Starting Cheyenne jobs

Users schedule jobs to run on the Cheyenne supercomputer by submitting them through the PBS Pro workload management system as described below.

Cheyenne and Casper users can also *submit PBS jobs from one system to another* and *craft job-dependency rules between jobs on both systems*. Review this documentation to learn how to use those capabilities. Also see Propagating environment settings to a PBS job for a description of a related change that was made when the capabilities were introduced.

To submit jobs to run on Casper *data analysis and visualization nodes*, see this documentation instead.

Common causes of job failures

Page contents

PBS job basics

Batch jobs

To submit a *batch* job to the Cheyenne queues, use the **qsub** command followed by the name of your PBS batch script file.

```
qsub script_name
```

Alternative - qcmd: For running *resource-intensive tasks* such as CESM and WRF builds or other compiles with three or more threads, CISL provides **qcmd**. This script starts a non-interactive job on a single batch node in the Cheyenne "regular" queue with a default wall-clock time of 1 hour. The first CESM example below uses the default settings. The second shows how to specify a longer wall-clock time if needed.

```
qcmd -- ./case.build
qcmd -l walltime=12:00:00 -- ./case.build
```

Output from the build script is printed to the user's terminal window.

Interactive jobs

To start an *interactive* job on Cheyenne, use the **qsub** command as in this example, which includes a *select statement.*

```
qsub -X -I -l select=1:ncpus=36:mpiprocs=36 -l walltime=01:00:00 -q regular -A project_code
```

Alternative - qinteractive: You can start an interactive job in the share queue for one hour by executing the CISL "qinteractive" script. The script is available if the **ncarenv** environment module is loaded, as it is by default.

```
qinteractive -X
```

You can also customize the job by including additional arguments, as in this example:

```
qinteractive -X -l select=2:ncpus=10:mpiprocs=10
```

Be sure to include **-X** as shown in the examples in order to use X11. This assumes you logged in with -X (or -Y) as well.

Specifying a project code

If you have more than one project to which you can charge your work, specify which project to charge by setting your **PBS_ACCOUNT** environment variable to the appropriate project code. Otherwise, the system will choose a project code randomly. You can set that in your start files for convenience.

Resource requests and restrictions

Some resource *requests* and *restrictions* are specified "per chunk" in a select statement. On Cheyenne, a chunk generally means a node, and your select statement defines the resources you need on each node. The first select statement in the interactive example above requests one chunk of resources and specifies the number of CPUs and MPI processes on the node.

Here are some additional examples.

Per-chunk resource example	Type	Description
-l select=1:mpiprocs=36	Request	Specify the number of MPI tasks you wish to utilize with the loaded MPI library
-l select=1:ncpus=36	Request	Specify the number of CPUs you wish to use on the node
-l select=1:ompthreads=36	Request	Specify the number of OpenMP threads to start on the node (defaults to ncpus if not set explicitly)
-l select=1:vmem=1GB	Restriction	Limit the amount of virtual memory for use by all concurrent processes on the node

Other resources are "global," which means that they affect all running processes/threads that you request for the job. Cheyenne jobs require a walltime *req uest* in the format shown, but resource *restrictions* are optional.

If a global resource *restriction* is exceeded during program execution, the job is terminated.

Global resource example	Type	Description
-l cput=10:00	Restriction	Limit the amount of CPU time a job can use across all CPUs
-l file=1GB	Restriction	Limit the maximum size of any single file that the job can create
-l pcput=10:00	Restriction	Limit the amount of CPU time the job can use on each CPU
-l pmem=2GB	Restriction	Limit the amount of physical memory that any single process can use
-l pvmem=2GB	Restriction	Limit the amount of virtual memory that the job can use
-l walltime=01:00:00	Request	Specify the amount of wall-clock time (**hours:minutes:seconds**) that the job can use (up to the queue limit)

Job scripts

Create your job script after you carefully consider: the system's usable memory and your program's memory requirements, which queue to specify, wall-clock time, and other parameters:

Your PBS job script can be no larger than 10 MB. Larger scripts, such as those that encode decadal climate data, should be modified to invoke a separate data encoding script.

Usable memory – The system's usable memory and *how to specify your memory requirements* in a job script are discussed below. To determine *how much memory your program requires to run*, see Checking memory use.

Job queues – Determine which queue is most appropriate for your job. Provide accurate wall-clock times to help fit your job into the earliest possible run opportunity.

Output and error logs –

- **DO** – Include the **-k eod** option to force output and error logs to be written to your GLADE working directory rather than stored by default in /tmp, which can fill and cause your job to fail without giving you clear error information. The programs and scripts you run in your batch job may produce large amounts of text output, especially when debugging errors. When you write those logs to GLADE instead of /tmp, the file size limit depends only on your disk quota. Run **gladequota** to check on your use of space.
- **DO NOT** – Don't write unwanted log and error output to restricted targets such as **/dev/null**, **/dev/shm**, **/var/tmp**, or **/var/spool/pbs**. The result will be an error message and rejection of your job. Direct any unwanted output files to your own /glade/scratch space and either delete them or wait for them to be purged.

Following is an example of a basic PBS script for running an MPI job. Each line that begins with #PBS is a PBS directive. Customize this and other sample scripts by substituting your own job name, project code, queue specification, email address, and so on where indicated.

```
#!/bin/tcsh
### bash users replace /tcsh with /bash -l
#PBS -N job_name
#PBS -A project_code
#PBS -l walltime=01:00:00
#PBS -q queue_name
#PBS -j oe
#PBS -k eod
#PBS -m abe
#PBS -M your_email_address
#PBS -l select=2:ncpus=36:mpiprocs=36

### Set TMPDIR as recommended
setenv TMPDIR /glade/scratch/$USER/temp
### bash users: export TMPDIR=/glade/scratch/$USER/temp
mkdir -p $TMPDIR

### Run the executable
mpiexec_mpt ./executable_name.exe
```

The **select statement** directive in the example indicates that the job requires:

- 2 chunks of resources (on non-shared Cheyenne queues, this is 2 nodes)
- 36 cores or individual processors (ncpus=36) on each node
- 36 MPI processes (mpiprocs=36) on each node.

CISL recommends that users **set TMPDIR** when running batch jobs on the Cheyenne compute nodes, as explained here: Storing temporary files with TMPDIR.

The **mpiexec_mpt** command is a wrapper script that sends information from PBS to the HPE Message Passing Toolkit (MPT), which includes the HPE MPI library. With some manual configuration, you can use **mpirun** instead. See Intel MPI and Open MPI.

Specifying use of large-memory nodes

Cheyenne nodes have either 64 GB of total memory (45 GB usable) or 128 GB of total memory (109 GB usable). If you need to ensure that a job runs on the large-memory nodes, include **mem=109GB** in your select statement as in this example. ***Do not*** *include it if you are submitting a job to the "share" queue*.

```
#PBS -l select=2:ncpus=36:mpiprocs=36:mem=109GB
```

Specify the use of large-memory nodes *only when your job requires it*. Limiting the types of nodes you can use can extend your wait time in the queue.

Loading modules in a batch script

Users sometimes need to execute module commands from within a batch job – to load an application, for example, or to load or remove other modules.

To ensure that the module commands are available, insert the following in your batch script if you need to include module commands.

In a tcsh script:

```
source /etc/profile.d/modules.csh
```

In a bash script:

```
source /etc/profile.d/modules.sh
```

Once that is included, you can add the **module purge** command to your script if necessary, then load just the modules that are needed to establish the software environment your job requires.

Using a login environment on batch nodes

It is possible to use the login environment for some jobs that you run on the Cheyenne system's exclusive-use batch nodes. For example, you might want to compile your code in the system's "regular" queue as opposed to the "share" queue, which limits jobs to using no more than 18 cores. Another job might need to use libraries that are available by default only on the login nodes and you also need to use more than 18 cores.

To run such a job on the batch nodes but using the login node environment, include this resource request as a PBS directive in your job script, or add it to your qsub command:

```
-l inception=login
```