Job Scheduling with PBS Pro

Brian Vanderwende
CISL Consulting Services

November 9, 2021

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

PBS Pro provides resources beyond the login nodes

- HPC compute nodes on Cheyenne
- High-throughput computing, high-memory, visualization, and GPGPU nodes on Casper
- JupyterHub jobs on both Cheyenne and Casper

Cheyenne and Casper each have their own unique PBS server that manages job scheduling. These servers are “peers” that can recognize each other:

You can now submit jobs to either system from any location on Cheyenne and Casper and create dependencies between them!

Why shouldn’t I just run my script on a login node?

Login nodes are a **shared resource** and so we expect and enforce fair usage of CPU cores and memory. Your session may be terminated if you run resource-intensive applications. Use login nodes for:

- **Script editing**
- **File movement**
- **Simple compiles (use 8 or less make jobs)**
- **Submitting jobs…**

Anatomy of a PBS compute job

A PBS job is a pool of requested resources with which you can run a *batch* script of commands or *interactively* run commands within a shell / interface

In PBS, resources are defined either at the *job level* or the *chunk level*:

- **Chunk** - setting defines the type of resources making up this particular portion (often N-nodes)
- **Job** - setting applies to ALL resource chunks in the allocated pool

Chunk resources
- **ncpus**
- mpiprocs
- ompthreads
- mem
- ngpus
- cpu_type

Job resources
- **walltime**
- place
- gpu_type

Important PBS terminal commands to remember

Job management commands:

- **qsub**	Submit batch scripts to a chosen job queue
- **qinteractive**	Submit interactive resource requests to a Cheyenne queue
- **execcasper**	Submit interactive resource requests to Casper queue
- **qdel**	Delete (cancel or kill) a pending or running job

Job query commands:

- **qstat**	Information about <u>recent</u> pending, running, or finished jobs
- **qhist**	Historical information about finished jobs only

Starting a batch job on Cheyenne with qsub

Submit: qsub my_chey_job.pbs

- Any #PBS *directives* can be overridden by qsub arguments
- Batch job will start in a clean environment (with your ~/.profile or ~/.tcshrc settings loaded)
- Job-specific environment settings should go into the script
- Once submitted, job will wait in specified queue until resources are available

```
#!/bin/bash
#PBS -A PROJ0001
#PBS -N chey_batch_job
#PBS -j oe
#PBS -k oed
#PBS -q regular
#PBS -l walltime=10:00:00
#PBS -l select=1:ncpus=8:mpiprocs=2:ompthreads=4

### Initialize job environment for application
export TMPDIR=/glade/scratch/$USER/temp
mkdir -p $TMPDIR
module purge
module load ncarenv gnu/9.1.0 mpt/2.22

### Run hybrid OpenMP+MPI application
mpiexec_mpt omplace ./app

### Store job statistics in log file
qstat -f $PBS_JOBID
```

Interactive jobs start a shell on a compute node

Use **qinteractive** and **execcasper** to start interactive jobs on **Cheyenne** and **Casper** respectively.

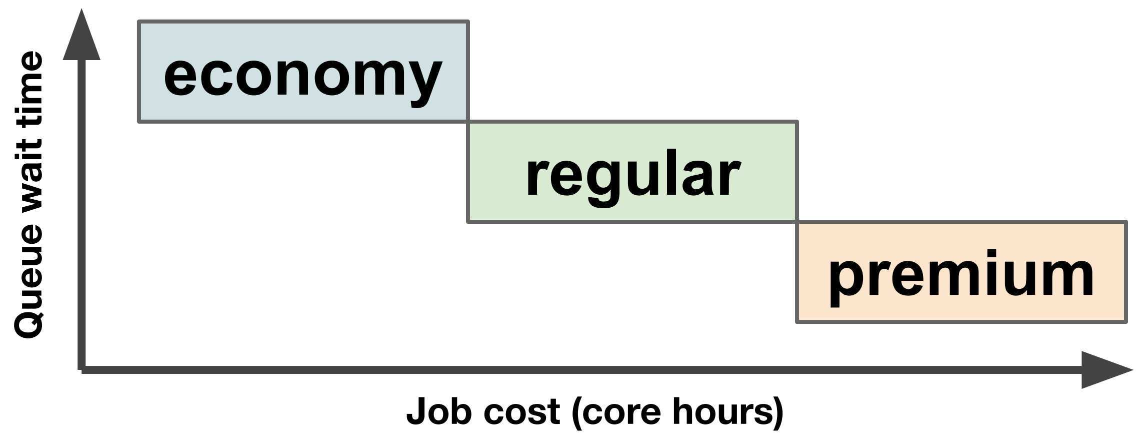
- Default settings give you 1 CPU core using your native shell (bash or tcsh) with **1 hr walltime on the share queue** or **6 hrs and 10 GB memory on casper**.
- Custom PBS settings can be passed to either command and short-form settings (listed on right) are provided as well

--nchunks=N
--ntasks=N
--nthreads=N
--mem=NGB
--cpu=type
--ngpus=1-8
--gpu=type

```
# These two calls to execcasper both request a single core with 20 GB of memory
cheyenne1$ execcasper -A PROJ0001 -l select=1:ncpus=1:mem=20GB
cheyenne1$ execcasper -A PROJ0001 --mem=20GB
```

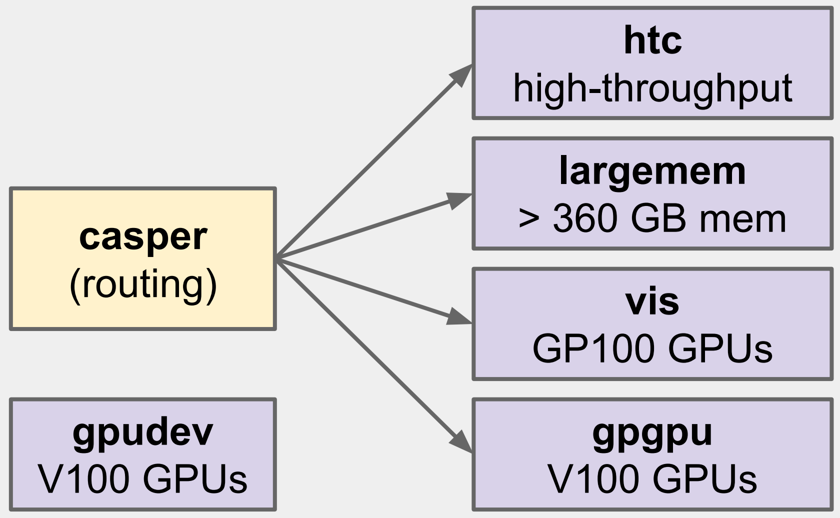
Queues may assign priority or route to a shared node

On Cheyenne, you may assign your job a priority and charge factor which are inversely proportional. All such jobs give you exclusive use of a full node.



share queue for small jobs (<=18 tasks) on a shared node (cores and memory)

Casper jobs are submitted to a "routing" queue called **casper**.



Tailor your job by specifying custom resources

On Casper, all requested resources on nodes are exclusive to the job occupying them (using Linux control groups), except for gp100 GPUs

- If you need access to a GPU, you must specify an *ngpus* amount in your PBS select statement (and always provide a *gpu_type*).
- **Always** specify a per-chunk memory request for Casper jobs. If you exceed the requested amount, your program will use NVMe swap space and run *much slower*.
- If you do not specify *ompthreads*, the variable OMP_NUM_THREADS will be assigned to the *ncpus* amount.

Why aren't my jobs running?

- **Queue limits** - wallclock limits (e.g., 12-hr for Cheyenne jobs), GPU limits (32 V100s), core limits (18 CPUs on share queue)
- **Resource conflicts** - e.g., a job that requests gpu_type=gp100 and cpu_type=cascadelake; we have no nodes that satisfy both requirements
- **Large requests** - asking for a large amount of popular resources (e.g., 32 V100 GPUs) will result in difficult to place jobs
- **Heavy usage** - if the queue is busy, and you have submitted many jobs recently already, your relative priority will likely be low

Before you suspect a system issue, consider the conditions above and inspect the job using **qstat** for validity

Interacting with PBS as a JupyterHub user

JupyterHub provides you with a web-based compute environment for running Jupyter Notebooks and terminal sessions in NCAR systems.

Batch servers in JupyterHub spawn PBS jobs

JupyterHub jobs use core-hour resources; be mindful about stopping your servers when work is completed

- **Casper login sessions spawn a PBS job too; please limit yourself to a single Casper login server at any one time**

https://jupyterhub.hpc.ucar.edu/

A word about peer-submission and queue names...

If you submit a job to Casper from a Cheyenne login node (or vice versa), you must append the server name to the queue. Consider always appending the server name if you frequently use both systems.

On Cheyenne:		**On Casper:**
regular	→	regular**@chadmin1.ib0.cheyenne.ucar.edu**
casper**@casper-pbs**	←	casper

qinteractive *and* ***execcasper*** *will handle server specification for you!*

Specify dependencies between jobs (and across servers!)

Use job dependencies to run subsequent jobs based on exit status of original job:

`-W depend=<condition>:<jobid>`

- Jobs are **held** until the dependency is satisfied
- Jobs are then **pending**, but still may wait for resources in queue

after - all listed jobs begin execution

afterany - all listed jobs finish

afterok - all listed jobs succeed

afternotok - all listed jobs fail

```
# Example using Bash syntax
# Submit CFD jobs to Cheyenne and store job ids
cheyenne1$ J1=$(qsub -q regular run_ens1.pbs)
cheyenne1$ J2=$(qsub -q regular run_ens2.pbs)

# Submit data postprocessing job to Casper
# eligible to run if original two jobs succeed
cheyenne1$ qsub -q casper@casper-pbs -W
depend=afterok:$J1:$J2 run_proc.pbs
```

Querying active and recent jobs using peer-enabled qstat

qstat provides information on pending, running, held, and recently finished jobs. We cache output with a 10-second refresh rate to improve PBS performance.

qstat <jobid> - show single job

qstat <queue> - show jobs in queue

qstat -u <user> - show user's jobs

qstat -f <jobid> - show detailed info

qstat -x - include recent history

```
# Show my jobs in wide format
cheyenne1$ qstat -w -u $USER

# Show all known jobs on casper queue
cheyenne1$ qstat -x casper@casper-pbs

# qstat recognizes system names in addition to
# PBS server names (these three are equivalent)
cheyenne1$ qstat 12345
casper$ qstat 12345.chadmin1.ib0.cheyenne.ucar.edu
casper$ qstat 12345.cheyenne
```

Getting historical records for past PBS jobs

CISL provides **qhist** on Cheyenne and Casper to query past jobs:

```
qhist [-d DAYS] [-p START-END] [-u USER] [-j JOBID] …
```

- By default, **qhist** outputs all jobs from the current day, but has arguments to change time period and filter jobs by user, project, queue and more.
- **qhist** will only show you jobs from the native PBS server (e.g., Cheyenne jobs from Cheyenne nodes)

qhist allows you to quickly query CPU and memory usage of past jobs!

```
# Query my jobs from past week on Casper and find top 5 by memory use
casper$ qhist -u $USER -p 20210322-20210326 -s memory | head -n 6
Job ID    User       Queue    Nodes NCPUs NGPUs Finish     Mem(GB)    CPU(%)   Elap(h)
15259     vanderwb   htc          1     1     0 23-1942       10.0       2.0      0.08
15268     vanderwb   htc          1     1     0 23-1957        5.0       4.0      0.06
15337     vanderwb   htc          1     1     0 23-2043        5.0       3.0      0.08
15346     vanderwb   htc          1     1     0 23-2059        5.0       2.0      0.20
15057     vanderwb   htc          1     1     0 23-1523        1.0      12.0      0.20

# Get long-form output from the top job from above list
casper$ qhist -p 20210323 -j 15259 -l
15259.casper-pbs
   User            = vanderwb
   Queue           = htc
   Job Submit      = 2021-03-23T19:37:29
   ...
   Used Mem(GB)    = 10.0
   Avg CPU (%)     = 2.0
   Waittime (h)    = 0.00
   Walltime (h)    = 6.00
   Elapsed (h)     = 0.08
   Job Name        = STDIN
   Exit Status     = 0
   Account         = SCSG0001
   Resources       = 1:ncpus=1:mpiprocs=1
   Node List       = crhtc62
```

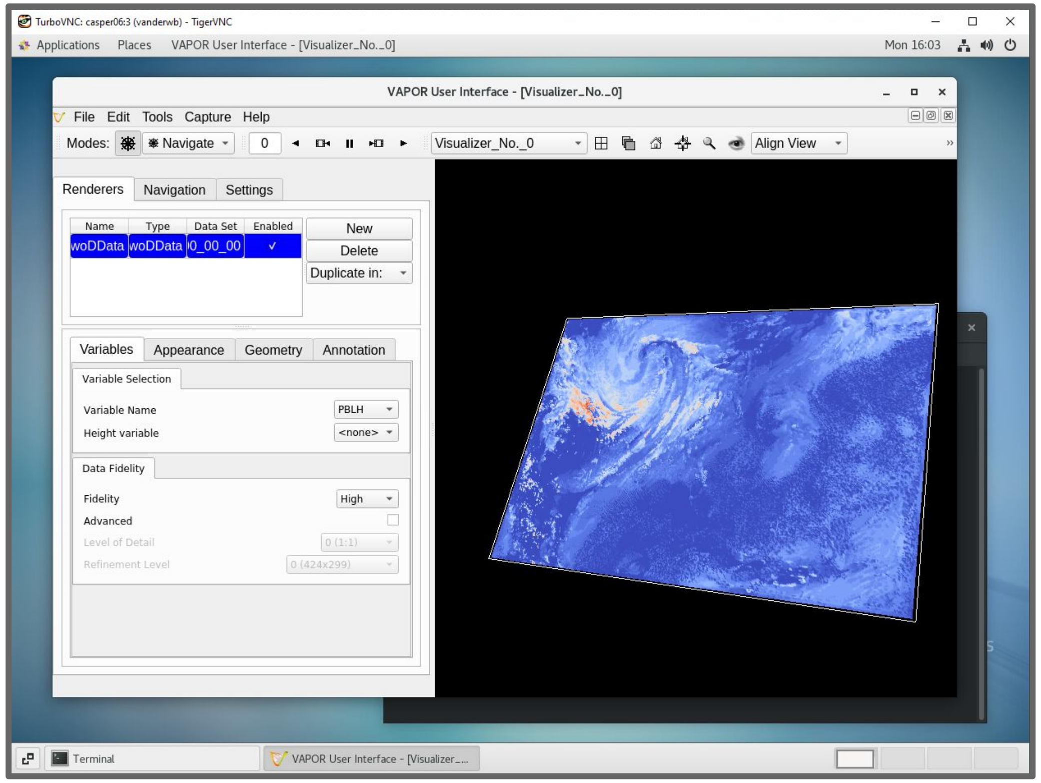
qcmd and vncmgr for specialized job submissions

CISL maintains two additional job submission scripts for special cases:

qcmd - run a non-interactive job that outputs directly to the terminal (e.g. a CESM build)

vncmgr - start a VNC remote desktop on a Casper gp100 node for graphically-intensive work

```
cheyenne$ qcmd -A <project> -- ./case.build
cheyenne$ vncmgr create -A <project> [SESSION]
```



Name	Type	Data Set	Enabled
woDData	woDData	0_00_00	✓

Variable Selection

Variable Name: PBLH

Height variable: <none>

Data Fidelity

Fidelity: High

Advanced

Level of Detail: 0 (1:1)

Refinement Level: 0 (424x299)

Some recommendations for user initialization files

Jobs will initialize a shell using ~/.profile (bash) or ~/.tcshrc (tcsh/csh)

- You can set default project codes to be used by:
  - **qinteractive** and **qcmd** **export PBS_ACCOUNT=<project>**
  - **execcasper** and **vncmgr** **export DAV_PROJECT=<project>**
- Don't include interactive commands in your init files as they can block batch job execution
- Init files are read by both Cheyenne and Casper jobs, so use if statements to limit execution of system-specific commands ($NCAR_HOST)
- In general, only put commands relevant to *all* anticipated workflows in your initialization files

Getting assistance from the CISL Help Desk

https://www2.cisl.ucar.edu/user-support/getting-help

- ~~Walk-in: ML 1B Suite 55~~
- Web: http://support.ucar.edu
- Phone: 303-497-2400

Specific questions from today and/or feedback:

- Email: vanderwb@ucar.edu