Introduction to Cheyenne for New Users

14 November 2019
Consulting Services Group
Sidd Ghosh
Topics to cover

- Overview of our computing environment
- Signing into the systems and managing data
- Accessing and building software
- Submitting jobs using the PBS and Slurm schedulers
- Customizing your user environment

This is only an introduction; for more usage details, see: https://www2.cisl.ucar.edu/resources/
Overview of NCAR systems

HPC - simulation

**Cheyenne** - 4032 nodes
- 2-socket 18-core Intel Broadwell Xeon CPUs
- 3164 nodes with 64 GB mem
- 864 nodes with 128 GB mem
- SUSE Enterprise Linux 12
- PBS job scheduler

Data Analysis, Visualization (DAV) Machine Learning/Deep Learning

**Casper** - 26 nodes featuring:
- 2-socket 18-core Intel Skylake
- 2 TB local NVMe SSD storage
- CentOS 7 Linux
- Slurm job scheduler

GPUs available:
- 8 nodes - 1 NVIDIA GP100
- 2 nodes - 4 NVIDIA Tesla V100s
- 4 nodes - 8 NVIDIA Tesla V100s
First steps in our HPC environment

Systems Accounting Manager

- [https://sam.ucar.edu/](https://sam.ucar.edu/)
- Contains information about your allocation(s) and user account
- Can change default project, shell, and more

Logging in to the system

- Use your authentication method (YubiKey or Duo YubiKey is being phased out) along with your username to login:
  
  ```bash
  ssh -X -l username cheyenne.ucar.edu
  ```

- You will be placed on one of six login nodes
Be mindful of your usage on shared resources like the login nodes

• Your programs coexist with those of 10-100s of other users for processing and memory
• Therefore, limit your usage to:
  • Reading and writing text/code
  • Compiling smaller programs
  • Performing (small) data transfers
  • Interacting with the job schedulers
• Programs that use excessive resources on the login nodes will be terminated
**Personal data storage at NCAR**

**GLADE parallel hard-disk storage**
- Optimized for parallel input/output
- Accessible from all HPC systems

<table>
<thead>
<tr>
<th>File space</th>
<th>Quota</th>
<th>Backup</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>25 GB</td>
<td>Yes</td>
<td>Settings, code, scripts</td>
</tr>
<tr>
<td>/glade/u/home/$USER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work</td>
<td>1 TB</td>
<td>No</td>
<td>Compiled codes, models</td>
</tr>
<tr>
<td>/glade/work/$USER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash</td>
<td>N/A</td>
<td>No</td>
<td>Fast temp space</td>
</tr>
<tr>
<td>/glade/flash/$USER</td>
<td></td>
<td></td>
<td>By request</td>
</tr>
<tr>
<td>Scratch</td>
<td>10 TB</td>
<td>Purged!</td>
<td>Run directories, temp output</td>
</tr>
<tr>
<td>/glade/scratch/$USER</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Keep track of usage with “gladequota”*
Collaborative and long-term storage

- Dedicated GLADE project spaces
- Campaign Storage for publication-scale storage lifespans (5-year purge)
- HPSS tape archive is being phased out

Access to these spaces is contingent on your project/lab status. See our web documentation for more details.
Moving data to and from GLADE

- For short transfers, you can use `scp/sftp` to transfer files
- Large or lengthy transfers will benefit from **Globus**
  - To use Globus, create a Globus ID if you need an account, and search for **NCAR GLADE** or **NCAR Campaign Storage** endpoints
  - CISL endpoints currently can be activated for up to 30-days
  - Globus has a web interface and a command-line interface
  - **Globus Connect Personal** can manage transfers from your local workstation as well
- Transfers to and from the HPSS tape archive are made using the HSI interface and HTAR utility
CISL builds software for users that is accessible via environment modules

- Modules provide access to program binaries (e.g., ncl, Python, ifort)
- Many modules will help you compile and link to common libraries (e.g., netCDF, MPI)
- Modules also prevent you from loading conflicting software into your environment
- Note that Cheyenne and Casper each have independent collections of modules!
Using environment modules

- **module load/unload <software>**
- **module avail** - show all currently-loadable modules
- **module list** - show loaded modules
- **module purge** - remove all loaded modules
- **module save/restore <name>** - create/load a saved set of software
- **module spider <software>** - search for a particular module
Considerations when compiling software

- Use `ncarcompilers` module along with library modules (e.g., netcdf) to simplify compiling and linking (*it adds include and link flags for you*)
- When using MPI, make sure you run with the same library with which you compiled your parallel code
- **We strongly recommend you build code for the machine on which you will run**
  - Cheyenne and Casper have different CPUs and operating systems
Run large tasks on compute nodes using batch jobs

- Many tasks require too many resources to run on a login node
- Schedule these tasks to run on the Cheyenne compute nodes using PBS or on Casper nodes using Slurm

```
ssh cheyenne.ucar.edu
ssh casper.ucar.edu
```

Cheyenne and Casper use separate allocations!
Run large tasks on compute nodes using batch jobs

- Many tasks require too many resources to run on a login node
- Schedule these tasks to run on the Cheyenne compute nodes using **PBS** or on Casper nodes using **Slurm**
- Jobs request a given number of compute tasks for an estimated wall-time on specified hardware
- Jobs use core-hours, which are charged against your selected project/account
  - Remaining resources are viewable in SAM
- Temporary files are often written by programs - set TMPDIR variable to scratch space to avoid job failures
Example PBS and Slurm batch job scripts

```bash
$ cat basic_mpi.pbs
#!/bin/tcsh
#PBS -N hello_pbs
#PBS -A <project_code>
#PBS -j oe
#PBS -o pbsjob.log
#PBS -q regular
#PBS -l walltime=00:05:00
#PBS -l select=2:ncpus=8:mpiprocs=8
### Set temp to scratch
setenv TMPDIR /glade/scratch/${USER}/temp
mkdir -p $TMPDIR
module load mpt/2.19
### Run MPT MPI Program
mpiexec_mpt ./hello_world
```

```bash
$ cat basic_mpi.slurm
#!/bin/tcsh
#SBATCH -J hello_slurm
#SBATCH -A <project_code>
#SBATCH -o slurmjob.log
#SBATCH -p dav
#SBATCH -t 00:05:00
#SBATCH -N 2
#SBATCH --ntasks-per-node=8
### Set temp to scratch
setenv TMPDIR /glade/scratch/${USER}/temp
mkdir -p $TMPDIR
module load openmpi/3.1.2
### Run Open MPI Program
srun ./hello_world
```

```bash
$ cat basic_mpi.slurm
#!/bin/bash -l
#SBATCH -J hello_slurm
...
Interacting with the job schedulers

**PBS on Cheyenne**
- `qsub <script>` - submit batch job
- `qstat <jobid>` - query job status
- `qdel <jobid>` - delete/kill a job
- `qinteractive -A <project>`
  Run an interactive job
- `qcmd -A <project> -- cmd.exe`
  Run a command on a single compute node

**Slurm on DAV**
- `sbatch <script>` - submit batch job
- `squeue -j <jobid>` - query job status
- `scancel <jobid>` - delete/kill a job
- `execdav -A <project>`
  Run interactive job on DAV

See our Casper tutorial and documentation for more details on requesting memory/GPUs with the execdav utility.
Using threads/OpenMP parallelism on Cheyenne with MPT

**Only OpenMP**

```bash
#!/bin/tcsh
#PBS -l select=1:ncpus=10:ompthreads=10

# Run program with 10 threads
./executable_name
```

**Hybrid MPI/OpenMP**

```bash
#!/bin/tcsh
#PBS -l select=2:ncpus=36:mpiprocs=1:ompthreads=36

module load mpt/2.19

# Run program with one MPI task and 36 OpenMP threads per node (two nodes)
mpiexec_mpt omplace ./executable_name
```
Running serial code on multiple data files using command file jobs

Command file contents

```bash
./cmd1.exe < input1 > output1
./cmd2.exe < input2 > output2
./cmd3.exe < input3 > output3
./cmd4.exe < input4 > output4
```

PBS Job script

```
#!/bin/tcsh
#PBS -l select=1:ncpus=4:mpiprocs=4
module load mpt/2.19
# This setting is required to use command files
setenv MPI_SHEPHERD true
mpiexec_mpt launch_cf.sh cmdfile
```

Optimal if commands have similar runtimes
Placing Casper jobs on specific resources

```bash
$ cat gpu_job.slurm
#!/bin/tcsh
#SBATCH -J sample_gpu
#SBATCH -n 6
#SBATCH --ntasks-per-node=3
#SBATCH -t 05:00
#SBATCH -A <project_code>
#SBATCH -p dav
#SBATCH --C v100
#SBATCH --mem=100G
#SBATCH --gres=gpu:v100:2
...
```

- This job can only run on a node with 100 GB of free memory and 2 V100 GPUs
- If multiple resources are specified, they must be compatible
  - Otherwise, the job will be stuck in a pending state
## PBS job submission queues

<table>
<thead>
<tr>
<th>PBS Queue</th>
<th>Priority</th>
<th>Wall clock</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>premium</td>
<td>1</td>
<td>12 h</td>
<td>Jobs are charged at 150% of regular rate</td>
</tr>
<tr>
<td>regular</td>
<td>2</td>
<td>12 h</td>
<td>Most production compute jobs go here</td>
</tr>
<tr>
<td>economy</td>
<td>3</td>
<td>12 h</td>
<td>Jobs are charged at 70% of regular rate</td>
</tr>
<tr>
<td>share</td>
<td>N/A</td>
<td>6 h</td>
<td>Memory is shared among all users on a node Jobs are limited to 18 cores or less</td>
</tr>
</tbody>
</table>

**Job charges depend on the queue:**

**Exclusive:** wall-clock hours ✗ nodes ✗ 36 cores/node ✗ queue factor

**Shared:** core-seconds / 3600 (DAV jobs are shared as well)
When running programs with GUI (e.g., VAPOR), use a TurboVNC session

VNC can be used to run a remote GNOME/KDE desktop

Usage:

```
vncserver_submit -a <project>
```
(or set DAV_PROJECT environment variable)
Shell startup files - customizing your default environment

**tcsh/csh**

```
$ cat ~/.tcshrc
alias rm "rm -i"

# Add programs built for each cluster
if ( $HOSTNAME =~ cheyenne* ) then
    setenv PATH ~/local/ch/bin:$PATH
else
    setenv PATH ~/local/dav/bin:$PATH
endif

# Settings for interactive shells
if ( $?prompt ) then
    set prompt = "%n@%m:%~"
endif
```

**bash**

```
$ cat ~/.profile
alias rm="rm -i"

# Add programs built for each cluster
if [[ $HOSTNAME == cheyenne* ]]; then
    export PATH=~/local/ch/bin:$PATH
else
    export PATH=~/local/dav/bin:$PATH
fi

# Settings for interactive shells
if [[ $- == *i* ]]; then
    PS1="\u@\h:\w> "
fi
```
Changing your default modules

- If you commonly load certain modules, you may wish to have them load automatically when logging onto a cluster
- The right way to do so is with saved module sets:

  module load ncl python nco mkl
  module save save default

- Make multiple sets and load them using module restore <set>
- Don’t put module load commands in your shell startup files!
CISL User Services

• Service Desk / Knowledge Base: support.ucar.edu
• Walk-in: ML 1B Suite 55
• Phone: 303-497-2400