Quick Tutorial for Portable Batch System (PBS)

The Portable Batch System (PBS) system is designed to manage the distribution of batch jobs and interactive sessions across the available nodes in the cluster. This web page provides an introduction to basic PBS usage.

---

1. PBS Commands
2. Setting PBS Job Attributes
3. Requesting PBS Resources
4. PBS Environment Variables
5. Specifying a Node Configuration
6. Selecting a Queue
7. Running a batch job
8. Stopping a running job
9. Using an interactive session
10. Monitoring a job
11. FAQ

---

1. PBS Commands

The following is a list of the basic PBS commands. They are organized according to their functionality.

Job Control

- qsub: Submit a job
- qdel: Delete a batch job
- qsig: Send a signal to batch job
- qhold: Hold a batch job
- qrls: Release held jobs
- qrerun: Rerun a batch job
- qmove: Move a batch job to another queue

Job Monitoring

- qstat: Show status of batch jobs
- qselect: Select a specific subset of jobs

Node Status

- pbsnodes: List the status and attributes of all nodes in the cluster.

Miscellaneous

- qalter: Alter a batch job
- qmsg: Send a message to a batch job

More information on each of these commands can be found on their manual pages.

---

2. Setting PBS Job Attributes

PBS job attributes can be set in two ways:

1. as command-line arguments to `qsub`, or
2. as PBS directives in a control script submitted to `qsub`.

The following is a list of some common PBS job attributes.

Attribute	Values	Description
`-l`	comma separated list of required resources (*e.g.* `nodes=2,cput=00:10:00`)	Defines the resources that are required by the job and establishes a limit to the amount of resources that can be consumed. If it is not set for a generally available resource, the PBS scheduler uses default values set by the system administrator. Some common resources are listed in the Requesting PBS Resources section.
`-N`	name for the job	Declares a name for the job
`-o`	[hostname:]pathname	Defines the path to be used for the standard output (STDOUT) stream of the batch job.
`-e`	[hostname:]pathname	Defines the path to be used for the standard error (STDERR) stream of the batch job.
`-p`	integer between -1024 and +1023	Defines the priority of the job. Higher values correspond to higher priorities.
`-q`	name of destination queue, server, or queue at a server	Defines the destination of the job. The default setting is sufficient for most purposes.

---

3. Requesting PBS Resources

PBS resources are requested by using the `-l` job attribute (see the Setting PBS Job Attributes section).

The following is a list of some common PBS resources. A complete list can be found in the `pbs_resources` manual pages.

Resource	Values	Description
`nodes`	See the Specifying a Node Configuration section.	Declares the node configuration for the job.
`walltime`	`hh:mm:ss`	Specifies the estimated maximum wallclock time for the job.
`cput`	`hh:mm:ss`	Specifies the estimated maximum CPU time for the job.
`mem`	positive integer optionally followed by `b`, `kb`, `mb`, or `gb`	Specifies the estimated maximum amount of RAM used by job. By default, the integer is interpreted in units of bytes.
`ncpus`	positive integer	Declares the number of CPUs requested.

---

4. PBS Environment Variables

The following is a list of the environment variables set by PBS for every job.

Variable	Description
PBS_ENVIRONMENT	set to PBS_BATCH to indicate that the job is a batch job; otherwise, set to PBS_INTERACTIVE to indicate that the job is a PBS interactive job
PBS_JOBID	the job identifier assigned to the job by the batch system
PBS_JOBNAME	the job name supplied by the user
PBS_NODEFILE	the name of the file that contains the list of the nodes assigned to the job
PBS_QUEUE	the name of the queue from which the job is executed
PBS_O_HOME	value of the HOME variable in the environment in which qsub was executed
PBS_O_LANG	value of the LANG variable in the environment in which qsub was executed
PBS_O_LOGNAME	value of the LOGNAME variable in the environment in which qsub was executed
PBS_O_PATH	value of the PATH variable in the environment in which qsub was executed
PBS_O_MAIL	value of the MAIL variable in the environment in which qsub was executed
PBS_O_SHELL	value of the SHELL variable in the environment in which qsub was executed
PBS_O_TZ	value of the TZ variable in the environment in which qsub was executed
PBS_O_HOST	the name of the host upon which the qsub command is running
PBS_O_QUEUE	the name of the original queue to which the job was submitted
PBS_O_WORKDIR	the absolute path of the current working directory of the qsub command

---

5. Specifying a Node Configuration

The nodes resource_list item (*i.e.* the node configuration) declares the node requirements for a job. It is a string of individual node specifications separated by plus signs, +. For example,

```
3+2:fast
```

requests 3 plain nodes and 2 ``fast'' nodes. A node specification is generally one of the following types

- the name of a particular node in the cluster,
- a node with a particular set of properties (*e.g.* fast and compute),
- a number of nodes,

- a number of nodes with a particular set of properties.

In the last case, the number of nodes is specified first, and the properties of the nodes are listed afterwards, colon-separated. For instance,

```
4:fast:compute:db
```

Two important properties that may be specified are:

`shared`	indicates that the nodes are not to be allocated exclusively for the job. If this property is not specified, the PBS scheduler will allocate each processor exclusively to the job. This does not necessarily mean that the node has been exclusively allocated. For example, a job may only have been allocated a single processor on a given node. **Note**: The `shared` property may only be used as a global modifier.
`ppn=<number of processors per node>`	requests a certain number of processors per node be allocated.

Global Modifiers

The node configuration may also have one or more global modifiers of the form `#<property>` appended to the end of it which is equivalent to appending `<property>` to each node specification individually. That is,

```
4+5:fast+2:compute#large
```

is completely equivalent to

```
4:large+5:fast:large+2:compute:large
```

The `shared` property is a common global modifier.

Common Node Configurations

The following are some common node configurations. For each configuration, both the exclusive and shared versions are shown.

1. specified number of nodes:

   ```
   nodes=<num nodes>
   nodes=<num nodes>#shared
   ```

2. specified number of nodes with a certain number of processors per node:

   ```
   nodes=<num nodes>:ppn=<num procs per node>
   nodes=<num nodes>:ppn=<num procs per node>#shared
   ```

3. specific nodes:

   ```
   nodes=<list of node names separated by '+'>
   nodes=<list of node names separated by '+'>#shared
   ```

4. specified number of nodes with particular properties

```
nodes=<num nodes>:<property 1>:...
nodes=<num nodes>:<property 1>:...#shared
```

---

6. Selecting a Queue

PBS is often set up with multiple job queues which assist the PBS scheduler to determine the order in which to run jobs. These are typically organized to sort jobs by size. A user should select a queue based on an estimate for the size of his/her job. A list of the available queues can be found by using the, `qmgr` utility with the `p s` command.

Note:

- If the resource requirements for a job exceed the maximum values for the queue it is submitted to, the job will automatically be terminated by PBS.
- Unless otherwise specified (see the Setting PBS Jobs Attributes section), jobs are submitted to the `test_queue`, which is the most resource constrained queue.

---

7. Running a batch job

To run a job in batch mode, a control script is submitted to the PBS system using the `qsub` command:

```
qsub [options] <control script>
```

PBS will then queue the job and schedule it to run based on the jobs priority and the availability of computing resources.

The control script is essentially a shell script that executes the set commands that a user would manually enter at the command-line to run a program. The script may also contain PBS directives that are used to set attributes for the job. Here is a simple script that prints some environment information and runs the program `hello_world` simultaneously on four nodes.

```
#!/bin/sh
### Job name
#PBS -N hello_world_job
### Output files
#PBS -o hello_world_job.stdout
#PBS -e hello_world_job.stderr
### Queue name
#PBS -q dqueue
### Number of nodes
#PBS -l nodes=4:compute#shared

# Print the default PBS server
echo PBS default server is $PBS_DEFAULT

# Print the job's working directory and enter it.
echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR

# Print some other environment information
echo Running on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
NODES=`cat $PBS_NODEFILE`
echo $NODES

# Compute the number of processors
```

```
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS nodes

# Run hello_world
for NODE in $NODES; do
  ssh $NODE "hello_world" &
done

# Wait for background jobs to complete.
wait
```

A few special features of this control script should be noted:

- The lines beginning with `#PBS` are the PBS directives that set job attributes. Actually, the job attributes are just the options that can be passed to `qsub` as command-line arguments. A few of the more common attributes are listed in the Setting PBS Job Attributes section. For a complete list, see the OPTIONS section of `qsub` manual pages.
- PBS automatically defines several environment variables that start with the `PBS_` prefix that contain information about the environment in which the control script is executing. A complete list of the PBS environment variables can be found in the PBS Environment Variables section or in the `qsub` manual pages.
- The `wait` command after the `for`-loop is essential to make sure that the PBS jobs waits for the background jobs to complete before exiting.

---

8. Stopping a running job

1. Find the job ID using the `qstat` command. The job ID is the number listed under the "Job ID" field (**NOT** including the server name). For example, if the job ID listed by `qstat` is 394.master.amcl, the job ID is 394.
2. Send the running job the `KILL` signal by using the command

```
qsig -s KILL <job ID>
```

---

9. Using an interactive sessions

For debugging purposes, it is often useful to be able to use an ``interactive'' session. From a user's perspective, an interactive session is not very different from logging directly into and working on a compute node. However, there are a few essential differences:

- PBS will automatically attempt to place you on a node with a minimal load
- an interactive session may give the user control of multiple nodes/processors
- if sufficient resources are available, a user may request exclusive use of nodes/processors.

A user requests an interactive session by using the command

```
qsub -I
```

To request a specific node/processor configuration for the session, the optional `-l` command-line argument may be used. For example, the command

```
qsub -I -l nodes=2:compute#shared
```

will start an interactive session consisting of 2 nodes from the pool of ``compute'' nodes that are not exclusively allocated. More information on various node configurations can be found in the Specifying a Node Configuration section.

As for batch jobs, PBS sets several environment variables for interactive sessions. When multiple nodes are requested, the `$PBS_NODEFILE` variable is particularly important because is contains a list of the nodes that have been allocated to the session.

---

10. Monitoring a job

Users can monitor jobs in the PBS system using the `qstat` command. Most commonly, it is used to examine the current load on the system. However, it can also be used to extract detailed information (*e.g.* wallclock time, cpu time, memory use) about specific jobs in progress.

By default, `qstat` returns information about *all* jobs in system. Information about specific jobs is retrieved by providing `qstat` a list of job id's. More information can be found in the `qstat` manual pages.