE_HOME/mom_priv/prologue</b>	The administrative script to be run before job execution
<b>TORQUE_HOME/mom_priv/epilogue</b>	The administrative script to be run after job execution

## A.4.7 Signal Handling

`pbs_mom` handles the following signals:

Signal	Description
<b>SIGHUP</b>	Causes <code>pbs_mom</code> to re-read its configuration file, close and reopen the log file, and reinitialize resource structures.
<b>SIGALRM</b>	The signal is used to limit the time taken by child processes, such as the prologue and epilogue. You can set the alarm timeout with the <code>-a</code> option. If a timeout occurs, it is logged in the <code>pbs_mom</code> log file.
<b>SIGINT and SIGTERM</b>	Results in <code>pbs_mom</code> exiting without terminating any running jobs. This is the action for the following signals as well: <code>SIGXCPU</code> , <code>SIGXFSZ</code> , <code>SIGCPULIM</code> , and <code>SIGSHUTDN</code> .
<b>SIGUSR1, SIGUSR2</b>	Causes the MOM to increase and decrease logging levels, respectively.
<b>SIGPIPE, SIGINFO</b>	Are ignored.
<b>SIGBUS, SIGFPE, SIGILL, SIGTRAP, and</b>	Cause a core dump if the <code>PBSCOREDUMP</code> environmental variable is defined.

Signal	Description
<b>SIGSYS</b>	

All other signals have their default behavior installed.

## A.4.8 Exit Status

If the `pbs_mom` command fails to begin operation, the server exits with a value greater than zero.

### Related Topics

- [pbs\\_server\(8B\)](#)

### Non-Adaptive Computing Topics

- [pbs\\_sched\\_basl\(8B\)](#)
- [pbs\\_sched\\_tcl\(8B\)](#)
- PBS External Reference Specification (included in the Torque download tarball in `doc/v2_2_ers.pdf`)
- [PBS Administrator Guide](#)

## A.5 pbs\_server

(*PBS Server*) pbs batch system manager

### A.5.1 Synopsis

```
pbs_server [-a active] [-A acctfile] [-c] [-d config_path] [-f
force overwrite] [-F] [-H hostname] [--ha] [-l location] [-L
logfile] [-n don't send hierarchy] [-p port] [-S scheduler_
port] [-t type] [-v] [--about] [--version]
```

### A.5.2 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 ensure 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.

The `pbs_server` is multi-threaded, which leads to quicker response to client commands, is more robust, and allows for higher job throughput.

### A.5.3 Options

Option	Name	Description
<b>-a</b>	active	Specifies if scheduling is active or not. This sets the <code>scheduling</code> server attribute. 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. Jobs would then need to be run manually or via an external scheduler. If this option is not specified, the server will retain the prior value of the scheduling attribute.
<b>-A</b>	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 <code>TORQUE_HOME/server_priv/accounting</code> directory.
<b>-c</b>	wait_for_moms	This directs <code>pbs_server</code> to send the MOM hierarchy only to MOMs that request it for the first 10 minutes. After 10 minutes, it attempts to send the MOM hierarchy to MOMs that haven't requested it already. This greatly reduces traffic on start up.
<b>-d</b>	config_directory	Specifies the path of the directory that is home to the server's configuration files, <code>PBS_HOME</code> . A host may have multiple servers. Each server must have a different configuration directory. The default configuration directory is given by the symbol <code>TORQUE_HOME</code> , which is typically <code>var/spool/torque</code> .
<b>-f</b>	force overwrite	Forces an overwrite of the server database. This can be useful to bypass the <code>yes/no</code> prompt when running something like <code>pbs_server -t create</code> and can ease installation and configuration of Torque via scripts.
<b>-F</b>	fork	Do not fork.

Option	Name	Description
		<div>  This option is useful when running under systemd (Red Hat 7-based or SUSE 12-based systems). </div>
<b>--ha</b>	high_availability	Starts server in high availability mode (for details, see <a href="#">5.2 Server High Availability</a> ).
<b>-H</b>	hostname	Causes the server to start under a different hostname as obtained from <code>gethostname(2)</code> . 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 <code>gethostname(2)</code> .
<b>-l</b>	location	<p>Specifies where to find the scheduler (for example, Moab) when it does not reside on the same host as Torque.</p> <div> <pre>pbs_server -l &lt;other_host&gt;:&lt;other_port&gt;</pre> </div>
<b>-L</b>	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 <code>PBS_HOME/server_logs</code> directory.
<b>-n</b>	no send	This directs <code>pbs_server</code> to not send the hierarchy to all the MOMs on startup. Instead, the hierarchy is only sent if a MOM requests it. This flag works only in conjunction with the local MOM hierarchy feature. See <a href="#">2.3.3 Setting Up the MOM Hierarchy (Optional)</a> .
<b>-p</b>	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.
<b>-S</b>	scheduler_port	Specifies the port number to which the server should connect when contacting the scheduler. The argument <code>scheduler_conn</code> is of the same syntax as under the <code>-M</code> option.
<b>-t</b>	type	<p>If the job is rerunnable or restartable, and <code>-t create</code> is specified, the server will discard any existing configuration files, queues, and jobs, and initialize configuration files to the default values. The server is idled.</p> <div>  If <code>-t</code> is not specified, the job states will remain the same. </div>

## A.5.4 Files

File	Description
<b>TORQUE_HOME/server_priv</b>	Default directory for configuration files, typically /var/spool/torque/server_priv
<b>TORQUE_HOME/server_logs</b>	Directory for log files recorded by the server

## A.5.5 Signal Handling

On receipt of the following signals, the server performs the defined action:

Action	Description
<b>SIGHUP</b>	The current server log and accounting log are closed and new log files opened. This allows for the prior logs to be renamed and new logs started from the time of the signal. Usage example: <pre># mv 20170816 20170816.old &amp;&amp; kill -HUP \$(pgrep pbs_server)</pre>
<b>SIGINT</b>	Causes an orderly shutdown of pbs_server.
<b>SIGUSR1, SIGUSR2</b>	Causes server to increase and decrease logging levels, respectively.
<b>SIGTERM</b>	Causes an orderly shutdown of pbs_server.
<b>SIGSHUTDN</b>	On systems (Unicos) where SIGSHUTDN is defined, it also causes an orderly shutdown of the server.
<b>SIGPIPE</b>	This signal is ignored.

All other signals have their default behavior installed.

## A.5.6 Exit Status

If the server command fails to begin batch operation, the server exits with a value greater than zero.

---

## Related Topics

- [pbs\\_mom \(8B\)](#)
- [pbsnodes \(8B\)](#)
- [qmgr \(1B\)](#)
- [qrun \(8B\)](#)
- [qsub \(1B\)](#)
- [qterm \(8B\)](#)

## Non-Adaptive Computing Topics

- [pbs\\_connect\(3B\)](#)
- [pbs\\_sched\\_basl\(8B\)](#)
- [pbs\\_sched\\_tcl\(8B\)](#)
- [qdisable\(8B\)](#)
- [qenable\(8B\)](#)
- [qstart\(8B\)](#)
- [qstop\(8B\)](#)
- PBS External Reference Specification (included in the Torque download tarball in [doc/v2\\_2\\_ers.pdf](#))

## A.6 pbs\_track

Starts a specified executable and directs `pbs_mom` to start monitoring its lifecycle and resource usage.

### A.6.1 Synopsis

```
pbs_track -j <JOBID> [-b] [-a] <executable> [args]
```

### A.6.2 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 instructs 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.

A.6.3 Options

Option	Description
-a	Adopt a process into a running job. The user must either own the process or have permission to adopt it. <div><pre>pbs_track -j 3 -a 12345</pre><div>Adopts process 12345 into job 3.</div></div>
-b	Instead of having <code>pbs_track</code> send its PID to Torque, it will <code>fork()</code> first, send the child PID to Torque, and then execute from the forked child. This essentially 'backgrounds' <code>pbs_track</code> so that it will return after the new process is launched.
-j <JOBID>	Job ID the new process should be associated with.

A.6.4 Operands

The `pbs_track` command accepts a path to a program/executable (`<executable>`) and, optionally, one or more arguments to pass to that program.

A.6.5 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.

---

## Related Topics

- [pbsdsh\(1B\)](#)

## Non-Adaptive Computing Topics

- [tm\\_spawn\(3B\)](#)

## A.7 pbsdsh

The *pbsdsh* command distributes tasks to nodes under pbs.

**i** Some limitations exist in the way that *pbsdsh* can be used. Note the following situations are not currently supported:

- Running multiple instances of *pbsdsh* concurrently within a single job.
- Using the *-o* and *-s* options concurrently; although requesting these options together is permitted, only the output from the first node is displayed rather than output from every node in the chain.

### A.7.1 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]
```

### A.7.2 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.

**i** Note that under particularly high workloads, the `pbsdsh` command may not function properly.

### A.7.3 Options

Option	Name	Description
<b>-c</b>	copies	The program is spawned on the first Copies nodes allocated. This option is mutually exclusive with <code>-n</code> .
<b>-h</b>	hostname	The program is spawned on the node specified.
<b>-n</b>	node	The program is spawned on one node, which is the n-th node allocated. This option is mutually exclusive with <code>-c</code> .
<b>-o</b>	---	Capture stdout of the spawned program. Normally stdout goes to the job's output.
<b>-s</b>	---	If this option is given, the program is run in turn on each node, one after the other.
<b>-u</b>	---	The program is run once on each node (unique). This ignores the number of allocated processors on a given node.
<b>-v</b>	---	Verbose output about error conditions and task exit status is produced.

### A.7.4 Operands

The first operand, program, is the program to execute. Additional operands are passed as arguments to the program.

### A.7.5 Standard Error

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

## A.7.6 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.

### Related Topics

- [qsub\(1B\)](#)

### Non-Adaptive Computing Topics

- [tm\\_spawn\(3B\)](#)

## A.8 pbsnodes

PBS node manipulation.

### A.8.1 Synopsis

```
pbsnodes [-{a|x|xml|-xml}] [-q] [-s server] [node[:property]
pbsnodes -l [-q] [-s server] [state] [nodename[:property ...]
pbsnodes -m <running|standby|suspend|hibernate|shutdown> <host
list>
pbsnodes [-{c|d|o|r}] [-q] [-s server] [-n -l] [-N "note"] [-A
"append note"] [node[:property]
```

### A.8.2 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. For more information, see [9.4 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 absence 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.

## A.8.3 NUMA-Awareness

When Torque is configured with NUMA-awareness and configured with `--enable-groups`, the number of total *and* the number of available sockets, numachips (numa nodes), cores, and threads are returned when the status of nodes are queried by Moab (a call is made to `pbsnodes`). See the section [12.5 pbsnodes with NUMA-Awareness](#) for additional information and examples.

## A.8.4 Options

Option	Description
<b>-a</b>	All attributes of a node or all nodes are listed. This is the default if no flag is given.
<b>-A</b>	Append a note attribute to existing note attributes. The <code>-N</code> note option will overwrite exiting note attributes. <code>-A</code> will append a new note attribute to the existing note attributes delimited by a ',' and a space.
<b>-c</b>	Clear OFFLINE from listed nodes.
<b>-d</b>	Print MOM diagnosis on the listed nodes. Not yet implemented. Use <a href="#">momctl</a> instead.
<b>-l</b>	<p>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. State strings are 'active', 'all', 'busy', 'down', 'free', 'job-exclusive', 'job-sharing', 'offline', 'reserve', 'state-unknown', 'time-shared', and 'up'.</p> <ul style="list-style-type: none"> <li>Using <i>all</i> displays all nodes and their attributes.</li> <li>Using <i>active</i> displays all nodes that are job-exclusive, job-sharing, or busy.</li> <li>Using <i>up</i> displays all nodes in an 'up state'. Up states include job-exclusive, job-sharing, reserve, free, busy and time-shared.</li> <li>All other strings display the nodes that are currently in the state indicated by the string.</li> </ul>
<b>-m</b>	<p>Set the hosts in the specified host list to the requested power state. If a compute node does not support the energy-saving power state you request, the command returns an error and leaves the state unchanged.</p> <p>In order for the command to wake a node from a low-power state, Wake-on-LAN (WOL) must be enabled for the node.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p> In order for the command to wake a node from a low-power state, Wake-on-LAN must be enabled for the node and it must support the <code>g</code> WOL packet. For more information, see <a href="#">4.4 Changing Node Power States</a>.</p> </div> <p>The allowable power states are:</p>

Option	Description
	<ul style="list-style-type: none"> <li>• <b>Running:</b> The node is up and running.</li> <li>• <b>Standby:</b> CPU is halted but still powered. Moderate power savings but low latency entering and leaving this state.</li> <li>• <b>Suspend:</b> Also known as Suspend-to-RAM. Machine state is saved to RAM. RAM is put into self-refresh mode. Much more significant power savings with longer latency entering and leaving state.</li> <li>• <b>Hibernate:</b> Also known as Suspend-to-disk. Machine state is saved to disk and then powered down. Significant power savings but very long latency entering and leaving state.</li> <li>• <b>Shutdown:</b> Equivalent to <code>shutdown now</code> command as root.</li> </ul> <p>The host list is a space-delimited list of node host names. See <a href="#">A.8.5 Examples</a>.</p>
<b>-n</b>	Show the 'note' attribute for nodes that are DOWN, OFFLINE, or UNKNOWN. This option requires <b>-l</b> .
<b>-N</b>	Specify a 'note' attribute. This enables an admin to add an arbitrary annotation to the listed nodes. To clear a note, use <code>-N ""</code> or <code>-N n</code> .
<b>-o</b>	Add the OFFLINE state. This is different from being marked DOWN. OFFLINE prevents new jobs from running on the specified nodes. This gives the admin 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.
<b>-p</b>	Purge the node record from pbs_server. Not yet implemented.
<b>-q</b>	Suppress all error messages.
<b>-r</b>	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.
<b>-s</b>	Specify the PBS server's hostname or IP address.
<b>-x</b> <b>-xml</b> <b>--xml</b>	Same as <b>-a</b> , but the output has an XML-like format.

## A.8.5 Examples

*Example A-3: host list*

```
pbsnodes -m shutdown node01 node02 node03 node04
```

With this command, `pbs_server` tells the `pbs_mom` associated with nodes01-04 to shut down the node.

The `pbsnodes` output shows the current power state of nodes. In this example, note that `pbsnodes` returns the MAC addresses of the nodes.

```
pbsnodes
nuc1
  state = free
  power_state = Running
  np = 4
  ntype = cluster
  status = retime=1395765676,macaddr=0b:25:22:92:7b:26
, cpuclock=Fixed,varattr=,jobs=,state=free,netload=1242652020,gres=,loadave=0.16,ncpus=
6,physmem=16435852kb,availmem=24709056kb,totmem=33211016kb,idletime=4636,nusers=3,nse
sions=12,sessions=2758 998 1469 2708 2797 2845 2881 2946 4087 4154 4373
6385,uname=Linux bdaw 3.2.0-60-generic #91-Ubuntu SMP Wed Feb 19 03:54:44 UTC 2025
x86_64,opsys=linux
  note = This is a node note
  mom_service_port = 15002
  mom_manager_port = 15003

nuc2
  state = free
  power_state = Running
  np = 4
  ntype = cluster
  status = retime=1395765678,macaddr=2c:a8:6b:f4:b9:35
, cpuclock=OnDemand:800MHz,varattr=,jobs=,state=free,netload=12082362,gres=,loadave=0.0
0,ncpus=4,physmem=16300576kb,availmem=17561808kb,totmem=17861144kb,idletime=67538,nuse
rs=2,nsessions=7,sessions=2189 2193 2194 2220 2222 2248 2351,uname=Linux nuc2 2.6.32-
431.el6.x86_64 #1 SMP Sun Nov 22 03:15:09 UTC 2025 x86_64,opsys=linux
  mom_service_port = 15002
  mom_manager_port = 15003
```

---

## Related Topics

- [pbs\\_server\(8B\)](#)

## Non-Adaptive Computing Topics

- PBS External Reference Specification (included in the Torque download tarball in `doc/v2_2_ers.pdf`)

## A.9 qalter

Alter batch job.

## A.9.1 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] [-
L numa_list]
[-m mail_options] [-M mail_list] [-n] [-N name] [-o path_name]
[-p priority] [-q] [-r y|n] [-S path_name_list] [-t array_range] [-
u user_list]
[-v variable_list] [-W additional_attributes] [-x exec_host]
job_identifier ...
```


## A.9.2 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 that owns each job.


## A.9.3 Options

Option	Name	Description
<b>-a</b>	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.
<b>-A</b>	account_string	Replaces the account string associated with the job. This attribute cannot be altered once the job has begun execution.
<b>-c</b>	checkpoint_interval	Replaces the interval at which the job will be checkpointed. If the job executes upon a host that does not support checkpointing, this option will be ignored. The interval argument is specified as:

Option	Name	Description
		<ul style="list-style-type: none"> <li><i>n</i> – No checkpointing is to be performed.</li> <li><i>s</i> – Checkpointing is to be performed only when the server executing the job is shutdown.</li> <li><i>c</i> – Checkpointing is to be performed at the default minimum cpu time for the queue from which the job is executing.</li> <li><i>c=minutes</i> – 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.</li> </ul> <div>  This value must be greater than zero. If the number is less than the default checkpoint time, the default time will be used.         </div> <p>This attribute can be altered once the job has begun execution, but the new value does not take effect unless the job is rerun.</p>
<b>-e</b>	path_name	<p>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</p> <p>Where <code>hostname</code> is the name of a host to which the file will be returned and <code>path_name</code> is the path name on that host in the syntax recognized by POSIX 1003.1.</p> <p>The argument will be interpreted as follows:</p> <ul style="list-style-type: none"> <li><i>path_name</i> – Where <code>path_name</code> is not an absolute path name, then the <code>qalter</code> 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 <code>hostname</code> component.</li> <li><i>hostname:path_name</i> – Where <code>path_name</code> is not an absolute path name, then the <code>qalter</code> 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 <code>hostname</code>.</li> </ul> <p>This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.</p>
<b>-h</b>	hold_list	<p>Updates the types of holds on the job. The <code>hold_list</code> argument is a string of one or more of the following characters:</p> <ul style="list-style-type: none"> <li><i>u</i> – Add the USER type hold.</li> <li><i>s</i> – Add the SYSTEM type hold if the user has the appropriate level of privilege. (Typically reserved to the batch administrator.)</li> <li><i>o</i> – 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.)</li> </ul>

Option	Name	Description
		<ul style="list-style-type: none"> <li><i>n</i> – Set to none and clear the hold types that could be applied with the user's level of privilege. Repetition of characters is permitted, but 'n' cannot appear in the same option argument with the other three characters.</li> </ul> <p>This attribute can be altered once the job has begun execution, but the hold will not take effect unless the job is rerun.</p>
<b>-j</b>	join	<p>Declares which standard streams of the job will be merged together. The join argument value can be the characters 'oe' or 'eo', or the single character 'n'.</p> <p>An 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. An 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.</p> <p>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.</p> <div>  If using either the <b>-e</b> or the <b>-o</b> option and the <b>-j eo oe</b> option, the <b>-j</b> option takes precedence and all standard error and output messages go to the chosen output file. </div>
<b>-k</b>	keep	<p>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.</p> <p>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:</p> <ul style="list-style-type: none"> <li><i>n</i> – No streams are to be retained.</li> <li><i>e</i> – The standard error 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: <code>job_name.es</code>sequence. Where <code>job_name</code> is the name specified for the job, and <code>sequence</code> is the sequence number component of the job identifier.</li> <li><i>o</i> – 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: <code>job_name.o</code>sequence. Where <code>job_name</code> is the name specified for the job, and <code>sequence</code> is the sequence number component of the job identifier.</li> <li><i>eo</i> – Both the standard output and standard error streams will be retained.</li> </ul>



Option	Name	Description
		<ul style="list-style-type: none"> <li><i>oe</i> – Both the standard output and standard error streams will be retained.</li> </ul> <p>This attribute cannot be altered once the job has begun execution.</p>
<b>-l</b>	resource_list	<p>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]] , ...]</p> <p>For the complete list of resources that can be modified, see <a href="#">3.1.3 Requesting Resources</a>.</p> <p>If a requested modification to a resource would exceed the resource limits for jobs in the current queue, the server will reject the request.</p> <p>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 can only lower the limit for those resources.</p>
<b>-L</b>	NUMA_resource_list	<div>  This uses a different syntax than the <code>-l resource_list</code> option.         </div> <p>Defines the NUMA-aware resource requests for NUMA hardware. This option will work with non-NUMA hardware. See the section <a href="#">12.4 -L NUMA Resource Request</a> for the syntax and valid values.</p>
<b>-m</b>	mail_options	<p>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 that consists of the single character 'n', or one or more of the characters 'a', 'b', and 'e'.</p> <p>If the character 'n' is specified, no mail will be sent.</p> <p>For the letters 'a', 'b', and 'e':</p> <ul style="list-style-type: none"> <li><i>a</i> – Mail is sent when the job is aborted by the batch system.</li> <li><i>b</i> – Mail is sent when the job begins execution.</li> <li><i>e</i> – Mail is sent when the job ends.</li> </ul>
<b>-M</b>	user_list	<p>Replaces the list of users to whom mail is sent by the execution server when it sends mail about the job.</p> <p>The user_list argument is of the form: user [@host] [, user [@host] , ...]</p>
<b>-n</b>	node-exclusive	<p>Informs pbs_server that this job should not share nodes. Note that this needs to be enforced by the scheduler and is not enforced by pbs_server.</p>
<b>-N</b>	name	<p>Renames the job. The name specified can be up to and including 15</p>

Option	Name	Description
		characters in length. It must consist of printable, nonwhite space characters with the first character alphabetic.
<b>-o</b>	path	<p>Replaces the path to be used for the standard output stream of the batch job. The path argument is of the form: <code>[hostname:]path_name</code></p> <p>Where <code>hostname</code> is the name of a host to which the file will be returned and <code>path_name</code> is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:</p> <ul style="list-style-type: none"> <li>• <i>path_name</i> – Where <code>path_name</code> is not an absolute path name, then the <i>qalter</i> 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.</li> <li>• <i>hostname:path_name</i> – Where <code>path_name</code> is not an absolute path name, then the <i>qalter</i> 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.</li> </ul> <p>This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.</p>
<b>-p</b>	priority	<p>Replaces the priority of the job. The priority argument must be an integer between -1024 and +1023 inclusive.</p> <p>This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.</p>
<b>-q</b>	quick	Use an asynchronous (non-blocking) alter call to the server.
<b>-r</b>	[y/n]	<p>Declares whether the job is rerunable (see the <a href="#">qrerun</a> command). The option argument <code>c</code> is a single character. PBS recognizes the following characters: <code>y</code> and <code>n</code>. If the argument is <code>'y'</code>, the job is marked rerunable.</p> <p>If the argument is <code>'n'</code>, the job is marked as not rerunable.</p>
<b>-S</b>	path	<p>Declares the shell that interprets the job script.</p> <p>The option argument <code>path_list</code> is in the form:</p> <pre>path[@host] [, path[@host] , ...]</pre> <p>Only one path can be specified for any host named. Only one path can 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</p>

Option	Name	Description
		<p>(without a host) will be selected.</p> <p>If the <code>-S</code> option is not specified, the option argument is the null string, or no entry from the <code>path_list</code> is selected, the execution will use the login shell of the user on the execution host.</p> <p>This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.</p>
<b>-t</b>	array_range	<p>The <code>array_range</code> argument is an integer ID or a range of integers. Multiple IDs or ID ranges can be combined in a comma-delimited list. Examples: <code>-t 1-100</code> or <code>-t 1,10,50-100</code></p> <p>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.</p> <p>An optional 'slot limit' can be specified to limit the number 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 <code>array_request</code> and is delimited from the array by a percent sign (%).</p> <pre>qalter 15.napali[] -t %20</pre> <p>Here, the array <code>15.napali[]</code> is configured to allow a maximum of 20 concurrently running jobs.</p> <p>Slot limits can be applied at job submit time with <code>qsub</code>, or can be set in a global server parameter policy with <code>max_slot_limit</code>.</p>
<b>-u</b>	user_list	<p>Replaces the user name under which the job is to run on the execution system.</p> <p>The <code>user_list</code> argument is of the form: <code>user[@host][,user[@host],...]</code></p> <p>Only one user name can be given for per specified host. Only one of the user specifications can be supplied without the corresponding host specification. That user name will be used for execution on any host not named in the argument list.</p> <p>This attribute cannot be altered once the job has begun execution.</p>
<b>-v</b>	variable_list	<p>Expands the list of environment variables that are exported to the job.</p> <p><code>variable_list</code> names environment variables from the <code>qsub</code> command environment that are made available to the job when it executes. The <code>variable_list</code> is a comma-separated list of strings of the form <code>variable</code> or <code>variable=value</code>. These variables and their values are passed to the job. See <a href="#">A.23.9 Environment Variables</a> for more information on environment variables.</p>

Option	Name	Description
<b>-W</b>	additional_attributes	<p>The <code>-W</code> 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.</p> <p>To see the attributes PBS currently supports within the <code>-W</code> option, see <a href="#">-W additional_attributes</a> below.</p>
<b>-x</b>	exec_host	Modify the <code>exec_host</code> field of the job.

## **-W additional\_attributes**

The following table lists the attributes PBS currently supports with the `-W` option:

Attribute	Description
<b>depend=dependency_list</b>	<p>Redefines the dependencies between this and other jobs. The <code>dependency_list</code> is in the form: <code>type[:argument[:argument...]][,type:argument...]</code></p> <p>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 is not fully specified in the form: <code>seq_number.server.name</code>, 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).</p> <ul style="list-style-type: none"> <li>• <i>synccount:count</i> – 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.</li> <li>• <i>syncwith:jobid</i> – 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.</li> <li>• <i>after:jobid [:jobid...]</i> – This job can be scheduled for execution at any point after jobs jobid have started execution.</li> <li>• <i>afterok:jobid [:jobid...]</i> – This job can be scheduled for execution only after jobs jobid have terminated with no errors. See the csh warning under 'Extended Description'.</li> <li>• <i>afternotok:jobid [:jobid...]</i> – This job can be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning under 'Extended Description'.</li> <li>• <i>afterany:jobid [:jobid...]</i> – This job can be scheduled for execution after jobs jobid have terminated, with or without</li> </ul>

Attribute	Description
	<p>errors.</p> <ul style="list-style-type: none"> <li>• <i>on:count</i> – This job can 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.</li> <li>• <i>before:jobid [:jobid...]</i> – When this job has begun execution, then jobs jobid... can begin.</li> <li>• <i>beforeok:jobid [:jobid...]</i> – If this job terminates execution without errors, then jobs jobid... can begin. See the csh warning under 'Extended Description'.</li> <li>• <i>beforenotok:jobid [:jobid...]</i> – If this job terminates execution with errors, then jobs jobid... can begin. See the csh warning under 'Extended Description'.</li> <li>• <i>beforeany:jobid [:jobid...]</i> – When this job terminates execution, jobs jobid... can begin.</li> </ul> <p>If any of the before forms are used, the job referenced by jobid must have been submitted with a dependency type of on.</p> <p>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.</p> <p>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.</p>
<b>group_list=g_list</b>	<p>Alters the group name under which the job is to run on the execution system.</p> <p>The g_list argument is of the form: group[@host] [,group[@host] , ...]</p> <p>Only one group name can be given per specified host. Only one of the group specifications can be supplied without the corresponding host specification. That group name will be used for execution on any host not named in the argument list.</p>
<b>stagein=file_list</b> <b>stageout=file_list</b>	<p>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[, ...]</p> <p>The name local_file is the name on the system where the job executes. It can be an absolute path or a path relative to the home directory of the user. The name remote_file is the destination name</p>

Attribute	Description
	on the host specified by hostname. The name can be absolute or relative to the user's home directory on the destination host.

## A.9.4 Operands

The *qalter* command accepts one or more *job\_identifier* operands of the form:  
`sequence_number[.server_name][@server]`

## A.9.5 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.

## A.9.6 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.

## A.9.7 Copyright

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright © 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at [Read The Single UNIX<sup>®</sup> Specification](#).

---

### Related Topics

- [A.11 qdel](#)
- [A.14 qhold](#)
- [A.19 qrls](#)
- [A.23 qsub](#)

## A.10 qchkpt

Checkpoint pbs batch jobs.

### A.10.1 Synopsis

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

### A.10.2 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.

### A.10.3 Options

None.

### A.10.4 Operands

The *qchkpt* command accepts one or more *job\_identifier* operands of the form:

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

### A.10.5 Examples

```
$ # request a checkpoint for job 3233
$ qchkpt 3233
```

### A.10.6 Standard Error

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

### A.10.7 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.

---

## Related Topics

- [qhold\(1B\)](#)
- [qrls\(1B\)](#)
- [qsub\(1B\)](#)
- [qalter\(1B\)](#)

## Non-Adaptive Computing Topics

- [pbs\\_alterjob\(3B\)](#)
- [pbs\\_holdjob\(3B\)](#),
- [pbs\\_rlsjob\(3B\)](#)
- [pbs\\_job\\_attributes\(7B\)](#)
- [pbs\\_resources\\_unicos8\(7B\)](#)

# A.11 qdel

*(delete job)*

## A.11.1 Synopsis

```
qdel [{-a <asynchronous delete>|-b <secs>|-m <message>|-p
<purge>|-t <array_range>|-W <delay>}]
<JOBID>[ <JOBID>]... | 'all' | 'ALL'
```

## A.11.2 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 can 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 admin). This delay can be overridden by the `-W` option.

See the [PBS ERS section 3.1.3.3, Delete Job Request](#) for more information.

### A.11.3 Options

Option	Name	Description
<b>-a</b>	asynchronous delete	Performs an asynchronous delete. The server responds to the user before contacting the MOM. The option <code>qdel -a all</code> performs <code>qdel all</code> due to restrictions from being single-threaded.
<b>-b</b>	seconds	Defines the maximum number of seconds <code>qdel</code> will block attempting to contact <code>pbs_server</code> . If <code>pbs_server</code> is down, or for a variety of communication failures, <code>qdel</code> will continually retry connecting to <code>pbs_server</code> for job submission.  This value overrides the <code>CLIENTRETRY</code> parameter in <code>torque.cfg</code> . This is a non-portable Torque extension. Portability-minded users can use the <code>PBS_CLIENTRETRY</code> environmental variable. A negative value is interpreted as infinity. The default is 0.
<b>-p</b>	purge	Forcibly purges 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.
<b>-t</b>	array_range	The <code>array_range</code> argument is an integer ID or a range of integers. Multiple IDs or ID ranges can be combined in a comma-delimited list (examples: <code>-t 1-100</code> or <code>-t 1,10,50-100</code> ). The command acts on the array (or specified range of the array) just as it would on an individual job.  <div> When deleting a range of jobs, you must include the subscript notation after the job ID (for example, 'qdel -t 1-3 98432[]').</div>
<b>-m</b>	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.

Option	Name	Description
<b>-W</b>	delay	Specifies the wait delay between the sending of the SIGTERM and SIGKILL signals. The argument is the length of time in seconds of the delay.

### A.11.4 Operands

The *qdel* command accepts one or more *job\_identifier* operands of the form: *sequence\_number* [*.server\_name*] [*@server*]

Or

*all*

### A.11.5 Examples

```
# Delete a job array
$ qdel 1234[]

# Delete one job from an array
$ qdel 1234[1]

# Delete all jobs, including job arrays
$ qdel all

# Delete selected jobs from an array
$ qdel -t 2-4,6,8-10 64[]
```

**i** There is not an option that allows you to delete all job arrays without deleting jobs.

### A.11.6 Standard Error

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

### A.11.7 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.

## Related Topics

- [qsub\(1B\)](#)
- [qsig\(1B\)](#)

## Non-Adaptive Computing Topics

- [pbs\\_deljob\(3B\)](#)

## A.12 qgpumode



This command is deprecated; use the `nvidia-smi` utility instead. See [System Management Interface SMI](#) and [NVIDIA System Management Interface program](#) for more information.

(GPU mode)

### A.12.1 Synopsis

```
qgpumode -H host -g gpuid -m mode
```

### A.12.2 Description


The `qgpumode` command specifies the mode for the GPU. This command triggers an immediate update of the `pbs_server`.



For additional information about options for configuring GPUs, see 'NVIDIA GPUs' in the *Moab Workload Manager Administrator Guide*.

### A.12.3 Options

Option	Description
<b>-H</b>	Specifies the host where the GPU is located.
<b>-g</b>	Specifies the ID of the GPU. This varies depending on the version of the NVIDIA driver used. For driver 260.x, it is 0, 1, and so on. For driver 270.x, it is the PCI bus address (i.e., 0:5:0).

Option	Description
<b>-m</b>	<p>Specifies the new mode for the GPU:</p> <ul style="list-style-type: none"> <li>• <b>0 (Default/Shared)</b>: Default/shared compute mode. Multiple threads can use <code>cudaSetDevice()</code> with this device.</li> <li>• <b>1 (Exclusive Thread)</b>: Compute-exclusive-thread mode. Only one thread in one process is able to use <code>cudaSetDevice()</code> with this device.</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p> Support for <code>Exclusive Thread</code> was discontinued with CUDA 8, in favor of <code>Exclusive Process</code>.</p> </div> <ul style="list-style-type: none"> <li>• <b>2 (Prohibited)</b>: Compute-prohibited mode. No threads can use <code>cudaSetDevice()</code> with this device.</li> <li>• <b>3 (Exclusive Process)</b>: Compute-exclusive-process mode. Many threads in one process are able to use <code>cudaSetDevice()</code> with this device.</li> </ul> <div style="border: 1px dashed #ccc; padding: 5px; margin: 10px 0;"> <pre>qgpumode -H node01 -g 0 -m 1</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0; text-align: center;"> <p><i>This puts the first GPU on node01 into mode 1 (exclusive)</i></p> </div> <div style="border: 1px dashed #ccc; padding: 5px; margin: 10px 0;"> <pre>qgpumode -H node01 -g 0 -m 0</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0; text-align: center;"> <p><i>This puts the first GPU on node01 into mode 0 (shared)</i></p> </div>

## Related Topics

- [A.13 qgpureset](#)

## A.13 qgpureset

*(reset GPU)*

### A.13.1 Synopsis

```
qgpureset -H host -g gpuid -p -v
```

### A.13.2 Description

The `qgpureset` command resets the GPU.

### A.13.3 Options

Option	Description
<b>-g</b>	Specifies the ID of the GPU. This varies depending on the version of the NVIDIA driver used. For driver 260.x, it is 0, 1, and so on. For driver 270.x, it is the PCI bus address (i.e., 0:5:0).
<b>-H</b>	Specifies the host where the GPU is located.
<b>-p</b>	Specifies to reset the GPU's permanent ECC error count.
<b>-v</b>	Specifies to reset the GPU's volatile ECC error count.

#### Related Topics

- [A.12 qgpumode](#)

## A.14 qhold

*(hold job)*

### A.14.1 Synopsis

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

### A.14.2 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 can place a USER hold upon any job the user owns. An 'operator', who is a user with 'operator privilege,' can place either an USER or an OTHER hold on any job. The batch administrator can 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 requested hold attribute. This will have no effect unless the job is rerun with the `qrerun` command.

### A.14.3 Options

Option	Name	Description
<b>-h</b>	hold_list	<p>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:</p> <ul style="list-style-type: none"> <li><i>u</i> – USER</li> <li><i>o</i> – OTHER</li> <li><i>s</i> – SYSTEM</li> </ul>
<b>-t</b>	array_range	<p>The array_range argument is an integer ID or a range of integers. Multiple IDs or ID ranges can be combined in a comma-delimited list (examples: <code>-t 1-100</code> or <code>-t 1, 10, 50-100</code>).</p> <p>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.</p>

### A.14.4 Operands

The `qhold` command accepts one or more `job_identifier` operands of the form: `sequence_number[.server_name][@server]`

### A.14.5 Example

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

### A.14.6 Standard Error

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

## A.14.7 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.

---

### Related Topics

- [qrls\(1B\)](#)
- [qalter\(1B\)](#)
- [qsub\(1B\)](#)

### Non-Adaptive Computing Topics

- [pbs\\_alterjob\(3B\)](#)
- [pbs\\_holdjob\(3B\)](#)
- [pbs\\_rlsjob\(3B\)](#)
- [pbs\\_job\\_attributes\(7B\)](#)
- [pbs\\_resources\\_unicos8\(7B\)](#)

## A.15 qmgr

(*PBS Queue Manager*) PBS batch system manager.

### A.15.1 Synopsis

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

### A.15.2 Description

The *qmgr* command provides an administrator interface to query and configure batch system parameters (see [Appendix B: Server 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 requires PBS Manager privilege. Setting or unsetting server or queue attributes requires PBS Operator or Manager privilege.

**i** 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 'Torque/PBS Integration Guide - RM Access Control' in the *Moab Workload Manager Administrator Guide*.

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 can be entered on a single line. A command can 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*.

### A.15.3 Options

Option	Name	Description
<b>-a</b>	---	Abort <i>qmgr</i> on any syntax errors or any requests rejected by a server.
<b>-c</b>	command	Execute a single command and exit <i>qmgr</i> .
<b>-e</b>	---	Echo all commands to standard output.
<b>-n</b>	---	No commands are executed, syntax checking only is performed.
<b>-z</b>	---	No errors are written to standard error.

### A.15.4 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.

## A.15.5 Standard Input

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

## A.15.6 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.

## A.15.7 Standard Error

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

## A.15.8 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 an object.

Commands are:

Command	Description
<b>active</b>	Sets the active objects. If the active objects are specified, and the name is not given in a <i>qmgr</i> cmd the active object names will be used.
<b>create</b>	Creates a new object, applies to queues and nodes.
<b>delete</b>	Destroys an existing object, applies to queues and nodes.

Command	Description
<b>set</b>	Defines or alters attribute values of the object.
<b>unset</b>	Clears the value of attributes of the object. <div>  This form does not accept an OP and value, only the attribute name. </div>
<b>list</b>	Lists the current attributes and associated values of the object.
<b>print</b>	Prints all the queue and server attributes in a format that will be usable as input to the <i>qmgr</i> command.
<b>names</b>	A list of one or more names of specific objects. The name list is in the form: <code>[name] [@server] [,queue_name[@server] ...]</code> 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 affected.
<b>attr</b>	Specifies the name of an attribute of the object that is to be set or modified. If the attribute is one that consists of a set of resources, then the attribute is specified in the form: <code>attribute_name.resource_name</code>
<b>OP</b>	Operation to be performed with the attribute and its value: <ul style="list-style-type: none"> <li>• '=' – set the value of the attribute. If the attribute has an existing value, the current value is replaced with the new value.</li> <li>• '+=' – increase the current value of the attribute by the amount in the new value.</li> <li>• '-=' – decrease the current value of the attribute by the amount in the new value.</li> </ul>
<b>value</b>	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 enclosed 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
```

## A.15.9 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.

---

### Related Topics

- [pbs\\_server](#)(8B)

### Non-Adaptive Computing Topics

- [pbs\\_queue\\_attributes](#) (7B)
- [pbs\\_server\\_attributes](#) (7B)
- [qstart](#) (8B), [qstop](#) (8B)
- [qenable](#) (8B), [qdisable](#) (8)
- PBS External Reference Specification (included in the Torque download tarball in [doc/v2\\_2\\_ers.pdf](#))

## A.16 qmove

Move PBS batch jobs.

### A.16.1 Synopsis

```
qmove destination jobId [jobId ...]
```

### A.16.2 Description

To move a job is to remove the job from the queue in which it resides and instantiate the job in another queue. The *qmove* command issues a Move Job batch request to the batch server that currently owns each job specified by `jobId`.

A job in the `Running`, `Transiting`, or `Exiting` state cannot be moved.

### A.16.3 Operands

The first operand, the new `destination`, is one of the following:

```
queue
@server
queue@server
```

If the `destination` operand describes only a queue, then `qmove` will move jobs into the queue of the specified name at the job's current server. If the `destination` operand describes only a batch server, then `qmove` will move jobs into the default queue at that batch server. If the `destination` operand describes both a queue and a batch server, then `qmove` will move the jobs into the specified queue at the specified server.

All following operands are `jobIds`, which specify the jobs to be moved to the new destination. The `qmove` command accepts one or more `jobId` operands of the form: `sequenceNumber [.serverName] [@server]`

## A.16.4 Standard Error

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

## A.16.5 Exit Status

Upon successful processing of all the operands presented to the `qmove` command, the exit status will be a value of zero. If the `qmove` command fails to process any operand, the command exits with a value greater than zero.

---

### Related Topics

- [A.23 qsub](#)

### Non-Adaptive Computing Topics

- `pbs_movejob(3B)`

## A.17 qorder

Exchange order of two PBS batch jobs in any queue.

### A.17.1 Synopsis

```
qorder job1_identifier job2_identifier
```

## A.17.2 Description

To order two jobs is to exchange the jobs' positions in the queue(s) in which the jobs reside. The two jobs must be located on the same server. No attribute of the job, such as priority, is changed. The impact of changing the order in the queue(s) is dependent on local job schedule policy. For information about your local job schedule policy, contact your system admin.

 A job in the **running** state cannot be reordered.

## A.17.3 Operands

Both operands are `job_identifiers` that specify the jobs to be exchanged. The `qorder` command accepts two `job_identifier` operands of the following form: `sequence_number[.server_name][@server]`

The two jobs must be in the same location, so the server specification for the two jobs must agree.

## A.17.4 Standard Error

The `qorder` command will write diagnostic messages to standard error for each error occurrence.

## A.17.5 Exit Status

Upon successful processing of all the operands presented to the `qorder` command, the exit status will be a value of zero. If the `qorder` command fails to process any operand, the command exits with a value greater than zero.

---

### Related Topics

- [A.23 qsub](#)
- [A.16 qmove](#)

### Non-Adaptive Computing Topics

- `pbs_orderjob(3B)`
- `pbs_movejob(3B)`

## A.18 qrerun

*(Rerun a batch job)*

### A.18.1 Synopsis

```
qrerun [{-f}] <JOBID>[ <JOBID>] ...
```

### A.18.2 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).

### A.18.3 Options

Option	Description
<b>-f</b>	Force a rerun on a job

```
qrerun -f 15406
```

**i** The *qrerun all* command is meant to be run if all of the compute nodes go down. If the machines have actually crashed, then we know that all of the jobs need to be restarted. The behavior if you don't run this would depend on how you bring up the *pbs\_mom* daemons, but by default would be to cancel all of the jobs.

Running the command makes it so that all jobs are requeued without attempting to contact the moms on which they should be running.

### A.18.4 Operands

The `qrerun` command accepts one or more `job_identifier` operands of the form:  
`sequence_number[.server_name][@server]`

## A.18.5 Standard Error

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

## A.18.6 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.

## A.18.7 Example

```
> qrerun 3233
```

(Job 3233 will be re-run.)

---

### Related Topics

- [qsub\(1B\)](#)
- [qalter\(1B\)](#)

### Non-Adaptive Computing Topics

- [pbs\\_alterjob\(3B\)](#)
- [pbs\\_rerunjob\(3B\)](#)

## A.19 qrls

*(Release hold on PBS batch jobs)*

### A.19.1 Synopsis

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

## A.19.2 Description

The *qrls* command removes or releases holds that exist on batch jobs.

A job may have one or more types of holds that 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 can 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.

**i** If you run *qrls* on an array subjob, *pbs\_server* will correct the slot limit holds for the array to which it belongs.

## A.19.3 Options

Command	Name	Description
<b>-h</b>	hold_list	<p>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:</p> <ul style="list-style-type: none"> <li>• <i>u</i> – USER</li> <li>• <i>o</i> – OTHER</li> <li>• <i>s</i> – SYSTEM</li> </ul>
<b>-t</b>	array_range	<p>The array_range argument is an integer ID or a range of integers. Multiple IDs or ID ranges can be combined in a comma-delimited list. Examples: <i>-t 1-100</i> or <i>-t 1,10,50-100</i></p> <p>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.</p>

## A.19.4 Operands

The *qrls* command accepts one or more job\_identifier operands of the form: *sequence\_number* [*.server\_name*] [*@server*]



## A.19.5 Example

```
> qrls -h u 3233 release user hold on job 3233
```

## A.19.6 Standard Error

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

## A.19.7 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.

---

### Related Topics

- [qsub\(1B\)](#)
- [qalter\(1B\)](#)
- [qhold\(1B\)](#)

### Non-Adaptive Computing Topics

- [pbs\\_alterjob\(3B\)](#)
- [pbs\\_holdjob\(3B\)](#)
- [pbs\\_rlsjob\(3B\)](#)

# A.20 qrun

*(Run a batch job)*

## A.20.1 Synopsis

```
qrun [{-H <HOST>|-a}] <JOBID>[ <JOBID>] ...
```

## A.20.2 Overview

The *qrun* command runs a job.

### A.20.3 Format

-a	
Format	---
Default	---
Description	Run the job(s) asynchronously.
Example	<pre>qrun -a 15406</pre>

-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	<pre>qrun -H hostname 15406</pre>

### A.20.4 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.

### A.20.5 Example

```
> qrun 3233
```

(Run job 3233.)

# A.21 qsig

(Signal a job)

## A.21.1 Synopsis

```
qsig [{-s <SIGNAL>}] <JOBID>[ <JOBID>] ...  
[-a]
```

## A.21.2 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 that owns the job.

## A.21.3 Options

Option	Name	Description
-a	asynchronously	Makes the command run asynchronously.
-s	signal	<p>Declares which signal is sent to the job.</p> <p>The signal argument is either a signal name (e.g., SIGKILL), the signal name without the SIG prefix (e.g., KILL), or an 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 <i>qsig</i>. If it doesn't recognize the signal name, try issuing the signal number instead.</p> <p>Two special signal names, 'suspend' and 'resume', are used to suspend and resume jobs.</p> <p>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</p>

Option	Name	Description
		<p>in all tasks on all nodes in the job. Resume sends a SIGCONT to all processes in all tasks on all nodes.</p> <p>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.</p> <div>  Interactive jobs may not resume properly because the top-level shell will background the suspended child process. </div>

## A.21.4 Operands

The *qsig* command accepts one or more *job\_identifier* operands of the form: *sequence\_number* [*.server\_name*] [*@server*]

## A.21.5 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
```

## A.21.6 Standard Error

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

## A.21.7 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.

---

### Related Topics

- [qsub\(1B\)](#)

## Non-Adaptive Computing Topics

- pbs\_sigjob(3B)
- pbs\_resources\_\*(7B) where \* is system type
- PBS ERS

## A.22 qstat

Show status of PBS batch jobs.

### A.22.1 Synopsis

```
qstat [-c] [-C] [-f [-1]] [-W site_specific] [job_
identifier... | destination...] [time]
qstat [-a|-i|-r|-e|--xml] [-c] [-n [-1]] [-s] [-G|-M] [-R] [-u
user_list]
[job_identifier... | destination...]
qstat -Q [-f [-1]] [-c] [-W site_specific] [destination...]
qstat -q [-c] [-G|-M] [destination...]
qstat -B [-c] [-f [-1]] [-W site_specific] [server_name...]
qstat -t [-c] [-C]
```

### A.22.2 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.

**i** You can configure Torque with `CFLAGS='DTXT'` to change the alignment of text in *qstat* output. This noticeably improves *qstat -r* output.

### A.22.3 Options

Option	Description
<b>-a</b>	All jobs are displayed in the alternative format (see <a href="#">A.22.5 Standard Output</a> ). 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.
<b>-B</b>	Specifies that the request is for batch server status and that the operands are the names of servers.
<b>-c</b>	Completed jobs are not displayed in the output. If desired, you can set the <code>PBS_QSTAT_NO_COMPLETE</code> environment variable to cause all <i>qstat</i> requests to not show completed jobs by default.
<b>-C</b>	Specifies that Torque will provide only a condensed output (job name, resources used, queue, state, and job owner) for jobs that have not changed recently. See <a href="#">job_full_report_time</a> . Jobs that have recently changed will continue to send a full output.
<b>-e</b>	If the operand is a job ID or not specified, only jobs in executable queues are displayed. Setting the <code>PBS_QSTAT_EXECONLY</code> environment variable will also enable this option.
<b>-f</b>	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.
<b>-G</b>	Show size information in gigabytes.
<b>-i</b>	Job status is displayed in the alternative format. For a destination ID operand, statuses for jobs at that destination that are not running are displayed. This includes jobs that are queued, held or waiting. If an operand is a job ID, status for that job is displayed regardless of its state.
<b>-1</b>	In combination with <a href="#">-n</a> , the -1 option puts all of the nodes on the same line as the job ID. In combination with <a href="#">-f</a> , attributes are not folded to fit in a terminal window. This is intended to ease the parsing of the <i>qstat</i> output.
<b>-M</b>	Show size information, disk or memory in mega-words. A word is considered to be 8 bytes.
<b>-n</b>	In addition to the basic information, nodes allocated to a job are listed.
<b>-q</b>	Specifies that the request is for queue status, which should be shown in the alternative format.
<b>-Q</b>	Specifies that the request is for queue status and that the operands are destination

Option	Description
	identifiers.
<b>-r</b>	If an operand is a job ID, status for that job is displayed. For a destination ID operand, statuses for jobs at that destination that are running are displayed; this includes jobs that are suspended. Note that if there is no walltime given for a job, then elapsed time does not display.
<b>-R</b>	In addition to other information, disk reservation information is shown. Not applicable to all systems.
<b>-s</b>	In addition to the basic information, any comment provided by the batch administrator or scheduler is shown.
<b>-t</b>	Normal <i>qstat</i> output displays a summary of the array instead of the entire array, job for job. <i>qstat -t</i> 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 <i>qstat -t</i> for the preceding array: 189[299].napali STDIN[299] dbeer 0 Q batch
<b>-u</b>	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, statuses for jobs at that destination that are owned by the user(s) listed in <i>user_list</i> are displayed. The syntax of the <i>user_list</i> is: <i>user_name[@host][,user_name[@host],...]</i> Host names can 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.
<b>--xml</b>	Same as <b>-a</b> , but the output has an XML-like format.

## A.22.4 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 [A.23 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.

## A.22.5 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.
- The CPU time used.
- The job state:

Item	Description
<b>C</b>	Job is completed after having run.
<b>E</b>	Job is exiting after having run.
<b>H</b>	Job is held.



Item	Description
<b>Q</b>	Job is queued, eligible to run or routed.
<b>R</b>	Job is running.
<b>T</b>	Job is being moved to new location.
<b>W</b>	Job is waiting for its execution time ( <b>-a</b> option) to be reached.
<b>S</b>	(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 [A.22.7 Configuration](#) 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 (not the number of nodes in use).
- 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 job's current state.
- The amount of CPU time or wall time used by the job (hh:mm).

When any of the above options or the **-r** option is used to request an alternative display format, a column with the requested memory for the job is displayed. If more than one type of memory is requested for the job, either through server or queue parameters or command line, only one value can be displayed. The value displayed depends on the order

the memory types are evaluated with the last type evaluated being the value displayed. The order of evaluation is `dmem`, `mem`, `pmem`, `pvmem`, `vmem`.

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 job's 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 can 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 [2.3 Advanced 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 can request.
- The maximum amount of CPU time a job in the queue can request.
- The maximum amount of wall time a job in the queue can request.
- The maximum amount of nodes a job in the queue can 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 can 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 can 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 [2.3 Advanced Configuration](#) section for details. If Tcl is not being used, the full display for the server consists of the header line: `Server: server name`

Followed by one line per server attribute of the form: `attribute_name = value`

### A.22.6 Standard Error

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

## A.22.7 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_statqueue(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.

## A.22.8 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.

### Related Topics

- [qalter\(1B\)](#)
- [qsub\(1B\)](#)

### Non-Adaptive Computing Topics

- [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

## A.23 qsub

Submit PBS job.

### A.23.1 Synopsis

```
qsub [-a date_time] [-A account_string] [-b secs] [-c checkpoint_
options] [-C directive_prefix] [-d path] [-D path] [-e path] [-f] [-
F] [-h] [-i idle_slot_limit] [-I] [-j join] [-k keep] [-K kill_
delay] [-l resource_list] [-L NUMA_resource_list] [-m mail_
options] [-M user_list] [-n node_exclusive] [-N name] [-o path] [-p
```

```
priority] [-P user[:group]] [-q destination] [-r] [-S path_to_
shell(s)] [-t array_request] [-T script] [-u userlist] [-v
variable_list] [-V] [-w path] [-W additional_attributes] [-x] [-X]
[-z] [script]
```

## A.23.2 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 [A.23.9 Environment Variables](#)). Typically, the script is a shell script that will be executed by a command shell such as `sh` or `csh`.

Options on the `qsub` command allow the specification of attributes that 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` that has 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:

Variable	Description
<b>PBS_ARRAYID</b>	Each member of a job array is assigned a unique identifier (see <code>-t</code> option).
<b>PBS_ENVIRONMENT</b>	Set to <code>PBS_BATCH</code> to indicate the job is a batch job, or to <code>PBS_INTERACTIVE</code> to indicate the job is a PBS interactive job (see <code>-I</code> option).
<b>PBS_GPUIFILE</b>	The name of the file containing the list of assigned GPUs. For more information about how to set up Torque with GPUs, see 'Accelerators' in the <i>Moab Workload Manager Administrator Guide</i> .
<b>PBS_JOBID</b>	The job identifier assigned to the job by the batch system. It can be used in the stdout and stderr paths. Torque replaces <code>\$PBS_JOBID</code> with the job's jobid (for example, <code>#PBS -o /tmp/\$PBS_JOBID.output</code> ).
<b>PBS_JOBNAME</b>	The job name supplied by the user.


Variable	Description
<b>PBS_NODEFILE</b>	The name of the file contains the list of nodes assigned to the job (for parallel and cluster systems).
<b>PBS_O_HOST</b>	The name of the host upon which the <i>qsub</i> command is running.
<b>PBS_O_QUEUE</b>	The name of the original queue to which the job was submitted.
<b>PBS_O_WORKDIR</b>	The absolute path of the current working directory of the <i>qsub</i> command.
<b>PBS_QUEUE</b>	The name of the queue from which the job is executed.
<b>PBS_SERVER</b>	The hostname of the pbs_server that <i>qsub</i> submits the job to.

### A.23.3 Options


Option	Argument	Description
<b>-a</b>	date_time	<p>Declares the time after which the job is eligible for execution. The date_time argument is in the form: <code>[[ [CC] YY] MM] DD] hhmm [ . SS]</code></p> <p>Where <code>CC</code> is the first two digits of the year (the century), <code>YY</code> is the second two digits of the year, <code>MM</code> is the two digits for the month, <code>DD</code> is the day of the month, <code>hh</code> is the hour, <code>mm</code> is the minute, and the optional <code>SS</code> is the seconds.</p> <p>If the month (<code>MM</code>) is not specified, it will default to the current month if the specified day (<code>DD</code>) is in the future. Otherwise, the month will be set to next month. Likewise, if the day (<code>DD</code>) is not specified, it will default to today if the time (<code>hhmm</code>) is in the future. Otherwise, the day will be set to tomorrow.</p> <p>For example, if you submit a job at 11:15 am with a time of <code>-a 1110</code>, the job will be eligible to run at 11:10 am tomorrow.</p>
<b>-A</b>	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 that executes the job. See section 2.7.1 of the PBS External Reference Specification (included in the Torque download tarball in <code>doc/v2_2_ers.pdf</code> ).
<b>-b</b>	seconds	Defines the maximum number of seconds <i>qsub</i> will block attempting to contact pbs_server. If pbs_server is down, or for a variety of

Option	Argument	Description
		<p>communication failures, <i>qsub</i> will continually retry connecting to <i>pbs_server</i> for job submission.</p> <p>This value overrides the <code>CLIENTRETRY</code> parameter in <code>torque.cfg</code>. This is a non-portable Torque extension. Portability-minded users can use the <code>PBS_CLIENTRETRY</code> environmental variable. A negative value is interpreted as infinity. The default is 0.</p>
<b>-c</b>	checkpoint_options	<p>Defines the options that will apply to the job. If the job executes upon a host that does not support checkpoint, these options will be ignored.</p> <p>checkpoint options are:</p> <ul style="list-style-type: none"> <li>• <i>none</i> – No checkpointing is to be performed.</li> <li>• <i>enabled</i> – Specify that checkpointing is allowed but must be explicitly invoked by either the <a href="#">qhold</a> or <a href="#">qchkpt</a> commands.</li> <li>• <i>shutdown</i> – Specify that checkpointing is to be done on a job at <i>pbs_mom</i> shutdown.</li> <li>• <i>periodic</i> – Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the <code>\$checkpoint_interval</code> option in the MOM config file or by specifying an interval when the job is submitted</li> <li>• <i>interval=minutes</i> – 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.</li> <li>• <i>depth=number</i> – Specify a number (depth) of checkpoint images to be kept in the checkpoint directory.</li> <li>• <i>dir=path</i> – Specify a checkpoint directory (default is <code>/var/spool/torque/checkpoint</code>).</li> </ul>
<b>-C</b>	directive_prefix	<p>Defines the prefix that declares a directive to the <i>qsub</i> command within the script file (see the paragraph on script directives under <a href="#">A.23.11 Extended Description</a>).</p> <p>If the <code>-C</code> option is presented with a <code>directive_prefix</code> argument that is the null string, <i>qsub</i> will not scan the script file for directives.</p>
<b>-d</b>	path	<p>Defines the working directory path to be used for the job. If the <code>-d</code> option is not specified, the default working directory is the home directory. This option sets the environment variable <code>PBS_O_INITDIR</code>.</p>
<b>-D</b>	path	<p>Defines the root directory to be used for the job. This option sets the environment variable <code>PBS_O_ROOTDIR</code>.</p>
<b>-e</b>	path	<p>Defines the path to be used for the standard error stream of the batch</p>




Option	Argument	Description
		<p>job. The path argument is of the form: <code>[hostname:]path_name</code>  Where <code>hostname</code> is the name of a host to which the file will be returned, and <code>path_name</code> is the path name on that host in the syntax recognized by POSIX.</p> <div>  When specifying a directory for the location you need to include a trailing slash. </div> <p>The argument will be interpreted as follows:</p> <ul style="list-style-type: none"> <li>• <code>path_name</code> – where <code>path_name</code> is not an absolute path name, then the <code>qsub</code> 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 <code>hostname</code> component.</li> <li>• <code>hostname:path_name</code> – where <code>path_name</code> is not an absolute path name, then the <code>qsub</code> 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 user's home directory on the <code>hostname</code> system.</li> <li>• <code>path_name</code> – where <code>path_name</code> specifies an absolute path name, then the <code>qsub</code> will supply the name of the host on which it is executing for the <code>hostname</code>.</li> <li>• <code>hostname:path_name</code> – where <code>path_name</code> specifies an absolute path name, the path will be used as specified.</li> </ul> <p>If the <code>-e</code> option is not specified, the default file name for the standard error stream will be used. The default name has the following form:  <code>job_name.esquence_number</code>  Where <code>job_name</code> is the name of the job (see the <code>-N</code> name option) and <code>sequence_number</code> is the job number assigned when the job is submitted.</p>
<b>-f</b>	---	<p>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.</p>

Option	Argument	Description
		<p><b>i</b> If Torque is compiled with <code>PBS_NO_POSIX_VIOLATION</code> (there is no config option for this), you have to use <code>-W fault_tolerant=true</code> to mark the job as fault tolerant.</p>
<b>-F</b>	---	<p>Specifies the arguments that will be passed to the job script when the script is launched. The accepted syntax is: <code>qsub -F "myarg1 myarg2 myarg3=myarg3value" myscript2.sh</code></p> <p><b>i</b> Quotation marks are required. <code>qsub</code> will fail with an error message if the argument following <code>-F</code> is not a quoted value. The <code>pbs_mom</code> server will pass the quoted value as arguments to the job script when it launches the script.</p>
<b>-h</b>	---	Specifies that a user hold be applied to the job at submission time.
<b>-i</b>	idle_slot_limit	<p>Sets an idle slot limit for the job array being submitted. If this parameter is set for a non-array job, it will be rejected. Additionally, if the user requests an idle slot limit that exceeds the server parameter's default, the job will be rejected. See also the <a href="#">idle_slot_limit</a> server parameter.</p> <pre>\$ qsub -t 0-99 -i 10 script.sh</pre> <p><i>The submitted array will only instantiate 10 idle jobs; instead of all 100 jobs at submission time.</i></p>
<b>-I</b>	---	<p>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 <code>qsub</code> to the terminal session in which <code>qsub</code> is running. Interactive jobs are forced to not rerunable. See <a href="#">A.23.11 Extended Description</a> for additional information of interactive jobs.</p>
<b>-j</b>	join	<p>Declares if the standard error stream of the job will be merged with the standard output stream of the job.</p> <p>An option argument value of <code>oe</code> directs that the two streams will be merged, intermixed, as standard output. An option argument value of <code>eo</code> directs that the two streams will be merged, intermixed, as standard error.</p> <p>If the join argument is <code>n</code> or the option is not specified, the two streams will be two separate files.</p>


Option	Argument	Description
		<div>  If using either the <code>-e</code> or the <code>-o</code> option and the <code>-j eo oe</code> option, the <code>-j</code> option takes precedence and all standard error and output messages go to the chosen output file. </div>
<b>-k</b>	keep	<p>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.</p> <p>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'.</p> <ul style="list-style-type: none"> <li><i>e</i> – The standard error 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: <code>job_name.esquence</code>. Where <code>job_name</code> is the name specified for the job, and <code>sequence</code> is the sequence number component of the job identifier.</li> <li><i>o</i> – 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: <code>job_name.osequence</code>. Where <code>job_name</code> is the name specified for the job, and <code>sequence</code> is the sequence number component of the job identifier.</li> <li><i>eo</i> – Both the standard output and standard error streams will be retained.</li> <li><i>oe</i> – Both the standard output and standard error streams will be retained.</li> <li><i>n</i> – Neither stream is retained.</li> </ul>
<b>-K</b>	kill_delay	<p>When set on a job, overrides server and queue kill_delay settings. The kill_delay value is a positive integer. The default is 0. See <a href="#">kill_delay</a> for more information.</p>
<b>-l</b>	resource_list	<p>Defines the resources that are required by the job and establishes a limit to the amount of resources that can be consumed. See <a href="#">3.1.3 Requesting Resources</a> for more information.</p> <p>If not set for a generally available resource, such as CPU time, the limit is infinite. The <code>resource_list</code> argument is of the form: <code>resource_name[=[value]][,resource_name[=[value]],...]</code></p>

Option	Argument	Description
		<p><b>i</b> In this situation, you should request the more inclusive resource first. For example, a request for procs should come before a gres request.</p> <p><code>qsub</code> supports the mapping of <code>-l gpus=X</code> to <code>-l gres=gpus:X</code>. This allows users who are using NUMA systems to make requests such as <code>-l ncpus=20:gpus=5</code> indicating they are not concerned with the GPUs in relation to the NUMA nodes they request, they only want a total of 20 cores and 5 GPUs.</p> <p>If multiple <code>-l</code> options are specified for the same resource, only the last resource list is submitted. For example, with <code>qsub -l nodes=1:ppn=1 -l nodes=1:ppn=2</code>, the request for 1 node and 1 process per node will be ignored, and the request for 1 node and 2 processes per node will be submitted to the server.</p> <p><b>i</b> <code>-l</code> supports some Moab-only extensions. See <a href="#">3.1.3 Requesting Resources</a> for more information on native Torque resources. <code>qsub -W x=</code> is recommended instead (supports more options). See <a href="#">-W</a> for more information.</p> <p>For information on specifying multiple types of resources for allocation, see 'Multi-Req Support' in the <i>Moab Workload Manager Administrator Guide</i>.</p>
<b>-L</b>	NUMA_resource_list	<p><b>i</b> This uses a different syntax than the <code>-l resource_list</code> option.</p> <p>Defines the NUMA-aware resource requests for NUMA hardware. This option will work with non-NUMA hardware. See the section <a href="#">12.4 -L NUMA Resource Request</a> for the syntax and valid values.</p>
<b>-m</b>	mail_options	<p>Defines the set of conditions under which the execution server will send a mail message about the job. The <code>mail_options</code> argument is a string that consists of either the single character 'n' or 'p', or one or more of the characters 'a', 'b', 'e', and 'f'.</p> <p>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.</p> <p>If the character 'p' is specified, mail will never be sent for the job.</p> <p>For the characters 'a', 'b', 'e' and 'f':</p> <ul style="list-style-type: none"> <li><i>a</i> – Mail is sent when the job is aborted by the batch system.</li> <li><i>b</i> – Mail is sent when the job begins execution.</li> <li><i>e</i> – Mail is sent when the job terminates.</li> <li><i>f</i> – Mail is sent when the job terminates with a non-zero exit code.</li> </ul>

Option	Argument	Description
		If the <code>-m</code> option is not specified, mail will be sent if the job is aborted.
<b>-M</b>	user_list	<p>Declares the list of users to whom mail is sent by the execution server when it sends mail about the job.</p> <p>The <code>user_list</code> argument is of the form: <code>user[@host] [, user[@host] , ...]</code></p> <p>If unset, the list defaults to the submitting user at the <code>qsub</code> host (i.e., the job owner).</p>
<b>-n</b>	node_exclusive	<p>Allows a user to specify an exclusive-node access/allocation request for the job. This will set <code>node_exclusive = True</code> in the output of <code>qstat -f &lt;job ID&gt;</code>.</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>i</b> For Moab, the following options are equivalent to <code>'-n'</code>:</p> <pre>&gt; qsub -l naccesspolicy=singlejob jobscript.sh # OR &gt; qsub -W x=naccesspolicy:singlejob jobscript.sh</pre> </div> <p>By default, this only applies for cpusets, and only for compatible schedulers (see <a href="#">4.6 Linux cpuset Support</a>).</p> <p>For systems that use Moab <i>and</i> have cgroups enabled, the recommended manner for assigning all cores is to use NUMA syntax: <code>-L tasks=&lt;count&gt;:lprocs=all:place=node</code>.</p> <p>With cgroups, the <code>(-l)</code> syntax (lowercase L) will, by default, restrict to the number of cores requested, or to the <code>resources_default.procs</code> value (i.e., 1 core, typically). In order to override this behavior and have Moab assign all the cores on a node while using <code>-l...singlejob</code> and/or <code>-n</code> (in other words, without <code>-L...lprocs=all...</code>), you must also set <code>RMCFG[&lt;torque&gt;]FLAGS=MigrateAllJobAttributes</code> in <code>moab.cfg</code>.</p>
<b>-N</b>	name	<p>Declares a name for the job. The name specified can be an unlimited number of characters in length. It must consist of printable, nonwhite space characters with the first character alphabetic.</p> <p>If the <code>-N</code> 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 <code>STDIN</code>.</p>
<b>-o</b>	path	<p>Defines the path to be used for the standard output stream of the batch job. The path argument is of the form: <code>[hostname:]path_name</code></p>

Option	Argument	Description
		<p>Where <code>hostname</code> is the name of a host to which the file will be returned, and <code>path_name</code> is the path name on that host in the syntax recognized by POSIX.</p> <div>  When specifying a directory for the location you need to include a trailing slash. </div> <p>The argument will be interpreted as follows:</p> <ul style="list-style-type: none"> <li>• <i>path_name</i> – where <code>path_name</code> is not an absolute path name, then the <code>qsub</code> 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 <code>hostname</code> component.</li> <li>• <i>hostname:path_name</i> – where <code>path_name</code> is not an absolute path name, then the <code>qsub</code> 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 user's home directory on the <code>hostname</code> system.</li> <li>• <i>path_name</i> – where <code>path_name</code> specifies an absolute path name, then the <code>qsub</code> will supply the name of the host on which it is executing for the <code>hostname</code>.</li> <li>• <i>hostname:path_name</i> where <code>path_name</code> specifies an absolute path name, the path will be used as specified.</li> </ul> <p>If the <code>-o</code> option is not specified, the default file name for the standard output stream will be used. The default name has the following form:  <i>job_name.osequence_number</i></p> <p>Where <code>job_name</code> is the name of the job (see the <code>-N</code> name option) and <code>sequence_number</code> is the job number assigned when the job is submitted.</p>
<b>-p</b>	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.
<b>-P</b>	user [:group]	Allows a root user or manager to submit a job as another user. Torque treats proxy jobs as though the jobs were submitted by the supplied username.
<b>-q</b>	destination	<p>Defines the destination of the job. The destination names a queue, a server, or a queue at a server.</p> <p>The <code>qsub</code> command will submit the script to the server defined by the destination argument. If the destination is a routing queue, the</p>

Option	Argument	Description
		<p>job may be routed by the server to a new destination.</p> <p>If the <code>-q</code> option is not specified, the <code>qsub</code> command will submit the script to the default server. See <a href="#">A.23.9 Environment Variables</a> and the PBS ERS section 2.9.4, Default Server (<a href="#">PBS Pro™ Release 5.2 External Reference Specification</a>).</p> <p>If the <code>-q</code> option is specified, it is in one of the following three forms:</p> <ul style="list-style-type: none"> <li>• <code>queue</code></li> <li>• <code>@server</code></li> <li>• <code>queue@server</code></li> </ul> <p>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.</p> <p>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.</p> <p>If the destination argument names both a queue and a server, the job will be submitted to the named queue at the named server.</p>
<b>-r</b>	y/n	<p>Declares whether the job is rerunnable (see <a href="#">A.18 qrerun</a>). The option argument is a single character, either y or n.</p> <p>If the argument is 'y', the job is rerunnable. If the argument is 'n', the job is not rerunnable. The default value is y, rerunnable.</p>
<b>-S</b>	path_list	<p>Declares the path to the desired shell for this job.</p> <pre>qsub script.sh -S /bin/tcsh</pre> <p>If the shell path is different on different compute nodes, use the following syntax:</p> <pre>path[@host][,path[@host],...]</pre> <pre>qsub script.sh -S</pre> <pre>/bin/tcsh@node1,/usr/bin/tcsh@node2</pre> <p>Only one path can be specified for any host named. Only one path can 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.</p> <p>If the <code>-S</code> option is not specified, the option argument is the null string, or no entry from the <code>path_list</code> is selected, the execution will use the user's login shell on the execution host.</p>
<b>-t</b>	array_	Specifies the task IDs of a job array. Single task arrays are allowed.

Option	Argument	Description
	request	<p>The <code>array_request</code> argument is an integer ID or a range of integers. Multiple IDs or ID ranges can be combined in a comma-delimited list. Examples: <code>-t 1-100</code> or <code>-t 1,10,50-100</code></p> <p>An optional <i>slot limit</i> 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 <code>array_request</code> and is delimited from the array by a percent sign (%):</p> <pre>qsub script.sh -t 0-299%5</pre> <p>This sets the slot limit to 5. Only 5 jobs from this array can run at the same time.</p> <p>You can use <a href="#">qalter</a> to modify slot limits on an array. The server parameter <a href="#">max_slot_limit</a> can be used to set a global slot limit policy.</p>
<b>-T</b>	script	<p>Specifies a prologue or epilogue script for the job. The full name of the scripts are <code>prologue.&lt;script_name&gt;</code> or <code>epilogue.&lt;script_name&gt;</code>, but you only specify the <code>&lt;script_name&gt;</code> portion when using the <code>-T</code> option. For example, <code>qsub -T prescript</code> specifies the <code>prologue.prescript</code> script file.</p>
<b>-u</b>		<div>  <p>This option is deprecated and will not work as previously documented. Use <a href="#">-P</a>.</p> </div>
<b>-v</b>	variable_list	<p>Expands the list of environment variables that are exported to the job.</p> <p>In addition to the variables described in the 'Description' section above, <code>variable_list</code> names environment variables from the <code>qsub</code> command environment that are made available to the job when it executes. The <code>variable_list</code> is a comma-separated list of strings of the form <code>variable</code> or <code>variable=value</code>. These variables and their values are passed to the job. Note that <code>-v</code> has a higher precedence than <code>-V</code>, so identically named variables specified via <code>-v</code> will provide the final value for an environment variable in the job. Example: <code>-v V1,V2=,V3=myval</code> causes environment variables <code>V1</code> and <code>V2</code> to be set within the job to the value taken from the submitting shell's environment. <code>V3</code> is set within the job to the specified value <code>myval</code>. If <code>V1</code> and <code>V2</code> are undefined in the submitting shell's environment, they will be defined within the job, but have no value.</p>
<b>-V</b>	---	<p>Declares that all environment variables in the <code>qsub</code> commands environment are to be exported to the batch job.</p>



Option	Argument	Description
<b>-w</b>	path	Defines the working directory path to be used for the job. If the <b>-w</b> option is not specified, the default working directory is the current directory. This option sets the environment variable PBS_O_WORKDIR.
<b>-W</b>	additional_attributes	<div> <p><b>i</b> Use <b>'-W x='</b> as pass-through for scheduler-only job extensions. See 'Resource Manager Extensions' in the <i>Moab Workload Manager Administrator Guide</i> for a list of scheduler-only job extensions.</p> <p>For legacy purposes, <code>qsub -l</code> will continue to support some scheduler-only job extensions. However, when in doubt, use <b>'-W x='</b>.</p> </div> <p>The <b>-W</b> option allows for the specification of additional job attributes. The general syntax of <b>-W</b> is in the form: <b>-W attr_name=attr_value</b>.</p> <p>You can use multiple <b>-W</b> options with this syntax: <b>-W attr_name1=attr_value1 -W attr_name2=attr_value2</b>.</p> <div> <p><b>i</b> 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.</p> </div> <p>PBS currently supports the following attributes within the <b>-W</b> option:</p> <ul style="list-style-type: none"> <li> <b>depend=dependency_list</b> – Defines the dependency between this and other jobs. The <i>dependency_list</i> is in the form:           <pre>type[:argument[:argument...]] [,type:argument...]</pre> <p>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 <code>seq_number.server.name</code>, it will be expanded according to the default server rules that 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). For more information, see <a href="#">depend=dependency_list Valid Dependencies</a>.</p> </li> <li> <b>group_list=g_list</b> – Defines the group name under which the job is to run on the execution system. The <i>g_list</i> argument is of the form: <code>group[@host][,group[@host],...]</code>. Only one group name can be given per specified host. Only one of the group specifications can be supplied without the corresponding host specification. That group name will be used for execution on any host not named in the argument list. If not           </li> </ul>

Option	Argument	Description
		<p>set, the <code>group_list</code> defaults to the primary group of the user under which the job will be run.</p> <ul style="list-style-type: none"> <li>• <i>interactive=true</i> – If the interactive attribute is specified, the job is an interactive job. The <code>-I</code> option is an alternative method of specifying this attribute.</li> <li>• <i>job_radix=&lt;int&gt;</i> – To be used with parallel jobs. It directs the Mother Superior of the job to create a distribution radix of size <code>&lt;int&gt;</code> between sisters. See <a href="#">3.1.2 Managing Multi-Node Jobs</a>.</li> <li>• <i>stagein=file_list</i> – Specifies which files are staged (copied) in before job start.</li> <li>• <i>stageout=file_list</i> – Specifies which files are staged (copied) 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 <code>file_list</code> is in the form: <code>local_file@hostname:remote_file[,...]</code>, regardless of the direction of the copy.</li> </ul> <p>The name <code>local_file</code> is the name of the file on the system where the job executed. It can be an absolute path or relative to the home directory of the user. The name <code>remote_file</code> is the destination name on the host specified by <code>hostname</code>. The name can 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 (<code>rcp</code>) call on the execution system in the following manner:</p> <ul style="list-style-type: none"> <li>◦ For <code>stagein</code>: <code>rcp hostname:remote_file local_file</code></li> <li>◦ For <code>stageout</code>: <code>rcp local_file hostname:remote_file</code></li> </ul> <p>Data staging examples:</p> <pre>-W stagein=/tmp/input.txt@headnode: /home/user/input.txt -W stageout=/tmp/output.txt@headnode: /home/user/output.txt</pre> <p>If Torque has been compiled with <code>wordexp</code> support, then variables can be used in the specified paths. Currently, only <code>\$PBS_JOBID</code>, <code>\$HOME</code>, and <code>\$TMPDIR</code> are supported for <code>stagein</code>.</p> <ul style="list-style-type: none"> <li>• <i>umask=XXX</i> – Sets <code>umask</code> used to create <code>stdout</code> and <code>stderr</code> spool files in <code>pbs_mom</code> spool directory. Values starting with 0 are treated as octal values; otherwise, the value is treated as a decimal <code>umask</code> value.</li> </ul>



Option	Argument	Description
<b>-x</b>	---	By default, if you submit an interactive job with a script, the script will be parsed for PBS directives but the rest of the script will be ignored since it's an interactive job. The <code>-x</code> option allows the script to be executed in the interactive job and then the job completes. For example:  <pre>script.sh #!/bin/bash ls ---end script---</pre> <pre>qsub -I script.sh qsub: waiting for job 5.napali to start dbeer@napali:~# &lt;displays the contents of the directory, because of the ls command&gt; qsub: job 5.napali completed</pre>
<b>-X</b>	---	Enables X11 forwarding. The <code>DISPLAY</code> environment variable must be set.
<b>-z</b>	---	Directs that the <code>qsub</code> command is not to write the job identifier assigned to the job to the commands standard output.

## depend=dependency\_list Valid Dependencies

**i** For job dependencies to work correctly, you must set the [keep\\_completed](#) server parameter.

Dependency	Description
<code>synccount:count</code>	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.
<code>syncwith:jobid</code>	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, <code>jobid</code> is the job identifier of the first job in the set.
<code>after:jobid[:jobid...]</code>	This job can be scheduled for execution at any point after jobs <code>jobid</code> have started execution.
<code>afterok:jobid[:jobid...]</code>	This job can be scheduled for execution only

Dependency	Description
	after jobs jobid have terminated with no errors. See the csh warning in <a href="#">A.23.11 Extended Description</a> .
<code>afternotok:jobid[:jobid...]</code>	This job can be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning in <a href="#">A.23.11 Extended Description</a> .
<code>afterany:jobid[:jobid...]</code>	This job can be scheduled for execution after jobs jobid have terminated, with or without errors.
<code>on:count</code>	This job can 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).
<code>before:jobid[:jobid...]</code>	When this job has begun execution, then jobs jobid... can begin.
<code>beforeok:jobid[:jobid...]</code>	If this job terminates execution without errors, then jobs jobid... can begin. See the csh warning in <a href="#">A.23.11 Extended Description</a> .
<code>beforenotok:jobid[:jobid...]</code>	If this job terminates execution with errors, then jobs jobid... can begin. See the csh warning in <a href="#">A.23.11 Extended Description</a> .
<code>beforeany:jobid[:jobid...]</code>	When this job terminates execution, jobs jobid... can 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.
<p><b>i</b> 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. For examples, see <a href="#">Dependency Examples</a> below.</p>	

Dependency	Description
<code>afterstartarray:arrayid[count]</code>	After this many jobs have started from arrayid, this job can start.
<code>afterokarray:arrayid[count]</code>	This job can be scheduled for execution only after jobs in arrayid have terminated with no errors.
<code>afternotokarray:arrayid[count]</code>	This job can be scheduled for execution only after jobs in arrayid have terminated with errors.
<code>afteranyarray:arrayid[count]</code>	This job can be scheduled for execution after jobs in arrayid have terminated, with or without errors.
<code>beforestartarray:arrayid[count]</code>	Before this many jobs have started from arrayid, this job can start.
<code>beforeokarray:arrayid[count]</code>	If this job terminates execution without errors, then jobs in arrayid can begin.
<code>beforenotokarray:arrayid[count]</code>	If this job terminates execution with errors, then jobs in arrayid can begin.
<code>beforeanyarray:arrayid[count]</code>	<p>When this job terminates execution, jobs in arrayid can begin.</p> <p>If any of the before forms are used, the jobs referenced by arrayid must have been submitted with a dependency type of on.</p> <p>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.</p>
<div>  Error processing of the existence, state, or condition of the 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. </div> <div>  Jobs can depend on single job dependencies and array dependencies at the same time. </div>	
<code>afterok:jobid</code>	This job can be scheduled for execution only

Dependency	Description
<code>[ :jobid... ],afterokarray:arrayid [count]</code>	after jobs jobid and jobs in arrayid have terminated with no errors.

## 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.

```
qsub script.sh -W depend=afterok:360976,afterokarray:360977[]
```

Job 360976 and all jobs in array 360977 have to successfully finish for the dependency to be satisfied.

## A.23.4 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 that would cause *qsub* to terminate.

## A.23.5 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 '-'.

## A.23.6 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.

## A.23.7 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.

## A.23.8 Standard Error

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

## A.23.9 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 that identifies directives in the script.

The environment variable `PBS_CLIENTRETRY` defines the maximum number of seconds *qsub* will block (see the **-b** option). Despite the name, currently *qsub* is the only client that supports this option.

## A.23.10 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. See [Appendix K: torque.cfg Configuration File](#) for more information on these parameters.

## A.23.11 Extended Description

## Script Processing

A job script can 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 nonwhite 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 nonwhite 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.

## 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 wants to exit. If the user response 'yes', *qsub* exits and the job is aborted.

Once 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:

Sequence	Description
<code>~.</code>	<i>qsub</i> terminates execution. The batch job is also terminated.
<code>~susp</code>	Suspend the <i>qsub</i> program if running under the C shell. 'susp' is the suspend character (usually Cntl-Z).
<code>~asusp</code>	Suspend the input half of <i>qsub</i> (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).

### A.23.12 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.

---

## Related Topics

- [qalter](#)(1B)
- [qdel](#)(1B)
- [qhold](#)(1B)
- [qrls](#)(1B)
- [qsig](#)(1B)
- [qstat](#)(1B)
- [pbs\\_server](#)(8B)

## Non-Adaptive Computing Topics

- [pbs\\_connect](#)(3B)
- [pbs\\_job\\_attributes](#)(7B)
- [pbs\\_queue\\_attributes](#)(7B)
- [pbs\\_resources\\_iris5](#)(7B)
- [pbs\\_resources\\_sp2](#)(7B)
- [pbs\\_resources\\_sunos4](#)(7B)
- [pbs\\_resources\\_unicos8](#)(7B)
- [pbs\\_server\\_attributes](#)(7B)
- [qselect](#)(1B)
- [qmove](#)(1B)
- [qmsg](#)(1B)
- [qrerun](#)(1B)

## A.24 qterm

Terminate processing by a PBS batch server.

### A.24.1 Synopsis


```
qterm [-l] [-t type] [server...]
```

## A.24.2 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 or 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.

## A.24.3 Options

Option	Name	Description
<b>-l</b>	local	Terminate processing only if the active server is local to where <i>qterm</i> is being executed.
<b>-t</b>	type	<p>Specifies the type of shut down:</p> <ul style="list-style-type: none"> <li><i>quick</i> – This is the default action if the <i>-t</i> option is not specified. This option is used when you want running jobs to 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.</li> </ul> <div>  The immediate and delay types are deprecated.         </div>

## A.24.4 Operands

The server operand specifies which servers are to shut down. If no servers are given, then the default server will be terminated.

## A.24.5 Standard Error

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

## A.24.6 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.

### Related Topics (Non-Adaptive Computing Topics)

- *pbs\_server*(8B)
- *qmgr*(8B)
- *pbs\_resources\_aix4*(7B)
- *pbs\_resources\_iris5*(7B)
- *pbs\_resources\_sp2*(7B)
- *pbs\_resources\_sunos4*(7B)
- *pbs\_resources\_unicos8*(7B)

## A.25 trqauthd

(Torque authorization daemon)

### A.25.1 Synopsis

```
trqauthd -d
trqauthd -D
trqauthd -F
trqauthd --logfile_dir
trqauthd -n
```


### A.25.2 Description

The *trqauthd* daemon replaced the *pbs\_iff* authentication process. When users connect to *pbs\_server* by calling one of the Torque utilities or by using the Torque APIs, the new user connection must be authorized by a trusted entity, which runs as root. The advantage of *trqauthd*'s doing this rather than *pbs\_iff* is that *trqauthd* is resident, meaning you do not need to be loaded every time a connection is made; multi-threaded; scalable; and more easily adapted to new functionality than *pbs\_iff*.

*trqauthd* can remember the currently active *pbs\_server* host, enhancing high availability functionality. Previously, *trqauthd* tried to connect to each host in the

TORQUE\_HOME/<server\_name> file until it could successfully connect. Because it now remembers the active server, it tries to connect to that server first. If it fails to connect, it will go through the <server\_name> file and try to connect to a host where an active *pbs\_server* is running.

You have the option when starting *trqauthd* to disable *trqauthd* from logging anything. In addition, the -F (don't fork) option is available when running under *systemd*.

 If you run *trqauthd* before starting *pbs\_server*, you will receive a warning that no servers are available. To avoid this message, start *pbs\_server* before running *trqauthd*.

A.25.3 Options

-d — Terminate	
Format	---
Default	---
Description	Terminate <i>trqauthd</i> .
Example	<div>trqauthd -d</div>

-D — Debug	
Format	---
Default	---
Description	Run <i>trqauthd</i> in debug mode.
Example	<div>trqauthd -D</div>

-F — Fork	
Format	---
Default	---

<b>-F — Fork</b>	
<b>Description</b>	Prevents the system from forking. Useful when running under systemd (Red Hat 7-based or SUSE 12-based systems).
<b>Example</b>	<pre>trqauthd -F</pre>

<b>--logfile_dir — Specify log file directory</b>	
<b>Format</b>	=<path>
<b>Default</b>	---
<b>Description</b>	Specifies custom directory for <i>trqauthd</i> log file.
<b>Example</b>	<pre>trqauthd --logfile_dir=/logs</pre>

<b>-n — No Logging</b>	
<b>Format</b>	---
<b>Default</b>	---
<b>Description</b>	Disables <i>trqauthd</i> from logging anything.
<b>Example</b>	<pre>trqauthd -n</pre>

## 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

<code>acl_group_hosts</code>	<code>acl_host_enable</code>	<code>acl_hosts</code>
<code>acl_logic_or</code>	<code>acl_user_hosts</code>	<code>allow_node_submit</code>
<code>allow_proxy_user</code>	<code>auto_node_np</code>	<code>automatic_requeue_exit_code</code>
<code>cgroup_per_task</code>	<code>checkpoint_defaults</code>	<code>clone_batch_delay</code>
<code>clone_batch_size</code>	<code>copy_on_rerun</code>	<code>default_gpu_mode</code>
<code>default_queue</code>	<code>disable_automatic_requeue</code>	<code>disable_server_id_check</code>
<code>display_job_server_suffix</code>	<code>dont_write_nodes_file</code>	<code>down_on_error</code>
<code>email_batch_seconds</code>	<code>exit_code_canceled_job</code>	<code>ghost_array_recovery</code>
<code>gres_modifiers</code>	<code>idle_slot_limit</code>	<code>interactive_jobs_can_roam</code>
<code>job_exclusive_on_use</code>	<code>job_force_cancel_time</code>	<code>job_full_report_time</code>
<code>job_log_file_max_size</code>	<code>job_log_file_roll_depth</code>	<code>job_log_keep_days</code>
<code>job_nanny</code>	<code>job_start_timeout</code>	<code>job_stat_rate</code>
<code>job_suffix_alias</code>	<code>job_sync_timeout</code>	<code>keep_completed</code>
<code>kill_delay</code>	<code>legacy_vmem</code>	<code>lock_file</code>
<code>lock_file_check_time</code>	<code>lock_file_update_time</code>	<code>log_events</code>
<code>log_file_max_size</code>	<code>log_file_roll_depth</code>	<code>log_keep_days</code>
<code>log_level</code>	<code>mail_body_fmt</code>	<code>mail_domain</code>

mail_from	mail_subject_fmt	managers
max_job_array_size	max_slot_limit	max_threads
max_user_queueable	max_user_run	min_threads
moab_array_compatible	mom_job_sync	next_job_number
node_check_rate	node_pack	node_ping_rate
node_submit_exceptions	no_mail_force	np_default
operators	pass_cpuclock	poll_jobs
query_other_jobs	record_job_info	record_job_script
resources_available	scheduling	sendmail_path
submit_hosts	tcp_incoming_timeout	tcp_timeout
thread_idle_seconds	timeout_for_job_delete	timeout_for_job_requeue
use_jobs_subdirs		

acl_group_hosts	
<b>Format</b>	group@host [.group@host] ...
<b>Default</b>	---
<b>Description</b>	Users who are members of the specified groups will be able to submit jobs from these otherwise untrusted hosts. Users who aren't members of the specified groups will not be able to submit jobs unless they are specified in <code>acl_user_hosts</code> .


acl_host_enable	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE



acl_host_enable	
<b>Description</b>	When set to TRUE, hosts not in the <code>pbs_server</code> nodes file must be added to the <code>acl_hosts</code> list in order to get access to <code>pbs_server</code> .
acl_hosts	
<b>Format</b>	<HOST> [ , <HOST> ] . . . <i>or</i> <HOST> [ range ] <i>or</i> <HOST*> Where the asterisk (*) can appear anywhere in the host name.
<b>Default</b>	Not set.
<b>Description</b>	<p>Specifies a list of hosts that can have access to <code>pbs_server</code> when <code>acl_host_enable</code> is set to TRUE. This does not enable a node to submit jobs. To enable a node to submit jobs use <code>submit_hosts</code>.</p> <div> <p><b>i</b> Hosts that are in the <code>TORQUE_HOME/server_priv/nodes</code> file do not need to be added to this list.</p> <pre>Qmgr: set queue batch acl_hosts="hostA,hostB" Qmgr: set queue batch acl_hosts+=hostC Qmgr: set server acl_hosts="hostA,hostB" Qmgr: set server acl_hosts+=hostC</pre> </div> <div> <p><b>i</b> The wildcard (*) character can appear anywhere in the host name, and ranges are supported; these specifications also work for managers and operators.</p> <pre>Qmgr: set server acl_hosts = "galaxy*.tom.org" Qmgr: set server acl_hosts += "galaxy[0-50].tom.org"</pre> </div>
acl_logic_or	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to TRUE, the user and group queue ACLs are logically ORed. When set to FALSE, they are ANDed.
acl_user_hosts	
<b>Format</b>	group@host [ .group@host ] . . .

acl_user_hosts	
<b>Default</b>	---
<b>Description</b>	The specified users are allowed to submit jobs from otherwise untrusted hosts. By setting this parameter, other users at these hosts will not be allowed to submit jobs unless they are members of specified groups in <code>acl_group_hosts</code> .
allow_node_submit	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , allows all hosts in the <code>PBSHOME/server_priv/nodes</code> file (MOM nodes) to submit jobs to <code>pbs_server</code> . To only allow <code>qsub</code> from a subset of all MOMs, use <code>submit_hosts</code> .
allow_proxy_user	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , specifies that users can proxy from one user to another. Proxy requests will be either validated by <code>ruserok()</code> or by the scheduler.
auto_node_np	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	DISABLED
<b>Description</b>	When set to <code>TRUE</code> , automatically configures a node's <code>np</code> (number of processors) value based on the <code>ncpus</code> value from the status update. Requires full manager privilege to set or alter.
automatic_requeue_exit_code	
<b>Format</b>	<LONG>

automatic_requeue_exit_code	
<b>Default</b>	---
<b>Description</b>	This is an exit code, defined by the admin, that tells <code>pbs_server</code> to requeue the job instead of considering it as completed. This allows the user to add some additional checks that the job can run meaningfully, and if not, then the job script exits with the specified code to be requeued.



cgroup_per_task	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	<p>When set to <code>FALSE</code>, jobs submitted with the <code>-L</code> syntax will have <i>one</i> cgroup created per host unless they specify otherwise at submission time. This behavior is similar to the pre-6.0 <code>cpuset</code> implementation.</p> <p>When set to <code>TRUE</code>, jobs submitted with the <code>-L</code> syntax will have <i>one</i> cgroup created per task unless they specify otherwise at submission time.</p> <div style="border: 1px solid #000; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p> Some MPI implementations are not compatible with using one cgroup per task.</p> </div> <p>See <a href="#">12.4 -L NUMA Resource Request</a> for more information.</p>

checkpoint_defaults	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	<p>Specifies for a queue the default checkpoint values for a job that does not have checkpointing specified. The <code>checkpoint_defaults</code> parameter only takes effect on execution queues.</p> <div style="border: 1px dashed #000; border-radius: 10px; padding: 10px; margin: 10px 0;"> <pre>set queue batch checkpoint_defaults="enabled, periodic, interval=5"</pre> </div>

clone_batch_delay	
<b>Format</b>	<INTEGER>




clone_batch_delay	
<b>Default</b>	1
<b>Description</b>	Specifies the delay (in seconds) between clone batches (see <a href="#">clone_batch_size</a> ).

clone_batch_size	
<b>Format</b>	<INTEGER>
<b>Default</b>	256
<b>Description</b>	Job arrays are created in batches of size <i>X</i> . <i>X</i> jobs are created, and after the <a href="#">clone_batch_delay</a> , <i>X</i> more are created. This repeats until all are created.

copy_on_rerun	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	<p>When set to <code>TRUE</code>, Torque will copy the output and error files over to the user-specified directory when the <code>qrerun</code> command is executed (i.e., a job preemption). Output and error files are only created when a job is in running state before the preemption occurs.</p> <div> <p> <code>pbs_server</code> and <code>pbs_mom</code> need to be on the same version.</p> <p> When you change the value, you must perform a <code>pbs_server</code> restart for the change to effect.</p> </div>

default_gpu_mode	
<b>Format</b>	<STRING>
<b>Default</b>	<code>exclusive_thread</code>
<b>Description</b>	Determines what GPU mode will be used for jobs that request GPUs but do not request a GPU mode. Entries are <code>exclusive_thread</code> , <code>exclusive</code> , <code>exclusive_process</code> , <code>default</code> , and <code>shared</code> .

default_gpu_mode	
	<div> <p><b>i</b> If you are using CUDA 8 or newer, the default of <code>exclusive_thread</code> is no longer supported. If the server specifies an <code>exclusive_thread</code> setting, the MOM will substitute an <code>exclusive_process mode</code> setting. We recommend that you set the default to <code>exclusive_process</code>.</p> <p><b>i</b> If you upgrade your CUDA library, you must rebuild Torque.</p> </div>
default_queue	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	Indicates the queue to assign to a job if no queue is explicitly specified by the submitter.
disable_automatic_requeue	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	Normally, if a job cannot start due to a transient error, the MOM returns a special exit code to the server so that the job is requeued instead of completed. When this parameter is set, the special exit code is ignored and the job is completed.
disable_server_id_check	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , 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.

disable_server_id_check	
	 If you have <code>disable_server_id_check</code> set to <code>TRUE</code> , a user could request a group to which they do not belong. Setting <code>VALIDATEGROUP</code> to <code>TRUE</code> in the <code>torque.cfg</code> file prevents such a scenario (see <a href="#">Appendix K: torque.cfg Configuration File</a> ).
display_job_server_suffix	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	<p>When set to <code>TRUE</code>, Torque will display both the job ID and the host name. When set to <code>FALSE</code>, only the job ID will be displayed.</p> <p> If set to <code>FALSE</code>, the environment variable <code>NO_SERVER_SUFFIX</code> must be set to <code>TRUE</code> for <code>pbs_track</code> to work as expected.</p> <p> <code>display_job_server_suffix</code> should not be set unless the server has no queued jobs. If it is set while the server has queued jobs, it will cause problems correctly identifying job IDs with all existing jobs.</p>
dont_write_nodes_file	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , the nodes file cannot be overwritten for any reason; <code>qmgr</code> commands to edit nodes will be rejected.
down_on_error	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	When set to <code>TRUE</code> , <code>pbs_server</code> will mark nodes that report an error from their node health check as down and unavailable to run jobs. See <a href="#">A.4.5 Health Check</a> for more information.

email_batch_seconds	
<b>Format</b>	<INTEGER>
<b>Default</b>	0
<b>Description</b>	If set to a number greater than 0, emails will be sent in a batch every specified number of seconds, per addressee. For example, if this is set to 300, then each user will only receive emails every 5 minutes in the most frequent scenario. The addressee would then receive one email that contains all of the information that would've been sent out individually before. If it is unset or set to 0, then emails will be sent for every email event.

exit_code_canceled_job	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	When set, the exit code provided by the user is given to any job that is canceled, regardless of the job's state at the time of cancellation.

ghost_array_recovery	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	When <code>TRUE</code> , array subjobs will be recovered regardless of whether the <code>.AR</code> file was correctly recovered. This prevents the loss of running and queued jobs. However, it may <i>no longer</i> enforce a per-job slot limit or handle array dependencies correctly, as some historical information will be lost. When <code>FALSE</code> , array subjobs will not be recovered if the <code>.AR</code> file is invalid or non-existent.

gres_modifiers	
<b>Format</b>	Comma-separated list of user IDs
<b>Default</b>	---
<b>Description</b>	List of users granted permission to modify the gres resource of their own running jobs. Note that users do not need special permission to modify the gres

gres_modifiers	
	resource of their own queued jobs.
idle_slot_limit	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	<p>Sets a default idle slot limit that will be applied to all arrays submitted after it is set.</p> <p>The idle slot limit is the maximum number of subjobs from an array that will be instantiated at once. For example, if this is set to 2, and an array with 1000 subjobs is submitted, then only two will ever be idle (queued) at a time. Whenever an idle subjob runs or is deleted, then a new subjob will be instantiated until the array no longer has remaining subjobs.</p> <p>If this parameter is set, and user during job submission (using <i>qsub -i</i>) requests an idle slot limit that exceeds this setting, that array will be rejected. See also the <i>-i</i> option.</p>
<b>Example</b>	<pre>qmgr -c 'set server idle_slot_limit = 50'</pre>
interactive_jobs_can_roam	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	By default, interactive jobs run from the login node that they submitted from. When <b>TRUE</b> , interactive jobs can run on login nodes other than the one where the jobs were submitted from.
job_exclusive_on_use	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When <b>job_exclusive_on_use</b> is set to <b>TRUE</b> , pbsnodes will show job-exclusive on a node when there's at least one of its processors running a job. This differs with the default behavior, which is to show job-exclusive on a node



job_exclusive_on_use	
	when all of its processors are running a job.
<b>Example</b>	<pre>set server job_exclusive_on_use=TRUE</pre>

job_force_cancel_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	Disabled
<b>Description</b>	If a job has been deleted and is still in the system after <i>x</i> seconds, the job will be purged from the system. This is mostly useful when a job is running on a large number of nodes and one node goes down. The job cannot be deleted because the MOM cannot be contacted. The <i>qdel</i> fails and none of the other nodes can be reused. This parameter can be used to remedy such situations.

job_full_report_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	Sets the time in seconds that a job should be fully reported after any kind of change to the job, even if condensed output was requested.

job_log_file_max_size	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	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 <i>current day</i> 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 <i>pbs_server</i> (meaning the log will not be rolled).


job_log_file_roll_depth	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	This sets the maximum number of new log files that are kept in a day if the <code>job_log_file_max_size</code> parameter is set. For example, if the roll depth is set to 3, no file can roll higher than <code>&lt;filename.3&gt;</code> . If a file is already at the specified depth, such as <code>&lt;filename.3&gt;</code> , the file is deleted so it can be replaced by the incoming file roll, <code>&lt;filename.2&gt;</code> .

job_log_keep_days	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	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	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , 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_start_timeout	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	Specifies the <code>pbs_server</code> to <code>pbs_mom</code> TCP socket timeout in seconds that is used when the <code>pbs_server</code> sends a job start to the <code>pbs_mom</code> . It is useful when the MOM has extra overhead involved in starting jobs. If not specified, then the <code>tcp_timeout</code> parameter is used.

job_stat_rate	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	If the mother superior has not sent an update by the specified time, at the specified time, <code>pbs_server</code> requests an update on job status from the mother superior.

job_suffix_alias	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	<p>Allows the job suffix to be defined by the user.</p> <div>  <code>job_suffix_alias</code> should not be set unless the server has no queued jobs. If it is set while the server has queued jobs, it will cause problems correctly identifying job IDs with all existing jobs. </div>
<b>Example</b>	<pre>qmgr -c 'set server job_suffix_alias = biology'</pre> <p>When a job is submitted after this, its <code>jobid</code> will have <code>.biology</code> on the end: <code>14.napali.biology</code>. If <code>display_job_server_suffix</code> is set to <code>false</code>, it would be named <code>14.biology</code>.</p>

job_sync_timeout	
<b>Format</b>	<INTEGER>
<b>Default</b>	60
<b>Description</b>	When a stray job is reported on multiple nodes, the server sends a kill signal to one node at a time. This timeout determines how long the server waits between kills if the job is still being reported on any nodes.

keep_completed	
<b>Format</b>	<INTEGER>

keep_completed	
<b>Default</b>	300
<b>Description</b>	The amount of time (in seconds) a job will be kept in the queue after it has entered the completed state. <code>keep_completed</code> <i>must</i> be set for job dependencies to work. For more information, see <a href="#">3.5 Keeping Completed Jobs</a> .

kill_delay	
<b>Format</b>	<INTEGER>
<b>Default</b>	If using <code>qdel</code> , 2 seconds If using <code>qrerun</code> , 0 (no wait)
<b>Description</b>	<p>Specifies the number of seconds between sending a SIGTERM and a SIGKILL to a job you want to cancel. It is possible that the job script, and any child processes it spawns, can receive several SIGTERM signals before the SIGKILL signal is received.</p> <div> <p><b>i</b> All MOMs must be configured with <code>\$exec_with_exec</code> true in order for <code>kill_delay</code> to work, even when relying on default <code>kill_delay</code> settings.</p> <p><b>i</b> If <code>kill_delay</code> is set for a queue, the queue setting overrides the server setting. See <code>kill_delay</code> in <a href="#">Appendix N: Queue Attributes</a>.</p> </div>
<b>Example</b>	<pre>qmgr -c "set server kill_delay=30"</pre>

legacy_vmem	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to true, the <code>vmem</code> request will be the amount of memory requested for each node of the job. When it is unset or false, <code>vmem</code> will be the amount of memory for the entire job and will be divided accordingly.

lock_file	
<b>Format</b>	<STRING>
<b>Default</b>	torque/server_priv/server.lock
<b>Description</b>	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 <code>torque/server_priv</code> .

lock_file_check_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	9
<b>Description</b>	Specifies how often (in seconds) a high availability server will check to see if it should become active.

lock_file_update_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	3
<b>Description</b>	Specifies how often (in seconds) the thread will update the lock file.

log_events	
<b>Format</b>	Bitmap
<b>Default</b>	---
<b>Description</b>	By default, the server logs all events. To customize this, perform Boolean OR operations on the binary representation of each of the following bitmaps (or 'enablement bits') to put into effect, then convert the end result to decimal and assign it to <code>log_events</code> :

**log\_events**

```
#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_CLIENTAUTH 0x0200 /* TRQAUTHD login events */
#define PBSEVENT_SYSLOG 0x0400 /* pass this event to the syslog as well (if
defined) */
#define PBSEVENT_FORCE 0x8000 /* set to force a message */
```

For example, if you want to log only internal error, system/server, job-related, and job-usage events, set `log_events` to 27 (1 (0x01) + 2 (0x02) + 8 (0x08) + 16 (0x10)) in `qmgr`:

```
Qmgr: set server log_events = 27
```

**log\_file\_max\_size**

<b>Format</b>	<INTEGER>
<b>Default</b>	0
<b>Description</b>	Specifies a soft limit, in kilobytes, for the server's log file. The file size is checked every 5 minutes, and if the current day file size 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 less than or equal to 0 will be ignored by <code>pbs_server</code> (the log will not be rolled).

**log\_file\_roll\_depth**

<b>Format</b>	<INTEGER>
<b>Default</b>	1
<b>Description</b>	If <code>log_file_max_size</code> is set, controls how deep the current day log files will be rolled before they are deleted.

**log\_keep\_days**

<b>Format</b>	<INTEGER>
---------------	-----------

log_keep_days	
<b>Default</b>	0
<b>Description</b>	Specifies how long (in days) a server or MOM log should be kept.

log_level	
<b>Format</b>	<INTEGER>
<b>Default</b>	0
<b>Description</b>	Specifies the pbs_server logging verbosity. Maximum value is 7.

mail_body_fmt	
<b>Format</b>	A printf-like format string
<b>Default</b>	PBS Job Id: %i Job Name: %j Exec host: %h %m %d
<b>Description</b>	<p>Override the default format for the body of outgoing mail messages. A number of printf-like format specifiers and escape sequences can be used:</p> <ul style="list-style-type: none"> <li>• \n – new line</li> <li>• \t – tab</li> <li>• \\ – backslash</li> <li>• \' – single quote</li> <li>• \" – double quote</li> <li>• %d – details concerning the message</li> <li>• %h – PBS host name</li> <li>• %i – PBS job identifier</li> <li>• %j – PBS job name</li> <li>• %m – long reason for message</li> <li>• %o – job owner</li> <li>• %q – job's queue</li> <li>• %r – short reason for message</li> <li>• %R – resources requested summary</li> <li>• %u – resources used summary</li> <li>• %w – working directory</li> <li>• %% – a single %</li> </ul>

mail_body_fmt	
<b>Example</b>	<pre> %o      job owner          dbeer@nalthis %R      resources requested summary  walltime=600 nodes=2:ppn=6 %u      resources used summary      cput=600 vmem=1043246kb mem=1003241kb %w      working directory          /home/dbeer/hemalurgy/ </pre>
mail_domain	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	Override the default domain for outgoing mail messages. If set, emails will be addressed to <user>@<hostdomain>. If unset, the job's <code>Job_Owner</code> attribute will be used. If set to <code>never</code> , Torque will never send emails.
mail_from	
<b>Format</b>	<STRING>
<b>Default</b>	adm
<b>Description</b>	Specify the name of the sender when Torque sends emails.
mail_subject_fmt	
<b>Format</b>	A printf-like format string
<b>Default</b>	PBS JOB %i
<b>Description</b>	<p>Override the default format for the subject of outgoing mail messages. A number of printf-like format specifiers and escape sequences can be used:</p> <ul style="list-style-type: none"> <li>• <code>\n</code> – new line</li> <li>• <code>\t</code> – tab</li> <li>• <code>\\</code> – backslash</li> <li>• <code>\'</code> – single quote</li> <li>• <code>\"</code> – double quote</li> <li>• <code>%d</code> – details concerning the message</li> <li>• <code>%h</code> – PBS host name</li> <li>• <code>%i</code> – PBS job identifier</li> </ul>



mail_subject_fmt	
	<ul style="list-style-type: none"> <li>• %j – PBS job name</li> <li>• %m – long reason for message</li> <li>• %o – job owner</li> <li>• %q – job's queue</li> <li>• %r – short reason for message</li> <li>• %R – resources requested summary</li> <li>• %u – resources used summary</li> <li>• %w – working directory</li> <li>• %% – a single %</li> </ul>
<b>Example</b>	<pre> %o      job owner          dbeer@nalthis %R      resources requested summary  walltime=600 nodes=2:ppn=6 %u      resources used summary      cput=600 vmem=1043246kb mem=1003241kb %w      working directory          /home/dbeer/hemalurgy/ </pre>

managers	
<b>Format</b>	<user>@<host.sub.domain>[,<user>@<host.sub.domain>...]
<b>Default</b>	root@localhost
<b>Description</b>	List of users granted batch administrator privileges. The host, sub-domain, or domain name can be wildcarded by the use of an asterisk character (*). Requires full manager privilege to set or alter.

max_job_array_size	
<b>Format</b>	<INTEGER>
<b>Default</b>	Unlimited
<b>Description</b>	Sets the maximum number of jobs that can be in a single job array.

max_slot_limit	
<b>Format</b>	<INTEGER>
<b>Default</b>	Unlimited
<b>Description</b>	This is the maximum number of jobs that can run concurrently in any job array.


max_slot_limit	
	<p>Slot limits can be applied at submission time with <code>qsub</code>, or it can be modified with <code>qalter</code>.</p> <pre>qmgr -c 'set server max_slot_limit=10'</pre> <p>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."</p>
max_user_run	
<b>Format</b>	<INTEGER>
<b>Default</b>	Unlimited
<b>Description</b>	This limits the maximum number of jobs a user can have running for the given server.
<b>Example</b>	<pre>qmgr -c "set server max_user_run=5"</pre>
max_threads	
<b>Format</b>	<INTEGER>
<b>Default</b>	<code>min_threads * 20</code>
<b>Description</b>	This is the maximum number of threads that should exist in the thread pool at any time. See <a href="#">5.3 Setting min_threads and max_threads</a> for more information.
max_user_queueable	
<b>Format</b>	<INTEGER>
<b>Default</b>	Unlimited
<b>Description</b>	<p>When set, <code>max_user_queueable</code> places a system-wide limit on the amount of jobs that an individual user can queue.</p> <pre>qmgr -c 'set server max_user_queueable=500'</pre>

min_threads	
<b>Format</b>	<INTEGER>
<b>Default</b>	(2 * the number of procs listed in /proc/cpuinfo) + 1. If Torque is unable to read /proc/cpuinfo, the default is 10.
<b>Description</b>	This is the minimum number of threads that should exist in the thread pool at any time. See <a href="#">5.3 Setting min_threads and max_threads</a> for more information.

moab_array_compatible	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	This parameter places a hold on jobs that exceed the <a href="#">slot limit</a> in a job array. When one of the active jobs is completed or deleted, one of the held jobs goes to a queued state.

mom_job_sync	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	<p>When set to TRUE, specifies that the <i>pbs_server</i> 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, it will be automatically cleaned up and purged. (Enabled by default.)</p> <p>Jobs that are no longer reported by the mother superior are automatically purged by <i>pbs_server</i>. Jobs that <i>pbs_server</i> instructs the MOM to cancel have their processes killed in addition to being deleted.</p>

next_job_number	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	Specifies the ID number of the next job. If you set your job number too low and Torque repeats a job number that it has already used, the job will fail. Before

next_job_number	
	<p>setting next_job_number to a number lower than any number that Torque has already used, you must clear out your .e and .o files.</p> <div>  If you use Moab Workload Manager (and have configured it to synchronize job IDs with Torque), then Moab will generate the job ID and next_job_number will have no effect on the job ID. See 'Resource Manager Configuration' in the <i>Moab Workload Manager Administrator Guide</i> for more information. </div>

node_check_rate	
<b>Format</b>	<INTEGER>
<b>Default</b>	600
<b>Description</b>	Specifies the minimum duration (in seconds) that a node can fail to send a status update before being marked down by the pbs_server daemon.


node_pack	
<b>Description</b>	Deprecated.

node_ping_rate	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	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.

node_submit_exceptions	
<b>Format</b>	String
<b>Default</b>	---
<b>Description</b>	When set in conjunction with allow_node_submit, these nodes will not be allowed to submit jobs.

no_mail_force	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , eliminates all emails when <code>mail_options</code> (see <a href="#">A.23 qsub</a> ) is set to 'n'. The job owner won't receive emails when a job is deleted by a different user or a job failure occurs. If <code>no_mail_force</code> is unset or is <code>FALSE</code> , then the job owner receives emails when a job is deleted by a different user or a job failure occurs.

np_default	
<b>Format</b>	<INTEGER>
<b>Default</b>	---
<b>Description</b>	<p>Allows the admin to unify the number of processors (<code>np</code>) on all nodes. The value can be dynamically changed. A value of 0 tells <code>pbs_server</code> to use the value of <code>np</code> found in the nodes file. The maximum value is 32767.</p> <div>  <code>np_default</code> sets a minimum number of <code>np</code> per node. Nodes with less than the <code>np_default</code> get additional execution slots. </div>

operators	
<b>Format</b>	<user>@<host.sub.domain>[,<user>@<host.sub.domain>...]
<b>Default</b>	root@localhost
<b>Description</b>	List of users granted batch operator privileges. Requires full manager privilege to set or alter.


pass_cpuclock	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	If set to <code>TRUE</code> , the <code>pbs_server</code> daemon passes the option and its value to the <code>pbs_mom</code> daemons for direct implementation by the daemons, making the CPU frequency adjustable as part of a <a href="#">resource request</a> by a job submission.

pass_cpuclock	
	<p>If set to <code>FALSE</code>, the <i>pbs_server</i> daemon creates and passes a <code>PBS_CPUCLOCK</code> job environment variable to the <i>pbs_mom</i> daemons that contains the value of the <code>cpuclock</code> attribute used as part of a resource request by a job submission. The CPU frequencies on the MOMs are not adjusted. The environment variable is for use by prologue and epilogue scripts, enabling admins to log and research when users are making <code>cpuclock</code> requests, as well as researchers and developers to perform CPU clock frequency changes using a method outside of that employed by the Torque <i>pbs_mom</i> daemons.</p>

poll_jobs	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	<p>If set to <code>TRUE</code>, <i>pbs_server</i> will poll job info from MOMs over time and will not block on handling requests that require this job information.</p> <p>If set to <code>FALSE</code>, no polling will occur and if requested job information is stale, <i>pbs_server</i> may block while it attempts to update this information. For large systems, this value should be set to <code>TRUE</code>.</p>

query_other_jobs	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	When set to <code>TRUE</code> , specifies whether or not non-admin users can view jobs they do not own.

record_job_info	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	This must be set to <code>TRUE</code> in order for job logging to be enabled.


record_job_script	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	<p>If set to <code>TRUE</code>, this adds the contents of the script executed by a job to the log.</p> <div>  For <code>record_job_script</code> to take effect, <code>record_job_info</code> must be set to <code>TRUE</code>. </div>

resources_available	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	<p>Allows overriding of detected resource quantities (see <a href="#">N.3 Assigning Queue Resource Limits</a>). <code>pbs_server</code> must be restarted for changes to take effect. Also, <code>resources_available</code> is constrained by the smaller of <code>queue.resources_available</code> and <code>server.resources_available</code>.</p>

scheduling	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	---
<b>Description</b>	<p>Allows <code>pbs_server</code> to be scheduled. When <code>FALSE</code>, <code>pbs_server</code> is a resource manager that works on its own. When <code>TRUE</code>, Torque allows a scheduler, such as Moab or Maui, to dictate what <code>pbs_server</code> should do.</p>

sendmail_path	
<b>Format</b>	<STRING>
<b>Default</b>	<code>/usr/lib/sendmail</code> or the path set with the <code>configure --with-sendmail</code> configure option.
<b>Description</b>	<p>Sets the path to the sendmail executable. If this attribute is set, it will override either the path discovered by Torque during the build or the path explicitly set with the <code>configure --with-sendmail=&lt;path&gt;</code> configure option during the build.</p>

submit_hosts	
<b>Format</b>	<HOSTNAME>[,<HOSTNAME>]...
<b>Default</b>	Not set.
<b>Description</b>	<p>Hosts in this list are able to submit jobs. This applies to any node whether within the cluster or outside of the cluster.</p> <p>If <a href="#">acl_host_enable</a> is set to TRUE and the host is not in the <code>PBSHOME/server_priv/nodes</code> file, then the host must also be in the <a href="#">acl_hosts</a> list.</p> <p>To allow <code>qsub</code> from all compute nodes instead of just a subset of nodes, use <a href="#">allow_node_submit</a>.</p>

tcp_incoming_timeout	
<b>Format</b>	<INTEGER>
<b>Default</b>	600
<b>Description</b>	<p>Specifies the timeout for incoming TCP connections to <code>pbs_server</code>. Functions exactly the same as <a href="#">tcp_timeout</a>, but governs incoming connections while <code>tcp_timeout</code> governs only outgoing connections (or connections initiated by <code>pbs_server</code>).</p> <div> <p> If you use Moab Workload Manager, prevent communication errors by giving <code>tcp_incoming_timeout</code> at least twice the value of the Moab <code>RMPOLLINTERVAL</code>. See <code>RMPOLLINTERVAL</code> in the <i>Moab Workload Manager Administrator Guide</i> for more information.</p> </div>

tcp_timeout	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	<p>Specifies the timeout for idle outbound TCP connections. If no communication is received by the server on the connection after the timeout, the server closes the connection. There is an exception for connections made to the server on port 15001 (default); timeout events are ignored on the server for such connections established by a client utility or scheduler. Responsibility rests with the client to close the connection first (see <a href="#">Appendix F: Large Cluster Considerations</a> for additional information).</p> <p>Use <a href="#">tcp_incoming_timeout</a> to specify the timeout for idle inbound TCP</p>



tcp_timeout	
	connections.

thread_idle_seconds	
<b>Format</b>	<INTEGER>
<b>Default</b>	300
<b>Description</b>	This is the number of seconds a thread can be idle in the thread pool before it is deleted. If threads should not be deleted, set to -1. Torque will always maintain at least <code>min_threads</code> number of threads, even if all are idle.

timeout_for_job_delete	
<b>Format</b>	<INTEGER> (seconds)
<b>Default</b>	120
<b>Description</b>	The specific timeout used when deleting jobs because the node they are executing on is being deleted.

timeout_for_job_requeue	
<b>Format</b>	<INTEGER> (seconds)
<b>Default</b>	120
<b>Description</b>	The specific timeout used when requeuing jobs because the node they are executing on is being deleted.

use_jobs_subdirs	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	Not set (FALSE).
<b>Description</b>	<p>Lets an admin direct the way <code>pbs_server</code> will store its job-related files. Improves the handling of large number of jobs.</p> <ul style="list-style-type: none"> <li>When <code>use_jobs_subdirs</code> is unset (or set to FALSE), job and job array</li> </ul>

**use\_jobs\_subdirs**

files will be stored directly under `$PBS_HOME/server_priv/jobs` and `$PBS_HOME/server_priv/arrays`.

- When `use_jobs_subdirs` is set to `TRUE`, job and job array files will be distributed over 10 subdirectories under their respective parent directories. This method helps to keep a smaller number of files in a given directory.



This setting does not automatically move existing job and job array files into the respective subdirectories. If you choose to use this setting (`TRUE`), you must first:

- set `use_jobs_subdirs` to `TRUE`,
- shut down the Torque server daemon,
- in the `contrib` directory, run the `use_jobs_subdirs_setup` python script with `-m` option,
- start the Torque server daemon.

```
> qmgr -c 'set use_jobs_subdirs=TRUE'
```

## Appendix C: Node Manager (MOM) Configuration

Under Torque, MOM (machine-oriented miniserver) configuration is accomplished using the `mom_priv/config` file located in the PBS directory on each execution server. You must create this file and insert any desired lines in a text editor (blank lines are allowed). When you modify the `mom_priv/config` file, you must restart `pbs_mom`.

The following examples demonstrate two methods of modifying the `mom_priv/config` file:

```
> echo "$loglevel 3" > /var/spool/torque/mom_priv/config
```

```
> vim /var/spool/torque/mom_priv/config
...
$loglevel 3
```

In this appendix:

[C.1 MOM Parameters](#)

[C.2 Node Features and Generic Consumable Resource Specification](#)

### Related Topics

- [Appendix A: Commands Overview](#)
- [Appendix G: Prologue and Epilogue Scripts](#)

## C.1 MOM Parameters


These parameters go in the `mom_priv/config` file. They control various behaviors for the MOMs.

<code>arch</code>	<code>\$attempt_to_make_dir</code>	<code>\$check_poll_time</code>
<code>\$configversion</code>	<code>\$cputmult</code>	<code>\$cuda_visible_devices</code>
<code>\$down_on_error</code>	<code>\$enablemomrestart</code>	<code>\$exec_with_exec</code>
<code>\$ext_pwd_retry</code>	<code>\$force_overwrite</code>	<code>\$ideal_load</code>

\$igncput	\$ignmem	\$ignvmem
\$ignwalltime	\$job_exit_wait_time	\$job_output_file_umask
\$job_starter	\$job_starter_run_privileged	\$jobdirectory_sticky
\$log_directory	\$log_file_max_size	\$log_file_roll_depth
\$log_file_suffix	\$log_keep_days	\$logevent
\$loglevel	\$max_conn_timeout_micro_sec	\$max_join_job_wait_time
\$max_load	\$max_physical_memory	\$max_swap_memory
\$memory_pressure_duration	\$memory_pressure_threshold	\$mom_hierarchy_retry_time
\$mom_host	\$node_check_interval	\$node_check_on_job_end
\$node_check_on_job_start	\$node_check_on_job_start	\$nodefile_suffix
\$nospool_dir_list	opsys	\$pbsclient
\$pbsserver	\$presetup_prologue	\$prologalarm
\$rcpcmd	\$reduce_prolog_checks	\$reject_job_submission
\$remote_checkpoint_dirs	\$remote_reconfig	\$resend_join_job_wait_time
\$restricted	size[fs=<FS>]	\$source_login_batch
\$source_login_interactive	\$spool_as_final_name	\$status_update_time
\$thread_unlink_calls	\$timeout	\$tmpdir
\$use_smt	\$usecp	\$varattr
\$wallmult	\$xauthpath	


arch	
<b>Format</b>	<STRING>



arch	
<b>Description</b>	Specifies the architecture of the local machine. This information is used by the scheduler only.
<b>Example</b>	<pre>arch ia64</pre>

\$attempt_to_make_dir	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	<p>When set to <i>TRUE</i>, specifies that you want Torque to attempt to create the output directories for jobs if they do not already exist.</p> <div>  Torque uses this parameter to make the directory as the <i>user</i> and not as <i>root</i>. Torque will create the directory (or directories) <b>ONLY</b> if the user has permissions to do so. </div>
<b>Example</b>	<pre>\$attempt_to_make_dir true</pre>

\$check_poll_time	
<b>Format</b>	<STRING>
<b>Default</b>	<b>45</b>
<b>Description</b>	Amount of time (in seconds) between checking running jobs, polling jobs, and trying to resend obituaries for jobs that haven't sent successfully.
<b>Example</b>	<pre>\$check_poll_time 90</pre>

\$configversion	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the version of the config file data.
<b>Example</b>	<pre>\$configversion 113</pre>

\$cputmult	
<b>Format</b>	<FLOAT>
<b>Description</b>	<p>CPU time multiplier.</p> <div>  If set to 0.0, MOM level cputime enforcement is disabled. </div>
<b>Example</b>	<pre>\$cputmult 2.2</pre>

\$cuda_visible_devices	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<p><b>TRUE</b></p> <div>  This is disabled by default when cgroups are enabled, because it becomes repetitive at that time. If you still want to have the environment variable with cgroups enabled, then you need to set this parameter to TRUE. </div>
<b>Description</b>	<p>When set to TRUE, the MOM will set the CUDA_VISIBLE_DEVICES environment variable for jobs using NVIDIA GPUs. If set to FALSE, the MOM will not set CUDA_VISIBLE_DEVICES for any jobs.</p> <div>  For CUDA &lt; 7, \$CUDA_VISIBLE_DEVICES is set to the absolute indices of the GPUs your job will use, so if you are using GPUs 2 and 3, then the variable will be set to 2,3. If you are using CUDA &gt;= 7.0, then it will be set to the relative index, starting from 0, so if you are using GPUs 2 and 3, the variable will be set to 0,1. This is necessary because of a change in the CUDA implementation that came out in version 7. </div>
<b>Example</b>	<pre>\$cuda_visible_devices true</pre>

\$down_on_error	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>TRUE</b>
<b>Description</b>	<p>Causes the MOM to report itself as state 'down' to <i>pbs_server</i> in the event of a failed health check. See <a href="#">A.4.5 Health Check</a> for more information.</p>

\$down_on_error	
<b>Example</b>	<code>\$down_on_error true</code>

\$enablemomrestart	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	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; therefore making upgrades easier. The check is made by comparing the mtime of the <i>pbs_mom</i> executable. Command-line args, the process name, and the PATH env variable are preserved across restarts. We recommend that this not be enabled in the config file, but enabled when desired with <i>momctl</i> (see <a href="#">A.3.5 Resources</a> for more information).
<b>Example</b>	<code>\$enablemomrestart true</code>

\$exec_with_exec	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	<i>pbs_mom</i> uses the <i>exec</i> command to start the job script rather than the Torque default method, which is to pass the script's contents as the input to the shell. This means that if you trap signals in the job script, they will be trapped for the job. Using the default method, you would need to configure the shell to also trap the signals.
<b>Example</b>	<code>\$exec_with_exec true</code>

\$ext_pwd_retry	
<b>Format</b>	<INTEGER>
<b>Default</b>	<b>3</b>
<b>Description</b>	Specifies the number of times to retry checking the password. Useful in cases where external password validation is used, such as with LDAP.

\$ext_pwd_retry	
<b>Example</b>	<code>\$ext_pwd_retry = 5</code>

\$force_overwrite	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	When set to true, forces the output files to be overwritten each time a job is started.
<b>Example</b>	<code>\$force_overwrite true</code>

\$ideal_load	
<b>Format</b>	<FLOAT>
<b>Description</b>	Ideal processor load.
<b>Example</b>	<code>\$ideal_load 4.0</code>

\$igncpu	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	Ignores limit violation pertaining to CPU time.
<b>Example</b>	<code>\$igncpu true</code>

\$ignmem	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	Ignores limit violations pertaining to physical memory.



\$ignmem	
<b>Example</b>	<code>\$ignmem true</code>

\$ignvmem	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	Ignore limit violations pertaining to virtual memory.
<b>Example</b>	<code>\$ignvmem true</code>

\$ignwalltime	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	Ignore walltime (do not enable MOM based walltime limit enforcement).
<b>Example</b>	<code>\$ignwalltime true</code>

\$job_exit_wait_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	<b>600</b>
<b>Description</b>	This is the timeout (in seconds) to clean up parallel jobs after one of the sister nodes for the parallel job goes down or is otherwise unresponsive. The MOM sends out all of its kill job requests to sisters and marks the time. Additionally, the job is placed in the substate <code>JOB_SUBSTATE_EXIT_WAIT</code> . The MOM then periodically checks jobs in this state and if they are in this state for more than the specified time, death is assumed and the job gets cleaned up. Default is 600 seconds (10 minutes).
<b>Example</b>	<code>\$job_exit_wait_time 300</code>

\$job_output_file_umask	
<b>Format</b>	<STRING>
<b>Description</b>	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.
<b>Example</b>	<pre>\$job_output_file_umask 027</pre>

\$job_starter	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the fully qualified pathname of the job starter. If this parameter is specified, instead of executing the job command and job arguments directly, the MOM will execute the job starter, passing the job command and job arguments to it as its arguments. The job starter can be used to launch jobs within a desired environment.
<b>Example</b>	<pre>\$job_starter /var/torque/mom_priv/job_starter.sh &gt; cat /var/torque/mom_priv/job_starter.sh #!/bin/bash export FOOHOME=/home/foo ulimit -n 314 \$*</pre>

\$job_starter_run_privileged	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	When set to TRUE, specifies that you want Torque to execute the \$job_starter script with elevated privileges.
<b>Example</b>	<pre>\$job_starter_run_privileged true</pre>

\$jobdirectory_sticky	
<b>Format</b>	<BOOLEAN>

\$jobdirectory_sticky	
<b>Default</b>	<b>FALSE</b>
<b>Description</b>	When set to TRUE, the job directory on the MOM can have a sticky bit set.
<b>Example</b>	<pre>\$jobdirectory_sticky true</pre>

\$log_directory	
<b>Format</b>	<STRING>
<b>Default</b>	<b>TORQUE_HOME/mom_logs/</b>
<b>Description</b>	Changes the log directory. 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.
<b>Example</b>	<pre>\$log_directory /opt/torque/mom_logs/</pre>

\$log_file_max_size	
<b>Format</b>	<INTEGER>
<b>Description</b>	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.
<b>Example</b>	<pre>\$log_file_max_size = 100</pre>


\$log_file_roll_depth	
<b>Format</b>	<INTEGER>
<b>Description</b>	Specifies how many times a log file will be rolled before it is deleted.
<b>Example</b>	<pre>\$log_file_roll_depth = 7</pre>


\$log_file_suffix	
<b>Format</b>	<STRING>
<b>Description</b>	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.
<b>Example</b>	<pre>\$log_file_suffix %h = 20100223.mybox \$log_file_suffix foo = 20100223.foo</pre>

\$log_keep_days	
<b>Format</b>	<INTEGER>
<b>Description</b>	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.
<b>Example</b>	<pre>\$log_keep_days 10</pre>

\$logevent	
<b>Format</b>	<INTEGER>
<b>Description</b>	<p>Creates an event mask enumerating which log events will be recorded in the MOM logs. By default, all events are logged.</p> <p>These are the events that can be chosen:</p> <pre> ERROR      0x0001    internal errors SYSTEM     0x0002    system (server) &amp; (trqauthd) events ADMIN      0x0004    admin events JOB        0x0008    job related events JOB_USAGE  0x0010    End of Job accounting SECURITY   0x0020    security violation events SCHED      0x0040    scheduler events DEBUG      0x0080    common debug messages DEBUG2     0x0100    less needed debug messages CLIENTAUTH 0x0200    TRQAUTHD login events SYSLOG     0x0400    pass this event to the syslog as well </pre> <div>  The listed events are shown here with hexadecimal values; however, a decimal value must be used when setting \$logevent. </div>
<b>Example</b>	<pre>\$logevent 1039</pre>

\$logevent	
	<i>Log ERROR, SYSTEM, ADMIN, JOB and SYSLOG events. This has a hexadecimal value of 0x40F.</i>
\$loglevel	
<b>Format</b>	<INTEGER>
<b>Description</b>	Specifies the verbosity of logging with higher numbers specifying more verbose logging. Values can range between 0 and 7.
<b>Example</b>	<code>\$loglevel 4</code>
\$max_conn_timeout_micro_sec	
<b>Format</b>	<INTEGER>
<b>Default</b>	<b>10000</b>
<b>Description</b>	Specifies how long (in microseconds) <code>pbs_mom</code> should wait for a connection to be made. Default value is 10,000 (.01 sec).
<b>Example</b>	<code>\$max_conn_timeout_micro_sec 30000</code> <i>Sets the connection timeout on the MOM to .03 seconds.</i>
\$max_join_job_wait_time	
<b>Format</b>	<INTEGER>
<b>Default</b>	<b>600</b>
<b>Description</b>	<p>The interval to wait (in seconds) for jobs stuck in a prerun state before deleting them from the MOMs and requeuing them on the server. Default is 600 seconds (10 minutes).</p> <p> If a MOM is completely idle, it can take as long as the next MOM-to-server update time to requeue a failed job.</p>
<b>Example</b>	<code>\$max_join_job_wait_time 300</code>

\$max_load	
<b>Format</b>	<FLOAT>
<b>Description</b>	Maximum processor load.
<b>Example</b>	<code>\$max_load 4.0</code>

\$max_physical_memory	
<b>Format</b>	<INTEGER> <unit>
<b>Description</b>	<p>Restrict the amount of memory available to jobs on this node to the specified amount, which cannot exceed the amount of memory on the machine and must be greater than 0. Default is to use all available memory on the host.</p> <div> <p><b>i</b> When cgroups are enabled, this limits the whole of the machine and doesn't specifically limit each socket or NUMA node. If you have 2 NUMA nodes and 32 GB of memory, but you limit the machine to 30, it won't force a job requesting 16 GB to span NUMA nodes, but once that jobs starts, there would only be 14 GB remaining in use for jobs.</p> <p><b>i</b> If you are using this setting, availmem (as reported in pbsnodes) is no longer accurate, as we do not know what portion of used memory and swap are by jobs and what portion are from the operating system. Since availmem is no longer accurate, you need to set NODEAVAILABILITYPOLICY to DEDICATED if you are using Moab or Maui.</p> </div>
<b>Example</b>	<code>\$max_physical_memory 30gb</code>

\$max_swap_memory	
<b>Format</b>	<INTEGER> <unit>
<b>Description</b>	Restrict the amount of swap available to jobs on this node to the specified amount, which cannot exceed the amount of swap on the machine and must be greater than 0. If you want to disallow swap, this must be set to a very low value instead of 0. Default is to use all available memory on the host.

\$max_swap_memory	
	<p><b>i</b> If you are using this setting, availmem (as reported in pbsnodes) is no longer accurate, as we do not know what portion of used memory and swap are by jobs and what portion are from the operating system. Since availmem is no longer accurate, you need to set NODEAVAILABILITYPOLICY to DEDICATED if you are using Moab or Maui.</p>
<b>Example</b>	<pre>\$max_swap_memory 5gb</pre>
\$memory_pressure_duration	
<b>Format</b>	<INTEGER>
<b>Description</b>	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. This can only be used with <a href="#">\$memory_pressure_threshold</a> .
<b>Example</b>	<pre>\$memory_pressure_duration 5</pre>
\$memory_pressure_threshold	
<b>Format</b>	<INTEGER>
<b>Description</b>	<p>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. Memory_pressure_threshold is only valid with memory_pressure enabled in the root cpuset.</p> <p>To enable, login as the super user and execute the command <code>echo 1 &gt;&gt; /dev/cpuset/memory_pressure_enabled</code>. See the <a href="#">cpuset man page</a> for more information concerning memory pressure.</p>
<b>Example</b>	<pre>\$memory_pressure_threshold 1000</pre>

\$mom_hierarchy_retry_time	
<b>Format</b>	<SECONDS>
<b>Default</b>	<b>90</b>
<b>Description</b>	Specifies the amount of time (in seconds) that a MOM waits to retry a node in the hierarchy path after a failed connection to that node.
<b>Example</b>	<pre>\$mom_hierarchy_retry_time 30</pre>

\$mom_host	
<b>Format</b>	<STRING>
<b>Description</b>	Sets the local hostname as used by <i>pbs_mom</i> .
<b>Example</b>	<pre>\$mom_host node42</pre>

\$node_check_script	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the fully qualified pathname of the health check script to run (see <a href="#">13.10 Compute Node Health Check</a> for more information).
<b>Example</b>	<pre>\$node_check_script /opt/batch_tools/nodecheck.pl</pre>

\$node_check_interval	
<b>Format</b>	<STRING>
<b>Description</b>	<p>Specifies the number of MOM intervals between subsequent executions of the health check specified by <a href="#">\$node_check_script</a>. This value defaults to 1 indicating the check is run every MOM interval (see <a href="#">13.10 Compute Node Health Check</a> for more information). The interval number can be followed by a comma-separated list of events that will initiate a health check.</p> <p><code>\$node_check_interval</code> has two special strings that can be set:</p> <ul style="list-style-type: none"> <li><i>jobstart</i> – makes the node health script run when a job is started (before the prologue script).</li> </ul>




\$node_check_interval	
	<ul style="list-style-type: none"> <li><i>jobend</i> – makes the node health script run after each job has completed on a node (after the epilogue script).</li> </ul> <div> <p><b>i</b> The node health check can be configured to run before or after the job with the 'jobstart' and/or 'jobend' options. However, the job environment variables do not get passed to node health check script, so it has no access to those variables at any time.</p> <p><b>i</b> Using 'jobstart' and/or 'jobend' options is deprecated and may be removed in a future release. To initiate health checks at job start and job end, set the <code>\$node_check_on_job_start</code> and/or <code>\$node_check_on_job_end</code> parameters.</p> </div>
<b>Example</b>	<div> <pre>\$node_check_interval 5,jobstart</pre> </div> <div> <p><i>Execute the health check every 5 MOM intervals and when a job starts.</i></p> </div>

\$node_check_on_job_end	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	If set to true, initiates a health check when a job ends.
<b>Example</b>	<div> <pre>\$node_check_on_job_end=false</pre> </div>

\$node_check_on_job_start	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	If set to true, initiates a health check when a job starts.
<b>Example</b>	<div> <pre>\$node_check_on_job_start=true</pre> </div>

\$nodefile_suffix	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the suffix to append to a host names to denote the data channel

\$nodefile_suffix	
	network adapter in a multi-homed compute node.
<b>Example</b>	<pre>\$nodefile_suffix i</pre> <p><i>With the suffix of "i" and the control channel adapter with the name node01, the data channel would have a hostname of node01i.</i></p>

\$nospool_dir_list	
<b>Format</b>	<STRING>
<b>Description</b>	<p>If this is configured, the job's output is spooled in the working directory of the job or the specified output directory.</p> <p>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.</p> <p>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.</p> <p>Alternatively, you can use the \$spool_as_final_name parameter to force the job to spool directly to the final output.</p> <div>  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. </div>
<b>Example</b>	<pre>\$nospool_dir_list /home/mike/jobs/, /var/tmp/spool/</pre>

opsys	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the operating system of the local machine. This information is used by the scheduler only.
<b>Example</b>	<pre>opsys RHEL3</pre>

\$pbsclient	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies machines that the MOM daemon will trust to run resource manager commands via <a href="#">momctl</a> . This can include machines where monitors, schedulers, or admins require the use of this command.
<b>Example</b>	<pre>\$pbsclient node01.teracluster.org</pre>

\$pbsserver	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the machine running pbs_server.
<b>Example</b>	<pre>\$pbsserver node01.teracluster.org</pre>

\$presetup_prologue	
<b>Format</b>	<STRING>
<b>Description</b>	A full path to the presetup prologue for all jobs on this node. If set, this script executes before any setup for the job occurs (such as becoming the user, creating the output files, or changing directories). As a result, no output from this script will appear in the job's output.
<b>Example</b>	<pre>\$presetup_prologue /opt/kerberos_integration.sh</pre>

\$prologalarm	
<b>Format</b>	<INTEGER>
<b>Description</b>	Specifies maximum duration (in seconds) that the MOM will wait for the job prologue or job epilogue to complete. The default value is 300 seconds (5 minutes). When running parallel jobs, this is also the maximum time a sister node will wait for a job to start.
<b>Example</b>	<pre>\$prologalarm 60</pre>

\$rcpcmd	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the full path and optional additional command line args to use to perform remote copies.
<b>Example</b>	<pre>\$rcpcmd /usr/local/bin/scp -i /etc/sshauth.dat</pre>

\$reduce_prolog_checks	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	If enabled, Torque will only check if the file is a regular file and is executable, instead of the normal checks listed on the prologue and epilogue page. Default is FALSE.
<b>Example</b>	<pre>\$reduce_prolog_checks true</pre>

\$reject_job_submission	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	If set to TRUE, jobs will be rejected and the user will receive the message, "Jobs cannot be run on mom %s." Default is FALSE.
<b>Example</b>	<pre>\$reject_job_submission true</pre>

\$remote_checkpoint_dirs	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies which server checkpoint directories are remotely mounted. It tells the MOM which directories are shared with the server. Using remote checkpoint directories eliminates the need to copy the checkpoint files back and forth between the MOM and the server. All entries must be on the same line, separated by a space.
<b>Example</b>	<pre>\$remote_checkpoint_dirs /checkpointFiles /bigStorage /fast</pre> <p><i>This informs the MOM that the /checkpointFiles,</i></p>

\$remote_checkpoint_dirs	
	<i>/bigStorage, and /fast directories are remotely mounted checkpoint directories.</i>
\$remote_reconfig	
<b>Format</b>	<STRING>
<b>Description</b>	Enables the ability to remotely reconfigure pbs_mom with a new config file. Default is disabled. Enable by setting to true, yes, or 1. For more information on how to reconfigure MOMs, see <a href="#">A.3 momctl-r</a> .
<b>Example</b>	<pre>\$remote_reconfig true</pre>
\$resend_join_job_wait_time	
<b>Format</b>	<INTEGER>
<b>Description</b>	This is the timeout for the Mother Superior to re-send the join job request if it didn't get a reply from all the sister MOMs. The resend happens only once. Default is 5 minutes.
<b>Example</b>	<pre>\$resend_join_job_wait_time 120</pre>
\$restricted	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies hosts that can be trusted to access MOM services as non-root. By default, no hosts are trusted to access MOM services as non-root.
<b>Example</b>	<pre>\$restricted *.teracluster.org</pre>
size[fs=<FS>]	
<b>Format</b>	N/A
<b>Description</b>	Specifies that the available and configured disk space in the <FS> filesystem is

size[fs=<FS>]	
	<p>to be reported to the pbs_server and scheduler.</p> <div> <p><b>i</b> To request disk space on a per job basis, specify the file resource as in <code>qsub -l nodes=1, file=1000kb</code>.</p> <p><b>i</b> Unlike most MOM config options, the <i>size</i> parameter is not preceded by a "\$" character.</p> </div>
<b>Example</b>	<pre>size[fs=/localscratch]</pre> <p><i>The available and configured disk space in the /localscratch filesystem will be reported.</i></p>

\$source_login_batch	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	Specifies whether or not MOM will source environment setup files, such as <code>/etc/profile</code> , for <i>batch</i> jobs. Parameter accepts true, false, yes, no, 1 and 0. Default is TRUE.
<b>Example</b>	<pre>\$source_login_batch False</pre> <p><i>MOM will bypass the sourcing of /etc/profile, etc. type files.</i></p>

\$source_login_interactive	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	Specifies whether or not MOM will source environment setup files, such as <code>/etc/profile</code> , for <i>interactive</i> jobs. Parameter accepts true, false, yes, no, 1 and 0. Default is TRUE.
<b>Example</b>	<pre>\$source_login_interactive False</pre> <p><i>MOM will bypass the sourcing of /etc/profile, etc. type files.</i></p>

\$spool_as_final_name	
<b>Format</b>	<BOOLEAN>

\$spool_as_final_name	
<b>Description</b>	This makes the job write directly to its output destination instead of a spool directory. This allows users easier access to the file if they want to watch the jobs output as it runs.
<b>Example</b>	<pre>\$spool_as_final_name true</pre>

\$status_update_time	
<b>Format</b>	<INTEGER>
<b>Description</b>	Specifies the number of seconds between subsequent MOM-to-server update reports. Default is 45 seconds.
<b>Example</b>	<pre>\$status_update_time 120</pre> <p><i>MOM will send server update reports every 120 seconds.</i></p>

\$thread_unlink_calls	
<b>Format</b>	<BOOLEAN>
<b>Description</b>	Threads calls to unlink when deleting a job. Default is FALSE. If it is set to TRUE, pbs_mom will use a thread to delete the job's files.
<b>Example</b>	<pre>\$thread_unlink_calls true</pre>

\$timeout	
<b>Format</b>	<INTEGER>
<b>Description</b>	Specifies the number of seconds before a TCP connection on the MOM will timeout. Default is 300 seconds.
<b>Example</b>	<pre>\$timeout 120</pre> <p><i>A TCP connection will wait up to 120 seconds before timing out.</i></p>

\$tmpdir	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies a directory to create job-specific scratch space.
<b>Example</b>	<pre>\$tmpdir /localscratch</pre>


\$use_smt	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	<b>TRUE</b>
<b>Description</b>	<p>Indicates that the user would like to use SMT. If set, each logical core inside of a physical core will be used as a normal core for cpusets. This parameter is on by default.</p> <div> <p><b>i</b> \$use_smt is deprecated. Use the syntax to control whether or not threads or cores are used.</p> <p>If you use SMT, you will need to set the <i>np</i> attribute so that each logical processor is counted.</p> </div>
<b>Example</b>	<pre>\$use_smt false</pre>

\$usecp	
<b>Format</b>	<HOST>:<SRCDIR> <DSTDIR>
<b>Description</b>	Specifies which directories should be staged (see <a href="#">7.2 NFS and Other Networked Filesystems</a> ).
<b>Example</b>	<pre>\$usecp *.fte.com:/data /usr/local/data</pre> <p><i>Submission hosts in domain fte.com will map /data directory on submit host to /usr/local/data on compute host</i></p>

\$varattr	
<b>Format</b>	<INTEGER> <STRING>



\$varattr	
<b>Description</b>	<p>Provides a way to keep track of dynamic attributes on nodes.</p> <p>&lt;INTEGER&gt; 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. If set to less than <a href="#">\$status_update_time</a>, the script will run only after the server gets the update. Should preferably be set to a multiple of <a href="#">\$status_update_time</a>.</p> <p>&lt;STRING&gt; is the script path. This script should check for whatever dynamic attributes are desired, and then output lines in this format: name=value</p> <p>Include any arguments after the script's full path. These features are visible in the output of <a href="#">pbsnodes -a</a></p> <pre>varattr=Matlab=7.1;Octave=1.0.</pre> <p>For information about using <code>\$varattr</code> to request dynamic features in Moab, see 'REQATTR' in the <i>Moab Workload Manager Administrator Guide</i>.</p>
<b>Example</b>	<pre>\$varattr 25 /usr/local/scripts/nodeProperties.pl arg1 arg2 arg3</pre>

\$wallmult	
<b>Format</b>	<FLOAT>
<b>Description</b>	<p>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 <code>cputmult</code> is used for cpu time.</p> <div>  If set to 0.0, MOM level walltime enforcement is disabled. </div>
<b>Example</b>	<pre>\$wallmult 2.2</pre>

\$xauthpath	
<b>Format</b>	<STRING>
<b>Description</b>	Specifies the path to the xauth binary to enable X11 forwarding.
<b>Example</b>	<pre>\$xauthpath /opt/bin/xauth</pre>

## C.2 Node Features and Generic Consumable Resource Specification

Node features (a.k.a. 'node properties') are opaque labels that 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 nodes file on the `pbs_server` head node. This file can be used to specify an arbitrary number of node features.

Additionally, per node consumable generic resources can 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 resource information can be routed in by specifying a path preceded by an exclamation point. (!) 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.

`mom_priv/config:`

```
$clienthost 241.13.153.7
tape !/opt/rm/gettapecount.pl
matlab !/opt/tools/getlicensecount.pl
```

## Appendix D: Diagnostics and Error Codes

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 grabs the node file, server and MOM log files, and captures the output of `qmgr -c 'p s'`. These are put in a tar file.

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, '20251130' would be the date for November 30th, 2025. 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.

**Table D-1: Torque error codes**

Error Code Name	Number	Description
PBSE_FLOOR	15000	No error
PBSE_UNKJOBID	15001	Unknown job ID error
PBSE_NOATTR	15002	Undefined attribute
PBSE_ATTRRO	15003	Cannot set attribute, read only or insufficient permission
PBSE_IVALREQ	15004	Invalid request
PBSE_UNKREQ	15005	Unknown request
PBSE_TOOMANY	15006	Too many submit retries
PBSE_PERM	15007	Unauthorized Request
PBSE_IFF_NOT_FOUND	15008	trqauthd unable to authenticate

Error Code Name	Number	Description
PBSE_MUNGE_NOT_FOUND	15009	Munge executable not found, unable to authenticate
PBSE_BADHOST	15010	Access from host not allowed, or unknown host
PBSE_JOBEXIST	15011	Job with requested ID already exists
PBSE_SYSTEM	15012	System error
PBSE_INTERNAL	15013	PBS server internal error
PBSE_REGROUTE	15014	Dependent parent job currently in routing queue
PBSE_UNKSIG	15015	Unknown/illegal signal name
PBSE_BADATVAL	15016	Illegal attribute or resource value for
PBSE_MODATRRUN	15017	Cannot modify attribute while job running
PBSE_BADSTATE	15018	Request invalid for state of job
PBSE_UNKQUE	15020	Unknown queue
PBSE_BADCRED	15021	Invalid credential
PBSE_EXPIRED	15022	Expired credential
PBSE_QUNOENB	15023	Queue is not enabled
PBSE_QACCESS	15024	Access to queue is denied
PBSE_BADUSER	15025	Bad UID for job execution
PBSE_HOPCOUNT	15026	Job routing over too many hops
PBSE_QUEEXIST	15027	Queue already exists
PBSE_ATTRRTYPE	15028	Incompatible type
PBSE_QUEBUSY	15029	Cannot delete busy queue

Error Code Name	Number	Description
PBSE_QUENBIG	15030	Queue name too long
PBSE_NOSUP	15031	No support for requested service
PBSE_QUENOEN	15032	Cannot enable queue, incomplete definition
PBSE_PROTOCOL	15033	Batch protocol error
PBSE_BADATLST	15034	Bad attribute list structure
PBSE_NOCONNECTS	15035	No free connections
PBSE_NOSERVER	15036	No server specified
PBSE_UNKRESC	15037	Unknown resource type
PBSE_EXCQRESC	15038	Job exceeds queue resource limits
PBSE_QUENODFLT	15039	No default queue specified
PBSE_NORERUN	15040	Job is not rerunnable
PBSE_ROUTEREJ	15041	Job rejected by all possible destinations (check syntax, queue resources, ...)
PBSE_ROUTEEXPD	15042	Time in Route Queue Expired
PBSE_MOMREJECT	15043	Execution server rejected request
PBSE_BADSCRIPT	15044	(qsub) cannot access script file
PBSE_STAGEIN	15045	Stage-in of files failed
PBSE_RESCUNAV	15046	Resource temporarily unavailable
PBSE_BADGRP	15047	Bad GID for job execution
PBSE_MAXQUED	15048	Maximum number of jobs already in queue
PBSE_CKPSY	15049	Checkpoint busy, may retry

Error Code Name	Number	Description
PBSE_EXLIMIT	15050	Resource limit exceeds allowable
PBSE_BADACCT	15051	Invalid Account
PBSE_ALRDYEXIT	15052	Job already in exit state
PBSE_NOCOPYFILE	15053	Job files not copied
PBSE_CLEANEOUT	15054	Unknown job ID after clean init
PBSE_NOSYNCMSTR	15055	No master found for sync job set
PBSE_BADDEPEND	15056	Invalid Job Dependency
PBSE_DUPLIST	15057	Duplicate entry in list
PBSE_DISPROTO	15058	Bad DIS based Request Protocol
PBSE_EXECHERE	15059	Cannot execute at specified host because of checkpoint or stagein files
PBSE_SISREJECT	15060	Sister rejected
PBSE_SISCOMM	15061	Sister could not communicate
PBSE_SVRDOWN	15062	Request not allowed: Server shutting down
PBSE_CKPSHORT	15063	Not all tasks could checkpoint
PBSE_UNKNODE	15064	Unknown node
PBSE_UNKNODEATR	15065	Unknown node-attribute
PBSE_NONODES	15066	Server has no node list
PBSE_NODENBIG	15067	Node name is too big
PBSE_NODEEXIST	15068	Node name already exists
PBSE_BADNDATVAL	15069	Illegal value for

Error Code Name	Number	Description
PBSE_MUTUALEX	15070	Mutually exclusive values for
PBSE_GMODERR	15071	Modification failed for
PBSE_NORELYMOM	15072	Server could not connect to MOM
PBSE_NOTSNODE	15073	No time-share node available
PBSE_JOBTYPE	15074	Wrong job type
PBSE_BADACLHOST	15075	Bad ACL entry in host list
PBSE_MAXUSERQUED	15076	Maximum number of jobs already in queue for user
PBSE_BADDISALLOWTYPE	15077	Bad type in disallowed_types list
PBSE_NOINTERACTIVE	15078	Queue does not allow interactive jobs
PBSE_NOBATCH	15079	Queue does not allow batch jobs
PBSE_NORERUNABLE	15080	Queue does not allow rerunable jobs
PBSE_NONONRERUNABLE	15081	Queue does not allow nonrerunable jobs
PBSE_UNKARRAYID	15082	Unknown Array ID
PBSE_BAD_ARRAY_REQ	15083	Bad Job Array Request
PBSE_BAD_ARRAY_DATA	15084	Bad data reading job array from file
PBSE_TIMEOUT	15085	Time out
PBSE_JOBNOTFOUND	15086	Job not found
PBSE_NOFAULTTOLERANT	15087	Queue does not allow fault tolerant jobs
PBSE_NOFAULTINTOLERANT	15088	Queue does not allow fault intolerant jobs

Error Code Name	Number	Description
PBSE_NOJOBARRAYS	15089	Queue does not allow job arrays
PBSE_RELAYED_TO_MOM	15090	Request was relayed to a MOM
PBSE_MEM_MALLOC	15091	Error allocating memory - out of memory
PBSE_MUTEX	15092	Error allocating controlling mutex (lock/unlock)
PBSE_THREADATTR	15093	Error setting thread attributes
PBSE_THREAD	15094	Error creating thread
PBSE_SELECT	15095	Error in socket select
PBSE_SOCKET_FAULT	15096	Unable to get connection to socket
PBSE_SOCKET_WRITE	15097	Error writing data to socket
PBSE_SOCKET_READ	15098	Error reading data from socket
PBSE_SOCKET_CLOSE	15099	Socket close detected
PBSE_SOCKET_LISTEN	15100	Error listening on socket
PBSE_AUTH_INVALID	15101	Invalid auth type in request
PBSE_NOT_IMPLEMENTED	15102	This functionality is not yet implemented
PBSE_QUEENOTAVAILABLE	15103	Queue is currently not available
PBSE_TMPDIFFOWNER	15104	tmpdir owned by another user
PBSE_TMPNOTDIR	15105	tmpdir exists but is not a directory
PBSE_TMPNONAME	15106	tmpdir cannot be named for job
PBSE_CANTOPENSOCKET	15107	Cannot open demux sockets
PBSE_CANTCONTACTSISTERS	15108	Cannot send join job to all sisters



Error Code Name	Number	Description
PBSE_CANTCREATETMPDIR	15109	Cannot create tmpdir for job
PBSE_BADMOMSTATE	15110	Mom is down, cannot run job
PBSE_SOCKET_INFORMATION	15111	Socket information is not accessible
PBSE_SOCKET_DATA	15112	Data on socket does not process correctly
PBSE_CLIENT_INVALID	15113	Client is not allowed/trusted
PBSE_PREMATURE_EOF	15114	Premature End of File
PBSE_CAN_NOT_SAVE_FILE	15115	Error saving file
PBSE_CAN_NOT_OPEN_FILE	15116	Error opening file
PBSE_CAN_NOT_WRITE_FILE	15117	Error writing file
PBSE_JOB_FILE_CORRUPT	15118	Job file corrupt
PBSE_JOB_RERUN	15119	Job cannot be rerun
PBSE_CONNECT	15120	Cannot establish connection
PBSE_JOBWORKDELAY	15121	Job function must be temporarily delayed
PBSE_BAD_PARAMETER	15122	Parameter of function was invalid
PBSE_CONTINUE	15123	Continue processing on job. (Not an error)
PBSE_JOBSTATE	15124	Current sub state does not allow transaction
PBSE_CAN_NOT_MOVE_FILE	15125	Error moving file
PBSE_JOB_RECYCLED	15126	Job is being recycled

Error Code Name	Number	Description
PBSE_JOB_ALREADY_IN_QUEUE	15127	Job is already in destination queue
PBSE_INVALID_MUTEX	15128	Mutex is NULL or otherwise invalid
PBSE_MUTEX_ALREADY_LOCKED	15129	The mutex is already locked by this object
PBSE_MUTEX_ALREADY_UNLOCKED	15130	The mutex has already been unlocked by this object
PBSE_INVALID_SYNTAX	15131	Command syntax invalid
PBSE_NODE_DOWN	15132	A node is down. Check the MOM and host
PBSE_SERVER_NOT_FOUND	15133	Could not connect to batch server
PBSE_SERVER_BUSY	15134	Server busy. Currently no available threads

## Appendix E: Preparing to Upgrade

In this appendix:

[E.1 Considerations Before Upgrading](#)

[E.2 To Upgrade](#)

[E.3 Rolling Upgrade](#)

### E.1 Considerations Before Upgrading

Torque is flexible in regard to how it can be upgraded. In most cases, a Torque 'shutdown' followed by a *configure, make, make install* procedure as documented in this guide is all that is required (see [2.1.3 Installing Torque Resource Manager](#)). This process will preserve existing configuration and in most cases, existing workload.

A few considerations are included below:

- If upgrading from OpenPBS or PBSPro, 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 (see [pbs\\_mom Options](#)), running jobs may be lost. In such cases, running jobs should be allowed to be completed or should be requeued before upgrading Torque.
- The server and the MOMs must run at the same major version, and the `pbs_mom` version should never exceed the `pbs_server` version, even down to the patch level. Problems can arise when running the MOM at a higher version. Most such combinations do not get tested, and unexpected failures and job losses may occur.
- When upgrading from early versions of Moab, you may encounter a problem where Moab core files are regularly created in `/opt/moab`. This can be caused by old Torque library files used by Moab that try to authorize with the old Torque `pbs_iff` authorization daemon. You can resolve the problem by removing the old version library files from `/usr/local/lib`.

### E.2 To Upgrade

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.

The directions to install and configure Torque are in the section [2.1.3 Installing Torque Resource Manager](#). Also note additional instructions in the [PBS Administrator Guide](#) and README.building\_40.

Note that you may need to install libssl-dev in order for the source to make successfully. Specifically, the build system is looking for libssl.so and libcrypto.so. For non-RPM setups, you may need to make a symbolic link from the ssl and crypto libraries to the respective .so names.

## E.3 Rolling Upgrade

If you are upgrading to a new point release of your current version (for example, from 9.1.2 to 9.1.3) and not to a new major release from your current version, you can use the following procedure to upgrade Torque without taking your nodes offline.

### To Perform a Rolling Upgrade in Torque

1. Enable the `pbs_mom` flag on the MOMs you want to upgrade. The `enablemomrestart` option causes a MOM to check if its binary has been updated and restart itself at a safe point when no jobs are running. You can enable this in the MOM configuration file, but it is recommended that you use *momctl* instead.

```
> momctl -q enablemomrestart=1 -h :ALL
```

The `enablemomrestart` flag is enabled on all nodes.

2. Replace the `pbs_mom` binary, located in `/usr/local/bin` by default. `pbs_mom` will continue to run uninterrupted because the `pbs_mom` binary has already been loaded in RAM.

```
> torque-package-mom-linux-x86_64.sh --install
```

The next time `pbs_mom` is in an idle state, it will check for changes in the binary. If `pbs_mom` detects that the binary on disk has changed, it will restart automatically, causing the new `pbs_mom` version to load.

After the `pbs_mom` restarts on each node, the `enablemomrestart` parameter will be set back to false (0) for that node.

**i** If you have a cluster with high utilization, you may find that the nodes never enter an idle state so `pbs_mom` never restarts. When this occurs, you must manually take the nodes offline and wait for the running jobs to complete before restarting `pbs_mom`. To set the node to an offline state, which will allow running jobs to complete but will not allow any new jobs to be scheduled on that node, use `pbsnodes -o <nodeName>`. After the new MOM has started, you must make the node active again by running `pbsnodes -c <nodeName>`.

## Appendix F: Large Cluster Considerations

Torque has enhanced much of the communication found in the original OpenPBS project. This has resulted in a number of key advantages including support for:

- larger clusters.
- more jobs.
- larger jobs.
- 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`, and `tcp_timeout`.

In this appendix:

[F.1 Scalability Guidelines](#)

[F.2 End-User Command Caching](#)

[F.3 Moab and Torque Configuration for Large Clusters](#)

[F.4 Starting Torque in Large Environments](#)

[F.5 Other Considerations](#)

### F.1 Scalability Guidelines

In very large clusters (in excess of 1,000 nodes), it may be advisable to tune a number of communication layer timeouts. By default, PBS MOM daemons timeout on inter-MOM messages after 60 seconds. In Torque, this can be adjusted by setting the timeout parameter in the `mom_priv/config` file (see [Appendix C: Node Manager \(MOM\) Configuration](#)). If 15059 errors (cannot receive message from sisters) are seen in the MOM logs, it may be necessary to increase this value.

Client-to-server communication timeouts are specified via the `tcp_timeout` server option using the `qmgr` command.

Sometimes, ulimits for the `pbs_mom` are inherited from the shell that spawns `pbs_mom`. A workaround is to modify the current shell's `ulimit` or add an entry to the init script that spawns `pbs_mom`.

**i** 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.2 End-User Command Caching

### 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 that 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 ))

#echo AGE=$AGE
```

```

for i in 1 2 3;do
  if [ "$AGE" -gt $CACHETIME ] ; then
    #echo "cache is stale "

    if [ -f $TMPFILE.1 ] ; then
      #echo someone else is updating cache

      sleep 5

      NOW=`date +%s`

      TMPFILEMTIME=`stat -c %Z $TMPFILE`

      AGE=$(( $NOW - $TMPFILEMTIME ))
    else
      break;
    fi
  fi
done

if [ -f $TMPFILE.1 ] ; then
  #echo someone else is hung

  rm $TMPFILE.1
fi

if [ "$AGE" -gt $CACHETIME ] ; then
  #echo updating cache

  $CMDPATH -f > $TMPFILE.1

  mv $TMPFILE.1 $TMPFILE

fi

#echo "using cache"

cat $TMPFILE

exit 0

```

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.3 Moab and Torque Configuration for Large Clusters

There are a few basic configurations for Moab and Torque that can potentially improve performance on large clusters.

### Moab Configuration

In the `moab.cfg` file, add:

1. `RMPOLLINTERVAL 30, 30` - This sets the minimum and maximum poll interval to 30 seconds.
2. `RMCFG[<name>] FLAGS=ASYNCSTART` - This tells Moab not to block until it receives a confirmation that the job starts.
3. `RMCFG[<name>] FLAGS=ASYNCDELETE` - This tells Moab not to block until it receives a confirmation that the job was deleted.

### Torque Configuration

1. Follow [F.4 Starting Torque in Large Environments](#) recommendations.
2. Increase `job_start_timeout` on `pbs_server`. The default is 300 (5 minutes), but for large clusters the value should be changed to something like 600 (10 minutes). Sites running very large parallel jobs might want to set this value even higher.
3. Use a node health check script on all MOM nodes. This helps prevent jobs from being scheduled on bad nodes and is especially helpful for large parallel jobs.
4. Make sure that `ulimit -n` (maximum file descriptors) is set to `unlimited`, or a very large number, and not the default.
5. For clusters with a high job throughput it is recommended that the server parameter `max_threads` be increased from the default. The default is  $(2 * \text{number of cores} + 1) * 10$ .
6. If you have the server send emails, set `email_batch_seconds` appropriately. Setting this parameter will prevent `pbs_server` from forking too frequently and increase the server's performance. See the parameter [email\\_batch\\_seconds](#) for more information on this server parameter.

## F.4 Starting Torque in Large Environments

If running Torque in a large environment, use these tips to help Torque start up faster.

### Fastest Possible Start Up

1. Create a MOM hierarchy, even if your environment has a one-level MOM hierarchy (meaning all MOMs report directly to *pbs\_server*), and copy the file to the *mom\_priv* directory on the MOMs. See [2.3.3 Setting Up the MOM Hierarchy \(Optional\)](#) for more information.
2. Start *pbs\_server* with the *-n* option. This specifies that *pbs\_server* won't send the hierarchy to the MOMs unless a MOM requests it. See *-n* for more information.
3. Start the MOMs normally.

### If no daemons are Running

1. Start *pbs\_server* with the *-c* option.
2. Start *pbs\_mom* without the *-w* option.

### If MOMs are Running and just Restarting *pbs\_server*

1. Start *pbs\_server* without the *-c* option.

### If Restarting a MOM or all MOMs

1. Start *pbs\_mom* without the *-w* option. Starting it with *-w* causes the MOMs to appear to be down.

## F.5 Other Considerations

In this section:

[F.5.1 job\\_stat\\_rate](#)

[F.5.2 poll\\_jobs](#)

[F.5.3 Scheduler Settings](#)

[F.5.4 File System](#)

[F.5.5 Network ARP Cache](#)

### 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 enables 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 Scheduler Settings

If using Moab, there are a number of parameters that can be set on the scheduler, which may improve Torque performance. In an environment containing a large number of short-running jobs, the `JOBAGGREGATIONTIME` parameter (see 'Moab Parameters' in the *Moab Workload Manager Administrator Guide*) can be set to reduce the number of workload and resource queries performed by the scheduler. This parameter enables sites with bursty job submissions to process job events in groups decreasing total job scheduling cycles and allowing the scheduler to make more intelligent choices by aggregating job submissions and choosing between the jobs. If the `pbs_server` daemon is heavily loaded and PBS API timeout errors (i.e., 'Premature end of message') are reported within the scheduler, the `TIMEOUT` attribute of the `RMCFG` parameter can be set with a value of between 30 and 90 seconds.

### F.5.4 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.5.5 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
```

(The exact syntax to set the ARP cache size may vary according to OS version.) Use `sysctl -p` to reload this file.

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 and 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.

In this appendix:

- [G.1 MOM Prologue and Epilogue Scripts](#)
- [G.2 Script Order of Execution](#)
- [G.3 Script Environment](#)
- [G.4 Per Job Prologue and Epilogue Scripts](#)
- [G.5 Prologue and Epilogue Scripts Time Out](#)
- [G.6 Prologue Error Processing](#)

## G.1 MOM Prologue and Epilogue Scripts

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. The 'Mother Superior' is the `pbs_mom` on the first node allocated for a job. While it is technically a sister node, it is not a 'Sister' for the purposes of the following table.

 The initial working directory for each script is `TORQUE_HOME/mom_priv/`.

Script	Execution Location	Script Location	Execute As	File Permissions
<b>Prologue Scripts</b>				
<b>presetup.prologue</b>	Mother Superior	8th argument	root	Readable and executable by root and NOT writable by anyone but root (e.g., <code>-r-x---</code> )

Script	Execution Location	Script Location	Execute As	File Permissions
<b>prologue</b>	Mother Superior	8th argument	root	Readable and executable by root and NOT writable by anyone but root (e.g., <code>-r-x----</code> )
<b>prologue.parallel</b>	Sister	---	root	Readable and executable by root and NOT writable by anyone but root (e.g., <code>-r-x----</code> )
<b>prologue.user</b>	Mother Superior	---	user	Readable and executable by root and other (e.g., <code>-r-x----r-x</code> )
<b>prologue.user.parallel</b>	Sister	---	user	Readable and executable by root and other (e.g., <code>-r-x----r-x</code> )
<b>Epilogue Scripts</b>				
<b>epilogue</b>	Mother Superior	11th argument	root	Readable and executable by root and NOT writable by anyone but root (e.g., <code>-r-x----</code> )
<b>epilogue.parallel</b>	Sister	---	root	Readable and executable by root and NOT writable by anyone but root (e.g., <code>-r-x----</code> )

Script	Execution Location	Script Location	Execute As	File Permissions
<b>epilogue.precancel</b>	Mother Superior This script runs after a job cancel request is received from pbs_server and before a kill signal is sent to the job process.	---	user	Readable and executable by root and other (e.g., <code>-r-x---r-x</code> )
<b>epilogue.user</b>	Mother Superior	---	user	Readable and executable by root and other (e.g., <code>-r-x---r-x</code> )
<b>epilogue.user.parallel</b>	Sister	---	user	Readable and executable by root and other (e.g., <code>-r-x---r-x</code> )

## G.2 Script Order of Execution

When jobs start, the order of script execution is `prologue` followed by `prologue.user`. On job exit, the order of execution is `epilogue.user` followed by `epilogue` unless a job is canceled. In that case, `epilogue.precancel` is executed first. `epilogue.parallel` is executed only on the Sister nodes when the job is completed.

**i** The `epilogue` and `prologue` scripts are controlled by the system administrator. A user `epilogue` and `prologue` script can be used on a per job basis (see [G.4 Per Job Prologue and Epilogue Scripts](#) for more information).

**i** The node health check can be configured to run before or after the job with the 'jobstart' and/or 'jobend' options. However, the job environment variables do not get passed to node health check script, so it has no access to those variables at any time.

**i** Root squashing is now supported for `epilogue` and `prologue` scripts.

## G.3 Script Environment

The `prologue` and `epilogue` scripts can be very simple. On most systems, the script must declare the execution shell using the `#!<SHELL>` syntax (for example, `#!/bin/sh`). In addition, the script may want to process context sensitive arguments passed by Torque to the script.

In this section:

[G.3.1 Prologue Environment](#)

[G.3.2 Epilogue Environment](#)

[G.3.3 Environment Variables](#)

[G.3.4 Standard Input](#)

### G.3.1 Prologue Environment

The following arguments are passed to the `presetup.prologue`, `prologue`, `prologue.user`, and `prologue.parallel` scripts:

Argument	Description
<b>argv[1]</b>	Job ID.
<b>argv[2]</b>	Job execution user name.
<b>argv[3]</b>	Job execution group name.
<b>argv[4]</b>	Job name.
<b>argv[5]</b>	List of requested resource.
<b>argv[6]</b>	Job execution queue.
<b>argv[7]</b>	Job account.
<b>argv[8]</b>	Job script location.
<b>argv[9]</b>	Comma-separated list of each host in the job. For example, if a job is using 10 cores on each of <code>roshar</code> , <code>nalthis</code> , <code>elantris</code> , and <code>scadrial</code> , this argument



Argument	Description
	will have the value: <code>roshar,nalthis,elantris,scadrial</code> . Defined only for <code>presetup.prologue</code> .

## G.3.2 Epilogue Environment

Torque supplies the following arguments to the `epilogue`, `epilogue.user`, `epilogue.precancel`, and `epilogue.parallel` scripts:

Argument	Description
<b>argv[1]</b>	job id
<b>argv[2]</b>	job execution user name
<b>argv[3]</b>	job execution group name
<b>argv[4]</b>	job name
<b>argv[5]</b>	session id
<b>argv[6]</b>	list of requested resource limits
<b>argv[7]</b>	list of resources used by job
<b>argv[8]</b>	job execution queue
<b>argv[9]</b>	job account
<b>argv[10]</b>	job exit code
<b>argv[11]</b>	job script location

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.

**i** The cancel job command (`qdel`) will take as long to return as the `epilogue.precancel` script takes to run. For example, if the script runs for 5 minutes, it takes 5 minutes for `qdel` to return.

### G.3.3 Environment Variables

For all scripts, the environment passed to the script is empty. When submitting a job through *qsub* or *msub -E*, Torque defines variables.

In this topic:

[G.3.3.A qsub](#)

[G.3.3.B msub -E](#)

#### G.3.3.A qsub

When submitting a job through *qsub*, Torque defines the following variables:

Variable	Description
\$PBS_MSHOST	Mother superior's hostname.
\$PBS_RESOURCE_NODES	-l nodes request made to the job, if any.
\$PBS_O_WORKDIR	Job's working directory.
\$PBS_NODENUM	Node index for the job of the node where this prologue or epilogue is executing.
\$PBS_NUM_NODES	Number of nodes requested for the job (1 if no -l nodes request was made).
\$PBS_NP	Number of execution slots used for the job. For example, -l nodes=2:ppn=4 will have \$PBS_NP defined as 8.
\$PBS_NUM_PPN	ppn request, if one was made. If more than one was made, it will be the first one. For example: -l nodes=2:ppn=3+4:ppn=2 will have this variable set to 3.
\$PBS_NODEFILE	Path to the job's nodefile.

#### G.3.3.B msub -E

If you submit the job using *msub -E*, these Moab environment variables are available:

- MOAB\_CLASS
- MOAB\_GROUP
- MOAB\_JOBARRAYINDEX
- MOAB\_JOBARRAYRANGE
- MOAB\_JOBID
- MOAB\_JOBNAME
- MOAB\_MACHINE
- MOAB\_NODECOUNT
- MOAB\_NODELIST
- MOAB\_PARTITION
- MOAB\_PROCCOUNT
- MOAB\_QOS
- MOAB\_TASKMAP
- MOAB\_USER

See 'msub' in the *Moab Workload Manager Administrator Guide* for more information.

### G.3.4 Standard Input

Standard input for both scripts is connected to a system dependent file. Currently, for all systems this is `/dev/null`.

Except for epilogue scripts of an interactive job, `prologue.parallel`, `epilogue.precancel`, and `epilogue.parallel`, the standard output and error are connected to output 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`.

For `prologue.parallel` and `epilogue.parallel`, the user will need to redirect `stdout` and `stderr` manually.

## G.4 Per Job Prologue and Epilogue Scripts

Torque 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 read privileges, and the permissions must not be writeable by group or other.

```
/home/usertom/dev/
```

```
-r-x----- 1 usertom usertom 24 2025-11-09 16:11 prologue_script.sh
-r-x----- 1 usertom usertom 24 2025-11-09 16:11 epilogue_script.sh
```

**Example G-1:**

```
$ 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.5 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.

**i** While Torque is executing the `epilogue`, `epilogue.user`, or `epilogue.precancel` scripts, the job will be in the *E* (exiting) state.

If an `epilogue.parallel` script cannot open the `.OU` or `.ER` files, an error is logged but the script is continued.

## G.6 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 G-2:*

Following are example prologue and epilogue scripts that write the arguments passed to them in the job's standard out file:


prologue	
<b>Script</b>	<pre>#!/bin/sh echo "Prologue Args:" echo "Job ID: \$1" echo "User ID: \$2" echo "Group ID: \$3" echo ""  exit 0</pre>
<b>stdout</b>	<pre>Prologue Args: Job ID: 13724.node01 User ID: user1 Group ID: user1</pre>

epilogue	
Script	<pre>#!/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</pre>
stdout	<pre>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:</pre>

Example G-3:

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

```

**i** 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

In this appendix:

[H.1 Configuring Multiple Servers to Run on the Same Node](#)

[H.2 Configuring the First Torque](#)

[H.3 Configuring the Second Torque](#)

[H.4 Bringing the First Torque Server Online](#)

[H.5 Bringing the Second Torque Server Online](#)

### H.1 Configuring Multiple Servers to Run 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.2 Configuring the First Torque

```
./configure --with-server-home=/usr/spool/torque1 --bindir=/usr/spool/torque1/bin --  
sbindir=/usr/spool/torque1/sbin
```

Then `make` and `make install` will place the first Torque into `/usr/spool/torque1` with the executables in their corresponding directories.

### H.3 Configuring the Second Torque

```
./configure --with-server-home=/usr/spool/torque2 --bindir=/usr/spool/torque2/bin --  
sbindir=/usr/spool/torque2/sbin
```

Then `make` and `make install` will place the second Torque into `/usr/spool/torque2` with the executables in their corresponding directories.

### H.4 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.



In this example, the first Torque server will accept batch requests on port 35000 and communicate with the MOMs on port 35001. The first Torque MOM will try to connect to the server on port 35000 and it will listen for requests from the server on port 35001. (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 -t create
> pbs_mom -S 35000 -M 35001
```

Afterwards, when using a client command to make a batch request, it is necessary to specify the server name and server port (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.5 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 36001, and communicate via TCP 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 TCP 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 server name and server port (36002):

```
> pbsnodes -a -s node01:36000
> qsub -q @node01:36000 /tmp/script.pbs
```

## Appendix I: Security Overview

The authorization model for Torque uses a daemon called `trqauthd`. The job of the `trqauthd` daemon is the same as `pbs_iff`. The difference is that `trqauthd` is a resident daemon whereas `pbs_iff` is invoked by each client command. `pbs_iff` is not scalable and is prone to failure under even small loads. `trqauthd` is very scalable and creates the possibility for better security measures in the future.

### **trqauthd Authorization Theory**

The key to security of `trqauthd` is the assumption that any host that has been added to the Torque cluster has been secured by the admin. `trqauthd` does not do authentication, just authorization of users. Given that the host system is secure, the following is the procedure by which `trqauthd` authorizes users to `pbs_server`:

1. Client utility makes a connection to `pbs_server` on a dynamic port.
2. Client utility sends a request to `trqauthd` with the user name and port.
3. `trqauthd` verifies the user ID and then sends a request to `pbs_server` on a privileged port with the user ID and dynamic port to authorize the connection.
4. `trqauthd` reports results of the server to client utility.

`trqauthd` uses UNIX Domain Sockets for communication from the client utility. UNIX Domain Sockets have the ability to verify that a user is who they say they are by using security features that are part of the file system.

## Appendix J: Job Submission Filter (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 - Verify 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 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
```

The same command line arguments passed to *qsub* will be passed to the submit filter and in the same order. 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 and must be available on each of the nodes where users can submit jobs. Because the submit filter is likely to run multiple times for each job submission, all operations in the submit filter must be idempotent (i.e., they must produce the same results if called more than once with the same input parameters).

By default, the submit filter must be located at `/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 file (see [Appendix K: torque.cfg Configuration 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


Administrators can configure the `torque.cfg` file [located in `PBS_SERVER_HOME` (`/var/spool/torque` by default)] to alter the behavior of the `qsub` command on specific host machines where the file resides. This file contains a list of parameters and values separated by whitespace. This only affects `qsub`, and only on each specific host with the file.

## Configuration Parameters

CLIENTRETRY	DEFAULTCKPT	FAULT_TOLERANT_BY_DEFAULT
HOST_NAME_SUFFIX	INTERACTIVE_PORT_RANGE	QSUBHOST
QSUBSENDUID	QSUBSLEEP	RERUNNABLEBYDEFAULT
SERVERHOST	SUBMITFILTER	TRQ_IFNAME
VALIDATEGROUP	VALIDATEPATH	

CLIENTRETRY	
Format	<INT>
Default	0
Description	Seconds between retry attempts to talk to <code>pbs_server</code> .
Example	<div><pre>CLIENTRETRY 10</pre><p><i>Torque waits 10 seconds after a failed attempt before it attempts to talk to <code>pbs_server</code> again.</i></p></div>

DEFAULTCKPT	
Format	One of <code>None</code> , <code>Enabled</code> , <code>Shutdown</code> , <code>Periodic</code> , <code>Interval=minutes</code> , <code>depth=number</code> , or <code>dir=path</code>
Default	<code>None</code>
Description	Default value for job's checkpoint attribute. For a description of all possible

DEFAULTCKPT	
	<p>values, see <a href="#">qsub</a>.</p> <div>  This default setting can be overridden at job submission with the <code>qsub -c</code> option. </div>
<b>Example</b>	<div> <pre>DEFAULTCKPT Shutdown</pre> </div> <div> <i>By default, Torque checkpoints at pbs_mom shutdown.</i> </div>

FAULT_TOLERANT_BY_DEFAULT	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	Sets all jobs to fault tolerant by default (see <a href="#">-i</a> for more information on fault tolerance).
<b>Example</b>	<div> <pre>FAULT_TOLERANT_BY_DEFAULT TRUE</pre> </div> <div> <i>Jobs are fault tolerant by default. They will not be canceled based on failed polling, no matter how many nodes fail to report.</i> </div>

HOST_NAME_SUFFIX	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	Specifies a hostname suffix. When <code>qsub</code> submits a job, it also submits the username of the submitter and the name of the host from which the user submitted the job. Torque appends the value of <code>HOST_NAME_SUFFIX</code> to the hostname. This is useful for multi-homed systems that may have more than one name for a host.
<b>Example</b>	<div> <pre>HOST_NAME_SUFFIX -ib</pre> </div> <div> <i>When a job is submitted, the -ib suffix is appended to the host name.</i> </div>

INTERACTIVE_PORT_RANGE	
<b>Format</b>	<INTEGER>-<INTEGER>
<b>Default</b>	---
<b>Description</b>	Sets a range of ports for interactive jobs. The minimum port must be greater than 1024, and the maximum port must be greater than the minimum port, or else the setting will be ignored.
<b>Example</b>	<pre>INTERACTIVE_PORT_RANGE 20000-20100</pre> <p><i>Force all interactive listening ports on this host to be between 20000 and 20100, inclusive.</i></p>

QSUBHOST	
<b>Format</b>	<HOSTNAME>
<b>Default</b>	---
<b>Description</b>	The hostname given as the argument of this option will be used as the PBS_O_HOST variable for job submissions. By default, PBS_O_HOST is the hostname of the submission host. This option enables admins to override the default hostname and substitute a new name.
<b>Example</b>	<pre>QSUBHOST host1</pre> <p><i>The default hostname associated with a job is host1.</i></p>

QSUBSENDUID	
<b>Format</b>	N/A
<b>Default</b>	---
<b>Description</b>	Integer for job's PBS_O_UID variable. Specifying the parameter name anywhere in the config file enables the feature. Removing the parameter name disables the feature.
<b>Example</b>	<pre>QSUBSENDUID</pre> <p><i>Torque assigns a unique ID to a job when it is submitted by qsub.</i></p>

QSUBSLEEP	
<b>Format</b>	<INT>
<b>Default</b>	0
<b>Description</b>	Specifies time, in seconds, to sleep between a user's submitting and Torque's starting a <i>qsub</i> command. Used to prevent users from overwhelming the scheduler.
<b>Example</b>	<pre>QSUBSLEEP 2</pre> <p><i>When a job is submitted with <i>qsub</i>, it will sleep for 2 seconds.</i></p>

RERUNNABLEBYDEFAULT	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	TRUE
<b>Description</b>	Specifies if a job is re-runnable by default. Setting this to false causes the re-runnable attribute value to be false unless the user specifies otherwise with the <i>qsub-r</i> option.
<b>Example</b>	<pre>RERUNNABLEBYDEFAULT FALSE</pre> <p><i>By default, <i>qsub</i> jobs cannot be rerun.</i></p>

SERVERHOST	
<b>Format</b>	<STRING>
<b>Default</b>	localhost
<b>Description</b>	If set, the <i>qsub</i> command will open a connection to the host specified by the SERVERHOST string.
<b>Example</b>	<pre>SERVERHOST orion15</pre> <p><i>The server will open socket connections and communicate using <i>serverhost orion15</i>.</i></p>



SUBMITFILTER	
<b>Format</b>	<STRING>
<b>Default</b>	/usr/local/sbin/torque_submitfilter
<b>Description</b>	Specifies the location of the submit filter used to preprocess job submission. See <a href="#">Appendix J: Job Submission Filter (qsub Wrapper)</a> for additional information.
<b>Example</b>	<pre>SUBMITFILTER /usr/local/sbin/torque_submitfilter</pre> <p><i>The location of the submit filter is specified as /usr/local/sbin/torque_submitfilter.</i></p>

TRQ_IFNAME	
<b>Format</b>	<STRING>
<b>Default</b>	null
<b>Description</b>	Allows you to specify a specific network interface to use for outbound Torque requests. The string is the name of a network interface, such as <i>eth0</i> or <i>eth1</i> , depending on which interface you want to use.
<b>Example</b>	<pre>TRQ_IFNAME eth1</pre> <p><i>Outbound Torque requests are handled by eth1.</i></p>

VALIDATEGROUP	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	Validate submit user's group on <a href="#">qsub</a> commands. <i>VALIDATEGROUP</i> also checks any groups requested in <i>group_list</i> at the submit host. Set <i>VALIDATEGROUP</i> to TRUE if you set <a href="#">disable_server_id_check</a> to TRUE.
<b>Example</b>	<pre>VALIDATEGROUP TRUE</pre> <p><i>qsub verifies the submitter's group ID.</i></p>

VALIDATEPATH	
Format	<BOOLEAN>
Default	TRUE
Description	Validate local existence of -d and/or -w working directories.
Example	<div>VALIDATEPATH FALSE</div> <div><i>qsub does not validate the path.</i></div>

## Appendix L: Torque Quick Start Guide

In this appendix:

- [L.1 Initial Installation](#)
- [L.2 Initialize/Configure Torque on the Server \(pbs\\_server\)](#)
- [L.3 Install Torque on the Compute Nodes](#)
- [L.4 Configure Torque on the Compute Nodes](#)
- [L.5 Configure Data Management on the Compute Nodes](#)
- [L.6 Update Torque Server Configuration](#)
- [L.7 Start the pbs\\_mom Daemons on Compute Nodes](#)
- [L.8 Verify Correct Torque Installation](#)
- [L.9 Enable the Scheduler](#)
- [L.10 \(Optional\) Startup/Shutdown Service Script for Torque/Moab](#)

### Related Topics

- [2.3 Advanced Configuration](#)

## L.1 Initial Installation

Download the latest Torque build from [Adaptive Computing Torque Downloads](#).

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 -xzf torque.tar.gz
> cd torque
> ./configure
> make
> make install
```

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

**i** 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`.

**i** 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 [2.6 Testing Server Configuration](#).

## L.3 Install Torque on the Compute Nodes

To configure a compute node, do the following on each machine (see Section 3.2.1 of [PBS Administrator Guide](#) 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 (i.e., 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, you must configure the MOM daemon to trust the `pbs_server` host. The recommended method for doing this is to create the `TORQUE_HOME/server_name` file with the server hostname in it. Alternatively, you can add a `$pbs_server` line to the `TORQUE_HOME/mom_priv/config` file.

Additional config parameters can be added to `TORQUE_HOME/mom_priv/config` (see [Appendix C: Node Manager \(MOM\) Configuration](#) for details).

See [2.2.1 Specifying Compute Nodes](#) for more information about configuring `pbs_server` to identify compute nodes.

## 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.

Run the trqauthd daemon to run client commands (see [2.2.4 Configuring trqauthd for Client Commands](#)). This enables running client commands.

## L.8 Verify 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 section.

## L.9 Enable 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, see Moab-Torque Integration Guide in the *Moab Workload Manager Administrator Guide*. If using pbs\_sched, start this daemon now.

**i** If you are installing ClusterSuite, Torque and Moab were configured at installation for interoperability and no further action is required.

## L.10 (Optional) Startup/Shutdown Service Script for Torque/Moab

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 Red Hat, etc.).

## Appendix M: BLCR Acceptance Tests

This section contains a description of the testing done to verify the functionality of the BLCR implementation.

In this appendix:

- [M.1 Test Environment](#)
- [M.2 Test 1 - Basic Operation](#)
- [M.3 Test 2 - Persistence of Checkpoint Images](#)
- [M.4 Test 3 - Restart After Checkpoint](#)
- [M.5 Test 4 - Multiple Checkpoint/Restart](#)
- [M.6 Test 5 - Periodic Checkpoint](#)
- [M.7 Test 6 - Restart from Previous Image](#)

### M.1 Test Environment

All these tests assume the following test program and shell script `test.sh`:

```
#include
int main( int argc, char *argv[] )
{
    int i;

    for (i=0; i<100; i++)
    {
        printf("i = %d\n", i);
        fflush(stdout);
        sleep(1);
    }
}
#!/bin/bash

/home/test/test
```

### M.2 Test 1 - Basic Operation

In this section:



- [M.2.1 Introduction](#)
- [M.2.2 Test Steps](#)
- [M.2.3 Possible Failures](#)
- [M.2.4 Successful Results](#)

## M.2.1 Introduction

This test determines if the proper environment has been established.

## M.2.2 Test Steps

Submit a test job and then issue a hold on the job:

```
> qsub -c enabled test.sh
999.xxx.yyy
> qhold 999
```

## M.2.3 Possible Failures

Normally the result of *qhold* is nothing. If an error message is produced saying that *qhold* is not a supported feature, then one of the following configuration errors might be present:

- The Torque images may not have been configured with `--enable-bldr`
- BLCR support may not be installed into the kernel with `insmod`
- The config script in `mom_priv` may not exist with `$checkpoint_script` defined
- The config script in `mom_priv` may not exist with `$restart_script` defined
- The config script in `mom_priv` may not exist with `$checkpoint_run_exe` defined
- The scripts referenced in the config file may not exist
- The scripts referenced in the config file may not have the correct permissions

## M.2.4 Successful Results

If no configuration was done to specify a specific directory location for the checkpoint file, the default location is off of the Torque directory, which in my case is

```
/var/spool/torque/checkpoint.
```

Otherwise, go to the specified directory for the checkpoint image files. This was done by either specifying an option on job submission (i.e., `-c dir=/home/test`) or by setting an attribute on the execution queue. This is done with the command `qmgr -c 'set queue batch checkpoint_dir=/home/test'`.

Doing a directory listing shows the following:

```
# find /var/spool/torque/checkpoint
/var/spool/torque/checkpoint
/var/spool/torque/checkpoint/999.xxx.yyy.CK
/var/spool/torque/checkpoint/999.xxx.yyy.CK/ckpt.999.xxx.yyy.1205266630
# find /var/spool/torque/checkpoint |xargs ls -l
-r----- 1 root root 543779 2025-03-11 14:17
/var/spool/torque/checkpoint/999.xxx.yyy.CK/ckpt.999.xxx.yyy.1205266630

/var/spool/torque/checkpoint:
total 4
drwxr-xr-x 2 root root 4096 2025-03-11 14:17 999.xxx.yyy.CK

/var/spool/torque/checkpoint/999.xxx.yyy.CK:
total 536
-r----- 1 root root 543779 2025-03-11 14:17 ckpt.999.xxx.yyy.1205266630
```

Doing a `qstat -f` command should show the job in a held state, *job\_state = H*. Note that the attribute `checkpoint_name` is set to the name of the file seen above.

If a checkpoint directory has been specified, there will also be an attribute *checkpoint\_dir* in the output of `qstat -f`:

```
$ qstat -f
Job Id: 999.xxx.yyy
  Job_Name = test.sh
  Job_Owner = test@xxx.yyy
  resources_used.cput = 00:00:00
  resources_used.mem = 0kb
  resources_used.vmem = 0kb
  resources_used.walltime = 00:00:06
  job_state = H
  queue = batch
  server = xxx.yyy
  Checkpoint = u
  ctime = Tue Mar 11 14:17:04 2025
  Error_Path = xxx.yyy:/home/test/test.sh.e999
  exec_host = test/0
  Hold_Types = u
  Join_Path = n
  Keep_Files = n
  Mail_Points = a
  mtime = Tue Mar 11 14:17:10 2025
  Output_Path = xxx.yyy:/home/test/test.sh.o999
  Priority = 0
  qtime = Tue Mar 11 14:17:04 2025
  Rerunable = True
  Resource_List.needsnodes = 1
  Resource_List.nodect = 1
```

```

Resource_List.nodes = 1
Resource_List.walltime = 01:00:00
session_id = 9402 substate = 20
Variable_List = PBS_O_HOME=/home/test,PBS_O_LANG=en_US.UTF-8,
    PBS_O_LOGNAME=test,
    PBS_O_PATH=/usr/local/perltests/bin:/home/test/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games,
    PBS_O_SHELL=/bin/bash,PBS_SERVER=xxx.yyy,
    PBS_O_HOST=xxx.yyy,PBS_O_WORKDIR=/home/test,
    PBS_O_QUEUE=batch
euser = test
egroup = test
hashname = 999.xxx.yyy
queue_rank = 3
queue_type = E comment = Job started on Tue Mar 11 at 14:17
exit_status = 271
submit_args = test.sh
start_time = Tue Mar 11 14:17:04 2025
start_count = 1
checkpoint_dir = /var/spool/torque/checkpoint/999.xxx.yyy.CK
checkpoint_name = ckpt.999.xxx.yyy.1205266630

```

**i** The value of `Resource_List.*` is the amount of resources requested.

## M.3 Test 2 - Persistence of Checkpoint Images

### M.3.1 Introduction

This test determines if the checkpoint files remain in the default directory after the job is removed from the Torque queue.

Note that this behavior was requested by a customer but in fact may not be the right thing to do as it leaves the checkpoint files on the execution node. These will gradually build up over time on the node being limited only by disk space. The right thing would seem to be that the checkpoint files are copied to the user's home directory after the job is purged from the execution node.

### M.3.2 Test Steps

Assuming the steps of Test 1 (see [M.2 Test 1 - Basic Operation](#)), delete the job and then wait until the job leaves the queue after the completed job hold time. Then look at the contents of the default checkpoint directory to see if the files are still there.

```

> qsub -c enabled test.sh
999.xxx.yyy
> qhold 999

```

```
> qdel 999
> sleep 100
> qstat
>
> find /var/spool/torque/checkpoint
... files ...
```

### M.3.3 Possible Failures

The files are not there, did Test 1 actually pass?

### M.3.4 Successful Results

The files are there.

## M.4 Test 3 - Restart After Checkpoint

### M.4.1 Introduction

This test determines if the job can be restarted after a checkpoint hold.

### M.4.2 Test Steps

Assuming the steps of Test 1 (see [M.2 Test 1 - Basic Operation](#)), issue a `qrls` command. Have another window open into the `/var/spool/torque/spool` directory and tail the job.

### M.4.3 Successful Results

After the `qrls`, the job's output should resume.

## M.5 Test 4 - Multiple Checkpoint/Restart

### M.5.1 Introduction

This test determines if the checkpoint/restart cycle can be repeated multiple times.

## M.5.2 Test Steps

Start a job and then while tailing the job output, do multiple `qhold`/`qrls` operations:

```
> qsub -c enabled test.sh
999.xxx.yyy
> qhold 999
> qrls 999
> qhold 999
> qrls 999
> qhold 999
> qrls 999
```

## M.5.3 Successful Results

After each `qrls`, the job's output should resume. Also tried `while true; do qrls 999; qhold 999; done` and this seemed to work as well.

# M.6 Test 5 - Periodic Checkpoint

## M.6.1 Introduction

This test determines if automatic periodic checkpoint will work.

## M.6.2 Test Steps

Start the job with the option `-c enabled,periodic,interval=1` and look in the checkpoint directory for checkpoint images to be generated about every minute:

```
> qsub -c enabled,periodic,interval=1 test.sh
999.xxx.yyy
```

## M.6.3 Successful Results

After each `qrls`, the job's output should resume. Also tried "`while true; do qrls 999; qhold 999; done`" and this seemed to work as well.

## M.7 Test 6 - Restart from Previous Image

### M.7.1 Introduction

This test determines if the job can be restarted from a previous checkpoint image.

### M.7.2 Test Steps

Start the job with the option `-c enabled,periodic,interval=1` and look in the checkpoint directory for checkpoint images to be generated about every minute. Do a `qhold` on the job to stop it. Change the attribute `checkpoint_name` with the `qalter` command. Then do a `qrls` to restart the job.

```
> qsub -c enabled,periodic,interval=1 test.sh
999.xxx.yyy
> qhold 999
> qalter -W checkpoint_name=ckpt.999.xxx.yyy.1234567
> qrls 999
```

### M.7.3 Successful Results

The job output file should be truncated back and the count should resume at an earlier number.

# Appendix N: Queue Attributes

This appendix provides information on the different queue attributes.

In this appendix:

[N.1 Queue Attribute Reference](#)

[N.2 Attributes](#)

[N.3 Assigning Queue Resource Limits](#)

## N.1 Queue Attribute Reference



In addition to information on the different queue attributes, this appendix lists some queue resource limits. See [N.3 Assigning Queue Resource Limits](#).

**i** For Boolean attributes, *T*, *t*, *1*, *Y*, and *y* are all synonymous with 'TRUE,' and *F*, *f*, *0*, *N*, and *n* all mean 'FALSE.'

## N.2 Attributes

<a href="#">acl_groups</a>	<a href="#">disallowed_types</a>	<a href="#">max_user_queueable</a>	<a href="#">resources_default</a>
<a href="#">acl_group_enable</a>	<a href="#">enabled</a>	<a href="#">max_user_run</a>	<a href="#">resources_max</a>
<a href="#">acl_group_sloppy</a>	<a href="#">features_required</a>	<a href="#">priority</a>	<a href="#">resources_min</a>
<a href="#">acl_hosts</a>	<a href="#">ghost_queue</a>	<a href="#">queue_type</a>	<a href="#">route_destinations</a>
<a href="#">acl_host_enable</a>	<a href="#">keep_completed</a>	<a href="#">req_information_max</a>	<a href="#">started</a>
<a href="#">acl_logic_or</a>	<a href="#">kill_delay</a>	<a href="#">req_information_min</a>	
<a href="#">acl_users</a>	<a href="#">max_queueable</a>	<a href="#">required_login_property</a>	
<a href="#">acl_user_enable</a>	<a href="#">max_running</a>	<a href="#">resources_available</a>	

acl_groups	
<b>Format</b>	<GROUP>[@<HOST>][+<USER>[@<HOST>]]..
<b>Default</b>	---
<b>Description</b>	Specifies the list of groups that can submit jobs to the queue. If <code>acl_group_enable</code> is set to true, only users with a primary group listed in <code>acl_groups</code> can


acl_groups	
	<p>utilize the queue.</p> <div>  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. </div>
<b>Example</b>	<div> <pre>&gt; qmgr -c "set queue batch acl_groups=staff" &gt; qmgr -c "set queue batch acl_groups+=ops@h1" &gt; qmgr -c "set queue batch acl_groups+=staff@h1"</pre> </div> <div>  Used in conjunction with <a href="#">acl_group_enable</a>. </div>

acl_group_enable	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	If TRUE, constrains Torque to only allow jobs submitted from groups specified by the <a href="#">acl_groups</a> parameter.
<b>Example</b>	<div> <pre>qmgr -c "set queue batch acl_group_enable=true"</pre> </div>

acl_group_sloppy	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	If TRUE, <a href="#">acl_groups</a> will be checked against all groups of which the job users is a member.
<b>Example</b>	---

acl_hosts	
<b>Format</b>	<HOST>[+<HOST>]...
<b>Default</b>	---




acl_hosts	
<b>Description</b>	Specifies the list of hosts that can submit jobs to the queue.
<b>Example</b>	<pre>qmgr -c "set queue batch acl_hosts=h1+h1+h1"</pre> <div>  Used in conjunction with <code>acl_host_enable</code>. </div>

acl_host_enable	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	If <code>TRUE</code> , constrains Torque to only allow jobs submitted from hosts specified by the <code>acl_hosts</code> parameter.
<b>Example</b>	<pre>qmgr -c "set queue batch acl_host_enable=true"</pre>

acl_logic_or	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	If <code>TRUE</code> , user and group acls are logically ORed together, meaning that either acl may be met to allow access. If <code>FALSE</code> or unset, then both acls are ANDed, meaning that both acls must be satisfied.
<b>Example</b>	<pre>qmgr -c "set queue batch acl_logic_or=true"</pre>

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

acl_users	
<b>Example</b>	<pre>&gt; qmgr -c "set queue batch acl_users=john" &gt; qmgr -c "set queue batch acl_users+=steve@h1" &gt; qmgr -c "set queue batch acl_users+=stevek@h1"</pre> <div>  Used in conjunction with <code>acl_user_enable</code>. </div>

acl_user_enable	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	If TRUE, constrains Torque to only allow jobs submitted from users specified by the <code>acl_users</code> parameter.
<b>Example</b>	<pre>qmgr -c "set queue batch acl_user_enable=true"</pre>

disallowed_types	
<b>Format</b>	<type>[+<type>]...
<b>Default</b>	---
<b>Description</b>	Specifies classes of jobs that are not allowed to be submitted to this queue. Types are interactive, batch, rerunable, nonrerunable, fault_tolerant, fault_intolerant, and job_array.
<b>Example</b>	<pre>qmgr -c "set queue batch disallowed_types = interactive" qmgr -c "set queue batch disallowed_types += job_array"</pre>

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

features_required	
<b>Format</b>	feature1[,feature2[,feature3...]]
<b>Default</b>	---
<b>Description</b>	Specifies that all jobs in this queue will require these features in addition to any they may have requested. A feature is a synonym for a property.
<b>Example</b>	<pre>qmgr -c 's q batch features_required=fast'</pre>

ghost_queue	
<b>Format</b>	<BOOLEAN>
<b>Default</b>	FALSE
<b>Description</b>	Intended for automatic, internal recovery (by the server) only. If set to <b>TRUE</b> , the queue rejects new jobs, but permits the server to recognize the ones currently queued and/or running. Unset this attribute in order to approve a queue and restore it to normal operation. See <a href="#">13.1 Automatic Queue and Job Recovery</a> for more information regarding this process.
<b>Example</b>	<pre>qmgr -c "unset queue batch ghost_queue"</pre>

keep_completed	
<b>Format</b>	<INTEGER>
<b>Default</b>	0
<b>Description</b>	Specifies the number of seconds jobs should be held in the Completed state after exiting. For more information, see <a href="#">3.5 Keeping Completed Jobs</a> .
<b>Example</b>	<pre>qmgr -c "set queue batch keep_completed=120"</pre>

kill_delay	
<b>Format</b>	<INTEGER>

kill_delay	
<b>Default</b>	2
<b>Description</b>	<p>Specifies the number of seconds between sending a SIGTERM and a SIGKILL to a job in a specific queue that you want to cancel. It is possible that the job script, and any child processes it spawns, can receive several SIGTERM signals before the SIGKILL signal is received.</p> <div> <p><b>i</b> All MOMs must be configured with <code>\$exec_with_exec true</code> in order for <code>kill_delay</code> to work, even when relying on default <code>kill_delay</code> settings.</p> <p><b>i</b> This setting overrides the server setting. See <code>kill_delay</code> in <a href="#">Appendix B: Server Parameters</a>.</p> </div>
<b>Example</b>	<pre>qmgr -c "set queue batch kill_delay=30"</pre>

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


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



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


max_user_run	
<b>Format</b>	<INTEGER>
<b>Default</b>	unlimited
<b>Description</b>	This limits the maximum number of jobs a user can have running from the given queue.
<b>Example</b>	<pre>qmgr -c "set queue batch max_user_run=10"</pre>

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

queue_type	
<b>Format</b>	One of <i>e</i> , <i>execution</i> , <i>r</i> , or <i>route</i> (see <a href="#">5.1.5 Creating a Routing Queue</a> )
<b>Default</b>	---
<b>Description</b>	Specifies the queue type.

queue_type	
	 This value must be explicitly set for all queues.
<b>Example</b>	<pre>qmgr -c "set queue batch queue_type=execution"</pre>

req_information_max	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	<p>Specifies the maximum resource limits allowed for jobs submitted to a queue.</p> <p> These limits apply only to the qsub -L job submission option.</p> <p>Values are lprocs, node, socket, numachip, core, thread, memory, swap, and disk.</p> <p> If a maximum core count is specified, jobs with usecores must have lprocs&lt;= the maximum core count; jobs without usecores are rejected. If a maximum thread count is specified, lprocs must be &lt;= the maximum thread count.</p>
<b>Example</b>	<pre>qmgr -c "set queue batch req_information_max.lprocs=8"</pre>


req_information_min	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	<p>Specifies the minimum resource limits allowed for jobs submitted to a queue.</p> <p> These limits apply only to the qsub -L job submission option.</p> <p>Values are lprocs, node, socket, numachip, core, thread, memory, swap, and disk.</p>

req_information_min	
	<p><b>i</b> If a minimum core count is specified, jobs with usecores must have <code>lprocs</code> <math>\geq</math> the minimum core count; jobs without usecores are rejected. If a minimum thread count is specified, <code>lprocs</code> must be <math>\geq</math> the minimum thread count.</p>
<b>Example</b>	<pre>qmgr -c "set queue batch req_information_min.lprocs=2"</pre>

required_login_property	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	Adds the specified login property as a requirement for all jobs in this queue.
<b>Example</b>	<pre>qmgr -c 's q &lt;queue name&gt; required_login_property=INDUSTRIAL'</pre>

resources_available	
<b>Format</b>	<STRING>
<b>Default</b>	---
<b>Description</b>	Specifies to cumulative resources available to all jobs running in the queue. See <a href="#">13.9.6 qsub will not allow the submission of jobs requesting many processors</a> for more information.
<b>Example</b>	<pre>qmgr -c "set queue batch resources_available.nodect=20"</pre> <p><b>i</b> You must restart <code>pbs_server</code> for changes to take effect. Also, <code>resources_available</code> is constrained by the smallest of <code>queue.resources_available</code> and <code>server.resources_available</code>.</p>

resources_default	
<b>Format</b>	<STRING>


resources_default	
<b>Default</b>	---
<b>Description</b>	Specifies default resource requirements for jobs submitted to the queue.
<b>Example</b>	<pre>qmgr -c "set queue batch resources_default.walltime=3600"</pre> <div>  See <a href="#">5.1.2 Setting Queue Resource Controls</a> for more information about setting queue resource requirements and the use of <code>-l</code> and <code>-L</code> job submission syntaxes. </div>

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

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

route_destinations	
<b>Format</b>	<queue>[@<host>]
<b>Default</b>	---
<b>Description</b>	Specifies the potential destination queues for jobs submitted to the associated routing queue.




route_destinations	
	<div><div></div><div>This attribute is only valid for routing queues (see <a href="#">5.1.5 Creating a Routing Queue</a>).</div></div>
Example	<div><pre>&gt; qmgr -c "set queue route route_destinations=fast" &gt; qmgr -c "set queue route route_destinations+=slow" &gt; qmgr -c "set queue route route_destinations+=medium@hostname"</pre></div> <div>To set multiple queue specifications, use multiple commands:</div> <div><pre>&gt; qmgr -c 's s route_destinations=batch' &gt; qmgr -c 's s route_destinations+=long' &gt; qmgr -c 's s route_destinations+=short'</pre></div>

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

### N.3 Assigning Queue Resource Limits

Administrators can use resource limits to help direct what kind of jobs go to different queues. There are four queue attributes where resource limits can be set: [resources\\_available](#), [resources\\_default](#), [resources\\_max](#), and [resources\\_min](#). The list of supported resources that can be limited with these attributes are *arch*, *mem*, *n\_cpus*, *nodect*, *nodes*, *pmem*, *procct*, *pvmem*, *vmem*, and *walltime*.

Resource	Format	Description
<b>arch</b>	string	Specifies the administrator defined system architecture required.
<b>mem</b>	<a href="#">size</a>	Amount of physical memory used by the job. (Ignored on Darwin, Digital UNIX, Free BSD, HP-UX 11, IRIX, NetBSD, and SunOS. Also ignored on Linux if number of nodes is not 1. Not implemented on AIX and HP-UX 10.)

Resource	Format	Description
<b>ncpus</b>	integer	<p>Sets the number of processors in one task where a task cannot span nodes.</p> <div>  You cannot request both ncpus and nodes in the same queue. </div>
<b>nodect</b>	integer	Sets the number of nodes available. By default, Torque will set the number of nodes available to the number of nodes listed in the <code>TORQUE_HOME/server_priv/nodes</code> file. <code>nodect</code> can be set to be greater than or less than that number. Generally, it is used to set the node count higher than the number of physical nodes in the cluster.
<b>nodes</b>	integer	Specifies the number of nodes.
<b>pmem</b>	size	Specifies the maximum amount of physical memory to be used by any single process of the job. (Ignored on Fujitsu. Not implemented on Digital UNIX and HPUX.)
<b>procct</b>	integer	<p>Sets limits on the total number of execution slots (procs) allocated to a job. The number of procs is calculated by summing the products of all node and ppn entries for a job.</p> <p>For example <code>qsub -l nodes=2:ppn=2+3:ppn=4 job.sh</code> would yield a <code>procct</code> of 16. <math>2*2</math> (<math>2:ppn=2</math>) + <math>3*4</math> (<math>3:ppn=4</math>).</p>
<b>pvmem</b>	size	Amount of virtual memory used by any single process in a job.
<b>vmem</b>	size	Amount of virtual memory used by all concurrent processes in the job.
<b>walltime</b>	seconds, or [[HH:]MM:]SS	Amount of real time during which a job can be in a running state.

**size**

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

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

---

## Related Topics

- [5.1 Queue Configuration](#)