

Table of Contents

<u>PBS on Pleiades</u>	1
<u>Overview</u>	1
<u>Queue Structure</u>	2
<u>Mission Shares Policy on Pleiades</u>	4
<u>Resources Request Examples</u>	6
<u>Default Variables Set by PBS</u>	9
<u>Sample PBS Script for Pleiades</u>	10
<u>Pleiades devel Queue</u>	11

PBS on Pleiades

Overview

Overview

On Pleiades, PBS (version 11.2) is used to manage batch jobs that run on the compute nodes (4 different processor types, 11,776 nodes, and 126,720 cores in total). PBS features that are common to all NAS systems are described in other articles. Read the following articles for Pleiades-specific PBS information:

- [Queue Structure](#)
- [Resource Request Examples](#)
- [Default Variables Set by PBS](#)
- [Sample PBS Scripts](#)

Queue Structure

You should be aware of the PBS queue structure to help with batch job management. To find the maximum and default NCPUS (number of CPUs), the maximum and default wall-time, the priority of the queue, and whether the queue is disabled or stopped, use the command:

```
% qstat -Q
```

This command also provides statistics of jobs in the states of queued (Q), held (H), running (R), or exiting (E).

Note that the queue structure will change from time to time. Below is a snapshot of the output from this command on August 21, 2012.

```
%qstat -Q
Queue      Ncpus/      Time/      State counts
name       max/def     max/def    jm T/_Q/_H/W/_R/E/B pr  Info
=====
alphanst   --/ --     120:00/ 01:00 -- 0/_0/_0/0/_0/0/0 0 disabled stopped
dpr        --/ 8       --/ 00:10 -- 0/_0/_0/0/_0/0/0 0 disabled
smd2       2048/ 8     12:00/ 01:00 -- 0/_0/_0/0/_0/0/0 16 disabled
diags      --/ --     120:00/ 04:00 -- 0/_0/_0/0/_0/0/0 0
wide       --/ --     120:00/ 01:00 -- 0/_0/_0/0/_3/0/0 101 disabled
rtf        --/ 8       --/ 01:00 -- 0/_0/_0/0/_0/0/0 65 disabled
somd_spl   --/ 8     240:00/ 01:00 -- 0/_0/_0/0/_0/0/0 25 disabled
heomd_spl  --/ 8     120:00/ 01:00 -- 0/_0/_0/0/_2/0/0 45
smd1       --/ 8     240:00/   -- -- 0/_0/_0/0/_0/0/0 16
vlong      8192/ 8     384:00/120:00 -- 0/_1/_0/0/_1/0/0 0
arnd_spl   4900/ 8     120:00/ 01:00 -- 0/_0/_0/0/_0/0/0 15 disabled
ded_time   --/ --       --/ 01:00 -- 0/_0/_1/0/_0/0/0 0
idle       --/ --       --/   -- -- 0/_0/_0/0/_0/0/0 0 disabled
testing    --/ --       --/ 00:30 -- 0/_0/_0/0/_0/0/0 0
sls_aero1  --/ 8     48:00/ 01:00 -- 0/_60/_0/0/_56/0/0 45
kepler     --/ 8     120:00/ 01:00 -- 0/_0/_0/0/_0/0/0 101
sls_aero2  408/ 8     144:00/ 01:00 -- 0/_0/_0/0/_0/0/0 45 disabled
low      --/ 8     04:00/ 00:30 -- 0/_0/_0/0/_0/0/0 -10
gpu_long_free --/ 24    24:00/ 01:00 -- 0/_0/_0/0/_0/0/0 0
normal   --/ 8     08:00/ 01:00 -- 0/_77/17/0/104/0/0 0
long     8192/ 8     120:00/ 01:00 -- 0/156/_3/0/338/0/0 0
debug    1025/ 8     02:00/ 00:30 -- 0/_6/_0/0/_7/0/0 15
route      --/ 8       --/   -- -- 0/_0/_2/0/_0/0/0 0
devel    4800/ 1     02:00/   -- -- 0/_22/_0/0/_9/0/0 60
gpu      --/ 8     08:00/ 01:00 -- 0/_1/_0/0/_0/0/0 0
```

To view more information about each queue, use:

```
% qstat -fQ queue_name
```

In the output of this command, you will find additional information such as:

acl_groups

Lists all GIDs that are allowed to run in the queue. For the **normal**, **debug**, **long**, **low**, and **devel** queues, all GIDs should be included. Special queues, such as **esmd_spl**, **arnd_spl**, **somd_spl**, **clv_spl**, etc., typically allow a few GIDs only.

default_chunk.model

Specifies the default processor model, if you do not specify the processor model yourself. All queues, except **devel** and **gpu**, default to using nodes with Harpertown model processors. See [Pleiades devel Queue](#) for the defaults for the **devel** queue and [GPU Basics](#) for more information on the **gpu** queue.

resources_min.ncpus

If defined, specifies the minimum NCPUs required for the queue. The **wide** queue requires using a minimum of 1024 CPUs.

max_run

If defined, specifies the maximum number of jobs each user is allowed to run in the queue. For the **devel** queue, the maximum number is set at 10. For the **debug** queue, it is set at 2.

TIP: To request using the Nehalem-EP, Westmere, or Sandy Bridge nodes, use the attributes **model=neh**, **model=wes**, or **model=san** in your **resource_list**. To explicitly request Harpertown nodes, use **model=har**. You can apply the model attribute to any queue.

For example:

```
#PBS -q long
#PBS -l select=1:ncpus=12:model=wes
```

Mission Shares Policy on Pleiades

Mission Directorate shares have been implemented on Pleiades since Feb. 10, 2009. Implementing shares guarantees that each Mission Directorate gets its fair share of resources.

The share to which a job is assigned is based on the GID used by the job. Once all the cores within a Mission Directorate's share have been assigned, other jobs assigned to that share must wait, even if cores are available in a different Mission Directorate's share, with the following exception:

When a Mission Directorate is not using all of its cores, other Mission Directorates can borrow those cores, but only for jobs that will finish within 4 hours. When part of the resource is unavailable, the total number of cores decreases, and each Mission Directorate loses a proportionate number of cores.

You can display the share distribution by adding the `-W shares=` option to the `qstat` command. For example:

```
%qstat -W shares=
```

Group	Share%	Use%	Share	Exempt	Use	Avail	Borrowed	Ratio	Waiting
Overall	100	0	159748	0	960	158788	0	0.01	960
ARMD	24	18	38109	0	29680	8429	0	0.78	22512
HEOMD	23	21	36521	0	34312	2209	0	0.94	28416
SMD	51	50	80981	0	80968	13	0	1.00	113920
NAS	2	0	3175	0	0	3175	0	0.00	20240

Mission shares are calculated by combining the mission's HECC share of the shared assets combined with the mission-specific assets. The mission shares on Oct 3, 2011 are shown in the second column of the above display. The amount of resources used and borrowed by each mission and resources each mission is waiting for are also displayed.

An in-house utility, `qs`, provide similar information with details that break the resources into the Harpertown, Nehalem-EP, Westmere, and Sandy Bridge processor types and is available at `/u/scicon/tools/bin/qs`.

The `-h` option of `qs` provides instructions on how to use it:

```
% /u/scicon/tools/bin/qs -h
```

```
usage: qs [-u] [-n N] [-b] [-p] [-d] [-r] [-f M,N] [-q N] [-t] [-v] [-h] [--file f]
        -u          : show used resources only; don't show queued jobs
        -n N        : show time remaining before N nodes are free
        -b          : order segments in bars to help understand borrowing
        -p          : plain output: i.e. no colors or highlights
```


Resources Request Examples

Since Pleiades consists of four different processor types (Harpertown, Nehalem-EP, Westmere, and Sandy Bridge), you will benefit from keeping the following in mind when requesting PBS resources for your job:

Charging on the usage of the four Pleiades processor types is based on a common Standard Billing Unit which is on a per-node basis. The SBU rate for each of the Pleiades processor types is:

Processor Type	SBU Rate (per node)
Sandy Bridge	1.82 (16 cores in a node)
Westmere	1.0 (12 cores in a node)
Nehalem-EP	0.80 (8 cores in a node)
Harpertown	0.45 (8 cores in a node)

The actual amount of memory per node through PBS is slightly less than 7.6 GB per node for Harpertown, 22.5 GB per node for Nehalem-EP and Westmere, and about 31 GB per node for Sandy Bridge.

For the **normal**, **long**, and **debug** queue, use the **model=[har, neh, wes, san]** attribute to request the processor type(s) for your job. If the processor type is not specified in your PBS resource list, the job is routed to use the default processor type, Harpertown, for most queues. For the **devel** queue, see Pleiades devel queue for more information. For the **gpu** queue, see GPU Basics.

Example 1

Here are some examples of requesting certain processor models for a 128-process job:

```
#PBS -l select=16:ncpus=8:model=har
#to run all 8 cores on each of 16 Harpertown nodes

#PBS -l select=32:ncpus=4:model=har
#to run on only 4 cores on each of 32 Harpertown nodes
#(note: will be charged for 32 nodes = 256 cores)

#PBS -l select=16:ncpus=8:model=neh
#to run all 8 cores on each of 16 Nehalem-EP nodes

#PBS -l select=11:ncpus=12:model=wes
#to run all 12 cores on each of 11 Westmere nodes
#(4 cores in 11th node will go unused)

#PBS -l select=8:ncpus=16:model=san
#to run all 16 cores on each of 8 Sandy Bridge nodes
```

Note that you can specify both the queue type (`-q normal, debug, long, low`) and the processor type (`-l model=har, neh, wes, san`). For example:

```
#PBS -q normal
#PBS -l select=16:ncpus=8:model=neh
```

If your application can run on any of the four processor types, you may want to submit your job to a processor type that has more nodes unoccupied by other running jobs. Doing this can possibly reduce the wait time of your job. The script `node_stats.sh` provides information about the total, used, and free nodes for each processor type. For example:

```
% /u/scicon/tools/bin/node_stats.sh
```

```
Pleiades Nodes Total: 11240
Pleiades Nodes Used : 10485
Pleiades Nodes Free : 755
```

```
Harpertown   Total: 4063, Used: 4015, Free: 48
Nehalem      Total: 1253, Used: 847, Free: 406
Westmere     Total: 3957, Used: 3916, Free: 41
SandyBridge  Total: 1288, Used: 1128, Free: 160
GPU nodes    Total: 56, Used: 2, Free: 54
Devel queue  Total: 623, Used: 577, Free: 46
```

```
Currently queued jobs are requesting: 8259 Harpertown, 2142 Nehalem, 9633 Westmere,
```

TIP: Add `/u/scicon/tools/bin` to your path in `.cshrc` or other shell start-up scripts to avoid having to type in the complete path for this tool.

For each job, you can also identify which processor models are used by looking at the "Model" field of the output of the command:

```
% qstat -a -W o=+model
```

Example 2

The Harpertown nodes in rack 32 have 16 GB memory/node instead of 8 GB per node.

This example shows a request of 2 nodes with bigmem in rack 32.

```
#PBS -l select=2:ncpus=8:bigmem=true:model=har
```

Example 3

For a multi-node PBS job, the NCPUs used in each node can be different. This is useful for jobs that need more memory for some processes, but less for other processes. Resource requests can be done in "chunks" for a job with varying NCPUs per node.

This example shows a request of two resource chunks. In the first chunk, 1 CPU in 1 node, and in the second chunk, 8 CPUs in each of three other nodes are requested:

```
#PBS -l select=1:ncpus=1+3:ncpus=8
```

Example 4

A PBS job can run across different processor types. This can be useful in two scenarios:

- When you cannot find enough free nodes within one model for your job
- When some of your processes need more memory while others need much less

This can be accomplished by specifying the resources in chunks, with one chunk requesting one processor type and another chunk requestings a different processor type.

Here is an example of how to request 1 Westmere node (which provides 24 GB per node) and 3 Harpertown nodes (which provides 8 GB per node), under either of the following circumstances:

```
#PBS -lplace=scatter:excl:group=model
```

```
#PBS -lselect=1:ncpus=12:mpiprocs=12:model=wes+3:ncpus=8:mpiprocs=8:model=har
```

Default Variables Set by PBS

You can use the `env` command--either in a PBS script or on the command line of a PBS interactive session--to find out what environment variables are set within a PBS job. In addition to the PBS_xxxx variables, the following two are useful to know:

NCPUS

Defaults to number of CPUs that you requested for the node.

OMP_NUM_THREADS

Defaults to 1 unless you explicitly set it to a different number. If your PBS job runs an OpenMP or MPI/OpenMP application, this variable sets the number of threads in the parallel region.

FORT_BUFFERED

Defaults to 1. Setting this variable to 1 enables records to be accumulated in the buffer and flushed to disk later.

Sample PBS Script for Pleiades

```
#PBS -S /bin/csh
#PBS -N cfd
# This example uses the Harpertown nodes
# User job can access ~7.6 GB of memory per Harpertown node.
# A memory intensive job that needs more than ~0.9 GB
# per process should use less than 8 cores per node
# to allow more memory per MPI process. This example
# asks for 64 nodes and 4 MPI processes per node.
# This request implies 64x4 = 256 MPI processes for the job.
#PBS -l select=64:ncpus=8:mpiprocs=4:model=har
#PBS -l walltime=4:00:00
#PBS -j oe
#PBS -W group_list=a0801
#PBS -m e

# Currently, there is no default compiler and MPI library set.
# You should load in the version you want.
# Currently, MVAPICH or SGI's MPT are available in 64-bit only,
# you should use a 64-bit version of the compiler.

module load comp-intel/11.1.072
module load mpi-sgi/mpt.2.04.10789

# By default, PBS executes your job from your home directory.
# However, you can use the environment variable
# PBS_O_WORKDIR to change to the directory where
# you submitted your job.

cd $PBS_O_WORKDIR

# use of dplace to pin processes to processors may improve performance
# Here you request to pin processes to processors 2, 3, 6, 7 of each node.
# This helps for using the Harpertown nodes, but not for Nehalem-EP or
# Westmere-EP nodes

# The resource request of select=64 and mpiprocs=4 implies
# that you want to have 256 MPI processes in total.
# If this is correct, you can omit the -np 256 for mpiexec
# that you might have used before.

mpiexec dplace -s1 -c2,3,6,7 ./grinder < run_input > output

# It is a good practice to write stderr and stdout to a file (ex: output)
# Otherwise, they will be written to the PBS stderr and stdout in /PBS/spool,
# which has limited amount of space. When /PBS/spool is filled up, any job
# that tries to write to /PBS/spool will die.

# -end of script-
```

Pleiades devel Queue

NAS provides a special **devel** queue that provides faster turnaround when doing development work.

Currently, 512 Westmere nodes and 144 Sandy Bridge nodes are reserved for the Pleiades **devel** queue, 24x7. The maximum walltime allowed is 2:00:00 and the maximum NCPUS is 4800. Each user is allowed to have only one job running in the **devel** queue at any one time.

To specify the Westmere processor type to be used for your job in the **devel** queue, do:

```
#PBS -l select=xx:ncpus=yy:model=wes
```

If you want to use the Sandy Bridge nodes, do:

```
#PBS -l select=xx:ncpus=yy:model=san
```

To submit your job to the **devel** queue, do:

```
pfe20% qsub -q devel job_script  
1234.pbspl1.nas.nasa.gov
```

To check the status of your job submitted to the **devel** queue, do:

```
pfe20% qstat devel -u your_username
```