Getting Started With CCV

Mark Howison

User Services & Support
This talk...

- Assumes you have some familiarity with a Unix shell
- Provides examples and best practices for typical usage of CCV systems
- Is a condensed form of the documentation available at:

  http://brown.edu/ccv/doc
Overview

- Connecting to CCV
- Transferring files
- Available software
- Running and monitoring jobs
- Compiling and linking your own code
Logging in

- CCV uses the Secure Shell (SSH) protocol
- You will need an SSH client
  - Linux / OS X comes with a command-line client
  - Windows: try PuTTY or Cygwin
- Connect to the “ssh” server
  - Linux / OS X / Cygwin:
    `ssh username@ssh.ccv.brown.edu`
  - PuTTY: enter `ssh.ccv.brown.edu` in “Host Name”
Login nodes

- When you ssh to Oscar, you are placed on either login001 or login002
  - The login nodes are intended for tasks like:
    - writing, compiling, and debugging code
    - transferring and managing files
    - submitting and managing jobs
  - They are shared by all users logged into CCV
  - Please don't run your jobs directly on the login nodes!
    - It disrupts other users
    - Start an interactive or batch job instead
Virtual Network Computing

- OR connect through CCV’s Virtual Network Computing client, available here:
  - http://brown.edu/ccv/vnc
# File systems

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Your <strong>home</strong> directory:</td>
</tr>
<tr>
<td>→ /home/&lt;user&gt;</td>
<td>• optimized for many small files (&lt;1MB)</td>
</tr>
<tr>
<td></td>
<td>• nightly backups</td>
</tr>
<tr>
<td></td>
<td>• 10GB quota</td>
</tr>
<tr>
<td>~/data</td>
<td>Your <strong>data</strong> directory:</td>
</tr>
<tr>
<td>→ /gpfs/data/&lt;group&gt;</td>
<td>• optimized for reading large files (&gt;1MB)</td>
</tr>
<tr>
<td></td>
<td>• nightly backups</td>
</tr>
<tr>
<td></td>
<td>• quota is by group (usually &gt;=256GB)</td>
</tr>
<tr>
<td>~/scratch</td>
<td>Your <strong>scratch</strong> directory:</td>
</tr>
<tr>
<td>→ /gpfs/scratch/&lt;user&gt;</td>
<td>• optimized for reading/writing large files (&gt;1MB)</td>
</tr>
<tr>
<td></td>
<td>• NO BACKUPS</td>
</tr>
<tr>
<td></td>
<td>• purging: files older than 30 days may be deleted</td>
</tr>
<tr>
<td></td>
<td>• 512GB quota</td>
</tr>
</tbody>
</table>

Note: class or temporary accounts may not have a ~/data directory!
Transferring files

- **RData**: mount your **home**, **data**, and **scratch** directories on your local system over CIFS (Brown network only)
  - [http://brown.edu/ccv/doc/cifs](http://brown.edu/ccv/doc/cifs)

- **scp**: a command used from the terminal
  - Copy from Oscar:
    ```
    scp username@ssh.ccv.brown.edu:/remote/path /local/path
    ```
  - Copy to Oscar:
    ```
    scp /local/path username@ssh.ccv.brown.edu:/remote/path
    ```

- **rsync**: a command like **scp** but with more features

- **GUI “Secure Copy” programs**
  - e.g. WinSCP, Fugu (Mac), CyberDuck (Mac)
Available software

- Software is organized using Environment Modules
- Load a software module with `module load <name>`
- Loading a module alters your environment, paths, etc:

```
[user@login001 ~]$ module display intel
-- -----------------------------------------------
/gpfs/runtime/modulefiles/intel/12.0.4:
setenv MKL -L/gpfs/runtime/opt/intel/12.0.4/mkl/lib/intel64 -lmkl_rt -liomp5 -lpthread
prepend-path PATH /gpfs/runtime/opt/intel/12.0.4/bin
prepend-path MANPATH /gpfs/runtime/opt/intel/12.0.4/man/en_US
prepend-path LD_LIBRARY_PATH /gpfs/runtime/opt/intel/12.0.4/lib/intel64:/gpfs/runtime/opt/intel/12.0.4/mkl/lib/intel64
...

[user@login001 ~]$ module load intel

[user@login001 ~]$ which icc
/gpfs/runtime/opt/intel/12.0.4/bin/icc
```
Available software (Cont'd)

- View available modules with `module avail`
  - Or search for a specific package with `module avail <package>`

- View your loaded modules with `module list`

- Module commands in your `~/.modules` file will automatically execute when you login
  - Add a module to your default list with `echo "module load <name>" >>~/./modules`

- If you need software that is not installed, submit a request at [http://brown.edu/ccv/protected/software](http://brown.edu/ccv/protected/software)
Running jobs

- An **interactive** job allows you to:
  - Interact with a program by typing input, using a GUI, etc.
    - But if your connection is interrupted, the job will abort
  - Quickly stop and restart a program, e.g. to test out different parameters or for debugging
  - Run on resources shared with other users

- A **batch** job allows you to:
  - Submit a script that will run without any intervention
  - Access dedicated resources for your job
  - Run for long periods of time without worrying about other users interfering with your job, or your connection dying
Queues on Oscar

- Oscar's compute nodes are organized into queues
- Choose the queue that best matches the needs of your job
- Nodes are packed by user: if you have multiple jobs that request less than a full node, the jobs can share a node
  - But other users' jobs will not run on that same node as yours

<table>
<thead>
<tr>
<th>Queue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dq</td>
<td>Default queue with most of the compute nodes: 8-, 12-, or 16-core; 24GB to 48GB of memory; all Intel based</td>
</tr>
<tr>
<td>gpu</td>
<td>Specialized compute nodes (8-core, 24GB, Intel) each with 2 NVIDIA GPU accelerators</td>
</tr>
<tr>
<td>timeshare</td>
<td>Large memory nodes that are shared by multiple users: 12-, 16-, or 32-core; 64GB to 128GB of memory; both Intel and AMD</td>
</tr>
<tr>
<td>debug</td>
<td>Dedicated nodes for fast turn-around, but with a short time limit of 40 node-minutes</td>
</tr>
</tbody>
</table>
Interactive jobs

- You can start an interactive job from the login node with:
  ```bash
  interact [-n cores] [-t walltime]
            [-m memory] [-q queue] [-X]
  ```

- Defaults are 1 core, 30:00, 4GB, timeshare queue:
  ```bash
  [user@login001 ~]$ interact
  ```

- To request more cores/time/memory:
  ```bash
  [user@login001 ~]$ interact -n 8 -m 64g -t 4:00:00
  ```

- To enable X forwarding (e.g. to run a GUI program):
  ```bash
  [user@login001 ~]$ interact -X
  ```

- To request a different queue:
  ```bash
  [user@login001 ~]$ interact -q debug
  ```
Batch jobs

- Specify resources and the commands to run in a file called a **batch script**

```bash
#!/bin/bash
#PBS -N MyJobName
#PBS -l walltime=2:00:00

# execute your commands:
my_program <args>
```

- Submit the script to a **queue** with:
  ```bash
  qsub myjob.sh
  qsub -q timeshare myjob.sh
  ```

- Default resources are:
  - 1 node, 1 core, 1 minute (e.g. always specify a walltime!)
# Monitoring jobs

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allq</td>
<td>List all jobs in all queues. Hint: page through the list with `allq</td>
</tr>
<tr>
<td>allq &lt;queue&gt;</td>
<td>List all jobs in the specified queue</td>
</tr>
<tr>
<td>myq</td>
<td>List only your own jobs in all queues</td>
</tr>
<tr>
<td>myq &lt;user&gt;</td>
<td>List another user's jobs</td>
</tr>
<tr>
<td>qdel &lt;jobid&gt;</td>
<td>Delete a job (the batch script is terminated)</td>
</tr>
<tr>
<td>qstat -f &lt;jobid&gt;</td>
<td>See full details for a job</td>
</tr>
<tr>
<td>checkjob &lt;jobid&gt;</td>
<td>Diagnose a blocked job</td>
</tr>
</tbody>
</table>

The output of a running job is **spooled** to a file that you can view with:

```
tail <jobid>.mgt.OU
```

To receive an email when your job begins, ends, or aborts add the PBS options:

```
#PBS -m abe
#PBS -M oscar_user@brown.edu
```
Batch jobs for threaded programs

- A **threaded** program can use additional cores on a node
- The **ppn** property controls the *processors (cores) per node*
- The **$PBS_NP** shell variable is a shortcut for the total *number of processors (cores) = nodes x ppn*

```bash
#!/bin/bash
#PBS -N MyJobName
#PBS -l walltime=2:00:00
#PBS -l nodes=1:ppn=8

# execute your (threaded) commands:
my_program -t $PBS_NP <args>
```
Batch jobs for MPI programs

- An MPI program can communicate among multiple nodes
- Use the `nodes` property and the `$PBS_NP` shell variable

```bash
#!/bin/bash
#PBS -N MyJobName
#PBS -l walltime=2:00:00
#PBS -l nodes=4:ppn=8

# execute your MPI command on 32 cores
# across 4 nodes:
mpirun -n $PBS_NP my_mpi_program <args>
```
Job arrays

- Job arrays are a special feature for running parameter sweeps
- Add `#PBS -t <range>`
  - Where range is of the form 1-N or 0,2,4-8 etc.
  - A job with the name `<jobname>[i]` is created for each value in the range
  - Each job is given a different `$PBS_ARRAYID` value from the range
Job arrays

- This job will use 32 cores to sweep over 32 parameters
  - The parameters are located in 32 different files named: `input_file_1`, `input_file_2`, etc.
  - In the queue, you will see 32 individual jobs named: `MySweep[1]`, `MySweep[2]`, etc.

```bash
#!/bin/bash
#PBS -N MySweep
#PBS -l walltime=1:00:00
#PBS -t 1-32
#PBS -l nodes=1:ppn=1

echo "Starting sweep $PBS_ARRAYID on $HOSTNAME:$PBS_VNODENUM"
my_program "input_file_$PBS_ARRAYID"
```
Compiling

- GNU compiler suite is the default on Oscar
- Compile a single source file with:
  
  gcc  -g  -O2  -o  myprogram  myprogram.c
  g++  -g  -O2  -o  myprogram  myprogram.cpp
  gfortran  -g  -O2  -o  myprogram  myprogram.f90
- -g flag = generate debugging symbols
- -O2 flag = use a higher level of optimization
Intel and PGI Compilers

- The Intel compiler suite is available with
  module load intel
  icc
  icpc
  ifort

- The Portland Group compilers are available with
  module load pgi
  pgcc
  pgCC
  pgf77
  pgf90
Many modules include environment variable shortcuts that contain linking directions, e.g.

**FFTω**

```bash
gcc -o fftw-app fftw-app.c $FFTω
$FFTω = -I/gpfs/runtime/opt/fftw/3.2.2/include -L/gpfs/runtime/opt/fftw/3.2.2/lib -lfftw3
```

**GotoBLAS**

```bash
gcc -o blas-app blas-app.c $GOTO
$GOTO = -I/gpfs/runtime/opt/gotoblas2/1.13/include -L/gpfs/runtime/opt/gotoblas2/1.13/lib -lgoto2 -lpthread -lgfortran
```
Contact Info

For help or support:
- (401) 863-7557
- support@ccv.brown.edu
- Hours:
  8:30 – 5:00 (Academic Year)
  8:00 – 4:00 (Summer)

Keeping in touch:
- System updates from Twitter: ccv_brown
- User mailing list: users@ccv.brown.edu
- Website: http://brown.edu/ccv