Starting Casper jobs with PBS

This documentation describes how to use PBS Pro to submit jobs to run on nodes in the Casper cluster. *Unless GPUs are required, run jobs that require the use of more than one compute node on Cheyenne.*

Procedures for starting both **interactive jobs** and **batch jobs** on Casper are described below. Also:

- Compile your code on Casper nodes if you will run it on Casper.
- See Calculating charges to learn how core-hours charges are calculated for jobs that run on Casper.

Begin by logging in on Casper (casper.ucar.edu) or Cheyenne (cheyenne.ucar.edu).

Wall-clock

The wall-clock limit on the Casper cluster is 24 hours except as noted below.

Specify the hours your job needs as in the examples below. Use either the **hours:minutes:seconds** format or **minutes:seconds**.

Page contents

Interactive jobs

Starting a remote command shell with execcasper

Run the **execcasper** command to start an interactive job. Invoking it *without an argument* will start an interactive shell on the *first available HTC node*. The default wall-clock time is 6 hours.

To use another type of node, include a *select statement* specifying the resources you need. The **execcasper** command accepts all PBS flags and resource specifications as detailed by **man qsub**. Some common requests include:

- -A project_code (defaults to DAV_PROJECT value that you set in your start file)
- -l walltime=HH:MM:SS (defaults to 6 hours)
- -l select=1:ncpus=#:mpiprocs=#:ompthreads=#:mem=#GB:ngpus=#:cpu_type=skylake
  (or cascadelake)
- -l gpu_type=gp100 (or v100)
- -q gpudev (This will direct your interactive job to the "gpudev" queue rather than the default "casper" submission queue.)

Additionally, execcasper provides some convenience flags to simplify requesting job resources. If you specify a select statement as described above, it will take precedence over these flags.

- --nchunks=# - number of chunks specified in a select statement (e.g., select=<numchunks>)
- --ntasks=# - number of MPI tasks assigned to each chunk (mpiprocs)
- --nthreads=# - number of SMP threads assigned to each chunk (ompthreads)
- --mem=#GB - amount of memory in gigabytes assigned to each chunk
- --ngpus=# - number of GPUs assigned to each chunk
- --gpu=<type> to specify the gpu_type resource
- --cpu=<type> to specify the cpu_type resource

The following execcasper invocations, the first with a select statement and the second with a memory flag, are equivalent.

```
execcasper -A PROJ0001 -l select=1:ncpus=1:mem=20GB
execcasper -A PROJ0001 --mem=20GB
```

If you do not include a resource specification by using either a select statement or convenience flags, you will be assigned 1 CPU with 10 GB of memory and no GPUs.

If no project is assigned with either the **-A** option or the **DAV_PROJECT** environment variable, any valid project listed for your username will be chosen at random.

Starting a virtual desktop with vncmgr

If your work with complex programs such as MATLAB and VAPOR requires the use of virtual network computing (VNC) server and client software, use **vncmgr** instead of **execcasper**.

Using **vncmgr** simplifies configuring and running a VNC session in a Casper batch job. How to do that is documented here.

Batch jobs

Prepare a batch script by following one of the examples below. Most Casper batch jobs use the "casper" *submission* queue. The exception is for GPU development jobs, which are submitted to the "gpudev" submission queue.

Be aware that the system **does not** import your login environment by default, so make sure your script loads the software modules that you will need to run the job.

> **Caution:** Avoid using the PBS **-V** option to propagate your environment settings to the batch job; it can cause odd behaviors and job failures when used in submissions to Casper from Cheyenne. If you need to forward certain environment variables to your job, use the lower-case **-v** option to specify them. (See **man qsub** for details.)

When your job script is ready, use **qsub** to submit it from the Casper login nodes.

Examples:

```
qsub script_name
   or
qsubcasper script_name
```

See Managing and monitoring jobs for other commonly used PBS commands.

Specifying resource requests with select statements

Users can request certain resources from PBS via a select statement. This syntax allows you to request any number of resource *chunks*, which will include one or more CPUs, and optionally MPI tasks, Open MP threads, GPUs, custom amounts of node memory, and nodes with a certain CPU type. The general format of a select statement is:

```
#PBS -l select=<numchunks>:ncpus=#:<optional>=#...
```

Additionally, the types of GPUs assigned to the job can be constrained by an additional flag, as in this example:

```
#PBS -l gpu_type=gp100
```

Minimizing your use of resource specifications reduces the length of time your job waits in the queue. In general, ask for the smallest amount of each resource required to perform your task in a timely manner. Also, study the node table on the main Casper page to avoid requesting resource combinations that are not possible. For example, a job submission requesting both gp100s and Cascade Lake processors will sit in the queue indefinitely because the system does not offer that configuration.

Two types of GPUs are available:

- NVIDIA Tesla V100 GPUs for intensive GPGPU computing and machine learning/deep learning; a feature called *GPU isolation* limits the use of a particular GPU to a single user
- NVIDIA Quadro GP100 GPUs for visualization and light GPGPU workloads

---

GPU development jobs

A submission queue called "gpudev" is available between 8 a.m. and 5:30 p.m. Mountain time Monday to Friday to support application development and debugging efforts on general purpose and ML/AI GPU applications. This queue provides rapid access to up to 4 V100 GPUs, avoiding the sometimes lengthy queue wait times in the "gpgpu" execution queue.

Job submissions to this queue are limited to 30 minutes walltime instead of the 24-hour wallclock limit for all other submissions. **All jobs submitted to the queue** must request one or more V100 GPUs (up to four) in their resource directives. Node memory can be specified explicitly as usual, but by default jobs will be assigned N/4 of the total memory on a node, where N is the number of V100 GPUs requested.

Concurrent resource limits

Job limits are in place to ensure short dispatch times and a fair distribution of system resources. The specific limits that apply to your submission depend on the resources requested by your job. Based on your request, your submission will be classified as shown in the table.

Submission queue	Job category (execution queue)	Job resource requests	Limits
casper 24-hour wallclock limit	**largemem**	mem > 361 GB ncpus <= 36 ngpus = 0	Up to 5 jobs eligible for execution at any one time (more can be queued)
	htc	mem <= 361 GB ncpus <= 144 ngpus = 0	Up to 468 CPUs in use per user at any one time Up to 4680 GB memory per user at any one time (across all jobs in category)
	vis	gpu_type=gp100	Up to 2 GPUs in use per user at any one time Individual jobs are limited to a single gp100 (no multi-GPU jobs)
	gpgpu	gpu_type=v100	Up to 32 GPUs in use per user at any one time; users may submit jobs requesting more than 32 GPUs for execution on weekends.
gpudev 30-minute wallclock limit		ncpus <= 36 1 <= ngpus <= 4	Queue is only operational from 8 a.m. to 5:30 p.m. Mountain time, Monday to Friday. Users may have only one active job in the queue at any time.

NVMe node-local storage

Casper nodes each have 2 TB of local NVMe solid-state disk (SSD) storage. Some is used to augment memory to reduce the likelihood of jobs failing because of excessive memory use.

NVMe storage can also be used *while a job is running*. (Recommended only for I/O-intensive jobs.) Data stored in **/local_scratch/pbs.$PBS_JOBID** are deleted when the job ends.

To use this disk space while your job is running, include the following in your batch script after customizing as needed.

```
### Copy input data to NVMe (can check that it fits first using "df -h")
cp -r /glade/scratch/$USER/input_data /local_scratch/pbs.$PBS_JOBID

### Run script to process data (NCL example takes input and output paths as command line arguments)
ncl proc_data.ncl /local_scratch/pbs.$PBS_JOBID/input_data /local_scratch/pbs.$PBS_JOBID/output_data

### Move output data before the job ends and your output is deleted
mv /local_scratch/pbs.$PBS_JOBID/output_data /glade/scratch/$USER/
```

Script examples

The examples below show how to create a script for running a **high-throughput computing (HTC)** job. Such jobs typically use only a few CPU cores and likely do not require the use of an MPI library or GPU.

See this page for more script examples: Casper job script examples

For bash users

Insert your own project code where indicated and customize other settings as needed for your own job.

```
#!/bin/bash -l
### Job Name
#PBS -N htc_job
### Charging account
#PBS -A project_code
### Request one chunk of resources with 1 CPU and 10 GB of memory
#PBS -l select=1:ncpus=1:mem=10GB
### Allow job to run up to 30 minutes
#PBS -l walltime=30:00
### Route the job to the casper queue
#PBS -q casper
### Join output and error streams into single file
#PBS -j oe

export TMPDIR=/glade/scratch/$USER/temp
mkdir -p $TMPDIR

### Load Python module and activate NPL environment
module load ncarenv python
ncar_pylib

### Run analysis script
python myscript.py datafile.dat
```

For tcsh users

Insert your own project code where indicated and customize other settings as needed for your own job.

You may see this harmless warning at the start of your PBS batch job output logs on Casper:

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

It simply means terminal interaction via **stdin** is not possible because the script is being run in a batch environment. If you need to provide input via stdin, use **execcasper** to launch an interactive session instead of running a batch job.

```
#!/bin/tcsh
### Job Name
#PBS -N htc_job
### Charging account
#PBS -A project_code
### Request one chunk of resources with 1 CPU and 10 GB of memory
#PBS -l select=1:ncpus=1:mem=10GB
### Allow job to run up to 30 minutes
#PBS -l walltime=30:00
### Route the job to the casper queue
#PBS -q casper
### Join output and error streams into single file
#PBS -j oe

setenv TMPDIR /glade/scratch/$USER/temp
mkdir -p $TMPDIR

### Load Python module and activate NPL environment
module load ncarenv python
ncar_pylib

### Run analysis script
python myscript.py datafile.dat
```

Compiling your code

CISL recommends using the **default Intel, GNU or PGI compilers** for parallel programs, and compiling on a Skylake node unless the code will run exclusively on Cascade Lake nodes.

- Load the compiler.
- Load the **openmpi** module if you plan to use MPI.
- Compile your code as you usually do.

Serial programs can use any compiler.