PBS Training Class Notes

PBS Pro Release 5.1

(Three Day Class)

TM

www.PBSpro.com

VERIDIAN

Veridian Systems, Inc. is an operating company of the Veridian Corporation.
For more information about Veridian, visit the corporate website at: www.veridian.com.

PBS Training Class Notes (Three Day Class)

Release: 5.1, Printed: Fall 2001

Primary authors include:

James Patton Jones
Rita Williams

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

Veridian Systems
PBS Products Dept.
2672 Bayshore Parkway, Suite 810
Mountain View, CA 94043

Phone: +1 (650) 967-4675
FAX: +1 (650) 967-3080

URL: www.pbspro.com
Email: sales@pbspro.com

Contents

Key Tables and Figures

The Portable Batch System (PBS)
User and Administrator Training
Three Day Class

James Patton Jones
jjones@pbspro.com

Veridian
PBS Products Dept.

2672 Bayshore Parkway, Suite 810
Mountain View, CA 94043
(650) 967-4675
www.pbspro.com

VERIDIAN

PBS Training

- Logistics
- Outline:
  - Speaker & Class Introductions
  - PBS Introduction
  - Installation and Basic Configuration
  - Using PBS
  - Advanced Configuration
  - Administration
  - Troubleshooting

PBS - ***The Portable Batch System***

Flexible workload management and job scheduler

- Unified interface to all computing resources
  - All major UNIXs supported, heterogeneous environment, SMPs & clusters, parallel jobs (MPI)
  - Single interface handles both interactive and batch processing
  - GUI tools for user and administrator
  - POSIX batch standard
  - Source code included

- Fully configurable scheduler module -- any site policy
  - fair share, load balancing, priorities, back-filling, meta-scheduling
- Sophisticated fault tolerance, accounting, security (ACLs), automatic file staging
- Professional services: commercial support & training

www.pbspro.com

The History of PBS

- NASA developed COSMIC NQS 15+ years ago to manage batch queuing on the supercomputers of the time, and it quickly became the *de facto* standard.
- Later, the introduction of parallel and distributed memory machines created a need for an NQS replacement.
- Many (35+) batch systems were developed by sites around the world, but none met all the needs of NASA and other large government labs.

The History of PBS, cont.

- NASA embarked upon a project to produce a replacement for NQS. The new system had to be:
  - Capable of managing parallel and cluster systems as well as traditional supercomputers.
  - Extensible and maintainable.
  - Able to support any scheduling policy.
  - POSIX 1003.2d compliant
- The result was PBS: the Portable Batch System.

The History of PBS, cont.

- Between 1995 and 1998, NASA distributed PBS to about 70 sites in the U.S. under a restricted-access beta-software program.
- In 1998 MRJ Technology Solutions, the R&D contractor that developed PBS for NASA, took over development, distribution and support.
- There are now over 4000 registered PBS sites around the world.
- In late 1999 MRJ was acquired by the Veridian Corporation, and in April 2000 the PBS team moved to Veridian Systems to become the PBS Products Department.

About Veridian

- An information technology solution company delivering trusted solutions in the areas of national defense, critical infrastructure and essential business systems.
- Privately-held, $650M annual revenues, 5,000 IT professionals, more than 50 locations world-wide
- Known for building strong, long-term relationships with highly sophisticated customer base
- Board of Directors includes Dr. Joseph P. Allen, Neil A. Armstrong, and Dr. Sally K. Ride
- **PBS Products Dept** -- Original developers of PBS
  - Focused on hardening, enhancing, and supporting PBS
  - Offices in California, Georgia, and Virginia

Two versions of PBS:

- OpenPBS
  - *de facto* standard scheduler for Linux Clusters
  - open source, source code distribution
  - based on last NASA release of PBS
  - active user community (> 1500 active sites)
  - supported and maintained by Veridian
- PBS Pro
  - new enhanced, “hardened”, professional version
    - binary distribution, new docs, lots of new features
  - primary engineering focus of PBS team
  - open technology
    - advantages of open source, in a commercial package
    - binary, “shrink-wrapped” package
    - source code availability

Two versions of PBS:

- This class covers:
  - PBS Pro 5.1
    - (released Summer 2001)
  - OpenPBS 2.3
    - (released Sept 2000)
  - PBS Futures
    - PBS Pro 5.2 due early 2002
    - OpenPBS 2.4 due fall 2001

Currently Supported Systems

Workstations/Servers

- Sun SPARC w/ Solaris 2.3-2.8
- DEC ALPHA w/ Digital Unix 4.x, Tru64
- HP 9000 w/ HP-UX 9.x, 10.x, 11.x
- IBM RS/6000 w/ AIX 3.2, 4.1-4.3
- SGI systems w/ IRIX 5.x, 6.1-6.5.x
- Intel & Alpha systems: FreeBSD, NetBSD, Redhat Linux 5.x, 6.x, SGI Linux

Parallel Supercomputers

- Cray T3D w/ UNICOSMK
- Cray T3E w/ UNICOS/mk2
- SGI O2000/O3000 w/ IRIX 6.4, 6.5.x
- IBM SP-series w/ AIX 3.2, 4.1-4.2 with (PSSP 2.1) and AIX 4.3 (PSSP 2.3)

Vector Supercomputers

- Cray SV1 w/ UNICOS 10
- Cray C90 w/ UNICOS 8, 9, 10
- Cray J90 w/ UNICOS 8, 9, 10
- Fujitsu VPP300 w/ UXP/v

PBS Pro 5.1 -- Summer 2001

- Full SMP cluster support
- Increased fault tolerance
- Increased integration with specific systems
- Simplified, more flexible installation for binary release
- Preemptive Scheduling
- Enhanced advance reservation features
  - ACL control over reservations
- Ability to tie specific nodes to a queue
- Simplified user authentication for sites with common user name space
- New node specification syntax (now consistent across all architectures, offers user greater control)
- Support for OpenMP jobs
- For details see:
  - http://www.PBSpro.com/new_features.html

PBS Pro 5.2 and Beyond...

- System-specific capabilities
  - Sun CRE (parallel task spawning will use the PBS TM API)
  - SGI MPI and job accounting
- Windows 2000 and Mac OS-X ports
- Web-enabled interface
- High-availability option
- Automatic resource detection and configuration
- High-throughput file-staging module
- Accounting, metrics, and reporting tools
- New security model (PKI, Kerberos, DCE)
- Standards & computational grids

Recent PBS News...

- PBS being bundled in various Cluster Kits:
  - IBM, SGI, OSCAR, ...
- Reduced pricing for *PBS Pro for Linux*
- Educational discounts announced
- PBS *Advance Reservation* feature demonstrated at Supercomputing 2000 Conference
- PBS articles published in *Mountain View Voice* newspaper (May 2001), *HPCwire* (Apr 2001), NASA *Ames Astrogram* (Apr 2001), and *Aerospace Innovations* magazine (Dec 2000).
- PBS Press Releases published in *HPCwire* (Nov 2000) and *SysAdmin* Magazine (Jan 2001).

Resource Management 101

- A resource management or batch queuing system has three primary roles:
  - ***Queuing*** of work or tasks to be run on a computer. Users submit their jobs to the batch system where they are queued up until the system is ready to run them
  - ***Scheduling***, or the process of selecting which jobs to run when and where, according to a predetermined policy. Sites try to balance competing needs and goals on the system; scheduling is often wrought with compromise. You can't please all the users all the time...
  - ***Monitoring***, tracking and reserving system resources, and enforcing usage policy. Covers user-level and system level monitoring; also monitoring of the scheduling algorithms to see how well they are meeting the stated goals

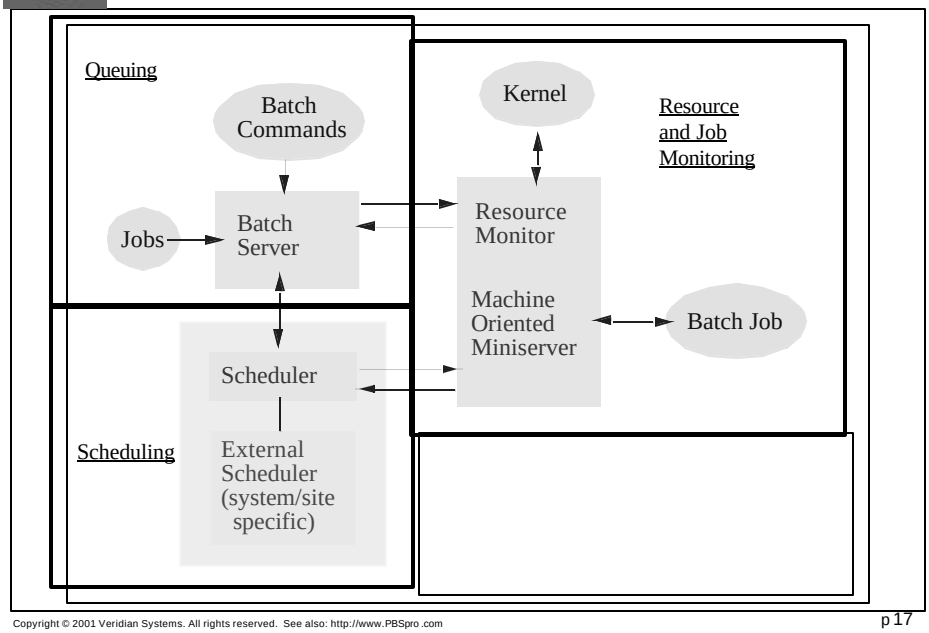
The Anatomy of PBS: System Overview

- A client-server TCP-based system
- Can run on a standalone system, or in a distributed, heterogeneous UNIX environment
- Utilizes UNIX security and access control (reverse hostname/IP validation, ruserok(), ACLs)
- Completely modular (eg. security model can be replaced; username-mapping hooks provided, etc.)

The Anatomy of PBS: Components

- PBS Daemons
  - batch server
  - resource monitor + mini-server
  - scheduler
- PBS User Commands
  - qsub, qstat, qdel, qalter, qhold, qrels, xpbs, …
- PBS Administrator Commands
  - qmgr, qrerun, pbsnodes, tracejob, xpbsmon, ...

The Anatomy of PBS: The Big Picture



The Anatomy of PBS: Batch Server Daemon

PBS Batch Server

- Maintains queues and jobs
- Communicates with:
  - Client commands
  - Mini-server
  - Resource Monitor
  - Scheduler
  - Other servers (forwarding jobs, status requests)

The Anatomy of PBS: MOM Daemon

Machine Oriented Mini-server (MOM)

- MOM
  - Executes jobs at request of Server
- Resource Monitor
  - Monitors resource usage of running jobs
  - Enforces limits on jobs
  - Reports system resource limits, configuration (e.g. memory, CPU utilization, swap rate, etc.)

The Anatomy of PBS: Scheduler Daemon

The Scheduler

- The PBS scheduler serves in the role of implementing local site policy.
  - Queries list of running and queued jobs from the Server
  - Queries queue limits, etc. from the Server
  - Queries resource consumption and availability from MOM
  - Sorts available jobs according to local policy
  - Selects and runs jobs according to local policy, available resources, pending deadlines, etc.

The Anatomy of PBS: Admin Commands

- Communicate directly with Server
- PBS admin (and root) can query, modify, delete… any job in system
- Shutdown/Restart Server
- Move jobs between servers
- Manually run, kill, stop, resume jobs
- qmgr - manager interface (queue definition, start/stop queues, edit ACLs, etc.)
- xpbsmon - tcl/tk GUI to monitor PBS environment

The Anatomy of PBS: Client Commands

- Communicate directly with Server
- Submit, query, alter, delete, … user's own jobs
- Can be installed on standalone systems
- qsub, qstat, qalter, qdel, qhold, qrls,
- xpbs - tcl/tk GUI to client commands

Using the PBS website as a resource...

www.PBSpro.com

www.OpenPBS.org

- Software Downloads
- Frequently Asked Questions (FAQ)
- Searchable archive of pbs-users Mailing List
- Postscript and PDF versions of PBS Docs:
  - Administrator's Guide -- configuration information
  - External Reference Spec. -- overview of all components, glossary of terms, manual pages for all external PBS API calls.
  - User's Guide -- New in PBS Pro 5.1

PBS Administrator Guide (AG)

- Intended for **System Administrators** or **System Managers**
- Provides information on downloading, configuring, building, installing, testing, and maintaining PBS.
- This class will cover highlights and critical information.
- In addition, this class will point out specific sections of the Admin Guide useful for particular tasks not covered in this class.

PBS External Reference Specification (ERS)

- Recommended Reading:
- **Programmers** wishing to add an interface to PBS to their code should read the sections listed for the general user and chapter 4.
- **System Operators** should read the sections listed under general user plus chapters 6 and 7.
- **System Administrators** are advised to read the entire ERS with the possible exception of chapters 4 and 11.
- The **General User** may wish to read the ERS Chapters 1, 2.2, 2.7, 3.2 - 3.4, and 5.

PBS User Guide (UG)

- New in PBS Pro 5.1
- Intended for **PBS Users**
- Provides information on creating, submitting, monitoring, tracking, and manipulating PBS jobs.
- This class will cover highlights and critical information.
- In addition, this class will point out specific sections of the User Guide useful for particular tasks not covered in this class.

Acquiring / Downloading PBS

- The latest version of PBS Pro (as well as previous releases and beta-versions of new software) can be downloaded from the Online Customer Support Center on the the PBS Website:
  - http://www.PBSpro.com/UserArea/
  - (SiteID and password is required for access.)
- The latest OpenPBS release can be download from:
  - http://www.OpenPBS.org/UserArea/
  - (SiteID and password is required for access.)

Installing PBS

- Two methods for installing PBS:
  - From Binary Distribution
  - From Source Distribution
- PBS Pro is normally installed from Binary Dist., but source distribution is available if you sign the source code license.
- OpenPBS is shipped as a source distribution (except both binary RPMs are available for Redhat Linux)

Installing PBS from Binary Distribution

1. Read Admin Guide and plan general configuration
2. Decide where PBS files are to go
3. Prepare distribution media
4. Install PBS using `INSTALL` tool
5. Install PBS license key
6. Review/Edit nodes file
7. Start PBS daemons
8. Test by running a few jobs

Installing PBS from Binary Distribution

- Prepare distribution media:
  - Via CD-ROM
    - load and mount PBS Pro CD-ROM
    - change directory to CD-ROM mount point

```
# mount /cdrom
# cd /cdrom
```

  - Via Download
    - download install package
    - extract package to temporary location

```
# mkdir /tmp/pbs_tmp
# cd /tmp/pbs_tmp
# uncompress /tmp/pbspro_5_1.tar.Z
# tar -xvf /tmp/pbspro_5_1.tar
```

Installing PBS from Binary Distribution

- Run installation program, answering questions...

```
# ./INSTALL
PBS Installation:
       1. Server, execution and commands
       2. Execution only
       3. Commands only
(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

Processing package instance <pbs64> from
</cdrom/pbspro_5_0/solaris/pbs64>
pbs64    (sparc) 5.0
Veridian Systems PBS department
## Processing package information.
## Processing system information.
...
```


Installing PBS from Binary Distribution

```
## 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
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.
```

Installing PBS Pro License Key

- The `INSTALL` program will next ask for and install your PBS license key(s):

```
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'
```


Building and Installing PBS from Source

1. Read Admin Guide and plan general configuration
2. Decide where PBS source and objects are to go
3. Unzip and untar the distribution file in to the source tree
4. Change to the directory which is to be the top of the object tree
5. Select “configure” options; run configure from top of object tree
6. Compile PBS by typing “**`make`**” at top of object tree
7. Install PBS by typing “**`make install`**” at top of object tree
8. Create node file
9. Bring up and configure the Server
10. Configure and bring up the MOM
11. Test by hand scheduling a few jobs
12. Configure and start a scheduler program
13. Set the server to active by enabling scheduling

Installing PBS from Source Distribution

- From the top of your object tree, run the `configure` command with the options specific to your site, then build and install PBS:

```
# ./configure [options]
# make
# make install
# make postinstall
```

- The last step shown ("`make postinstall`" applies only to PBS Pro, not OpenPBS)
- Admin Guide has detailed description of all the `configure` options, and various suggestions.

Configuration Considerations

- The following are options to "configure" used on an example Origin3000 system:

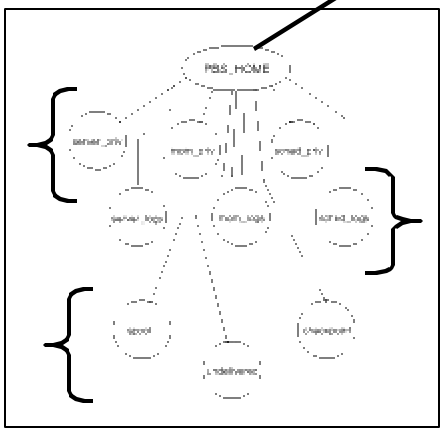
```
Configure --prefix=/usr/local                                          \
          --enable-docs                                                \
          --enable-gui                                                 \
          --set-cc='cc'                                                \
          --set-cflags='-n32'                                          \
          --with-tcl=/usr/local/pkg/tcl-tk/tcl8.0-tk8.0-64             \
          --set-server-home=/usr/spool/PBS                             \
          --set-server-name-file=/usr/spool/PBS/default.name           \
          --set-environ=/usr/spool/PBS/pbs_environment                 \
          --set-disable-syslog                                         \
          --set-sched='cc'                                             \
          --set-mansuffix=''                                           \
          --with-scp                                                   \
          --enable-shell-pipe
```

The PBS Directory Structure

- Overview of Structure

"root" of PBS tree often called "/pbs"

daemon configuration directories



daemon log directories

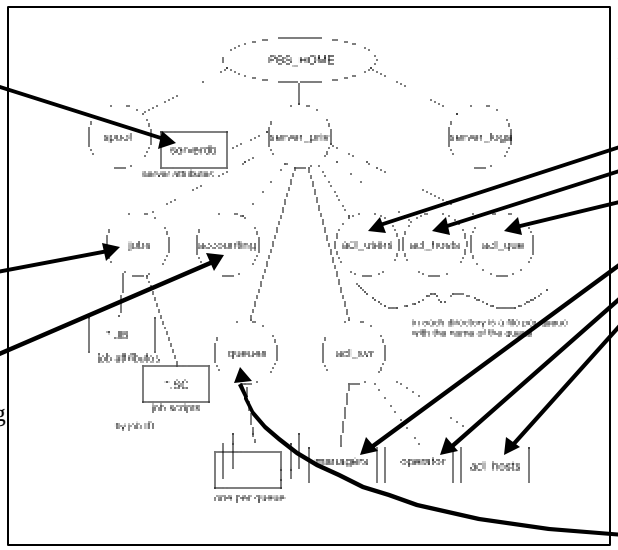
various file-related directories

The PBS Directory Structure: server_priv

Database of server and queue information

Contains two files for each job on server

Contains a file per day of accounting information



Access Control Lists (ACLs) for:
- users
- hosts (user)
- queues

- managers
- operators
- hosts (server)

Contains one file for each queue defined on server

Configuration Files: Server

- queue manager utility (qmgr)
  - create, modify, and delete queues, queue limits, etc.
  - set or modify server limits
  - set or modify Access Control Lists
  - set or modify scheduling status (idle or active)
  - set or modify scheduling interval
  - set or modify server defaults
  - CLI access only
  - online help (type “help” at qmgr prompt)
  - detailed discussion also in Admin Guide and ERS

Configuration Files: Server

- queue manager utility (qmgr)

```
alabama % qmgr
Max open servers: 4

Qmgr: print server
# 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

Qmgr: quit
```

Configuration Files: Server

- Access control lists (and how to use them)
  - can prevent or allow access to PBS
  - per-user, per-host, and/or per-domain
  - can be set on individual queues:

```
Qmgr: set queue workq acl_user_enable = True
Qmgr: set queue workq acl_users = utley
Qmgr: set queue workq acl_users += maryh
Qmgr: set queue workq acl_users += jjones@alabama.pbspro.com
Qmgr: set queue workq acl_users += hender@*.pbspro.com
```

Permits access by named users

Permits access by named user ONLY from named host

Permits access by user from any host on named network

  - can also be set for the entire server:

```
Qmgr: set server acl_host_enable = True
Qmgr: set server acl_hosts = *.pbspro.com
Qmgr: set server acl_hosts -= www.pbspro.com
Qmgr: set queue acl_hosts -= gateway.pbspro.com
```

Permits access from any host on named network

Deny all access from named hosts

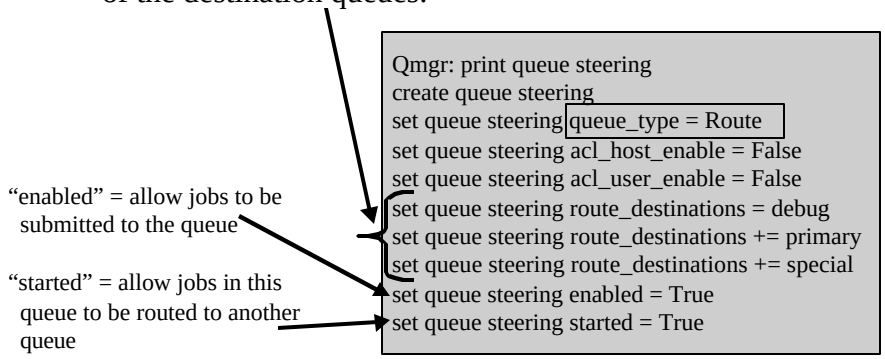
Configuration Files: Server

- Server attributes

```
Qmgr: print server
…
# Set server attributes.
#
Automatic scheduling?...  set server scheduling = True
Include a host ACL?...    set server acl_host_enable = False
Define the PBS managers:... set server managers = jjones@origin.pbspro.com
                          set server managers += utley@origin.pbspro.com
                          set server managers += hender@origin.pbspro.com
What queue to use if not specified?... set server default_queue = workq
Volume of log messages to be written:... set server log_events = 511
PBS email "reply-to" address to use:... set server mail_from = adm
Allow users to view other users' jobs?... set server query_other_jobs = True
Define default resources:... set server resources_default.ncpus = 1
                          set server resources_default.pmem = 256mb
Automatic scheduling polling interval:... set server scheduler_iteration = 300
Define default node for scheduling:... set server default_node = o2k01
```

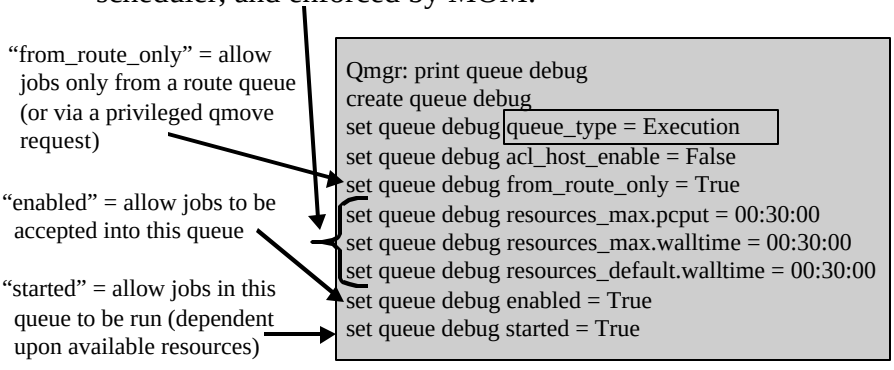
Configuration Files: Server

- Queue Attributes
  - Two types of Queues: Route and Execution
  - Route Queues funnel jobs into other queues by comparing resources requested by a job to the resource limits on each of the destination queues:



Configuration Files: Server

- Queue Attributes
  - Execution Queues hold jobs until they are either run or moved elsewhere. Typically execution queues have resource limits defined by the local site, utilized by the scheduler, and enforced by MOM.

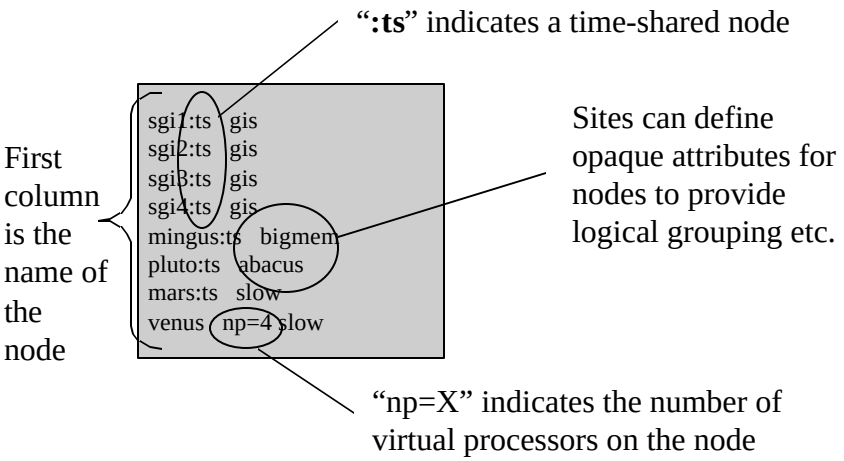


Configuration Files: Nodes File

- What is a “node”?
  - Within PBS a node is a computational unit, one or more of which can be allocated to a job in a *shared* or *exclusive* manner.
  - A node is characterized by:
    - a unique memory address space
    - a unique operating system image
    - one or more IP addresses
    - a PBS MOM daemon

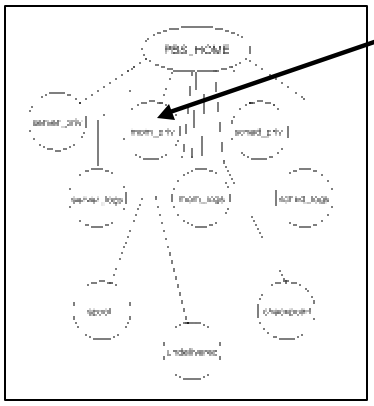
Configuration Files: Nodes File

- Example Nodes File:



Directory Structure: MOM

- Where the MOM configuration file is located



The configuration file for each MOM is located in the PBS filetree under the directory mom_priv in the file call “config”.

Configuration Files: MOM

- Privileged and restricted connections
  - By default, MOM refuses all connections or requests for information from all but the host she is running on.
  - But each MOM can be configured to permit connections from clients on specific hosts (or networks) that which to query information from her.
  - Privileged connections are those that MOM will trust completely, and will accept ‘run job’ requests from.
  - Restricted connections are those that MOM will only answer resource requests from.

Configuration Files: MOM

- Privileged and restricted connections

Example MOM config file from fermi:

Privileged connection:

```
$clienthost localhost
$clienthost turing
$clienthost turing-ec
$clienthost steger
$clienthost steger-ec
$clienthost hopper
$clienthost hopper-ec
$clienthost piglet
$clienthost evelyn
$clienthost fermi
$clienthost fermi-hp
```

Restricted connection:

```
$restricted *.veridian.com
$logevent 0x3f
```


Configuration Files: MOM

- Controlling what's written to MOM's log file
  - A "$logevent" entry in MOM's config file sets the mask that determines which event types are logged by pbs_mom.

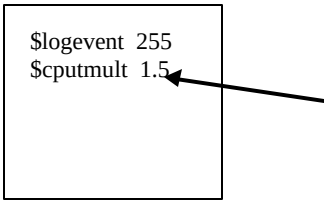
```
$logevent  0x1ff
$logevent   255
```

Sets the log event mask to 0x1ff (511) which enables logging of all events including debug events.

Sets the log event mask to 0x0ff (255) which enables all events except debug events.

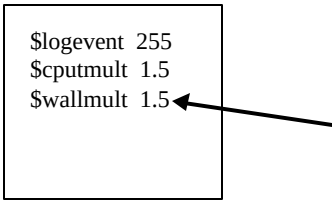
Configuration Files: MOM

- Adjusting CPU charges across systems
  - A "$cputmult" entry in MOM's config file sets a factor used to adjust cpu time used by a job. This is used to allow adjustment of time charged and limits enforced where the job might run on systems of different speeds.



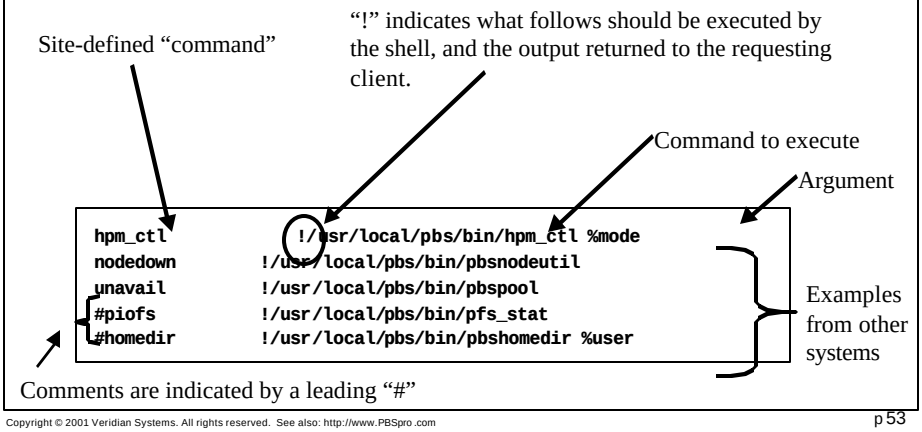
Configuration Files: MOM

- Adjusting Walltime charges across systems
  - A "$wallmult" entry in MOM's config file sets a factor used to adjust wall time used by a job.



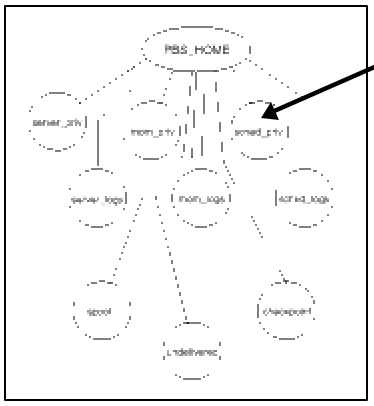
Configuration Files: MOM

- Extending MOM's query capabilities
  - Sites can add entries to the MOM config file to have certain commands executed upon request.



Directory Structure: Scheduler

- Where the Scheduler configuration file is located:



The configuration file for the Scheduler is located in the PBS filetree under the directory sched_priv in the file call "config" or "sched_config".

PBS Scheduling Concepts

- The concept of “site- or system-specific” policy.
  - A site tries to balance competing demands on the systems:
    - Users want fast turnaround of their jobs.
    - Managers want the highest possible utilization of the system.
    - Administrators want a static system (set it up and leave it alone).
  - The modular design of the PBS scheduler allows a site to create and implement exactly the policy they would like to have applied to running their jobs. They can do this by either combining scheduling features to implement their desired policy, or customizing the scheduler code to meet their needs.

PBS Scheduling Concepts

The PBS Pro Standard Scheduler

- The PBS Pro scheduler (called “standard”) is a sophisticated general purpose scheduler implementing a variety of (selectable) scheduling algorithms, including:
  - first-in, first-out
  - round robin
  - load-balancing
  - load-stacking
  - fair share
- Sites can edit the configuration file to change behavior (see section *Standard Scheduler* in the Admin Guide.)

OpenPBS Schedulers

- Several system-specific schedulers are included in OpenPBS which are optimized for specific systems:
  - Cray C90/J90 UNICOS
  - Cray T3e UNICOS/mk
  - IBM SP
  - SGI Origin2000
- The most of the feature set of all the above schedulers is provided in the PBS Pro Standard Scheduler

The Easily Extended Scheduler

- The PBS scheduler is designed to be easily extended. Two interfaces are provided to implement any scheduling policy:
  - TCL/tk - scripting language interface (good for quick prototyping)
  - C - most PBS scheduler are written in C using the PBS API
- For details on customizing or extending the PBS scheduler module, see the Customization sections of the Admin Guide and ERS.

Configuration Files: Scheduler

- Examples of scheduler configuration files
- See the actual files on each system or printouts available in class.
- Many configuration parameters.
- PBS administrators use input from management and feedback from the users to optimize the scheduling of a particular system to the local needs.

Options for PBS Pro Scheduler

round_robin:	False	all	Run a job from each queue in turn
by_queue:	True	prime	Review jobs by queue, rather than looking at all jobs on server
by_queue:	True	non_prime	
strict_fifo:	False	ALL	Run jobs in strict FIFO order
fair_share:	False	ALL	Run jobs based on usage and share values
backfill:	True	ALL	Run short jobs around long-waiting jobs
help_starving_jobs	True	ALL	Give preference to long-waiting jobs
sort_queues	True	ALL	Sort queues by priority attribute
load_balancing:	False	ALL	Load balance between timeshared nodes
load_balancing_rr:	False	ALL	Load balance via round-robin through node list
sort_by:	shortest_job_first	ALL	Sort key; see Admin Guide for list of options
log_filter:	256		Filter out debug log messages
dedicated_prefix:	ded		Queues with this prefix are “dedicated-time”
max_starve:	24:00:00		How long should a job wait before its “starving”
half_life:	24:00:00		Define “half-life” for decay of recent usage
unknown_shares:	10		Number of shares for the “unknown” group
sync_time:	1:00:00		Amount of time between syncing usage data to disk

Starting and Stopping PBS Pro

- Control File
  - `/etc/pbs.conf`
  - controls which daemons should be running
  - each node should have own copy
- PBS start/stop script
  - reads /etc/pbs.conf file to determine which daemons should be started
  - `/etc/init.d/pbs [ status | stop | start ]`

Manually Starting OpenPBS

- The recommended order of starting the OpenPBS daemons is:
  - pbs_sched [options]
  - pbs_server [options]
  - pbs_mom [options]

Manually Stopping OpenPBS

- The recommended order of stopping the OpenPBS daemons is:
  - qterm (to perform an orderly shutdown of the Server)
  - kill -TERM {MOM's PID} {Scheduler's PID}
- The current PIDs of all the daemons is written to a file in each daemon's _priv directory under the name {daemon}.lock, eg:
  - /PBS/sched_priv/sched.lock

Using PBS

- The next section focuses on Using PBS:
  - The PBS User Environment
  - Creating, Submitting, Tracking, and Monitoring Jobs
  - Using the xpbs Graphical Interface
  - Using the various PBS features and commands

The PBS Environment

- The Two Faces of PBS
  - Command Line Interface (CLI)
    - type commands at UNIX prompt
    - available on all systems
  - Graphical User Interface (GUI)
    - a graphical point-and-click interface
    - available on sub-set of systems

  - This class will cover both interfaces

The PBS Environment

- The Two Batch Job Types
  - A PBS “batch” job is a shell script containing the set of commands to be run on the supercomputer.
    - Examples will be shown below
  - A PBS “interactive-batch” job request is treated like a batch job (it is queued up, and must wait for resources to become available before it can run). But once it is started, the user’s terminal input and output are connected to the job in what appears to be an “rlogin” session. It appears the user is logged into the supercomputer.
  - Most of this class will focus on “batch” jobs

The PBS Environment

- Various variables are set by PBS in the environment of batch jobs. Some of the more useful include:

```
PBS_O_HOME=/u/jjones
PBS_O_LOGNAME=jjones
PBS_O_PATH=/usr/new/bin:/usr/local/bin:/bin
PBS_O_SHELL=/sbin/csh
PBS_O_TZ=PST8PDT
PBS_O_HOST=origin.pbspro.com
PBS_O_WORKDIR=/u/jjones
PBS_O_QUEUE=submit
PBS_JOBNAME=INTERACTIVE
PBS_JOBID=16386.origin.pbspro.com
PBS_QUEUE=origin
PBS_ENVIRONMENT=PBS_INTERACTIVE
```


The PBS Environment

- The full list of environment variables can be seen in several places:
  - manual page for “qsub”
  - PBS *User Guide*
  - PBS *External Reference Spec*.

The PBS Environment

- PBS Requirements for Successful Job Submission
  - User must have access to the requested resources/host
    - rsh/rcp
    - ssh/scp
    - target hostname in .rhosts or /etc/hosts.equiv
  - User must have a valid group/account
  - Some sites require users to have a non-zero allocation

The PBS Environment

- Setting Up Your Own PBS Environment
  - A user's job may not run if the user's start-up files (.cshrc, .login, or .profile) contain commands which attempt to set terminal characteristics. Any such activity should be skipped by placing a test of the environment variable PBS_ENVIRONMENT (or for NQS compatibility, ENVIRONMENT). This can be done as shown in the following sample .login:

```
...
setenv MANPATH /usr/man:/usr/local/man:$MANPATH
if ( ! $?PBS_ENVIRONMENT ) then
      do terminal stuff here
endif
```

The PBS Environment

- Setting Up Your Own PBS Environment
  - Users should also be aware that commands in their startup files should not generate output when run under PBS. As in the previous example, commands that do write to stdout should not be run for a PBS job. This can be done as shown in the following sample .login:

```
...
setenv MANPATH /usr/man:/usr/local/man:$MANPATH
if ( ! $?PBS_ENVIRONMENT ) then
      do terminal stuff here
      run setup commands with output here
endif
```


The PBS Environment

- Setting Up Your Own PBS Environment
  - When PBS jobs run, the “exit status” of the last command executed in the job is reported by csh to PBS as the “exit status” of the job. However, if you have commands in your .logout, csh will report this exit status instead. To prevent this, you need to preserve the job’s exit status in your .logout file:

```
set EXITVAL = $status

  previous contents of .logout here

exit $EXITVAL
```

Submitting a PBS Job

- What is a PBS Job? A shell script with resource attributes and the list of commands to be run. E.g.:

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -j oe

cd ${HOME}/PBS/test
./subrun
```

Shell to execute job

PBS Resource Definitions

The rest of the script consists of the commands the user wants to run

- Let's assume this script is in a file called "mysubrun"

Submitting a PBS Job

- Using the PBS qsub command for job submission:

```
% qsub mysubrun
16387.origin.pbspro.com
```

- Can override resource attributes contained in job script by specifying them on the command line

```
% qsub -l ncpus=16 -l walltime=4:00:00 mysubrun
16388.origin.pbspro.com

% qsub -l mem=1gb,walltime=3:00:00 mysubrun
16389.origin.pbspro.com
```

Submitting a PBS Job

- When submitting a job via either qsub or xpbs, you get back a “batch job identifier”:

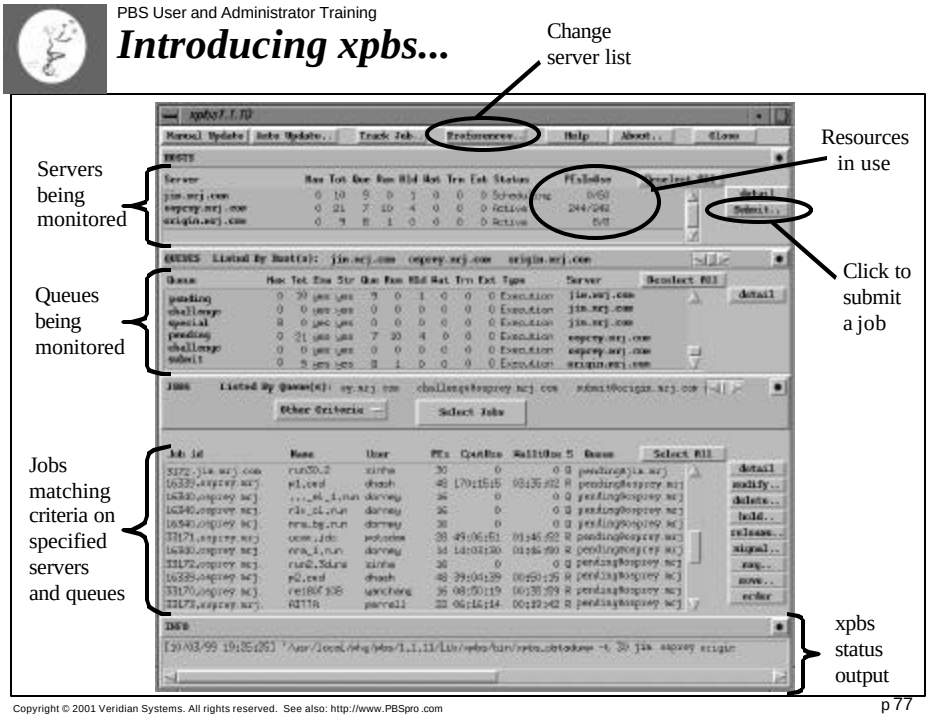
```
% qsub mysubrun
16387.origin.pbspro.com
```

- The identifier is a ‘handle’ to the job. You’ll need it for any actions involving the job, such as:
  - checking job status
  - modifying the job
  - tracking the job
  - deleting the job

Relationship between PBS and xPBS

- xPBS is built on top of the PBS commands
- Each “task” that you perform using xPBS is converted into the necessary PBS command and then run on your behalf
- The “Info” window at the bottom of the main xPBS displays always shows the most recent PBS command executed. This is handy for teaching yourself complex PBS commands.

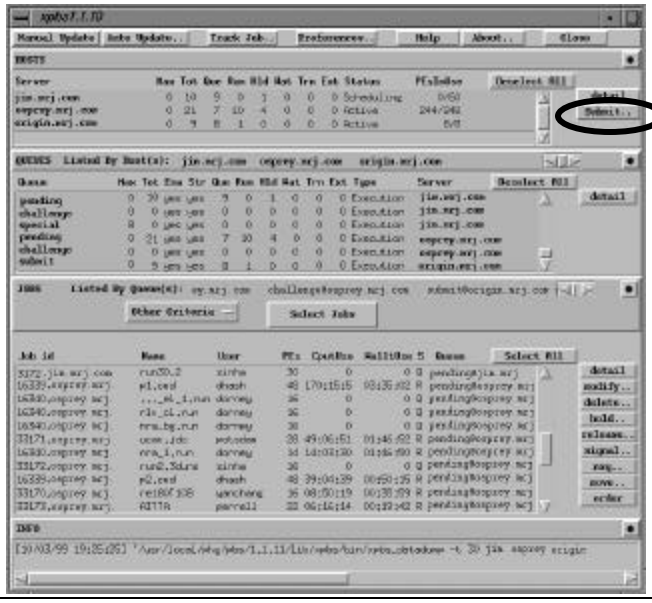
Introducing xpbs...



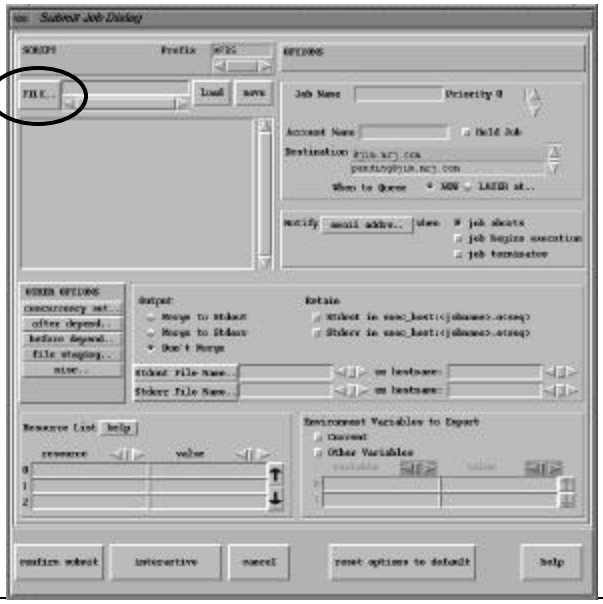
The xPBS Configuration File

- xPBS has a configure file:
  - when you close the xpbs window, it asks if you want to save your changes. If you click yes, it will write the configuration information to your home directory in a file called: .xpbsrc
  - There is also a system-wide configuration file that sets defaults for the anyone who doesn't have a .xpbsrc

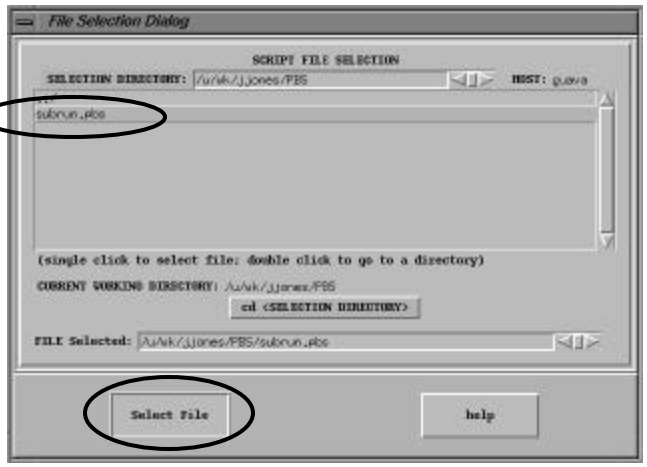
Submitting a job with xpbs



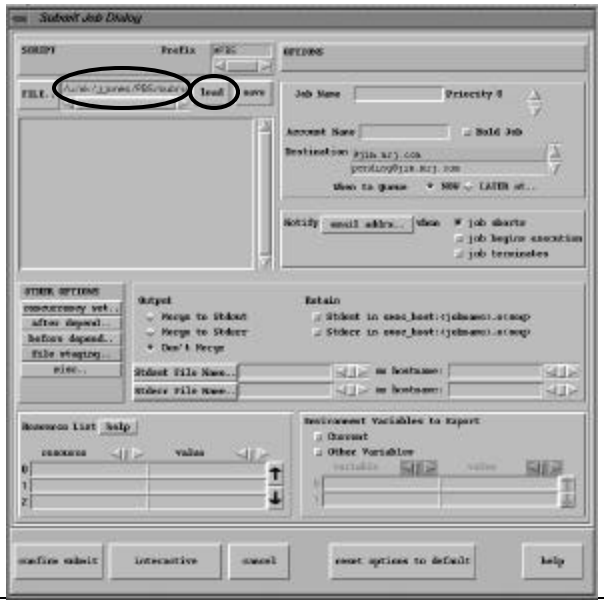
Submitting a job with xpbs



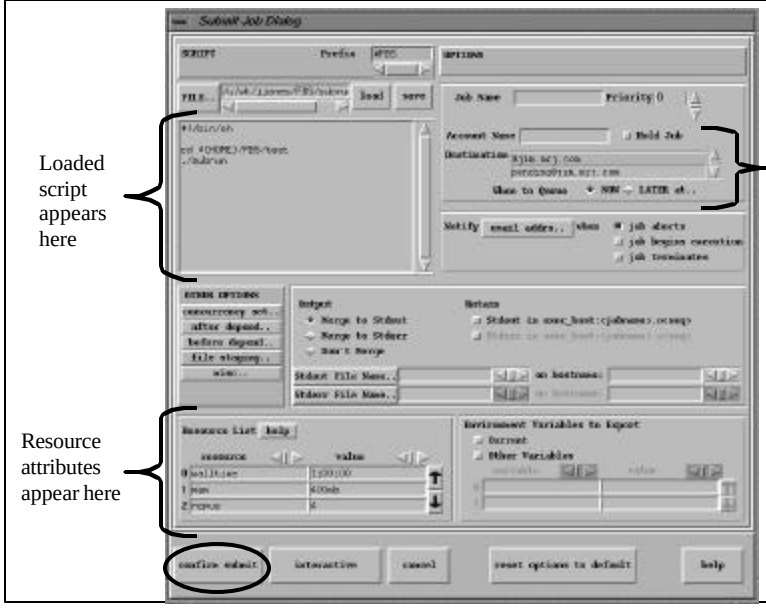
Submitting a job with xpbs



Submitting a job with xpbs

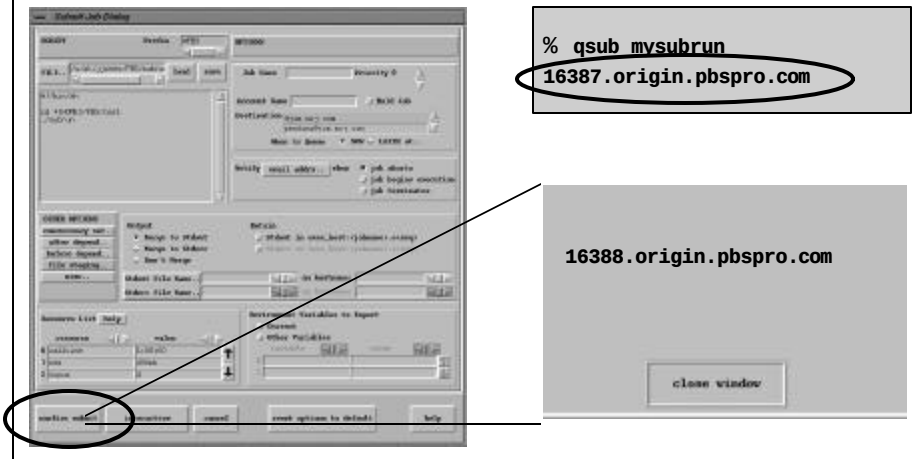


Submitting a job with xPBS

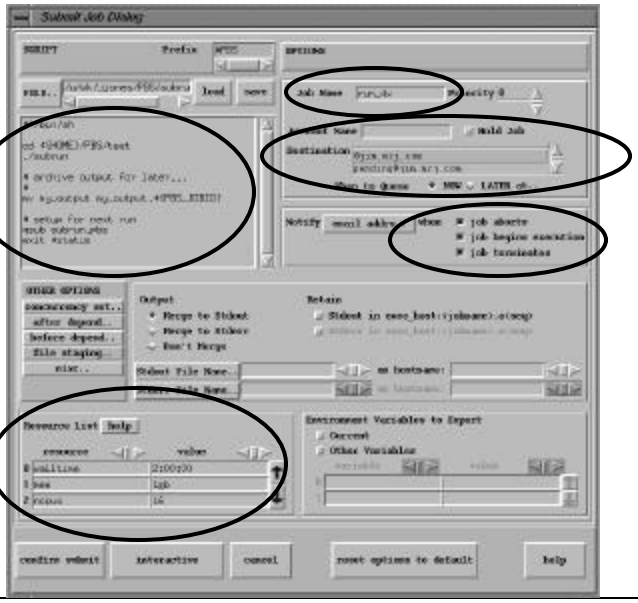


Submitting a job with xPBS

- Just as with qsub, when you submit a job using xpbs you will receive a JobID for the job.



Modifying an existing Job script with xpbs



Checking Job Status

- "qstat" is your friend
- by far the most used command in PBS
- used for:
  - getting status of jobs, queues, servers
  - finding out why a job is not running
  - seeing who else is running
  - viewing how full the system is

Checking Job Status

- Checking job status using qstat:

```
% qstat 16387
Job id            Name             User             Time Use S Queue
----------------- ---------------- ---------------- -------- - -----
16387.origin      subrun.pbs       jjones           00:06:39 R submit
```

Job Identifier

Job/Script Name

User Login Name

Time used in current state

Current State

Current Queue Name

R = Running
Q = Queued
E = Exiting
T = Transiting

Checking Job Status

- Checking job status using qstat:

Requested Resources

CPUs Memory Nodes Walltime

```
% qstat -a 16387

                                                                Req'd    Req'd   Ela
Job ID          Username Queue    Jobname    SessID  TSK Memory  Nds wallt S wallt
--------------- -------- -------- ---------- ------- --- ------- --- ----- - -----
16387.origin    jjones   submit   subrun.pbs   10806   4   400mb   2 01:00 R 00:06
```

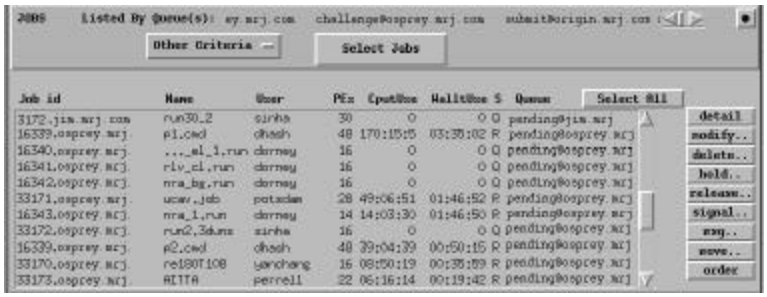
See previous slide

Session Identifier

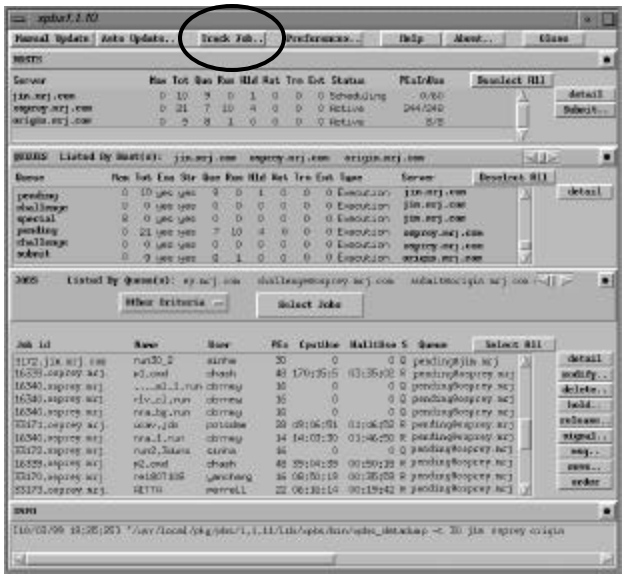
Current State

Time used in current state

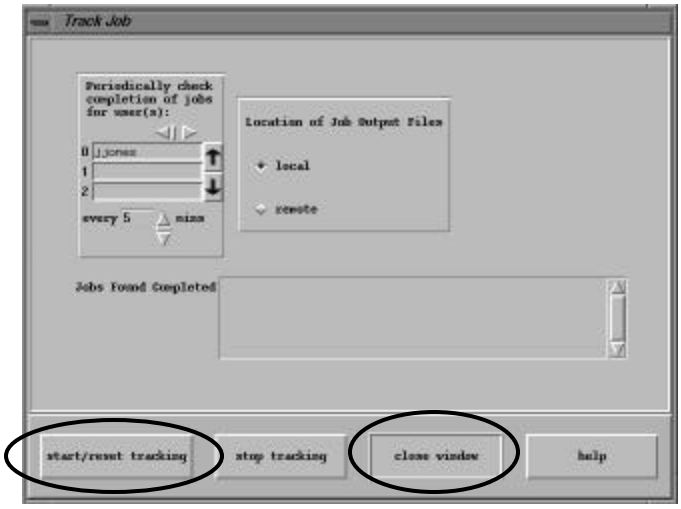
Checking Job Status



Tracking Jobs with xPBS



Tracking Jobs with xPBS



PBS System Resources

- You can request a variety of resources that can be allocated and used by your job:
  - CPUs
  - Memory
  - Time (walltime or cputime)
  - Disk space

PBS System Resources

- Use the resource list ("-l") option to qsub to specify requested system resources to PBS:

```
% qsub -l ncpus=16 -l walltime=4:00:00 mysubrun
16388.origin.pbspro.com

% qsub -l mem=1gb,walltime=3:00:00 mysubrun
16389.origin.pbspro.com
```

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -j oe

cd ${HOME}/PBS/test
./subrun
```


PBS System Resources

- Example System Resources

Resource	SGI O2k
Wallclock Time	walltime
Largest file size	file
Architecture	arch
Software	software
CPU Time	cput
Per-Process CPU Time	pcput
Memory	mem
Number of CPUs	ncpus

More Job Submission Options

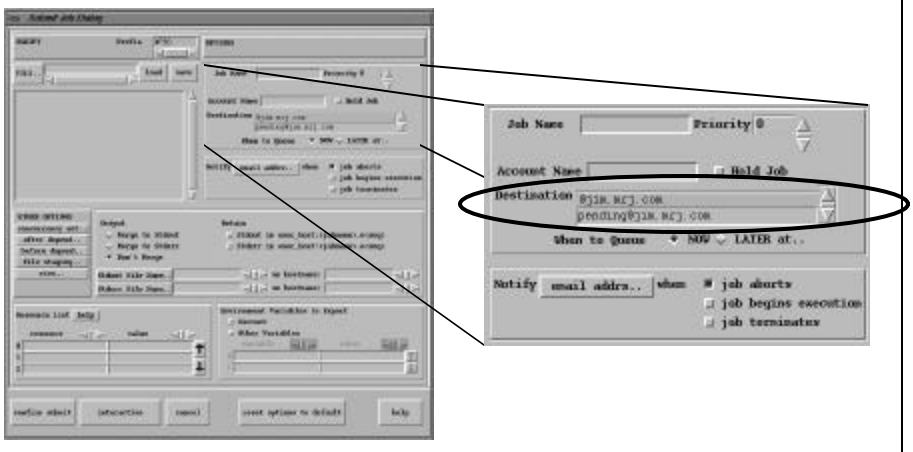
- Specifying Queue and/or Server
  - "-q QueueName"

```
% qsub -q challenge mysubrun
16390.origin.pbspro.com

% qsub -q special@steger mysubrun
16391.origin.pbspro.com
```

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -j oe
#PBS -q background

cd ${HOME}/PBS/test
./subrun
```


More Job Submission Options

- Specifying Queue and/or Server



More Job Submission Options

- Redirecting PBS generated output and error files

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -o my_output_file
#PBS -e my_error_file

cd ${HOME}/PBS/test
./subrun
```

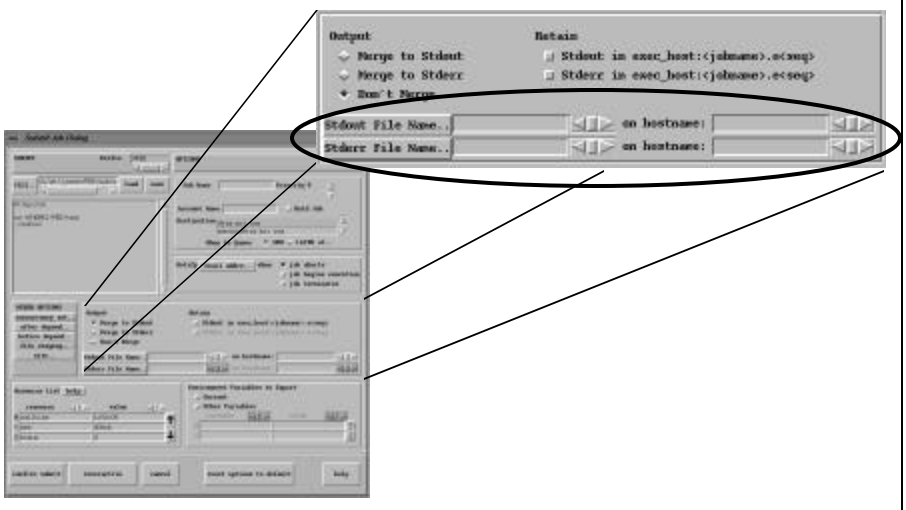
"stdout" output of program

"stderr" error output of program

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -o alabama:/u/jjones/my_output_file
#PBS -e alabama:/u/jjones/my_error_file

cd ${HOME}/PBS/test
./subrun
```


More Job Submission Options

- Redirecting PBS generated output and error files



More Job Submission Options

- Specifying e-mail notification
  - "-m MailOptions":
    - a - send mail when job is aborted by batch system
    - b - send mail when job begins execution
    - e - send mail when job ends execution
    - n - do not send mail

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -m abe

cd ${HOME}/PBS/test
./subrun
```


More Job Submission Options

- Specifying e-mail notification

More Job Submission Options

- Expanding and exporting job environment variables

  "-v variable_list" - expands list of variables available to job

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -v myvariable=true,mycolor=green

cd ${HOME}/PBS/test
./subrun
```

  "-V" - export all variables from qsub environment to job

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -V

cd ${HOME}/PBS/test
./subrun
```

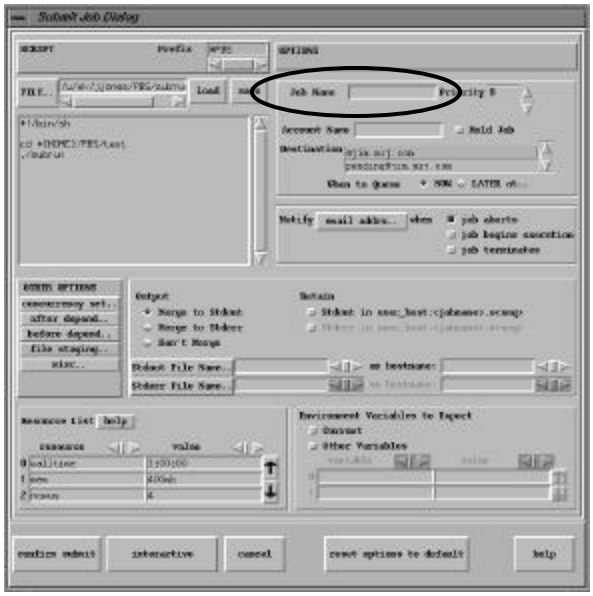

More Job Submission Options

- Specifying a job name
  - "-N jobname"

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -N Submarine

cd ${HOME}/PBS/test
./subrun
```

More Job Submission Options

Specifying
a job name



More Job Submission Options

- How to run interactive batch jobs

  Use the "-I" option to qsub, specifying any resource requirements needed just like a normal batch job:

```
% qsub  -I   -l walltime=1:00:00,mem=400mb,ncpus=4
waiting for job 1761.sunfarm.pbspro.com to start. . .
```

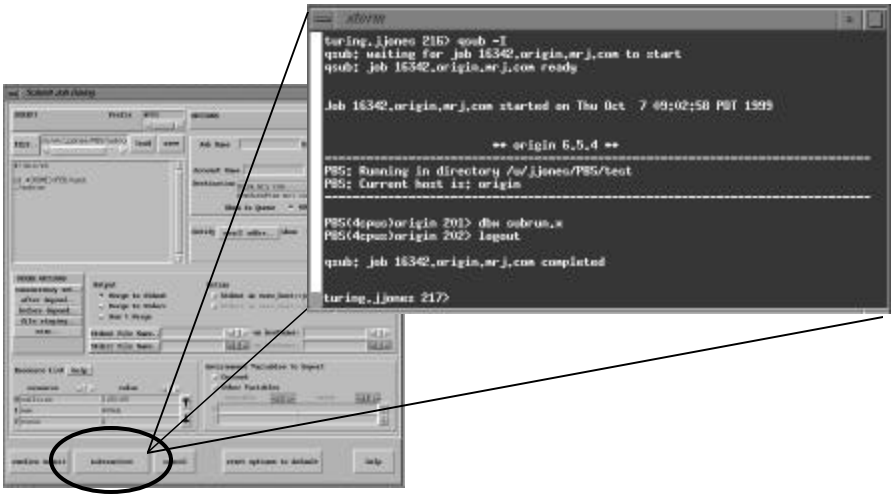
  When resources are available, then job will be run:

```
% qsub  -I  -l walltime=1:00:00,mem=400mb,ncpus=4
waiting for job 1761.sunfarm.pbspro.com to start. . .
qsub: job 1761.sunfarm.pbspro.com ready

PBS(sunfarm:4cpus)%
PBS(sunfarm:4cpus)%
```

  Once started the job appears as a login session on the system. However, resource limits will be enforced by PBS.

More Job Submission Options

- How to run interactive batch jobs



More Job Submission Options

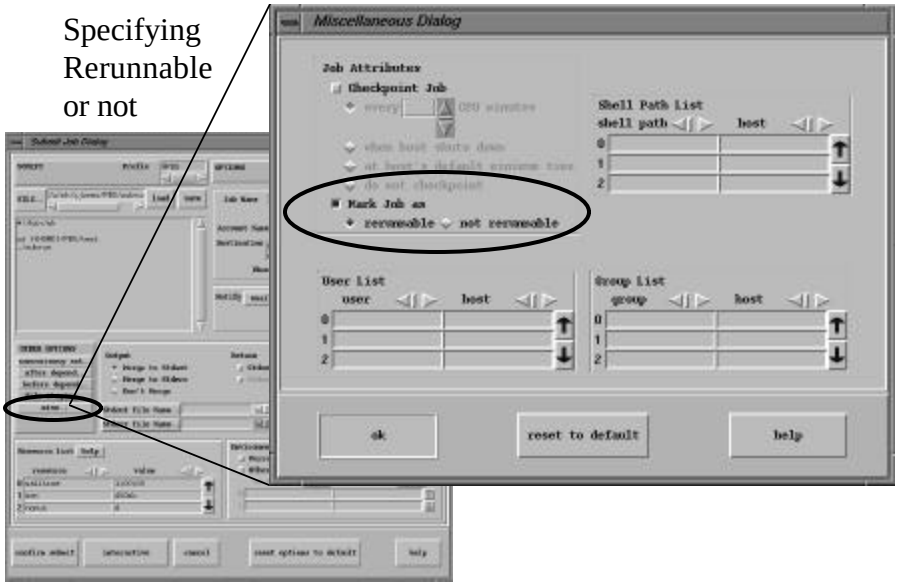
- Marking a job as “rerunnable” or not
  - “-r y|n” - “y” means the job is rerunnable
    - “n” means the job is not rerunnable

If not specified, the job is assumed to be rerunnable

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -r n

cd ${HOME}/PBS/test
./subrun
```

More Job Submission Options

Specifying Rerunnable or not

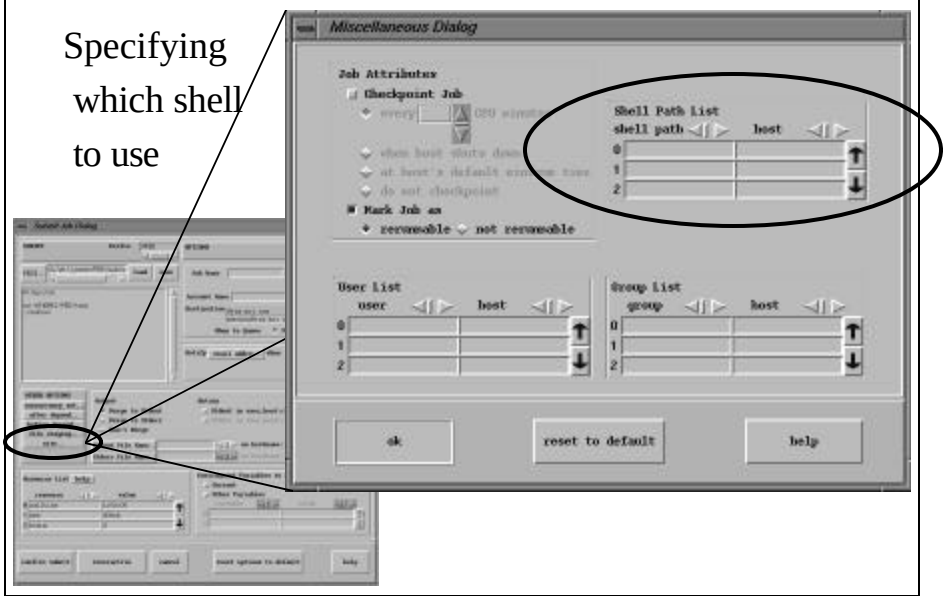


More Job Submission Options

- Specifying which shell to use
  - "-S path_list" - declares the shell that interprets the job script. Users can specify to use any shell that is installed on the system

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -S /usr/local/pkg/tcsh

cd ${HOME}/PBS/test
./subrun
```

More Job Submission Options



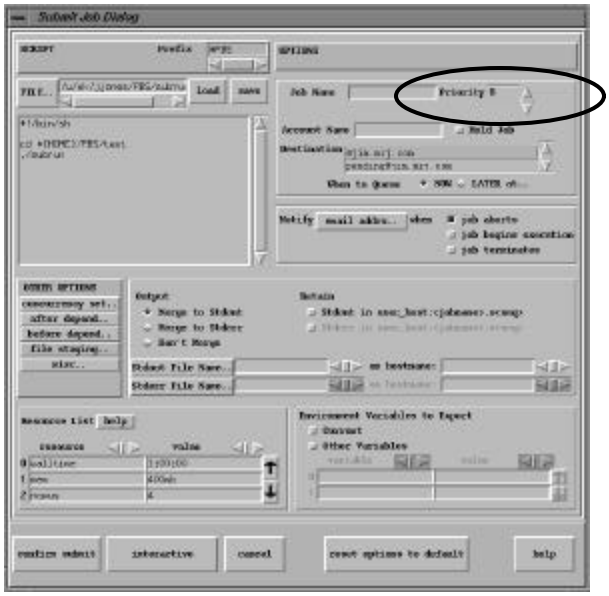
More Job Submission Options

- Setting a job's priority
  - "-p priority" - defines the priority of the job. Value must be between -1023 and +1023 inclusive.

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -p 400

cd ${HOME}/PBS/test
./subrun
```

More Job Submission Options

Setting a job's priority



More Job Submission Options

- Deferring execution
  - "-a date-time" - declares the time after which the job is eligible for execution

Time in 24-hour clock

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb
#PBS -l ncpus=4
#PBS -a 10220700

cd ${HOME}/PBS/test
./subrun
```

= Oct. 22 7:00 am

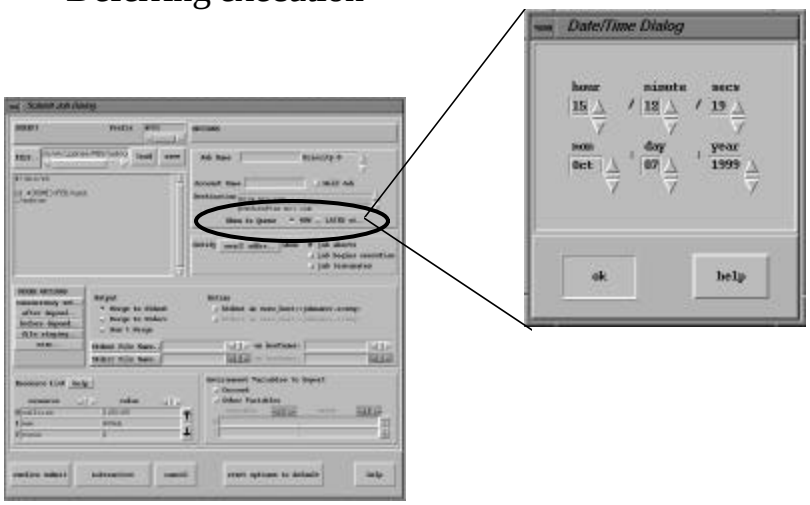
Format:
[[[[CC]YY]MM]DD]hhmm.s

Century, Year, Month, and Day are all optional.

Date is evaluated right to left.

More Job Submission Options

- Deferring execution



More Job Submission Options

- Holding a job (delaying execution)
  - "-h" - specifies a user hold be applied to the job at the time of submission

```
% qsub -h subrun.pbs
16393.origin.pbspro.com
```

More Job Submission Options

Holding a job (delaying execution)

More Job Submission Options

- Using a “here document” with qsub...

A simple example...

```
% qsub
date
hostname
myprogram
^D
34463.origin.pbspro.com
%
```

Job Dependencies

- Defining Job dependencies
  - "-W depend=*dependency-list*" where dependency list is one of:

Concurrent Dependencies	• synccount:count • syncwith:jobid • on:count
After Dependencies	• after:jobid[:jobid…] • afterok:jobid[jobid…] • afternotok:jobid[jobid…] • afterany:jobid[jobid…]
Before Dependencies	• before:jobid[:jobid…] • beforeok:jobid[jobid…] • beforenotok:jobid[jobid…] • beforeany:jobid[jobid…]

Job Dependencies

- Defining Job dependencies, most common uses...

Job 3 should start ***after*** jobs 1 and 2 have ***ended***:

```
% qsub job1.pbs
16394.origin.pbspro.com
% qsub job2.pbs
16395.origin.pbspro.com
% qsub -W depend=afterany:16394:16395 job3.pbs
16396.origin.pbspro.com
```

Job 2 should start only if job 1 ends with no errors (its shell returns a no-error status):

```
% qsub job1.pbs
16397.origin.pbspro.com
% qsub -W depend=afterok:16397 job2.pbs
16398.origin.pbspro.com
```

Job Dependencies

- Defining Job dependencies, less common uses...

Job 3 should start after jobs 1 and 2 have ***started***:

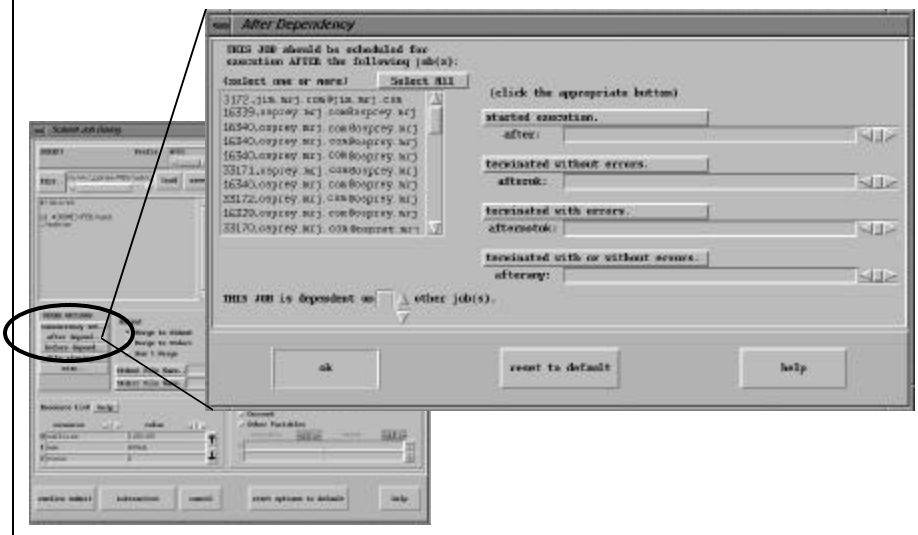
```
% qsub job1.pbs
16399.origin.pbspro.com
% qsub job2.pbs
16400.origin.pbspro.com
% qsub -W depend=after:16399:16400 job3.pbs
16401.origin.pbspro.com
```

Jobs 1, 2, and 3 should run concurrently:

```
% qsub -W depend=synccount:2 job1.pbs
16402.origin.pbspro.com
% qsub -W depend=syncwith:16402 job2.pbs
16403.origin.pbspro.com
% qsub -W depend=syncwith:16403 job2.pbs
16404.origin.pbspro.com
```


Job Dependencies

Defining Job (after) dependencies



File Staging

- What is “file staging”

  File staging is a way to specify which files should be copied onto the execution host before the job starts, and which should be copied off the execution host when it completes.

  This is most relevant in a metasystem environment where you don't know in advance where the eventual execution host will be. It is inconvenient to keep (and synchronize) multiple copies of data on all potential execution hosts, so jobs must expect to perform staging of some kind. (File staging is not required if the remote files are available on a globally shared file system.)

  One strategy is to perform remote copies as part of the job script, but the time needed to do the transfers is somewhat unpredictable and is charged against any wall clock limit for the job. PBS stagein/stageout is not charged against job wall clock limits.

File Staging

The formats allow multiple file specifications per command:

```
#PBS -W stagein=filespec1,filespec2,...
#PBS -W stageout=filespec1,filespec2,…
```

but each spec is a long enough string that it can go on a line of its own:

```
#PBS -W stagein=filespec1
#PBS -W stagein=filespec2
```

In each case, a file specification includes a 'local' filename on the execution host, and a 'remote' file on some remote host. An @ symbol is the pivot point separating the two.

```
#PBS -W stagein=LocalFile@RemoteHost:RemoteFile
```

LocalFile = the file on the local (execution) host

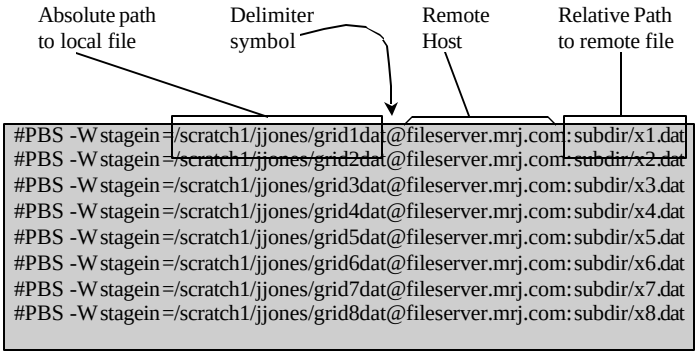
@ = delimiter symbol

RemoteHost:RemoteFile = the file on a remote host

File Staging

Either one of the filenames may include an absolute path, or a relative path with respect to the user's home directory.

Example:



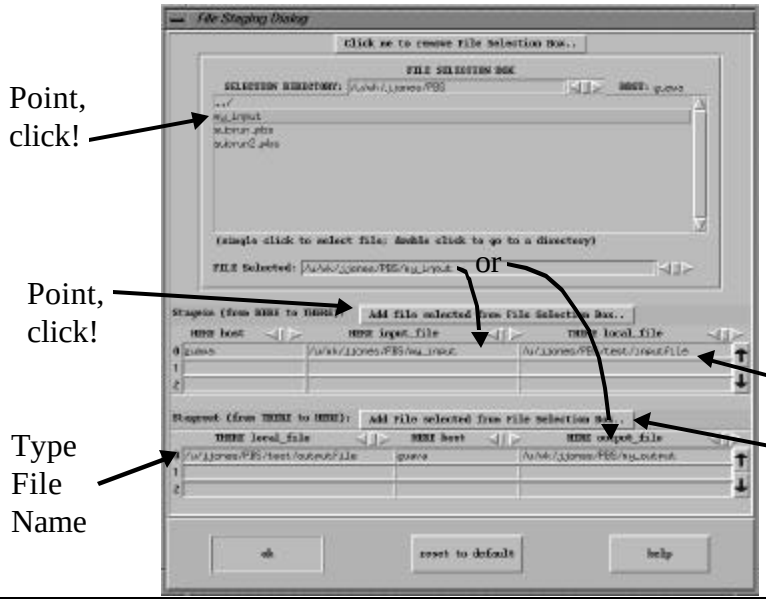
Files staged onto an execution host are deleted from the execution host at the end of the job.

File Staging

- Using xPBS for file staging is easier!

Point, Click!

File Staging with xPBS



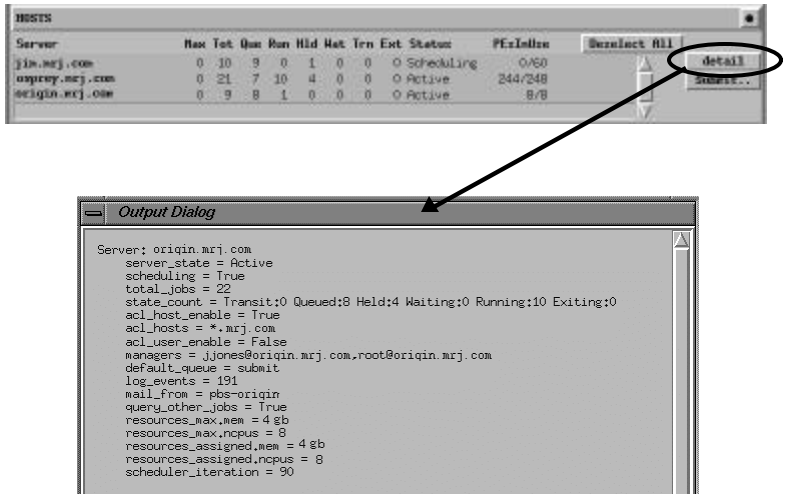
Job Monitoring Part II

- Requesting batch server status in brief and in full
  - "qstat -B" - displays brief summary of default server
  - "qstat -Bf" - displays full summary of default server

```
% qstat -B
Server           Max Tot Que Run Hld Wat Trn Ext Status
---------------- --- --- --- --- --- --- --- --- ----------
origin.mrj.com     0  10   4   6   0   0   0   0 Active

% qstat -Bf
[ full display, as shown on next slide ]
```

Job Monitoring Part II

- Requesting batch server status (brief and full) with xpbs



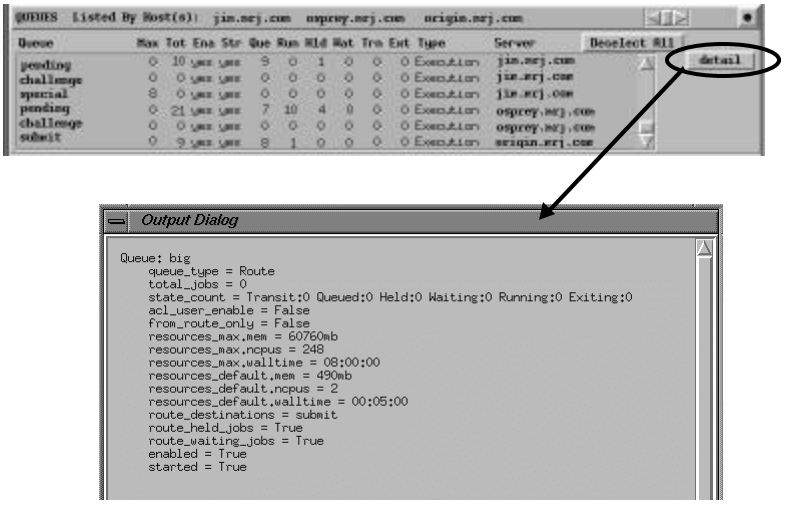
Job Tracking Part II

- Requesting queue status in brief and in full
  - "qstat -Q" - displays brief summary of queues
  - "qstat -Qf" - displays detail queue information

```
% qstat -Q
Queue              Max Tot Ena Str Que Run Hld Wat Trn Ext Type
---------------- --- --- --- --- --- --- --- --- --- --- ----------
submit             0   4 yes yes   4   0   0   0   0   0 Execution
special            0   0 yes yes   0   0   0   0   0   0 Execution
challenge          0   0 yes yes   0   0   0   0   0   0 Execution
background         0   0 yes yes   0   0   0   0   0   0 Execution

% qstat -Qf submit
[ full display, as shown on next slide ]
```

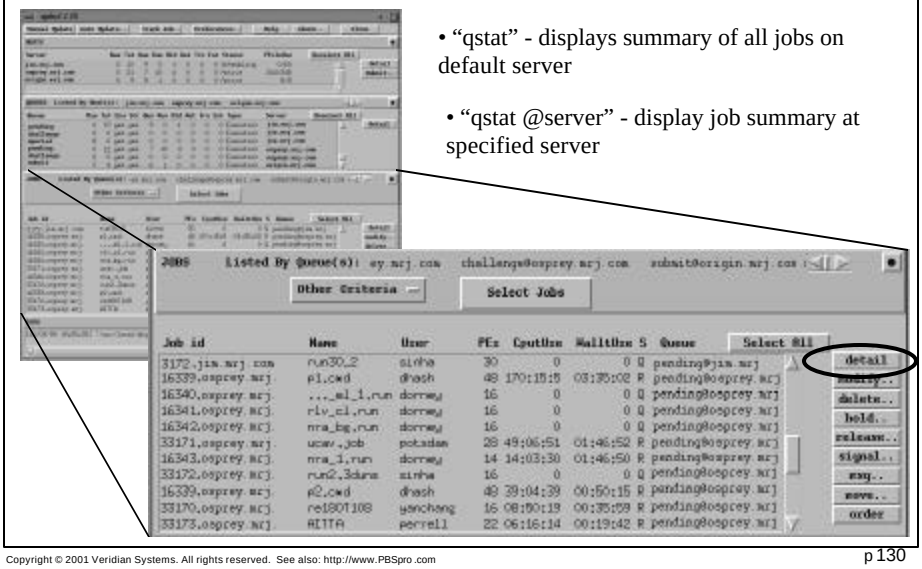
Job Tracking Part II

- Requesting queue status (brief and full) with xpbs



Job Tracking Part II

- Requesting all jobs on default or specific server



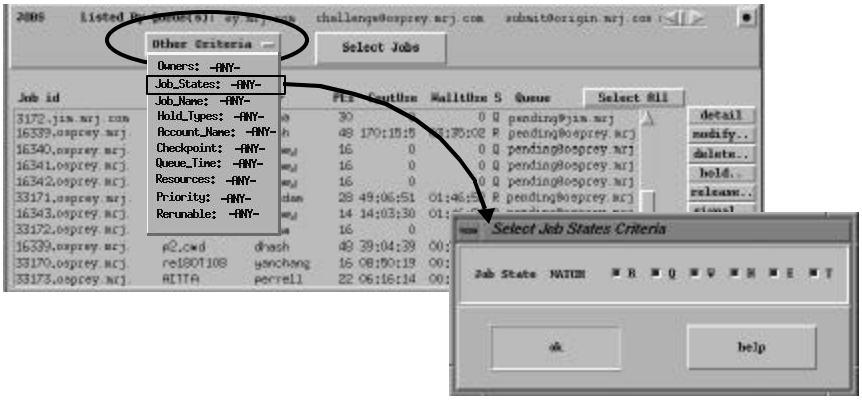
- "qstat" - displays summary of all jobs on default server
- "qstat @server" - display job summary at specified server

Job Tracking Part II

- Requesting full status of all jobs or specific job
  - "qstat -f jobid" - display full status info for specified job

Job Tracking Part II

- Displaying all running jobs on server or specific queue
  - "qstat -r" - displays all running jobs on server
  - "qstat -r -q submit " - displays all running jobs in queue "submit"



Job Tracking Part II

- Displaying memory resources in GB for all queues or servers
  - "qstat -G …" - Show size information in giga-bytes
- Displaying memory resources in MW for all queues or servers
  - "qstat -M …" - Show size information in mega-words
- Note: because xpbs as a fixed job status window, it is not currently possible to display GB/MW in the jobs status window of xpbs

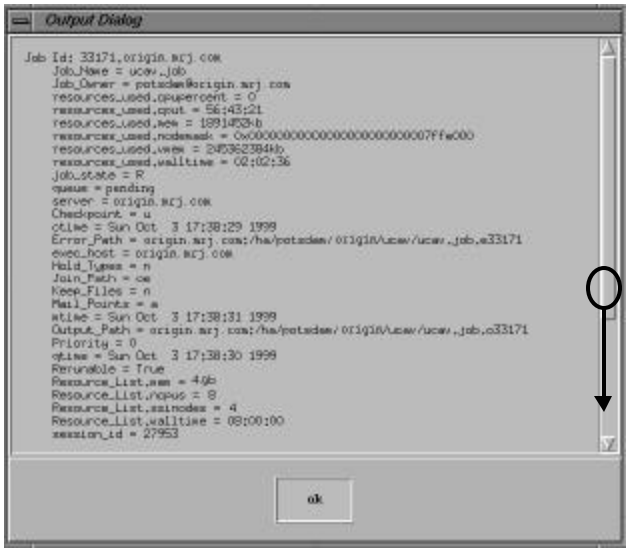
Job Tracking Part II

- Finding out why a particular job is not running
  - "qstat -s jobid" - displays the status comment for the specified job

```
% qstat -s 16387
                                                          Req'd      Req'd   Elap
Job ID       Username Queue    Jobname    SessID  TSK Memory  Nds wallt S wallt
------------ -------- -------- ---------- ------- --- ------- --- ----- - -----
16387.origin jjones   submit   mysubrun     10806   4   400mb   2 01:00 Q 01:16
    Insufficient cpu resources available to run job.
```

  - Under xPBS, use the "Detailed Job Display" to see the job status information (discussed earlier and below).

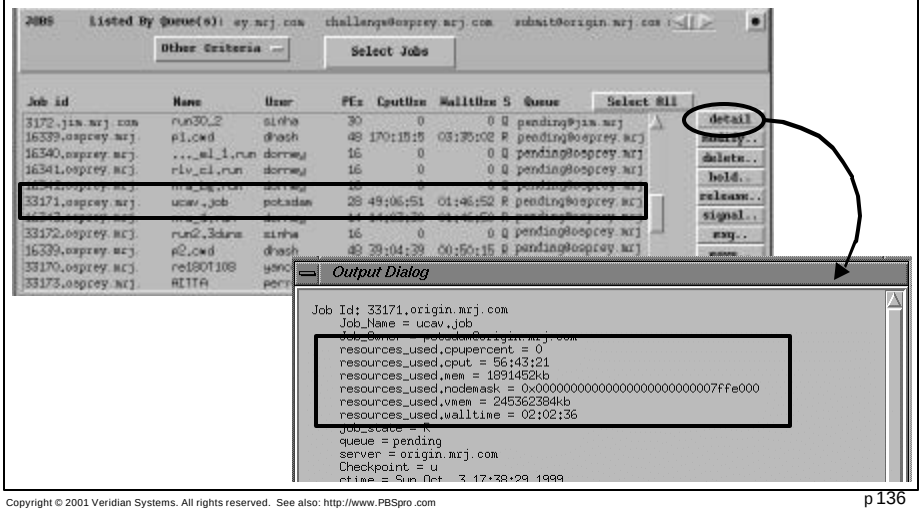
Job Tracking Part II



Scroll down to view "comment" field. Will contain job status information: when started or why not running.

Job Tracking Part II

- Where to find a summary of resources used by a specific job



Modifying Job Attributes

- Why Alter a Job's Attributes?
  - Mistake on resource requirements
  - Previous job failed, so you want to change a job attribute
  - Previous job ran out of time, so you want to alter a currently queued job (queued jobs only-- not running)
  - Specified the wrong stage-in/-out files, need to change it

Modifying Job Attributes

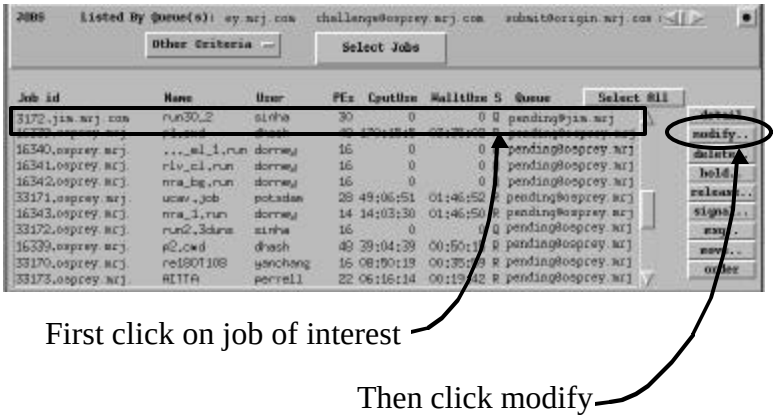
- What attributes can/cannot be changed
  - Most attributes can be changed by the owner of the job while the job is still queued.
  - However, once a job begins execution, the resource limits cannot be changed, including:
    - cputime
    - walltime
    - number of cpus
    - memory

Modifying Job Attributes

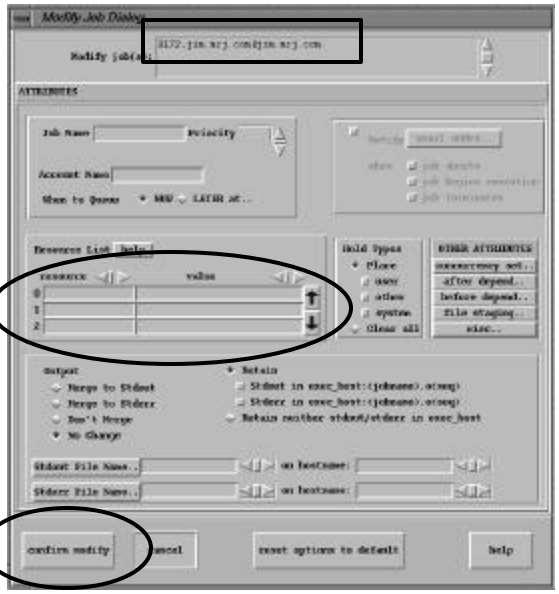
- How to modify job attributes
  - "qalter job-resources job-list" - modifies the specified job-resources for the specified job(s)

```
% qalter -l ncpus=16 16387
```


Modifying Job Attributes



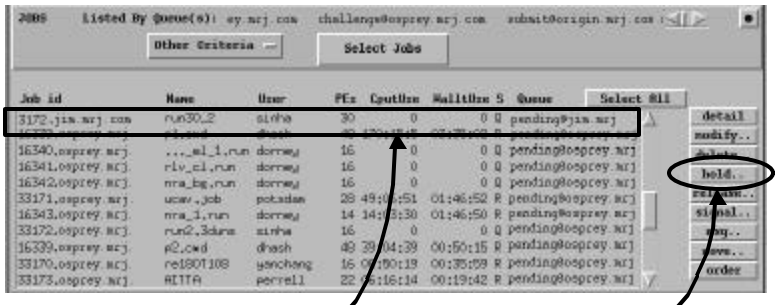
Modifying Job Attributes



Holding and Releasing Jobs

- Putting a job on hold
  - "qhold jobid" - places a user hold on a queued job

```
% qhold 3172
```



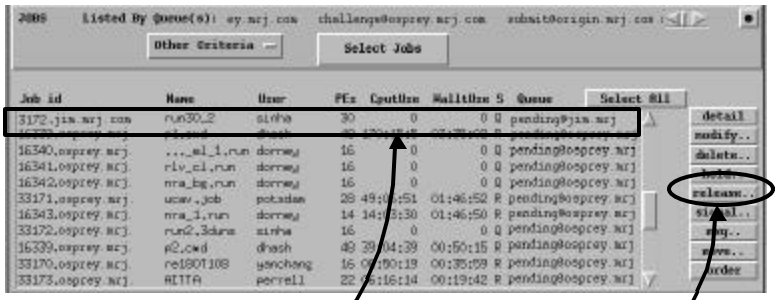
First click on job of interest

Then click hold

Holding and Releasing Jobs

- Releasing a held job
  - "qrls jobid" - releases the hold from the specified job

```
% qrls 3172
```



First click on job of interest

Then click release

Sending Messages to PBS Jobs

- When sending a message comes in handy
  - When you want external events recorded in the batch job
  - Useful to administrators (such as for writing a notification message to running jobs that some system failure, like a failed disk drive, has occurred, and may affect the job).
  - Also useful for absent-minded user to remind themselves of spouse's birthday (say, via crontab)

Sending messages to PBS Jobs

- How to send a message to PBS job
  - "qmsg -E message jobid" - writes message into error file of the specified job(s)
  - "qmsg -O message jobid" - writes message into output file of the specified job(s)

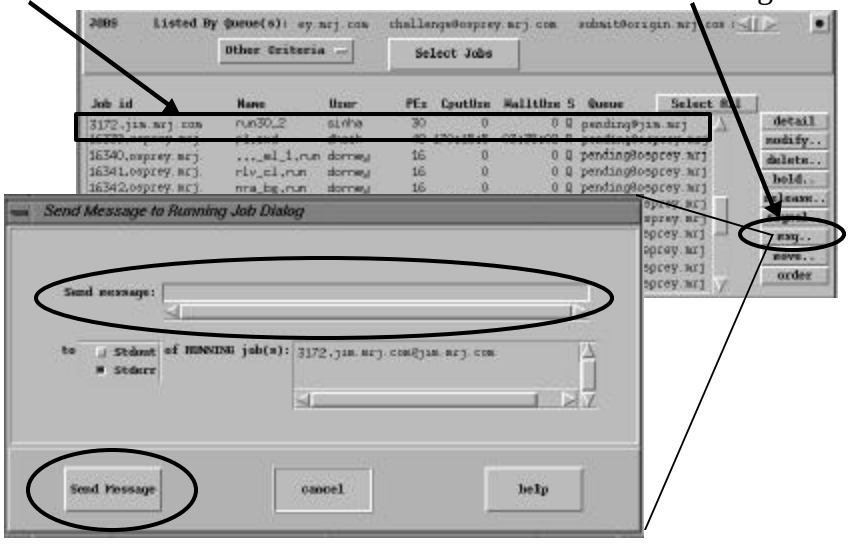
```
% qmsg -O 'this will appear in the job .o file' 16387
```

```
% qmsg -E 'this will appear in the job .e file' 16387
```


Sending messages to PBS Jobs

First click on job of interest

Then click msg



Changing Order of Jobs within Queue

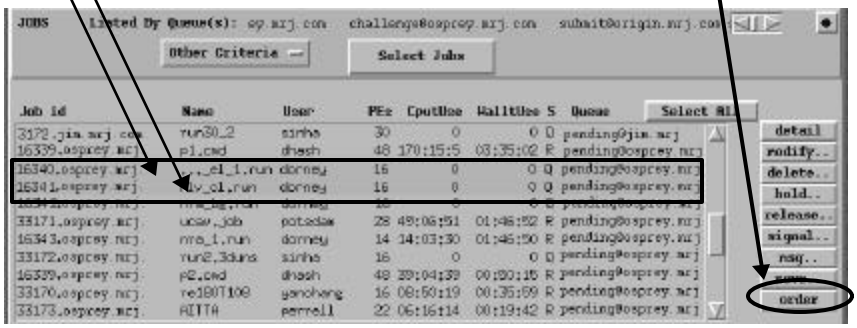
- Requirements for changing queue order of jobs
  - Jobs must be queued not running
  - Can only change order of your own jobs
- "qorder jobid1 jobid2" to swap the order of two jobs within the same queue

```
% qorder 16340 16341
```


Changing Order of Jobs within Queue

First click on jobs of interest

Then click order



Moving Jobs Around

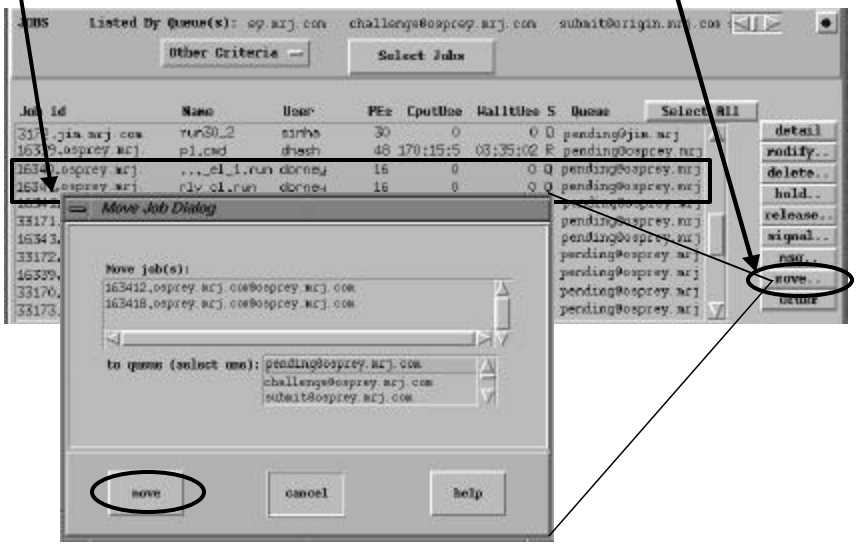
- Requirements for moving jobs
  - Jobs must be queued, not running
  - Can only move your own jobs
- How to move a job from one queue to another
  - "qmove queue-name job-list" - moves the specified jobs into the named queue

```
% qmove special 16341
```


Moving Jobs Around

First click on job(s) of interest

Then click move



Sending Signals to a PBS Job

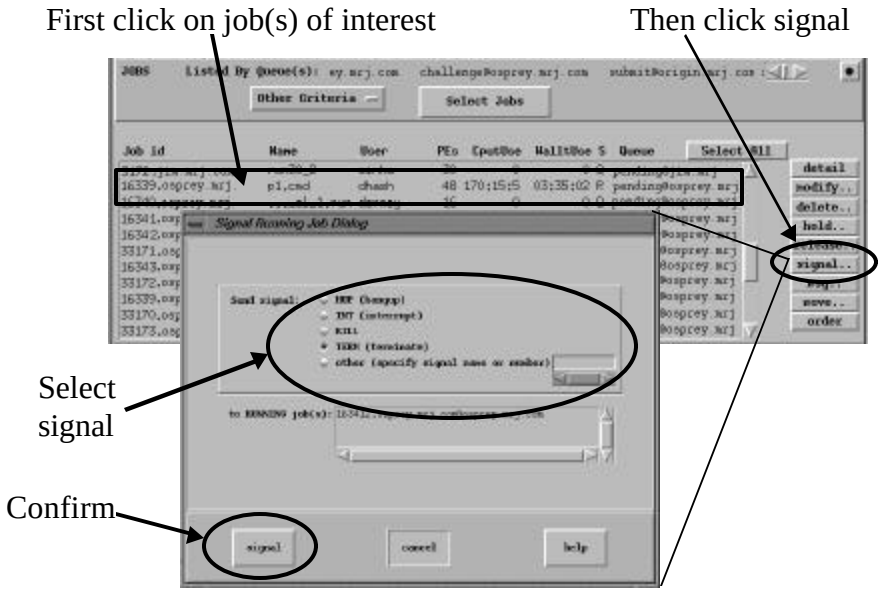
- Why send a signal?
  - To force a program to take specific action. The default action upon receiving a signal is to terminate the program. However, jobs can trap most signals, and take action based on the signal received.
- What signals are available?
  - Common signals:
    - HUP - hangup (often used by program to request reinitialization)
    - TERM - terminate (when program feels like it)
    - INT - interrupt (when program gets good and ready)
    - KILL - kill (now, regardless of how program feels about it)
    - USR1, USR2 - signals reserved for use by the end user
  - Less Common signals:
    - QUIT ILL TRAP ABRT EMT FPE BUS SEGV SYS PIPE ALRM USR1 USR2 CHLD PWR WINCH URG POLL STOP TSTP CONT

Sending Signals to a PBS Job

- Some things to remember about signals:
  - Can only signal your own jobs
  - Can only signal running jobs
- How to send signals to PBS jobs
  - “qsig -s signal jobid” - requests the specified signal be sent to the indicated job(s). The “signal” can either be the name of the signal, or its corresponding unsigned number.

```
% qsig  -s USR1 16341

% qsig  -s 16   16341
```

Sending Signals to a PBS Job



Defining what jobs to Display

- Rather than displaying information about all jobs or a single job, it is often useful to display jobs matching specific characteristics
- How to select jobs based on characteristics
  - "qselect characteristic(s)" - returns a list of jobIDs that match the characteristics specified
  - Example: list job that are in a "hold" state:

```
% qselect -s H
16330.osprey.pbspro.com
16312.osprey.pbspro.com
16311.osprey.pbspro.com
```

Defining what jobs to Display

- Using “qselect” to specify job characteristics
  - “-a .OP.date-time”
  - “-h holdlist”
  - “-l resourcelist.OP.value
  - “-N name”
  - “-p .OP.priority”
  - “-q destination”
  - “-r rerun”
  - “-s states”
  - “-u userlist”

“.OP.” refers to a comparison operator comparison. If not specified, equal (“.eq.”) is assumed. The available operators are:

- .eq. - equal
- .ne. - not equal
- .ge. - greater than or equal
- .gt. - greater than
- .le. - less than or equal
- .lt. - less than

Ex. Select jobs requesting 1 gb of memory or more:

```
% qselect -l mem.ge.1gb
16333.osprey.pbspro.com
16313.osprey.pbspro.com
```

Ex. Select running jobs (state R) that requested fewer than 16 cpus:

```
% qselect -s R -l ncpus.lt.16
16328.osprey.pbspro.com
16313.osprey.pbspro.com
```


Defining what jobs to display

- ...But a list of jobIDs is not exactly enlightening.
- Most useful to hand the list of jobIDs to qstat for display:

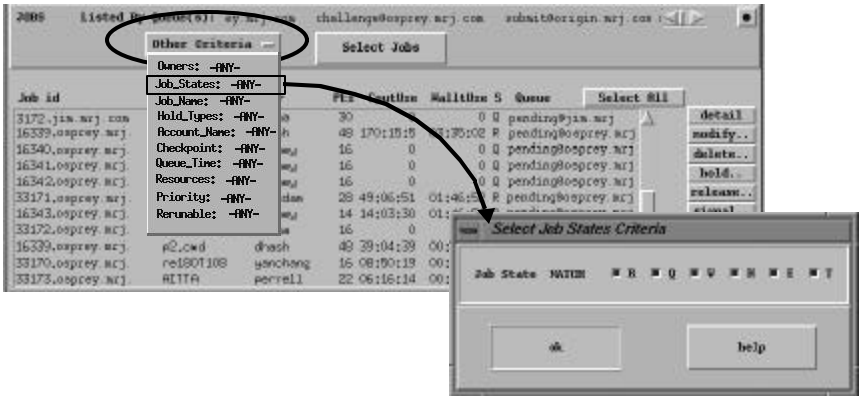
Command between “backquotes” is executed, result is passed to qstat

```
% qstat `qselect -l mem.ge.1gb`
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
16333.osprey     runfull          rogers           00:04:39 Q submit
16313.osprey     pbsjob           terryn           00:02:21 R pending
```

```
% qstat -a `qselect -s R -l ncpus.lt.16`
                                                            Req'd      Req'd   Elap
Job ID       Username Queue    Jobname    SessID  TSK Memory  Nds wallt S wallt
------------ -------- -------- ---------- ------- --- ------- --- ----- - -----
16328.origin jjones   submit   mysubrun     10806   4   400mb   2 01:00 R 00:06
16313.origin jjones   submit   mysubrun     10126   4   400mb   2 01:00 R 00:06
```

Defining what job to display...

- Use the Other Criteria selection box to specify all the criteria that describe the jobs you are interested in seeing displayed.



Deleting PBS Jobs

- Users: You can only delete your own jobs no matter how hard you try (but operators and managers can delete any job)

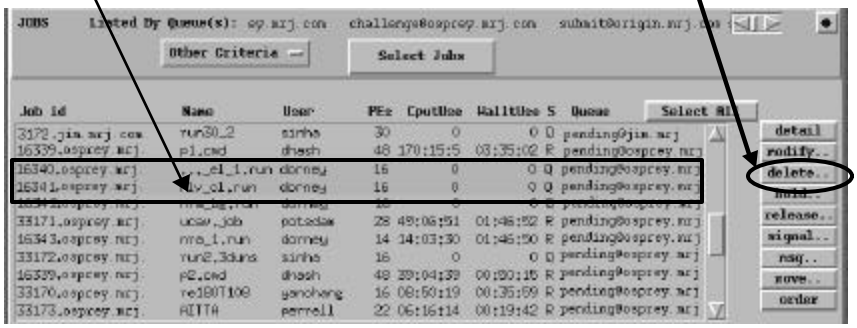
- How to delete jobs
  - "qdel jobid(s)" - deletes the specified job(s)

```
% qdel 16333
```

Deleting PBS Jobs

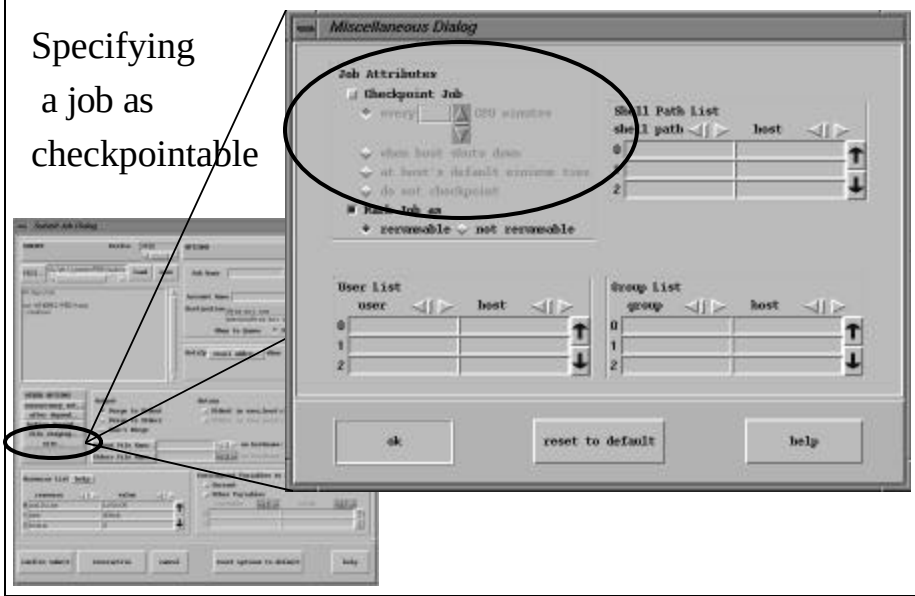
First click on job(s) of interest

Then click delete



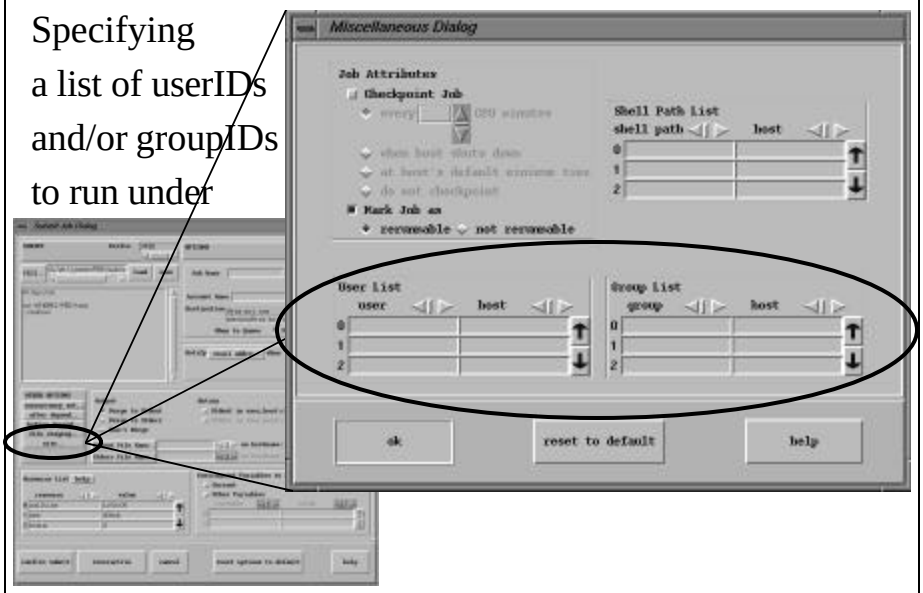
More Job Submission Options...

Specifying a job as checkpointable



More Job Submission Options...

Specifying
a list of userIDs
and/or groupIDs
to run under



What are "Advance Reservations"?

- In the real-world
  - "I'd like to fly to Chicago for Thanksgiving…"
  - Airline, hotel, rental car, dinner
- Issues
  - cost, guarantee
- In the computer world…
  - ad hoc, manual process

Using PBS Reservations

Reserve {R} for time T

- pbs_rsub -R 1400 -E 1600 -l nodes=4
  - Make a reservation (2-4 pm for 4 nodes)
  - returns reservation/queue id, e.g., “R1234”
- pbs_rdel R1234 / pbs_rstat
  - Delete a reservation, query status
- qsub -q R1234 myjob
  - Standard qsub interface to submit jobs to run under a reservation

System Specific Environments

- How to submit PBS jobs on a multi-node cluster:

A sample script...

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l nodes=4
#PBS -j oe

cd ${HOME}/PBS/test
./subrun
```

A sample submit line:

```
% qsub -l nodes=16 mysubrun
```

System Specific Environments

- How to submit PBS jobs on an SMP System:

A sample script...

```
#!/bin/sh
#PBS -l cput=1:00:00
#PBS -l ncpus=4
#PBS -j oe

cd ${HOME}/PBS/test
./subrun
```

A sample submit line:

```
% qsub -l ncpus=16 mysubrun
```


Advanced PBS Administration

- This section returns to PBS Administration:
  - advanced topics (security, log files, ...
  - day-to-day administration of PBS
  - using the xpbsmon Graphical Interface
  - using the manager options of xpbs
  - trouble-shooting and problem resolution

Security within PBS: Overview

- There are three parts to security in PBS:
  - Internal Security
    - Can the Daemons be trusted?
  - Authentication
    - How do we believe a client about who it is?
  - Authorization
    - Is a client entitled to have the requested action performed?

Security within PBS: Internal Security

- Much effort has been made to insure the various PBS daemons themselves cannot be a target of opportunity in an attack on the system.
- Security of the daemons' environments
  - Each daemon resets its environment when it starts
- Internal Daemon security checks
  - verifies existence of required directories
  - verifies permissions of PBS directories and files
  - verifies ownership of PBS directories and files
  - Daemon startup will fail if problem are discovered
  - Check daemon logs for info on failed security checks

Security within PBS: Host Authentication

- PBS uses a combination of information to authenticate a host:
  - A client request from a privileged port is trusted
  - Otherwise, the name of the host which is making a client request must be in the credential included with the request, AND it must match the IP network layer opinion as to the host's identity

Security within PBS: Host Authorization

- Access to the pbs_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 each MOM's configuration file. By default, only "localhost" and the name returned by gethostbyname(2) are allowed.
- Access to pbs_sched is not limited other than it must be from a privileged port.

Security within PBS: User Authentication

- *Is the user who s/he claims to be?*
- The PBS Server authenticates the user name included in a request with the supplied PBS credential.
- The credential is (by default) created by the PBS Interface Facility (pbs_iff) a "setuid root" program that sends a user's true identity and hostname.
- This security method is modular and can be replaced with other authentication methods such as Kerberos (with a bit of work…).

Security within PBS: User Authorization

- *Is the user entitled to make the request of the Server under that name?*
- PBS (as shipped) assumes consistent user names within a set of systems managed by PBS. Thus PBS compares the username of request to the username of the owner of a job. If they match, the request is approved.
- PBS also provides the ability to translate usernames between different systems, if a consistent user name space is not available.

Security within PBS: Command Privilege

- What are privileged and unprivileged commands?
- Most commands in PBS are unprivileged-- any person with an account on the machine can run the command (but authorization still applied)
- Some commands are “privileged” -- they require special authorization to run

Security within PBS: Levels of Privilege

- There are three levels of privilege for PBS commands:
  - ***user***
    - qalter, qdel, qhold, qmove, qmsg, qrerun, qrls, qsig, qstat
    - can only be used to affect jobs owned by that user
    - qstat - can be configured to display only the user’s jobs or all jobs
  - ***operator***
    - all commands above with added privilege (able to act on any job)
    - qdisable, qenable, qrun, qstart, qstop, qterm
    - pbs_mom, pbs_sched, pbs_server, pbsnodes
  - ***administrator***
    - all commands above with full privilege
    - qmgr

Security within PBS: Root Owned Jobs

- PBS by default will reject any job which would execute under the UID of zero (typically root)
- To change this behavior, add the owner of the job (typically root) to the server attribute acl_roots

```
qmgr: set server acl_roots += root
```


Job Prologue & Epilogue Scripts

- PBS provides the ability to run a site supplied script/program before and/or after each job runs
- Provides the capability to perform initialization or cleanup of resources, such as temporary directories or scratch files; or to write “banners” or “resource usage reports”.
- In a multi-node job, the script is run only on the first node allocated (Mother Superior)

Job Prologue & Epilogue Scripts

- If present, the prologue/epilogue is run with root privilege.
- But prologue/epilogue must adhere to specific rules:
  - must be in the PBS_HOME/mom_priv directory with the name “prologue” or “epilogue”
  - must be owned by root
  - must be readable and executable by root
  - cannot be writable by anyone but root

Job Prologue & Epilogue Arguments

- The prologue is invoked with the following arguments:
  - argv[1] is the job id
  - argv[2] is the user name under which the job executes
  - argv[3] is the group name under which the job executes
- The epilogue is called with the above, plus:
  - argv[4] is the job name
  - argv[5] is the session id
  - argv[6] is the requested resource limits (list)
  - argv[7] is the list of resources used
  - argv[8] is the name of the queue in which the job resides
  - argv[9] is the account string, if one exists

Job Prologue & Epilogue Environment

- For both the prologue and epilogue:
  - **envp** The environment passed to the script/program is null
  - **cwd** The current working directory is the user's home directory
  - **input** When invoked, both scripts/programs 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 file which contains 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

Alternate Version/Test System

- 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.
- The daemon home directories must be pre-built. The easiest method is to alter the PBS_HOME variable by use of the **--set-server-home** option to configure, rerun configure and remake PBS.

Alternate Version/Test System

- The following commands will start the three daemons with a home directory of /tmp/altpbs and four port around 13001, the Server on 13001, MOM on 13002 and 13003, and the Scheduler on 13004:

```
pbs_server -t create -d /tmp/altpbs -p 13001 -M 13002 -R 13003 -S 13004
pbs_mom -d /tmp/altpbs -M 13002 -R 13003
pbs_sched -d /tmp/altpbs -S 13004 -r script_file
```


Alternate Version/Test System

- Jobs may be directed to the test system by using the **`server:port`** syntax on the **`-q`** option.
- 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 job
qstat -Bf host:13001
qstat @host:13001
```

Installing an Updated Batch System

- Eventually, you will want to upgrade to a newer version of PBS.
- It is recommended that you first follow the above procedures (testing an alternate version) to test the new version in your environment.
- The PBS Admin Guide (see section “Installing an Updated Batch System”) details the full procedure for migrating to a new version of PBS.

Installing an Updated Batch System

1. With the old batch system running, disable the queues and stop scheduling by setting “scheduling=false”.

2. Backup the pool of jobs in PBS_HOME(old)/server_priv/jobs Tar may used for this.

Assuming the change is a minor update (change in third digit of the release version number) or a local change where the job structure did not change from the old version to the new, it is likely that you could start the new system in the old HOME and all jobs would be recovered.

However 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 if the job structure has changed or the move is required for other reasons.

To move the jobs, continue with the following steps...

Installing an Updated Batch System

3. Start the new PBS Server in its new home. If the new home is different from the directory when it was compiled, use the -d option. Use the -t option if the Server has not been configured for the new directory. Also start with an alternative port using the -p option. Turn off attempts to schedule with the -a option:

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

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. Remember, you will need to use the **:port** syntax when commanding the new Server.

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


Installing an Updated Batch System

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 jobid(s)
# 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
...
```

Installing an Updated Batch System

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 and move everything to the real home using the following to copy just the jobs:

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

or, if the real (new) home is already set up:

```
# cd new_homeserver_priv/jobs
# cp * real_homeserver_priv/jobs
```

At this point, you are ready to bring up and enable the new batch system.

Installing an Updated 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 queue@host 123.job.host:13001
```

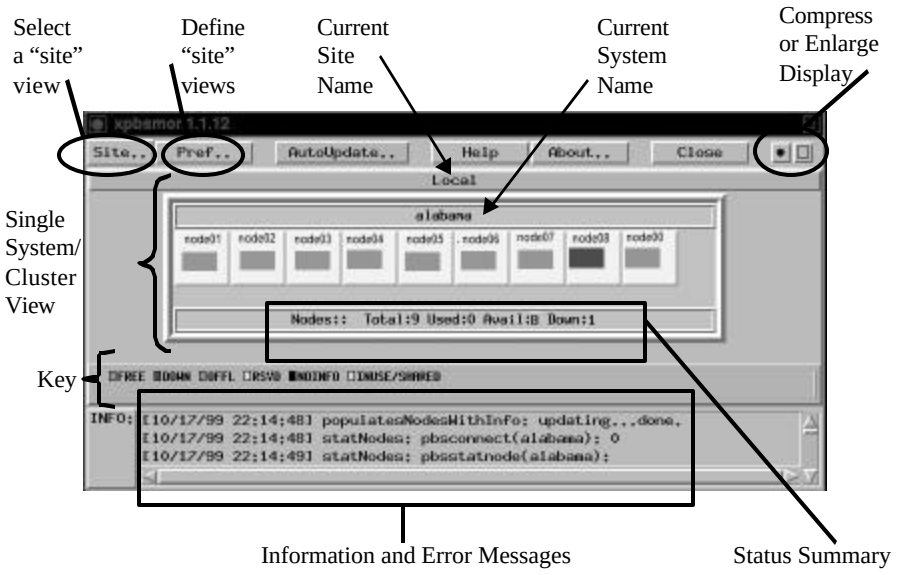
However, that will only move the job to the end of the queue it is already in. The Server receiving the move request (13001), will compare the destination server name, host, with its own name only, not including the port. Hence it will match and it will not send the job where you intended. To get the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

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

xPBSmon: a PBS monitoring tool

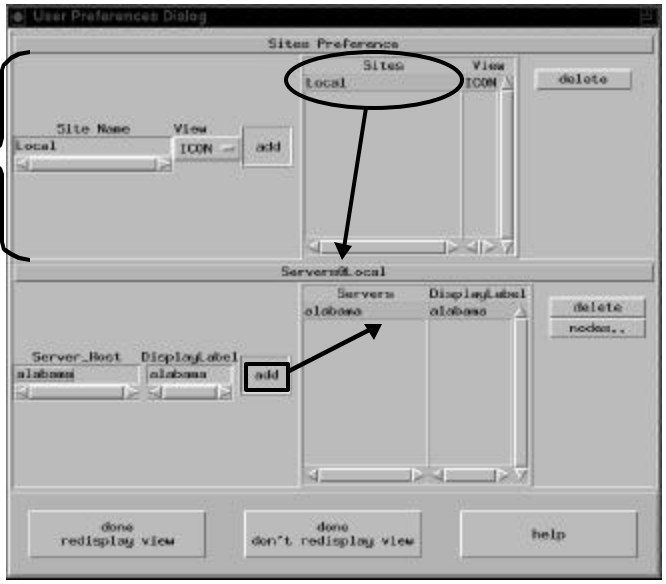
- Introducing xpbsmon, a graphical monitoring tool
- Used for:
  - Visually monitoring individual machines, clusters of machines, entire sites, or multiple sites
  - Reports node/host status (up/down/idle/inuse/offline)
  - Reports which jobs are running per node

xPBSmon: Tour of the GUI



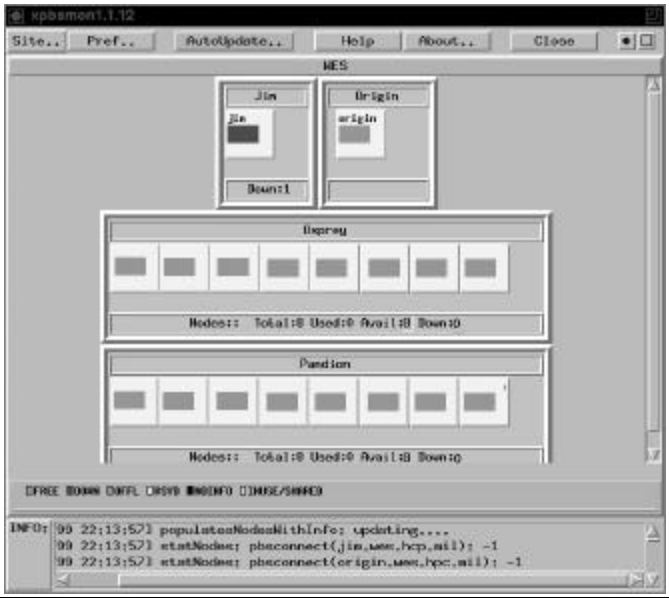
xPBSmon: Preferences/Customization

Add or delete "site" definitions

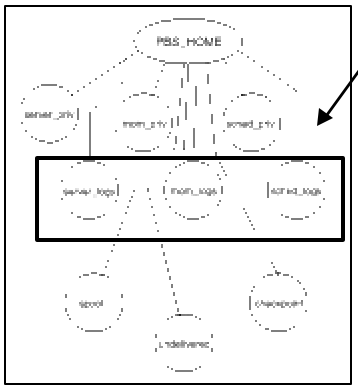


xPBSmon: Multi-system Display

Sample "WES" site display showing two standalone systems and two cluster systems



The PBS Log Files: The Daemon Logs



The daemon log files can be found under the PBS filetree on the system where the PBS Server runs. Exception: there will be a MOM log file under the PBS filetree on each execution host.

The log files are created daily and named by date in the form:
yyyymmdd

The PBS Log Files: The Daemon Logs

- How the PBS daemon log files are formatted

Date and time stamp:
mm/dd/yyyy hh:mm:ss

The name of the server which logged the message

The name of the specified object

date_time;event;server_name;object_type;object_name;message

The type of the event that triggered the log message:

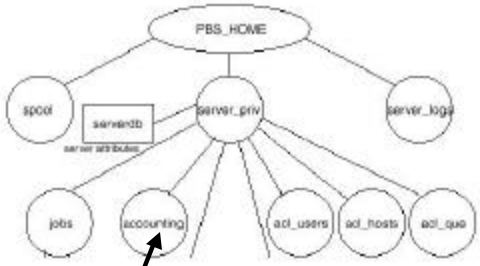
1 Internal PBS errors
2 System (OS) errors
4 Admin related events
8 Job related events
16 Job resource usage
32 Security related events
64 Scheduler related
128 common debug messages
256 rare debug messages

The type of object which the message is about:

Svr - server
Que - queue
Job - job
Req - request
Fil - file

The text of the log message

The PBS Log Files: The Accounting Log



The accounting log file can be found under the server_priv directory of the PBS filetree, and are created one per day, named in the format: yyyymmdd

The PBS Log Files: The Accounting Log

The PBS accounting log has a different format from the daemon log files:

Date and time stamp: mm/dd/yyyy hh:mm:ss

An ascii text string (whose content depends on the record type) in the format: blank separated keyword=value

Date time;record_type;job_id;message_text

A single character indicating the type of the record:

The job identifier

A - job was aborted by the server
D - job was deleted by request
E - job ended (terminated execution)
C - job was checkpointed and held
Q - job entered a queue
R - job was rerun
S - job execution started
T - job was restarted from a checkpoint file

The PBS Log Files

- How these files can be used in troubleshooting
  - If PBS detects an error it is reported to the corresponding daemon log file.
  - These logs can be used for post-mortem analysis and tracking down the cause of a job failure.
  - Can also be used to monitor the health of the system and to anticipate potential problems.

The PBS Log Files

- But that's a lot of files to look in for one job! How do I know where to start, and how do I find the message about a particular job?

- Good question! And the answer is…

The PBS Log Files: Tracejob

- The Tracejob utility
- Tracejob is used to extract all the messages about a particular job from all the available PBS log files, sort the entries by time order, and display the message in a "more readable" format.
- Usage is:

tracejob [-n days] jobid

An optional number of days back in time to look for the job information. Default is today only.

The job identifier of job that you are searching for.

The PBS Log Files: Tracejob

Search logs over past four days

Search for jobid 1400

```
jim% tracejob -n 4 1400
Job: 1400.origin.pbspro.com
10/16/1999 17:07:11  S    Job Queued at request of jctwang@origin.pbspro.com,
                          owner = jctwang@origin.pbspro.com, job name = shk_runA,
                          queue = workq
10/16/1999 17:07:12  S    Job Modified at request of Scheduler@origin.pbspro.com
10/16/1999 17:07:12  S    Job moved to jim at request of Scheduler@origin.pbspro.com
10/16/1999 17:07:12  S    Job Run at request of Scheduler@origin.pbspro.com
10/16/1999 17:07:14  M    Started, pid = 3234
10/16/1999 17:10:13  M    killm: sid=3234 sig=9
10/16/1999 17:10:13  M    task 1 terminated
10/16/1999 17:10:13  M    Terminated
10/16/1999 17:10:13  S    Exit_status=0 resources_used.cput=00:02:03
                          resources_used.mem=31676kb
                          resources_used.mppmem=1525184kb
                          resources_used.mppt=00:00:01 resources_used.sds=0kb
                          resources_used.walltime=00:03:00
10/16/1999 17:10:13  M    kill_job
10/16/1999 17:10:13  M    Obit sent
```

S = Server message

M = MOM message

Managing the Daemon Log Files

- The PBS logs can contain a LOT of messages. Refer to "Daemon Log Files" section of this class for controlling which events are written to each of the logs.
- The administrator can override the filename (automatic close/reopen) of the PBS event logs.
- Normally, the daemon event logs are written to PBS_HOME/(daemon)_logs/
- This location can be overridden with the "`-L pathname`" option to the daemons; pathname must be an absolute path.
- However, using the "-L" option disables the automatic close/reopen.

Managing the Daemon Log Files, cont.

- All daemons will close and reopen the same named log file on receipt of SIGHUP.
- The 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 PBS_HOME/daemon_logs
# mv current_file archive_file
# kill -HUP `cat ../daemon_priv/daemon.lock
```

Checking status of PBS Daemons

- What does a healthy system look like?
  - All the PBS daemons should be running
  - Qstat should return quickly with status information
  - Should be able to submit a job
  - Batch jobs should be running

Checking status of PBS Daemons

- What daemons should be running?

```
# ps -eaf |grep pbs_
   root 3428    691: /pbs/sbin/pbs_mom -r -c /pbs/mom_priv/config
   root 3429    6:40 /pbs/sbin/pbs_sched
   root 3430   20:32 /pbs/sbin/pbs_server
   root 1808    0:00 grep pbs_
```

Where should the daemons be running?

System	Server	MOM	Scheduler
frontend	x		x
node1		x	
node2		x	
node3		x	
node4		x	
...			
node64		x	
SMP	x	x	x

Checking status of PBS Daemons

- How to ask for server status
- A quick command to check the health of the PBS Server daemon is: `qstat -B`

```
% qstat -B
Server            Max Tot Que Run Hld Wat Trn Ext Status
---------------- --- --- --- --- --- --- --- --- ----------
origin.pbspro.com   0  10   4   6   0   0   0   0 Active
```

- Can also “tail” the server log file in order to watch for error messages to appear.

Checking status of PBS Daemons

- How to monitor MOM
- Perhaps the best way to monitor MOM is to use “tail” on the current log file. However, as with the server log, a LOT of messages are written to the log.
- And in cluster configurations this is not really practical.
- Therefore, xpbsmon is perhaps the best tool for monitoring the health of one or more MOMs

Troubleshooting PBS

- This section of the class discusses various problems or errors that the PBS Administrator may need to resolve.
- A Troubleshooting Index to this section is included in Appendix B.
- A list of PBS error codes is included in Appendix C.

Troubleshooting PBS

- File most useful in tracking down problems…
  - The Server log file
    - contains all the events for a given job from submission up until it is executed, and all messages after execution.
  - The MOM log file has the needed information for the period while the job was running

Troubleshooting PBS

- *Translating PBS error codes into English*
- All the PBS error codes are listed in a file (usually) installed as: /usr/pbs/include/pbs_error.h
- The quickest way to convert a PBS error code into something meaningful, is to grep the error code from the file:

```
% grep 15002 /usr/pbs/include/pbs_error.h
#define PBSE_NOATTR    15002    /* undefined Attribute */
```


Troubleshooting PBS

- Client commands unable to connect to server
- Probable Causes:
  - client (or client host) not permitted due to ACLs
  - network could be down
  - server could be down
- Verify that the network path between the client and the server is up and functioning
- Check that the server is indeed running
- Check the server log file for messages or errors

Troubleshooting PBS

- Daemons unable to connect to MOM
- Probable Causes:
  - Daemon (or daemon host) not permitted to connect
  - network could be down
  - MOM could be down
- Verify that the daemon host has a *$clienthost* entry in the MOM config file
- Verify that the network path between the daemon and the MOM is up and functioning
- Check that the MOM is indeed running
- Check the MOM log file for messages or errors

Troubleshooting PBS

- GUI (or other client commands) unable to query MOM
- Probable Causes:
  - client (or client host) not permitted to connect
  - network could be down
  - MOM could be down
- Verify that the client host has a *$restricted* entry in the MOM config file
- Verify that the network path between the client and the MOM is up and functioning
- Check that the MOM is indeed running
- Check the MOM log file for messages or errors

Troubleshooting PBS

- Server reports “no destination”
- This error only occurs when a user submits a job, without specifying a queue, to a PBS server where there is no default queue defined on the system.
- Solution:
  - Define a default queue via qmgr:
  - qmgr: set server default queue = xxxxxxx

Troubleshooting PBS

- Non Delivery of output
  - The failure to deliver output files can occur for a variety of reasons. In most cases, the reason or an error message is included in the email that PBS send the user. E.g.:

```
From: adm@origin.pbspro.com (Accounting Files Owner)          3:05 AM

 Subject: PBS JOB 18316.origin.pbspro.com
    To:    utley@sunfarm.pbspro.com

PBS Job Id: 18316.origin.pbspro.com
Job Name:   STDIN

Post job file processing error; job 18316.origin.pbspro.com on host sdf1
Unable to copy file 18316.sdf1..OU to fs1.pbspro.com:/pbs/STDIN.o18316
>>> error from copy
    rcp: /pbs/STDIN.o18316: Permission denied
>>> end error output
Output retained on that host in: /usr/spool/pbs/undelivered/18316.sdf1..OU
```

Troubleshooting PBS

- Job prologue or epilogue times out
- To prevent a bad script/program or error condition within the script from delaying PBS, MOM places an alarm around the scripts/program 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 by changing the define of **`PBS_PROLOG_TIME`** within `src/resmom/prolog.c`

Troubleshooting PBS

- Normally, the prologue script/program 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 writeble 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.

Troubleshooting PBS

- Interactive job will be aborted on any non-zero prologue exit
- The administrator must exercise great caution in setting up the prologue to prevent jobs from being flushed from the system.
- Epilogue script exit values are logged, if non-zero, but have no impact on the state of the job.

Troubleshooting PBS

- A job is rejected upon submission
- This error usually occurs for one of three reasons:
  - The user doesn't have access to the requested queue
  - The job resource requests exceed one or more queue or server limits
  - The user has exceeded their allocation (local site mod)

Troubleshooting PBS

- A job is rejected after submission
- Possible causes include:
  - The job requests resources that are not available on the system submitted to (such as asking for mppe on the SP).
  - The scheduler determines that it cannot ever run the job.
  - The scheduler determines that running the job would cause the user to exceed her/his allocation.

Troubleshooting PBS

- A job reports “hop count exceeded”
- This is a rare error and is usually indicative of a system configuration error. PBS keeps track of the number of queues that a given job moves through (or “hops”) before getting to a final destination. If this number exceeds an internal limit, the job is rejected. Typically this only happens when a routing queue has another routing queue as a destination which in turn points back to the first queue. The result is that the job bounces back and fourth until its hop count is exceeded and the job is rejected.

Troubleshooting PBS

- No jobs are running
- Several thing to check if you notice that there are no jobs running:
  - Are there any jobs queued?
  - Is the system scheduled down?
  - Is “server scheduling = true” in the Server?
  - Is the scheduler running?
  - Are there any error messages in any of the daemon logs?
  - Are the server and MOM daemons up and running?

Troubleshooting PBS

- Why is email coming from “adm”?
  - By default, PBS sends the email as Administrator or “adm”.
  - But PBS allows a site to define the “return-address” and “sender” of email sent to the user by PBS. This is so that if the user were to “reply” to the message, the email will go to a more useful address.
  - Use qmgr to change the “return-address” of PBS email:

```
% qmgr
Qmgr: set server mail_from = pbs-help
Qmgr: quit
%
```

Troubleshooting PBS

- How to allow/prevent users from viewing other user's jobs
- By default PBS allows all users to view other user's jobs via qstat, xpbs, etc.
- A site can change this on a per-server basis via qmgr:

```
% qmgr
Qmgr: set server query_other_jobs = true
Qmgr: quit
%
```

- Setting the value to "false" prevents users from seeing each other's jobs.

Troubleshooting PBS

- Most common user questions
  - Why isn't my job running?
  - Where did my job go?
  - Why did my job die?

Submitting Change Requests & Bug Reports

- The gnats **send-pr** software is used for sending problem reports about PBS. Most problem reports and change requests can be submitted using the web interface to send-pr:

  http://www.PBSpro.com/UserArea/send-pr.html

- However, given the nature of HTML forms, TABS and newlines are removed from the form input, making the submission of source code problematic. If you need to submit a problem report containing source code, it is recommended that you use send-pr directly from your site. A copy of send-pr, configured for use with PBS, is available for download from the above URL.

- Before using the online problem reporting system for the first time, please review the usage instructions on the above webpage.

Submitting Change Requests & Bug Reports

- When submitting a problem report or change request, it is important that you select a classification such that it will be routed to the correct developer (Category), and so that it will be handled with the appropriate urgency. In light of this, below are the definitions for the various levels of Severity and Priority.

Severity:

Critical - A problem which prevents the software from being used or has major impact on delivery of services on the associated system.

Serious - A problem which impacts the quality of service, but does not prevent its delivery of service. This includes situations where the software does not perform as specified and the actual behavior is unacceptable or random, infrequent appearances of a major problem.

Non-critical - A problem in the software that is inconvenient, but which does not greatly impact its use.

Priority:

High - Please address this problem before any other submitted by our site. It is also felt that it should be addressed before others at the same level of Severity.

Medium - This is important to our site and we need it to be addressed in a timely manner.

Low - Not a major issue, please address it when you have time.

Class survey

- To help us improve this class, please complete our online class survey:
- http://www.pbspro.com/training/critique.html
- Be sure to select the correct class name/identifier for your site.

Thank you for taking the time to complete the survey.

Appendix A: Common Errors and Solutions

Appendix B: PBS Error Codes

```
/*
 * The error returns possible to a Batch Request
 *
 * Each error is prefixed with the string PBSE_ for Portable (Posix)
 * Batch System Error.  The numeric values start with 15000 since the
 * POSIX Batch Extensions Working group is 1003.15
 */
#ifndef PBSE_
#define PBSE_           15000
#define PBSE_NONE       0               /* no error
#define PBSE_UNKJOBID   15001           /* Unknown Job Identifier
#define PBSE_NOATTR     15002           /* Undefined Attribute
#define PBSE_ATTRRO     15003           /* attempt to set READ ONLY attribute
#define PBSE_IVALREQ    15004           /* Invalid request
#define PBSE_UNKREQ     15005           /* Unknown batch request
#define PBSE_TOOMANY    15006           /* Too many submit retries
#define PBSE_PERM       15007           /* No permission
#define PBSE_BADHOST    15008           /* access from host not allowed
#define PBSE_JOBEXIST   15009           /* job already exists
#define PBSE_SYSTEM     15010           /* system error occurred
#define PBSE_INTERNAL   15011           /* internal server error occurred
#define PBSE_REGROUTE   15012           /* parent job of dependent in rte que
#define PBSE_UNKSIG     15013           /* unknown signal name
#define PBSE_BADATVAL   15014           /* bad attribute value
#define PBSE_MODATRRUN  15015           /* Cannot modify attrib in run state
#define PBSE_BADSTATE   15016           /* request invalid for job state
#define PBSE_UNKQUE     15018           /* Unknown queue name
#define PBSE_BADCRED    15019           /* Invalid Credential in request
#define PBSE_EXPIRED    15020           /* Expired Credential in request
#define PBSE_QUNOENB    15021           /* Queue not enabled
#define PBSE_QACESS     15022           /* No access permission for queue
#define PBSE_BADUSER    15023           /* Bad user - no password entry
#define PBSE_HOPCOUNT   15024           /* Max hop count exceeded
#define PBSE_QUEEXIST   15025           /* Queue already exists
#define PBSE_ATTRTYPE   15026           /* incompatable queue attribute type
#define PBSE_QUEBUSY    15027           /* Queue Busy (not empty)
#define PBSE_QUENBIG    15028           /* Queue name too long
#define PBSE_NOSUP      15029           /* Feature/function not supported
#define PBSE_QUENOEN    15030           /* Cannot enable queue,needs add def
#define PBSE_PROTOCOL   15031           /* Protocol (ASN.1) error
#define PBSE_BADATLST   15032           /* Bad attribute list structure
#define PBSE_NOCONNECTS 15033           /* No free connections
#define PBSE_NOSERVER   15034           /* No server to connect to
#define PBSE_UNKRESC    15035           /* Unknown resource
#define PBSE_EXCQRESC   15036           /* Job exceeds Queue resource limits
#define PBSE_QUENODFLT  15037           /* No Default Queue Defined
#define PBSE_NORERUN    15038           /* Job Not Rerunnable
#define PBSE_ROUTEREJ   15039           /* Route rejected by all destinations
#define PBSE_ROUTEEXPD  15040           /* Time in Route Queue Expired
#define PBSE_MOMREJECT  15041           /* Request to MOM failed
#define PBSE_BADSCRIPT  15042           /* (qsub) cannot access script file
```

```
#define PBSE_STAGEIN    15043           /* Stage In of files failed
#define PBSE_RESCUNAV   15044           /* Resources temporarily unavailable
#define PBSE_BADGRP     15045           /* Bad Group specified
#define PBSE_MAXQUED    15046           /* Max number of jobs in queue
#define PBSE_CKPBSY     15047           /* Checkpoint Busy, may be retries
#define PBSE_EXLIMIT    15048           /* Limit exceeds allowable
#define PBSE_BADACCT    15049           /* Bad Account attribute value
#define PBSE_ALRDYEXIT  15050           /* Job already in exit state
#define PBSE_NOCOPYFILE 15051           /* Job files not copied
#define PBSE_CLEANEDOUT 15052           /* unknown job id after clean init
#define PBSE_NOSYNCMSTR 15053           /* No Master in Sync Set
#define PBSE_BADDEPEND  15054           /* Invalid dependency
#define PBSE_DUPLIST    15055           /* Duplicate entry in List
#define PBSE_DISPROTO   15056           /* Bad DIS based Request Protocol
#define PBSE_EXECTHERE  15057           /* cannot execute there
#define PBSE_SISREJECT  15058           /* sister rejected
#define PBSE_SISCOMM    15059           /* sister could not communicate
#define PBSE_SVRDOWN    15060           /* req rejected -server shutting down
#define PBSE_CKPSHORT   15061           /* not all tasks could checkpoint
#define PBSE_UNKNODE    15062           /* Named node is not in the list
#define PBSE_UNKNODEATR 15063           /* node-attribute not recognized
#define PBSE_NONODES    15064           /* Server has no node list
#define PBSE_NODENBIG   15065           /* Node name is too big
#define PBSE_NODEEXIST  15066           /* Node name already exists
#define PBSE_BADNDATVAL 15067           /* Bad node-attribute value
#define PBSE_MUTUALEX   15068           /* State values are mutually exclusive
#define PBSE_GMODERR    15069           /* Error(s) during global mod of nodes
#define PBSE_NORELYMOM  15070           /* could not contact MOM
#define PBSE_NOTSNODE   15071           /* no time-shared nodes
/*
**      Resource monitor specific
*/
#define PBSE_RMUNKNOWN  15201           /* resource unknown
#define PBSE_RMBADPARAM 15202           /* parameter could not be used
#define PBSE_RMNOPARAM  15203           /* a parameter needed did not exist
#define PBSE_RMEXIST    15204           /* something specified didn't exist
#define PBSE_RMSYSTEM   15205           /* a system error occured
#define PBSE_RMPART     15206           /* only part of reservation made
#define RM_ERR_UNKNOWN  PBSE_RMUNKNOWN
#define RM_ERR_BADPARAM PBSE_RMBADPARAM
#define RM_ERR_NOPARAM  PBSE_RMNOPARAM
#define RM_ERR_EXIST    PBSE_RMEXIST
#define RM_ERR_SYSTEM   PBSE_RMSYSTEM
```

Appendix C: Glossary of PBS Terms

Account is an 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.

Attribute is an inherent characteristic of the parent object. Typically, this is a data item whose value affects the operation or behavior of the object and is settable by owner of the object. For example, the user may supply values for attributes of a job.

Batch or **batch processing**, is the capability of running jobs outside of the interactive login session. In this document, batch implies a more complex subsystem which provides for additional control over job scheduling and resource contention.

Batch Server is a persistent subsystem (daemon) upon a single host which provides batch processing capability.

Batch System is a set of batch servers that are configured for processing. The system may consist of multiple hosts, each with multiple servers.

Cluster is a set of execution "servers" or hosts on which a single batch server manages batch jobs. A cluster may be made up of a set of workstations, multiple cpu systems, or a set of nodes in a parallel system.

Complex or queue complex in NQS was a set of queues within a batch server. The purpose of a complex was to provide additional control over resource usage. The advanced scheduling features of PBS eliminates the requirement for complexes.

Destination is the location within the batch system where the job is sent for processing or executing. IN PBS, a destination may uniquely define a single queue at a single batch server or it may map into many locations.

Destination Identifier is a string which names the destination. It is in two parts and has the format queue@server where server is the name of a batch server and queue is the string identifying a queue on that server.

File Staging is the movement of files between a specified location and the execution host. See "Stage In" and "Stage Out" .

Group is a 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 is an artificial restriction which prevents a job from being selected for processing. There are three types of holds, which is applied by the job owner, (or operator) which is applied by the batch operator or administrator, and which is applied by the system itself or the batch system administrator.

Job or batch job is 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. In POSIX terms, a job is a session group. A session is a processes group the member processes cannot leave.

Manager or **Batch System Manager** is a person authorized to use all restricted capabilities of the batch system. The manager may act upon the batch system, queues, or jobs. Also called the administrator.

Operator or **Batch Operator** is a person authorized to use some but not all of the restricted capabilities of the batch system.

Owner of a job is the user who submitted the job to the batch system.

PBS is short for **Portable Batch System**.

POSIX refers to the various standards being developed by the "Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society" under standard P1003. There are a number of subcommittees under POSIX, those of interest to this project are:

POSIX.1 System Application Program Interface (the system calls).
POSIX.2 The command shell language.
POSIX.3 Test Methods
POSIX.10 Super Computing Profile
POSIX.12 Protocol Independent Interfaces (one of the many network working groups)
POSIX.14 Multiprocessor Working Group
POSIX.15 Batch Queuing Extensions. This standard has been approved as 1003.2d.

Queue is a collection of jobs (or job related tasks) within the batch queuing system. Each queue has a set of associated attributes which determine what actions are performed upon each job within the queue. Typical attributes include queue name, queue priority, resource limits, destination(s) and job count limits. Selection/scheduling of jobs is implementation defined. The use of the term "queue" does not imply the ordering is "first in, first out."

Rerunable If a batch job can be terminated and its execution restarted from the beginning without harmful side effects, then the job is said to be rerunable.

Stage In is to move a file or files to the host prior to the batch job beginning execution.

Stage Out is to move a file or files off of the host after the batch job completes execution.

User is a user of the compute system. Each user is identified by a unique character string, the user name; and by a unique number, the user id.

User ID is a numeric identifier uniquely assigned to each user. Privilege to access system resources and services is typically established by the user id.