PBS Pro™ 5.3
Administrator Guide

PBS Pro™

Release 5.3

Administrator Guide

Portable Batch System™ Administrator Guide

PBS-3BA01, Release: PBS Pro™ 5.3, Updated: December 6, 2002

Edited by: James Patton Jones

Contributing authors include: Albeaus Bayucan, Robert L. Henderson, James Patton Jones, Casimir Lesiak, Bhroam Mann, Bill Nitzberg, Tom Proett.

For more information, redistribution, licensing, or additional copies of this publication, contact:

	Phone: +1 (650) 967-4675
PBS Products Dept.	FAX: +1 (650) 967-3080
2672 Bayshore Parkway, Suite 810	URL: www.pbspro.com
Mountain View, CA 94043	Email: sales@pbspro.com
	support@pbspro.com

For online purchases, visit: store.pbspro.com

Table of Contents

Acknowledgements .... **ix**
Preface .... **xi**
1 Introduction **1**
Book Organization 1
What is PBS Pro? 2
About the PBS Team 4
2 Concepts and Terms **5**
PBS Components 6
Defining PBS Terms 8
3 Pre-Installation Planning **13**
New Features in PBS Pro 5.3 13
Server - MOM Protocol Change 14
Changes to Time-shared and Cluster Nodes 14
Planning 14
Interfacing with Globus 15
Single Execution System 16
Multiple Execution Systems 17
4 Installation **19**
Overview 19
Media Setup 20
Default Install Options 21
Installation on UNIX/Linux Systems 21
Installation on Windows 2000 Systems 24
Installing the PBS License 25
Using Floating Licenses 26
Installing Multiple PBS Licenses 27
5 Installation from Source **29**
Tar File 29
Optional Components 30
Build Steps 30
Building PBS 31
Overview 32
Build Details 35

Make File Targets 44
Machine Dependent Build Instructions 44
Install Options.. 51
6 Upgrading PBS..53
Overlay Upgrade.. 53
Migration Upgrade.. 54
Alternate Test Systems 57
Dependent Jobs and Test Systems 59
7 Configuring the Server 61
Network Addresses and Ports 61
qmgr.. 62
Default Configuration 66
Server Attributes.. 66
Queue Attributes.. 72
Nodes 77
Node Attributes... 81
SGI Weightless CPU Support................................ 86
Job Attributes.. 86
PBS Resources.. 86
Advanced Configuration Options 89
8 Configuring MOM 93
MOM Config File 93
Job Memory Limit Enforcement.......................... 101
SGI Non-cpuset Memory Enforcement 101
Job NCPUS Limit Enforcement 102
Enhanced SGI “cpusets” Support 104
Static Resources... 106
Dynamic Resources 107
Idle Workstation Cycle Harvesting...................... 108
MOM Globus Configuration 110
Examples... 110
9 Configuring the Scheduler 113
Default Configuration 113
New Scheduler Features 115
Tunable Parameters.. 117
Dynamic Consumable Resources 133
10 Example Configurations.......................................137
Single Node Time-sharing System 138
Single Timesharing Node with Separate Server .. 139
Multi-node Timesharing Cluster.......................... 140
Multi-node Space-sharing Cluster 142

11 Administration..143
/etc/pbs.conf.. 143
Starting PBS Daemons.. 144
Starting and Stopping PBS: Windows 2000 / XP 151
Stopping PBS.. 151
Checkpoint/Restart Under PBS 152
Start/Stop/Enable/Disable Queues....................... 154
Security 155
Internal Security.. 155
External Security... 158
Root Owned Jobs 159
Managing PBS and Multi-node Parallel Jobs...... 159
SGI Job Container / Limits Support 160
Job Prologue/Epilogue Scripts/Programs 160
Use and Maintenance of Logfiles 163
Using the syslog facility with PBS Pro................ 171
Interpreting PBS Exit Codes................................ 171
PBS tracejob Command...................................... 172
pbsnodes Command.. 173
PBS Graphical Interfaces (GUIs) 174
Using Job Comments 178
pbsfs (PBS Fairshare) command.......................... 178
PBS Pro on Scyld Beowulf Clusters.................... 179
12 Problem Solving 181
Clients Unable to Contact Server......................... 181
Nodes Down 182
Requeueing a Job "Stuck" on a Down Node....... 183
Non Delivery of Output 183
Job Cannot be Executed...................................... 184
Handling Jobs on Failed Nodes 184
Running Jobs with No Active Processes 185
Getting Help.. 185
13 Customizing PBS..187
Shell Invocation 187
TCL/tk Interface to PBS API............................... 189
Additional Build Options.................................... 189
Site Modifiable Source Files............................... 192
Implementing a Custom Scheduler...................... 194
14 Appendix A: Error Codes 201
15 Appendix B: Request Codes...................................207
16 Index 211

Acknowledgements

PBS Pro is the enhanced commercial version of the PBS software originally developed for NASA. The NASA version had a number of corporate and individual contributors over the years, for which the PBS developers and PBS community are most grateful. Below we provide formal legal acknowledgements to corporate and government entities, then special thanks to individuals.

The NASA version of PBS contained software developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and MRJ Technology Solutions. In addition, it included software developed by the NetBSD Foundation, Inc., and its contributors, as well as software developed by the University of California, Berkeley and its contributors.

Other contributors to the NASA version of PBS include Bruce Kelly and Clark Streeter of NERSC; Kent Crispin and Terry Heidelberg of LLNL; John Kochmar and Rob Pennington of *Pittsburgh Supercomputing Center*; and Dirk Grunwald of *University of Colorado, Boulder*. The ports of PBS to the Cray T3e and the IBM SP SMP were funded by *DoD USAERDC*, Major Shared Research Center; the port of PBS to the Cray SV1 was funded by DoD MSIC.

No list of acknowledgements for PBS would possibly be complete without special recognition of the first two beta test sites. Thomas Milliman of the *Space Sciences Center* of the *University of New Hampshire* was the first beta tester. Wendy Lin of *Purdue University* was the second beta tester and holds the honor of submitting more problem reports than anyone else outside of NASA.

Preface

Intended Audience

This document provides the system administrator with the information required to install, configure, and manage the Portable Batch System (PBS). PBS is a workload management system that provides a unified batch queuing and job management interface to a set of computing resources.

Related Documents

The following publications contain information that may also be useful in the management and administration of PBS.

PBS-3BU01	**PBS Pro User Guide**: Provides an overview of PBS Pro and serves as an introduction to the software, explaining how to use the user commands and graphical user interface to submit, monitor, track, delete, and manipulate jobs.
PBS-3BE01	**PBS External Reference Specification**: Discusses in detail the PBS application programming interface (API), security within PBS, and intra-daemon communication.

Ordering Software and Publications

To order additional copies of this manual and other PBS publications, or to purchase additional software licenses, contact your reseller or the PBS Products Department. Contact information is included on the copyright page of this document.

Document Conventions

PBS documentation uses the following typographic conventions.

abbreviation	If a PBS command can be abbreviated (such as subcommands to `qmgr`) the shortest acceptable abbreviation is underlined.
`command`	This fixed width font is used to denote literal commands, filenames, error messages, and program output.
input	Literal user input is shown in this bold, fixed-width font.
`manpage`(x)	Following UNIX tradition, manual page references include the corresponding section number in parentheses appended to the manual page name.
terms	Words or terms being defined, as well as variable names, are in italics.

Chapter 1
Introduction

This book, the **Administrator Guide** to the Portable Batch System, Professional Edition (PBS Pro) is intended as your knowledgeable companion to the PBS Pro software. This edition pertains to PBS Pro in general, with specific information for version 5.3. It covers both the standard binary distribution, as well as the optional source code distribution.

1.1 Book Organization

This book is organized into 13 chapters, plus an appendix. Depending on your intended use of PBS, some chapters will be critical to you, and others can be safely skipped.

Chapter 1	**Introduction**: Gives an overview of this book, PBS, and the PBS team.
Chapter 2	**Concepts and Terms**: Discusses the components of PBS and how they interact, followed by definitions of terms used in PBS.
Chapter 3	**Pre-Installation Planning**: Helps the reader plan for a new installation of PBS.
Chapter 4	**Installation**: Covers the installation of the binary distribution of PBS and software licenses.

Chapter 5	**Installation from Source**: Covers the installation of PBS from source code. This chapter can be skipped if you are installing the binary distribution.
Chapter 6	**Upgrading PBS**: Provides important information for sites that are upgrading from a previous version of PBS.
Chapter 7	**Configuring the Server**: Describes how to configure the PBS Server daemon.
Chapter 8	**Configuring MOM**: Describes how to configure the PBS MOM daemons.
Chapter 9	**Configuring the Scheduler**: Describes how to configure the PBS Scheduler daemon.
Chapter 10	**Example Configurations**: Provides examples and sample configurations.
Chapter 11	**Administration**: Discusses PBS management and administration.
Chapter 12	**Problem Solving**: Discusses how to trouble-shoot PBS problems, and describes the tools provided by PBS to assist with problem solving.
Chapter 13	**Customizing PBS**: Provides information on customizing PBS. This chapter can be skipped by most sites.
Appendix A	**Error Codes**: Provides a listing and description of the PBS error codes.
Appendix B	**Requesst Codes**: Provides a listing and description of the PBS Request codes.

1.2 What is PBS Pro?

PBS Pro is the professional version of the Portable Batch System (PBS), a flexible resource and workload management system, originally developed to manage aerospace computing resources at NASA. PBS has since become the leader in supercomputer workload management and the *de facto* standard on Linux clusters.

Today, growing enterprises often support hundreds of users running thousands of jobs across different types of machines in different geographical locations. In this distributed heterogeneous environment, it can be extremely difficult for administrators to collect detailed, accurate usage data or to set system-wide resource priorities. As a result, many computing resource are left under-utilized, while others are over-utilized. At the same time, users are confronted with an ever expanding array of operating systems and platforms. Each year, scientists, engineers, designers, and analysts must waste countless hours learning the nuances of different computing environments, rather than being able to focus on their core priorities. PBS Pro addresses these problems for computing-intensive enterprises such as science, engineering, finance, and entertainment.

Now you can use the power of PBS Pro to take better control of your computing resources. This product enables you to unlock the potential in the valuable assets you already have. By reducing dependency on system administrators and operators, you will free them to focus on other actives. PBS Pro can also help you to efficiently manage growth by tracking real usage levels across your systems and by enhancing effective utilization of future purchases.

1.2.1 History of PBS

In the past, UNIX systems were used in a completely interactive manner. Background jobs were just processes with their input disconnected from the terminal. However, as UNIX moved onto larger and larger processors, the need to be able to schedule tasks based on available resources increased in importance. The advent of networked compute servers, smaller general systems, and workstations led to the requirement of a networked batch scheduling capability. The first such UNIX-based system was the Network Queueing System (NQS) from NASA Ames Research Center in 1986. NQS quickly became the *de facto* standard for batch queueing.

Over time, distributed parallel systems began to emerge, and NQS was inadequate to handle the complex scheduling requirements presented by such systems. In addition, computer system managers wanted greater control over their compute resources, and users wanted a single interface to the systems. In the early 1990's NASA needed a solution to this problem, but found nothing on the market that adequately addressed their needs. So NASA lead an international effort to gather requirements for a next-generation resource management system. The requirements and functional specification were later adopted as an IEEE POSIX standard (1003.2d). Next, NASA funded the development of a new resource management system compliant with the standard. Thus the Portable Batch System (PBS) was born.

PBS was quickly adopted on distributed parallel systems and replaced NQS on traditional supercomputers and server systems. Eventually the entire industry evolved toward distributed parallel systems, taking the form of both special purpose and commodity clusters. Managers of such systems found that the capabilities of PBS mapped well onto cluster computers.

The next chapter in the PBS story began when Veridian (the R&D contractor that developed PBS for NASA) released the Portable Batch System Professional Edition (PBS Pro), a complete workload management solution.

1.3 About the PBS Team

The PBS Pro product is brought to you by the same team that originally developed PBS for NASA. In addition to the core engineering team, the PBS Products department includes individuals who have supported PBS on computers all around the world, including the largest supercomputers in existence. The staff includes internationally-recognized experts in resource- and job-scheduling, supercomputer optimization, message-passing programming, parallel computation, and distributed high-performance computing.

In addition, the PBS team includes co-architects of the NASA Metacenter (the first full-production geographically distributed meta-computing environment), co-architects of the Department of Defense MetaQueueing Project, co-architects of the NASA Information Power Grid, and co-chair of the Global Grid Forum’s Scheduling Group.

Chapter 2
Concepts and Terms

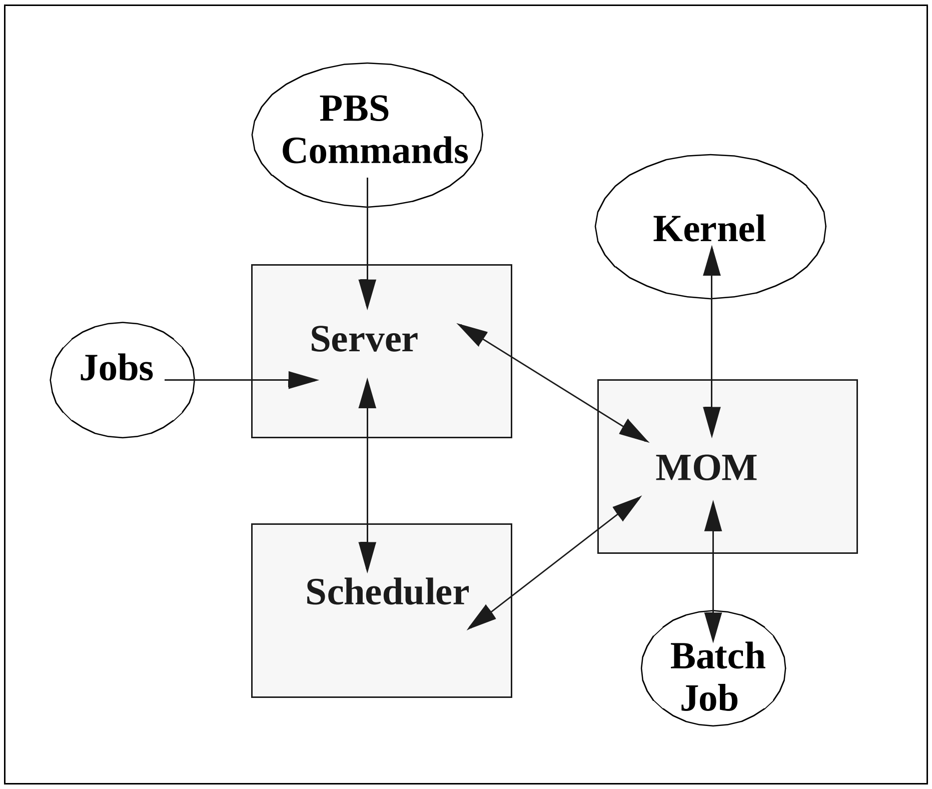
PBS is a distributed workload management system. As such, PBS handles the management and monitoring of the computational workload on a set of one or more computers. Modern workload/resource management solutions like PBS include the features of traditional batch queueing but offer greater flexibility and control than first generation batch systems (such as the original batch system NQS).

Workload management systems have three primary roles:

Queuing	The collecting together of work or tasks to be run on a computer. Users submit tasks or “jobs” to the resource management system where they are queued up until the system is ready to run them.
Scheduling	The process of selecting which jobs to run when and where, according to a predetermined policy. Sites balance competing needs and goals on the system(s) to maximize efficient use of resources (both computer time and people time).
Monitoring	The act of tracking and reserving system resources and enforcing usage policy. This covers both user-level and system-level monitoring as well as monitoring of the scheduling algorithms to see how well they are meeting the stated goals

2.1 PBS Components

PBS consist of two major component types: user-level commands and system daemons. A brief description of each is given here to help you make decisions during the installation process.



Commands PBS supplies both command line programs that are POSIX 1003.2d conforming and a graphical interface. These are used to submit, monitor, modify, and delete jobs. These *client commands* can be installed on any system type supported by PBS and do not require the local presence of any of the other components of PBS.

There are three classifications of commands: user commands (which any authorized user can use), operator commands, and Manager (or administrator) commands. Operator and Manager commands require specific access privileges, as discussed in section 11.7 “Security” on page 155.

Job Server

The *Job Server* daemon is the central focus for PBS. Within this document, it is generally referred to as *the Server* or by the execution name *pbs_server.* All commands and daemons communicate with the Server via an *Internet Protocol* (IP) network. The Server's main function is to provide the basic batch services such as receiving/creating a batch job, modifying the job, protecting the job against system crashes, and running the job. Typically there is one Server managing a given set of resources.

Job Executor (MOM)

The *Job Executor* is the daemon that actually places the job into execution. This daemon, *pbs_mom*, is informally called *MOM* as it is the mother of all executing jobs. (MOM is a reverse-engineered acronym that stands for Machine Oriented Miniserver.) MOM places a job into execution when it receives a copy of the job from a Server. MOM creates a new session that is as identical to a user login session as is possible. For example, if the user's login shell is csh, then MOM creates a session in which `.login` is run as well as `.cshrc`. MOM also has the responsibility for returning the job's output to the user when directed to do so by the Server. One MOM daemon runs on each computer which will execute PBS jobs.

A special version of MOM, called the *Globus MOM*, is available if it is enabled during the installation of PBS. It handles submission of jobs to the Globus environment. Globus is a software infrastructure that integrates geographically distributed computational and information resources. Globus is discussed in more detail in the "Globus Support" section of the **PBS User Guide**.

Job Scheduler

The *Job Scheduler* daemon, *pbs_sched*, implements the site's policy controlling when each job is run and on which resources. The Scheduler communicates with the various MOMs to query the state of system resources and with the Server to learn about the availability of jobs to execute. The interface to the Server is through the same API as used by the client commands. Note that the Scheduler interfaces with the Server with the same privilege as the PBS Manager.

2.2 Defining PBS Terms

The following section defines important terms and concepts of PBS. The reader should review these definitions before beginning the planning process prior to installation of PBS. The terms are defined in an order that best allows the definitions to build on previous terms.

Node	A *node* to PBS is a computer system with a single *operating system* (OS) image, a unified virtual memory space, one or more CPUs and one or more IP addresses. Frequently, the term *execution host* is used for node. A computer such as the SGI Origin 3000, which contains multiple CPUs running under a single OS, is one node. Systems like the IBM SP and Linux clusters, which contain separate computational units each with their own OS, are collections of nodes. Nodes can be defined as either *cluster nodes* or *timeshared nodes*, as discussed below.
Nodes & Virtual Processors	A node may be declared to consist of one or more *virtual processors (VPs)*. The term virtual is used because the number of VPs declared does not have to equal the number of real processors on the physical node. The default number of virtual processors on a node is the number of currently functioning physical processors; the PBS Manager can change the number of VPs as required by local policy.
Cluster Node	A node whose purpose is geared toward running parallel jobs is called a *cluster node*. If a cluster node has more than one virtual processor, the VPs may be assigned to different jobs (*job-shared*) or used to satisfy the requirements of a single job (*exclusive*). This ability to temporally allocate the entire node to the exclusive use of a single job is important for some multi-node parallel applications. Note that PBS enforces a one-to-one allocation scheme of cluster node VPs ensuring that the VPs are not over-allocated or over-subscribed between multiple jobs.
Timeshared Node	In contrast to cluster nodes are hosts that **always** service multiple jobs simultaneously, called *timeshared nodes*. Often the term *host* rather than node is used in conjunction with time-shared, as in *timeshared host.* A timeshared node will never be allocated exclusively or temporarily-shared. However, unlike cluster nodes, a timeshared node **can** be over-committed if the local policy specifies to do so.

Cluster This is any collection of nodes controlled by a single instance of PBS (i.e., by one PBS Server).

Exclusive VP An exclusive VP is one that is used by one and only one job at a time. A set of VPs is assigned exclusively to a job for the duration of that job. This is typically done to improve the performance of message-passing programs.

Temporarily-shared VP A *temporarily-shared node* is one where one or more of its VPs are temporarily shared by jobs. If several jobs request multiple temporarily-shared nodes, some VPs may be allocated commonly to both jobs and some may be unique to one of the jobs. When a VP is allocated on a temporarily-shared basis, it remains so until all jobs using it are terminated. Then the VP may be re-allocated, either again for temporarily-shared use or for exclusive use.

If a host is defined as timeshared, it will never be allocated exclusively or temporarily-shared.

Load Balance A policy wherein jobs are distributed across multiple timeshared hosts to even out the workload on each host. Being a policy, the distribution of jobs across execution hosts is solely a function of the Job Scheduler.

Queue A *queue* is a named container for jobs within a Server. There are two types of queues defined by PBS, *routing* and *execution*. A *routing queue* is a queue used to move jobs to other queues including those that exist on different PBS Servers. Routing queues are similar to the old NQS pipe queues. A job must reside in an *execution queue* to be eligible to run and remains in an execution queue during the time it is running. In spite of the name, jobs in a queue need not be processed in queue order (first-come first-served or *FIFO*).

Node Attribute Nodes have attributes associated with them that provide control information. The attributes defined for nodes are: state, type (ntype), the list of jobs to which the node is allocated, properties, max_running, max_user_run, max_group_run, and both assigned and available resources (“resources_assigned” and “resources_available”).

Node Property A set of zero or more *properties* may be given to each node in order to have a means of grouping nodes for allocation. The property is nothing more than a string of alphanumeric characters (first charac-

	ter must be alphabetic) without meaning to PBS. The PBS administrator may assign to nodes whatever property names desired. Your choices for property names should be relayed to the users.
Portable Batch System	PBS consists of one Job Server (pbs_server), one or more Job Scheduler (pbs_sched), and one or more execution servers (pbs_mom). The PBS System can be set up to distribute the workload to one large timeshared system, multiple time shared systems, a cluster of nodes to be used exclusively or temporarily-shared, or any combination of these.

The remainder of this chapter provides additional terms, listed in alphabetical order.

Account	An *account* is arbitrary character string, which may have meaning to one or more hosts in the batch system. Frequently, account is used as a grouping for charging for the use of resources.
Administrator	See Manager.
API	PBS provides an *Application Programming Interface (API)* which is used by the commands to communicate with the Server. This API is described in the **PBS External Reference Specification**. A site may make use of the API to implement new commands if so desired.
Attribute	An *attribute* is an inherent characteristic of a parent object (Server, queue, job, or node). Typically, this is a data item whose value affects the operation or behavior of the object and can be set by the owner of the object. For example, the user can supply values for attributes of a job.
Batch or Batch Processing	This refers to the capability of running jobs outside of the interactive login environment.
Complex	A *complex* is a collection of hosts managed by one batch system. It may be made up of nodes that are allocated to only one job at a time or of nodes that have many jobs executing at once on each node or a combination of these two scenarios.
Destination	This is the location within PBS where a job is sent for processing. A destination may uniquely define a single queue at a single Server or it may map into many locations.

Destination Identifier	This is a string that names the destination. It is composed two parts and has the format queue@server where server is the name of a PBS Server and queue is the string identifying a queue on that Server.
File Staging	*File staging* is the movement of files between a specified location and the execution host. See “Stage In” and “Stage Out” below.
Group ID (GID)	This numeric identifier is uniquely assigned to each group (see Group).
Group	*Group* refers to collection of system users (see Users). A user must be a member of a group and may be a member of more than one. Within UNIX and POSIX systems, membership in a group establishes one level of privilege. Group membership is also often used to control or limit access to system resources.
Hold	An artificial restriction which prevents a job from being selected for processing. There are three types of holds. One is applied by the job owner, another is applied by the operator or administrator, and a third applied by the system itself or the PBS administrator.
Job or Batch Job	The basic execution object managed by the batch subsystem. A job is a collection of related processes which is managed as a whole. A job can often be thought of as a shell script running in a POSIX session. (A session is a process group the member processes cannot leave.) A non-singleton job consists of multiple tasks of which each is a POSIX session. One *task* will run the job shell script.
Manager	The *manager* is the person authorized to use all restricted capabilities of PBS. The Manager may act upon the Server, queues, or jobs. The Manager is also called the administrator.
Operator	A person authorized to use some but not all of the restricted capabilities of PBS is an *operator*.
Owner	The owner is the user who submitted the job to PBS.
POSIX	This acronym refers to the various standards developed by the “Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society” under standard P1003.

Rerunable	If a PBS job can be terminated and its execution restarted from the beginning without harmful side effects, the job is rerunable.
Stage In	This process refers to moving a file or files to the execution host prior to the PBS job beginning execution.
Stage Out	This process refers to moving a file or files off of the execution host after the PBS job completes execution.
User	Each system *user* is identified by a unique character string (the user name) and by a unique number (the user id).
Task	*Task* is a POSIX session started by MOM on behalf of a job.
User ID (UID)	Privilege to access system resources and services is typically established by the *user id*, which is a numeric identifier uniquely assigned to each user (see User).
Virtual Processor (VP)	See Cluster Node.

Chapter 3
Pre-Installation Planning

This chapter presents two sets of information needed prior to installing PBS. First, a reference to new features in this release of PBS Pro is provided. Next is the information necessary to make certain planning decisions.

3.1 New Features in PBS Pro 5.3

The *Release Notes* included with this release of PBS Pro lists new features in this version of PBS Pro, information on upgrading, and any warning or caveats. Be sure to review the release notes, as they may contain information that was not available when this book went to press.

Admin	Support for checkpoint-restart of SGI jobs running within IRIX cpusets. See “Irix OS-Leveling Checkpoint With cpusets” on page 104.
Admin	Support for running multiple jobs with a single IRIX cpuset. See “IRIX Shared cpusets for Small Jobs” on page 104.
Admin	Improved DCE Support. See “Compiling in DCE Support” on page 50.

Scheduler	Enhancements to “Cycle Harvesting” of otherwise unused workstations. See “Idle Workstation Cycle Harvesting” on page 108.
Scheduler	Multiple sorts for choosing between available nodes. See “Job Sorting by Resource” on page 115.

3.2 Server - MOM Protocol Change

The Server - MOM protocol has changed in release 5.3. Therefore a 5.3 Server cannot communicate with a MOM from a prior version, nor can a 5.3 MOM communicate with a Scheduler of an earlier version.

3.3 Changes to Time-shared and Cluster Nodes

For planning purposes, note that PBS Pro 5.1 reduced the differences between time-shared and cluster nodes to:

1. Time-shared nodes are first choice for jobs that do not have a node specification.
2. Time-shared nodes may not be requested for exclusive use with the `#excl` suffix.
3. More processes than CPUs can be run on time-shared nodes but not on cluster nodes.
4. If load balancing by "load average" is activated in the Job Scheduler, it applies only to time-shared nodes.

Allocation of cluster nodes remains based on the number of (virtual) processors.

3.4 Planning

PBS is able to support a wide range of configurations. It may be installed and used to control jobs on a single system or to load balance jobs on a number of systems. It may be used to allocate nodes of a cluster or parallel system to both parallel and serial jobs. It can also deal with a mix of these situations.

3.4.1 Single Timesharing System

If PBS is to be installed on a time-sharing system, all three daemons may reside on that system; or you may place the Server (`pbs_server`) and/or the Scheduler (`pbs_sched`) on a “front end” system. MOM (`pbs_mom`) must run on the system where jobs are to be executed.

3.4.2 Timesharing Cluster

If PBS is to be installed on a collection of time-sharing systems, a MOM must be on each execution host and the Server and Scheduler may be installed on one of the systems or on a front end system.

3.4.3 Large SMP systems

For large SMP systems (e.g. SGI Origin or Cray vector systems) it is best to plan on treating these as “time-shared” systems rather than “cluster nodes”, in order to maximize the scheduling efficiency on these systems.

3.4.4 Running on Scyld Beowulf Clusters

If installing PBS on a Scyld Beowulf cluster, note that the cluster is controlled by a central "master node" which will run a single MOM. There is no need to have PBS installed on any of the compute nodes of a Scyld Beowulf cluster.

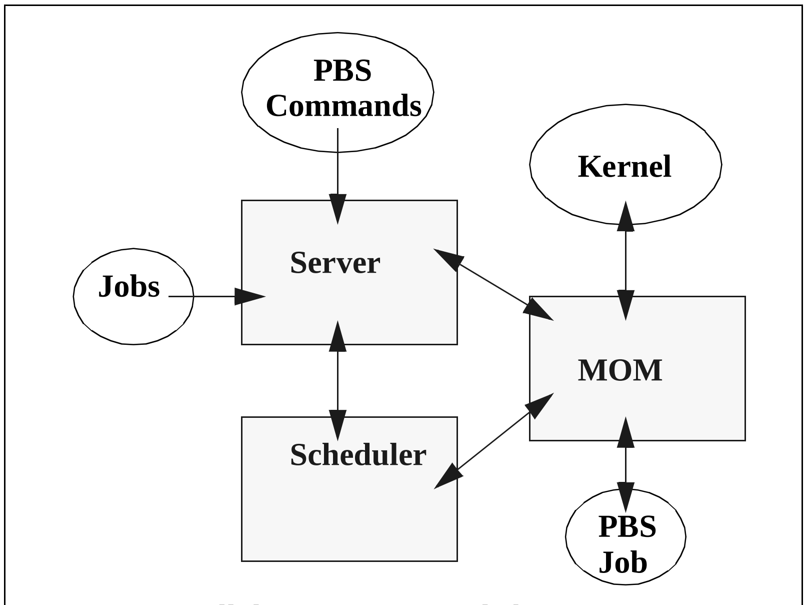
3.5 Interfacing with Globus

If Globus support is enabled, then a separate `pbs_mom_globus` must be run. In order to forward a job to a globus site, `pbs_mom_globus` will read user's grid credential on the host where it is running. Thus, `pbs_mom_globus` should run on the host from which users submit their PBS Globus jobs. Globus-specific configuration information is given throughout the PBS documentation. Specifically, the following sections should be reviewed by any site wishing to deploy PBS with Globus support enabled.

“Globus Toolkit” on page 30
“Server Support for Globus” on page 91
“MOM Globus Configuration” on page 110
“Manually Starting Globus MOM” on page 150
“Globus Support” in the **PBS User Guide**.

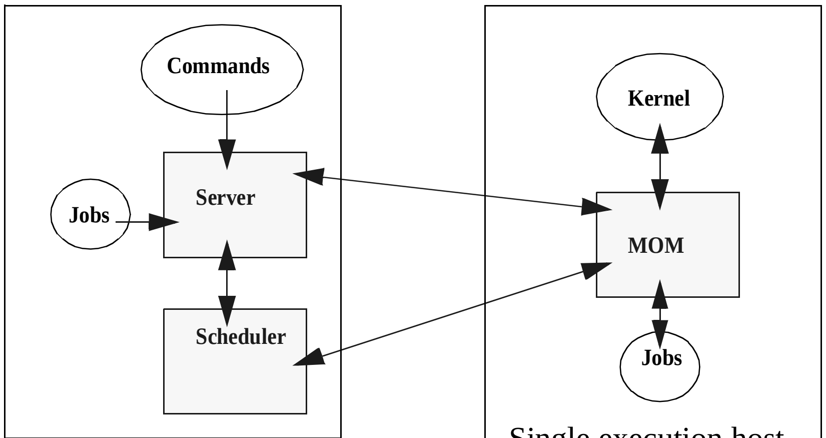
3.6 Single Execution System

If you are installing PBS on a single system, you are ready to install, configure the daemons, and select your scheduling policy.



All daemons on a single host.

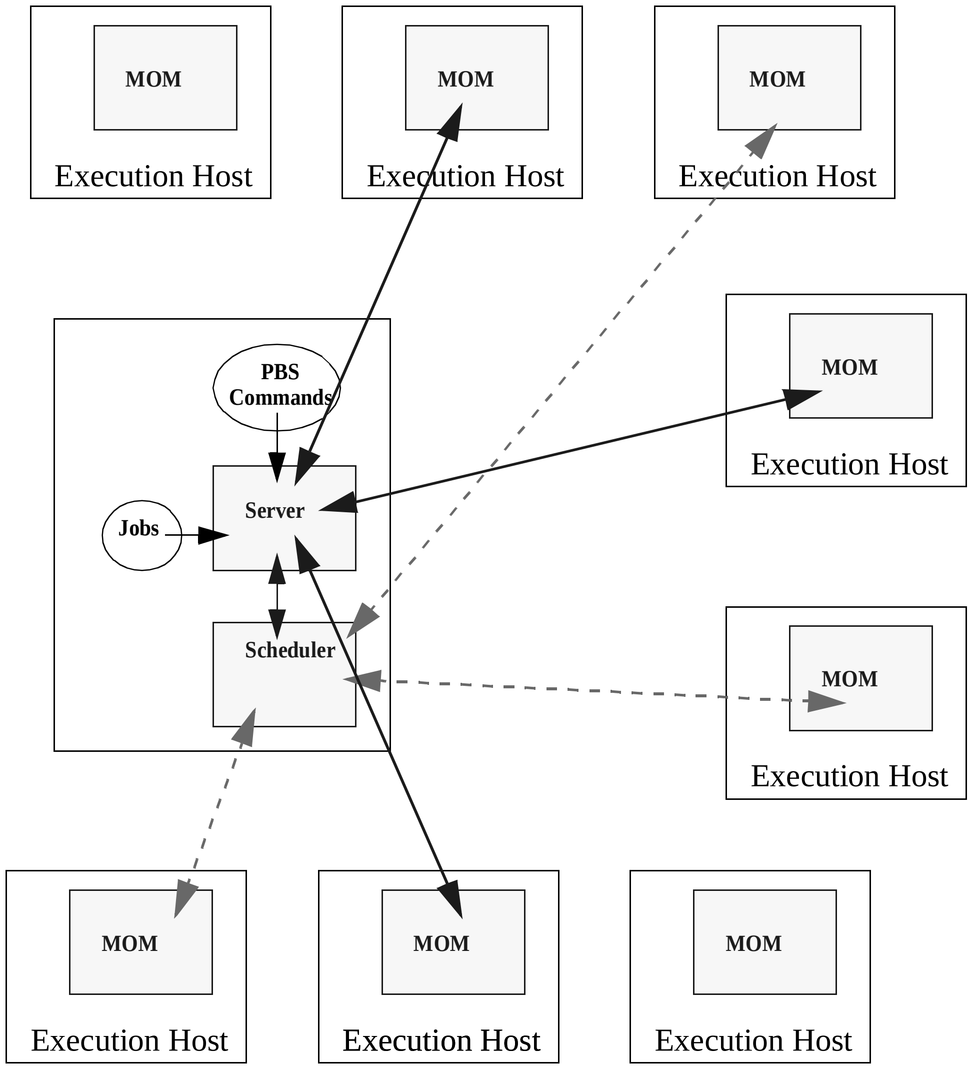
If you wish, the PBS Server and Scheduler, `pbs_server` and `pbs_sched`, can run on one system and jobs can execute on another. This is a trivial case of multiple execution systems discussed in the next section.



Single execution host.

3.7 Multiple Execution Systems

If you are planning to run PBS on more than one computer, you will need to install the execution daemon (`pbs_mom`) on each system where jobs are expected to execute. The following diagram illustrates this for an eight node cluster.



Chapter 4
Installation

This chapter discusses the installation procedures for the PBS binary distribution package. This is the normal installation method for most sites. However, if you intend to install from the source code package, you should skip this chapter and read Chapter 5 “Installation from Source” on page 29.

Important: Be sure to read the Release Notes included in the PBS Pro distribution, as it will contain any information that was unavailable when this book went to press. The Release Notes also contain a detailed description of all new features in a given release.

4.1 Overview

The PBS software can be installed from the PBS CD-ROM or downloaded from the PBS website. The installation procedure is slightly different depending on the distribution source. However, the basic steps of PBS installation are:

Step 1	Prepare distribution media
Step 2	Extract and install the software
Step 3	Acquire a PBS license
Step 4	Install the license

4.2 Media Setup

If installing from the PBS CD-ROM, follow these instructions, with superuser (root) privilege to set up the media for installation:

Step 1 Insert the PBS CD into the system CD-ROM drive

Step 2 If your systems doesn't run an automounter daemon for the CD-ROM drive, then you'll need to manually mount the CD-ROM onto the system:

```
# mount /cdrom
```

Step 3 Change directory to the mount point

```
# cd /cdrom
```

If not installing from CD-ROM, follow these instructions:

Step 1 Download the distribution file from the PBS website. (Follow the instructions you received with your order confirmation.)

Step 2 Move the distribution file to `/tmp` on the system on which you intend to install PBS,

then, as superuser:

Step 3 Create a temporary location under `/tmp` from which to install the distribution

Step 4 Change directory to the temporary installation directory

Step 5 Uncompress the distribution file

Step 6 Extract the distribution file

```
# mkdir /tmp/pbs_tmp
# cd /tmp/pbs_tmp
# gunzip /tmp/pbspro_5.3.0-arch.tar.gz
# tar -xvf /tmp/pbspro_5.3.0-arch.tar
# cd PBSPro_5.3.0
```

4.3 Default Install Options

The installation program installs the various PBS components into specific locations on the system. The installation program allows you to override these default locations if you wish. (Note that some operating systems' software installation programs do not permit software relocation, and thus you are not able to override the defaults on those systems.) The locations are written to the `/etc/pbs.conf` file created by the installation process. For details see the description of "/etc/pbs.conf" on page 143.

4.4 Installation on UNIX/Linux Systems

For a given system, the PBS install script uses the native package installer provided with that system. This means that the PBS package should install into what is considered the "normal" location for third-party software. The following examples shows a typical installation under the Sun Solaris operating system. The process is very similar for other operating systems, but may vary depending on the native package installer on each system.

```
# ./INSTALL
Installation of PBS

The following directory will be the root of the
installation. Several subdirectories will be created
if they don't already exist: bin, sbin, lib, man and
include.
Execution directory? [/opt/pbs]

PBS needs to have a private directory (referred to as
"PBS_HOME" in the documentation) where it can permanently
store information.
Home directory? [/usr/spool/PBS]
/usr/spool/PBS does not exist, I'll make it.
done

[ Description of the different configuration options ]

PBS Installation:
        1. Server, execution and commands
        2. Execution only
        3. Commands only
(1|2|3)?
```

You need to decide what kind of PBS installation you want for each machine in your cluster. There are three possibilities: a Server node, an execution node, or a client host. If you are going to run PBS on a single timesharing host, install the full Server package (option 1). If you are going to have a cluster of machines, you need to pick one to be the front end and install the Server package (option 1) there. Then, install the execution package (option 2) on all the compute nodes in the cluster. The client package (option 3) is for hosts which will not be used for execution but need to have access to PBS. It contains the commands, the GUIs and man pages. This gives the ability to submit jobs and check status.

The following sections illustrate the differences between installation on a single server system versus a cluster of workstations.

4.4.1 Installation on a Single UNIX Server

For the following examples, we will assume that you are installing PBS on a single large Server/execution host, on which all the daemons will run, and from which users will submit jobs. Example of such a system might be an SMP system such as an SGI Origin3000 or a Cray T90.

To achieve this, we select option **1** to the question shown in the example above, followed by “all” when asked which packages to add, as shown:

```
(1|2|3)? 1

Installing PBS for a Server Host.
The following packages are available:
  1  pbs64     pbs64    (sparc) 5.0

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: all

Veridian Systems PBS department
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

Next the installation program will ask you to confirm that it is acceptable to install setuid/setgid programs as well as to run installation sub-programs as root. You should answer yes (or “**y**”) to both of these questions, as shown below.

```
## Checking for setuid/setgid programs.

The following files are being installed with setuid and/or
setgid permissions:
  /opt/pbs/sbin/pbs_iff <setuid root>
  /opt/pbs/sbin/pbs_rcp <setuid root>

Do you want to install these as setuid/setgid files
[y,n,?,q] y

This package contains scripts which will be executed with
super-user permission during the process of installing
this package.

Do you want to continue with the installation of <pbs64>
[y,n,?] y
```

Next, the installation program will proceed to extract and install the PBS package(s) that you selected above. The process should look similar to the example below.

```
Installing pbs64 as <pbs64>

## Installing part 1 of 1.
/etc/init.d/pbs
[ listing of files not shown for brevity ]

## Executing postinstall script.
*** PBS Installation Summary
***
*** The PBS Server has been installed in /opt/pbs/sbin.
***
*** The PBS commands have been installed in /opt/pbs/bin.
***
*** This host has the PBS Server installed, so
*** the PBS commands will use the local server.
*** The PBS command server host is mars
***
*** PBS Mom has been installed in /opt/pbs/sbin.
***
*** The PBS Scheduler has been installed in /opt/pbs/sbin.
***
Installation of <pbs64> was successful.
```

4.4.2 Installing MOM with SGI "cpuset" Support

PBS Pro for SGI IRIX systems provides optional (site-selectable) support for IRIX "cpusets". A cpuset is a named region of the SGI system which contains a specific set of CPUs and associated memory. PBS has the ability to use the cpuset feature to "fence" PBS jobs into their own cpuset. This helps to prevent different jobs from interfering with each other. To enable use of this feature, a different PBS MOM binary needs to be installed, as follows:

```
# /etc/init.d/pbs stop
# cd /usr/pbs/sbin
# rm pbs_mom
# ln -s pbs_mom.cpuset pbs_mom
```

Additional information on configuring and using IRIX cpusets is discussed later in this manual. For scheduler configuration details, see section 9.3.1 "Scheduler Support for SGI IRIX cpusets" on page 124.

4.5 Installation on Windows 2000 Systems

When PBS is installed on a cluster, a MOM daemon must be on each execution host, and the Server and Scheduler should be installed on one of the systems or on a front-end system.

For Windows 2000 clusters, PBS is provided in a single package that contains:

> the **PBS Administrator Guide** in PDF form,
> the **PBS User Guide** in PDF form,
> the PBS Pro software, and
> supporting text files (software license, README, release notes, etc.)

The PBS Pro `install` program will walk you through the installation process. If you are installing from the PBS Pro CD-ROM, insert the CD-ROM into your computer's CD-ROM drive, browse to your CD-ROM drive, and double-click on the Install program icon.

Alternatively, you can download the latest PBS Pro package from the PBS Pro Web site, and save it to your hard drive. From there you can manually run the self-extracting `pbspro.exe` package, and then the installation program, as shown below.

```
C:\> PBSpro_5_3_0-windows.exe
```

Important: Note that you must be logged in as Administrator to run the PBS installation program.

The installation program will prompt you for the names of directories for the different parts of PBS and the type of installation (full, server-only, execution host only). Next, you will be prompted for your software license key(s), as discussed in the following section. Following this, the install program will prompt to set up the nodes file, the hosts.equiv file, and the MOM config file. If unsure, you can take the defaults offered. You can always edit these files after the installation is complete, as described in the various configuration chapters in this manual

4.6 Installing the PBS License

The PBS license manager can handle multiple license keys in the PBS license file. This is useful when you expand your PBS configuration: you can simply add the additional licenses. This section describes adding a single license (such as following an initial installation of PBS Pro). The next section discusses adding multiple licenses.

Note that when requesting or generating your license key(s), the number of CPUs field should correspond with the total number of actual CPUs on which you wish to run PBS jobs.

When the installation of PBS is complete, you will need to install your PBS license key. If you already have your PBS license key, type it in when prompted by the license installation program, as shown below.

.

```
PBS license installation

Using /usr/spool/PBS as PBS_HOME
To get a license, please visit
        www.pbspro.com/license.html
 or call PBSpro toll free at 877-905-4PBS
 and have the following information handy:

***     host name:      mars.pbspro.com
***     host id:        12927f28
***     site id from the PBSPro package
***     number of cpus you purchased

Please enter the license string(s) (^d to end).
? 5-00020-99999-0044-PfV/fjuivg-5Jz

Installing: 5-00020-99999-0044-PfV/fjuivg-5Jz

Please enter the next license string(s) (^d to end).
?
Would you like to start PBS now (y|[n])? n
To start PBS, type '/etc/init.d/pbs start'
#
```

At this point, you can probably safely skip forward to Chapter 7 “Configuring the Server” on page 61.

However, if you have not yet received your PBS license key, follow the instructions printed by the installation program (see example above) to receive your key. Then rerun the PBS license key installation program as root:

```
# /usr/pbs/etc/pbs_setlicense
```

4.7 Using Floating Licenses

New in release 5.2, PBS can be purchased with *floating licenses*. With floating licenses, you may have more CPUs configured online than the number of CPUs licensed. Nodes become licensed as needed to run jobs up to the number of floating licenses. The licenses are released from a node when no jobs remain running on the node.

The Server attribute "Licenses" shows the number of floating licenses currently available.

A node attribute "pcpus" show the number of physical CPUs on the node, which determines the number of licenses required for that node. Another node attribute "license" shows the node "license state":

`u`	unlicensed
`l`	licensed with a node-lock or fixed license
`f`	licensed with a floating license

4.8 Installing Multiple PBS Licenses

As mentioned earlier, it is possible to add multiple licenses to the PBS License file. This can be done during installation of PBS Pro, or at some future time. If the installation program detects that you already have a PBS license file, it will prompt you as to what you want done: keep the file, replace the file, or append to it. Specify the option that corresponds to the action you wish to take.

Then, if you have multiple license keys, simply type them in when prompted by the license installation program, as shown in the example below.

Note that you can invoke the license key installation program directly (as may be needed following an increase in the size of the system or cluster managed by PBS), using the “-a” (append) option:

```
# /usr/pbs/etc/pbs_setlicense -a
```

Alternatively, you can manually edit the licence file, adding the new license as a separate line in the file. Then you will need to shutdown the Server with “`qterm -t quick`” and restart it for the new license to be recognized.

```
PBS license installation

Using /usr/spool/PBS as PBS_HOME
A license already exists in
 /usr/spool/server_priv/license_file.

Would you like to (k)eep it, (r)replace it, (a)ppend to
it, or (q)uit
([k]|a|q)? a

To get a license, please visit
     www.pbspro.com/UserArea/license_list.php
 or call PBSpro toll free at 877-905-4PBS
 and have the following information handy:

***     host name:      mars.pbspro.com
***     host id:        12927f28
***     site id from the PBSPro package
***     number of cpus you purchased

Please enter the license string(s) (^d to end).
? 5-00020-99999-0044-PfV/fjuivg-5Jz

Installing: 5-00020-99999-0044-PfV/fjuivg-5Jz

Please enter the next license string(s) (^d to end).
? 5-00020-99999-0010-XdsXdfssf-5Xj

Installing: 5-00020-99999-0010-XdsXdfssf-5Xj

Please enter the next license string(s) (^d to end).
?

Would you like to start PBS now (y|[n])? n
To start PBS, type '/etc/init.d/pbs start'
#
```

Chapter 5

Installation from Source

This chapter explains how to build and install PBS from the source code distribution. If you are installing from the binary distribution package, you should skip this chapter, and continue with Chapter 7 “Configuring the Server” on page 61.

5.1 Tar File

The PBS source distribution is provided as a single tar file. The tar file contains:

1. This document in both postscript and PDF form.
2. A “configure” script, and all source code, header files, and make files required to build and install PBS.
3. A full set of manual page sources. These are troff input files.

When the PBS tar file is extracted, a subtree of directories is created in which all the above mentioned files are created. The name of the top level directory of this subtree will reflect the release version and patch level of the version of PBS being installed. For example, the directory for PBS Pro 5.3 will be named pbspro_5_3_0.

5.2 Optional Components

To build PBS from source and to include support for optional features, several third party applications are required. This section describes these additional requirements.

5.2.1 POSIX make

PBS uses a `configure` script, generated by GNU's `autoconf` facility, to produce makefiles. If you have a POSIX-compliant `make` program then the makefiles generated by `configure` will try to take advantage of POSIX `make` features. If the `make` program that exists on your system is unable to process the makefiles, try using GNU's `make` program.

5.2.2 tcl/tk

If the Tcl/tk based GUI (`xpbs` and `xpbsmon`) or a Tcl based custom Scheduler is used, the Tcl header files and libraries are required. Versions of Tcl prior to 8.0 cannot be used with PBS. The official site for Tcl is:

```
http://www.scriptics.com/
ftp://ftp.scriptics.com/pub/tcl/tcl8_0
```

5.2.3 Globus Toolkit

If the Globus feature is to be enabled, ensure that Globus clients and libraries are installed. Check the following site for obtaining the Globus software:

```
http://www.globus.org/
```

5.3 Build Steps

To generate a usable PBS installation, the following steps are required. Each step is explained in detail in the subsequent sections of this book.

Step 1 Read this guide and plan a general configuration of hosts and PBS.

Step 2 Decide where the PBS source and objects are to go. See also section 3.4 “Planning” on page 14.

Step 3 Unzip and untar the distribution file into the source tree. See section 5.5 “Overview” on page 32.

Step 4 Change the current working directory to that directory which is to be the top of the object tree. See section 5.5 “Overview” on page 32.

Step 5 Select “configure” options and run `configure` from the top of the object tree. See section 5.6 “Build Details” on page 35.

Step 6 Compile the PBS modules by typing `make` at the top of the object tree. See section 5.7 “Make File Targets” on page 44.

Step 7 As superuser, change directory to the top of the object tree, and install the PBS modules by typing `make install`.

Step 8 As superuser, complete the installation of PBS and set up license key file by typing `make postinstall`.

Step 9 Choose and follow a method for installing MOM. See section 5.9 “Install Options” on page 51.

Details of building PBS are given below. After installation, you can continue with Chapter 7 “Configuring the Server” on page 61 of this manual. If, however, you are planning to use an alternate scheduler you may wish to read section 13.5 “Implementing a Custom Scheduler” on page 194 before continuing with the configuration of PBS.

5.4 Building PBS

This section explains in detail the steps to build and install PBS. In the following descriptions, the *source tree* is the subtree that gets created when the PBS tarfile is extracted. The *target tree* is the subtree of parallel directories in which the object modules are actually compiled. This tree may (and generally should) be separate from the source tree. So it is recommended that you create a separate directory to be the top of the target tree. The target tree is fleshed out with subdirectories and makefiles when `configure` is run.

The PBS build and installation process is accomplished using the make file and subdirectories that got created by running `configure`, a script created for PBS using GNU’s autoconf facility. You should change directory to the top level of the PBS target tree when doing the build and subsequent installation of PBS. This installation procedure requires more manual configuration than is "typical" for many packages. There are a number of options which involve site policy and therefore cannot be determined automagically.

5.5 Overview

The normal PBS build procedure is to separate the source tree from the target tree. This allows the placement of a single copy of the source on a shared file system from which multiple different target systems can be built. Also, the source can be protected from accidental destruction or modification by making the source read-only. However, if you choose, objects may be made within the source tree.

An overview of the "configure", compile, installation and PBS configuration steps is listed here. Detailed explanation of symbols will follow. It is recommended that you read completely through these instructions before beginning the installation. To install PBS:

Step 1 Place the tar file on the system where you would like to maintain the source.

Step 2 Untar the tar file. For example:

```
# cd /usr/local/src
# tar xpf /CDROM/PBSPro_5.2.tar
#
```

It will untar in the current directory producing a single directory named for the current release and patch number. Under that directory will be several files and subdirectories. This directory and the subdirectories make up the *source tree*. You may write-protect the source tree at this point should you so choose.

In the top directory are two files, named `Release_Notes` and `INSTALL`. The `Release_Notes` file contains information about the release contents, changes since the last release and a reference to this guide for installation instructions. The `INSTALL` file consists of standard notes about the use of GNU's configure.

Step 3 If you choose (as recommended) to have separate target and source trees, then create the top level directory of what will become the *target tree* at this time. The target tree must reside on a file system mounted on the same architecture as the target system for which you are generating the PBS binaries. This may be the same system that holds the source or it may not. Change directory to the top of the target tree. For example,

```
cd /usr/local/pbs/obj
```

Step 4 Make a job Scheduler choice. A unique feature of PBS is its external Scheduler module. This allows a site to implement any policy of its choice. To provide even more freedom in implementing policy, PBS provides two Scheduler frameworks. Schedulers may be developed in the C language, or the Tcl scripting language.

As distributed, `configure` will default to a C language based Scheduler known as *Standard*. This Scheduler can be configured to several common scheduling policies. When this Scheduler is installed, certain configuration files are installed in `/usr/spool/PBS/sched_priv/`. You will need to modify these files for your site. These files are discussed in Chapter 9 “Configuring the Scheduler” on page 113.

To change the selected Scheduler, see the two `configure` options `--set-sched` and `--set-sched-code` in the Features and Package Options section of this chapter. Additional information on the use and configuration of other Schedulers can be found in section 13.5 “Implementing a Custom Scheduler” on page 194.

Step 5 Read section 5.6 “Build Details” on page 35 then, from within the top of the target tree created in step 3, type the following command

```
# /usr/local/src/pbspro_52/configure [options]
#
```

If you are building at the top of the source tree, type

```
./configure [options]
```

This will generate the complete target tree (starting with the current working directory), a set of header files, and all the make files needed to build PBS. Re-running the `configure` script will only need to be done if you choose to change options specified on the configure command line. See section 5.6 “Build Details” on page 35 for information on the configure options, or type

```
./configure --help
```

No options are absolutely required. Note that while GNU’s C compiler (gcc) will work with PBS, the vendor supplied C compiler is usually preferable. If you wish to build the GUI to PBS, and the Tcl libraries are not in the normal place (`/usr/local/lib`) then you will need to specify `--with-tcl=directory` giving the path to the Tcl libraries. If you want to enable Globus support within PBS, then you will need to specify the definitions for *SSL_DIR* and *LDAP_DIR* directories using `--with-globus=DIR`.

Running `configure` without any (other) options will prepare the build process to produce a working PBS system with the following defaults (which can be overridden as described below):

A. User commands are installed in `/usr/local/bin`.
B. Daemons and administrative commands are installed in /usr/local/sbin
C. Working directory for the daemons is `/usr/spool/PBS`
D. C-based Standard Scheduler will be selected

Because the number of options you select may be large and because each option is rather wordy you may wish to create a shell script consisting of the configure command and the selected options.

Step 6 After running the `configure` script, the next step is to compile PBS by typing

```
make
```

from the top of the target tree.

Step 7 To install PBS you must be running with root privileges. As root, type

```
make install
```

from the top of the target tree. This generates the working directory structures required for running PBS and installs the programs in the proper executable directories.

When the working directories are made, they are also checked to see that they have been setup with the correct ownership and permissions. This is performed to ensure that files are not tampered with and the security of PBS and your system are not compromised.

Running `make install` will create the `/etc/pbs.conf` file, or if it already exists then `/etc/pbs.conf.new` containing the locations for the executables and PBS_HOME set to what was specified on the configure options. (For details, see “/etc/pbs.conf” on page 143.)

Step 8 Complete the installation of PBS and set up the license key file by typing, as root:

```
make postinstall
```

First the `postinstall` program will create the PBS Server database (if it doesn’t already exist). When complete, it will contain the default queue and Server configuration described throughout this manual.

Next, `postinstall` will run `pbs_setlicense` which will prompt you for your PBS license key(s). For a full description of the license key installation process, with examples, see “Installing the PBS License” on page 25.

5.6 Build Details

While the overview gives sufficient information to build a basic PBS system, there are lots of options available to allow you to custom tailor PBS to suite your specific environment. The following tables list detailed information on the options to the `configure` script. This information is broken into three separate tables: generic `configure` options, directory and file options, and feature-specific options.

The table below lists the generic configure options available. These alter the behavior of `configure` itself, and therefore do not affect the functionality of PBS.

Generic Configure Options	**Description and Defaults**
`--cache-file=`*`FILE`*	Cache the system configuration test results in file *FILE* Default: `config.cache`
`--help`	Prints out information on the available options.
`--no-create`	Do not create output files.
`--quiet,--silent`	Do not print “checking” messages.
`--version`	Print the version of autoconf that created configure.
`--enable-depend-cache`	Turn on configure’s ability to cache *makedepend* information across runs of configure. This can be bad if the user makes certain configuration changes and reruns configure, but it can save time in the hands of experienced developers. Default: disabled

This second table lists configure options which allow you to specify the directories in which the PBS objects will be placed, as well as the location of specific files.

Directory and File Options	**Description and Defaults**
`--prefix=`*`PREFIX`*	Install files in subdirectories of directory *PREFIX* Default: **/usr/local**
`--exec-prefix=`*`EPREFIX`*	Install architecture dependent files in subdirectories of *EPREFIX*. The value specified will be written to the `/etc/pbs.conf` file. Default: *PREFIX (***/usr/local***)*
`--bindir=`*`DIR`*	Install user executables (commands) in subdirectory *DIR*. Default: *EPREFIX*/bin (**/usr/local/bin**)
`--sbindir=`*`DIR`*	Install System Administrator executables in subdirectory *DIR*. This includes certain administrative commands and the daemons. Default: *EPREFIX*/sbin (**/usr/local/sbin**)

Directory and File Options	Description and Defaults
`--libdir=`*`DIR`*	Object code libraries are placed in *DIR*. This includes the PBS API library, libpbs.a. Default: *PREFIX*/lib (**/usr/local/lib**
`--includedir=`*`DIR`*	C language header files are installed in *DIR*. Default: *PREFIX*/include (**/usr/local/include**)
`--mandir=`*`DIR`*	Install man pages in *DIR*. Default: *PREFIX*/man (**/usr/local/man**)
`--srcdir=`*`TREE`*	PBS sources can be found in directory *TREE*. Default: location of the `configure` script.
`--x-includes=`*`DIR`*	X11 header files are in directory *DIR*. Default: attempts to autolocate the header files
`--x-libraries=`*`DIR`*	X11 libraries are in directory DIR. Default: attempts to autolocate the libraries
`--with-globus=`*`DIR`*	Adding this option will enable Globus support within PBS, causing a separate pbs_mom_globus daemon to be compiled and directories created. *DIR* should specify the GLOBUS_LOCATION directory for the Globus header files, libraries, and `etc/makefile_header`.. See also --set-globus-flavor.
`--set-globus-flavor=`*`C`*	Sets the name of the Globus "flavor" to be used when buidling PBS.
`--with-ssl=SSL`*`DIR`*	Searches *SSLDIR* for the SSL include files and libraries. Use this option only if `--with-globus` could not expand *SSL_DIR*.
`--with-ldap=`*`DIR`*	Previously used in conjuntion with `--with-globus`, but in no longer required.

This third table lists the feature-specific options to configure. In general, these options take the following forms:

`--disable-`*`FEATURE`*	`Do not compile for` *`FEATURE`*`, same as --enable-`*`FEATURE`*`=no`
--enable-*FEATURE*	Compile for *FEATURE*
--with-*PACKAGE*	Compile to include *PACKAGE*

`--without-PACKAGE` *Do not compile to include PACKAGE, same as* `--with-PACKAGE=no`

--set-*OPTION* Set the value of *OPTION*

The recognized feature/package specific options of PBS are:

Feature Option	Description and Default
`--enable-server`	Build (or not build) the PBS Job Server, `pbs_server`. Normally all components (Commands, Server, MOM, and Scheduler) are built. Default: enabled
`--enable-mom`	Build (or not build) the PBS job execution daemon, `pbs_mom`. Default: enabled
`--enable-clients`	Build (or not build) the PBS commands. Default: enabled
`--set-pbs-conf-file=FILE`	Set the name of the file that will contain global information about the PBS installation including paths and port numbers. This value may be overridden at runtime by setting the `PBS_CONF_FILE` environment variable. If a full path is not provided, the filename is relative to the server home directory. Default value is `/etc/pbs.conf`.
`--set-pbs-mach-type=TYPE`	Define machine type to build PBS for, if not the same as returned by `uname`.
`--with-dce=DIR_PREFIX`	Use this to specify the root directory where DCE is installed.
`--with-des=DIR_PREFIX`	Use this to set the root directory where the DES crypto package is installed. This package needs to be available if `--with-dce` is specified.

Feature Option	Description and Default
`--with-tcl=`*`TDIR`*	Use this option if you wish Tcl based PBS features compiled and the Tcl libraries are not in `/usr/local/lib`. These Tcl based features include the GUI interface, `xpbs` and `xpbsmon`. If the following option, `--with-tclx`, is set, use this option only if the Tcl libraries are not co-located with the Tclx libraries. When set, *TDIR* must specify the absolute path of the directory containing the Tcl Libraries. Default: if `--enable-gui` is enabled, Tcl utilities are built; otherwise they are not built.
`--with-tclx=`*`TDIR`*	Use this option if you wish the Tcl based PBS features to be based on Tclx. This option implies `--with-tcl`. Default: Tclx is not used.
`--enable-gui`	Build the `xpbs` and `xpbsmon` GUI. Only valid if `--with-tcl` is set. Default: enabled
`--set-cc=`*`cprog`*	Specify which C compiler should be used. This will override the *CC* environment setting. If only `--set-cc` is specified, then *CC* will be set to `cc` Default: `gcc` (after all, configure is from GNU also)
`--set-cflags=`*`FLAGS`*	Set the compiler flags. This is used to set the *CFLAGS* variable. If only `--set-cflags` is specified, then *CFLAGS* is set to `""`. Default: *CFLAGS* is set to a best guess for the system type.
`--enable-debug`	Builds PBS with debug features enabled. This causes the daemons to remain attached to standard output and produce vast quantities of messages. Default: disabled
`--set-tmpdir=`*`DIR`*	Set the temporary directory in which `pbs_mom` will create temporary scratch directories for jobs. Used on Cray systems only. Default: **`/tmp`**

Feature Option	Description and Default
`--set-default-server=HOST`	Set the name of the host that clients will contact when not otherwise specified in the command invocation. It must be the primary network name of the host. The value specified will be written to `/etc/pbs.conf`. Default: the name of the host on which PBS is being compiled.
`--set-server-home=`*`DIR`*	Sets the top level directory name for the PBS working directories, *PBS_HOME*. This directory **MUST reside on a file system which is local to the host** on which any of the daemons are running. That means you must have a local file system on any system where a pbs_mom is running as well as where pbs_server and/or pbs_sched is running. PBS uses synchronous writes to files to maintain state. We recommend that the file system has the same mount point and path on each host, that enables you to copy daemons from one system to another rather than having to build on each system. The value specified will be written to `/etc/pbs.conf`. Default: **/usr/spool/PBS**
`--set-server-name-file=`*`FILE`*	Set the file name which will contain the name of the default Server. This file is used by the commands to determine which Server to contact. If *FILE* is not an absolute path, it will be evaluated relative to the value of `--set-server-home`, *PBS_HOME*. The value specified will be written to the `/etc/pbs.conf` file. Default: `server_name`
`--set-environ=`*`PATH`*	Set the path name of the file containing the environment variables used by the daemons and placed in the environment of the jobs. For AIX based systems, we suggest setting this option to `/etc/environment`. Relative path names are interpreted relative to the value of `--set-server-home`, *PBS_HOME*. The value specified will be written to `/etc/pbs.conf`. Default: the file `pbs_environment` in the directory *PBS_HOME*.

Feature Option	Description and Default
`--enable-syslog`	Enable the use of syslog for error reporting. This is in addition to the normal PBS logs. Default: disabled
`--enable-plock-daemons=`*`WHICH`*	Enable daemons to lock themselves into memory to improve performance. The argument *WHICH* is the bit-wise-or of 1 for pbs_server, 2 for pbs_sched, and 4 for pbs_mom (7 is all three daemons). This option is recommended for Unicos systems. It must **not** be used for AIX systems. Note, this feature uses the plock() system call which is not available on Linux and bsd derived systems. Before using this feature, check that plock(3) is available on the system. Default: disabled.
`--set-sched=`*`TYPE`*	Set the Scheduler (language) type. If set to `cc` a C based Scheduler will be compiled. If set to `tcl`, a Tcl based Scheduler will be used. If set to `no`, no Scheduler will be compiled. Default: cc
`--set-sched-code=`*`PATH`*	Sets the name of the file or directory containing the source for the Scheduler. This is only used for C Schedulers, where `--set-sched` is set to `cc`. For C Schedulers, this should be a directory name. If the path is not absolute, it will be interpreted relative to *`SOURCE_TREE`*`/src/scheduler.`*`SCHED_TYPE`*`/samples`. Default: `standard` (C based Scheduler)
`--enable-tcl-qstat`	Builds qstat with the Tcl interpreter extensions. This allows site and user customizations. Only valid if `--with-tcl` is already present. Default: disabled
`--set-tclatrsep=`*`CHAR`*	Set the character to be used as the separator character between attribute and resource names in Tcl/Tclx scripts. Default: `"."`

Feature Option	Description and Default
`--set-mansuffix=`*`CHAR`*	Set the character to be used as the man page section suffix letter. For example, the qsub man page is installed as `man1/qsub.1B`. To install without a suffix, `--set-mansuffix=””`. Default: `“B”`
`--set-qstatrcfile=`*`FILE`*	Set the name of the file that qstat will use if there is no .qstatrc file in the user’s home directory. This option is only valid when `--enable-tcl-qstat` is set. If *FILE* is a relative path, it will be evaluated relative to the PBS Home directory, see `--set-server-home`. Default: *`PBS_HOME`*`/qstatrc`
`--with-scp`	Directs PBS to attempt to use the *Secure Copy Program* `scp` when copying files to or from a remote host. This option cause `PBS_SCP` to be set in `/etc/pbs.conf` with the indicated path. Default: `sbindir/pbs_rcp`
`--enable-shell-pipe`	When enabled, pbs_mom passes the name of the job script to the top level shell via a pipe. If disabled, the script file is the shell’s standard input file. Default: enabled
`--enable-rpp`	Use the Reliable Packet Protocol, RPP, over UDP for resource queries to MOM by the Scheduler. If disabled, TCP is used instead. Default: enabled
`--enable-nodemask`	Build PBS with support for SGI Origin2000 nodemask. Requires Irix 6.x. Default: disabled
`--enable-pemask`	Build PBS on Cray T3e with support for Scheduler controlled pe-specific job placement. Requires Unicos/MK2. Default: disabled
`--enable-srfs`	This option has been removed.

Feature Option	Description and Default
`--enable-sp2`	Turn on special features for the IBM SP. This option is only valid when the PBS machine type is aix4. The PBS machine type is automatically determined by the configure script. With PSSP software before release 3.1, access to two IBM supplied libraries, libjm_client.a and libSDR.a, are required. These libraries are installed when the ssp.clients fileset in installed, and PBS will expect to find them in the normal places for libraries. With PSSP 3.1 and later, libjm_client.a and libSDR.a are not required, instead libswitchtbl.a is used to load and unload the switch. See the discussion under the sub-section **IBM SP** in section 5.8 “Machine Dependent Build Instructions” on page 44. Default: disabled
`--enable-cpuset`	Setting this under Irix 6.x forces the use of cpusets. Cpuset is a named set of CPUs where member processes are to be run. Enabling this feature does not use Array Session tracking; instead, jobs are tracked by session ids. Default: disabled
`--enable-uselog`	Setting this option will send `pbs_rcp` file transfer statistics to the system’s log file. `pbs_rcp` is the default file transfer utility used by PBS to deliver user’s input/output/error files. In order for this option to work, ensure that **LOG_INFO**, **LOG_DELAY**, or **LOG_LOCAL4** messages are enabled for your syslog. Default: disabled

5.7 Make File Targets

The following target names are applicable for make:

all	The default target, it compiles everything.
build	Same as all.
depend	Builds the header file dependency rules.
install	Installs everything.
clean	Removes all object and executable files in the current subtree.
distclean	Leaves the object tree very clean. It will remove all files that were created during a build.
postinstall	Runs the script that installs the PBS license key(s).

5.8 Machine Dependent Build Instructions

There are a number of possible variables that are only used for a particular type of machine. If you are not building for one of the following types, you may ignore this section.

5.8.1 Linux

Redhat Linux version 4.x-7.x are supported with no special `configure` options required.

5.8.2 Sun Solaris Systems

On Solaris systems, if the system has more than 2GB of physical memory, PBS must be built with `--set-cflags="-g -xarch=v9"`, in order to correctly calculate memory amounts.

5.8.3 Digital UNIX

The following is the recommended value for CFLAGS when compiling PBS under Tru64 4.0D: `--set-cflags="-std1",` that is s-t-d-one.

5.8.4 HP-UX

When building from source under HPUX, the following environment variables must be defined before running "configure":

```
                    setenv CC cc
                    setenv AR ar
```

or

```
                    CC=cc; export CC
                    AR=ar; export AR
```

For HP-UX 11.x, `pbs_mom` must be built as a 64 bit object, i.e.:

```
                    CCOPTS=+DD64; export CCOPTS
```

The libraries must match of course. If you desire, you may rebuild the rest of PBS as a 32 bit object.

Under HP-UX 11.x, Tcl and Tk are not located in the directory expected by PBS's configure script. You will need to make a common directory for Tcl and Tk and point to it by setting

`--with-tcl=`*`new_directory`*.

If building for a V2500 or N4000 system with PA8500 processors, also set the following variable for performance reasons:

```
       CCOPTS=+DA2.0 ; export CCOPTS
```

If building for a PA2 system on which you may run 64 bit programs, you must build PBS with the following option added to *CFLAGS* so that PBS can correctly monitor memory use of 64 bit programs:

```
       --set-cflags="-D_PSTAT64 -Ae"
```

The following is the recommended value for CFLAGS when compiling PBS under HP-UX 10.x:

```
       --set-cflags="-Ae"
```

5.8.5 IBM Workstations

PBS supports IBM workstations running AIX 4.x and 5.x, however the AIX `man`(1) command has difficulty with the PBS man pages. When man pages are installed in *mandir* the default man page file name suffix, "B", must be removed. This can be accomplished with the configure option `--set-mansuffix=""`. Also, do not use the configure option:

```
    --enable-plock
```

on AIX workstations as it will crash the system by reserving all system memory.

5.8.6 IBM SP

Everything under the discussion about IBM Workstations also applies to the IBM SP series. Be sure to read the section 3.7 “Multiple Execution Systems” on page 17 before configuring the Server. Set the special SP option, `--enable-sp2` to compile special code to deal with the SP high speed switch.

If the library *`libswitchtbl.a`* is not detected, it is assumed that you are running with PSSP software prior to 3.1. In this case, the IBM `poe` command sets up the high speed switch directly and PBS interfaces with the IBM Resource (Job) Manager to track which nodes jobs are using. PBS requires two libraries, `libjm_client.a` and `libSDR.a`, installed with the ssp.clients fileset.

If the library `libswitchtbl.a` is detected, it is assumed you are running with PSSP 3.1 or later software. In this case, PBS takes on the responsibility of loading the high speed switch tables to provide node connectivity.

Important: The PBS_HOME directory, see `--set-server-home`, used by the `pbs_mom` located on each node, **must** be on local storage and must have an identical path on each node. If the directory is setup in a different path, then MOM will not be able to initialize the SP switch correctly.

The node names provided to the Server **must** match the node names shown by the st_status command. This should be the “reliable” node name.

Important: Regardless of the number of real processors per node, the number of virtual processors that may be declared to the Server is limited to the number of Switch windows supported by the PSSP software. At the present time, this is eight (8). Therefore only eight virtual processors may be declared per node.

With PSSP 3.1, two additional items of information must be passed to the job: the switch window id (via a file whose name is passed), and a *job key* which authorizes a process to use the switch. As `poe` does not pass this information to the processes it creates, an underhanded method had to be created to present them to the job. Two new programs are compiled and installed into the *bindir* directory, `pbspoe` and `pbspd`.

`pbspoe` is a wrapper around the real `poe` command. pbspoe must be used by the user in place of the real `poe`. `pbspoe` modifies the command arguments and invokes the real `poe`, which is assumed to be in `/usr/lpp/ppe.poe/bin`. If a user specifies:

```
pbspoe a.out args
```

that command is converted to the effective command:

```
/usr/lpp/ppe.poe/bin/poe pbspd job_key \
winid_file a.out args -hfile $PBS_NODEFILE
```

PBS_NODEFILE of course contains the nodes allocated by pbs. The `pbs_mom` on those nodes have loaded the switch table with the user's uid, the job key, and a window id of zero. `pbspd` places the job key into the environment as *MP_PARTITION* and the window id as *MP_MPI_NETWORK* `pbspd` then exec's a.out with the remaining arguments.

If the user specified a command file to `pbspoe` with `-cmdfile` *`file`* then `pbspoe` prefixes each line of the command file with `pbspd job_key` and copies it into a temporary file. The temporary file is passed to `poe` instead of the user's file.

`pbspoe` also works with `/usr/lpp/ppe.poe/bin/pdbx` and `/usr/lpp/ppe.poe/bin/xpdbx`. This substitution is done to make the changes as transparent to the user as possible.

Important: Not all `poe` arguments or capabilities are supported. For example, `poe` job steps are not supported.

For transparent usage, it is **necessary** that after PBS is installed that you perform these additional steps:

Step 1 Remove IBM's `poe`, `pdbx`, and `xpdbx` from `/usr/bin` or any directory in the user's normal path. Be sure to leave the commands in `/usr/lpp/ppe.poe/bin` which should not be in the user's path, or if in the user's path must be after `/usr/bin`.

Step 2 Create a link named `/usr/bin/poe` pointing to *`{bindir}`*`/pbspoe`. Also make links for `/usr/bin/pdbx` and `/usr/bin/xpdbx` which point to *`{bindir}`*`/pbspoe`.

Step 3 Be sure that `pbspd` is installed in a directory in the user's normal path on each and every node.

5.8.7 SGI Systems Running IRIX 6

If built for Irix 6.x, pbs_mom will track which processes are part of a PBS job using POSIX session numbers. The PBS machine type (PBS_MACH) is set to irix6.

Important: If you are compiling on an SGI system, you must have C compiler Version 7.3.1.3m or later to prevent a compiler introduced bug dealing with `_MIPS_SYMBOL_PRESENT()`.

PBS can be built with cpuset support by setting `--enable-cpuset`. A cpuset names a set of CPUs for a job. Processes spawned for a job will be restricted to this set (both cpu-wise and memory-wise).

Specifically, PBS sets the following flags during cpuset creation:

```
CPUSET_CPU_EXCLUSIVE,
CPUSET_MEMORY_LOCAL
CPUSET_MEMORY_EXCLUSIVE
CPUSET_MEMORY_MANDATORY
CPUSET_POLICY_KILL
CPUSET_EVENT_NOTIFY
```

See `cpusetCreate`(3) for more details.

The PBS machine type (PBS_MACH) is set to `irix6cpuset`.

A special job attribute *ssinodes* is expected to be set by the Scheduler, in order for `pbs_mom` to determine the list of CPUs to assign. *ssinodes* refers the number of single-system-image nodes to be assigned to a job. Another attribute, *nodemask* is set by `pbs_mom` to a hexadecimal string showing the nodes/CPUs assigned to a job-- 1 bit represents 1 node.

Finally, a special job attribute *hpm* can be used in a scheduler to take into account SGI's Hardware Performance Monitor (HPM). This is an attribute that allows users to take advantage of such software as `perfex`. SGI Origin2000's only allow one global counter at a time, so when the system is using it, users are unable to do so and vice versa. Specifying "`hpm`" in the job submission, specifies if it is to use the HPM counters.

The Standard Scheduler supports cpusets, as does one of the alternate schedulers: `sgi_origin_cpuset`. If you wish to use this alternate scheduler, review the configuration information in the documentation for this scheduler in the following directory of the source tree: `src/scheduler.cc/samples/sgi_origin_cpus`et. To enable this scheduler, be sure to specify these configure options:

```
--set-sched=cc
--set-sched-code=sgi_origin_cpuset
```

Important: IRIX 6 supports both 32 and 64 bit objects. In prior versions, PBS was typically built as a 32 bit object. Because of changes in structure sizes, PBS will not be able to recover any Server, queue, or job information recorded by a PBS Server built with 32 bit objects, or vice versa. Please read Chapter 6 “Upgrading PBS” on page 53 for instructions on dealing with this incompatibility.

5.8.8 Cray C90, J90, and T90 Systems

On the traditional Cray systems such as the C90, PBS supports Unicos versions 8, 9 and 10. The directory for *TMPDIR* will default to that defined by *JTMPDIR* in Unicos’s `/usr/include/tmpdir.h`. MOM under Unicos will create a temporary job scratch directory. By default, this is placed in `/tmp`. The location can be changed via `--set-tmpdir=`*`DIR`*

5.8.9 Unicos 10 with MLS

If you are running Unicos MLS, required in Unicos 10.0 and later, the following action is required after the system is built and installed. MOM updates **ue_batchhost** and **ue_batchtime** in the UDB for the user. In an MLS system, MOM must have the security capability to write the protected UDB. To grant this capability, change directory to wherever `pbs_mom` has been installed and type:

```
spset -i 16 -j daemon -k exec pbs_mom
```

You, the administrator, must have capabilities **secadm** and **class 16** to issue this command. You use the `setucat` and `setucls` commands to get to these levels if you are authorized to do so. The UDB **reclsfy** permission bit gives a user the proper authorization to use the `spset` command.

5.8.10 Cray T3E

On the Cray T3E MPP systems, PBS supports the microkernal-based Unicos/MK version 2. On this system PBS “cooperates” with the T3E Global Resource Manager (GRM) in order to run jobs on the system. This is needed primarily since jobs on the T3E must be run on physically contiguous processing elements (PEs). Previous discussions regarding the changing of *TMPDIR* are also applicable to the Cray T3E.

5.8.11 Compiling in DCE Support

Release 5.2 of PBS Pro introduced various code additions to support PBS in a DCE environment. As far as the administrator and/or user are concerned, these additions include:

1. Two new option arguments to the configure script
2. A new option argument to client qsub
3. Generation of a per job "credential" (.CR) file in both mom/jobs and server/jobs directories
4. Generation of a per job security context "export" file (.XP suffix) in mom/jobs.

To build for DCE support, one must have the DCE libraries and the DES encryption library available. The latter is part of the SSI cryptography package. One possible download site is:

```
ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL
```

There are two DCE-specific PBS configure options:

`--with-des` - the base directory where the des crypto package is installed.
`--with-dce` - the base directory for the installed DCE libraries.

Important: Currently, the credential refresh interval for the job is not determined from information in the security service's registry. It is instead a constant value chosen by the administrator. This value (number of seconds between refresh attempts) is set via the MOM’s config file parameter `$dce_refresh_delta`. If the parameter is not specified, the value will default to the compiled in value of 18000 (5 hours).

5.9 Install Options

There are four ways in which a MOM may be installed on each of the various execution hosts.

Step 1 The first method is to do a full install of PBS on each host. While this works, it is a bit wasteful.

Step 2 The second way is to rerun configure with the following options: `--disable-server --set-sched=no`

You may also choose `--disable-clients` but users often use the PBS commands within a job script so you will likely want to build the commands. You will then need to recompile and then do an install on each execution host.

Step 3 The third way is to install just MOM (and maybe the commands) on each system. If the system will run the same binaries as where PBS was compiled, then as root:

```
# cd src/resmom
# make install
# cd ../cmds
# make install#
```

If the system requires recompiling, do so at the top level to recompile the libraries and then proceed as above.

Step 4 The fourth requires that the system be able to execute the existing binaries and that the directories *`sbindir`* and *`bindir`* in which the PBS daemons and commands were installed during the initial full build be available on each host. These directories, unlike the *PBS_HOME* directory can reside on a network file system. If the target tree is accessible on the host, as root execute the following commands on each execution host:

```
sh {target_tree}/buildutils/pbs_mkdirs [-d new_dir] mom
sh {target_tree}/buildutils/pbs_mkdirs [-d new_dir] aux
sh {target_tree}/buildutils/pbs_mkdirs [-d new_dir] default
```

This will build the required portion of *PBS_HOME* on each host. Use the `-d` option if you wish to place *PBS_HOME* in a different place on the node. This directory must be on local storage on the node, not on a shared file system. If you use a different path for *PBS_HOME* than was specified when configure was run, you must also start pbs_mom with the corresponding `-d` option so she knows where *PBS_HOME* is located.

If the target tree is not accessible, copy the `pbs_mkdirs` shell script to each execution host and, again as root, execute it with the above operands.

Note that the initial run of the Server or any first time run after recreating the home directory must be with the `-t create` option. This option directs the Server to create a new Server database. This is best done by hand.

If a database is already present, it is discarded after receiving a positive validation response. At this point it is necessary to configure the Server. See Chapter 7 “Configuring the Server” on page 61 for details. The create option leaves the Server in a “idle” state. In this state the Server will not contact the Scheduler and jobs are not run, except manually via the `qrun`(1B) command.

Once the Server is up and configured, it can be placed in the “active” state by setting the Server attribute `scheduling` to a value of `true`:

```
# qmgr -c “set server scheduling=true”
```

The value of `scheduling` is retained across Server terminations/starts. After the Server is configured it may be placed into service.

You will now need to configure the PBS daemons as discussed in Chapter 7, Chapter 8, and Chapter 9.

Chapter 6 Upgrading PBS

This chapter provides important information on upgrading from a previous version of PBS. If you are not currently running PBS, you can safely skip this chapter at this time. However, be sure to refer to it before preforming a future upgrade of PBS.

There are two types of PBS upgrades: an *overlay* upgrade (in which you simply install the new binaries on top of the old ones) and a *migration* upgrade (in which you must transfer the jobs between the old and new versions). If the underlying job data structure changed between the two releases, a migration upgrade is needed.

Important: The Release Notes will indicate whether a migration upgrade is required or not.

6.1 Overlay Upgrade

Most new releases of PBS (especially minor versions and patch releases) will allow you to upgrade quickly. If the Release Notes so indicate, then follow these simple steps:

1. Shutdown PBS
2. Backup the Server’s job directory
3. Install the new version
4. Restart PBS

You may use `tar` to perform the backup of the jobs directory. This backup step is precautionary only; allowing jobs to be restored should an eror occur.

```
# cd /usr/spool/PBS/server_priv
# tar -cf /tmp/pbs_jobs_save jobs
```

Follow the normal installation procedures for installing the new version. The installation program will pick up your existing installation parameters from the `/etc/pbs.conf` file, and prompt you to confirm that you wish to use them.

6.2 Migration Upgrade

A migration upgrade is more complicated, and is needed when the PBS job structure is different between your currently installed version, and the new version.The release notes will indicate if you need to migrate (or move) jobs from the old version to the new.

PBS allows you to run two instances of PBS on the same system, by specifying alternative daemon directories and port numbers. When the new version is ready to be placed into service, you will probably want to move jobs from the old system to the new. The following procedure is suggested. All Servers must be run by root. The `qmgr` and `qmove` commands should be run by a PBS administrator (likely, root is good).

Step 1 With the old PBS daemons running, disable the queues by setting each queue's `enabled` attribute to false, and stop any further scheduling by setting the Server's `scheduling` attribute to false. This will prevent any new jobs from being accepted or started by PBS.

```
# qmgr
Qmgr: set queue workq enabled = false
Qmgr: set server scheduling = false
Qmgr: quit
```

Step 2 Backup this Server's jobs directory, `/usr/spool/PBS/server_priv/jobs` -- tar may used for this.

```
# cd /usr/spool/PBS/server_priv
# tar -cf /tmp/pbs_jobs_save jobs
```

If the job structure has changed, you will need to *move* the jobs from the old system to the new. The release notes will contain a warning whenever the upgrade will need a move of existing jobs to the new Server.

Note: You may find it useful to set the PBS environment variable **PBS_SERVER** (as discussed in the **PBS User Guide**) to either the old or new Server, with the port numbers specified, to avoid having to type them on the command lines as described below.

To move the jobs, continue with the following steps:

Step 3 Start the new PBS Server with a value of **PBS_HOME** different from the value that is being used by the current Server (check $PBS_HOME, or the value of **PBS_HOME** in /etc/pbs.conf). How you decide to do this will depend on whether you are starting fresh with the new Server or have used it in some test arrangement, and now want to cut over to it in the production environment. Use the '-t create' option for the Server if you starting fresh. To set the alternative port for the new Server, it's probably easier to that on the command line via the '-p port' option. Instruct both Servers, current and new, to turn off scheduling. Again, the easiest way, if you are starting up the new Server, is to specify the Server's '-a false' option on the command line.

```
# pbs_server -t create -d new_home -p 13001 -a fals
```

Keep in mind that: if the environment variable **PBS_SERVER** exists but doesn't have a value that is newserver:port or, if the environment variable doesn't exist and **PBS_SERVER** in /etc/pbs.conf is not the value *newserver*:*port* then, you will need to use the :port syntax in the command in order that the request to be sent to the new Server. Likely, this is the way that you will have things setup when doing the move of jobs to the new Server.

Step 4 Duplicate on the new Server the current queues and Server attributes (assuming you wish to do so). Enable each queue which will receive jobs at the new Server.

```
# qmgr -c “print server" > /tmp/q_config
# qmgr host:13001  < /tmp/q_config
# qenable workq@host:13001
# qenable someq@host:13001
 ...
```

Step 5 Now list the jobs at the original Server and move a few jobs one at a time from the old to the new Server:

```
# qstat
# qmove workq@host:13001 job_id
# qstat @host:13001
```

If all is going well, move the remaining jobs a queue at a time:

```
# qmove workq@host:13001 ‘qselect -q workq‘
# qstat workq@host:13001
# qmove someq@host:13001 ‘qselect -q someq‘
# qstat someq@host:13001
...
```

Step 6 At this point, all of the jobs should be under control of the new Server and located in the new Server’s home. If the new Server’s home is a temporary directory, shut down the new Server with `qterm newserver:port` and move everything to the real home as follows:

```
# cp -R new_home real_home
```

or, if the real (new) home is already set up, do the following to copy just the jobs from the jobs subdirectory:

```
# cd new_home/server_priv/jobs
# cp * real_home/server_priv/jobs
```

or, as a third alternative, if the real (new) PBS_HOME exists (including the *_priv directories), and this is also what you want it to be in the production environment, then merely update the value of **`PBS_HOME`** in the `/etc/pbs.conf` file.

Now you are ready to start and enable the new batch system.

You should be aware of one quirk when using `qmove`. If you wish to move a job from a Server running on a test port to the Server running on the normal port (15001), you may attempt, *unsuccessfully*, to use the following command:

```
# qmove workq@host 123.job.host:13001
```

However, doing so will only serve to move the job to the end of the queue it is already in. The Server receiving the move request (e.g. the Server running on port 13001), will compare the destination Server name, host, with its own name only, not including the port. Hence it will match and it will not send the job where you intended. To get the job to move to the Server running on the normal port you have to specify that port in the destination:

```
# qmove workq@host:15001 123.job.host:13001
```

6.3 Alternate Test Systems

Running an alternate or test version of PBS requires a couple extra steps. In particular, the alternate version will need a separate PBS directory structure from which to run, this must be created before attempting to start the alternate version.

If working from the standard binary distribution, you can copy your existing PBS directory tree to the new location. (`Tar` may be used for this.) However, you will want to ensure that you delete any job or reservation data that might have been copied to the alternate location.

If building PBS from source code, the easiest manner to create the alternate directory structure is to rerun the `configure` command with the --set-server-home option set the desired value for the alternate *PBS_HOME***.** Next, rebuild and install PBS. (See Chapter 5 for additional information on building from source.)

```
# cd existing_PBS_HOME
# tar -cvf /tmp/pbs_tree.tar .
# mkdir /path/to/alternate_PBS_HOME
# cd /path/to/alternate_PBS_HOME
# tar -xvf /tmp/pbs_tree.tar
# /bin/rm server_priv/jobs/*  mom_priv/jobs/*
# /bin/rm server_priv/resvs/*
```

Alternate or test copies of the various daemons may be run through the use of the command line options which set their home directory and service port. For example, the following commands would start the three daemons with a home directory of `/tmp/altpbs` and port assignments as follows: the Server on 13001, MOM on 13002 and 13003, optional MOM Globus on 13004, 13005, and the Scheduler on 13004.

```
# pbs_server -t create -d /tmp/altpbs -p 13001 -M 13002 \
   -R 13003 -S 13004 -g 13005 -G 13006
# pbs_mom -d /tmp/altpbs -M 13002 -R 13003
# pbs_sched -d /tmp/altpbs -S 13004 -r script_file
# pbs_mom_globus -d /tmp/altpbs -M 13005 -R 13006
```

Note that when the Server is started with a non-standard port number (i.e. with the `-p` option as shown above) the Server “name” becomes *host_name.domain:port*, where port is the numeric port number being used.

Jobs may now be directed to the test system by using the `-q` option to qsub with the `server:port` syntax. Status is also obtained using the `:port` syntax. For example, to submit a job to the default queue on the above test Server, request the status of the test Server, and request the status of jobs at the test Server:

```
# qsub  -q @host:13001 jobscript
# qstat -Bf host:13001
# qstat @host:13001
#
```

Important: If using job dependencies on or between test systems, there are minor problems of which you (and the users) need to be aware. The syntax of both the dependency string and the job `host:port` syntax use colons in an indistinguishable manner.

6.4 Dependent Jobs and Test Systems

If you have users running on a test batch system using an alternative port number, `-p` option to `pbs_server`, problems may occur with job dependency if the following requirements are not observed:
For a test system, the job identifier in a dependency specification must include at least the first part of the host name.

The colon in the port number specification must be escaped by a back-slash. This is true for both the Server and current Server portion of the jobid. For example:

```
23.test_host\:17000
23.old_host@test_host\:17000
23.test_host\:17000@diff_test_host\:18000
```

On a shell line, the back slash itself must be escaped from the shell, so the above become:

```
23.test_host\\:17000
23.old_host@test_host\\:17000
23.test_host\\:17000@diff_test_host\\:18000
```

These rules are not documented on the qsub/qalter man pages since the likelihood of the general user community finding themselves setting up dependencies with jobs on a test system is small and the inclusion would be generally confusing.

Chapter 7
Configuring the Server

Now that PBS has been installed, the Server and MOMs can be configured and the scheduling policy selected. The next three chapters will walk you through this process.

If you installed PBS from the binary distribution, then further configuration may not be required as the default configuration may completely meet your needs. However, you are advised to read this chapter to determine if the default configuration is indeed complete for you, or if any of the optional setting may apply.

7.1 Network Addresses and Ports

PBS makes use of fully qualified host names for identifying the jobs and their location. A PBS installation is known by the host name on which the Server is running. The name used by the daemons, or used to authenticate messages is the canonical host name. This name is taken from the primary name field, `h_name`, in the structure returned by the library call `gethostbyaddr()`. According to the IETF RFCs, this name must be fully qualified and consistent for any IP address assigned to that host.

The daemons and the commands will attempt to use `/etc/services` to identify the standard port numbers to use for communication. The port numbers need not be below the magic 1024 number. The service names that should be added to `/etc/services` are:

```
pbs                15001/tcp # PBS Server (pbs_server)
pbs_mom            15002/tcp # MOM to/from Server
pbs_resmon         15003/tcp # MOM RM requests
pbs_resmon         15003/udp # MOM RM requests
pbs_sched          15004/tcp # PBS Scheduler
pbs_mom_globus     15005/tcp # MOM Globus
pbs_resmon_globus  15006/tcp # MOM Globus RM requests
pbs_resmon_globus  15006/udp # MOM Globus RM requests
```

The port numbers listed are the default numbers used by this version of PBS. If you change them, be careful to use the same numbers on all systems. Note, the name *pbs_resmon* is a carry-over from early versions of PBS when there existed separate daemons for job execution (`pbs_mom`) and resource monitoring (`pbs_resmon`). The two functions were combined into `pbs_mom` though the term "resmom" might be found referring to the combined functions. If the services cannot be found in `/etc/services`, the PBS components will default to the above listed port numbers.

7.2 qmgr

The PBS manager command, `qmgr`, provides a command-line administrator interface to the PBS Server. There are several command line options to `qmgr`.

Option	Action
-a	Abort `qmgr` on any syntax errors or any requests rejected by a Server.
-c command	Execute a single command and exit `qmgr`. The command must be enclosed in quote marks, e.g. `qmgr -c "print server"`
-e	Echo all commands to standard output.
-n	No commands are executed, syntax checking only is performed.
-z	No errors are written to standard error.

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.

A command is terminated by a new line character or a semicolon (";") character. Multiple commands may be entered on a single line. A command may extend across lines by escaping the new line character with a back-slash ("\"). Comments begin with the "#" character and continue to end of the line. Comments and blank lines are ignored by `qmgr`. The syntax of each directive is checked and the appropriate request is sent to the Server(s). A `qmgr` directive takes one of the following forms:

```
command server [names] [attr OP value[,...]]
command queue  [names] [attr OP value[,...]]
command node   [names] [attr OP value[,...]]
```

Where command is the sub-command to perform on a object. Commands are:

Command	**Explanation**
active	Sets the active objects. If the active objects are specified, and the name is not given in a `qmgr` command the active object names will be used.
create	Create a new object, applies to queues and nodes.
delete	Destroy an existing object, applies to queues and nodes.
help	Prints command specific help and usage information
list	List the current attributes and associated values of the object.
print	Print settable queue and Server attributes in a format that will be usable as input to the `qmgr` command.
set	Define or alter attribute values of the object.
unset	Clear the value of the attributes of the object. Note, this form does no accept an OP and value, only the attribute name.

Other `qmgr` syntax definitions follow:

Variable	`qmgr` Variable/Syntax Description
names	The list of one or more names of specific objects. The name list is in the form: `[name][@server][,name[@server]...]` with no intervening white space. The name of an object is declared when the object is first created. If the name is `@server`, then all the object of specified type at the Server will be effected.
attr	Specifies the name of an attribute of the object which is to be set or modified. The attributes of objects are described in section 2 of the **PBS ERS**. If the attribute is one which consists of a set of resources, then the attribute is specified in the form: `attribute_name.resource_name`
OP	An operation to be performed with the attribute and its value:
`=`	Set the value of the attribute. If the attribute has an existing value, the current value is replaced with the new value.
`+=`	Increase the current value of the attribute by the amount in the new value.
`-=`	Decrease the current value of the attribute by the amount in the new value.
value	The value to assign to an attribute. If value includes white space, commas, or other special characters, such as “#”, the value string must be inclosed in quote marks (“ ”).

The `help,  list` and `print` sub-commands of `qmgr` can be executed by the general user. Creating or deleting a queue requires PBS Manager privilege. Setting or unsetting `Server` or `queue` attributes requires PBS Operator or Manager privilege.

The `qmgr` built-in help function, invoked using the “help” sub-command, is illustrated by the next example.

```
% qmgr
Qmgr: help
To get help on any topic, type help <topic>
Help is available on all commands and topics.
Available commands:
active, create, delete, set, unset, list, print, quit
Other topics: attributes, operators, names, and values
```

For example, requesting usage information on the “set” sub-command of `qmgr` would produce the following output.

```
% qmgr
Qmgr: help set
Syntax:
    set object [name][,name...] attribute[.resource] OP value
Objects can be "server" or "queue", "node"
The "set" command sets the value for an attribute on the spec-
ified object. If the object is "server" and name is not spec-
ified, the attribute will be set on all the servers specified
on the command line. For multiple names, use a comma separated
list with no intervening whitespace.
Examples:
set server s1 max_running = 5
set server managers = root
set server managers += susan
set node n1,n2 state=down
set queue q1@s3 resources_max.mem += 5mb
set queue @s3 default_queue = batch
```

Here are several more examples that illustrate using the `qmgr` command. Full explanation of these and other `qmgr` commands are given below in explanation of the specific tasks they accomplish.

```
% qmgr
Qmgr: create node mars ntype=cluster
Qmgr: set node mars resources_available.ncpus=2
Qmgr: create node venus properties=”inner,moonless”
Qmgr: set node mars properties = inner
Qmgr: set node mars properties += haslife
Qmgr: delete node mars
Qmgr: d n venus
```

Important: Commands can be abbreviated to their minimum unambiguous form (as shown in the last line in the example above).

7.3 Default Configuration

Server management consists of configuring the Server attributes, defining nodes, and establishing queues and their attributes. The default configuration from the binary installation sets the minimum Server settings, and some recommended settings for a typical PBS cluster. (The default Server configuration is shown below.) The subsequent sections in this chapter list, explain, and provide the default settings for all the Server's attributes for the default binary installation.

```
% qmgr
Qmgr: print server
# Create queues and set their attributes.
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
#
# Set server attributes.
#
set server scheduling = True
set server default_queue = workq
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

7.4 Server Attributes

This section explains all the available Server attributes and gives the default values for each. Note that the possible values for the "boolean" format are any of: "TRUE", "True", "true", "Y", "y", "1"; "FALSE", "False", "false", "N", "n", "0".

The privilege required to set or change some Server attributes has changed since the previous release. Specifically, `mail_from`, `resources_cost`, and `system_cost` now require Manager privilege. `comment` requires at least operator privilege.

acl_host_enable	When `true` directs the Server to use the `acl_hosts` access control lists. Requires full Manager privilege to set or alter. Format: boolean Default value: false = disabled `Qmgr: set server acl_host_enable=true`
acl_hosts	List of hosts which may request services from this Server. This list contains the fully qualified network name of the hosts. Local requests, i.e. from the Server's host itself, are always accepted even if the host is not included in the list. Wildcards ("*") may be used in conjunction with host names and host.subdomain names. Format: "[+|-]hostname.domain[,...]". Default value: all hosts `Qmgr: set server acl_hosts=*.pbspro.com`
acl_user_enable	Attribute which when true directs the Server to use the Server level `acl_users` access list. Requires full Manager privilege to set or alter. Format: boolean (see acl_group_enable). Default value: disabled `Qmgr: set server acl_user_enable=true`
acl_users	List of users allowed or denied the ability to make any requests of this Server. Requires full Manager privilege to set or alter. Format: "[+|-]user[@host][,...]". Default value: all users allowed `Qmgr: set server acl_users=bob,tom@sol,sue@sol`
acl_roots	List of superusers who may submit to and execute jobs at this Server. If the job execution id would be zero (0), then the job owner, root@host, must be listed in this access control list or the job is rejected. Format: "[+|-]user[@host][,...]". Default value: no root jobs allowed `Qmgr: set server acl_roots=host`
comment	A text string which may be set by the scheduler or other privileged client to provide information to PBS users. Format: any string. Default value: none `Qmgr: set server comment="Planets Cluster"`
default_node	A node specification to use if there is no other supplied specification. The default value allows jobs to share a single node. Format: a node specification string. Default value: 1#shared `Qmgr: set server default_node="1#shared"`

default_queue

The queue which is the target queue when a request does not specify a queue name.
Format: a queue name.
Default value: none, must be set to an existing queue

```
Qmgr: set server default_queue=workq
```

flatuid

Attribute which directs the Server to automatically grant authorization for a job to be run under the user name of the user who submitted the job even if the job was submitted from a different host. If not set `true`, then the Server will check the authorization of the job owner to run under that name if not submitted from the Server's host. See section 11.8.4 “User Authorization” on page 156 for usage and an important caveat.
Format: boolean
Default value: False

```
Qmgr: set server flatuid=True
```

log_events

A bit string which specifies the type of events which are logged, (see also section 11.14 “Use and Maintenance of Logfiles” on page 163).
Format: integer.
Default value: 511, all events

```
Qmgr: set server log_events=255
```

mail_from

The uid from which Server generated mail is sent to users.
Format: integer uid
Default value: 0 for root

```
Qmgr: set server mail_uid=1264
```

managers

List of users granted PBS Manager privileges.
The host, sub-domain, or domain name may be wild carded by the use of an * character. Requires Manager privilege to set or alter.
Format: “user@host.sub.domain[,user@host.sub.domain...]” .
Default value: root on the local host

```
Qmgr: set server managers+=boss@sol.pbspro.com
```

max_running

The maximum number of jobs allowed to be selected for execution at any given time. Advisory to the Scheduler, not enforced by the Server.
Format: integer.
Default value: none

```
Qmgr: set server max_running=24
```

max_group_run

The maximum number of jobs owned by any users in a single group that are allowed to be running from this queue at one time. This attribute is advisory to the Scheduler, it is not enforced by the Server.
Format: integer

Default value: none
```
Qmgr: set server max_group_run=16
```

max_user_run

The maximum number of jobs owned by a single user that are allowed to be running from this queue at one time. This attribute is advisory to the Scheduler, it is not enforced by the Server.
Format: integer
Default value: none
```
Qmgr: set server max_user_run=6
```

max_user_res.*RES*

The maximum amount of resource *RES* that any single user may consume. *RES* can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be specified as both a *hard* and *soft* limit. Most limits under PBS are *hard* limits (i.e. they cannot be exceeded by the job). However, a *soft* limit is a limit which *can* be exceeded, if additional resources are available when the job is being considered for running. Note that any job run using soft limits is eligible to be preempted by jobs of other users who are not exceeding the soft limits.
Format: resource specific
Default value: none
```
Qmgr: set server max_user_res.ncpus=6
Qmgr: set server max_user_res_soft.ncpus=3
```

The first line in the example above sets a normal (e.g. *hard*) limit of 3 CPUs as a maximum that any single user may consume. The second line in the example illustrates setting a *soft* limit of 6 CPUs on the same resource.

node_pack

Used when the Server is making job-placement decision, rather than the Scheduler, to control how multiple processor nodes are allocated to jobs. If this attribute is set to true, jobs will be assigned to the multiple processor nodes with the fewest free processors. This packs jobs into the fewest possible nodes leaving multiple processor nodes free for jobs which need many processors on a node. If set to `false`, jobs will be scattered across nodes reducing conflicts over memory between jobs. If unset, the jobs are packed on nodes in the order that the nodes are declared to the Server (in the nodes file).
Format: boolean
Default value: unset (assigns nodes in order declared)
```
Qmgr: set server node_pack=true
```

operators

List of users granted batch operator privileges.
Format of the list is identical with `managers` above. Requires full Manager privilege to set or alter.
Default value: root on the local host.
```
Qmgr: set server operators=sue,bob,joe,tom
```

query_other_jobs	The setting of this attribute controls whether or not general users, other than the job owner, are allowed to query the status of or select the job. Requires Manager privilege to set or alter (see `acl_host_enable`). Format: boolean Default value: false (users may not query or select jobs owned by other users.) `Qmgr:` **`set server query_other_jobs=true`**

resources_available	The list of resources and amounts available to jobs run by this Server. The sum of the resources of each type used by all jobs running by this Server cannot exceed the total amount listed here. Advisory to the Scheduler, not enforced by the Server.
	Format: “resources_available.resource_name=value[,...]”.
	`Qmgr:` **`set server resources_available.ncpus=16`**
	`Qmgr:` **`set server resources_available.mem=400mb`**

resources_cost	This attribute is currently unused.
resources_default	The list of default resource values that are set as limits for a job executing on this Server when the job does not specify a limit, and there is no queue default. Format: “resources_default.resource_name=value[,...] Default value: no limit `Qmgr:` **`set server resources_default.mem=8mb`** `Qmgr:` **`set server resources_default.ncpus=1`**
resources_max	The maximum amount of each resource which can be requested by a single job executing on this Server if there is not a `resources_max` valued defined for the queue in which the job resides. Format: “resources_max.resource_name=value[,...] Default value: infinite usage `Qmgr:` **`set server resources_max.mem=1gb`** `Qmgr:` **`set server resources_max.ncpus=32`**
scheduler_iteration	The time, in seconds, between iterations of attempts by the Server to schedule jobs. On each iteration, the scheduler examines the available resources and runnable jobs to see if a job can be initiated. This examination also occurs whenever a running job terminates or a new job is placed in the queued state in an execution queue. Format: integer seconds Default value: 10 minutes `Qmgr:` **`set server scheduler_iteration=300`**
scheduling	Controls if the Server will request job scheduling by the PBS Scheduler. If true, the scheduler will be called as required; if

false, the Scheduler will not be called and no job will be placed into execution unless the Server is directed to do so by a PBS operator or Manager. Setting or resetting this attribute to `true` results in an immediate call to the Scheduler.
Format: boolean (see `acl_host_enable`)
Default value: value of `-a` option when Server is invoked, if `-a` is not specified, the value is recovered from the prior Server run. If it has never been set, the value is “false”.
`Qmgr: set server scheduling=true`

system_cost This attribute is currently unused.

The following attributes are read-only: they are maintained by the Server and cannot be changed by a client.

FLicenses Shows the number of floating licenses currently available.

PBS_version The release version number of the Server.

resources_assigned
The total amount of certain resources allocated to running jobs.

server_name The name of the Server, which is the same as the host name. If the Server is listening to a non-standard port, the port number is appended, with a colon, to the host name. For example: `host.domain:9999`

state_count The total number of jobs managed by the Server currently in each state

server_state The current state of the Server. Possible values are:

State	**Meaning**
`Active`	The Server is running and will invoke the Scheduler as required to schedule jobs for execution.
`Idle`	The Server is running but will not invoke the Scheduler
`Scheduling`	The Server is running and there is an outstanding request to the job scheduler

State	Meaning
`Terminating`	The Server is terminating. No additional jobs will be scheduled
`Terminating, Delayed`	The Server is terminating in delayed mode. The Server will not run any new jobs and will shutdown when the last currently running job completes.

total_jobs	The total number of jobs currently managed by the Server.

7.5 Queue Attributes

Once you have the Server attributes set the way you want them, you will next want to review the queue attributes. The default (binary) installation creates one queue with the required attributes, as shown in the example below.

You may wish to change these settings or add other attributes or add additional queues. The following discussion will be useful in modifying the PBS queue configuration to best meet your specific needs.

```
% qmgr
Qmgr: print queue work
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
#
Qmgr:
```

There are two types of queues defined by PBS: routing and execution. A routing queue is a queue used to move jobs to other queues including those which exist on different PBS Servers. Routing queues are similar to the old NQS pipe queues. A job must reside in an execution queue to be eligible to run. The job remains in the execution queue during the time it is running. In spite of the name, jobs in a queue need not be processed in queue-order (first-come first-served or *FIFO*). A Server may have multiple queues of either or both types, but there must be at least one queue defined. Typically it will be an execution queue; jobs cannot be executed while residing in a routing queue.

Queue attributes fall into three groups: those which are applicable to both types of queues, those applicable only to execution queues, and those applicable only to routing queues. If an “execution queue only” attribute is set for a routing queue, or vice versa, it is simply ignored by the system. However, as this situation might indicate the administrator made a mistake, the Server will issue a warning message about the conflict. The same message will be issued if the queue type is changed and there are attributes that do not apply to the new type.

Queue public attributes are alterable on request by a client. The client must be acting for a user with Manager or Operator privilege. Certain attributes require the user to have full administrator privilege before they can be modified. The following attributes apply to both queue types:

acl_group_enable	Attribute which when true directs the Server to use the queue’s group access control list `acl_groups`. Default value: false = disabled `Qmgr: set queue QNAME acl_group_enable=true`
acl_groups	List which allows or denies enqueuing of jobs owned by members of the listed groups. The groups in the list are groups on the Server host, not submitting host. Format: “[+|-]group_name[,...]” Default value: all groups allowed `Qmgr: set queue QNAME acl_groups=math,physics`
acl_host_enable	Attribute which when true directs the Server to use the `acl_hosts` access list. Format: boolean Default value: disabled `Qmgr: set queue QNAME acl_host_enable=true`
acl_hosts	List of hosts which may enqueue jobs in the queue. Format: “[+|-]hostname[...]” Default value: all hosts allowed `Qmgr: set queue QNAME acl_hosts=sol,star`
acl_user_enable	Attribute which when true directs the Server to use the `acl_users` access list for this queue. Format: boolean (see acl_group_enable); Default value: disabled `Qmgr: set queue QNAME acl_user_enable=true`

acl_users

List of users allowed or denied the ability to enqueue jobs in this queue.
Format: “[+|-]user[@host][,...]”
Default value: all users allowed
`Qmgr: set queue QNAME acl_users=sue,bob@star`

enabled

Queue will or will not accept new jobs. When false the queue is *disabled* and will not accept jobs.
Format: boolean
Default value: disabled
`Qmgr: set queue QNAME enabled=true`

from_route_only

When true, this queue will accept jobs only when being routed by the Server from a local routing queue. This is used to force user to submit jobs into a routing queue used to distribute jobs to other queues based on job resource limits.
Default value: disabled
`Qmgr: set queue QNAME from_route_only=true`

max_queuable

The maximum number of jobs allowed to reside in the queue at any given time.
Default value: infinite
`Qmgr: set queue QNAME max_queuable=200`

max_running

The maximum number of jobs allowed to be selected from this queue for routing or execution at any given time. For a routing queue, this is enforced by the Server, if set.
`Qmgr: set queue QNAME max_running=16`

max_user_res.*RES*

The maximum amount of resource *RES* that any single user may consume. *RES* can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be both *hard* and *soft*. (See “max_user_res.RES” on page 69 for discussion of *hard* vs. *soft* limits.)
Format: resource specific
Default value: none
`Qmgr: set queue QNAME max_user_res.ncpus=6`
`Qmgr: set queue QNAME max_user_res_soft.ncpus=3`

priority

The priority of this queue against other queues of the same type on this Server. (A larger value is higher priority than a smaller value.) May affect job selection for execution/routing.
`Qmgr: set queue QNAME priority=123`

queue_type

The type of the queue: execution or route. This attribute must be explicitly set.
Format: “execution”, “e”, “route”, “r”.

```
Qmgr: set queue QNAME queue_type=route
Qmgr: set queue QNAME queue_type=execution
```

resources_max

The maximum amount of each resource which can be requested by a single job in this queue. The queue value supersedes any Server wide maximum limit.

```
Qmgr: set queue QNAME resources_max.mem=2gb
Qmgr: set queue QNAME resources_max.ncpus=32
```

resources_min

The minimum amount of each resource which can be requested by a single job in this queue.

```
Qmgr: set queue QNAME resources_min.mem=1b
Qmgr: set queue QNAME resources_min.ncpus=1
```

resources_default

The list of default resource values which are set as limits for a job residing in this queue and for which the job did not specify a limit. If not set, the default limit for a job is determined by the first of the following attributes which is set: Server’s `resources_default`, queue’s `resources_max`, Server’s `resources_max`. An unset resource is viewed as unlimited.
Format: “resources_default.*resource_name*=*value*”
Default value: none

```
Qmgr: set queue QNAME resources_default.mem=1b
Qmgr: set queue QNAME resources_default.ncpus=1
```

started

Jobs may be scheduled for execution from this queue. When false, the queue is considered *stopped*.

```
Qmgr: set queue QNAME started=true
```

The following attributes apply only to execution queues:

checkpoint_min

Specifies the minimum interval of cpu time, in minutes, which is allowed between checkpoints of a job. If a user specifies a time less than this value, this value is used instead.

```
Qmgr: set queue QNAME checkpoint_min=5
```

kill_delay

The amount of the time delay between the sending of SIGTERM and SIGKILL when a `qdel` command is issued against a running job.

Format: integer seconds
Default value: 2 seconds
`Qmgr: set queue QNAME kill_delay=5`

max_user_run	The maximum number of jobs owned by a single user that are allowed to be running from this queue at one time. `Qmgr: set queue QNAME max_user_run=5`
max_group_run	The maximum number of jobs owned by any users in a single group that are allowed to be running from this queue at one time. `Qmgr: set queue QNAME max_group_run=20`
resources_available	The list of resource and amounts available to jobs running in this queue. The sum of the resource of each type used by all jobs running from this queue cannot exceed the total amount listed here. `Qmgr: set queue QNAME resources_available.mem=1gb`

The following attributes apply only to route queues:

alt_router	If true, a site-supplied alternative job router function is used to determine the destination for routing jobs from this queue. Otherwise, the default, round-robin router is used. `Qmgr: set queue QNAME alt_router=true`
route_destinations	The list of destinations to which jobs may be routed. Default value: none, should be set to at least one destination. `Qmgr: set queue QNAME route_destinations=QueueTwo`
route_held_jobs	If true, jobs with a hold type set may be routed from this queue. If false, held jobs are not to be routed. `Qmgr: set queue QNAME route_held_jobs=true`
route_lifetime	The maximum time a job is allowed to exist in a routing queue. If the job cannot be routed in this amount of time, the job is aborted. If unset, the lifetime is infinite. Format: integer seconds Default infinite `Qmgr: set queue QNAME route_lifetime=600`
route_retry_time	Time delay between route retries. Typically used when the network between servers is down.

Format: integer seconds
Default value: PBS_NET_RETRY_TIME (30 seconds)

```
Qmgr: set queue QNAME route_retry_time=120
```

route_waiting_jobs

If true, jobs with a future `execution_time` attribute may be routed from this queue. If false, they are not to be routed.

```
Qmgr: set queue QNAME route_waiting_jobs=true
```

The following data items are read-only attributes of the queue. They are visible to client commands, but cannot be changed by them.

hasnodes — If true, indicates that the queue has nodes associated with it.
Format: boolean

total_jobs — The number of jobs currently residing in the queue.

state_count — Total number of jobs currently residing in the queue in each state.

resources_assigned

The total amount of certain types of resources allocated to jobs running from this queue.

Important: Note, an *unset* resource limit for a job is treated as an infinite limit.

7.6 Nodes

Where jobs will be run is determined by an interaction between the Scheduler and the Server. This interaction is affected by the contents of the PBS `nodes` file, and the system configuration onto which you are deploying PBS. Without this list of nodes, the Server will not establish a communication stream with the MOM(s) and MOM will be unable to report information about running jobs or notify the Server when jobs complete. If the PBS configuration consists of a single timeshared host on which the Server and MOM are running, all the jobs will run there. The Scheduler only needs to specify which job it wants run.

If you are running a timeshared complex with *one* or more execution hosts, then distributing jobs across the various hosts is a matter of the Scheduler determining on which host to place a selected job.

If your cluster is made up of cluster nodes and you are running distributed (multi-node) jobs, as well as serial jobs, the Scheduler typically uses the *Query Resource* or *Avail* request to the Server for each queued job under consideration. The Scheduler then selects one of the jobs that the Server replied could run, and directs that the job should be run. The Server will then allocate one or more virtual processors on one or more nodes as required to the job.

If your system contains both cluster nodes and one timeshared node, the situation is like the above, except you may wish to change the value of `default_node` to be that of the timeshared host. Jobs that do not ask for nodes will end up running on the timeshared host.

If you have a configuration supporting both cluster nodes and multiple time shared hosts, you have a complex system. The Scheduler must recognize which jobs request nodes and use the *Avail* request to the Server. It must also recognize which jobs are to be balanced among the timeshared hosts, and provide the host name to the Server when directing that the job be run. The Standard scheduler does this.

Important: Regardless of node type, each node must be defined in the Server’s nodes file, `/usr/spool/PBS/server_priv/nodes`. Time-shared nodes have `:ts` appended to their node name. Cluster nodes have no name suffix.

In PBS, allocation of cluster nodes (actually the allocation of virtual processors, VPs, of the nodes) to a job is handled by the Server. Each node must have its own copy of MOM running on it. If only timeshared hosts are to be served by the PBS batch system, the Job Scheduler must direct where the job should be run. If unspecified, the Server will execute the job on the host where it is running.

7.6.1 PBS Nodes File

A basic nodes file is created for you by the install procedure. This file contains only the name of the host from which the install was run and set as a time-shared host. If you have more than one host in your PBS cluster or you are not planning on running jobs on the Server’s host, you need to edit the list of nodes to reflect your site.

You may edit the nodes list in one of two ways. If the Server is not running, you may directly edit the `nodes` file with a text editor. If the Server is running, you should use `qmgr` to edit the list of nodes.

The *node list* is defined to the Server in the file `/usr/spool/PBS/server_priv/`

`nodes`. This is a simple text file with the specification of a single node per line in the file. The format of each line in the file is:

```
node_name[:ts] [attributes]
```

The *node name* is the network name of the node (host name), it does not have to be fully qualified (in fact it is best if it is as short as possible). The optional `:ts` appended to the name indicates that the node is a timeshared node.

Nodes can have attributes associated with them. Attributes come in three types: properties, name=value pairs, and name.resource=value pairs. Zero or more properties may be specified. The *property* is nothing more than a string of alphanumeric characters (first character must be alphabetic) without meaning to PBS. Properties are used to group classes of nodes for allocation to a series of jobs.

Any legal node name=value pair may be specified in the node file in the same format as on a `qsub` directive: `attribute.resource=value`. For example:

```
NodeA resource_available.ncpus=3 max_running=1
```

The expression `np=`*`NUMBER`* may be used within the `nodes` file as shorthand for `resources_available.ncpus=NUMBER`, which can be added to declare the number of virtual processors (VPs) on the node. *NUMBER* is a numeric string, for example `np=4`. This expression will allow the node to be allocated up to *NUMBER* times to one job or more than one job. If `np=`*`NUMBER`* is not specified for a cluster node, it is assumed to have one VP.

Important: Note that if these values are not explicitly set, they will take the value provided by MOM. But if they are explicitly set, that set value will be carried forth even across Server restarts.

These include:

```
resources_available.ncpus
resources_available.arch
resources_available.mem
```

Each item on the line must be separated by white space. The items may be listed in any order, except that the host name must always be first. Comment lines may be included if the first non-white space character is the pound sign ’#’. The following is an example of a possible nodes file for a cluster called “planets”:

```
# The first set of nodes are cluster nodes.
# Note that the properties are provided to
# logically group certain nodes together.
# The last node is a timeshared node.
#
mercury    inner moonless
venus      inner moonless np=1
earth      inner np=1
mars       inner np=2
jupiter    outer np=18
saturn     outer np=16
uranus     outer np=14
neptune    outer np=12
pluto:ts
```

7.6.2 Creating or Adding nodes:

After `pbs_server` is started, the node list may be entered or altered via the `qmgr` command:

```
    create node node_name [attribute=value]
```

where the attributes and their associated possible values are shown in the table below.

The `busy` state is set by the execution daemon, `pbs_mom`, when a load-average threshold is reached on the node. See *max_load* in MOM’s config file (“Static Resources” on page 106). The `job-exclusive` and `job-sharing` states are set when jobs are running on the node.

Important: All comma separated strings must be enclosed in quotes.

Below are several examples of setting node attributes via `qmgr`.

```
% qmgr
Qmgr: create node mars resources_available=2,ntype=cluster
Qmgr: create node venus properties=”inner,moonless”
```

Modify nodes: Once a node has been created, its attributes and/or properties can be modified using the following `qmgr` syntax:

`set node` *`node_name`* `[attribute[+|-]=value]`

where attributes are the same as for `create`. For example:

```
% qmgr
Qmgr: set node mars properties=inner
Qmgr: set node mars properties+=haslife
```

Delete nodes: Nodes can be deleted via `qmgr` as well, using the `delete node` syntax, as the following example shows:

```
% qmgr
Qmgr: delete node mars
Qmgr: delete node pluto
```

7.7 Node Attributes

A node has the following public attributes:

state — The state of the node, one of: `free, down, offline`
Format: string
`Qmgr: set node MyNode state=offline`

properties — Any alphanumeric string or comma separated set of strings, starting with an alphabetic character.
Format: string
`Qmgr: set node MyNode properties=”red,blue”`

ntype — The type of the node, one of : `cluster, time-shared`
Format: string
`Qmgr: set node MyNode ntype=cluster`

resources_available — List of resources available on node.
Format: resource list

resources_assigned — List of resources in use on node.
Format: resource list

max_running	Maximum number of running jobs; advisory to scheduler. Format: integer `Qmgr: set node MyNode max_running=22`
max_user_run	Maximum number of running jobs per user; advisory. Format: integer `Qmgr: set node MyNode max_user_run=4`
max_group_run	Maximum number of running jobs per group; advisory. Format: integer `Qmgr: set node MyNode max_group_run=8`
queue	Name of an execution queue (if any) associated with the node. Only jobs from the named queue will be run on associated node. Note: a node can be associated with one and only one queue. Format: queue specification `Qmgr: set node MyNode queue=MyQueue`
reservations	List of reservations pending on the node; read-only. Format: reservation specification
comment	General comment, e.g. the reason the node is marked down or off-line; can be set by Manager or Operator. Format: string `Qmgr: set node MyNode comment=”Down until 5pm”`

A node has the following read-only attributes:

pcpus	Shows the number of physical CPUs on the node, which determine the number of licenses required for that node.
license	Indicates the node "license state" as a single character, according to the following table:

`u`	unlicensed
`l`	licensed with a node lock or fixed license
`f`	licensed with a floating license

7.7.1 Setting Node Limits

It is possible to set limits on queues (and the Server) as to how many nodes a job can request. The `nodes` resource itself is a text string and difficult to limit. Thus, `nodect` (node count) is set by the Server to the integer number of nodes desired by the user as declared in the "`nodes`" resource specification. That declaration is parsed and the resulting total number of nodes is set in `nodect`. This is useful when an administrator wishes to place an integer limit, `resources_min` or `resources_max` on the number of nodes used by a job entering a queue.

Based on the earlier example of declaring nodes, if a user requested a nodes specification of: `3:inner+2:outer`, then `nodect` would get set to 5 (i.e. 3+2). Another read-only job attribute, `neednodes`, is initially set by the Server to the same value as `nodes`. `Neednodes` may be modified by the job Scheduler for special policies. The contents of `neednodes` determines which nodes are actually assigned to the job. `Neednodes` is visible to the Manager but not to an unprivileged user.

If you wish to set up a queue default value for "`nodes`" (a value to which the resource is set if the user does not supply one), corresponding default values must be set for "`nodect`" and "`neednodes`". For example:

```
% qmgr
Qmgr: set queue small resources_default.nodes=1:inner
Qmgr: set queue small resources_default.nodect=1
Qmgr: set queue small resources_default.neednodes=1:inner
```

Minimum and maximum limits are set for "`nodect`" only. For example:

```
% qmgr
Qmgr: set queue small resources_min.nodect=1
Qmgr: set queue small resources_max.nodect=15
Qmgr:
```

Important: Minimum and maximum values must **not** be set for either `nodes` or `neednodes` as their value are strings.

7.7.2 Nodes Specification Information

This section provides additional information about working with the PBS nodes file and nodes specification.

Step 1 If a single specific host is named in the Run Job request and the host is specified in the nodes file as a *timeshared* host, the Server will attempt to run the job on that host.

Step 2 If either:

(a) a specific host is named in the Run Job and the named node does not appear in the Server's `nodes` file as a timeshared host;

or

(b) a "+" separated list of hosts [or node properties] is specified in the Run Job request;

then, the Server attempts to allocate one (or more as requested) virtual processor on the named *cluster* node(s)named in the job. All of the named nodes have to appear in the Server's `nodes` file. If the allocation succeeds, the job [shell script] is run on the first of the nodes allocated.

Step 3 If no location was specified on the Run Job request, but the job requests nodes, then the required number of virtual processors on cluster nodes which match the request are allocated if possible. If the allocation succeeds, the job is run on the node allocated to match the first specification in the node request. Note, the Scheduler may modify the job's original node request, see the job attribute `neednodes`

For SMP nodes, where multiple virtual processors have been declared, the order of allocation of processors is controlled by the setting of the Server attribute `node_pack`

If set true, VPs will first be taken from nodes with the fewest free VPs. This *packs* jobs into the fewest possible nodes, leaving nodes available with many VPs for those jobs that need many VPs on a node.

If `node_pack` is set false, VPs are allocated from nodes with the most free VPs. This scatters jobs across the nodes to minimize conflict between jobs. If `node_pack` is not set to either true or false, i.e. *unset* then the VPs are allocated in the order that the nodes are declared in the Server's nodes file.

Important: Be aware that if `node_pack` is set, the internal order of nodes is changed. If `node_pack` is later unset, the order will no longer be changed, but it will not be in the order originally established in the `nodes` file.

Step 4 If the Server attribute `default_node` is set, its value is used. If it matches the name of a time-shared node, the job is run on that node. If the value can be mapped to a set of one or more free cluster nodes, they are allocated to the job.

Step 5 If `default_node` is not set, and at least one time-shared node is defined, that node is used. If more than one is defined, the first is selected for the job.

Step 6 The last choice is to act as if the job has requested `1#shared`. The job will have allocated to it an existing job-shared VP, or if none exists, then a free VP is allocated as job-shared.

The `exec_host` string and the `runjob` destination string is now of the form:

```
nodename/P[*C][+nodename/P[*C]...]
```

where P is the process index ranging from 0 to P-1, the number of requested processes on the node; and C is the number of CPUs per process. For example, a request of

```
-l nodes=alpha:ppn=2:ncpus=2+beta
```

would result in a execution host string of

```
alpha/0*2+beta/0+alpha/1*2
```

This allows the unique nodes to be ordered first in case a user wishes to run a single task per node to setup files, etc.

7.7.3 Node Comments

Nodes have a "comment" attribute which can be used to display information about that node. If the comment attribute has not be explicitly set by the PBS Manager and the node is down, it will be used by the PBS Server to display the reason the node was marked down. If the Manager has explicitly set the attribute, the Server will not overwrite the comment. The comment attribute may be set via the `qmgr` command:

```
% qmgr
Qmgr: set node pluto comment=”node will be up at 5pm”
```

7.8 SGI Weightless CPU Support

Submitting a job and requesting `-l ncpus=0` is now legal. In a non-cpuset SGI Irix 6.x environment, the job's kernel scheduling priority will be set "weightless". There will no allocation at the Server, Queue, or Node level of CPUs; i.e. `resources_assigned.ncpus` will not be incremented for this job.

Important: Because `ncpus=0` has no useful effect on any other system and can result in allowing too many jobs to be run, it is **strongly** recommended that jobs not be allowed to be submitted with `ncpus=0`. This may be done by setting a Server level resource default and a resources minimum via the `qmgr` command:

```
set server resources_default.ncpus=1
set queue q1 resources_min.ncpus=1
set queue q2 resources_min.ncpus=1
```

7.9 Job Attributes

Tabular listing of attributes of a PBS job are given in the **PBS User Guide** under the heading “Job Attributes”.

7.10 PBS Resources

PBS has a standard set of "resources" which may be specified as a job requirement. Common examples of the predefined standard resources are:

cput	amount of CPU time
mem	amount of real memory
ncpus	number of CPUs
nodes	number and type of execution nodes
walltime	amount of real clock time

Important: A complete listing of available PBS resources is given in Chapter 4 of the **PBS User Guide**.

Depending on site policy, the required resources may be considered against the availability of the resources to determine ordering of jobs for execution and placement of jobs on an execution host.

7.10.1 Defining New Resources

It is possible for the PBS Manager to define new resources within PBS. Once created, jobs may request these new resources and the Scheduler can be directed to consider the new resources in the scheduling policy. (See section "Dynamic Consumable Resources" on page 133 for instructions on how to configure the Scheduler to use the new resources you create.) To define one or more new resources, the Administrator creates a file, `(PBS_HOME)/server_priv/resourcedef`. Each line in the file defines a resource. The format of each line is:

```
RESOURCE_NAME [type=RTYPE] [flag=FLAGS]
```

`RESOURCE_NAME` is any string made up of alphanumeric characters, starting with a alphabetic character. The underscore character, "_", and the hyphen, "-", are also allowed.

`RTYPE` is the type of the resource value, one of the following key words:

long	the value is a long integer
float	the value is a floating point number
size	the value is a integer number following by a suffix denoting magnitude `k`, `m`, `g`, `t` and `b` for bytes or `w` for words.
string	a null terminated string

If not specified, the resource will default to type `long`.

`FLAGS` is a set of characters which indicate if the Server should accumulate the requested amounts of the resource in the attribute `resources_assigned` when the job is run. The value of `flag` is the concatenation of one or more of the following letters:

q	the amount is tracked at the Queue and Server level
n	the amount is tracked at the Node level, for all nodes assigned to the job
f	the amount is tracked at the Node level for only the first node allocated to the job

If not specified, the resource will not be accumulated in `resources_assigned`. For example, if the Administrator created the following lines in `/usr/spool/pbs/server_priv/resourcedef`:

```
kolas   type=long       flag=q
pandas  type=size       flag=qn
kiwi    type=size       flag=f
wombat  type=string     flag=q
```

A user may specify any of these resources on job submission, as the following example shows:

```
% qsub -l kolas=4,pandas=10gb,wombat=brown \
  -l kiwi=3kb -lnodes=2  script
```

If the job resides in the execution queue "workq" and is executing on nodes NodeA and NodeB, then

queue "workq" would show:

```
resources_assigned.kolas=4
resources_assigned.pandas=10gb
resources_assigned.wombat=brown
```

node NodeA, the first node allocated, would show:

```
resources_assigned.kiwi=3gb
resources_assigned.pandas=10gb
```

node NodeB would show:

```
resources_assigned.pandas=10gb
```

7.10.2 Resource Min/Max Attributes

Minimum and maximum queue and Server limits work with numeric valued resources, including time and size values. Generally, they do not work with string valued resources because of character comparison order. However, setting the `min` and `max` to the same value to force an exact match will work even for string valued resources.

```
% qmgr
Qmgr: set queue big resources_max.arch=unicos8
Qmgr: set queue big resources_min.arch=unicos8
Qmgr:
```

The above example can be used to limit jobs entering queue `big` to those specifying `arch=unicos8`. Again, remember that if `arch` is not specified by the job, the tests pass automatically and the job will be accepted into the queue.

7.11 Advanced Configuration Options

This section discusses several advanced configuration options, and provides, supplemental information for configuration of the PBS Server.

7.11.1 Selective Routing of Jobs into Queues

Often it is desirable to route jobs to various queues on a Server, or even between Servers, based on the resource requirements of the jobs. The queue *resources_min* and *resources_max* attributes discussed above make this selective routing possible. As an example, let us assume you wish to establish two execution queues, one for short jobs of less than one minute cpu time, and the other for long running jobs of one minute or longer. Call them `short` and `long`. Apply the `resources_min` and `resources_max` attribute as follows:

```
% qmgr
Qmgr: set queue short resources_max.cput=59
Qmgr: set queue long resources_min.cput=60
```

When a job is being enqueued, it's requested resource list is tested against the queue limits: `resources_min` <= job_requirement <= `resources_max`. If the resource test fails, the job is not accepted into the queue. Hence, a job asking for 20 seconds of cpu time would be accepted into queue `short` but not into queue `long`.

Important: Note, if the `min` and `max` limits are equal, only that exact value will pass the test.

You may wish to set up a routing queue to direct jobs into the queues with resource limits. For example:

```
% qmgr
Qmgr: create queue funnel queue_type=route
Qmgr: set queue funnel route_destinations =”short,long”
Qmgr: set server default_queue=funnel
```

A job will end up in either `short` or `long` depending on its cpu time request.

You should always list the destination queues in order of the most restrictive first as the first queue which meets the job's requirements will be its destination (assuming that queue is enabled). Extending the above example to three queues:

```
% qmgr
Qmgr: set queue short resources_max.cput=59
Qmgr: set queue long resources_min.cput=1:00
Qmgr: set queue long resources_max.cput=1:00:00
Qmgr: create queue huge queue_type=execution
Qmgr: set queue funnel route_destinations="short,long,huge"
Qmgr: set server default_queue=funnel
Qmgr:
```

A job asking for 20 minutes (20:00) of cpu time will be placed into queue `long`. A job asking for 1 hour and 10 minutes (1:10:00) will end up in queue `huge` by default.

Important: If a test is being made on a resource as shown with `cput` above, and a job does not specify that resource item (it does not appear in the `-l resource=`*`valuelist`* on the `qsub` command, the test will pass. In the above case, a job without a cpu time limit will be allowed into queue `short.` For this reason, together with the fact that an unset limit is considered to be an infinite limit, you may wish to add a default value to the queues or to the Server.

```
% qmgr
Qmgr: set queue short resources_default.cput=40
or
Qmgr: set server resources_default.cput=40
```

Either of these examples will ensure that a job without a cpu time specification is limited to 40 seconds. A `resources_default` attribute at a queue level only applies to jobs in that queue.

Important: Be aware of several important facts:

If a default value is assigned, it is done so after the tests against `min` and `max`. Default values assigned to a job from a queue `resources_default` are not carried with the job if the job moves to another queue. Those resource limits becomes unset as when the job was specified. If the new queue specifies default values, those values are assigned to the job while it is in the new queue. Server level default values are applied if there is no queue level default.

7.11.2 Recording Server Configuration

If you should you wish to record the configuration of a PBS Server for re-use later, you may use the `print` subcommand of `qmgr`(8B). For example,

```
% qmgr -c "print server" > /tmp/server.con
```

will record in the file `/tmp/server.con` the `qmgr` subcommands required to recreate the current configuration including the queues. The commands could be read back into `qmgr` via standard input:

```
% qmgr < /tmp/server.con
```

Node configuration information is not printed. To save the current node configuration information, make a copy of the `server_priv/nodes` file.

7.11.3 Server Support for Globus

If Globus support is enabled, then an entry must be made in the PBS node file with `:gl` appended to the name. This is the only case in which two nodes may be defined with the same node name. One may be a Globus node (MOM), and the other a non-Globus node.

Important: If you run both a Globus MOM and a normal MOM on the same site, the normal PBS MOM must be listed first in your nodes file. If not, some scheduling anomalies could appear.

Chapter 8

Configuring MOM

The execution server daemons, MOMs, require much less configuration than does the Server. The installation process creates a basic MOM configuration file which contains the minimum entries necessary in order to run PBS jobs. This chapter describes the MOM configuration file, and explains all the options available to customize the PBS installation to your site.

8.1 MOM Config File

The behavior of MOM is controlled via a configuration file which is read upon daemon initialization (start-up) and upon re-initialization (when `pbs_mom` receives a SIGHUP signal).

If the `-c` option is not specified when MOM is started, she will open the `config` file in the path specified in `/etc/pbs.conf`, if it exists. If it does not, MOM will continue anyway. This file may be placed elsewhere or given a different name, in which case `pbs_mom` must be started with the `-c` option with the new file name and path specified.

The configuration file provides several types of run time information to MOM: access control, 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 functions at initialization (and re-initialization).

Each configuration entry is on a single line with the component parts separated by white space. If the line starts with a pound sign ("#"), the line is considered to be a comment and is ignored.

The installation process creates a MOM configuration file with the following entries, which are explained in detail in the subsequent sections of this chapter.

```
$logevent 0x1ff
$clienthost server-hostname
```

8.1.1 MOM Initialization Values

An initialization value directive has a name which starts with a dollar sign ("$") and must be known to MOM via an internal table. (Directive that are specific to a single operating systems, such as SGI IRIX, do not start with a dollar sign ("$"). Currently the permitted initialization directives are:

`alloc_nodes_greedy`
On cpuset-enabled SGI systems, requests to allocate nodes close together (1) or anywhere (0, the default).

```
alloc_nodes_greedy 1
```

`checkpoint_upgrade`
If present, causes PBS to pass a special upgrade checkpoint flag to the SGI IRIX checkpoint system for use immediately prior to an Irix operating system upgrade. For details on use, see "Checkpointing Jobs Prior to SGI IRIX Upgrade" on page 154.

`$checkpoint_path`
If present, directs MOM to use the pathname following this directive as the location into which to write checkpoint files. (See also "Checkpoint/Restart Under PBS" on page 152.)

```
$checkpoint_path /path/to/ckpdevice
```

`$clienthost` Causes a host name to be added to the list of hosts which will be allowed to connect to MOM. Two host names are always allowed to connect to MOM: "localhost" and the name returned by the system call gethostname(). These names need not be specified in the configuration file. The Server's host must be either the same as MOM or be listed as a `clienthost` entry in each MOM's `config` file. Upon startup, MOM will send a restart notice to the Server. The IP addresses of all the hosts (nodes) in the Server `nodes` file will be forwarded by the Server to the MOM on each host listed in the `nodes` file. These hosts need not be in the various MOM's configuration file as they will be added internally when the list is received from the Server.

The hosts which are provided by the Server to MOM comprise a *sisterhood* of hosts. Any one of the sisterhood will accept connections from a Scheduler [*Resource Monitor* (RM) requests] or Server [jobs to execute] from within the sisterhood. They will also accept *Internal MOM* (IM) messages from within the sisterhood. For a sisterhood to be able to communicate IM messages to each other, they must all share the same RM port. For example, here are two lines for the configuration file which will allow the hosts "phobos" and "deimos" to connect to MOM:

```
$clienthost phobos
$clienthost deimos
```

`$cputmult` Sets a factor used to adjust cpu time used by a job. This is provided to allow adjustment of time charged and limits enforced where the job might run on systems with different cpu performance. If MOM's system is faster than the reference system, set `$cputmult` to a decimal value greater than 1.0. If MOM's system is slower, set `$cputmult` to a value between 1.0 and 0.0. The value is given by

value = speed_of_this_system / speed_of_reference_system

For example:

```
$cputmult 1.5
or
$cputmult 0.75
```

`cpuset_create_flags` *`arg`*

On cpuset-enabled SGI systems, defines whether to allocate either memory or CPUs in an exclusive fashion.

```
cpuset_create_flags CPUSET_CPU_EXCLUSIVE
cpuset_create_flags CPUSET_MEMORY_EXCLUSIVE
cpuset_create_flags CPUSET_POLICY_PAGE
```

`cpuset_destroy_delay` *`seconds`*
On cpuset-enabled SGI systems, specifies the number of seconds to wait before tearing down (deleting) a cpuset.

`cpuset_small_mem` *`arg`*
On cpuset-enabled SGI systems, disables running jobs on shared cpusets when set to “0”.

`cpuset_small_ncpus` *`arg`*
On cpuset-enabled SGI systems, disables running jobs on shared cpusets when set to “0”.

`$enforce` *`arg`* Specifies site-specific resource enforcement behavior of `mem` or `ncpus`. (See the `pbs_mom` manual page for details.) Arguments to the `enforce` directive include the following:

Attribute	**Type**	**Description**
`average_cpufactor`	float	ncpus factor to allow; default: 1.025
`average_percent_over`	int	percentage over the limit to allow; default: 50
`average_trialperiod`	int	minimum walltime before enforcement; default: 120 seconds
`cpuaverage`	boolean	if present, enforce this limit; default: off
`cpuburst`	boolean	if present, enforce this limit; default: off
`complexmem`	boolean	if present, enforce this limit; default: off
`delta_cpufactor`	float	ncpus factor to allow; default: 1.5
`delta_percent_over`	int	percentage over the limit to allow; default: 50

`delta_weightup`	float	weighting when average is moving up; default: 0.4
`delta_weightdown`	float	weighting when average is moving down; default: 0.1
`mem`	boolean	if present, enforce this limit; default: off

`$ideal_load` Declares the low water mark for load on a node. It works in conjunction with a `$max_load` directive. When the load average on the node drops below the `ideal_load`, MOM on that node will inform the Server that the node is no longer busy. For example:

```
$ideal_load 2.0
$max_load 3.5
```

`$kbd_idle` Enables support for "idle workstation cycle harvesting" as described later in this chapter. Details are provided on page 108.

```
$kbd_idle 1800 10 5
```

`$logevent` Sets the mask that determines which event types are logged by pbs_mom. For example:

```
$logevent 0x1ff

$logevent 255
```

The first example would set the log event mask to 0x1ff (511) which enables logging of all events including debug events. The second example would set the mask to 0x0ff (255) which enables all events except debug events. The values of events are listed in section 11.14 "Use and Maintenance of Logfiles" on page 163.

`$max_load` Declares the high water mark for load on a node. It is used in conjunction with a `$ideal_load` directive. When the load average exceeds the high water mark, MOM on that node will notify the Server that the node is busy. The state of the node will be shown as busy. A busy cluster node will not be allocated to jobs. This is useful in preventing allocation of jobs to nodes which are busy with interactive sessions.

A busy time-shared node may still run new jobs under the direction of the Scheduler. Both the $ideal_load and $max_load directives add a static resource, ideal_load and max_load, which may be queried by the Scheduler. These static resources are supported by the Standard scheduler when load-balancing jobs. Note that the Scheduler will attempt to predict the load on the node prior to running a job, by taking the current load, and adding the number of CPUs requested by a job being considered. If the result is greater than max_load, then the job will not be run.

max_shared_nodes

Specifies the number of nodes that should be allocated to shared jobs with cpuset-enabled SGI IRIX systems.

$prologalarm

Sets the time-out period in seconds for the prologue and epilogue scripts. (See discussion of the prologue and epilogue in section 11.13 “Job Prologue/Epilogue Scripts/Programs” on page 160.) An alarm is set to prevent the script from locking up the job if the script hangs or takes a very long time to execute. The default value is 30 seconds. An example:

```
$prologalarm 60
```

$restricted

Causes a host name to be added to the list of hosts which will be allowed to connect to MOM without needing to use a privileged port. The name specification allows for wildcard matching. Connections from the specified hosts are restricted in that only internal queries may be made. Only static resources from the config file will be reported and no control requests can be issued. This is to prevent any shell commands from being run by a non-root process.

This type of entry is typically used to specify hosts on which a monitoring tool, such as xpbsmon, can be run. Xpbsmon will query MOM for general resource information.

For example, here is a configuration file line which will allow queries from any host from the domain “pbspro.com”:

```
$restricted *.pbspro.com
```

`$suspendsig` An alternative signal can be specified via this directive to be used for suspending jobs.

```
$suspendsig signal_number
```

`$usecp` Directs MOM to use `/bin/cp` instead of `rcp` or `scp` for delivery of output files. If MOM is to move a file to a host other than her own, MOM normally uses a remote copy command (`scp` or `rcp`) to transfer the file. This applies to stage-in/out and delivery of the job's standard output/error. The destination is recorded as `hostx:/full/path/name`. So if `hostx` is not the same system on which MOM is running, she uses `scp` or `rcp`; if it is the same system, MOM uses `/bin/cp`.

However, if the destination file system is NFS mounted among all the systems in the PBS environment (cluster), `cp` may work better than `scp/rcp`. One or more `$usecp` directives in the config file can be used to inform MOM about those file systems where the `cp` command should be used instead of `scp/rcp`. The `$usecp` entry has the form:

`$usecp` *`hostspec`*:*`path_prefix new_prefix`*

The *`hostspec`* is either a fully qualified host–domain name or a wild-carded host–domain specification as used in the Server's host ACL attribute. The *`path_prefix`* is the leading (root) component of the fully qualified path for the NFS files as visible on the specified host. The *`new_prefix`* is the initial components of the path to the same files on MOM's host. If different mount points are used, the `path_prefix` and the `new_prefix` will be different. If the same mount points are used for the cross mounted file system, then the two prefixes will be the same. When given a file destination, MOM will:

Step 1 Match the `hostspec` against her host name. If they match, MOM will use the `cp` command to move the file. If the `hostspec` is `"localhost"` then MOM will also use `cp`.

Step 2 If the match in step one fails, MOM will match the host portion of the destination against each `$usecp hostspec` in turn. If the host matches, MOM matches the `path_prefix` against the initial segment of the destination name. If this matches, MOM will discard the host name, replace the initial segment of the path that matched against `path_prefix` with the `new_prefix` and use `cp` with the resulting destination.

Step 3 If the host is neither the local host nor matches any of the `$usecp` directives, MOM will use the `scp` or `rcp` command to move the file.

For example, a user named Beth on host phobos.pbspro.com submits a job while her current working directory is:

```
/u/wk/beth/proj
```

The destination for her output would be given by PBS as:
`phobos.pbspro.com:/u/wk/beth/proj/123.OU`.

The job runs on node jupiter.pbspro.com which has the user's home file system cross mounted as `/r/home/beth`. Either of the following entries in the config file on node jupiter will result in a `cp` copy to `/r/home/beth/proj/123.OU` instead of an `rcp` copy to `phobos.pbspro.com:/u/wk/beth/proj/123.OU`

```
$usecp phobos.pbspro.com:/u/wk/ /r/home/

$usecp *.pbspro.com:/u/wk/ /r/home/
```

Note that the destination is matched against the `$usecp` entries in the order listed in the config file. The first match of host and file prefix determines the substitution. Therefore, if you have the same physical file system mounted as `/scratch` on node mars and as `/workspace` on every other host, then the entries in the config file on jupiter should be in the following order:

```
$usecp mars.pbspro.com:/scratch /workspace
$usecp *.pbspro.com:/workspace /workspace
```

`$wallmult` Sets a factor used to adjust wall time usage by a job to a common reference system. The factor is used for walltime calculations and limits in the same way as `$cputmult` is used for cpu time.

`$tmp` Specifies the location of tmpdir, where the PATH argument names a secure (not world writable) directory, such as /usr/tmp.

```
$tmp PATH
```

Important: In the above listing of MOM configuration parameters, only those that apply solely to SGI IRIX systems do not have a “`$`” prepended to the name.

8.2 Job Memory Limit Enforcement

Enforcement of the "mem" (physical memory) resource usage is available on all platforms. Enforcement is configurable by a entry in MOM’s `config` file. By default, enforcement is off. If a `$enforce mem` statement appears in the `config` file, then jobs that exceed their specified amount of physical memory will be killed. There are two items to be aware of:

1. "mem" is a per job/ per node limit.
2. "mem" enforcement is polled, therefore a job may exceed its limit for up to two minutes before it is detected.

8.3 SGI Non-cpuset Memory Enforcement

Under Irix 6.5.x, there are two ways to determine the amount of real memory a set of processes are using. The "simple" way, as used by the `ps`(1) command, looks solely at the `pr_rssize` field of the `/proc/pinfo/*` entry for each process. The "complex" method uses special SGI calls to determine the "shared" state of each memory segment in each process.

The "simple" method is quick and clean. However, this method does not factor in shared memory segments, so the resulting usage figure for processes that are started by the `sproc`(2) call is too high. The shared segments are counted fully against each process. This "apparent" over usage can result in under loading of the physical memory in the system.

The "complex" method correctly factors in the shared memory segments and yields a more accurate report on the amount of physical memory used. However, the SGI `ioctl(PIOCMAP_SGI)` call requires that the kernel look at each memory segment. This can result in the calling program, `pbs_mom`, being blocked for an extended period of time on larger systems. Systems smaller than 32 CPUs are not likely to see a problem.

Earlier versions of PBS shipped with the "simple" memory calculation compiled; the "complex" option was only available if you recompiled from source. In PBS Pro 5.2, the memory calculation option is run-time selectable. By default, the "simple" option is enabled. With the addition of a `$enforce complexmem` statement in MOM's `config` file, the "complex" memory usage calculation is selected.

Important: If the "complex" method is selected, the administrator needs to monitor the MOM logs for a warning of the form "time lag N secs" where N is a number of seconds greater than five. If this message appear frequently, it means the Irix kernel is taking that long to respond to the `ioctl` call and the performance of `pbs_mom` may suffer. In that case, it is recommended that the site revert to the "simple" calculation or run the cpuset version of MOM.

8.4 Job NCPUS Limit Enforcement

Enforcement of the `ncpus` (number of CPUs used) is now available on all platforms. Enforcement is configurable by a set of entries in MOM's `config` file. By default, enforcement is off since it has not been enforced in the past.

Associated with this enforcement is a new read-only resource `cpupercent`. This is a report of the average percentage usage of one CPU. For example, a value of 50 means that during a certain period, the job used 50 percent of one CPU. A value of 300 means over the period, the job used an average of three CPUs. Enforcement is based on two separate checks. The first check is based on the polled sum of CPU time for all processes in the job. Each poll period, the total CPU time used, divided by the total walltime, is checked against the value specified for `ncpus` * 100. If this "average" value exceeds the number of CPUs requested and if the `cpuaverage` enforcement is turned on, the job is aborted. The second check during each poll period looks for sudden bursts of CPU activity. `cpupercent` is a moving weighted average based on the prior `cpupercent` and the amount of new CPU time used divided by the walltime for this period. This value can be weighted to ignore or punish sudden bursts of CPU activity. This enforcement is available if `cpuburst` is set in MOM's `config` file.

The following parameters, set via a `$enforce` statement in the `config` file, control the enforcement, weighting, and allowed overrun. (Types and description of each are given in the discussion of `$enforce` under “MOM configuration options” on page 105 above.

For the absolute cpu time / walltime calculation, the following `enforce` arguments are used: `cpuaverage`, `average_trialperiod`, `average_percent_over`, `average_cpufactor`. Given the default values, a job will be killed if:

(cput / walltime) > ((ncpus * 100 * average_cpufactor) + average_percent_over

This enforcement only occurs after the job has had `average_trialperiod` seconds of walltime.

For the weighted moving average, the following `enforce` arguments are used: `cpuburst`, `delta_percent_over`, `delta_cpufactor`, `delta_weightup`, `delta_weightdown`.

The new reported `cpupercent` value is determined by weighing the new cput/walltime for the last measurement period and the old `cpupercent`:

new_percent = change_in_cpu_time / change_in_walltime
weight = delta_weight[up|down] * walltime/max_poll_period
new_cpupercent = (new_percent * weight) + (old_cpupercent * (1-weight))

`delta_weight_up` is used if the `new_percent` is higher than the old `cpupercent` value and `delta_weight_down` is used if `new_percent` is lower than the old value. If `delta_weight_[up|down]` is 0.0, then the value for `cpupercent` will not change with time. If it is 1.0, `cpupercent` will become the `new_percent` for the poll period; that means `cpupercent` changes quickly. (`max_poll_period` is the maximum time between samples, set to 120 seconds.) The job will be killed if

new_cpupercent > ((ncpus * 100 * delta_cpufactor) + delta_percent_over)

The following entries in MOM’s `config` file would turn on enforcement of both average and burst with the default values:

```
$enforce cpuaverage
$enforce cpuburst
$enforce delta_percent_over 50
$enforce delta_cpufactor 1.05
$enforce delta_weightup 0.4
$enforce delta_weightdown 0.1
$enforce average_percent_over 50
$enforce average_cpufactor 1.025
$enforce average_trialperiod 120
```

8.5 Enhanced SGI “cpusets” Support

PBS Pro version 5.3 introduced enhancements for running PBS in conjunction with SGI “cpusets”. (An IRIX “cpuset” is a named region of an SGI Origin system composed of one or more nodeboards, and associated memory and CPUs.) This section discusses these new features in conjunction with previous versions’ support for “cpusets”.

8.5.1 Irix OS-Leveling Checkpoint With cpusets

MOM now makes use of new IRIX checkpointing features to allow the checkpointing and restart of jobs running within cpusets. This capability requires SGI IRIX version 6.5.16 or later.

8.5.2 IRIX Shared cpusets for Small Jobs

MOM supports two classes of jobs running within cpusets: a *small* job and a *multi-cpu* job. The small job, which is usually a single CPU job with memory requirements such that it will fit on a single nodeboard, will allow itself to be run within a cpuset where other similar jobs run. Multi-cpu jobs are those that require more resources than are available on a single nodeboard or which need to run exclusively within a cpuset for repeatability of performance. The small job runs on a shared cpuset whereas the multi-cpu job runs on an exclusive cpuset. A special job resource, `ssinodes`, if set, will force any job to be allocated exclusive cpusets. Also, up to `max_shared_nodes` (set in MOM’s `config` file) will be allowed to be assigned to shared cpusets. To find out which cpuset is assigned to a running job, the `alt_id` job attribute has a field called `cpuset` that will show this information.

To set a threshold as to the number of nodeboards that can be assigned to shared cpusets, set the following in MOM’s `config` file:

```
max_shared_nodes  <# of nodeboards>
```

If you don't want MOM to run small jobs within a shared cpuset, set the following in MOM’s config file:

```
cpuset_small_ncpus 0
cpuset_small_mem 0
```

8.5.3 MOM configuration options

In order to make MOM `config` file options consistent, the format of `cpuset_create_flags` for specifying the flags to pass to `cpusetCreate()` is now changed from

```
    $cpuset_create_flags <flags>
```

to

```
    cpuset_create_flags <flags>
```

Similarly, the format for `cpuset_destroy_delay` for setting the number of seconds to sleep before tearing down (deleting) a cpuset is changed from

```
    $cpuset_destroy_delay <secs>
```

to

```
    cpuset_destroy_delay <secs>
```

In addition, a new option is available:

```
    alloc_nodes_greedy <0 or 1>
```

If set to 0, for job requests of 64 or fewer `ssinodes`, MOM will only allocate eligible nodes that are within one router hop in the hypercube architecture, that is, those nodes that are physically sitting close to each other. Otherwise if set to 1, MOM will allocate nodes from all those available regardless of the distance (number of router hops) between them. This latter case is the default behavior.

Important: Upon daemon startup, MOM will not remove any cpusets that it did not create. The nodes in these cpusets will be removed from MOM's available nodepool.

8.5.4 Enhanced cpuset Resource Reporting

MOM will now report to the Server the actual number of CPUs and memory that are under the control of PBS. This allows the node's `resources_available.{ncpus,mem}` to reflect the amount of resources that come from nodeboards that are not part of the reserved, system cpusets (e.g. `boot`) and stuck cpusets. Be sure to `unset` any manual settings of `resources_available.{ncpus,mem}` in both the node and the Server to get this count automatically updated by MOM. Manual settings (i.e. those either put in the server's `nodes` file or via the `qmgr set node` construct) take precedence.

If manually setting the server's `resources_available.ncpus` attribute, be sure to use a value that is a multiple of the nodeboard size. This value should not be less than one nodeboard size, otherwise no jobs (including shareable jobs) will run. For example, if there are four cpus per nodeboard, don't set `resources_available.ncpus=3`, instead set `resources_available.ncpus=4` (8, 12, 16, and so on).

8.5.5 Node Allocation Rules within MOM

Some special node and CPU allocation rules are now enforced by MOM:

If `cpuset_create_flags` set during `cpusetCreate()` contains a flag for `CPUSET_CPU_EXCLUSIVE` or `CPUSET_MEMORY_EXCLUSIVE`, then CPU 0 will not be allowed to be part of a cpuset.

During allocation of nodes, nodeboard 0 will only be allocated if no other nodes are available to satisfy the request. Use of node 0 for jobs can be a source of performance degradation as the kernel heavily uses this node for system daemons. Usually, PBS with cpusets is used in conjunction with a boot cpuset which the system administrator creates which includes nodeboard 0.

8.6 Static Resources

To identify static resource names and values, the MOM configuration file can contain a list of resource name/value pairs, one pair per line, separated by white space. These are most often used by the alternate schedulers, but are listed here for completeness. The names can be anything and are not restricted to actual hardware. For example the entry "`pongsoft 1`" could be used to indicate to the Scheduler that a certain piece of software ("pong") is available on this system. Another example could be the number of tape drives of different types.

```
pongsoft 1
tapedat  1
tape8mm  1
```

8.7 Dynamic Resources

PBS provides the ability to extend the resource query capabilities of MOM by adding shell escapes to the MOM configuration file. While this feature is most often used by the alternate and site-customized schedulers, the functionality is described in full here. Another use is to add site-specific information to the PBS monitoring tool, `xpbsmon`.

If the first character of the value portion of a name/value pair is the exclamation mark ("!"), the entire rest of the line is saved to be executed through the services of the `system`(3) standard library routine. The first line of output from the shell command is returned as the response to the resource query.

The shell escape provides a means for the resource monitor to yield arbitrary information to the Scheduler and other client commands. Parameter substitution is done such that the value of any qualifier sent with the resource query, as explained below, replaces a token with a percent sign (%) followed by the name of the qualifier. For example, here is a configuration file line which gives a resource name of "echotest":

```
echotest !echo %xxx %yyy
```

If a query for "`echotest`" is sent with no qualifiers, the command executed would be "`echo %xxx %yyy`". If one qualifier is sent, "`echotest[xxx=hi]`", the command executed would be "`echo  hi  %yyy`". If two qualifiers are sent, "`echotest [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, "`echotest [zzz=snafu]`", an error is reported.

Another example would allow the Scheduler to have MOM query the existence of a file. The following entry would be placed in MOM's config file:

```
file_test !if test -f %file; then echo yes; else echo no; fi
```

The query string

```
file_test [file=/tmp/lockout]
```

would return “`yes`” if the file exists and “`no`” if it did not.

Another possible use of the shell command configuration entry is to provide a means by which the use of floating software licenses may be tracked. If a program can be written to query the license server, the number of available licenses could be returned to tell the Scheduler if it is possible to run a job that needs a certain licensed package.

8.8 Idle Workstation Cycle Harvesting

“Harvesting” of idle workstations is a method of expanding the available computing resources of your site by automatically including in your cluster unused workstations that otherwise would have sat idle. This is particularly useful for sites that have a significant number of workstation that sit on researchers desks and are unused during the nights and weekends. With this feature, when the “owner” of the workstation isn’t using it, the machine can be configured to be used to run PBS jobs.

If a system is configured for cycle harvesting, it becomes available for batch usage by PBS if its keyboard and mouse remain unused or idle for a certain period of time. The workstation will be shown in state "free" when the status of the node is queried. If the keyboard or mouse is used, the workstation becomes unavailable for batch work and PBS will suspend any running jobs on that workstation and not attempt to schedule any additional work on that workstation. The workstation will be shown in state "busy", and any suspended jobs will be shown in state "U", a new state.

Important: Jobs on workstations that become *busy* will not be migrated; they remain on the workstation until they complete execution, are rerun, or are deleted.

The cycle harvesting feature is enabled via a single entry in `pbs_mom`'s `config` file, `$kbd_idle`, and takes up to three parameters, as shown below.

```
$kbd_idle idle_available [ idle_busy [ idle_poll ] ]
```

These three parameters, representing time specified in seconds, control the transitions between *free* and *busy* states. Definitions follow.

`idle_available`	is the time that the workstation keyboard and mouse must be idle before the workstation is available to PBS.
`idle_busy`	is a time period which the keyboard or mouse must remain busy before the workstation "stays" unavailable. This is used to keep a single key stroke or mouse movement from keeping the work-station busy.
`idle_poll`	is how often the state of the keyboard and mouse are checked.

Let us consider the following example.

```
$kbd_idle 1800 10 5
```

Adding the above line to MOM's config file directs PBS to mark the workstation as *free* if the keyboard and mouse are idle for 30 minutes (1800 seconds), to mark the workstation as *busy* if the keyboard or mouse are used for 10 consecutive seconds, and the state of the keyboard/mouse is to be checked every 5 seconds.

The default value of *idle_busy* is 10 seconds, the default for *idle_poll* is 1 second. There is no default for *idle_available*; setting it to non-zero is required to activate the cycle harvesting feature.

Elaborating on the above example will help clarify the roll of the various times. Lets start with a workstation that has been in use for some time by its owner. The workstation is shown in state *busy*. Now the owner goes to lunch. After 1800 seconds (30 minutes), the system will change state to *free* and PBS may start assigning jobs to run on the system. At some point after the workstation has become *free* and a job is started on it, someone walks by and moves the mouse or enters a command. Within the next 5 seconds, `pbs_mom` notes the activity. The job is suspended and shown being in state "U" and the workstation is marked *busy*. If, after 10 seconds have passed and there is no additional keyboard/mouse activity, the job is resumed and the workstation again is shown as *free*. However, if keyboard/mouse activity continued during that 10 seconds, then the workstation would remain *busy* and the job would remain suspended for at least the next 1800 seconds.

Important: Some computer operating systems do not update the keyboard and mouse access times, causing PBS not to be able to query keyboard/mouse activity. Specific workstation OS's known not to work in this regard are AIX, Irix, and Tru64 Unix.

8.9 MOM Globus Configuration

For the optional Globus MOM, the same configuration mechanism applies as with the regular MOM except only three initiation value directives are applicable: `$clienthost`, `$restricted`, `$logevent`.

8.10 Examples

The following examples are for a site called "The WyeWidget Company" whose domain name is "wyewidget.com".

8.10.1 Single Server Example

The following is an example of a config file for pbs_mom where the batch system is a single large multi-processor server. We want to log most records and specify that the system has one 8mm tape drive. In addition, the Scheduler runs on a front end machine named front.widget.com.

```
$logevent 0x0ff
$clienthost front.wyewidget.com
```

8.10.2 Cluster Example

Now the WyeWidget Computer Center has expanded to two large systems. The new system has two tape drives and is 30% faster than the old system. The PBS Manager wishes to charge the users the same regardless of where their job runs. Basing the charges on the old system, she will need to multiple the time used on the new system by 1.3 to charge the same as on the old system. The config file for the "old" system stays the same. The config file for the "new" system is:

```
$logevent 0x0ff
$clienthost front.wyewidget.com
$cputmult 1.3
$wallmult 1.3
```

Now the WyeWidget Company has decided to assemble a cluster of PCs running Linux named “bevy”, as in a bevy of PCs. The Scheduler and Server are running on bevy-boss.wyewidget.com which also has the user’s home file systems mounted under /`u/home/`.

The nodes in the cluster are named bevy1.wyewidget.com, bevy2.wyewidget.com, etc. The user’s home file systems are NFS mounted as `/r/home/...` The administrator’s personal workstation, adm.wyewidget.com, is where she plans to run `xpbsmon` to do cluster monitoring. The config file for each MOM would look like:

```
$logevent 0x0ff
$clienthost bevyboss.wyewidget.com
$restricted adm.wyewidget.com
$usecp bevyboss.wyewidget.com:/u/home /r/home
```

Chapter 9
Configuring the Scheduler

Now that the Server and MOMs have been configured, we turn our attention to the Scheduler. As mentioned previously, the Scheduler is responsible for implementing the local site policy by which jobs are run, and on what resources. This chapter discusses the default configuration created in the installation process, and describes the full list of tunable parameters available for the PBS Standard Scheduler.

9.1 Default Configuration

This Standard Scheduler provides a wide range of scheduling policies. It provides the ability to sort the jobs in several different ways, in addition to FIFO order. It also has the ability to sort on user and group priority, as well as many other features. As distributed, it is configured with the following options (which are described in detail below).

1. Three specific system resources are checked to make sure they are not exceeded: `mem` (memory requested), `ncpus` (number of CPUs requested) and `arch` (architecture requested).

2. Queues are sorted into descending order by queue priority attribute to determine the order in which they are to be considered.

3. All jobs in the highest priority queue will be considered for execu-

tion before considering any jobs from the next highest priority queue.

4. The jobs within each queue are sorted into ascending order by requested cpu time (cput). The shortest job is placed first.

5. Jobs which have been queued for more than one day will be considered *starving* and extra measures will be taken to attempt to run them.

6. Any queue whose name starts with "ded" is treated as a dedicated time queue (see discussion below). Sample dedicated time file (`/var/spool/sched_priv/dedicated_time`) is included in the installation.

7. Prime time is set to 6:00 AM - 5:30 PM. Any holiday is considered non-prime. Standard U.S. federal holidays for the year 2002 are provided in the file `/var/spool/PBS/sched_priv/holidays`. These dates should be adjusted yearly to reflect your local holidays.

8. In addition, the scheduler utilizes the following attributes/ resources in making scheduling decisions:

Object	**Attribute/Resource**	**Comparison**
server, queue & node	`resources_available`	>= resources requested by job
server, queue & node	`max_running`	>= number of jobs running
server, queue & node	`max_user_run`	>= number of jobs running for a user
server, queue & node	`max_group_run`	>= number of jobs running for a group

Object	Attribute/Resource	Comparison
server & queue	`max_user_res`	>= usage of specified resource by user
server & queue	`max_user_res_soft`	>= usage of specified resource by user (see discussion of soft limits)
server & queue	`max_user_run_soft`	>= maximum running jobs for a user (see discussion of soft limits)
server & queue	`max_group_run_soft`	>= maximum running jobs for a group (see discussion of soft limits)
queue	`started`	= true
queue	`queue_type`	= execution
job	`job_state`	= queued / suspended
node	`loadave`	< configured limit (not on by default)
node	`arch`	= type requested by job
node	`host`	= name or property requested by job

9.2 New Scheduler Features

With each PBS Pro release, new features are added to the Standard Scheduler. This section discusses major new scheduler features available in PBS Pro 5.3, and how you may best take advantage of these changes.

9.2.1 Job Sorting by Resource

It is now possible to sort jobs by specific PBS resources, in addition to sorting by specific sorting algorithms. Any job resource (such as memory or walltime) can be specified as a sort key. For explanation of use, see "job_sort_key" in section "Tunable Parameters" on page 117.

9.2.2 Peer Scheduling:

PBS Pro 5.3 introduces the ability to have different PBS installations automatically run jobs from each other’s queues. This provides the ability to dynamically load-balance across multiple, separate PBS installations. (These cooperating PBS installations are referred to as “peers”.) When this feature is enabled and resources are available, PBS can “pull” jobs from one or more (remote) Peer Servers and run them locally. No job will be moved if it cannot run immediately.

This functionality is implemented by mapping a remote Peer’s queue to a local queue. When the scheduler determines that a remote job can run locally, it will move the job to the specified queue on the local Server and then run the job.

This release of the Peer Scheduling features requires a flat user namespace, meaning that there is an assumption that user “joe” on the remote Peer system(s) is the same as user “joe” on the local system. This restriction will be removed in a future release.

To configure Peer Scheduling, the following setting needs to be made in the local Server and all Peer Servers, via `qmgr`:

```
qmgr>  set server flatuid = true
```

Furthermore, in order to pull jobs from a remote Peer to the local Server, the remote Peer Server needs to grant manager access to the local Peer Server (and *vis á vis* if you wish to permit jobs to move in the opposite direction).

```
qmgr> set server managers += root@localServer.domain.com
```

Lastly, you need to configure the local Scheduler. Add one or more `peer_queue` entries to the local Scheduler’s configuration file, mapping a remote queue to a local queue. Format is:

`peer_queue: "`*`local_queue remote_queue@remote_server.domain`*`"`

```
peer_queue: “workq workq@remote_server.domain.com
peer_queue: “peerq workq@other_server.different.com
```

Since the scheduler maps the remote jobs to a local queue, any moved jobs are subject to the policies of the queue they are moved into. If remote jobs are to be treated differently then local jobs, this can be done on the queue level. A queue can be created exclusively for remote jobs and this will allow queue level policy to be set for remote jobs.For example, you can set a priority value on your queues, and enable sorting by priority to ensure that remotely queued jobs are always lower (or higher!) priority than locally queued jobs.

9.2.3 Node Sorting

New in PBS Pro 5.3 is the ability to sort via node resources, in addition to job sorting. A new Scheduler parameters, node_sort_key is available to specify which key you want to use for sorting. See “Tunable Parameters” on page 117 for details.

9.2.4 System-wide Dynamic Resources

Extending the dynamic consumable resource feature of PBS Pro, is the new system-wide version. Whereas the existing feature is per-node, the new features allows you to specify external resources that should be managed system-wide, such as an external license manager. See “Dynamic Consumable Resources” on page 133 for details.

9.3 Tunable Parameters

To tune the behavior of the Standard Scheduler, change directory to `/var/spool/PBS/sched_priv` and edit the scheduling policy configuration file `sched_config` or use the default values. This file controls the scheduling policy (the order in which jobs run). The format of the `sched_config` file is:

name: *value* [prime | non_prime | all | none]

`name` can not contain any whitespace, but *`value`* may if the string is double-quoted. *`value`* can be: true | false | number | string. Any line starting with a “#” is a comment, and is ignored. The third field allows you to specify that the setting is to apply during prime-time, non-prime-time, or all the time. A blank third field is equivalent to “`all`” which is both prime- and non-primetime. Note that the *value* and *all* are case-insensitive; thus TRUE = True = true.

The available options for the Standard Scheduler, and the default values, are as follows.

`assign_ssinodes`

Deprecated. See “Scheduler Support for SGI IRIX cpusets” on page 124.

`backfill`
boolean: Instead of draining the system until the starving job runs, the Scheduler will attempt to backfill smaller jobs around the starving jobs. It will first attempt to schedule other starving jobs around it, before moving onto normal jobs. The `help_starving_jobs` attribute needs to be on in conjunction with this attribute. Note: this feature only applies to starving jobs.
Default: `true all`

`backfill_prime`
boolean: Directs the scheduler not to run jobs which will overlap into primetime or non-primetime. This will drain the system before primetime or non-primetime starts, assisting with the problem of jobs starting before prime/non-prime and impacting the job start time. See also `prime_spill`.
Default: `false all`

`by_queue`
boolean: If `true`, the jobs will be run queue by queue; if `false`, the entire job pool in the Server is looked at as one large queue.
Default: `true all`

`cpus_per_ssinode`
Deprecated. See “Scheduler Support for SGI IRIX cpusets” on page 124 below.

`dedicated_prefix`
string: Queue names with this prefix will be treated as dedicated queues, meaning jobs in that queue will only be considered for execution if the system is in dedicated time as specified in the configuration file `sched_priv/dedicated_time`.
Default: `ded`

`fair_share`
boolean: This will enable the fair share algorithm. It will also turn on usage collecting and jobs will be selected based on a function of their recent usage and priority(shares). See also section 9.3.5 “Defining Fair Share” on page 128.
Default: `false all`

`fairshare_usage_res`
string: Specifies the resource to collect and use in fairshare calcuations. (Can be any valid PBS resource.)
Default: cput

`fairshare_entity`
string: Specifies the job attribute to use as the fairshare “entity” for which fairshare usage data will be collected. (Can be any

valid PBS job attribute, such as “euser”, “egroup”, “Account_Name”, or “queue”).
Default: euser

`half_life`
time: The half life for fair share usage; after the amount of time specified, the fairshare usage will be halved. Requires that `fair_share` be enabled. See also section 9.3.5 “Defining Fair Share” on page 128.
Default: `24:00:00`

`help_starving_jobs`
boolean: Setting this option will enable starving jobs support. Once jobs have waited for the amount of time given by `max_starve` they are considered starving. If a job is considered starving, then no jobs will run until the starving job can be run, unless backfilling is also specified. To use this option, the `max_starve` configuration attribute needs to be set as well. See also `backfill, max_starve`.
Default: `true all`

`job_sort_key`
string: Selects how the jobs should be sorted. (Replaced `sort_by` from previous versions of PBS Pro.) `job_sort_key` can be used to sort by either resources or by specific sorting routines. Multiple `job_sort_key` entries will be applied sequentially. The `HIGH` option implies descending sorting, `LOW` implies ascending. See example for details.
Syntax: `job_sort_key: “PBS_resource HIGH|LOW”`
Default: “cput low”

There are also three special case sorting routines, that can be used instead of a specific PBS resource:

Special Sort	Description
`fair_share_perc HIGH`	Sort based on the values in the resource group file. This should only used if strict priority sorting is needed. **Do not enable `fair_share_perc` sorting if using the `fair_share` scheduling option.** (This option was previously named “`fair_share`”).

Special Sort	Description
`preempt_priorty HIGH`	Sort jobs by preemption priority
`sort_priority HIGH|LOW`	Sort by the job priority attribute

The following example illustrate using resources as sorting parameter. Note that for each, you need to specify HIGH (descending) or LOW (ascending). Also, note that *resources* must be a quoted string.

```
job_sort_key: “ncpus HIGH” all
job_sort_key: “mem LOW” prime
```

`key` Deprecated. Use `job_sort_key`.

`load_balancing`

boolean: If set, the Scheduler will load balance the jobs across the nodes. The load balancing takes into consideration the load on each node as well as all resources specified in the “resource” list. See `smp_cluster_dist`.
Default: `false all`

`load_balancing_rr`

Obsolete. To duplicate this setting, enable `load_balancing` and set `smp_cluster_dist` to `round_robin`.

`log_filter` integer: Defines which event types to filter from the daemon logfile. The value should be set to the bitwise OR of the event classes which should be filtered. See also section 11.14 “Use and Maintenance of Logfiles” on page 163.
Default: `256` (DEBUG2)

`max_starve` time: The amount of time before a job is considered starving. This variable is used only if `help_starving_jobs` is set.
Default: `24:00:00`

`mem_per_ssinode`

Obsolete. See “Scheduler Support for SGI IRIX cpusets” on page 124 below.

`node_sort_key`

string: Defines sorting on resource_available values on nodes (i.e. total amounts, not free amounts). Options and usage are same as for `job_sort_key`, except only one special case sorting algorithm is applicable here: `sort_priority`.

`nonprimetime_prefix`

string: Queue names which start with this prefix will be treated as non-primetime queues. Jobs within these queues will only run during non-primetime. Primetime and nonprimetime are defined in the `holidays` file.
Default: `np_`

`primetime_prefix`

string: Queue names which start with this prefix will be treated as primetime queues. Jobs will only run in these queues during primetime. Primetime and non-primetime are defined in the `holidays` file.
Default: `p_`

`preemptive_sched`

string: Enable job preemption. See “Preemptive Scheduling” on page 126 for details.
Default: `true all`

`preempt_checkpoint`

Deprecated; use `preempt_order`.

`preempt_fairshare`

Deprecated; use `preempt_prio`.

`preempt_order`

quoted list: Defines the order of preemption methods which the scheduler will use on jobs. This order can change depending on the percentage of time remaining on the job. The ordering can be any combination of S C and R (for suspend, checkpoint, and requeue). The usage is an ordering (SCR) optionally followed by a percentage of time remaining and another ordering. Note, this has to be a quoted list("").
Default: `SCR`

```
preempt_order: "SR"
# or
preempt_order: "SCR 80 SC 50 S"
```

The first example above specifies that PBS should first attempt to use suspension to preempt a job, and if that is unsuccessful, then requeue the job. The second example says if the job has between 100-81% of requested time remaining, first try to suspend the job, then try checkpoint then requeue. If the job has between 80-51% of requested time remaining, then attempt suspend then checkpoint; and between 50% and 0% time remaining just attempt to suspend the job.

`preempt_prio` quoted list: Specifies the ordering of priority of different preemption levels. Two or more job types may be combined at the *same* priority level with a "+" between them (no whitespace). Comma-separated preemption levels are evaluation left to right, with each having lower priority than the preemption level preceding it. The table below lists the six preemption levels.
Default: `"express_queue, normal_jobs"`

express_queue	jobs in the preemption queue(s) preempt (see also `preempt_queue_prio`)
starving_jobs	when a job becomes starving it can preempt (requires `preempt_starving` be set to `true`)
fairshare	when a job uses too much of it's fairshare (requires `preempt_fairshare` be set to `true`)
queue_softlimits	jobs which are over their queue soft limits
server_softlimits	jobs which are over their server soft limits
normal_jobs	default for normal jobs

```
preempt_prio "starving_jobs, normal_jobs, fairshare"
# or
"starving_jobs, normal_jobs, starving_jobs+fairshare"
```

`preempt_queue_prio`

integer: Specifies the minimum queue priority required for a queue to be classified as an express queue.
Default: `150`

`preempt_requeue`

Deprecated; use `preempt_order`.

`preempt_starving`
Deprecated; use `preemp_prio`.

`preempt_suspend`
Deprecated; use `preempt_order`.

`prime_spill` time: Specifies the amount of time a job can "spill" over from non-primetime into primetime or from non-primetime into primetime. For example, if a job is in a primetime queue and it is currently primetime, but the job would cross over into non-primetime by one hour, the job would not run if `prime_spill` were set to less than one hour. Note, this option is only meaningful if `backfill_prime` is true. Also note that this option can be separately specified for prime- and non-primetime.
Default: `00:00:00`

`resources` string: Specifies the resources by which to schedule the system. Limits are set by setting resources_available.resourceName on the Server objects (nodes, queues, and servers). The scheduler will consider numeric (int or float) items as consumable resources and ensure that no more are assigned than are available (e.g. `ncpus` or `mem`). Any string resources will be compared using string comparisons (e.g. `arch`). See the description of the Server attribute `resources`; see also section “Dynamic Consumable Resources” on page 133.
Default: “`ncpus, mem, arch`” (number CPUs, memory, architecture)

`round_robin` boolean: If `true`, the queues will be cycled through in a circular fashion, attempting to run one job from each queue. If `false`, attempts to run all jobs from the current queue before processing the next queue. See `by_queue`.
Default: `false all`

`server_dyn_res` The Scheduler can be configured to call a shell program and replace the server’s `resource_available` values with new values returned by the shell program. See section “Dynamic Consumable Resources” on page 133 for details of usage.

`smp_cluster_dist`
string: Specifies how jobs should be distributed to all nodes of

the cluster. Options are: `pack`, `round_robin`, and `lowest_load`. `pack` means keep putting jobs onto one node until it is "full" and then move onto the next. `round_robin` is to put one job on each node in turn before cycling back to the first one. `lowest_load` means to put the job on the lowest loaded node.
Default: `pack all`

`sort_by`	Deprecated. Use `job_sort_key`.
`strict_fifo`	boolean: If `true`, jobs will be run in a strict FIFO order. This means if a job fails to run for any reason, no more jobs will run from that queue/Server during that scheduling cycle. If `strict_fifo` is not set, large jobs can be starved, i.e., not allowed to run because a never ending series of small jobs use the available resources. Also see the Server attribute `resources_available`, and the parameters `help_starving_jobs` and `backfill` above. Default: `false all`
`sync_time`	time: The amount of time between writing the fair share usage data to disk. Requires `fair_share` to be enabled. Default: `1:00:00`
`unknown_shares`	integer: The amount of shares for the "unknown" group. Requires `fair_share` to be enabled. See also section 9.3.5 "Defining Fair Share" on page 128. Default: `10`

9.3.1 Scheduler Support for SGI IRIX cpusets

As discussed earlier in this manual, PBS Pro supports the use of SGI IRIX "cpusets" (or named regions of an SGI IRIX system containing specific CPUs and associated memory). If support for SGI cpusets is desired, it needs to be enabled in the Scheduler. (See also section 4.4.2 "Installing MOM with SGI "cpuset" Support" on page 24).

The number of CPUs and amount of memory per node board within the system are queried directly from the MOM local to that system. This permits running clusters of SGI Origin systems with different hardware configurations managed by a single PBS Scheduler. The Scheduler will modify the `mem` and/or `ncpus` request of jobs into values that corresponds to nodeboard multiples (i.e. adjusting upward to the fewest number of entire nodeboards necessary to fill both the memory and CPU request of the job). The Scheduler also sets the `ssinodes` resource used by MOM.

Note: for support of "shared cpusets" (i.e. for small jobs that cannot use an entire nodeboard, and the site wishes to schedule multiple small jobs within a single cpuset), the Scheduler does not set the `ssinodes` resource. In this case, MOM sets the resource herself.

9.3.2 SMP Cluster Support

The Standard Scheduler schedules SMP clusters in an efficient manner. Instead of scheduling only via load average of nodes, it takes into consideration the resources specified at the server, queue, and node level. Furthermore, the administrator can explicitly select the resources to be considered in scheduling via an option in the Scheduler's configuration file (`resources`). The configuration parameter `smp_cluster_dist` allows you to specify how nodes are selected. The available choices are `pack` (pack one node until full), `round_robin` (put one job on each node in turn), or `least_loaded` (put one job on the least loaded node). The `smp_cluster_dist` parameter should be used in conjunction with `node_sort_key` to ensure efficient scheduling.

To use these features requires two steps: setting resource limits via the Server, and specifying the scheduling options. Resource limits are set using the `resources_available` attribute of nodes via `qmgr` just like on the server or queues. For example, to set maximum limits on a node called "node1" to 10 CPUs and 2 GB of memory:

```
Qmgr>set node node1 resources_available.ncpus = 10
Qmgr>set node node1 resources_available.mem=2GB
```

Important: Note that by default both `resources_available.ncpus` and `resources_available.mem` are set to the physical number reported by MOM on the node. Typically, you do not need to set these values, unless you do not want to use the actual values reported by MOM.

Next, the Scheduler options need to be set. For example, to enable SMP cluster scheduler to use the "round robin" algorithm during primetime, and the "pack" algorithm during non-primetime, set the following in the Scheduler's configuration file:

```
smp_cluster_dist: round_robin prime
smp_cluster_dist: pack        non_prime
```

Finally, specify the resources to use during scheduling (also in the Scheduler's configuration file):

```
resources: "ncpus, mem, arch"
```

9.3.3 Preemptive Scheduling

PBS provides the ability to preempt currently running jobs in order to run higher priority work. Preemptive scheduling is enabled by setting several parameters in the Scheduler's configuration file (discussed below, and in "Tunable Parameters" on page 117). Jobs utilizing advance reservations are not preemptable. If priority jobs (as defined by your settings on the preemption parameters) can not run immediately, the Scheduler looks for jobs to preempt, in order to run the higher priority job. A job can be preempted in several ways. The Scheduler can suspend the job (i.e. sending a SIGSTOP signal), checkpoint the job (if supported by the underlying operating system), or requeue the job. (The administrator can choose the order of these attempts via the `preempt_order` parameter.)

There are several Scheduler parameters to control preemption. The `preemptive_sched` parameter turns preemptive scheduling on and off. You can specify the relative priority between different types of jobs using `preempt_prio`. You set what queue priority is preemptive with the `preempt_queue_prio` parameter. The parameter `preempt_order` allows you to specify the order in which these methods should be applied. If one preemption method fails, the Scheduler tries the next. If the scheduler cannot find enough work to preempt in order to run a given job, it will not preempt any work.

The `preempt_prio` parameter provides a means of specifying the order of precedence that preemption should take. The ordering is evaluated from left to right. One special name (`normal_jobs`) is the default. If you want normal jobs to preempt other lower priority jobs, put `normal_job` in the middle of the `preempt_prio` list. If two or more levels are desired for one priority setting, the multiple levels may be indicated by putting a '+' between them. A complete listing of the preemption levels is provided in the scheduler tunable parameters section above.

Two preemption levels can be turned on and off by specific `sched_config` parameters: `preempt_fairshare` to enable preemption by fairshare; and `preempt_starving` to have starving jobs preempt other jobs. Both should be left on, and controlled via `preempt_prio`.

Soft run limits can be set or unset via `qmgr.` If unset, any job that would have been subject to this limits will instead be considered a "normal job". (Likewise, if `preempt_fairshare` or `preempt_starving` are set to `false`, applicable jobs will instead be treated as “normal jobs” too.) A new job sort has been added (`preempt_priorty`) which will sort jobs by their preemption priority. Note: it is a good idea to set this as the primary sort if preempt_prioorty is set to more than the default.

Important: Note that any queue with a priority 150 (default value) is treated as an express (i.e. high priority) queue.

Below is an example of (part of) the Scheduler’s configuration file showing how to enable preemptive scheduling and related parameters. Explanatory comments precede each configuration parameter.

```
# turn on preemptive scheduling
preemptive_sched:       TRUE  ALL

# set the queue priority level for express queues
preempt_queue_prio:     150

# specifiy the priority of jobs as: express queue (highest)
# then starving jobs, then normal jobs, followed by jobs
# who are staving but the user/group is over a soft limit,
# followed by users/groups over their soft limit but not
# starving
#
preempt_prio: "express_queue, starving_jobs, normal_jobs,
starving_jobs+over_server_limit, over_server_limit"

# specify when to each preemption method. If the first
# method fails, try the next method. If a job is has
# between 100-81% time remaining, try to suspend, then
# checkpoint then requeue. From 80-51% suspend and then
# checkpoint, but don't requeue. If between 50-0% time
# remaining, then just suspend it.
preempt_order: "SCR 80 SC 50 S"
```

9.3.4 Defining Dedicated Time

The file `/usr/spool/PBS/sched_priv/dedicated_time` defines the dedicated times for the scheduler. During dedicated time, only jobs in the dedicated time queues can be run (see `dedicated_prefix` above). The format of entries is:

```
# From Date-Time   To Date-Time
# MM/DD/YYYY HH:MM MM/DD/YYYY HH:MM
# For example
04/15/2001 12:00 04/15/2001 15:30
```

Important: To force the Scheduler to re-read the dedicated time file (which is needed after modifying the file), send the Scheduler daemon process a SIGHUP.

9.3.5 Defining Fair Share

PBS fairshare is similar to the UNICOS implementation of fairshare. Users are put in a fairshare group file. The file is read in and a tree is created. The tree consists of groups (nodes) and entities (leaves). Groups can contain groups. Every node and leaf has a number of shares associated with it. Priorities can be derived from these shares by taking a ratio of them to all the rest of the shares.

The fairshare capability allows an administrator to set which PBS resource is collected for fairshare usage. If unspecified, the default it is cput. The resource to use is specified in the `sched_config` file with the parameter: `fairshare_usage_res`. An Administrator can specify what the fairshare "entity" to use as well. It can be set to any PBS job attribute (i.e. `euser`, `egroup`, `Account_Name`, etc) via the `sched_config` file parameter: `fairshare_entity`. The administrator can also use the `pbsfs` command to display the fairshare usage tree. (See the `pbsfs(8B)` manual page, or "pbsfs (PBS Fairshare) command" on page 178 for usage details.)

Release 5.2 of PBS Pro introduced a change from a flat fairshare tree to a hierarchical fairshare tree. This permits enforcing fairshare across groups of people rather than just individuals. Previously, the fairshare tree was used to create an initial hierarchy of percentages for sorting and running jobs, but the enforcement was flat (i.e. per-user). Now the hierarchy is extended to include enforcement as well. For example, consider a situation where Sally and Bob are both in group A and Mark is in group B and Sally uses 100% of the machine for a month. In prior releases of PBS Pro, Bob and Mark would compete for cycles since neither of them had any usage (i.e. comparison of individuals within the fairshare tree). Now, with a hierarchal fairshare tree, since group A has all of Sally's usage, group B has much higher priority. Thus Mark's jobs would run before Bob's.

For example, say you have three nodes/leaves at one level with shares of 10, 20, and 10. The first entity/group has a priority of 25% or 10/40, the second has 50% or 20/40 and so on. A node with children can establish priorities among them via shares. So if in the example above the second group (50%) is actually a group with 4 users and all the users have equal shares, then each user has 1/4 of 50% or 12.5% of the machine.

If fair share or strict priority is going to be used, the resource group file `/var/spool/PBS/sched_priv/resource_group` may need to be edited. (If all users are considered equal, this file doesn't need to be edited.) Each line of the file should use the following format:

```
name unique_ID parent_group shares
```

`name`	The name of the specified entity or group
`unique_id`	A unique numeric identifier for the group or entity
`parent_group`	The name of the parent resource group to which this user/group belongs. The root of the share tree is called `root` and is added automatically to the tree by the Scheduler.
`shares`	The number shares (or priority) the user/group has in the specified resource group.

If you wish to specify how individual users should be ranked against each other, only user entries are required in the `resources_group` file, as shown the following example:

```
tom    60      root    5
bob    61      root    15
roy    62      root    15
jim    63      root    10
rob    64      root    25
ann    65      root    10
dan    66      root    10
```

Another option is to divide shares into "groups", and then name the users who are members of each group. The following example illustrates this configuration:

```
grp1    50      root    10
grp2    51      root    20
grp3    52      root    10
grp4    53      grp1    20
grp5    54      grp1    10
grp6    55      grp2    20
tom     60      root    5
bob     61      grp1    10
roy     62      grp2    10
jim     63      grp6    10
rob     64      grp6    10
ann     65      grp6    20
dan     66      grp3    10
sam     67      grp4    10
erin    68      grp4    10
will    69      grp5    10
```

The above example shows three top level groups (whose parent is the root group), three sub-groups (whose parent is a user-defined group), and 10 users belonging to different groups. The box below shows how the individual shares or percentages are calculated for groups and users, using the values from the above example. First we start with the root group as having 100% of the machine, but it has three member groups and one user, with share values of 10, 20, 10, and 5, respectively, for a total of 45 shares. The second line below shows calculating the actual percentage of the resources that group grp1 has. Specifically, the specified 10 shares are out of a total of 45 specified shares, or 22% of the total. The members of group grp1 (i.e. user bob and groups grp4 and grp5) each have specific shares of their group, all of which together total 40 shares specified for this group (i.e. 10+20+10). Thus their specified shares are of this total, resulting in these three entities having 25%, 50%, and 25%, respectively, of group grp1. We can further calculate how these share convert into shares of the total system by multiplying against the total percentage possessed by their group, grp1.

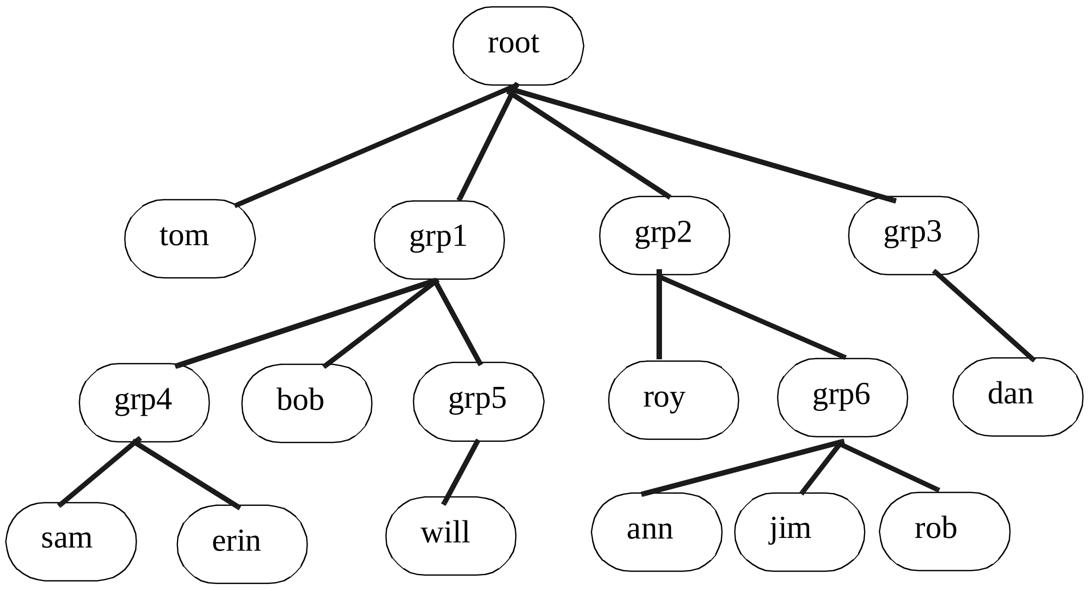
Two ways to visualize this share tree are given below, both a table of relative percentages, and a graphical representation of the relationship between users and groups within the fairshare tree.

```
root 100%
    grp1 10 10/45 = 100% * 22% = 22%
        bob 10 10/40 (25%) * 22% = 5.5%
        grp4 20 20/40 (50%) * 22% = 11%
            sam  10 10/20 (50%) * 11% = 5.5%
            erin 10 10/20 (50%) * 11% = 5.5%
        grp5 10 10/40 (25%) * 22% = 5.5%
            will 10 100% * 5.5% = 5.5%

    grp2 20 20/45 = 100% * 44% = 44%
        roy 10 10/30 (33%) * 44% = 14.5%
        grp6 20 20/30 (66%) * 44% = 29%
             jim 10 10/40 (25%) * 29% = 7.2%
             rob 10 10/40 (25%) * 29% = 7.2%
             ann 20 20/40 (50%) * .29% = 14.5%

    grp3 10 10/45 = 22%
        dan 10 100% * 22% = 22%

    tom 5 5/45 = 11%
```



9.3.6 Defining Strict Priority

Not to be confused with fair share (which considers past usage of each entity in the selection of jobs), the Standard Scheduler offers a sorting key called "`fair_share_perc`". Selecting this option enables the sorting of jobs based on the priorities specified in the fair share tree (as defined above in the `resource_group` file). A simple share tree will suffice. Every user's `parent_group` should be `root`. The amount of shares should be their desired priority. `unknown_shares` (in the scheduler's configuration file) should be set to one. Doing so will cause everyone who is not in the tree to share one share between them, making sure everyone else in the tree will have priority over them. Lastly, `job_sort_key` must be set to `"fair_share_perc HIGH"`. This will sort by the fair share tree which was just set up. For example:

```
usr1    60      root    5
usr2    61      root    15
usr3    62      root    15
usr4    63      root    10
usr5    64      root    25
usr6    65      root    30
```

9.3.7 Defining Primetime and Holidays

To have the scheduler utilize and enforce holidays, edit the `/usr/spool/PBS/sched_priv/holidays` file to handle prime time and holidays. The holidays file should use the UNICOS 8 holiday format. The ordering is important. Any line that begins with a "*" is considered a comment. The format of the holidays file is:

YEAR YYYY	This is the current year.
<day> <prime> <nonprime>	
<day> <prime> <nonprime>	

Day can be `weekday`, `saturday`, or `sunday`
Prime and *nonprime* are times when prime or non-prime time start. Times can either be HHMM with no colons(:) or the word "`all`" or "`none`"

<day> <date> <holiday>

Day is the day of the year between 1 and 365 (e.g. "1")
Date is the calendar date (e.g. "Jan 1")
Holiday is the name of the holiday (e.g. "New Year's Day")

```
HOLIDAYFILE_VERSION1
YEAR    2002
*            Prime   Non-Prime
* Day        Start   Start
*
  weekday    0600    1730
  saturday   none    all
  sunday     none    all
*
* Day of    Calendar      Company Holiday
* Year       Date          Holiday
    1        Jan 1         New Year's Day
   21        Jan 21        Martin Luther King Day
   49        Feb 18        President's Day
  147        May 27        Memorial Day
  185        Jul 4         Independence Day
  245        Sep 2         Labor Day
  280        Oct 7         Columbus Day
  315        Nov 11        Veteran's Day
  332        Nov 28        Thanksgiving
  359        Dec 25        Christmas Day
```

9.4 Dynamic Consumable Resources

It is possible to schedule resources where the number or amount available is outside of PBS's control. Such resources are called “dynamic consumable resources”, of which there are two types: *per-node* and *system-wide*.

Important: Because dynamic consumable resources are external to PBS, they are not completely under PBS’ control. Therefore it is possible for PBS to query and find a resource available, schedule a job to run to use that job, only to have an outside entity take that resource before the is able to use it.

9.4.1 Per-Node Dynamic Consumable Resources

For per-node dynamic consumable resources, the Scheduler will perform a resource query to each MOM to get the current availability for the resource and use that value for scheduling. These resources are specified in the Scheduler’s configuration file using the parameter, `mom_resources`, and are queried from every MOM in the cluster. If the MOM returns a value it will replace the `resources_available` value reported by the Server. If the MOM returns no value, the value from the Server is kept. If neither specify a value, the Scheduler sets the resource value to 0.

For example, say you have jobs that require a large amount of scratch disk space during their execution. To insure that sufficient space is available when starting the job, you first write a script that returns as a single line (with new-line) the amount of space available. This script is placed in `/usr/local/bin/scratchspace` on each node.

Next, edit the Server's resource definition file, (`.../server_priv/resourcedef`) adding a definition for the new resource. (See also “Defining New Resources” on page 87.) For this example, let's call our new resource "scratchspace":

```
scratchspace    type=size
```

Next, configure MOM to use the `scratchspace` script by entering one line into the `mom_priv/config` file:

```
scratchspace  !/usr/local/bin/scratchspace
```

Finally, edit the scheduler configuration file (`sched_priv/sched_config`), specifying this new resource that you want queried and used for scheduling:

```
mom_resources:  “scratchspace”
```

Jobs can now request this resource with

```
% qsub -l scratchspace=100mb ...
```

9.4.2 System-Wide Dynamic Consumable Resources

If you have a resource (such as a software license manager) that operates system-wide (rather than per-node), you should use the system-wide dynamic resource feature. Rather than querying each MOM in the cluster, in the system-wide configuration the Scheduler queries the Server for the requested resource.

As an illustrative example, suppose your company has ten licenses to an expensive Astrology program. Things are further complicated by the marketing department demanding you reserve two licenses for their business planning seances. You've been told to limit everyone else to at most eight licenses. You decide that PBS should do this, so you write a quick shell script (called `count_astro_licenses`) to query the license manager and report the total number of licenses available. (This could be useful, for example, if there are other programs outside of PBS that may use these licenses. Your script can detect this, and reduce the number that PBS should manage.). Now you are ready to configure PBS.

First, edit the Server's resource file (`.../server_priv/resourcedef`) adding a definition for your new resource. Let's call it "astrology":

```
astrology   type=long
```

Next, configure the Scheduler to use your little shell script to query the Astrology licenses, by entering one line into the `sched_priv/sched_config` file:

```
server_dyn_res: “astrology !/usr/bin/count_astro_licenses”
```

Jobs request the astrology license via:

```
% qsub -l astrology=1 ...
```

Chapter 10
Example Configurations

Up to this point in this manual, we have seen many examples of how to configure the individual PBS daemons, set limits, and otherwise tune a PBS installation. Those examples were used to illustrate specific points or configuration options. This chapter pulls these various examples together into configuration-specific scenarios which will hopefully clarify any remaining configuration questions. Four configuration models are discussed, each more complex than the one before it:

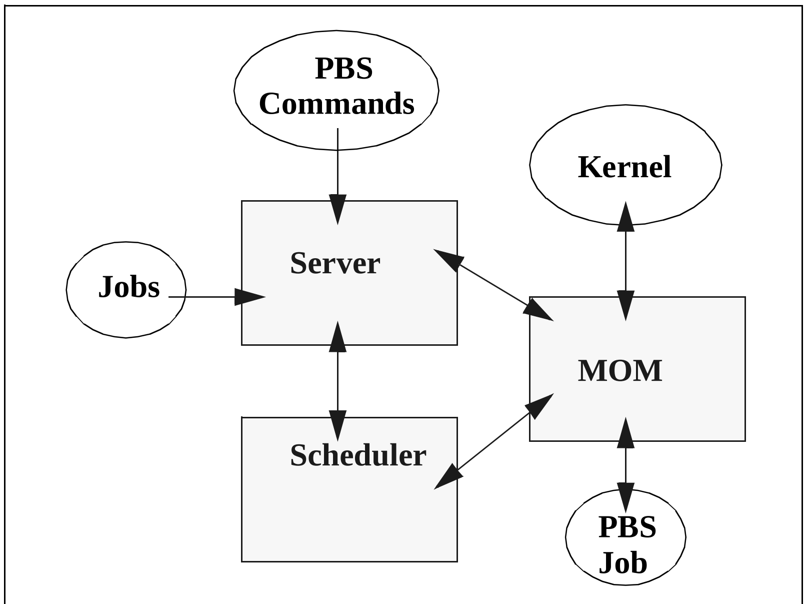
Single Node Time-sharing System
Single Node System with Separate PBS Server
Multi-node Time-sharing Cluster
Multi-node Space-sharing Cluster

For each of these possible configuration models, the following information is provided:

General description for the configuration model
Type of system the model is well suited for
Graphic illustration of the model
Contents of Server nodes file
Any required settings in Server
Contents of MOM configuration file
Required settings in Scheduler configuration file

10.1 Single Node Time-sharing System

Running PBS on a single node/host as a standalone time-sharing system is the least complex configuration. This model is most applicable to sites who have a single large Server system, or even a vector supercomputer. In this model, all three PBS daemons run on the same host, which is the same host on which jobs will be executed, as shown in the figure below.



All daemons on a single host.

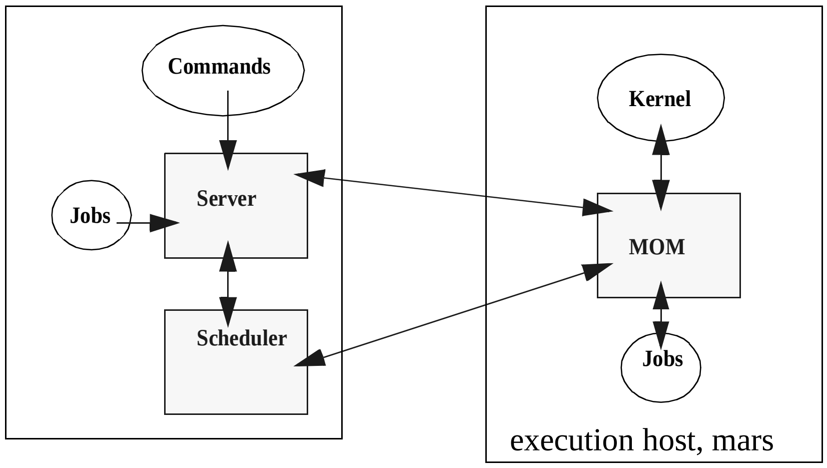
For this example, lets assume we have a 32-CPU server machine named “mars”. We want users to log into “mars” and jobs will be run via PBS on mars.

In this configuration, the default PBS `nodes` file (which should contain the name of the host on which the Server was installed) is sufficient. Our example `nodes` file would contain only one entry: `mars:ts`

The default MOM and Scheduler `config` files, as well as the default queue/Server limits are also sufficient. No changes are required from the default configuration.

10.2 Single Timesharing Node with Separate Server

A variation on this model would be to provide a “front-end” system that ran the PBS Server and Scheduler, and from which users submitted their jobs. Only the MOM daemon would run on our execution server, mars. This model is recommended when the user load would otherwise interfere with the computational load on the Server.



In this case, the PBS `server_priv/nodes` file would contain the name of our execution server mars, but this may not be what was written to the file during installation, depending on which options were selected. It is possible the hostname of the machine on which the Server was installed was added to the file, in which case you would need to either manually edit the `nodes` file, or use `qmgr(1B)` to manipulate the contents to contain one node: `mars:ts`. If the default scheduling policy, based on available CPUs and memory, meets your requirements, then no changes are required in either the MOM or Scheduler configuration files.

However, if you wish the execution node (mars) to be scheduled based on load average, the following changes are needed. Edit MOM’s `mom_priv/config` file so that it contains the target and maximum load averages, e.g.:

```
$ideal_load 30
$max_load 32
```

In the scheduler `sched_priv/config` file, the following options would need to be set:

```
load_balancing: true all
```

10.3 Multi-node Timesharing Cluster

The multi-node time-sharing cluster model is a very common configuration for PBS. In this model, there is typically a “front-end” system as we saw in the previous example, with a number of ‘back-end” compute nodes. The PBS Server and Scheduler are typically run on the front-end system, and a MOM daemon is run on each of the execution nodes, as shown in the diagram to the right.

In this model, the PBS `nodes` file will need to contain the list of all the nodes in the cluster, with the timesharing attribute `:ts` appended:

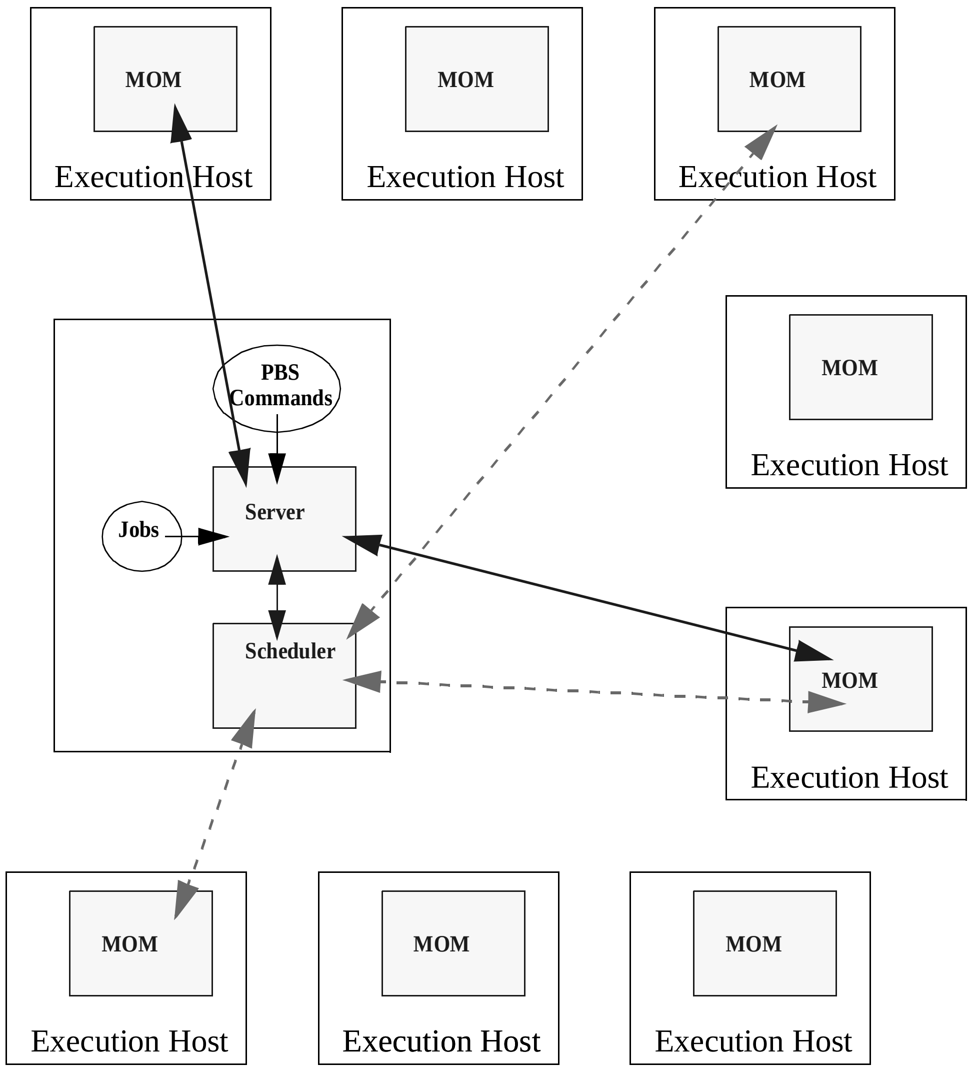
```
mercury:ts
venus:ts
earth:ts
mars:ts
jupiter:ts
saturn:ts
uranus:ts
neptune:ts
```

The MOM `config` file on each node will need two static resources added, to specify the target load for each node. If we assume each of the nodes in our planets cluster is a 32-processor system, then the following example shows what might be desirable values to add to the MOM `config` files:

```
$ideal_load 30
$max_load 32
```

Furthermore, suppose we want the Scheduler to load balance the workload across the available nodes, making sure not to run two job in a row on the same node (round robin node scheduling). We accomplish this by editing the Scheduler configuration file and enabling load_balancing:

```
load_balancing: true all
smp_cluster_dist: round_robin
```



This diagram illustrates multi-node cluster configurations wherein the Scheduler and Server communicate with the MOMs on the execution nodes. Jobs are submitted to the

Server, scheduled for execution by the Scheduler, and then transferred to a MOM when it's time to be run. MOM periodically sends status information back to the Server, and answers resource requests from the Scheduler.

10.4 Multi-node Space-sharing Cluster

A variation on the time-sharing cluster is the "space-shared" cluster. In this context, space-sharing refers to assigning an entire node (or set of nodes) to a single job at a time, for the duration of the job. This is usually done in order to achieve high-throughput and predictability of runtimes for a specific application (such as message passing parallel jobs).

In this model, the PBS `nodes` file would **not** have the `:ts` appended to the node names, e.g.:

```
mercury
venus
earth
mars
jupiter
saturn
uranus
neptune
```

There would be no edits required to either the Scheduler or the MOM `config` files.

Lastly, since in this model users specify their job resource requirements using the "`-l nodes=...`" syntax of `qsub`, we need to set node-specific limits in the Server:

```
# qmgr
Qmgr: set server resources_default.nodes = 1
Qmgr: set server resources_default.nodect = 1
Qmgr: set server resources_default.neednodes = 1
```

Chapter 11
Administration

This chapter covers information on the maintenance and administration of PBS, and as such is intended for the PBS Manager. Topics covered include: starting and stopping PBS, security within PBS, prologue/epilogue scripts, PBS accounting, configuration of the PBS GUIs, and using PBS with other products such as Globus.

11.1 /etc/pbs.conf

During the installation of PBS Pro, the following `/etc/pbs.conf` file was created:

```
PBS_CONF_START_SERVER=1
PBS_CONF_START_MOM=1
PBS_CONF_START_SCHED=1
PBS_HOME=/usr/spool/PBS
PBS_EXEC=/opt/pbs
PBS_SERVER=hostname.domain
```

This configuration file controls which daemons are to be running on the local system, directory tree location, and various runtime configuration options. Each node in a cluster should have its own `/etc/pbs.conf` file. The following table describes the available parameters as of the current version of PBS Pro:

Parameters	Meaning
PBS_BATCH_SERVICE_PORT	Port Server listens on
PBS_BATCH_SERVICE_PORT_DIS	DIS Port Server listens on
PBS_CONF_START_SERVER	set to 1 if Server is to run on this host
PBS_CONF_START_MOM	set to 1 if a MOM is to run on this host
PBS_CONF_START_SCHED	set to 1 if Scheduler is to run on this node
PBS_CONF_SYSLOG	controls use of syslog facility
PBS_CONF_SYSLOGSEVR	filters syslog messages by severity
PBS_ENVIRONMENT	location of `pbs_environment` file
PBS_EXEC	location of PBS bin and sbin directories
PBS_HOME	location of PBS working directories
PBS_MANAGER_GLOBUS_SERVICE_PORT	Port Globus Manager listens on
PBS_MANAGER_SERVICE_PORT	Port MOM Manager listens on
PBS_MOM_GLOBUS_SERVICE_PORT	Port Globus MOM listens on
PBS_MOM_SERVICE_PORT	Port MOM listens on
PBS_RCP	location of `rcp` command if `rcp` is used
PBS_SCHEDULER_SERVICE_PORT	Port Scheduler listens on
PBS_SCP	location of `scp` command if `scp` is used
PBS_SERVER	hostname of host running the Server

11.2 Starting PBS Daemons

The daemon processes, Server, Scheduler, MOM and the optional MOM Globus, must run with the real and effective uid of root. Typically the daemons are started automatically by the system upon reboot. The boot-time start/stop script for PBS is `/etc/init.d/pbs`. This script reads the `/etc/pbs.conf` file to determine which daemons should be started.

The startup script can also be run by hand to get status on the PBS daemons, and to start/stop all the PBS daemons on a given host. The command line syntax for the startup script is:

```
/etc/init.d/pbs [ status | stop | start ]
```

Alternately, you can start the individual PBS daemons manually, as discussed in the following sections. Furthermore, you may wish to change the options specified to various daemons, as discussed below.

11.2.1 Manually Starting MOM

MOM should be started at boot time. Typically there are no required options. It works best if MOM is started before the Server so she will be ready to respond to the Server's "are you there?" ping. Start MOM with the command line:

```
/usr/pbs/sbin/pbs_mom [options]
```

If MOM is taken down and the host system continues to run, MOM should be restarted with either of the following options:

`-p` This directs MOM to let running jobs continue to run. Because MOM is no longer the parent of the jobs, she will not be notified (**SIGCHLD**) when they die and so must poll to determine when the jobs complete. The resource usage information therefore may not be completely accurate.

`-r` This directs MOM to kill off any jobs which were left running.

Without either the `-p` or the `-r` option, MOM will assume the jobs' processes are non-existent due to a system restart, a cold start. She will not attempt to kill the processes and will request that any jobs which where running before the system restart be requeued.

Other command line options for MOM include:

`-a alarm` Used to specify the alarm timeout in seconds for computing a resource. Every time a resource request is processed, an alarm is set for the given amount of time. If the request has not completed before the given time, an alarm signal is generated. The default is 5 seconds.

`-C chkdir`	Specifies the path of the directory used to hold checkpoint files. [Currently this is only valid on Cray and SGI systems.] The default directory is `PBS_HOME/spool/checkpoint`, see the `-d` option. The directory specified with the `-C` option must be owned by root and accessible (rwx) only by root to protect the security of the checkpoint files.
`-c config`	Specifies an alternative configuration file, see description below. If this is a relative file name it will be relative to `PBS_HOME/mom_priv`, see the `-d` option. If the specified file cannot be opened, `pbs_mom` will abort. If the `-c` option is not supplied, `pbs_mom` will attempt to open the default configuration file "config" in `PBS_HOME/mom_priv`. If this file is not present, `pbs_mom` will log the fact and continue.
`-d directory`	Specifies the path of the directory which is the home of the servers working files, `PBS_HOME`. This option is typically used along with `-M` when debugging MOM. The default directory is given by `PBS_HOME` which is typically `/usr/spool/PBS`.
`-L logfile`	Specify an absolute path name for use as the log file. If not specified, MOM will open a file named for the current date in the `PBS_HOME/mom_logs` directory, see the `-d` option.
`-M port`	Specifies the port number on which MOM will listen for batch requests.
`-n nice_val`	Specifies the priority value of the daemon when it executes.
`-R port`	Specifies the port number on which 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.
`-x`	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.

11.2.2 Manually Starting the Server

Normally the PBS Server is started from the system boot file via a line such as:

```
/usr/pbs/sbin/pbs_server [options]
```

The command line options for the Server include:

`-a active`	Specifies if scheduling is active or not. This sets the server attribute `scheduling`. If the option argument is "true" ("True", "t", "T", or "1"), the server is *active* and the PBS job scheduler will be called. If the argument is "false" ("False", "f", "F", or "0), the server is *idle*, and the scheduler will not be called and no jobs will be run. If this option is not specified, the server will retain the prior value of the `scheduling` attribute.
`-d config`	Specifies the path of the directory which is home to the Server's configuration files, `PBS_HOME`. A host may have multiple Servers. Each Server must have a different configuration directory. The default configuration directory is `PBS_HOME` which is typically set to `/usr/spool/PBS`.
`-e mask`	Specifies a log event mask to be used when logging. See "log_events" on page 68.
`-p port`	Specifies the port number on which the Server will listen for batch requests. If multiple Servers are running on a single host, each must have its own unique port number. This option is for use in testing with multiple PBS systems on a single host.
`-A acctfile`	Specifies an absolute path name of the file to use as the accounting file. If not specified, the file is named for the current date in the `PBS_HOME/server_priv/accounting` directory.
`-L logfile`	Specifies an absolute path name of the file to use as the log file. If not specified, the file is one named for the current date in the `PBS_HOME/server_logs directory`, see the `-d` option.
`-M mom_port`	Specifies the host name and/or port number on which the server should connect to the MOMs. The option argument, *mom_port*, is

one of the forms: `host_name`, `[:]port_number`, or `host_name:port_number`. If `host_name` not specified, the local host is assumed. If `port_number` is not specified, the default port is assumed. See the `-M` option for `pbs_mom`.

`-R RPPport` Specifies the port number on which the Server should query the status of MOM. See the `-R` option for `pbs_mom`.

`-g globus_port` Specifies the host name and/or port number on which the Server should connect the PBS MOM Globus daemon. The option argument, *globus_port*, is one of the forms: `host_name`, `[:]port_number`, or `host_name:port_number`. If `host_name` not specified, the local host is assumed. If `port_number` is not specified, the default port is assumed.

`-G globus_RPP` Specifies the port number on which the Server should query the status of PBS MOM Globus daemon.

`-S sched_port` Specifies the port number to which the Server should connect when contacting the Scheduler. The option argument, *sched_port*, is of the same syntax as under the `-M` option.

`-t type` Specifies the impact on jobs when the Server restarts. *type* argument is:

Option	Effect Upon Job Running Prior to Server Shutdown
`hot`	All jobs in the Running state are retained in that state. Any job that was requeued into the Queued state from the Running state when the server last shut down will be run immediately, assuming the required resources are available. This returns the server to the same state as when it went down. After those jobs are restarted, then normal scheduling takes place for all remaining queued jobs. All other jobs are retained in their current state. If a job cannot be restarted immediately because of a missing resource, such as a node being down, the server will attempt to restart it periodically for up to 5 minutes. After that period, the server will revert to a normal state, as if warm started, and will no longer attempt to restart any remaining jobs which were running prior to the shutdown.

Option	Effect Upon Job Running Prior to Server Shutdown
`warm`	All jobs in the Running state are retained in that state. All other jobs are maintained in their current state. The job scheduler will typically make new selections for which jobs are placed into execution. Warm is the default if `-t` is not specified.
`cold`	All jobs are purged. Positive confirmation is required before this direction is accepted.
`create`	The server will discard any existing configuration files: server, nodes, queues and jobs, and initialize configuration files to the default values. The server is idled (scheduling set false).

11.2.3 Manually Starting the Scheduler

The Scheduler should also be started at boot time. If starting by hand, use the following command line:

```
/usr/pbs/sbin/pbs_sched [options]
```

There are no required options for the Standard Scheduler. Available options are listed below.

`-a alarm`	This specifies the time in seconds to wait for a schedule run to finish. If a script takes too long to finish, an alarm signal is sent, and the scheduler is restarted. If a core file does not exist in the current directory, `abort`() is called and a core file is generated. The default for *alarm* is 180 seconds.
`-d home`	This specifies the PBS home directory, `PBS_HOME`. The current working directory of the scheduler is `PBS_HOME/sched_priv`. If this option is not given, `PBS_HOME` defaults to `PBS_HOME` as defined in the `/etc/pbs.conf` file.
`-L logfile`	Specifies an absolute path name of the file to use as the log file. If not specified, the scheduler will open a file named for the current date in the `PBS_HOME/sched_logs` directory; see the `-d` option.

`-p file`	This specifies the "print" file. Any output from the Scheduler which is written to standard out or standard error will be written to this file. If this option is not given, the file used will be `PBS_HOME/sched_priv/sched_out`. See the `-d` option.
`-S port`	This specifies the port to use. If this option is not given, the default port for the PBS Scheduler is used.
`-R port`	This specifies the resource monitor port to use. If this option is not given, the default port for the PBS MOM is used.
`-c file`	Specify a configuration file. If this is a relative file name it will be relative to `PBS_HOME/sched_priv`, see the `-d` option. If the `-c` option is not supplied, `pbs_sched` will not attempt to open a configuration file.

The options that specify file names may be absolute or relative. If they are relative, their root directory will be `PBS_HOME/sched_priv`.

11.2.4 Manually Starting Globus MOM

The optional Globus MOM should be started at boot time if Globus support is desired. Note that the provided startup script does not start the Globus MOM. There are no required options. If starting manually, run it with the line:

```
/usr/pbs/sbin/pbs_mom_globus [options]
```

If Globus MOM is taken down and the host system continues to run, the Globus MOM should be restarted with the `-r` option. This directs Globus MOM to kill off processes running on behalf of a Globus job. See the **PBS External Reference Specificiation** (or the `pbs_mom_globus`(1B) manual page) for a more complete explanation.

If the `pbs_mom_globus` daemon is restarted without the `-r` option, the assumption that will be made is that jobs have become disconnected from the Globus gatekeeper due to a system restart (cold start). Consequentially, `pbs_mom_globus` will request that any Globus jobs that were being tracked and which where running be canceled and requeued.

11.3 Starting and Stopping PBS: Windows 2000 / XP

When PBS Pro is installed on either Microsoft Windows XP or 2000, the PBS daemons are registered as system services. As such, they will be automatically started and stopped when the systems boots and shuts down. However, there may come a time with you need to manually stop or restart the PBS daemons (such as shutting them down prior to a PBS software upgrade). The following example illustrates how to manually stop and restart the PBS daemons.

```
net stop pbs_sched
net stop pbs_mom
net stop pbs_server
net stop pbs_rshd
#     and to restart PBS:
net start pbs_server
net start pbs_mom
net start pbs_sched
net start pbs_rshd
```

11.4 Stopping PBS

PBS Pro version 5.1 introduced additional options to the `qterm` command that enable the administrator to shutdown, selectively, or inclusively, the various PBS daemons.

`-m`	directs that all `pbs_mom`s should be shutdown
`-s`	directs that the `pbs_sched` should be shut down

Thus, the following command will bring down the entire PBS complex:

```
# qterm -s -m
```

There is a third option ("`-t type`") available for the `qterm` command, which takes one of three arguments for the *type* of shutdown you wish to perform.

***type* of shutdown**	**Description**
`immediate`	All running jobs are to immediately stop execution. If checkpoint is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If checkpoint is not supported or the job cannot be checkpointed, running jobs are requeued if the rerunable attribute is true. Otherwise, jobs are killed. Normally the Server will not shutdown until there are no jobs in the running state. If the Server is unable to contact the MOM of running job, the job is still listed as running. The Server may be forced down by a second "`qterm -t immediate`" command.
`delay`	If checkpoint is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If a job cannot be checkpointed, but can be rerun, the job is terminated and requeued. Otherwise, running jobs are allowed to continue to run. Note, the operator or administrator may use the `qrerun` and `qdel` commands to remove running jobs.
`quick`	This is the default action if the `-t` option is not specified. This option is used when you wish that running jobs be left running when the Server shuts down. The Server will cleanly shutdown and can be restarted when desired. Upon restart of the Server, jobs that continue to run are shown as running; jobs that terminated during the Server's absence will be placed into the exiting state.

Important: Note that `qterm` now defaults to `qterm -t quick.` Also, note that the Server now does a quick shutdown upon receiving `SIGTERM`.

Important: Should you ever have the need to stop a MOM daemon but leave jobs managed by her running, the only way to do this is to kill MOM with a `SIGKILL` (`-9`).

11.5 Checkpoint/Restart Under PBS

PBS Pro supports operating system level checkpoint-restart where provided by the system. Currently both SGI IRIX and Cray UNICOS provide OS-level checkpoint packages, which PBS uses. (User's may optionally manage their own checkpointing from within their application. This is discussed further in the **PBS User Guide**.)

The location of the directory into which jobs are checkpointed can now be specified in a number of ways. In order of preference:

1. "`-C path`" command line option to `pbs_mom`
2. **`PBS_CHECKPOINT_PATH`** environment variable
3. "`$checkpoint_path path`" option in MOM's config file
4. default value

11.5.1 Manually Checkpointing a Job

On systems which provide OS-level checkpointing, the PBS administrator may manually force a running job to be checkpointed. This is done by using the `qhold` command. (Discussed in detail in the **PBS Users Guide**).

11.5.2 Checkpointing Jobs During PBS Shutdown

As installed, the `/etc/init.d/pbs` script will not result in PBS checkpointing jobs (on systems which provide OS-level checkpointing). This behavior allows for a faster shut down of the batch system at the expense of rerunning jobs from the beginning. If you prefer jobs to be checkpointed, then append the `-t immediate` option to the `qterm` statement in the script.

11.5.3 Suspending/Checkpointing Multi-node Jobs

The PBS suspend/resume and checkpoint/restart capabilities is supported for multi-node jobs. With checkpoint (on systems which provide OS-level checkpointing), the system must be able to save the complete session state in a file. This means any open socket will cause the operation to fail. PBS normally sets up a socket connection to a process (`pbs_demux`) which collects stdio streams from all tasks. If this is not turned off, the checkpoint cannot work. Therefore, a new job attribute has been added: `no_stdio_sockets`. See the `pbs_job_attributes` manual page for more details. If this attribute is true, the `pbs_demux` process will not be started and no open socket will prevent the checkpoint from working. The other place where PBS will use a socket that must be addressed is if the program `pbsdsh` is used to spawn tasks. There is a new option for `pbsdsh` '`-o`' that is used to prevent it from waiting for the spawned tasks to finish. This is done so no socket will be left open to the MOM to receive task manager events. If this is used, the shell must use some other method to wait for the tasks to finish.

11.5.4 Checkpointing Jobs Prior to SGI IRIX Upgrade

Under the SGI Irix operating system, the normal checkpoint procedure does not save shared libraries in the restart image in order to reduce the image size and time required to write it. This type of image cannot be restarted following a Irix operating system upgrade. In order to produce an image which can be restarted following an upgrade, a special flag is required when calling checkpoint. With PBS Pro 5.2, `pbs_mom` has a new `config` file option `$checkpoint_upgrade` which if present causes PBS to use the special upgrade checkpoint flag. It is recommended that this flag be set (and `pbs_mom` be restarted via `SIGHUP`) only when shutting down PBS just prior to upgrading your system.

11.6 Start/Stop/Enable/Disable Queues

In addition to the functionality offered by `qmgr`, PBS provides separate commands for manipulating the status of queues. This section briefly describes the four commands. See the corresponding manual pages for details of use.

The `qstart` command directs that a destination should process batch jobs. If the destination is an execution queue, the Server will begin to schedule jobs that reside in the queue for execution. If the destination is a routing queue, the Server will begin to route jobs from that queue

The `qstop` command directs that a destination should stop processing batch jobs. If the destination is an execution queue, the Server will cease scheduling jobs that reside in the queue for execution. If the destination is a routing queue, the Server will cease routing jobs from that queue.

The `qenable` command directs that a destination should accept batch jobs. This command sends a Manage request to the batch Server specified on the command line. If the command is accepted, the destination will accept Queue Job requests which specify the queue.

The `qdisable` command directs that a destination should no longer accept batch jobs. If the command is accepted, the destination will no longer accept Queue Job requests which specified the disabled queue. Jobs which already reside in the queue will continue to be processed. This allows a queue to be "drained."

11.7 Security

There are three parts to security in the PBS system:

Internal security	Can the daemons be trusted?
Authentication	How do we believe a client about who it is.
Authorization	Is the client entitled to have the requested action performed.

11.8 Internal Security

A significant effort has been made to insure the various PBS daemon themselves cannot be a target of opportunity in an attack on the system. The two major parts of this effort is the security of files used by the daemons and the security of the daemons environment.

Any file used by PBS, especially files that specify configuration or other programs to be run, must be secure. The files must be owned by root and in general cannot be writable by anyone other than root.

A corrupted environment is another source of attack on a system. To prevent this type of attack, each daemon resets its environment when it starts. The source of the environment is a file named by **PBS_ENVIRONMENT** set by the configure option `--set-environ`, defaulting to `*PBS_HOME*/pbs_environment`. If it does not already exists, this file is created during the install process. As built by the install process, it will contain a very basic path and, if found in root's environment, the following variables: **TZ**, **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME**. The `environment` file may be edited to include the other variables required on your system.

Important: Please note that **PATH** must be included. This value of **PATH** will be passed on to batch jobs. To maintain security, it is important that **PATH** be restricted to known, safe directories. Do NOT include "." in **PATH**. Another variable which can be dangerous and should not be set is **IFS**.

The entries in the **PBS_ENVIRONMENT** file can take two possible forms:

```
variable_name=value
variable_name
```

In the later case, the value for the variable is obtained from the daemons environment before the environment is reset.

11.8.1 Host Authentication

PBS uses a combination of information to authenticate a host. If a request is made from a client whose socket is bound to a privileged port (less than 1024, which requires root privilege), PBS believes the IP (Internet Protocol) network layer as to whom the host is. If the client request is from a non-privileged port, the name of the host which is making a client request must be included in the credential sent with the request and it must match the IP network layer opinion as to the host's identity.

11.8.2 Host Authorization

Access to the Server from another system may be controlled by an access control list (ACL). Access to `pbs_mom` is controlled through a list of hosts specified in the `pbs_mom`'s configuration file. By default, only "localhost" and the name returned by `gethostname`(2) are allowed. See the man pages `pbs_mom`(8B) for more information on the configuration file. Access to `pbs_sched` is not limited other than it must be from a privileged port.

11.8.3 User Authentication

The PBS Server authenticates the user name included in a request using the supplied PBS credential. This credential is supplied by `pbs_iff`.

11.8.4 User Authorization

PBS as shipped does not assume a consistent user name space within the set of systems which make up a PBS cluster. However, the administrator can enable this assumption, if desired. By default, the routine `site_map_user`() is called twice, once to map the name of the requester and again to map the job owner to a name on the Server's (local) system. If the two mappings agree, the requester is considered the job owner.

If running PBS in an environment that *does* have a flat user namespace, the administrator can disable these checks by setting the `flatuid` Server attribute to `True` via `qmgr`:

```
# qmgr
Qmgr: set server flatuid=True
```

If `flatuid` is set to `true`, a UserA on HostX who submits a job to the PBS Server on

HostY will *not* require an entry in the /etc/passwd file, nor a .rhosts entry on HostY for HostX, nor must HostX appear in HostY's `/etc/hosts.equiv` file.

In either case, if a job is submitted by *UserA@hostA* PBS will allow the job to be deleted or altered by *UserA@hostB*

Important: When flatuid is true, a password entry for the submitting user is not required on the server system. But, without the password entry, the Server does not know the user's group (it is not passed implicitly from qsub). Without knowing the group, the acl_group attribute on a queue cannot be checked. The default of "not allowed" will be returned. This applies to acl_group on the Server or Queue level.

If `flatuid` is NOT set to `true`, a user may supply a name under which the job is to be executed on a certain system. If one is not supplied, the name of the job owner is chosen to be the execution name-- see the `-u user_list` option of the `qsub`(1B) command. Authorization to execute the job under the chosen name is granted under the following conditions:

1. The job was submitted on the Server's (local) host and the submitter's name is the same as the selected execution name.

2. The host from which the job was submitted is declared trusted by the execution host in the `/etc/hosts.equiv` file or the submitting host and submitting user's name are listed in the execution users' `.rhosts` file. The system supplied library function, `ruserok`*()*, is used to make these checks.

If the above test to determine authorization are not sufficient to a site, the routine `site_check_user_map`*()* in the file `src/server/site_check_u.c` may be modified.

In addition to the above checks, access to a PBS Server and queues within that Server may be controlled by access control lists. (For details see "Server Attributes" on page 66 and "Queue Attributes" on page 72.)

11.8.5 Group Authorization

PBS allows a user to submit jobs and specify under which group the job should be executed. The user specifies a `group_list` attribute for the job which contains a list of group@host similar to the user list. See the `group_list` attribute under the -W option of `qsub`(1B). The PBS Server will ensure that the user is a member of the specified group by:

1. Checking if the specified group is the user's primary group in the password entry on the execution host. In this case the user's name does not have to appear in the group entry for his primary group.

2. Checking on the execution host for the user's name in the specified group entry in `/etc/group`.

The job will be aborted if both checks fail. The checks are skipped if the user does not supply a group list attribute. In this case the user's primary group from the password file will be used.

When staging files in or out, PBS also uses the selected execution group for the copy operation. This provides normal UNIX access security to the files. Since all group information is passed as a string of characters, PBS cannot determine if a numeric string is intended to be a group name or GID. Therefore when a group list is specified by the user, PBS places one requirement on the groups within a system: each and every group in which a user might execute a job MUST have a group name and an entry in `/etc/group`. If no `group_list` are ever used, PBS will use the login group and will accept it even if the group is not listed in `/etc/group`. Note, in this latter case, the `egroup` attribute value is a numeric string representing the GID rather than the group "name".

11.9 External Security

In addition to the security measures discussed above, PBS provides three levels of privilege: user, operator, and Manager. Users have user privilege which allows them to manipulate their own jobs. Manager or Operator privilege is required to set or unset attributes of the Server, queues, nodes, other peoples jobs, with the following exceptions:

Manager privilege is required to create and delete queues or nodes, and set/alter/unset:

node properties
server acl_host_enable
server acl_host_list
server acl_user_enable
server acl_users
server acl_roots

server managers
server operators
server query_other_jobs
server comment
server default_node
server node_pack

11.10 Root Owned Jobs

The Server will reject any job which would execute under the UID of zero unless the owner of the job, typically root on this or some other system, is listed in the Server attribute `acl_roots`.

11.11 Managing PBS and Multi-node Parallel Jobs

Many customers use PBS Pro in cluster configurations for the purpose of managing multi-node parallel applications. This sections provides the PBS administrator with information specific to this situation.

11.11.1 Interfacing MPICH with PBS Pro

If you are running an open source version of MPI, such as MPICH, then the `mpirun` command can be modified to check for the PBS environment and use the PBS supplied host file. In the case of MPICH, this is easily done by editing the `.../mpich/bin/mpirun.args` file and adding the following near line 40 (depending on the version being used):

```
if [ "$PBS_NODEFILE" != "" ]
then
     machineFile=$PBS_NODEFILE
fi
```

Important: Additional information regarding checkpointing of parallel jobs is given in “Suspending/Checkpointing Multi-node Jobs” on page 153.

11.12 SGI Job Container / Limits Support

SGI Job Container/Limit support has been added to PBS Pro. Each PBS job is placed in its own SGI Job container. Limits on the job are set as the `MIN(ULDB limit, PBS resource_list limit)`. The ULDB domains are set in the following order:

PBS_{queue name}
PBS
batch

Limits are set for the following resources: "cput" and "vmem". A job limit is NOT set for "mem" because the kernel does not factor in shared memory segments among `sproc`() processes, thus the system report usage is too high.

11.13 Job Prologue/Epilogue Scripts/Programs

PBS provides the ability for the administrator run a site supplied script (or program) before (`prologue`) and/or after (`epilogue`) each job runs. This provides the capability to perform initialization or cleanup of resources, such as temporary directories or scratch files. The scripts may also be used to write “banners” on the job’s output files. When multiple nodes are allocated to a job, these scripts are only run by the “Mother Superior”, the pbs_mom on the first node allocated. This is also where the job shell script is run. Note that both the prologue and epilogue are run with root privilege (not as the user), and neither are included in the job session, thus the prologue cannot be used to modify the job environment or change limits on the job.

If a `prologue` or `epilogue` script is not present, MOM continues in a normal manner. If present, the script is run with root privilege. In order to be run, the script must adhere to the following rules:

- The script must be in the `/usr/spool/PBS/mom_priv` directory with the name `prologue` for the script to be run before the job and the name `epilogue` for the script to be run after the job.
- The script must be owned by root.
- The script must be readable and executable by root.
- The script cannot be writable by anyone but root.

The “script” may be either a shell script or an executable object file. Typically, a shell script should start with a line of the form:

```
    #! /path/interpreter
```

For more information, see the rules described under `execve`(2) or `exec`(2) on your system.

11.13.1 Prologue and Epilogue Arguments

When invoked, the prologue is called with the following arguments:

argv[1]	the job id.
argv[2]	the user name under which the job executes.
argv[3]	the group name under which the job executes.

The epilogue is called with the above, plus:

argv[4]	the job name.
argv[5]	the session id.
argv[6]	the requested resource limits (list).
argv[7]	the list of resources used
argv[8]	the name of the queue in which the job resides.
argv[9]	the account string, if one exists.

For both the `prologue` and `epilogue`:

envp	The environment passed to the script is null.
cwd	The current working directory is the user's home directory.
input	When invoked, both scripts have standard input connected to a system dependent file. Currently, this file is `/dev/null`.
output	With one exception, the standard output and standard error of the scripts are connected to the files which contain the standard output and error of the job. If a job is an interactive PBS job, the standard output and error of the epilogue is pointed to `/dev/null` because the pseudo terminal connection used was released by the system when the job terminated.

11.13.2 Prologue Epilogue Time Out

To prevent an error condition within the prologue or epilogue from delaying PBS, MOM places an alarm around the scripts execution. This is currently set to 30 seconds. If the alarm sounds before the scripts has terminated, MOM will kill the script. The alarm value can be changed `$prologuealarm` MOM configuration parameter.

11.13.3 Prologue Error Processing

Normally, the prologue script should exit with a zero exit status. MOM will record in her log any case of a non-zero exit from a script. Exit status values and their impact on the job are:

-4	The script timed out (took too long). The job will be requeued.
-3	The `wait`(2) call waiting for the script to exit returned with an error. The job will be requeued.
-2	The input file to be passed to the script could not be opened. The job will be requeued.
-1	The script has a permission error, it is not owned by root and or is writable by others than root. The job will be requeued.
0	The script was successful. The job will run.
1	The script returned an exit value of 1, the job will be aborted.
>1	The script returned a value greater than one, the job will be requeued.

The above apply to normal batch jobs. Interactive-batch jobs (`-I` option) cannot be requeued on a non-zero status. The network connection back to `qsub` is lost and cannot be re-established. Interactive jobs will be aborted on any non-zero prologue exit.

Important: The administrator must exercise great caution in setting up the prologue to prevent jobs from being flushed from the system.

Epilogue script exit values which are non-zero are logged, but have no impact on the state of the job.

11.14 Use and Maintenance of Logfiles

The PBS system tends to produce lots of logfile entries. There are two types of logfiles: the event logs which record events from each PBS daemon (`pbs_server`, `pbs_mom`, and `pbs_sched`) and the PBS accounting log.

11.14.1 PBS Events

The amount of output in the PBS event logfiles depends on the specified log filters for each daemon. All three PBS daemons can be directed to record only messages pertaining to certain event types. The specified events are logically "or-ed" to produce a mask representing the events the local site wishes to have logged. The available events, and corresponding decimal and hexidecimal values are show below.

Value	Hex	Event Description
1	0x1	Internal PBS Errors.
2	0x2	System (OS) Errors, such as malloc failure.
4	0x4	Administrator related events, such as changing queue attributes.
8	0x8	Job related events: submitted, ran, deleted, ...
16	0x10	Job Resource Usage.
32	0x20	Security related events, such as attempts to connect from an unknown host.
64	0x40	When the scheduler was called and why.
128	0x80	First level, common, debug messages.
256	0x100	Second level, more rare, debug messages.

Everything turned on is of course 511. 127 is a good value to use. The event logging mask is controlled differently for the different daemons. The following table shows the log event attribute for each daemon, and page reference for details.

Daemon	Attribute and Reference	Notes
Server	See “log_events” on page 68.	Takes effect immediately with `qmgr`
MOM	See “$logevent” on page 97.	Requires `SIGHUP` to MOM
Scheduler	See “log_filter” on page 120.	Requires `SIGHUP` to Scheduler

11.14.2 The Event Logfiles

Each PBS daemon maintains separate event logfiles. The logfiles default to a file with the current date as the name in the `/usr/spool/PBS/(`*`daemon`*`)_logs` directory. This location can be overridden with the "`-L pathname`" option where pathname must be an absolute path.

If the default logfile name is used (no `-L` option), the log will be closed and reopened with the current date daily. This happens on the first message after midnight. If a path is given with the `-L` option, the automatic close/reopen does not take place. All daemons will close and reopen the same named log file on receipt of **SIGHUP**. The process identified (PID) of the daemon is available in its lock file in its home directory. Thus it is possible to move the current log file to a new name and send **SIGHUP** to restart the file:

```
# cd /usr/spool/PBS/DAEMON_logs
# mv current archive
# kill -HUP ‘cat ../DAEMON_priv/daemon.lock‘
#
```

11.14.3 Event Logfile Format

The daemon event logfile is a text file with each entry terminated by a new line. The format of an entry is:

```
date-time;event_code;server_name;object_type;object_name;message
```

The `date-time` field is a date and time stamp in the format:

`mm/dd/yyyy hh:mm:ss`.

The `event_code` is the type of event which triggered the event logging. It corresponding to the bit position, 0 to n, in the event mask (discussed above) of the daemon writing the event record.

The `server_name` is the name of the Server which logged the message. This is recorded in case a site wishes to merge and sort the various logs in a single file.

The `object_type` is the type of object which the message is about:

`Svr`	for server
`Que`	for queue
`Job`	for job
`Req`	for request
`Fil`	for file.

The `object_name` is the name of the specific object. `message_text` field is the text of the log message.

When reading the PBS event logfiles, you may see messages of the form "Type 19 request received from PBS_Server...". These "type codes" coorespond to different PBS batch requests. Appendix B contains a listing of all "types" and each corresponding batch request.

11.14.4 The Accounting Log

The PBS Server daemon maintains an accounting log. The log name defaults to `/usr/spool/PBS/server_priv/accounting/`*`yyyymmdd`* where *`yyyymmdd`* is the date. The accounting log files may be placed elsewhere by specifying the `-A` option on the `pbs_server` command line. The option argument is the full (absolute) path name of the file to be used. If a null string is given, then the accounting log will not be opened and no accounting records will be recorded. For example

```
# pbs_server -A ""
#
```

The accounting file is changed according to the same rules as the log files. If the default file is used, named for the date, the file will be closed and a new one opened every day on the first event (write to the file) after midnight. With either the default file or a file named with the `-A` option, the Server will close the accounting log and reopen it upon the receipt of a **SIGHUP** signal. This allows you to rename the old log and start recording again on an empty file. For example, if the current date is February 9, 2001 the Server will be writing in the file `20010209`. The following actions will cause the current accounting file to be renamed `feb9` and the Server to close the file and starting writing a new `20010209`.

```
# mv 20010209 feb9
# kill -HUP 1234     (the Server’s pid)
#
```

11.14.5 Accounting Log Format

The PBS accounting file is a text file with each entry terminated by a new line. The format of an entry is:

```
date time;record_type;id_string;message_text
```

The `date time` field is a date and time stamp in the format:

```
mm/dd/yyyy hh:mm:ss
```

The `id_string` is the job, reservation, or reservation-job identifier. The `messge_text` is ascii text. The content depends on the record type. The message text format is blank separated keyword=value fields. The `record_type` is a single character indicating the type of record. The types are:

`A`	Job was aborted by the server.
`B`	Beginning of reservation’s period. The `message_text` field contains details about the specified advance reservation. Possible attributes include :

Attribute	Explanation
`owner=`ownername	name of party who submitted the resources reservation request.
`name=`reservation_name	if submitter supplied a name string for the reservation.

Attribute	Explanation
`account=`account_string	if submitter supplied a to be recorded in accounting.
`queue=`queue_name	the name of the instantiated reservation queue if this is a general resources reservation. If the resources reservation is for a reservation job, this is the name of the queue to which the reservation-job belongs.
`ctime=`creation_time	time at which the resources reservation got created, seconds since the epoch.
`start=`period_start	time at which the reservation period is to start, in seconds since the epoch.
`end=`period_end	time at which the reservation period is to end, seconds since the epoch.
`duration=`reservation_duration	the duration specified or computed for the resources reservation, in seconds.
`nodes=`reserved_nodes	if nodes with specified properties are required, this string is the allocated set.
`authorized_users=`users_list	the list of acl_users on the queue that is instantiated to service the reservation.
`authorized_groups=`groups_list	if specified, the list of acl_groups on the queue that is instantiated to service the reservation.
`authorized_hosts=`hosts_list	if specified, the list of acl_hosts on the queue that is instantiated to service the reservation.
`resource_list=`resources_list	list of resources requested by the reservation. Resources are listed individually as, for example: resource_list.ncpus=16 resource_list.mem=1048676.

`C` Job was checkpointed and held.

`D` Job was deleted by request. The `message_text` will contain `requestor=user@host` to identify who deleted the job.

`E` Job ended (terminated execution). The `message_text` field contains detailed information about the job. Possible attributes include:

Attribute	Explanation
`user=`username	the user name under which the job executed.
`group=`groupname	the group name under which the job executed.
account=account_string	if job has an "account name" string.
`jobname=`job_name	the name of the job.
`queue=`queue_name	the name of the queue from which the job is executed.
`resvname`=reservation_name	if job draws its resources from a resources reservation and that reservation has a name.
`resvID`=reservation_ID_string	if job draws its resources from a resources reservation.
`resvjobID=`reservation_ID_string	if job is as a "reservation-job" (advance reservation of resources).
`ctime=`time	time in seconds when job was created (first submitted).
`qtime=`time	time in seconds when job was queued into current queue.
`etime=`time	time in seconds when job became eligible to run; no holds, etc.
`start=`time	time in seconds when job execution started.
`exec_host=`host	name of host on which the job is being executed.
`Resource_List.`resource=limit	list of the specified resource limits.

Attribute	**Explanation**
`session=`sesid	session number of job.
`alt_id=`id	Optional alternate job identifier. Included only for certain systems: Irix 6.x with Array Services - The alternate id is the Array Session Handle (ASH) assigned to the job. For irix6cpuset mom, the alternate id holds the name of the cpuset assigned to the job as well as resources assigned to the job. For example, alt_id=cpuset=357.sgi3:1024kb/1p
`end=`time	time in seconds when job ended execution.
`Exit_status=`value	the exit status of the job. If the value is less than 10000 (decimal) it is the exit value of the top level process of the job, typically the shell. If the value is greater than 10000, the top process exited on a signal whose number is given by subtracting 10000 from the exit value.
`Resources_used.resource=` limit	list of the specified resource limits.

`F` Resources reservation period finished.

`K` Scheduler or server requested removal of the reservation. The `message_text` field contains: `requestor=user@host` to identify who deleted the resources reservation.

`k` Resources reservation terminated by ordinary client - e.g. an owner issuing a `pbs_rdel` command. The `message_text` field contains: `requestor=user@host` to identify who deleted the resources reservation.

`Q` Job entered a queue. The message_text contains `queue=`name identifying the queue into which the job was placed. There will be a new Q record each time the job is routed or moved to a new (or the same) queue.

`R` Job was rerun.

`S` Job execution started. The `message_text` field contains:

Attribute	Explanation
`user=`username	the user name under which the job executed.
`group=`groupname	the group name under which the job executed.
`jobname=`job_name	the name of the job.
`queue=`queue_name	the name of the queue from which the job is executed.
`ctime=`time	time in seconds when job was created (first submitted).
`qtime=`time	time in seconds when job was queued into current queue.
`etime=`time	time in seconds when job became eligible to run; no holds, etc.
`start=`time	time in seconds when job execution started.
`exec_host=`host	name of host on which the job is being executed.
`Resource_List.resource=` limit	list of the specified resource limits.
`session=`sesid	session number of job.

`T` Job was restarted from a checkpoint file.

`U` Created unconfirmed resources reservation on Server. The `message_text` field contains `requestor=user@host` to identify who requested the resources reservation.

`Y` Resources reservation confirmed by the Scheduler. The `message_text` field contains the same item (items) as in a `U` record type.

For `Resource_List` and `Resources_used`, there is one entry per resource.

11.15 Using the syslog facility with PBS Pro

Traditionally, each PBS daemon logged various levels of information about events in its own log file. While having the advantage of a concise location for the information from a daemon, the disadvantage is that in a cluster, the logged information is scattered across each execution host.

If your site uses the `syslog` subsystem, PBS may be configured to make full use of it. The following entries in `/etc/pbs.conf` control the use of `syslog` by the PBS daemons:

PBS_CONF_SYSLOG=x	This variable controls the use of syslog and syslog "facility" under which the entries are logged. If x is: 0 - no syslogging 1 - logged via LOG_DAEMON facility 2 - logged via LOG_LOCAL0 facility 3 - logged via LOG_LOCAL1 facility ... 9 - logged via LOG_LOCAL7 facility
PBS_CONF_SYSLOGSEVR=y	This variable controls the severity level of messages that are logged, See `/usr/include/sys/syslog.h`. If y is: 0 - only LOG_EMERG messages are logged 1 - messages up to LOG_ALERT are logged ... 7 - messages up to LOG_DEBUG are logged

Important: `PBS_CONF_SYSLOGSEVR` is used in addition to PBS's `log_event` mask which controls the class of events (job, node, ...) that are logged.

11.16 Interpreting PBS Exit Codes

The PBS Server logs and accounting logs record an “exit status” of jobs. Zero or positive exit status is the status of the top level shell. Certain negative exit status are used internally and will never be reported to the user. The positive exit status values indicate which signal killed the job. Depending on the system, values greater than 128 (or on some systems 256, see wait(2) or waitpid(2) for more information) are the value of the signal that killed the job. To interpret (or "decode") the signal contained in the exit status value, subtract the base value from the exit status. For example, if a job had an exit status of 143, that indicates the jobs was killed via a SIGTERM (e.g. 143 - 128 = 15, signal 15 is SIGTERM). See the kill(1) manual page for a mapping of signal numbers to signal name on your operating system.

11.17 PBS tracejob Command

PBS includes `tracejob` utility to extract log messages for a particular job (from all log files available on the local host) and print them sorted into chronological order. Usage for the `tracejob` command is:

```
   tracejob [-a|s|l|m|v] [-w size] [-p path] [-n days] [-f filter]
```

The available options, and description of each, follows.

Option	Description
`-a`	Don't use accounting log files
`-c`	What message count is considered excessive
`-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 equiv
`-l`	Don't use scheduler log files
`-m`	Don't use mom log files
`-n`	Number of days in the past to look for job(s) [default 1]
`-p`	Path to PBS_SERVER_HOME
`-s`	Don't use server log files
`-w`	Number of columns of your terminal
`-v`	Verbose mode - show more error messages
`-z`	Toggle filtering excessive messages

Following example request all log message for a particular job from today's (the default date) log file.

```
% tracejob 475
Job: 475.riverside.pbspro.com
03/10/2002 14:29:15 S enqueuing into workq, state 1 hop 1
03/10/2002 14:29:15 S Job Queued at request of jjones, owner=
         jjones@mars.pbspro.com, job name = STDIN
03/10/2002 15:06:30 S Job Modified at request of Scheduler
03/10/2002 15:06:30 L Considering job to run
03/10/2002 15:06:30 S Job Run at request of Scheduler
03/10/2002 15:06:32 L Job run on node mars
03/10/2002 15:06:32 M Started, pid = 25282
03/10/2002 15:06:32 M Terminated
03/10/2002 15:06:32 M task 1 terminated
03/10/2002 15:06:32 M kill_job
03/10/2002 15:06:32 S Obit received
03/10/2002 15:06:32 S dequeuing from workq, state 5
03/10/2002 15:06:32 A user=jwang group=mrj jobname=subrun
         queue=workq ctime=1026928565 qtime=1026928565
         etime=1026928565 start=1026928848 exec_host=south/0
         Resource_List.arch=linux Resource_List.ncpus=1
         Resource_List.walltime=00:10:00 session=6022
         end=1026929149 Exit_status=0 resources_used.ncpus=1
         resources_used.cpupercent=0 resources_used.vmem=498kb
         resources_used.cput=00:00:00 resources_used.mem=224kb
         resources_used.walltime=00:05:01
```

Note that third column of the body of the display contains a single letter abbreviation (S, M, A, or L) which corresponds to the source of the log message (Server, MOM, Accounting, or Local-policy Scheduler log files).

11.18 pbsnodes Command

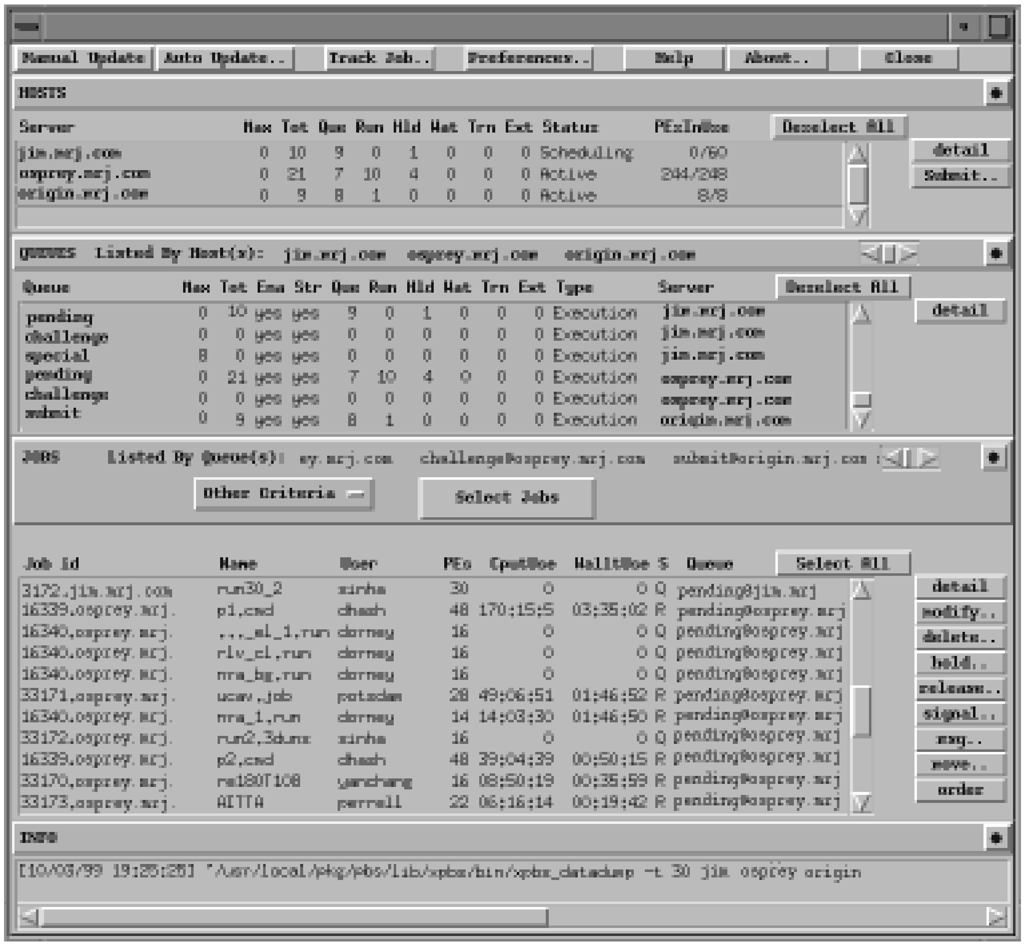
The `pbsnodes` command is used to query the status of nodes, or mark nodes down, free or off-line. Node information is obtained by sending a request to the PBS Server.

pbsnodes Usage	Action
`(no option)`	Prints the command usage syntax
`node1 node2`	Prints the status of nodes `node1` and `node2`
`-d node1`	The nodes specified as operands are marked DOWN and unavailable to run jobs. It is important that all the nodes known to be down are given as arguments on the command line. This is because nodes which are not listed are assumed to be UP and will be indicated as such if they were previously marked DOWN. I.e., "pbsnodes -d" will mark all nodes as free.
`-a`	All nodes and all their attributes are listed.
`-c node2`	Clear OFFLINE or DOWN from listed nodes. The listed nodes are "free" to be allocated to jobs.
`-l`	List all nodes marked in any way.
`-o node3`	Mark listed nodes as OFFLINE even if currently in use. This is different from being marked DOWN. An automated script that checks nodes being up or down and calls `pbsnodes` with a list of nodes down will not change the status of nodes marked OFFLINE. This gives the administrator a tool to hold a node out of service without changing the automatic script.
`-r node3`	Clear OFFLINE from listed nodes.
`-s`	Specify the PBS Server to which to connect.

Important: Only the `-d` option will change the marking for nodes which are not given on the command line.

11.19 PBS Graphical Interfaces (GUIs)

PBS currently provides two Graphical User Interfaces (GUIs): `xpbs` (intended primarily for users) and `xpbsmon` (intended for PBS operators and managers). Both are built using the Tool Control Language Toolkit (TCL/tk). The first section below discusses the user GUI, `xpbs`. The following section discusses `xpbsmon`.



```
HOSTS
Server               Max Tot Que Run Hld Wat Trn Ext Status      PEsInUse
jim.mrj.com            0  10   9   0   1   0   0   0 Scheduling      0/60
osprey.mrj.com         0  21   7  10   4   0   0   0 Active      244/248
origin.mrj.com         0   9   8   1   0   0   0   0 Active          8/8

QUEUES  Listed By Host(s):  jim.mrj.com  osprey.mrj.com  origin.mrj.com
Queue         Max Tot Ena Str Que Run Hld Wat Trn Ext Type      Server
pending         0  10 yes yes   9   0   1   0   0   0 Execution jim.mrj.com
challenge       0   0 yes yes   0   0   0   0   0   0 Execution jim.mrj.com
special         8   0 yes yes   0   0   0   0   0   0 Execution jim.mrj.com
pending         0  21 yes yes   7  10   4   0   0   0 Execution osprey.mrj.com
challenge       0   0 yes yes   0   0   0   0   0   0 Execution osprey.mrj.com
submit          0   9 yes yes   8   1   0   0   0   0 Execution origin.mrj.com

JOBS   Listed By Queue(s): ey.mrj.com  challenge@osprey.mrj.com  submit@origin.mrj.com :

Job Id             Name         User      PEs  CputUse   WallUse S Queue
3172.jim.mrj.com   run30_2      sinha      30         0         0 Q pending@jim.mrj
16339.osprey.mrj.  p1.cmd       dhash      48 170:15:5  03:35:02 R pending@osprey.mrj
16340.osprey.mrj.  ...._ml_1.run dorney    16         0         0 Q pending@osprey.mrj
16340.osprey.mrj.  rlv_cl.run   dorney     16         0         0 Q pending@osprey.mrj
16340.osprey.mrj.  mra_bg.run   dorney     16         0         0 Q pending@osprey.mrj
33171.osprey.mrj.  ucav.job     potsdam    28 49:06:51  01:46:52 R pending@osprey.mrj
16340.osprey.mrj.  mra_1.run    dorney     14 14:03:30  01:46:50 R pending@osprey.mrj
33172.osprey.mrj.  run2.3dums   sinha      16         0         0 Q pending@osprey.mrj
16339.osprey.mrj.  p2.cmd       dhash      48 39:04:39  00:50:15 R pending@osprey.mrj
33170.osprey.mrj.  ra180T108    yancheng   16 08:50:19  00:35:59 R pending@osprey.mrj
33173.osprey.mrj.  AITTA        perrell    22 06:16:14  00:19:42 R pending@osprey.mrj

INFO
[10/08/99 19:25:25] "/usr/local/pkg/pbs/lib/xpbs/bin/xpbs_datadump -t 30 jim osprey origin
```

11.19.1 xpbs GUI Configuration

xpbs provides a user-friendly point-and-click interface to the PBS commands. To run `xpbs` as a regular, non-privileged user, type:

```
% setenv DISPLAY your_workstation_name:0
% xpbs
```

To run `xpbs` with the additional purpose of terminating PBS Servers, stopping and starting queues, or running/rerunning jobs, then run:

```
% xpbs -admin
```

Important: Table 7 in the **PBS User Guide** lists all functionality of `xpbs`, and identifies which are available only via the `-admin` option.

Running `xpbs` will initialize the X resource database from various sources in the following order:

1. The RESOURCE_MANAGER property on the root window (updated via `xrdb`) with settings usually defined in the `.Xdefaults` file
2. Preference settings defined by the system administrator in the global `xpbsrc` file
3. User's `~/.xpbsrc` file-- this file defines various X resources like fonts, colors, list of PBS hosts to query, criteria for listing queues and jobs, and various view states.

The system administrator can specify a global resources file to be read by the GUI if a personal `.xpbsrc` file is missing: `/usr/pbs/lib/xpbs/xpbsrc`. Keep in mind that within an Xresources file (Tk only), later entries take precedence. For example, suppose in your `.xpbsrc` file, the following entries appear in order:

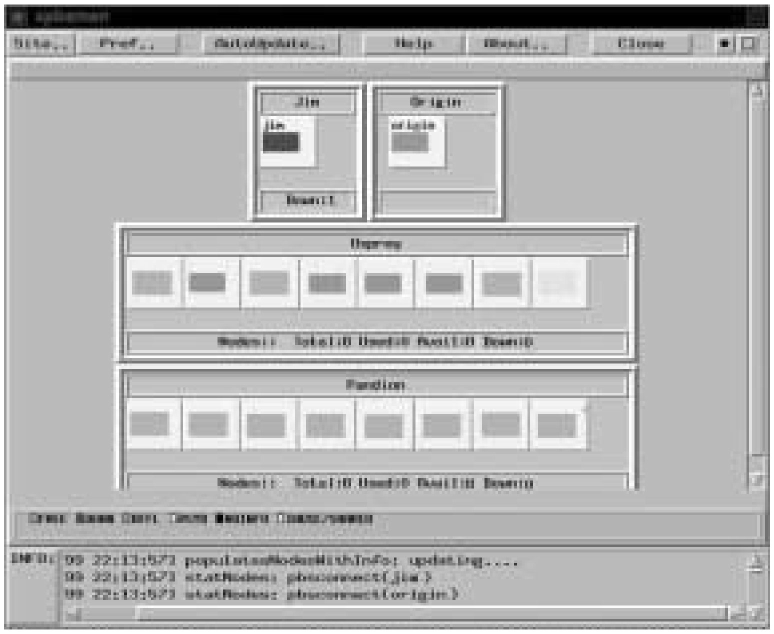
```
xpbsrc*backgroundColor: blue
*backgroundColor: green
```

The later entry "green" will take precedence even though the first one is more precise and longer matching. The things that can be set in the personal preferences file are fonts, colors, and favorite Server host(s) to query.

`xpbs` usage, command correlation, and further customization information is provided in the **PBS User Guide**, Chapter 5, “Using the xpbs GUI”.

11.19.2 xpbsmon GUI Configuration

xpbsmon is the node monitoring GUI for PBS. It is used for graphically displaying information about execution hosts in a PBS environment. Its view of a PBS environment consists of a list of sites where each site runs one or more Servers, and each Server runs jobs on one or more execution hosts (nodes).


Site..
Pref..
AutoUpdate..
Help
About..
Close
Jim
jim
Origin
origin
Nodes:: Total:0 Used:0 Avail:0 Down:0
Nodes:: Total:0 Used:0 Avail:0 Down:0
INFO:

```
09 22:13:57) populateNodesWithInfo: updating....
09 22:13:57) statNodes: pbsconnect(jim)
09 22:13:57) statNodes: pbsconnect(origin)
```

The system administrator needs to define the site's information in a global X resources file, */usr/pbs/lib/*`xpbsmon/xpbsmonrc` which is read by the GUI if a personal `.xpbsmonrc` file is missing. A default `xpbsmonrc` file usually would have been created already during installation, defining (under *sitesInfo resource) a default site name, list of Servers that run on a site, set of nodes (or execution hosts) where jobs on a particular Server run, and the list of queries that are communicated to each node's `pbs_mom.` If node queries have been specified, the host where `xpbsmon` is running must have been given explicit permission by the `pbs_mom` daemon to post queries to it. This is done by including a `$restricted` entry in the MOM's config file.

It is not recommended to manually update the *sitesInfo value in the `xpbsmonrc` file as its syntax is quite cumbersome. The recommended procedure is to bring up `xpbsmon`, click on "Pref.." button, manipulate the widgets in the Sites, Server, and Query Table dialog boxes, then click "Close" button and save the settings to a `.xpbsmonrc` file. Then copy this file over to the */usr/pbs/*`lib/xpbsmon/` directory.

11.20 Using Job Comments

Users tend to want to know what is happening to their job. PBS provides a special job attribute, `comment` which is available to the operator, manager, or the Scheduler program. This attribute can be set to a string to pass information to the job owner. It might be used to display information about why the job is not being run or why a hold was placed on the job. Users are able to see this attribute, when set, by using the `-f and -s` option of the `qstat` command. (For details see “Displaying Job Comments” in the **PBS User Guide**.) Operators and managers may use the `-W` option of the `qalter` command, for example

```
qalter -W comment="some text" job_id
```

11.21 pbsfs (PBS Fairshare) command

The pbsfs command allows the administrator to display or manipulate PBS fairshare usage data. This command should only be used when the scheduler is not running. There are multiple parts to a fairshare node and you can print these data in different formats.The data displayed is:

Data	**Description**
fairshare entity	the entity in the fairshare tree
group	the group ID the node is in (i.e. the node's parent)
cgroup	the group ID of this group
shares	the number of shares the group has
usage	the amount of usage
percentage	the percentage the entity has of the tree. Note that only the leaf nodes sum to 100%. If all of the nodes are summed, the result will be greater then 100%. Only the leaf nodes of the tree are fairshare entities.
usage / perc	The value the scheduler will use to the pick which entity has priority over another. The smaller the number the higher the priority.

Path from root	The path from the root of the tree to the node. This is useful because the scheduler will look down the path to compare two nodes to see which has the higher priority.
resource	The resource for which the scheduler accumulates usage for its fairshare calculations. This defaults to cput (cpu seconds) but can be set in the scheduler's config file.

11.22 PBS Pro on Scyld Beowulf Clusters

Running PBS Pro under Scyld Computing Corporation’s Beowulf operating system is a bit different from normal clusters. Given Scyld’s single system image, there needs to be only a single MOM running within the cluster, rather than on each node. When `pbs_mom` starts on the master node, the ownership of all the compute nodes is changed to 'root'. Each time a job runs, the ownership of the nodes chosen for the job will be changed to the user running the job.

The Scyld kernel allows processes running on the compute nodes to be tracked on the master node so `pbs_mom` can directly set limits and track usage. The `ncpus` resource is reported as the total number of CPUs reported on all the compute nodes configured. The actual number of CPUs available will vary as nodes run jobs, go down or otherwise become unavailable. Since ownership is assigned on a node rather than cpu basis, if you have multi-cpu nodes, there may be unused CPUs if a job asks for a number of nodes that is not an even multiple of the number of CPUs per node. The `physmem` resource is the sum of physical memory on all the compute nodes.

Information on running jobs on Scyld Beowulf clusters is given in section “Running Jobs on Scyld Beowulf Clusters” of the **PBS User Guide**.

Chapter 12
Problem Solving

The following is a list of common problems and recommended solutions. Additional information is always available online at the PBS website, www.pbspro.com. The last section in this chapter gives important information on how to get additional assistance from the PBS Support staff.

12.1 Clients Unable to Contact Server

If a client command (such as `qstat` or `qmgr`) is unable to connect to a Server there are several possibilities to check. If the error return is `15034`, "`No server to connect to`", check (1) that there is indeed a Server running and (2) that the default Server information is set correctly. The client commands will attempt to connect to the Server specified on the command line if given, or if not given, the Server specified by **`SERVER_NAME`** in `/etc/pbs.conf`.

If the error return is `15007`, "`No permission`", check for (2) as above. Also check that the executable `pbs_iff` is located in the search path for the client and that it is setuid root. Additionally, try running `pbs_iff` by typing:

```
pbs_iff -t server_host 15001
```

Where *`server_host`* is the name of the host on which the Server is running and `15001` is the port to which the Server is listening (if started with a different port number, use that number instead of `15001`). Check for an error message and/or a non-zero exit status. If `pbs_iff` exits with no error and a non-zero status, either the Server is not running or was installed with a different encryption system than was `pbs_iff`.

12.2 Nodes Down

The PBS Server determines the state of nodes (`up` or `down`), by communicating with MOM on the node. The state of nodes may be listed by two commands: `qmgr` and `pbsnodes`.

```
% qmgr
Qmgr: list node @active

% pbsnodes -a
Node jupiter
        state = down, state-unknown
        properties = sparc, mine
        ntype = cluster
```

A node in PBS may be marked “`down`” in one of two substates. For example, the state above of node “jupiter” shows that the Server has not had contact with MOM since the Server came up. Check to see if a MOM is running on the node. If there is a MOM and if the MOM was just started, the Server may have attempted to poll her before she was up. The Server should see her during the next polling cycle in 10 minutes. If the node is still marked “`down, state-unknown`” after 10+ minutes, either the node name specified in the Server’s node file does not map to the real network hostname or there is a network problem between the Server’s host and the node.

If the node is listed as

```
% pbsnodes -a
Node jupiter
        state = down
        properties = sparc, mine
        ntype = cluster
```

Then the Server has been able to ping MOM on the node in the past, but she has not responded recently. The Server will send a “ping” PBS message to every free node each ping cycle, 10 minutes. If a node does not acknowledge the ping before the next cycle, the Server will mark the node down.

On an IBM SP, a node may also be marked down if MOM on the node believes that the node is not connected to the high speed switch. When the Server receives an acknowledgment from MOM on the node, the node will again be marked up (free).

12.3 Requeueing a Job "Stuck" on a Down Node

Once a job is "running" it will be in that state until the Server hears otherwise from MOM. However, if the (first) node for a job fails/crashes/hangs, then MOM cannot tell the Server that the job is done. You have two options of how to handle this situation.

If you wish to have PBS simply remove the hung job from the system, use the "`-Wforce`" option to qdel:

```
% qdel -Wforce jobID
```

If instead you want PBS to requeue the job, and have it immediately eligible to run again, use the "`-Wforce`" option to qrerun:

```
% qrerun -Wforce jobID
```

The `-Wforce` option is required as a safe-guard since both actions are extraordinary.

12.4 Non Delivery of Output

If the output of a job cannot be delivered to the user, it is saved in a special directory: `/usr/spool/PBS/undelivered` and mail is sent to the user. The typical causes of non-delivery are:

1. The destination host is not trusted and the user does not have a `.rhosts` file.
2. An improper path was specified.
3. A directory in the specified destination path is not writable.
4. The user's `.cshrc` on the destination host generates output when executed.
5. The path specified by **`PBS_SCP`** in `/etc/pbs.conf` is incorrect.
6. The `/usr/spool/PBS/spool` directory on the execution host does not have the correct permissions. This directory must have mode 1777 (`drwxrwxrwxt`).

These are explained in the "Delivery of Output Files" section of the **PBS User Guide**.

12.5 Job Cannot be Executed

If a user receives a mail message containing a job id and the line "`Job cannot be executed`", the job was aborted by MOM when she tried to place it into execution. The complete reason can be found in one of two places, MOM's log file or the standard error file of the user's job.

If the second line of the message is "`See Administrator for help`", then MOM aborted the job before the job's files were set up. The reason will be noted in MOM's log. Typical reasons are a bad user/group account, checkpoint/restart file (Cray or SGI), or a system error.

If the second line of the message is "`See job standard error file`", then MOM had created the job's file and additional messages were written to standard error. This is typically the result of a bad resource request.

12.6 Handling Jobs on Failed Nodes

If a job is running and the first node assigned to the job goes down, the job can be re-queued (`qrerun`) or deleted (`qdel`). Neither of these actions are performed automatically because (1) it might be the network rather than the node that actually went down and the job is still running correctly, or (2) it might be that the `pbs_mom` on the node went down and the job is still running correctly. In either case, rather than waste the cycles spent so far, the administrator or user can allow the job to continue and when the network or `pbs_mom` is restarted, the work will not have been lost.

However, if the node is truly down, the administrator or user can delete the job or requeue it for execution later, by using the "`-W force`" option to `qdel` and `qrerun`, as shown below.

Important: Note, in either case, the output created by the job before the node went down will be discarded.

```
% qdel -W force 1123.veridian.com
or
% qrerun -W force 1124.veridian.com
```

12.7 Running Jobs with No Active Processes

On very rare occasions, PBS may be in a situation where a job is in the Running state but has no active processes. This should never happen as the death of the job's shell should trigger MOM to notify the Server that the job exited and end of job processing should begin. If this situation is noted, PBS offers a way out. Use the `qsig` command to send `SIGNULL`, signal 0, to the job. (Usage of the `qsig` command is provided in the **PBS User Guide**.) If MOM finds there are no processes then she will force the job into the exiting state.

12.8 Getting Help

If the material in the PBS manuals is unable to help you solve a particular problem, you may need to contact the PBS Support team for assistance. First, be sure to check the Customer Login area of the PBS Pro website, which has a number of ways to assist you in resolving problems with PBS. The two most frequently used are: the Tips & Advice page and the Submit Problem Report page.

The PBS Pro support team can also be reached directly via email and phone (contact information on the inside front cover of this manual).

Important: When contacting PBS Pro Support, please provide as much of the following information as possible:

PBS SiteID
Output of the following commands:

```
qstat -Bf
qstat -Qf
pbsnodes -a
```

If the question pertains to a certain type of job, include:

```
qstat -f job_id
```

If the question is about scheduling, also send your:
`(PBS_HOME)/sched_priv/sched_config` file.

To expand, renew, or change your PBS support contract, contact our Sales Department. (See contact information on the inside front cover of this manual.)

Chapter 13
Customizing PBS

This chapter addresses several ways that PBS can be customized for your site. While having the source code is the first step, there are specific actions other than modifying the code you can take.

13.1 Shell Invocation

When PBS starts a job, it invokes the user's login shell (unless the user submitted the job with the `-S` option). PBS passes the job script which is a shell script to the login process in one of two ways depending on how PBS was installed.

1. Name of Script on Standard Input
The default method (PBS built with --enable-shell-pipe) is to pass the name of the job script to the shell program. This is equivalent to typing the script name as a command to an interactive shell. Since this is the only line passed to the script, standard input will be empty to any commands. This approach offers both advantages and disadvantages:

+ Any command which reads from standard input without redirection will get an EOF.

+ The shell syntax can vary from script to script, it does not have to match the syntax for the user's login shell. The first line of the script, even before any #PBS directives, should be

`#!/`*`shell`* where *`shell`* is the full path to the shell of choice, `/bin/sh, /bin/csh`, ...

The login shell will interpret the `#!` line and invoke that shell to process the script.

- An extra shell process is run to process the job script.
- If the script does not include a `#!` line as the first line, the wrong shell may attempt to interpret the script producing syntax errors.
- If a non-standard shell is used via the -S option, it will not receive the script, but its name, on its standard input.

2. Script as Standard Input
The alternative method for PBS (built with `--disable-shell-invoke`), is to open the script file as standard input for the shell. This is equivalent to typing:

```
% /path/shell < script
```

This also offers advantages and disadvantages:

+ The user's script will always be directly processed by the user's login shell.

+ If the user specifies a non-standard shell (any old program) with the -S option, the script can be read by that program as its input.

- If a command within the job script reads from standard input, it may read lines from the script depending on how far ahead the shell has buffered its input. Any command line so read will not be executed by the shell. A command that reads from standard input with out explicit redirection is generally unwise in a batch job.

The choice of shell invocation methods is left to the site. It is recommended that all PBS execution servers (pbs_mom) within that site be built to use the same shell invocation method.

13.2 TCL/tk Interface to PBS API

The PBS Pro software includes a published and documented Application Programming Interface (API). Its use is discussed later in this chapter, and in detail in the **PBS External Reference Specification** (**ERS**). There is also a TCL/tk interface to PBS. Wrapped versions of many of the API calls are compiled into a special version of the TCL shell, called `pbs_tclsh`. (A special version of the tk window shell is also provided, called `pbs_wish`.). These tools were originally developed as part of a TCL scheduler interface, and as such are documented in the `pbs_sched_tcl(8B)` manual page.

13.3 Additional Build Options

Two header files within the subdirectory `src/include` provide additional configuration control over the Server and MOM. The modification of any symbols in the two files should not be undertaken lightly.

13.3.1 pbs_ifl.h

This header file contains structures, symbols and constants used by the API, libpbs.a, and the various commands as well as the daemons. Very little here should ever be changed. Possible exceptions are the following symbols. They must be consistent between all batch systems which might interconnect.

PBS_MAXHOSTNAME
: Defines the length of the maximum possible host name. This should be set at least as large as MAXHOSTNAME which may be defined in `sys/params.h.`

PBS_MAXUSER
: Defines the maximum possible length of a user login name.

PBS_MAXGRPN
: Defines the maximum possible length of a maximum possible group name.

PBS_MAXQUEUENAME
: Defines the maximum possible length of a maximum possible PBS queue name.

PBS_USE_IFF

If this symbol is set to zero (0), before the library and commands are built, the API routine `pbs_connect`() will not attempt to invoke the program `pbs_iff` to generate a secure credential to authenticate the user. Instead, a clear text credential will be generated. This credential is completely subject to forgery and is useful only for debugging the PBS system. You are strongly advised against using a clear text credential.

PBS_BATCH_SERVICE_PORT

Defines the port number at which the Server listens.

PBS_MOM_SERVICE_PORT

Defines the port number at which MOM, the execution mini-server, listens.

PBS_SCHEDULER_SERVICE_PORT

Defines the port number at which the Scheduler listens.

13.3.2 server_limits.h

This header file contains symbol definitions used by the Server and by MOM. Only those that *might* be changed are listed here. These should be changed with care. It is strongly recommended that no other symbols in `server_limits.h` be changed. If server_limits.h is to be changed, it may be copied into the include directory of the *target* (build) tree and modified before compiling.

NO_SPOOL_OUTPUT

If defined, directs MOM to not use a spool directory for the job output, but to place it in the user's home directory while the job is running. This allows a site to invoke quota control over the output of running batch jobs.

PBS_BATCH_SERVICE_NAME

This is the service name used by the Server to determine to which port number it should listen. It is set to pbs in quotes as it is a character string. Should you wish to assign PBS a service port in `/etc/services` change this string to the service name assigned. You should also update PBS_SCHEDULER_SERVICE_NAME as required.

PBS_DEFAULT_ADMIN

Defined to the name of the default administrator, typically "root". Generally only changed to simplify debugging.

PBS_DEFAULT_MAIL
: Set to user name from which mail will be sent by PBS. The default is "adm". This is overridden if the Server attribute `mail_from` is set.

PBS_JOBBASE
: The length of the job id string used as the basename for job associated files stored in the spool directory. It is set to 11, which is 14 minus the 3 characters of the suffixes like `.JB` and `.OU`. Fourteen is the guaranteed length for a file name under POSIX.

PBS_MAX_HOPCOUNT
: Used to limit the number of hops taken when being routed from queue to queue. It is mainly to detect loops.

PBS_NET_MAX_CONNECTIONS
: The maximum number of open file descriptors and sockets supported by the Server.

PBS_NET_RETRY_LIMIT
: The limit on retrying requests to remote Servers.

PBS_NET_RETRY_TIME
: The time between network routing retries to remote queues and for requests between the Server and MOM.

PBS_RESTAT_JOB
: To refrain from over burdening any given MOM, the Server will wait this amount of time (default 30 seconds) between asking her for updates on running jobs. In other words, if a user asks for status of a running job more often than this value, the prior data will be returned.

PBS_ROOT_ALWAYS_ADMIN
: If defined (set to 1), “root” is an administrator of the batch system even if not listed in the `managers` attribute.

PBS_SCHEDULE_CYCLE
: The default value for the elapsed time between scheduling cycles with no change in jobs queued. This is the initial value used by the Server, but it can be changed via `qmgr(`1B).

13.4 Site Modifiable Source Files

It is safe to skip this section until you have played with PBS for a while and want to start tinkering.

Certain functions of PBS appear to be likely targets of widespread modification by sites for a number of reasons. When identified, the developers of PBS have attempted to improve the easy of modification in these areas by the inclusion of special *site specific modification routines* The distributed default version of these files build a private library, libsite.a, which is included in the linking phase for the Server and for MOM.

13.4.1 Server Modifiable Files

`site_allow_u.c`	The routine in this file, `site_allow_u`() provides an additional point at which a user can be denied access to the batch system (Server). It may be used instead of or in addition to the Server `acl_user` list.
`site_alt_rte.c`	The function `site_alt_router`() allows a site to add decision capabilities to job routing. This function is called on a per-queue basis if the queue attribute `alt_router` is true. As provided, `site_alt_router`() just invokes the default router, `default_router`().
`site_check_u.c`	There are two routines in this file. The routine `site_check_user_map`() provides the service of authenticating that the job owner is privileged to run the job under the user name specified or selected for execution on the Server system. The routine `site_acl_check`() provides the site with the ability to restrict entry into a queue in ways not otherwise covered. For example, you may wish to check a bank account to see if the user has the funds to run a job in the specific queue.
`site_map_usr.c`	For sites without a common user name/uid space, this function, `site_map_user`() provides a place to add a user name mapping function. The mapping occurs at two times. First to determine if a user making a request against a job is the job owner, see “User Authorization”. Second, to map the submitting user (job owner) to an execution uid on the local machine.

site_*_attr_*.h

These files provide a site with the ability to add local attributes to the Server, queues, and jobs. The files are installed into the target tree "include" subdirectory during the first make. As delivered, they contain only comments. If a site wishes to add attributes, these files can be *carefully* modified.

The files are in three groups, by Server, queue, and job. In each group are site_*_attr_def.h files which are used to defined the name and support functions for the new attribute or attributes, and site_*_attr_enum.h files which insert a enumerated label into the set for the corresponding parent object. For Server, queue, node attributes, there is also an additional file that defines if the qmgr(1) command will include the new attribute in the set "printed" with the print server, print queue, or print nodes sub-commands.

site_resc_attr_def.h

This file allows a site to add local resources. It is included into the Server's resc_def_all.c file at compile time.

You should note that just adding attributes will have no effect on how PBS processes jobs. The main usage for new attributes would be in providing new Scheduler controls and/or information. The scheduling algorithm will have to be modified to use the new attributes. If you need MOM to do something different with a job, you will still need "to get down and dirty" with her source code.

13.4.2 MOM Modifiable Files

site_mom_chu.c

If a Server is sending jobs to more than one MOM, additional checking for execution privilege may be required at MOM's level. It can be added in this function site_mom_chkuser().

site_mom_ckp.c

Provide post-checkpoint, site_mom_postchk() and pre-restart site_mom_prerst() "user exits" for the Cray and SGI systems.

site_mom_jset.c

The function site_job_setup() allows a site to perform specific actions once the job session has been created and before the job runs.

13.5 Implementing a Custom Scheduler

PBS provides a separate process to determine which jobs should be placed into execution. This is a flexible mechanism by which you may implement a very wide variety of policies. The Scheduler uses the standard PBS API to communicate with the Server and an additional API to communicate with the PBS resource monitor, `pbs_mom`. Should the provided Schedulers be insufficient to meet your site's needs, it is possible to implement a replacement Scheduler using the provided APIs which will enforce the desired policies.

13.5.1 Scheduling Theory

The first generation UNIX batch system, NQS, and many of the other workload management systems use various queue-based controls to limit or schedule jobs. Queues would be turned on and off to control job ordering over time or have a limit of the number of running jobs in the queue. While PBS supports multiple queues and the queues have some of the "job scheduling" attributes used by other batch systems, the PBS Server does not by itself run jobs or enforce any of the restrictions implied by these queue attributes. In fact, the Server will happily run a *held* job that resides in a *stopped* queue with a zero limit on running jobs, if it is directed to do so. The direction may come from the operator, administrator, or the Scheduler. In fact, the Scheduler is nothing more than a client with administrator privilege.

If you chose to implement your site scheduling policy using a multiple-queue queue-based scheme, you may do so. The Server and queue attributes used to control job scheduling may be adjusted by a client with privilege, such as `qmgr`(8B), or by a program of your own creation. However, the controls are actually used by in the Scheduler, not the Server. The Scheduler must check the status of the Server and queues, as well as the jobs, determining the setting of the Server and queue controls. It then must use the settings of those controls in its decision making. Another possibility is the "whole pool" approach, wherein all jobs are in a single pool (single queue). The Scheduler evaluates each job on its merits and decides which, if any, to run. The policy can easily include factors such as time of day, system load, size of job, etc. Ordering of jobs in the queue need not be considered. The PBS team believes that this approach is superior for two reasons:

1. Users are not tempted to lie about their requirements in order to "game" the queue policy.

2. The scheduling can be performed against the complete set of current jobs resulting in better fits against the available

resources.

13.5.2 Scheduler – Server Interaction

In developing a scheduling policy, it may be important to understand when and how the Server and the Scheduler interact. The Server always initiates the scheduling cycle. When scheduling is active within the Server, the Server opens a connection to the Scheduler and sends a command indicating the reason for the scheduling cycle. The reasons or events that trigger a cycle are defined in `include/sched_cmd.`h. A description of each follows.

SCH_SCHEDULE_NEW
: A job newly becomes eligible to execute. The job may be a new job in an execution queue, or a job in an execution queue that just changed state from held or waiting to queued.

SCH_SCHEDULE_TERM
: An executing job terminates.

SCH_SCHEDULE_TIME
: The time interval since the prior cycle specified by the Server attribute `schedule_iteration` is reached.

SCH_SCHEDULE_CMD
: The Server attribute `scheduling` is set or reset to true. If set true, even if it's previous value was true, the Scheduler will be cycled. This provides the administrator/operator a means to force a scheduling cycle.

SCH_SCHEDULE_RECYC
: If the Scheduler was cycled and it requested one and only one job to be run, then the Scheduler will be recycled by the Server. This event is a bit abstruse. It exists to "simplify" a Scheduler. The Scheduler only need worry about choosing the one best job per cycle. If other jobs can also be run, it will get another chance to pick the next job. Should a Scheduler run none or more than one job in a cycle it is clear that it need not be recalled until conditions change and one of the other events trigger the next cycle.

SCH_SCHEDULE_FIRST
: If the Server recently recovered, the first scheduling cycle,

resulting from any of the above, will be indicated uniquely.

Once the Server has contacted the Scheduler and sent the reason for the contact, the Scheduler then becomes a privileged client of the Server. As such, it may command the Server to perform any action allowed to a Manager.

When the Scheduler has completed all activities it wishes to perform in this cycle, it will close the connection to the Server. While a connection is open, the Server will not attempt to open a new connection.

Note that the Server contacts the Scheduler to begin a scheduling cycle only if scheduling is active in the Server. This is controlled by the value of the Server attribute `scheduling`. If set true, scheduling is active and `qstat -B` will show the Server status as Active. If `scheduling` is set false, then the Server will not contact the Scheduler and the Server's status is shown as Idle. When started, the Server will recover the value for `scheduling` as it was set when the Server shut down. The value may be changed in two ways: the -a option on the pbs_server command line, or by setting `scheduling` to `true` or `false` via `qmgr`.

One point should be clarified about job ordering: Queues "are" and "are not" FIFOs. What is meant is that while jobs are ordered first in – first out in the Server and in each queue. That fact does **not** imply that running them in that order is mandated, required, or even desirable. That is a decision left completely up to site policy and implementation. The Server will maintain the order across restarts solely as a aid to sites that wish to use a FIFO ordering in some fashion.

13.5.3 Creating a New Scheduler

PBS provides two different interfaces for creating custom scheduler modules: the C programming language, and the Tool Command Language (TCL). This sections gives a high-level overview of each. For detailed information, consult the PBS External Reference Specification.

13.5.4 Tcl-Based Scheduling

The provided Tcl based Scheduler framework uses the basic Tcl interpreter with some extra commands for communicating with the PBS Server and Resource Monitor. The scheduling policy is defined by a script written in Tcl. A number of sample scripts are provided in the source directory `src/scheduler.tcl/sample_scripts`. The Tcl based Scheduler works, very generally, as follows:

1. On start up, the Scheduler reads the initialization script (if specified with the -i option) and executes it. Then, the body script is read into memory. This is the file that will be executed each time a “schedule” command is received from the Server. It then waits for a “schedule” command from the Server.

2. When a schedule command is received, the body script is executed. No special processing is done for the script except to provide a connection to the Server. A typical script will need to retrieve information for candidate jobs to run from the Server using `pbsselstat` or `pbsstatjob`. Other information from the Resource Monitor(s) will need to be retrieved by opening connections with `openrm`(3B) and submitting queries with `addreq`(3B) and getting the results with `getreq`(3B). The Resource Monitor connections must be closed explicitly with `closerm`(3B) or the Scheduler will eventually run out of file descriptors. When a decision is made to run a job, a call to `pbsrunjob`(3B) must be made.

3. When the script evaluation is complete, the Scheduler will close the TCP/IP connection to the Server.

13.5.5 Tcl-Based Scheduling Advice

The Scheduler does not restart the Tcl interpreter for each cycle. This gives the ability to carry information from one cycle to the next. It also can cause problems if variables are not initialized or "unset" at the beginning of the script when they are not expected to contain any information later on.

System load average is frequently used by a script. This number is obtained from the system kernel by pbs_mom. Most systems smooth the load average number over a time period. If one scheduling cycle runs one or more jobs and the next scheduling cycle occurs quickly, the impact of the newly run jobs will likely not be reflected in the load average. This can cause the load average to shoot way up especially when first starting the batch system. Also when jobs terminate, the delay in lowering the load average may delay the scheduling of additional jobs. The Scheduler redirects the output from “stdout” and “stderr” to a file. This makes it easy to generate debug output to check what your script is doing. It is advisable to use this feature heavily until you are fairly sure that your script is working well.

13.5.6 Implementing a Tcl Scheduler

The best advice is study the examples found in src/scheduler.tcl/sample_scripts. Then once you have modified or written a Scheduler body script and optionally an initialization script, place them in the directory `/usr/spool/PBS/sched_priv` and invoke the Scheduler by typing:

```
pbs_sched [-b script] [-i init_script]
```

See the `pbs_sched_tcl`(8B) man page for more information.

13.5.7 C-Based Scheduling

The C based Scheduler is similar in structure and operation to the Tcl Scheduler except that C functions are used rather than Tcl scripts:

1. On start up, the Scheduler calls `schedinit(argc, argv)` one time only to initialize whatever is required to be initialized.
2. When a schedule command is received, the function `schedule`*(*`cmd`, `connector`*)* is invoked. All scheduling activities occur within that function. The values of cmd are discussed in "Scheduler – Server Interaction" on page 195.
3. Upon return to the main loop, the connection to the Server is closed.

Several working Scheduler code examples are provided in the samples subdirectory. The following sections discuss certain of the sample schedulers including the default Scheduler Standard. The sources for the samples are found in `src/scheduler.cc/samples` under the scheduler type name, for example `src/scheduler.cc/samples/standard`.

13.5.8 Scheduling and File Staging

A decision must be made about when to begin to stage-in files for a job. The files must be available before the job executes. The amount of time that will be required to copy the files is unknown to PBS, that being a function of file size and network speed. If file in-staging is not started until the job has been selected to run when the other required resources are available, either those resources are "wasted" while the stage-in occurs, or another job is started which takes the resources away from the first job, and might prevent it from running. If the files are staged in well before the job is otherwise ready to run, the files may take up valuable disk space need by running jobs.

PBS provides two ways that file in-staging can be initiated for a job. If a run request is received for a job with a requirement to stage in files, the staging operation is begun and when completed, the job is run. Or, a specific stage-in request may be received for a job, see `pbs_stagein`(3B), in which case the files are staged in but the job is not run. When the job is run, it begins execution immediately because the files are already there.

In either case, if the files could not be staged-in for any reason, the job is placed into a wait state with a “execute at” time **PBS_STAGEFAIL_WAIT** 30 minutes in the future. A mail message is sent to the job owner requesting that s/he look into the problem. The reason the job is changed into wait state is to prevent the Scheduler from constantly retrying the same job which likely would keep on failing.

Figure 5.0 in Appendix B of the **PBS ERS** shows the (sub)state changes for a job involving file in staging. The Scheduler may note the substate of the job and chose to perform pre-staging via the `pbs_stagein()` call. The substate will also indicate completeness or failure of the operation. The Scheduler developer should carefully chose a stage-in approach based on factors such as the likely source of the files, network speed, and disk capacity.

Appendix A: Error Codes

The following table lists all the PBS error codes, their textual names, and a description of each.

Error Name	Error Code	Description
PBSE_NONE	0	No error
PBSE_UNKJOBID	15001	Unknown Job Identifier
PBSE_NOATTR	15002	Undefined Attribute
PBSE_ATTRRO	15003	Attempt to set READ ONLY attribute
PBSE_IVALREQ	15004	Invalid request
PBSE_UNKREQ	15005	Unknown batch request
PBSE_TOOMANY	15006	Too many submit retries
PBSE_PERM	15007	No permission
PBSE_BADHOST	15008	Access from host not allowed
PBSE_JOBEXIST	15009	Job already exists
PBSE_SYSTEM	15010	System error occurred
PBSE_INTERNAL	15011	Internal Server error occurred

Error Name	Error Code	Description
PBSE_REGROUTE	15012	Parent job of dependent in route queue
PBSE_UNKSIG	15013	Unknown signal name
PBSE_BADATVAL	15014	Bad attribute value
PBSE_MODATRRUN	15015	Cannot modify attrib in run state
PBSE_BADSTATE	15016	Request invalid for job state
PBSE_UNKQUE	15018	Unknown queue name
PBSE_BADCRED	15019	Invalid Credential in request
PBSE_EXPIRED	15020	Expired Credential in request
PBSE_QUNOENB	15021	Queue not enabled
PBSE_QACESS	15022	No access permission for queue
PBSE_BADUSER	15023	Bad user - no password entry
PBSE_HOPCOUNT	15024	Max hop count exceeded
PBSE_QUEEXIST	15025	Queue already exists
PBSE_ATTRTYPE	15026	Incompatible queue attribute type
PBSE_QUEBUSY	15027	Queue Busy (not empty)
PBSE_QUENBIG	15028	Queue name too long
PBSE_NOSUP	15029	Feature/function not supported
PBSE_QUENOEN	15030	Cannot enable queue, needs add def
PBSE_PROTOCOL	15031	Protocol (ASN.1) error
PBSE_BADATLST	15032	Bad attribute list structure
PBSE_NOCONNECTS	15033	No free connections
PBSE_NOSERVER	15034	No Server to connect to
PBSE_UNKRESC	15035	Unknown resource
PBSE_EXCQRESC	15036	Job exceeds Queue resource limits
PBSE_QUENODFLT	15037	No Default Queue Defined

Error Name	Error Code	Description
PBSE_NORERUN	15038	Job Not Rerunnable
PBSE_ROUTEREJ	15039	Route rejected by all destinations
PBSE_ROUTEEXPD	15040	Time in Route Queue Expired
PBSE_MOMREJECT	15041	Request to MOM failed
PBSE_BADSCRIPT	15042	(qsub) Cannot access script file
PBSE_STAGEIN	15043	Stage In of files failed
PBSE_RESCUNAV	15044	Resources temporarily unavailable
PBSE_BADGRP	15045	Bad Group specified
PBSE_MAXQUED	15046	Max number of jobs in queue
PBSE_CKPBSY	15047	Checkpoint Busy, may be retries
PBSE_EXLIMIT	15048	Limit exceeds allowable
PBSE_BADACCT	15049	Bad Account attribute value
PBSE_ALRDYEXIT	15050	Job already in exit state
PBSE_NOCOPYFILE	15051	Job files not copied
PBSE_CLEANEDOUT	15052	Unknown job id after clean init
PBSE_NOSYNCMSTR	15053	No Master in Sync Set
PBSE_BADDEPEND	15054	Invalid dependency
PBSE_DUPLIST	15055	Duplicate entry in List
PBSE_DISPROTO	15056	Bad DIS based Request Protocol
PBSE_EXECTHERE	15057	Cannot execute there
PBSE_SISREJECT	15058	Sister rejected
PBSE_SISCOMM	15059	Sister could not communicate
PBSE_SVRDOWN	15060	Request rejected -server shutting down
PBSE_CKPSHORT	15061	Not all tasks could checkpoint

Error Name	Error Code	Description
PBSE_UNKNODE	15062	Named node is not in the list
PBSE_UNKNODEATR	15063	Node-attribute not recognized
PBSE_NONODES	15064	Server has no node list
PBSE_NODENBIG	15065	Node name is too big
PBSE_NODEEXIST	15066	Node name already exists
PBSE_BADNDATVAL	15067	Bad node-attribute value
PBSE_MUTUALEX	15068	State values are mutually exclusive
PBSE_GMODERR	15069	Error(s) during global mod of nodes
PBSE_NORELYMOM	15070	Could not contact MOM
PBSE_NOTSNODE	15071	No time-shared nodes
PBSE_RESV_NO_ WALLTIME	15075	job reserv lacking walltime
PBSE_JOBNOTRESV	15076	not a reservation job
PBSE_TOOLATE	15077	too late for job reservation
PBSE_IRESVE	15078	internal reservation-system error
PBSE_UNKRESVTYPE	15079	unknown reservation type
PBSE_RESVEXIST	15080	reservation already exists
PBSE_resvFail	15081	reservation failed
PBSE_genBatchReq	15082	batch request generation failed
PBSE_mgrBatchReq	15083	qmgr batch request failed
PBSE_UNKRESVID	15084	unknown reservation ID
PBSE_delProgress	15085	delete already in progress
PBSE_BADTSPEC	15086	bad time specification(s)
PBSE_RESVMSG	15087	so reply_text can return a msg
PBSE_NOTRESV	15088	not a reservation

Error Name	Error Code	Description
PBSE_BADNODESPEC	15089	node(s) specification error
PBSE_LICENSECPU	15090	Licensed CPUs exceeded
PBSE_LICENSEINV	15091	License is invalid
PBSE_RESVAUTH_H	15092	Host not authorized to make AR
PBSE_RESVAUTH_G	15093	Group not authorized to make AR
PBSE_RESVAUTH_U	15094	User not authorized to make AR
PBSE_R_UID	15095	Bad effective UID for reservation
PBSE_R_GID	15096	Bad effective GID for reservation
PBSE_IBMSPSWITCH	15097	IBM SP Switch error
PBSE_LICENSEUNAV	15098	Floating License unavailable
PBSE_RMUNKNOWN	15201	resource unknown
PBSE_RMBADPARAM	15202	parameter could not be used
PBSE_RMNOPARAM	15203	a parameter needed did not exist
PBSE_RMEXIST	15204	something specified didn't exist
PBSE_RMSYSTEM	15205	a system error occurred
PBSE_RMPART	15206	only part of reservation made
RM_ERR_UNKNOWN	PBSE_RMUNKNOWN	
RM_ERR_BADPARAM	PBSE_RMBADPARAM	
RM_ERR_NOPARAM	PBSE_RMNOPARAM	
RM_ERR_EXIST	PBSE_RMEXIST	
RM_ERR_SYSTEM	PBSE_RMSYSTEM	
Resource monitor specific error codes		
PBSE_RMUNKNOWN	15201	Resource unknown
PBSE_RMBADPARAM	15202	Parameter could not be used

Error Name	Error Code	Description
PBSE_RMNOPARAM	15203	A parameter needed did not exist
PBSE_RMEXIST	15204	Something specified didn't exist
PBSE_RMSYSTEM	15205	A system error occurred
PBSE_RMPART	15206	Only part of reservation made

Appendix B: Request Codes

When reading the PBS event logfiles, you may see messages of the form “Type 19 request received from PBS_Server...”. These “type codes” correspond to different PBS batch requests. The following table lists all the PBS type codes and the corresponding request of each.

0	PBS_BATCH_Connect
1	PBS_BATCH_QueueJob
3	PBS_BATCH_jobscript
4	PBS_BATCH_RdytoCommit
5	PBS_BATCH_Commit
6	PBS_BATCH_DeleteJob
7	PBS_BATCH_HoldJob
8	PBS_BATCH_LocateJob
9	PBS_BATCH_Manager
10	PBS_BATCH_MessJob
11	PBS_BATCH_ModifyJob
12	PBS_BATCH_MoveJob

13	PBS_BATCH_ReleaseJob
14	PBS_BATCH_Rerun
15	PBS_BATCH_RunJob
16	PBS_BATCH_SelectJobs
17	PBS_BATCH_Shutdown
18	PBS_BATCH_SignalJob
19	PBS_BATCH_StatusJob
20	PBS_BATCH_StatusQue
21	PBS_BATCH_StatusSvr
22	PBS_BATCH_TrackJob
23	PBS_BATCH_AsyrunJob
24	PBS_BATCH_Rescq
25	PBS_BATCH_ReserveResc
26	PBS_BATCH_ReleaseResc
48	PBS_BATCH_StageIn
49	PBS_BATCH_AuthenUser
50	PBS_BATCH_OrderJob
51	PBS_BATCH_SelStat
52	PBS_BATCH_RegistDep
54	PBS_BATCH_CopyFiles
55	PBS_BATCH_DelFiles
56	PBS_BATCH_JobObit
57	PBS_BATCH_MvJobFile
58	PBS_BATCH_StatusNode
59	PBS_BATCH_Disconnect
62	PBS_BATCH_JobCred

63	PBS_BATCH_CopyFiles_Cred
64	PBS_BATCH_DelFiles_Cred
65	PBS_BATCH_GSS_Context
70	PBS_BATCH_SubmitResv
71	PBS_BATCH_StatusResv
72	PBS_BATCH_DeleteResv

Index

A
account 10
accounting log 163, 165
administration 143
administrator 10
aerospace 2
AIX 45
alternate or test version 57
API 10, 189
attribute
  comment 178
  defined 10
  job
    reference 86
  node
    comment 82
    license 82
    max_group_run 82
    max_running 82
    max_user_run 82
    ntype 81
    pcpus 82
    properties 81
    queue 82
    reservations 82
    resources_assigned 81
    resources_available 81
    state 81
  queue
    acl_group_enable 73
    acl_groups 73
    acl_host_enable 73
    acl_hosts 73
    acl_user_enable 73
    acl_users 74
    alt_router 76
    checkpoint_min 75
    enabled 74
    from_route_only 74
    hasnodes 77
    kill_delay 75
    max_group_run 76
    max_queuable 74
    max_running 74
    max_user_res 74
    max_user_run 76
    priority 74
    queue_type 75
    resources_assigned 77
    resources_available 76
    resources_default 75
    resources_max 75
    resources_min 75
    route_destinations 76

route_held_jobs 76
route_lifetime 76
route_retry_time 76
route_waiting_jobs 77
started 75
state_count 77
total_jobs 77
server
acl_host_enable 67
acl_hosts 67
acl_roots 67
acl_user_enable 67
acl_users 67
comment 67
default_node 67
default_queue 68
flatuid 68
licenses 71
log_events 68
mail_uid 68
managers 68
max_group_run 68
max_running 68
max_user_res 69
max_user_run 69
node_pack 69
operators 69
PBS_version 71
query_other_jobs 70
resources_assigned 71
resources_available 70
resources_cost 70
resources_default 70
resources_max 70
scheduler_iteration 70
scheduling 70
server_name 71
server_state 71
state_count 71
system_cost 71
total_jobs 72
autoconf 30
B
batch or batch processing 10
building PBS 31
C
CD-ROM 19
checkpoint_path 94
checkpointing
during shutdown 153
multi-node jobs 153
prior to SGI IRIX upgrade 154
cluster 9
commands 6
complex 10
configuration
MOM 93
Scheduler 113
Server 91
xpbs 174
xpbsmon 176
configure 30
cpusets 24, 48, 104, 106, 124
Cray 49
T3E 50
credential 156
customization
overview 187
site_map_user 156
cycle-stealing 108
D
Dedicated Time 127
default configuration 66
Defining Holidays 132
destination 10
identifier 11
Dynamic Consumable Resources 133
E
enforcement
ncpus 102
epilogue 160, 161
error codes 201

ERS 189
event log 164
examples 137
exclusive VP 9
External Reference Specificaion 189
External Reference Specification xi
F
Failed Nodes 184
fairshare 129
file staging 11
flat user namespace 156
flatuid 156
floating software licenses 108
G
Globus 15, 30
    Toolkit 30
Globus MOM 110
GNU 30
Grid Forum 4
group 11
    authorization 157
    ID (GID) 11
H
high-throughput 142
hold 11
holidays 132
HP-UX 45
I
IBM 45
    SP 46
idle workstations 108
Information Power Grid 4
installation
    overview 19
    source code 29
    Unix/Linux 21
    Windows 2000 24
Internal Security 155
Irix 48
J
Job
    comment 178
    using comments 178
Job Executor (MOM) 7
job or batch job 11
Job Scheduler 7
Job Server 7
job_sort_key 119, 120, 124
K
key 120
L
license
    floating 26
    keys 25
    manager 25
    multiple keys 27
    pbs_setlicense 27
Linux clusters 3
load balance 9
logfiles 163, 164
M
maintenance 143
manager 11
Manager privilege 158
message passing 142
meta-computing 4
MOM 7
    configuration 93
    dynamic resources 133
    parameters
        alloc_nodes_greedy 94
        checkpoint_upgrade 94
        clienthost 94
        cpuset_create_flags 95
        cpuset_destroy_delay 96
        cpuset_small_mem 96
        cpuset_small_ncpus 96
        cputmult 95
        enforce 96

ideal_load 97
kdb_idle 97
logevent 97
max_load 97
max_shared_nodes 98
prologalarm 98
restricted 98
usecp 99
wallmult 100
starting 144
MOM configuration 93
MOM Globus 110
starting 144
monitoring 5
MPI 159
MPICH 159
mpirun 159
N
name 129
NASA ix, 3
Ames Research Center 3
Information Power Grid 4
Metacenter 4
network
addresses 61
ports 61
services 61
Network Queueing System (NQS) 3
New Features in PBS Pro 5.2 13
New Scheduler Features 115
node 8
attribute 9
cluster 8
property 9
timeshared 8
node_sort_key 120
Nodes 77
nodes
and virtual processors 8
Nodes File 78
NQS 3, 5
O
operator 11
privilege 158
owner 11
P
package installer 21
parallel jobs 142
parent_group 129
pbs_iff 156, 181
pbs_mom 7, 17, 93
pbs_mom_globus 150
pbs_sched 7, 16
pbs_sched_tcl 189
pbs_server 7, 16
pbs_tclsh 189
pbs_wish 189
pbsnodes 173
pbspoe 47
poe 47
Portable Batch System 10
POSIX 30
defined 11
make 30
standard 3
predictability 142
Preemptive scheduling 126
preemptive scheduling 126
privilege 158
manager 158
Problem Report 185
prologue 160, 161, 162
PSSP 46
Q
qalter 178
qdel 184
qdisable 154
qenable 154
qmgr 52, 54, 62, 181
qmove 54
qrerun 184
qrun 52

qstart 154
qstat 181
qstop 154
qterm 56, 151
queue 9
  attributes 72
queuing 5
R
Redhat Linux 44
rerunable 12
resource
  cput 86
  defining new 87
  listing 86
  mem 86
  ncpus 86
  nodes 86
  walltime 86
Resources 86
Root Owned Jobs 159
S
Scheduler
  configuration 113
  dynamic resources 133
  parameters
    assign_ssinodes 117
    backfill 118
    backfill_prime 118
    by_queue 118
    cpus_per_ssinode 118
    dedicated_prefix 118
    fair_share 118
    fairshare_entity 118
    fairshare_usage_res 118
    half_life 119
    help_starving_jobs 119
    load_balancing 120
    load_balancing_rr 120
    log_filter 120
    max_starve 120
    mem_per_ssinode 120
    nonprimetime_prefix 121
    preempt_checkpoint 121
    preempt_fairshare 121
    preempt_order 121
    preempt_prio 122
    preempt_queue_prio 122
    preempt_requeue 122
    preempt_starving 123
    preempt_suspend 123
    preemptive_sched 121
    prime_spill 123
    primetime_prefix 121
    resources 123
    round_robin 123
    smp_cluster_dist 123
    sort_by 119
    strict_fifo 124
    sync_time 124
    unknown_shares 124
  starting 144
scheduling 5
scheduling policies 113
Scyld Beowulf 15
security 143, 155
selective routing 89
Server
  attributes 66
  recording configuration 91
  starting 144
server_dyn_res 123
setlicense 26
shares 129
site-customized schedulers 107
software upgrade 53
Solaris 44
sort_by 124
source code 29
  build procedure 32
  distribution 29

Space-sharing Cluster 142
stage in 12
stage out 12
Standard Scheduler 113
Starting
  Globus MOM 150
  MOM 145
  Scheduler 149
  Server 147
Starting PBS 144
Stopping PBS 151
Support team 185
suspendsig 99
Syslog 171
T
task 12
Tcl 30
TCL/tk 189
tcl/tk 30
temporarily-shared VP 9
Tips & Advice 185
tracejob 172
Tru64 44
Tunable Parameters 117
Type codes 165
U
Unicos 49
Unicos MLS 49
unique_id 129
upgrade
  migration 53
  overlay 53
user 12
  ID (UID) 12
  privilege 158
V
virtual processor (VP) 12
W
Windows 2000 24
workload management 3, 5
X
xpbs 30, 174, 175
  configuration 174
xpbsmon 30, 107, 174, 176

PBS Product Sales & Support
2672 Bayshore Parkway, Suite 810
Mountain View, CA 94043-1010
+1 650 967-4675
support@pbspro.com

www.pbspro.com

VERIDIAN