# OpenPBS Users Manual

**How to Write a PBS Batch Script**

PBS scripts are rather simple. An MPI example for user **your-user-name**:

**Example: MPI Code**

```
#PBS -N a_name_for_my_parallel_job
#PBS -l nodes=7,walltime=1:00:00
#PBS -S /bin/sh
#PBS -q cac
#PBS -M your-email-address
#PBS -m abe
#PBS -o  /users/your-user-name/output
#PBS -e /users/your-user-name/errors
#
echo "I ran on:"
cat $PBS_NODEFILE
#
export GMPICONF=/users/your-user-name/.gmpi/$PBS_JOBID
#
#cd to your execution directory first
cd ~
#
#use mpirun to run my MPI binary with 7 nodes for 1 hour
mpirun -np 14 ./your-mpi-program
```

The PBS script parameters are as follows:

**#PBS -N testjob** Name of the job in the queue is "testjob".
**#PBS -l nodes=7,walltime=1:00:00** Reserve 7 machines (14 processors), for 1 hour.
**#PBS -S /path/to/shell** Script is /bin/sh (see below)
**#PBS -q default** Submit to the queue named default.
**#PBS -M your-email-address** Email me at this address.
**#PBS -m abe** Email me when the job **a**borts, **b**egins, and **e**nds. **#PBS -o /users/your-user-name/output** Write stdout to this file.
**#PBS -e /users/your-user-name/errors** Write stderr to this file.

For complete information on PBS flags, use "man qsub". For further information on PBS, use "man pbs".

The MPI (mpirun) parameters are as follows:

**-np** Number pf processes.
**-stdin** <filename> Use "filename" as standard input.
**-t** Test but do not execute.

**Example: OpenMP Code**

If you're running OpenMP code (w/ 1 or 2 processes on these machines):

```
#PBS -N myparalleljob
#PBS -l nodes=1,walltime=90:00
#PBS -S /bin/sh
#PBS -q cac
#PBS -M your-email-address
#PBS -m abe
#PBS -o /users/your-user-name/output
#PBS -e /users/your-user-name/errors
#
echo "I ran on:"
cat $PBS_NODEFILE
#
export GMPICONF=/home/your-user-name/.gmpi/$PBS_JOBID
export NCPUS=2
#
# cd to your execution directory first
cd ~
./a.out
```

You may find it necessary to add the following to OpenMP jobs, should you run low on stack space due to the default stack size of 2 MB:

```
export MPSTKZ 8M
```

If you have a serial code just set 'nodes=1'.
For example:

```
#PBS -N testjob
#PBS -l nodes=1,walltime=24:00
#PBS -q queue-name
#PBS -M your-email-address
#PBS -m abe
#PBS -o /users/your-user-name/output
#PBS -e /users/your-user-name/errors
#
# cd to your execution directory first
cd ~/myrundir
executable < input1 > output1 &
```

Or if you want to maximize your use of a node by running two serial processes on the node (one process per CPU):

```
#PBS -N testjob
#PBS -S /bin/sh
#PBS -l nodes=1,walltime=24:00
#PBS -q queue-name
#PBS -M your-email-address
#PBS -m abe
#PBS -o /users/your-user-name/output
#PBS -e /users/your-user-name/errors
# <input
# cd to your execution directory first
cd ~/myrundir
executable < input1 > output1 &
executable < input2 > output2 &
wait   # make sure you wait, else the slower job will abort
```

Use "qstat -f -Q" to see a list of the currently active queues on the machine you're using. They may differ from the list above.

In this script, stdout and stderr will be directed into file JobName.o##. JobName was specified by the -N flag in the script file.

**How to Submit a PBS Batch Script**

To submit an PBS script simply type:

```
qsub scriptname
```

where scriptname is the name of your PBS script. Note that PBS runs your script under the your shell, unless otherwise told to do so. One benefit of running under /bin/sh is the csh is arguably broken in how it handles terminal-disconnected jobs (same goes for tcsh). Using csh or tcsh is fine, but you will receive error warnings at the beginning of your output file:

```
Warning: no access to stty (Bad file descriptor).
Thus no job control in this shell.
```

**How to Check the Status of a PBS Batch Job**

To check the status of your job in the queue, type:

```
qstat
```

Note: This will show only your jobs. To see all jobs in the queue, type:

qstat -a

**How to Cancel a PBS Batch Job**

If you realize that you made a mistake in your script file or if you've made modifications to your program since you submitted your job and you want to cancel your job, first get the "Job ID" by typing qstat.

For example:

qdel 203
*or*
qdel 203

**How to Query the PBS Queues**

To see the names of the available queues and their current parameters, type:

qstat -f -Q

The notable parameters in the output are Queue and resources_max.cput for the CPU limits.

**How do I choose myrinet nodes when running my OpenPBS job.**

In the above scripts you may have noticed the following

#PBS -l nodes=10,walltime=24:00

In order to make sure your myrinet code is run on nodes in cluster that have a myrinet card installed you will want to add the following to the above line.

#PBS –l nodes=10:myrinet
or
#PBS –l nodes=10:ppn=2:myrinet

This would cause pbs to only choose and use10 nodes with the pbs attribute myrinet. The latter example would cause pbs to only choose and use 10 nodes with 2 processors each and a myrinet card node attribute.

**Node Attribute Discriptions:**

Please note that by default, all jobs are run using the "general" node attribute unless you specify it as explained above.

*General:* Nodes that can run jobs that do not require a myrinet card installed

*Myrinet:* Nodes with myrinet cards installed

*Myritest:* Nodes that can run test jobs that require a myrinet card

*Test:* Nodes that can run test jobs that do not require a myrinet card

*Fatnode:* Nodes that can run jobs that require larger amounts of memory and more cpus. Currently those nodes are 4 cpu/12G memory nodes

**How do I specify the number of processors I want to use per node?**

You would specify the number of processors using the ppn(processor per node) attribute as such

#PBS –l nodes=10:ppn=2

Your job would then be run using 20 processors. If you specified ppn=1 then your job would be run using only 10 processors. It is actually best to specify less nodes with a larger number of  if you need to run singular jobs so that you can use both cpu's on a compute node. I.E.

To run 10 processes, it would be best to use something like this

#PBS –l nodes=5:ppn=2

**Where do I go to for more information?**

The best place to get information about PBS Script variables is by using the man page for qsub ie.

man qsub

For more information about deleting your job, access the qdel man page

man qdel

For more information about job status information, access the qstat man page

```
Man qstat
```

# Maui Users Manual

## Maui Introduction

The Maui Scheduler was designed to offer improved job management and scheduling to users while allowing users to continue 'business as usual'. In fact, users do not need to change anything in the way they submit and track jobs when Maui is installed. However, if a user chooses, there are many new features and commands which can be utilized to improve the user's ability to run jobs when, where, and how they want.

The Maui Scheduler, as its name suggests, is a scheduler. It is not a resource manager. A resource manager, such as PBS, Loadleveler, or LSF, manages the job queue and manages the compute nodes. A scheduler tells the resource manager what to do, when to run jobs, and where. Users typically submit jobs and query the state of the machine and jobs through the resource manager. When Maui is running, users can continue to issue the exact same resource manager commands as before. However, Maui also offers commands which provide additional information and capabilities.

Maui capabilities include many internal mechanisms to improve overall scheduling performance, allowing users to run more jobs on the same system and get their results back more quickly. Additionally, Maui allows users to create resource reservations which guarantee resource availability at particular times. Quality of service features are also enabled which allow a user to request improved job turnaround time, access to additional resources, or exemptions to particular policies automatically. (The site administrator may choose to make some of these capabilities only available at a higher 'job cost')

---

## Maui Overview

Maui is an advanced cluster scheduler capable of optimizing scheduling and node allocation decisions. It allows site administrators extensive control over which jobs are considered eligible for for scheduling, how the jobs are prioritized, and where these jobs are run. Maui supports advance reservations, QOS levels, backfill, and allocation management. Each of these features, if enabled, may require some adjustment on the part of the user to optimize system performance.

---

## Backfill

Backfill is a scheduling approach which allows some jobs to be run 'out of order' so long as they do not delay the highest priority jobs in the queue. In order to determine whether or not a job will be delayed, each job must supply an estimate of how long it will need to run. This estimate, known as a wallclock limit, is an estimation of the wall time (or elapsed time) from job start to job finish. It is often wise to slightly overestimate this

limit because the scheduler may be configured to kill jobs which exceed their wallclock limits. However, overestimating a job's wallclock time by too much will prevent the scheduler from being able to optimize the job queue as much as possible. The more accurate the wallclock limit, the more 'holes' Maui can find to start your job early.

Maui also provides the command showbf to allow users to see exactly what resources are available for immediate use. This can allow users to configure a job that will be able to run a soon as it is submitted by utilizing only available resources.

Backfill scheduling significantly improves the ability of the scheduler to utilize the available resources. Consequently, using backfill scheduling increases system utilization and throughput while decreasing average job queue time. Fortunately, backfill is a very forgiving algorithm, allowing even jobs with very poor wallclock estimates to benefit from it. However, better estimates will increase the amount of improvement backfill scheduling can provide for your jobs.

---

**Allocation Management**

Maui possesses interfaces to a number of allocation management systems such as PNNL's QBank. These systems allow each user to be given a portion of the total compute resources available on the system. These systems work by associating each user with one or more accounts. When a job is submitted, the user specifies which account should be charged for the resources consumed by the job. Default accounts may be specified to automate the account specification process in most cases. If such a system is being used at your site, your system administrators will inform you as to if and how accounts should be specified.

---

**Advance Reservations**

Advance reservations allow a site to set aside certain resources for specific uses over a given timeframe. Access to a given reservation is controlled by a reservation-specific access control list (ACL) which determines who or what can use the reserved resources. It is important to note that while reservation ACL's *allow* particular jobs to utilize reserved resources, they do not *force* the job to utilize these resources. Maui will attempt to locate the best possible combination of available resources whether these are reserved or unreserved. For example, in the figure below, note that job X, which meets access criteria for both reservation A and B, allocates a portion of its resources from each reservation and the remainder from resources outside of both reservations.

While by default, reservations make resources available to jobs which meet particular criteria, Maui can be configured to constrain jobs to only run within accessible

reservations. Specifically, jobs can be forced to run only within reserved resources on a job by job basis.

---

**Quality of Service (QOS)**

The Maui QOS features allow a site to grant special privileges to particular users. These benefits can include access to additional resources, exemptions from certain policies, access to special capabilities, and improved job prioritization. Each site determines which advantages are important to make available and to whom. If you are granted special QOS access, you can specify the QOS to use for your job using the **QOS** keyword.

---

**Statistics**

Maui tracks a large number of statistics to help users determine how well and how often their jobs are running. The showstats command provides detailed statistics on a per user, per group, and per account basis. Additionally, the command showgrid can be used to determine what types of jobs get the best scheduling performance allowing users to 'tune' their jobs to obtain optimal turnaround time.

---

**Diagnosis**

Maui provides the checkjob command to allow users to view a detailed status of each job they have submitted. This command show all job attribute and state information and also provides an analysis of whether or not the job can run. If the job is unable to run, this command will provide a breakdown of these reasons why. The showstart command provides an estimate of job start time beyond this.

If your job still will not start, contact your system administrator. He will have access to additional commands and detailed Maui logs which will reveal exactly why the job cannot run.

---

**Workload Information**

Maui offers an extensive array of job prioritization options to allow sites to control exactly how jobs run through the job queue. If your site administrators have chosen to take advantage of this, the job ordering shown by your resource manager queue listing command (i.e., llq, qstat) will not reflect this. Maui provides the showq command to display a relevant listing of both active and idle jobs.