Starting Casper Jobs with PBS Pro

Brian Vanderwende
CISL Consulting Services

March 30, 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.

Casper resources have been significantly expanded

- 62 new “high-throughput computing” (HTC) and 2 new high-memory nodes

High-throughput computing

For data processing and analysis workflows that require less compute and/or more memory than typical Cheyenne jobs

- **13 nodes** with Intel Skylake CPUs
- **62 nodes** with Intel Cascade Lake
- 36 CPU cores per node
- 380 GB of memory per node
- 1.6-2 TB of NVMe local SSD

High memory

For CPU-only tasks that require exceptional amounts of shared (single-node) memory

- **2 nodes** with Intel Cascade Lake
- 36 CPU cores per node
- 1.5 TB of memory per node
- 1.6 TB of NVMe local SSD

Graphics and visualization

For 3D rendering (VAPOR, ParaView) and graphical interfaces via remote desktops

- **9 nodes** with Intel Skylake CPUs
- NVIDIA Quadro GP100 GPUs
- 36 CPU cores per node
- 380 GB of memory per node
- 2 TB of NVMe local SSD

General-purpose GPU

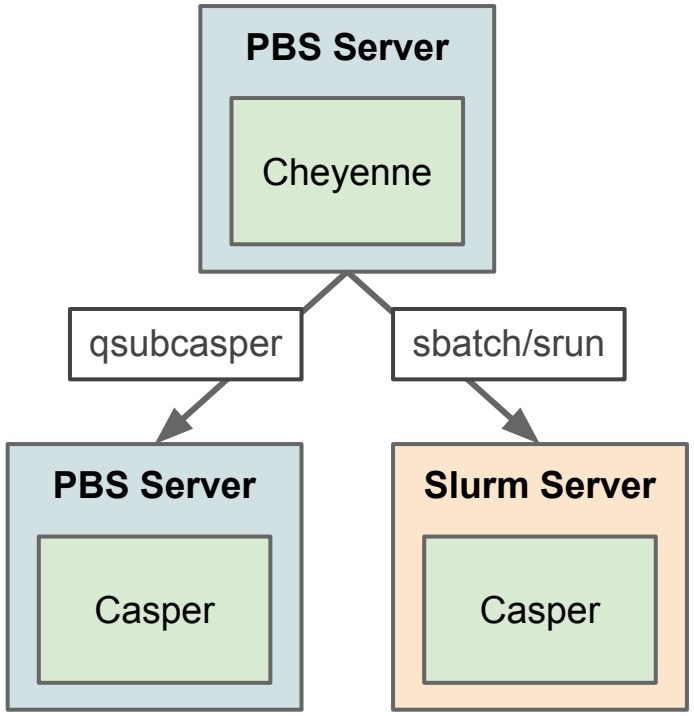
For development, testing, and running GPU-enabled models and also GPU-powered Machine Learning libraries

- **4 nodes** with 4x NVIDIA V100s
  - **2** with Intel Skylake CPUs
  - **2** with Intel Cascade Lake
  - 768 GB of node memory
- **6 nodes** with 8x NVIDIA V100s
  - All have Skylake CPUs
  - 1152 GB of node memory
- 2 TB of NVMe local SSD
- 32 GB of GPU RAM; NVLink

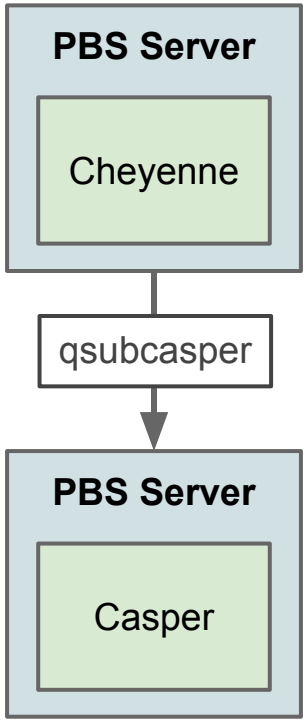
4 RDA nodes to support data processing for the public Research Data Archive

Working toward a “single-scheduler environment”

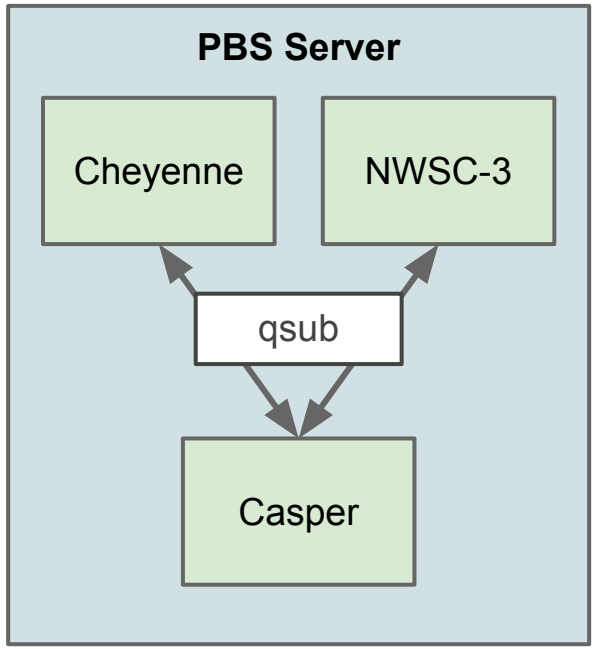
Now



April 7

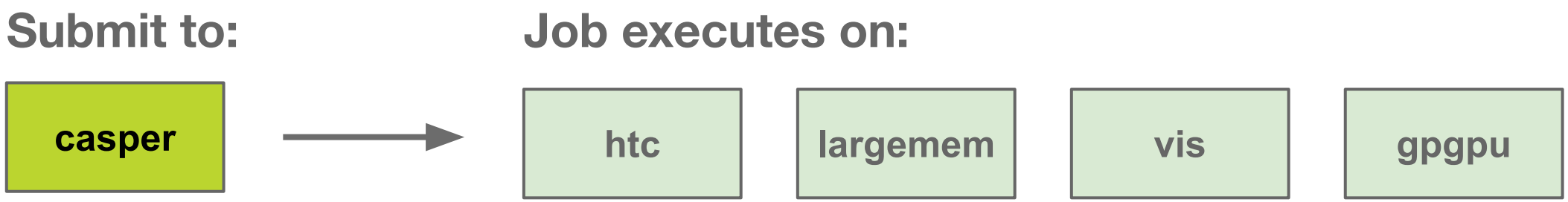


Summer 2021 & Beyond



Overview of how PBS Pro scheduling works on Casper

1. Submit batch or interactive job with **qsub, qsubcasper,** or **execcasper**
2. Jobs are initially submitted to the **casper** queue (a *routing queue* in PBS terminology), which is similar to the **dav** partition in Slurm
3. PBS then conditionally routes the job to an *execution queue* based on the specific resources requested
4. Your job will start with a default environment on Casper, so load modules and set environment variables at the start of your job



Submitting batch jobs with qsub and qsubcasper

```
#!/bin/bash
#PBS -A PROJ0001
#PBS -N ML_job
#PBS -j oe
#PBS -o mljob.log
#PBS -q casper
#PBS -l walltime=10:00:00
#PBS -l select=1:ncpus=8:mem=40GB:ngpus=1
#PBS -l gpu_type=v100

### Application temp data to scratch
export TMPDIR=/glade/scratch/$USER/temp
mkdir -p $TMPDIR

### Activate Python environment and run
module load python
ncar_pylib
python ml_driver.py

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

Submit from Cheyenne:

qsubcasper ml_script.pbs

Submit from Casper:

qsub ml_script.pbs

- Here, we request a 10-hour job with 8 CPU cores, 1 V100 GPU, and 40 GB of node memory
- This job is submitted to the **casper** queue and will execute on the **gpgpu** queue

Resource request fundamentals in PBS Pro

PBS has two types of resources - *job* resources and *chunk* resources. A job will consist of one or more chunks. Job resources will apply to all chunks.

Job-level:	walltime, gpu_type, cpu_type, place
Chunk-level:	ncpus, mpitasks, ompthreads, ngpus, mem

Each *job* resource is specified in its own directive, while *chunk* resources are collectively specified in a *select* statement.

Job-level:	`#PBS -l cpu_type=skylake`
Chunk-level:	`#PBS -l select=2:ncpus=8:mem=80GB`

Submitting interactive jobs with execcasper

execcasper provides a simple command for starting an interactive session

- Default resources: 1 core on 1 HTC node, 10 GB of node memory, and a six-hour walltime
- Unlike Slurm, all resources on the primary node are always assigned to the shell, and thus are available to any programs you run
- Specify a project using **-A** flag or by setting **DAV_PROJECT** env variable
- All qsub flags are supported by **execcasper**

```
# Set project in shell (tcsh here) and start 2-hour HTC session
cheyenne1$ setenv DAV_PROJECT PROJ0001
cheyenne1$ execcasper -l walltime=02:00:00

# Request 18 MPI processes and 4 V100 GPUs and 100GB of memory
casper-login2$ execcasper -l select=1:ncpus=18:mpitasks=18:ngpus=4:mem=100GB -l gpu_type=v100
```

Quick specification flags to easily customize resources

execcasper also provides custom flags to quickly modify a single resource without specifying entire select statement

--nchunks=N
--ntasks=N
--nthreads=N
--ngpus=1-8
--mem=NGB

```
# 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
```

Job dependencies in PBS Pro

- Dependencies are similar to Slurm dependencies:
  - **after** = all jobs in list have started
  - **afterok, afternotok** = all jobs in list have succeeded/failed
  - **afterany** = all jobs in list have exited with any status
- Not yet supported between Cheyenne and Casper jobs

```
#!/bin/bash
#PBS -N GPU_model
#PBS -A PROJ0001
#PBS -l walltime=10:00:00
#PBS -q casper
#PBS -l select=1:ncpus=8:mem=100GB:ngpus=4
#PBS -l gpu_type=v100

# Run model last in script to use correct exit code
mpirun ./model.exe
```

```
# Example using Bash syntax
# Submit initial jobs to PBS and capture job ids
casper$ J1=$(qsub run_ens1.pbs)
casper$ J2=$(qsub run_ens2.pbs)

# Submit secondary job with success conditions
casper$ qsub -W depend=afterok:$J1:$J2 run_proc.pbs
```

Unlike shared jobs on Cheyenne, resources are exclusive

- Any CPUs and node memory that you request are reserved for exclusive use by your job; no other jobs can access those resources
- The V100 GPUs are also scheduled for exclusive use
  - NVIDIA's multi-instance GPU is not supported by PBS at this time
- The GP100 GPUs are shared among all jobs on a visualization node
- Exclusive use means that your job is restricted to the resources you request
  - Caveat: NVMe swap space allows your job to proceed even if you run out of RAM at the cost of reduced performance

Summary of per-user resource limits for each job class

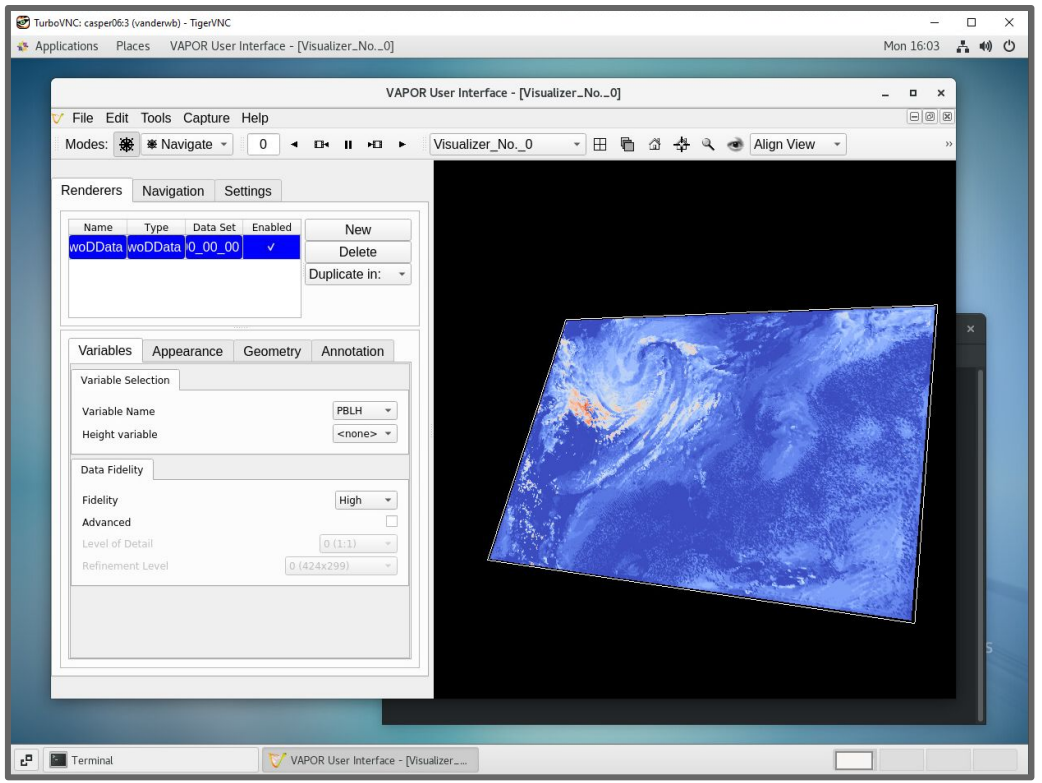
Job limits are intended to ensure short dispatch times and a fair distribution of Casper's resources.

Job Category	Job Characteristics	Concurrent Use Limits
htc	mem <= 361 GB; ncpus <= 36 ngpus = 0	<= 468 CPUs <= 4680 GB memory
largemem	mem > 361 GB; ncpus <= 36 ngpus = 0	Up to 5 jobs eligible (more can be queued)
vis	gpu_type = gp100	1-2 GPUs in use by running jobs
gpgpu	gpu_type = v100; ngpus > 1	1-16 GPUs in use by running jobs

Virtual desktop options: FastX and vncmgr

Graphically intensive applications are best run in a virtual desktop using either **FastX** or **TurboVNC** via **vncmgr**:

- Login-type KDE desktop session with low resource requirements -> use **FastX**
  - Can submit PBS Pro jobs from desktop session to access more resources
- Rendering or other resource intensive task in remote desktop -> use **vncmgr**



JupyterHub will use PBS for all batch sessions by April 7

Querying and deleting active PBS jobs on Casper

- Delete pending or running jobs using **qdel/qdelcasper <jobid>**
- Show active jobs using the **qstat** command (cached every 10 seconds)
  - Can show jobs running on opposite *server* using *@server* notation
    - Only supported for certain options (-u,-w,-s,-n,-x)

```
# Show current Casper jobs with nodes assigned to running jobs (-n option)
casper-login1$ qstat -n

# Show my (-u) active and recently completed (-x) jobs on Casper from Cheyenne login node
cheyenne1$ qstat -u $USER -x @casper
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
24248.casper-p* vanderwb htc      STDIN       74786   2   4    20gb 06:00 F 00:02
24293.casper-p* vanderwb htc      STDIN       80888   1   1    10gb 06:00 F 00:00
24295.casper-p* vanderwb vis      vncs-defa* 189039   1   1    10gb 04:00 F 00:05
```

Getting historical records for past PBS jobs

PBS Pro does not provide an equivalent to Slurm's **sacct** command, so CISL maintains the **qhist** command 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 allows you to quickly query CPU and memory usage of past jobs!

qhist will show records from the current server

```
# Query my jobs from past week on Casper and find top 5 by memory use
casper-login1$ 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-login1$ qhist -p 20210323 -j 15259 -l
15259.casper-pbs
   User           = vanderwb
   ...
   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
```

Command and interface migration from Slurm to PBS

Slurm commands

sbatch

salloc/srun

squeue

scancel

sacct

Slurm MPI support

Open MPI <= 4.0.5

PBS Pro commands

qsub

qsub -I

qstat

qdel

qhist

PBS MPI support

Open MPI >= 4.1.0

MVAPICH2

Migrated support

JupyterHub (April 7)

FastX

vncmgr (TurboVNC)

Accounts

Changes mostly invisible to end users!

Common mistakes when submitting to Casper

- **Using “qsub” on Cheyenne when attempting to submit to Casper**
  - Will get an “Unknown queue” message at submission time
- **Requesting an invalid resource amount or combination (e.g., ngpus=2 and gpu_type=gp100)**
  - Depending on specific request, may be rejected at submission time or job may end up in hold state (*verify job is eligible after new submission*)
- **Requesting less node memory than application requires**
  - Job is unlikely to fail because of NVMe “swap space”, but performance will likely decrease significantly when RAM is exhausted
- **Loading Cheyenne modules (e.g., mpt) in Casper script**
  - The job will fail at runtime with an Lmod error

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