

Batch Processing on OSC Systems

**Science & Technology Support
High Performance Computing**

**Ohio Supercomputer Center
1224 Kinnear Road
Columbus, OH 43212-1163**

Table of Contents

- **Batch Processing: Step by Step**
- **Minimum Batch Files for Each OSC System**
- **Advantages of Batch Processing**
- **Useful Batch File Header Lines**
- **Useful Batch Environmental Variables**
- **More PBS commands**

Batch Processing: Step by Step

- **Interactive Processing vs. Batch Processing: An example**
 - Structure of a Batch File
- **Description of Batch File Header Lines**
- **Procedure to run a Batch job**
- **Explanation of commands**
- **PBS log file**
- **Deleting a batch job**

Interactive Processing

- The way you are used to working on a workstation or laptop!
- Enter a command, output returned to monitor. Based on output, enter a command, output returned to monitor, repeat. User is **interacting** in real-time with the computer.
- Interactive use is easiest (and almost required) for tasks that involve user's analysis of previous command's output to determine the next command.
- Common interactive tasks: file/directory searching; directory management (clean-up, reorganization, etc.); file editing; code debugging (in any manner); use of window-based software; use of performance tools (although most create a raw data file); and the list goes on ...

Example Program

- Throughout this workshop the same tasks will be carried out in different ways. The tasks are:
 - 1) Changing from home directory to “work” directory
 - 2) Compiling a **bug-free** source code file
 - 3) Running the executable produced
 - 4) Examining the output
 - 5) Cleaning up removing unnecessary files

Example Program

- The program used simply adds together integers whose values are created in a loop nest (nest.c):

```
#include <stdio.h>
#include <math.h>

int main(int argc, char* argv[]) {
    float x;
    float rock=0;
    int i,j,N,M;

    if (argc == 3) {                // Check for correct argument count
        N = atoi(argv[1]);          // Read as arguments to the command
        M = atoi(argv[2]);
    }
    else {                          // Exit if incorrect number of arguments
        printf("Usage : ./a.out int int\n");
        exit();
    }

    // Set Loop Counts
    for(i=1;i<=N;i=i+1) {
        for(j=1;j<=M;j=j+1) {
            x=sqrt((float) ((10*i)+j)); // Calculate the square root
            rock=rock+x;                // Calculate Sum
        }
    }

    printf("For loop counts N=%d M=%d\n",N,M);
    printf("Sum=%f\n",rock);          // Output
}
```

Interactive Session (mck.osc.edu)

```
mck-login1:$ cd batchshop/mck
```

```
mck-login1:$ cc nest.c -lm
```

```
mck-login1:$ ./a.out 4 8
```

For loop counts N=4 M=8

Sum=944

```
mck-login1:$ rm a.out
```

Batch Processing of SAME tasks

- **Key fact**: In the previous interactive session the user already knew before logging in what commands they were going to type. The code had been debugged, everything is ready for a “production” run.
- **NO REAL-TIME INTERACTION IS REQUIRED**
 - Therefore, you can put the commands in a **batch file**
- A batch file is just a script (sequence of UNIX commands put into a file) that contains:
 - 1) The **EXACT SAME** commands you typed on the keyboard during the interactive session
 - 2) Some lines at the beginning of the file that tells the batch system software some parameters it needs to know. These opening lines are called the **header** of the batch file

Structure of a Minimum Batch File

```
#PBS -l walltime=00:01:00
```

```
#PBS -N nest
```

```
#PBS -j oe
```

```
#PBS -S /bin/ksh
```



Header

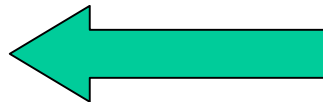
```
set -x
```

```
cd batchshop/mck
```

```
cc nest.c
```

```
./a.out 4 8
```

```
rm a.out
```



Same Unix Commands

Batch Processing Session

```
mck-login1:$ qsub mck.run
```

```
17182.nfs1.osc.edu
```

```
mck-login1:$ qstat -a
```

```
nfs1.osc.edu:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
17167.nfs1.osc.	osu3246	parallel	SiI3us08S0	3909	2	--	--	20:00	R	01:33
17168.nfs1.osc.	osu2614	serial	ohadd-eacr	28528	1	--	60mw	40:00	R	02:04
17182.nfs1.osc.	dje	serial	nest	--	1	--	--	00:01	Q	--

```
mck-login1:$ qstat -a
```

```
nfs1.osc.edu:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
17167.nfs1.osc.	osu3246	parallel	SiI3us08S0	3909	2	--	--	20:00	R	01:33
17168.nfs1.osc.	osu2614	serial	ohadd-eacr	28528	1	--	60mw	40:00	R	02:04

```
mck-login1:$ ls -lrt
```

```
total 40
```

```
-rw-r--r--    1 dje      G-0541      282 May 15 13:20 nest.c
-rw-r--r--    1 dje      G-0541      109 May 15 13:25 mck.run
-rw-----    1 dje      G-0541       89 May 15 13:27 nest.o17182
```

```
mck-login1:$ cat nest.o17182
```

```
+ cd batchshop/mck
```

```
+ cc nest.c
```

```
+ ./a.out 4 8
```

```
For loop counts N=4 M=8
```

```
Sum=944
```

```
+ rm a.out
```

```
mck-login1:$
```

Description of Batch File Header Lines

- The batch processing used on the OSC machines is called the **P**ortable **B**atch **S**ystem. It requires that each header line begin with **#PBS**. Note that the **#** symbol indicates to the shell that this line is just a comment from its point of view.
- **#PBS -l walltime=00:01:00**
 - WARNING: the character before the word walltime is a lower-case “e” not the digit character “1”. The “e” stands for limit
 - This header line tells PBS how much real time (the time on the clock on the wall) you expect the execution of the batch file commands to take.
 - Since the time format is hh:mm:ss, the limit on the wall clock time is 1 minute for this header line
 - You can estimate wall clock time by running your program once and using external or internal timing commands (see later chapters)

Description of Batch File Header Lines

- **#PBS -N nest**
 - This header line gives a Name to your batch job, here the name is nest
 - The job name is used by PBS in several ways
 - When you check on the status of batch job (qstat command) your job name is shown in the PBS information table
 - Your job name is the prefix for the log file name returned by PBS.
 - A PBS log file can be thought of as a screen dump. The log file contains everything that would have appeared on your monitor if the commands had been run interactively.
- **#PBS -j oe**
 - By default, PBS returns two log files: one for standard out data stream, the other for the standard error data stream
 - This option **j**oins these two log files into one.

Procedure to run a Batch job

1. Write a batch file

2. Use the `qsub` command to submit the file to PBS

- When PBS begins executing batch file commands, it will assume it is in your login directory, `$HOME`.
- It is a good idea to retain the job number identified in the return line.

3. Use the `qstat` command to check on your job status

4. PBS is done with your batch job when it no longer appears in the `qstat` output table

- Starts your job when requested resources become available
- Ends your job when batch-file commands finish or time limit reached

5. Examine the log file returned by PBS to see your job output

- Log file appears in the directory you submitted the job from

Explanation of PBS commands

- **qsub *batch_file_name***
 - Submits your batch job to PBS
 - Based on the header parameters, PBS puts your job in a queue in which your job “fits”
 - Returns you Job Identification Number (JIN=17182 in our example)
- **qstat -a**
 - Displays status information on **a**ll the batch jobs in PBS at that time
 - Output is a multi-columned, lengthy table containing many pieces of information
 - The first four columns identify the job:
 - JID that matches value returned from qsub,
 - Your OSC login ID
 - The name of the queue your job was placed in
 - The “internal” name for your job as set in the -N header line
 - The Required Time column shows the time limit you set in the batch file header (00:01 minute)
 - **MOST IMPORTANT COLUMN** is the S(tatus) column which shows a letter code for what your code is doing now. The code Q stands for queued and the code R stands for running.

The PBS log file

- The log file name is somewhat cryptic but there is a pattern to it:
 - *Internal_name.oJID*
 - The o after the period stands for output
- When the user views the returned log file, they see each UNIX command run in the order set in the batch file. If the command produced output it is shown on the lines immediately following the command name.
- In our sample log file, the UNIX commands are preceded by a “+”. This is due to the **set -x** command (ksh) executed first in the batch file. The set -x command causes each batch file command to be echoed to the monitor with a + in front. Without the set -x command (or its equivalent) only the actual screen output would appear in the log file.

Deleting a Batch Job

- Situations may arise in which the user may want to delete a file from its PBS queue. Such conditions might be resource limits set incorrectly; incorrect or missing commands in the batch file; the “infinite-loop” feeling that your program is taking too long to run; etc.
- The PBS command to delete a job is **qdel**. To use it, simply type

```
$ qdel JID
```

Batch Files for Different OSC Machines

- **Classification of OSC Systems**
- **Itanium 2 Cluster**
- **Pentium 4 Cluster**
- **SGI Altix**
- **Determining the Memory Limit**
- **SUN SunFire 6800**

OSC Supercomputer Systems

- **Distributed Memory Systems**
 - Itanium 2 Cluster (mck.osc.edu)
 - Pentium 4 Cluster (oscbw.osc.edu)
- **Used as Shared Memory Machines**
 - SGI Altix (compute node behind mck.osc.edu)
 - SUN SunFire 6800 (coe.osc.edu)
- **All four systems use PBS as the batch software, so their batch file and log files are very similar. It will be seen that with the Shared Memory machines a memory limit must also be set.**
- **The Cray SV-1ex (oscb.osc.edu) runs NQE for its batch processing. Attendees interested in NQE will find there is a simple translation of PBS options to NQE options.**

Itanium 2 Cluster

- **Minimum Batch File (from first chapter)**

```
#PBS -l walltime=00:01:00
#PBS -N nest
#PBS -j oe
set -x
cd batchshop/mck
cc nest.c
./a.out 4 8
rm a.out
```

- **Log File produced (also seen in first chapter)**

```
+ cd batchshop/mck
+ cc nest.c
+ ./a.out 4 8
For loop counts N=4 M=8
Sum=944
+ rm a.out
```

Pentium 4 Cluster

- **Minimum Batch File**

```
#PBS -l walltime=00:01:00
#PBS -N nest
#PBS -j oe
set -x
cd batchshop/bw
cc nest.c
./a.out 4 8
rm a.out
```

- **Log File produced**

```
+ cd batchshop/bw
+ cc nest.c
+ ./a.out 4 8
For loop counts N=4 M=8
Sum=944
+ rm a.out
```

SGI Altix

- **Minimum Batch File**

```
#PBS -l walltime=00:10:00
#PBS -l ncpus=4
#PBS -N nest
#PBS -j oe
set -x
cd batchshop/altix
cc nest.c
./a.out 4 8
rm a.out
```

- **Changes: specify CPU count**

SGI Altix Log File

```
+ cd batchshop/altix
+ cc nest2.c
+ ./a.out 4 8
For loop counts N=4 M=8
Sum=944
+ rm a.out
```

SunFire 6800

- **Minimum Batch File**

```
#PBS -l cput=00:10:00
#PBS -N nest
#PBS -j oe
set -x
cd batchshop/coe
cc nest.c
./a.out 4 8
rm a.out
```

- **Similar to batch file for Altix, although CPU count not required.**
- **Also use `cput` (CPU time) instead of `walltime`.**

SunFire 6800

- **Log file returned**

```
+ cd batchshop/coe
```

```
+ cc nest.c
```

```
+ ./a.out 4 8
```

```
For loop counts N=4 M=8
```

```
Sum=944
```

```
+ rm a.out
```

Advantages of Batch Processing

- **Interactive resource limits too small.**
 - `limit` UNIX command will show interactive limits on CPU time (session & process), memory size, disk size, etc.
- **Improves overall system efficiency by weighing user requirements against system load.**
- **Makes sure all users can get equal access to resources by enforcing a scheduling policy.**
- **Automatically keeps a log of your Unix commands and their output**
- **Only way to access > 2 chips for parallel processing**
- **Batch Processing concepts same for all batch software**
- **Learn PBS on one OSC machine, know it for all**

Useful Batch File Header Lines

- Header Line = qsub option
- Mailing options
- Rename the Log File
- Use a Different Shell
- *Parallel Processing*
- Starting Date & Time
- Special Queues

qsub Options

- The header lines of a PBS batch file are actually options to the `qsub` command. For example in our batch files we have put the header line

```
#PBS -j oe
```

We could have left out that header line and used that option when the batch file is submitted:

```
qsub -j oe batch_file
```

- It is recommended to put the options in the header section of the batch file so that the user has a record of values used.
- **In this chapter, other options (besides the minimum suggested) will be discussed with emphasis on the most useful.**
- Of course, all the `qsub` options can be found in its man page

Email from PBS

- Do I really have to keep checking on the status of my job with `qstat -a` to find out its progress?
- Answer:no By using the following batch file header lines, PBS will email you a message that your job has begun and that your job has ended, respectively

```
#PBS -m b
#PBS -m e
```
- In the “job ending” email message the return status of job is reported. A return status of 0 indicates everything was successful. It will also tell how much memory and time your job took
- There is also a `-m a` option that causes email to be sent if your job aborts

Sample End Email

Date: Sun, 18 May 2003 16:44:49 -0400
From: adm <adm@osc.edu>
To: dje@mck-login1.osc.edu
Subject: PBS JOB 17353.nfs1.osc.edu

PBS Job Id: 17353.nfs1.osc.edu
Job Name: liver
Execution terminated
Exit_status=0
resources_used.cput=00:02:34
resources_used.vmem=8128kb
resources_used.walltime=00:02:34

Log File Name

- Do I have to put up with that weird name for the log file returned by PBS?
- Answer: no Add the following header line and choose the name you want:

`#PBS -o file_name`

- This option stands for (o)utput. When the batch job is finished everything that would have been displayed on your monitor is contained in *file_name*.

Changing Shells

- If otherwise specified, the **UNIX shell** used to execute the batch file commands is your login shell
- The user can choose to run a batch job in a different shell if they desire. The header line is:

```
#PBS -S /bin/[csh|ksh|tcsh]
```

- Notice the full path name of the shell command must be used.
- **NOTE:** Echoing of commands in the C shell is enabled by the command

```
set echo
```

Also, the “echoed” commands are not proceeded by a ‘+’

Parallel Processing for Clusters

- One batch file header line performs the most critical step needed in parallel processing: setting the number of processors your code will run on.
- This syntax for this important header line is:

```
-l nodes=N:ppn=[1|2]
```
- The first part of this option specifies the number (N) of nodes you need. The maximum value for N depends on the nodes available on a given machine.
- It turns out that the OSC clusters have two processors in each node. The second part of the option indicate how many **processors per node** are used, 1 or 2. If the ppn section is omitted, PBS will default to 1 processor per node.

Parallel Processing Batch File

```
#PBS -l walltime=00:04:00
#PBS -N search
#PBS -j oe
#PBS -l nodes=4
set -x
cd batchshop/options
mpicc search.c
qstat -rn
mpiexec a.out < data3.txt
rm a.out
rm search.o
```

- The search program uses 4 processors to each search a quarter of an integer array for the value 11. When one processor has found it, that processor signals the others to stop searching.
- The `-n` option for `qstat` has been used to show what physical nodes the code is actually being run on.

Log File (ppn=1)

```
+ cd batchshop/options
+ mpicc search.c
search.c
+ qstat -rn
```

nfs1.osc.edu:

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
17433.nfs1.osc.	dje	parallel	search	28805	4	--	--	00:04	R	--

mck084/0+mck083/0+mck082/0+mck081/0

```
+ mpiexec a.out
+ < data3.txt
P:0 11 found at index=1499
P:0 I searched up to index 1499
P:1 I searched up to index 1858
P:2 I searched up to index 1881
P:3 I searched up to index 1886
+ rm a.out
+ rm search.o
```

Log File (ppn=2)

```
+ cd batchshop/options
+ mpicc search.c
search.c
+ qstat -rn
```

nfs1.osc.edu:

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Elap Time
17435.nfs1.osc.	dje	parallel	search	15382	2	--	--	00:04	R	--

mck076/1+mck076/0+mck075/1+mck075/0

```
+ mpiexec a.out
+ < data3.txt
P:3 I searched up to index 1662
P:0 11 found at index=1499
P:0 I searched up to index 1499
P:1 I searched up to index 1505
P:2 I searched up to index 1658
+ rm a.out
+ rm search.o
```

Parallel Processing for SMPs

- Specifying a processor count for the Altix and Sun systems is slightly different than on the clusters -- you use ncpus=N instead of nodes=N
- For example, the following would work on the Altix:

```
#PBS -l ncpus=8
#PBS -l walltime=10:00:00
#PBS -j oe
#PBS -N openmp
#PBS -S /bin/ksh
cd $TMPDIR
cp $HOME/openmp/a.out .
export OMP_NUM_THREADS=8
./a.out
```

Setting Execution Day & Time

- Is there a way I can tell PBS not to begin executing my job until a certain date and time?
- Answer:yes Use the header line:
`#PBS -a [YYYY][MM][DD]hhmm`
- This is the standard “(a)t” option. As in shown in syntax only the hours and minutes must be set. If the time set has already passed, PBS will assume the date is tomorrow
- Let say I wanted to pick a time in which a machine was not very busy. Say, this Saturday at 5 am. The header line would look like this:
`#PBS -a 200305240500`
- You can submit the job today, and it will be put in the (W)ait state until the date&time indicated. The output of `qstat -a` for a timed job looks as follows:

```
17450.nfs1.osc. osu2917  serial  hpmulti_de  16440  1  --  --  240:0 R 00:13
17453.nfs1.osc. dje      batch    dt        --  --  --  --  00:04 W  --
17454.nfs1.osc. utl0192  parallel fractaltda  31330  8  --  --  01:05 R  --
```

Queue Specification

- On many HPC systems special queues are set up and can only be used by permitted users. These queues might allow a huge amount of memory or time; a large number of parallel processors; and/or access to a third-party scientific/engineering software package.
- For example, on the Itanium 2 cluster there is special queue called “test” which is only open to the cluster system administration group
- If you have permission, you can specify the queue for your job with the header line

`#PBS -q queue_name`

Otherwise, just let PBS put your job in the appropriate queue.

Useful Batch Environmental Variables

- Using the /tmp directory
- Changing to your work directory
- Information Environment Variables

TMPDIR

- For many user the sizes of data files or executable files are so large they cannot be used in their home directories.
- The /tmp directory offers a huge amount of temporary disk space to all users of an OSC system. In addition, it is faster to access then \$HOME disk since it is in local memory.
- For a batch job, there is a unique subdirectory of /tmp associated just with that job. It comes into existence when the job is run and is deleted when the job is finished. The name of the /tmp subdirectory is stored in the environment variable TMPDIR
- In the batch file the user should copy all files needed to \$TMPDIR, cd to \$TMPDIR and run your code, and finally bring back output files to your \$HOME area.
- Note that “clean-up” at the end of the batch file is not needed since the \$TMPDIR directory and all its files are deleted when the job ends.

TMPDIR

- For some time the motd on oscbw.osc.edu has contained the following:

Some csh/tcsh users have reported problems with jobs prematurely aborting. The symptom is a message such as:

```
tset: standard error: Inappropriate ioctl for device
```

This reflects an effort to set your terminal type in a batch job. If you see this message, change your ~/.login file (look also at your ~/.cshrc) in the following way:

```
# Test and set UNIX terminal type
if ($ENVIRONMENT != "BATCH") then
  tset -Q -I
  if ( $status != 0 || "$TERM" == "unknown" ) then
    echo -n "Enter UNIX terminal type, "
    eval `tset -I -Q -s -m "?:vt100" | fgrep ' TERM '`
  endif
endif
```

Batch File using TMPDIR

```
#PBS -l walltime=00:04:00
#PBS -N nest
#PBS -j oe
set -x
cd batchshop/env
cp nest_stdio.c input $TMPDIR
cd $TMPDIR
cc nest_stdio.c
./a.out < input > output
cp output $HOME/batchshop/env
cd $HOME/batchshop/env
cat input
cat output
```

Returned Log File

```
+ cd batchshop/env
+ cp nest_stdio.c input
  /tmp/pbstmp.17478.nfs1.osc.edu
+ cd /tmp/pbstmp.17478.nfs1.osc.edu
+ cc nest_stdio.c
+ ./a.out
+ < input
+ > output
+ cp output /home/dje/batchshop/env
+ cd /home/dje/batchshop/env
+ cat input
4 8
+ cat output
Enter Loop Counts
For loop counts N=4 M=8
Sum=944
```

PBS_O_WORKDIR

- Is there a way that PBS can automatically cd to my working directory since I always start out in my home directory?
- Answer:mostly Once a user has used qsub to submit a batch job, the environment variable PBS_O_WORKDIR is filled with the absolute path of the directory from which qsub was executed.
- Usually, where the user has the files the batch job needs to work on is also where they submit from. Thus, the first line of their batch file can be made general purpose
`cd $PBS_O_WORKDIR`
- The user doesn't even have to remember the path to directory they are working in.

General Purpose Batch File

```
#PBS -l walltime=00:04:00
#PBS -N nest
#PBS -j oe
set -x
cd $PBS_O_WORKDIR
cc nest_stdin.c
./a.out < input
rm a.out
```

- **Log file returned:**

```
+ cd /a/dje/batchshop/env
+ cc nest_stdin.c
+ ./a.out
+ < input
Enter Loop Counts
For loop counts N=4 M=8
Sum=944
+ rm a.out
```

Information Variables

- **PBS has a number of built-in environment variables that preserve job information.**
 - PBS_O_HOST = hostname of machine running PBS
 - PBS_O_QUEUE = starting queue your job was put in
 - PBS_QUEUE = queue your job was executed in
 - PBS_JOBID = JID of your job
 - PBS_JOBNAME = “internal” name you gave job
 - PBS_NODEFILE = name of the file containing list of nodes your job used
- **On the next slide is a batch file and its return log file that show that these variables are filled with the correct values**

Sample Info Log File

```
+ cd /a/dje/batchshop/env  
+ qstat -rn
```

```
nfs1.osc.edu:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	S	Elap Time
17488.nfs1.osc. mck147/1	dje	serial	nest	13858	1	--	--	00:04	R	--

```
+ print mck-login1.osc.edu (PBS_O_HOST)  
mck-login1.osc.edu  
+ print batch (PBS_O_QUEUE)  
batch  
+ print serial (PBS_QUEUE)  
serial  
+ print 17488.nfs1.osc.edu (PBS_JOBID)  
17488.nfs1.osc.edu  
+ print nest (PBS_JOBNAME)  
nest  
+ cat /var/spool/pbs-mck/aux/17488.nfs1.osc.edu (PBS_NODEFILE)  
mck147
```

More PBS Commands

- `qpeek`
- `qstat` (more options)
- **Maui Scheduler Commands**

qpeek

- `qpeek` is a well-named command. It allows for the user to “peek” into the partially-completed log file of a running job. Thus, the user can see the progress of the job.
- On the next slide, the `qpeek` command is used to see at what batch file command the following job is at:

```
#PBS -l walltime=00:20:00
#PBS -N liver
#PBS -j oe
set -x
cd liver_ia64
./liver
rm liver
```

qpeek Demonstration

```
mck-login1:$ qstat -a
```

```
nfs1.osc.edu:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
17547.nfs1.osc.	dje	serial	liver	18385	1	--	--	00:20	Q	--

```
mck-login1:$ qpeek 17547
```

```
Job 17551 is not running!
```

```
mck-login1:$ qpeek 17547
```

```
+ cd liver_ia64
```

```
mck-login1:$ qpeek 17547
```

```
+ cd liver_ia64
```

```
+ ./liver
```

```
mck-login1:$ qpeek 17547
```

```
+ cd liver_ia64
```

```
+ ./liver
```

```
mck-login1:$ qpeek 17547
```

```
qstat: Unknown Job Id 17547.nfs1.osc.edu
```

```
Job 17547 is not running!
```

qstat

- **By using the qstat command the user can get a vast amount of information about jobs (as we have already seen) and the batch system queues themselves.**
- **We have already used there qstat options:**
 - a show status info on all jobs
 - r show status info on running jobs only
 - n show the nodes jobs are running on
- **The new options are used to check how busy the queues are and what the queue limits/properties are**
 - Q summary of load on each of the queues
 - q summary of limits on each queues
 - Qf | more detailed description of all queue properties

Itanium 2 Cluster Queues

```
mck-login1:~$ qstat -Q
```

Queue	Max	Tot	Ena	Str	Que	Run	Hld	Wat	Trn	Ext	Type
dedicated	0	4	yes	yes	4	0	0	0	0	0	Execution
batch	0	0	yes	yes	0	0	0	0	0	0	Route
parallel	0	15	yes	yes	7	8	0	0	0	0	Execution
serial	0	12	yes	yes	0	12	0	0	0	0	Execution
test	0	0	yes	yes	0	0	0	0	0	0	Execution
smp	0	0	no	no	0	0	0	0	0	0	Execution

```
mck-login1:~$ qstat -q
```

```
server: nfs1
```

Queue	Memory	CPU Time	Walltime	Node	Run	Que	Lm	State
dedicated	--	--	320:00:0	128	0	4	--	E R
batch	--	--	--	--	0	0	--	E R
parallel	--	--	320:00:0	64	8	7	--	E R
serial	--	--	320:00:0	1	12	0	--	E R
test	--	--	--	--	0	0	--	E R
smp	--	--	320:00:0	1	0	0	--	E R
					---	---		
					20	11		

Full Queue Description

```
mck-login1:$ qstat -Qf parallel
```

```
Queue: parallel
queue_type = Execution
total_jobs = 16
state_count = Transit:0 Queued:8 Held:0 Waiting:0 Running:8 Exiting:0
resources_max.nodect = 64
resources_max.nodes = 64:ppn=2
resources_max.vmem = 11800mb
resources_max.walltime = 320:00:00
resources_min.nodect = 2
resources_default.nodes = 2:ppn=1
resources_default.vmem = 3800mb
resources_default.walltime = 01:00:00
resources_assigned.mem = 0b
resources_assigned.nodect = 116
resources_assigned.vmem = 1837105153b
enabled = True
started = True
```

Pentium 4 Cluster Queues

```
piv-login1:~$ qstat -Q
```

Queue	Max	Tot	Ena	Str	Que	Run	Hld	Wat	Trn	Ext	Type
batch	32	0	no	no	0	0	0	0	0	0	Route
serial	0	0	no	no	0	0	0	0	0	0	Execution
parallel	0	0	no	no	0	0	0	0	0	0	Execution
test	0	0	yes	yes	0	0	0	0	0	0	Execution
amd	0	0	yes	yes	0	0	0	0	0	0	Route
amddedicated	0	0	yes	yes	0	0	0	0	0	0	Execution
amdserial	0	64	yes	yes	0	64	0	0	0	0	Execution
amdparallel	0	5	yes	yes	0	5	0	0	0	0	Execution
maintenance	0	0	yes	yes	0	0	0	0	0	0	Execution
matlab	0	0	yes	yes	0	0	0	0	0	0	Execution

```
piv-login1:~$ qstat -q
```

```
server: nfs1.osc.edu
```

Queue	Memory	CPU	Time	Walltime	Node	Run	Que	Lm	State
batch	--	--	--	--	--	0	0	32	D S
serial	--	160:00:0	320:00:0	320:00:0	1	0	0	--	D S
parallel	30gb	160:00:0	320:00:0	320:00:0	16	0	0	--	D S
test	--	--	--	--	--	0	0	--	E R
amd	--	--	--	--	--	0	0	--	E R
amddedicated	--	--	168:00:0	168:00:0	112	0	0	--	E R
amdserial	--	--	510:00:0	510:00:0	1	64	0	--	E R
amdparallel	--	--	168:00:0	168:00:0	48	5	0	--	E R
maintenance	--	--	--	--	--	0	0	--	E R
matlab	--	--	320:00:0	320:00:0	1	0	0	--	E R
						69	0		

Full Queue Description

```
mck:$ qstat -Qf parallel
```

```
Queue: parallel
queue_type = Execution
total_jobs = 5
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:6
Exiting:0
resources_max.nodect = 64
resources_max.nodes = 64:ppn=2
resources_max.walltime = 320:00:00
resources_min.cpus = 2
resources_default.nodect = 2
resources_default.nodes = 2:ppn=1
resources_default.walltime = 01:00:00
resources_assigned.nodect = 66
enabled = True
started = True
```

SunFire 6800 Queues

```
coe3:$ qstat -Q
```

Queue	Max	Tot	Ena	Str	Que	Run	Hld	Wat	Trn	Ext	Type
parallel	0	1	yes	yes	0	1	0	0	0	0	Execution
batch	0	0	no	no	0	0	0	0	0	0	Execution
serial	0	8	yes	yes	0	8	0	0	0	0	Execution
dedicated	0	0	yes	yes	0	0	0	0	0	0	Execution
coe	0	0	yes	yes	0	0	0	0	0	0	Route

```
coe3:$ qstat -q
```

```
server: coe3
```

Queue	Memory	CPU Time	Walltime	Node	Run	Que	Im	State
parallel	--	2000:00:	--	--	1	0	--	E R
batch	--	2000:00:	--	1	0	0	--	D S
serial	--	2000:00:	--	1	8	0	--	E R
dedicated	--	2000:00:	--	--	0	0	--	E R
coe	--	--	--	--	0	0	--	E R
					---	---		
					9	0		

Full Queue Description

```
coe3:$ qstat -Qf dedicated
```

```
Queue: dedicated  
queue_type = Execution  
total_jobs = 0  
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0  
resources_max.cput = 2000:00:00  
resources_max.ncpus = 24  
resources_max.vmem = 48gb  
resources_min.ncpus = 8  
resources_default.cput = 01:00:00  
resources_default.vmem = 4gb  
resources_assigned.ncpus = 0  
enabled = True  
started = True
```

Maui Scheduler Commands

- **OSC PBS software has been enhanced with the use of the Maui Scheduler to improve job flow.**
 - **Advance reservations**
 - **Backfill scheduling**
 - **Fairshare and quality-of-service (QOS) levels**
- **Maui also comes with its own set of useful commands:**
 - **showq** (list currently running and queued jobs)
 - **showstart** (estimates start time for a queued job)
 - **showbf** (tells what processors are available to “back-fill” the system)

Maui Scheduling Algorithm

- **Compute priorities for all jobs not currently running.**
- **Sort idle jobs in priority order from highest to lowest, removing any jobs which have had holds place on them or exceed policy limits.**
- **Starting with the highest priority job, attempt to run each job until there are not enough resources available to run the highest priority job remaining.**
- **Given current system conditions, compute when is the soonest time the highest priority job could run, and create a reservation for it at that time.**
- **Backfill any other idle jobs which will not cause the start time for the highest priority job to slip further into the future.**

Factors in Job Priority

- Recent usage
- Processor count
- How long the job has been queued
- Expansion factor (ratio of job length to queue time)

These factors tend to favor large processor-count, long-running jobs, as those are the most difficult to schedule. Smaller processor-count and/or shorter-running jobs are filled in using backfill scheduling.

showq Output

```
ACTIVE JOBS-----
```

JOBNAME	USERNAME	STATE	PROC	REMAINING	STARTTIME
17416	osu2614	Running	2	12:13:03	Mon May 19 09:07:50
17490	osu3111	Running	1	19:23:39	Mon May 19 16:18:26
17393	osu3250	Running	32	1:06:28:13	Sun May 18 21:23:00
17491	osu3230	Running	8	1:15:36:39	Mon May 19 16:31:26
17417	osu3250	Running	32	1:18:20:34	Mon May 19 09:15:21
17315	ucn012	Running	22	3:01:49:28	Sun May 18 14:44:15
17324	osu2614	Running	2	3:02:12:46	Sun May 18 15:07:33
17325	osu2614	Running	2	3:02:13:02	Sun May 18 15:07:49
17328	osu2614	Running	2	3:02:13:36	Sun May 18 15:08:23
17308	ucn012	Running	18	4:17:38:04	Sat May 17 18:32:51
17450	osu2917	Running	1	9:18:40:57	Mon May 19 11:35:44
17303	troy	Running	2	11:05:21:23	Sat May 17 14:16:10

12 Active Jobs 124 of 292 Processors Active (42.47%)
 63 of 146 Nodes Active (43.15%)

IDLE JOBS-----

JOBNAME	USERNAME	STATE	PROC	WCLIMIT		QUEUETIME
16346	watts	Idle	226	3:00:00	Fri May 9	08:50:16
16345	watts	Idle	196	3:00:00	Fri May 9	08:49:37
16344	watts	Idle	170	3:00:00	Fri May 9	08:48:46
16718	watts	Idle	256	20:00:00	Mon May 12	14:32:14
17357	ucn0733	Idle	64	0:15:00	Sun May 18	16:50:23
17470	ucn0733	Idle	32	0:20:00	Mon May 19	14:31:52
17186	osu2779	Idle	32	2:00:00:00	Thu May 15	13:54:18

7 Idle Jobs

BLOCKED JOBS-----

JOBNAME	USERNAME	STATE	PROC	WCLIMIT		QUEUETIME
---------	----------	-------	------	---------	--	-----------

Total Jobs: 19 Active Jobs: 12 Idle Jobs: 7 Blocked Jobs: 0

showstart Usage

```
mck-login1:$ showstart 16346
```

```
job 16346 requires 226 procs for 3:00:00  
Earliest start in          3:01:46:03 on Thu May 22 18:44:15  
Earliest completion in    3:04:46:03 on Thu May 22 21:44:15  
Best Partition: DEFAULT
```

showbf Usage

```
mck-login1:$ showbf
```

```
backfill window (user: 'dje' group: 'G-0541' partition: ALL)
```

```
Tue May 20 13:41:03
```

```
no procs available
```

Why Won't My Job Run?

There are a number of reasons why your job may not run immediately, even if there appears to be sufficient resources for it to run:

- Other users' jobs may have been assigned higher priority than your job, depending on what you're asking for. Especially in cases with small processor-count, long-running jobs, the scheduler may not be able to backfill the smaller job without interfering with a higher priority job's start time.
- There may be downtime or other system reservations in place. These will often be noted in the system's message of the day (`/etc/motd`) and/or the OSC notices web page (<http://oscinfo.osc.edu/notices/>).
- You or your group may be at the maximum CPU count or running job count for a user or group. These are generally set up such that a single user can use up to 1/2 of a given system, and that a single group can use up to 3/4 of the system.

Reporting Batch Problems to OSC Help

If you are having a problem with the batch system on any of OSC's machines, you should send email about it to oschelp@osc.edu. Including the following information will aid OSC's Science and Technology Support (STS) staff in diagnosing your problem quickly:

- User ID
- Name of the system you're submitting jobs to (oscbw, mck, coe)
- Job ID
- Job script
- Job output and/or error messages (preferably in context).