Introduction to High Performance Computing

UIC HPC WORKSHOP

UIC EXTREME COMPUTING

UIC HPC Professional Staff

Ramon Castillo

HPC Systems Administrator, ACCC

Greg Cross

HPC Technical Lead, ACCC

Himanshu Sharma

HPC Project Manager, RRC

Training Agenda

- Access
- Basic Linux Commands
- Storage on the Cluster
- Basic Script Creation
- Introduction to Job Submission
- Job Monitoring
- MPI Program Sample
- Additional Resources
- Questions & Answers
- Resources

Access

How to Access the Cluster

- Access requires a **UIC NetID** for login. Your password is your **ACCC Common Password**.
- Using Secure Shell (**SSH**) on Mac, Linux, and Unix:
  - Command requires terminal application window
  - **`ssh` *`netid`*`@login-1.extreme.uic.edu`**
- Using SSH on Windows
  - Requires third party application, such as **PuTTY**
  - Download PuTTY for free from http://www.putty.org/
    - From the PuTTY application select **SSH** radio button under **Session**
    - Enter **`login-1.extreme.uic.edu`** in **Host Name** field

UIC EXTREME COMPUTING

Access: File Transfers

- File transfers can be done via Secure Copy (**SCP**), which uses the same credentials as SSH.
- Using SCP on Mac, Linux, and Unix:
  - **`scp -r source_path dest_path`**
  - Use **`-r`** when recursively copying contents of a directory
  - The remote path (either source or destination) is formatted as ***`netid`*`@login-1.extreme.uic.edu:/path`**
- Using SCP on Windows:
  - PuTTY web site provides an analogous **PSCP** application which uses similar syntax to SCP.
  - **`pscp.exe`** is CLI-based; run within **`cmd.exe`** utility.

cribioinfo@uic.edu

Access: Useful SSH Flags and X11

How to Access the Cluster

- To allow an **X11 GUI application** from within SSH:
  - **`ssh -Y netid@login-1.extreme.uic.edu`**
- The X11 server is already available on Linux and Unix workstations. **XQuartz** is freely available for the Mac; **Xming** is freely available for Windows.

- Using SSH in **verbose mode** (helpful when debugging):
  - **`ssh -v netid@login-1.extreme.uic.edu`**

Basic Linux Commands: File System Navigation

- **`cd`**
  - The **c**hange **d**irectory command is used to navigate the file system:
    - **`cd *path*`**
- **`ls`**
  - The **l**i**s**t directory command is used to gain information on files and subdirectories (folders) in a directory. A good example with useful flags:
    - **`ls -lah`**
    - **`l`** = long (detailed), **`a`** = all (including hidden dot files), **`h`** = human-readable file sizes
- **`pwd`**
  - The **p**rint **w**orking **d**irectory command is used to determine the directory in which you are currently located with respect to the file system hierarchy. This command is useful for determining where you are if you get lost navigating the file system.
- **`mkdir`**
  - By default, this command will **m**a**k**e a new sub**dir**ectory (folder) relative to the directory in which you are currently located. You may however provide an absolute path to where you wish a subdirectory to be made.
  - **`mkdir test1`** `# creates test1 relative to current directory`
  - **`mkdir /export/home/*netid*/test2`** `# creates absolute path`

cribioinfo@uic.edu

Basic Linux Commands: File I/O and Editing

- **`cat`**
  - outputs (and con**cat**enates) the contents of one or more files to standard output (by default the screen)
  - **`cat`** ***`filename`*** *`filename …`*
- **`grep`**
  - Searches input **g**lobally for a **r**egular **e**xpression and **p**rints matching patterns (**regular expressions** are special character strings using a versatile search syntax)
  - **`grep`** ***`regex filename`***
- **`vi`** and **`nano`**
  - text-based file editors
  - **`vi filename`** —or— **`nano filename`**
  - **nano** is more intuitive; **vi** is more powerful

cribioinfo@uic.edu

Basic Linux Commands: File Permissions

- **chmod** ***mode filename*** *filename …*
  - **ch**anges **mod**es of **r**ead, **w**rite, and e**x**ecute file permissions, as well as other persistent setuid and ownership (“sticky bit”).
  - Permissions are ordered by **u**ser, **g**roup, & **o**ther user access.
  - Current ownership can be determined using the long directory listing (**`ls -l`**); e.g., `-rwxr-x---` indicates the user can read, write, and execute, the group can read and execute, and world has no access (the leading **`-`** means no persistence flags are set).
  - Mode parameters can be relatively changed ***m±n*** (where ***m*** is one or more of **u**, **g**, or **o**ther; **+** to add or **-** to remove; and ***n*** is one or more of **r**, **w**, or **x**),; e.g., **`chmod ug+x`** ***filename*** gives the user and group execute permission.
  - Mode parameters can be absolutely changed using octal-encoded values; e.g., **`chmod 0750`** ***filename*** specifies `-rwxr-x---`
  - N.b.: directories must have the execute bit set to be accessible.

cribioinfo@uic.edu

Basic Linux Commands: File Ownership

- **`chown` *`owner filename`* *`filename …`***
  - Changes the user (UIC NetID) owning the specified file(s) and/or directory(ies); wildcards such as **`*`** can be used (with caution).
  - ***`owner`*** can be the user only or user**`:`**group (colon-separated).
- **`chgrp` *`group filename`* *`filename …`***
  - Changes just the group ownership of a file or a directory.
  - On Extreme, the group is either "**`domain users`**" or a lab share group name provided by support staff.

Basic Linux Commands: Software & Modules

- **`which` *`executable`***
  - This command shows the default path to the specified ***`executable`***. Extreme provides multiple versions of software packages through modules; the output of this command may change depending on the module(s) loaded. When testing interactively, use this command to verify you are running the version you want.
- **`module` *`parameters`* `…`**
  - The **`module`** command is used to manage software packages on the cluster.
- **`module avail`**
  - Lists all the software modules available on the cluster.
- **`module load` *`modulename`***
  - This command loads the software package into your path. Keep in mind you must use this command in your submit scripts in order to call software packages.
- **`module list`**
  - This command displays active modules listed in the order they were loaded.
- **`module unload` *`modulename`***
  - This command removes the specified software package from your path.

Basic Linux Commands: MPI Modules Example

There are multiple implementations of the Message Passing Interface (MPI), a standard parallel computing framework. The environment has to be consistent between the building and execution of an MPI application. In this example, note how the path to **`mpirun`** (an MPI execution harness) changes depending on the module loaded.

```
$ which mpirun
/opt/openmpi/bin/mpirun

$ module load compilers/intel
$ which mpirun
/export/share/compilers/intel/impi/4.1.1.036/intel64/bin/mpirun

$ module load tools/mpich-3.0.4-icc
$ which mpirun
/export/share/tools/mpich-3.0.4-icc/bin/mpirun
```

Storage on the Cluster: File System Types & Paths

- **Home directories** are on **Persistent NFS Storage**
  - **`/export/home/`*`netid`***
- … as are **lab shares** (common space for research groups)
  - **`/export/home/`*`labshare`***

- Fast temporary **scratch** filesystem is on **Lustre**
  - **`/mnt/lustre/`*`netid`***

Storage on the Cluster: Lab Share Specifics

- How to **request** a lab share
  - Have the head of your lab group send an email to extreme@uic.edu specifying the need, beginning membership, the lab's cluster affiliation (department- or faculty-funded), and the point of contact for the lab/project responsible for maintaining the member access list.
- How to **gain access** to a lab share that already exists.
  - Send an email to extreme@uic.edu requesting access and cc: the faculty member or their designated membership manager. Please notify the responsible party in advance to ensure they receive and approve your request.

Basic Job Scripting: Parameter Checklist

- What are the main things to confirm when submitting a job?
  - **Cores** (number of CPU cores × number of nodes)
  - **Memory**
  - **Wall**clock **time** (expected duration of job run)
  - What is the **working directory** of your job
  - How to specify **input and output files**
  - What happens when you do not specify one of these parameters

Basic Job Scripting: Example Submit Script

```
#!/bin/bash
#PBS -l mem=20gb
#PBS -l walltime=20:00:00
#PBS -l nodes=1:ppn=8
#PBS -j oe
#PBS –m abe
#PBS -M email_address
#PBS -N jobname
#PBS -d /export/home/netid/work_dir

sleep 30
```

Basic Job Scripting: #PBS Headers

`#!/bin/bash`

Always specify the shell that the job script uses.

`#PBS -l mem=20gb`

This optional line tells the cluster how much memory your job intends to use and ensures that there is enough memory on the assigned nodes when you submit.

`#PBS -l walltime=20:00:00`

This line tells the cluster how long the job should run (HH:MM:SS).

`#PBS -l nodes=1:ppn=8`

This line specifies the number of nodes (physical compute nodes) and then the number of cores (processors) the job will need to run. The assigned resource is a product of these values.

Basic Job Scripting: #PBS Headers (con't.)

`#PBS -j oe`

Allows the user to join and otherwise manipulate the standard **o**utput and standard **e**rror into a single file.

`#PBS -m abe`

Requests a status email when a job **b**egins, **e**nds, or **a**borts.

`#PBS -M email_address`

Provides an email address in conjuction with the status email flag above.

`#PBS -N jobname`

Names the job with a custom label to allow a user to make it easily identifiable in the list of jobs (e.g., provided by **`qstat`**).

`#PBS -d /export/home/netid/work_dir`

Specifies the initial working directory for your job, generally where your job's data and/or program reside.

Basic Job Scripting: Running Your Application

```
sleep 30
```

Commands that follow the **#PBS** headers are what are actually executed on the cluster. In this example, the job simply sleeps for 30 seconds and then exits.

Depending on the complexity of your applications, you may also need to include **$PBS_…** variables. The most common is **$PBS_NODEFILE**, used in conjunction with MPI. This file contains one or more lines, each specifying a hostname for each core on the respective node(s) assigned to a job. Another useful variable is **$PBS_NP**, which is equal to the total number of cores assigned to the job (and is always equal to the line count of **$PBS_NODEFILE**).

Introduction to Job Submission

- Submit the job using a submit script requesting **one node**:
  - **`qsub -l nodes=1 submit_script`**
- Now request all **20** cores (**p**rocessors **p**er **n**ode) on **two nodes**:
  - **`qsub -l nodes=2:ppn=20 submit_script`**

The **`-l`** (lowercase L) precedes a comma-separated resource **l**ist. **`qsub`** can accept multiple "**`-l …`**" argument pairs. You may specify these parameters at the command line or in the script.

- Request an **i**nteractive job with **`-I`** (capital i):
  - **`qsub -I -l nodes=2:ppn=16`**

Job Monitoring

- **showq**
  - Displays information on all jobs that are active, queued, or blocked.
  - To only display only your jobs, pipe the output through **grep**:

    showq | grep *netid*
- **qstat**
  - An alternative queue monitoring application.
- **checkjob**
  - Good for gaining detailed information on an individual job or determine why it failed to run.
    - **checkjob -v *jobid***
      - Provides detailed information on the specified job and any error messages.
    - **checkjob -v -v *jobid***
      - Provides not only detailed information on the specified job and error messages, but displays the output of your submitted script.
- **qdel**
  - To use this command to cancel one of your jobs , use:

    qdel *jobid*

Basic Job Scripting: Invalid Resource Requests

~~`#PBS -l mem=256gb`~~

If you ask for a node with more than 128GB of memory the job will never run as each node only has 128 GB of RAM.

~~`#PBS -l walltime=720:00:00`~~

The maximum walltime is 240 hours (or 10 days). If you submit a job with a walltime longer than the maximum, the the job will not run.

~~`#PBS -l nodes=1:ppn=128`~~

The **`ppn`** value cannot exceed the per-compute core count. Extreme has 16-core "Generation 1" and 20-core "Generation 2" compute nodes. The default **`batch`** queue uses G1 nodes; **`edu_shared`** uses G2; users should inquire about the node generation for other queues. A valid equivalent to a 1 × 128 configuration would be **`nodes=8:ppn=16`**.

MPI Program Sample on Extreme

```
#include <mpi.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {

     int world_size;
     int rank;
     char hostname[256];
     char processor_name[MPI_MAX_PROCESSOR_NAME];
     int name_len;

     MPI_Init(&argc, &argv); // Initialize the MPI environment
     MPI_Comm_size(MPI_COMM_WORLD, &world_size);  // get the total number of processes
     MPI_Comm_rank(MPI_COMM_WORLD, &rank);        // get the processor rank number
     MPI_Get_processor_name(processor_name, &name_len);  // get the processor name

     gethostname(hostname, 255);                  // non-MPI function to get the
hostname
     printf("Hello world!  I am process number: %d from processor %s on host %s out of
%d processors\n", rank, processor_name, hostname, world_size);

     MPI_Finalize();

     return 0;
}
```


cribioinfo@uic.edu

Sample Script to Run MPI Program on Extreme

```
#!/bin/bash
#PBS -l nodes=2:ppn=20,walltime=1:00
#PBS -N MPIsample
#PBS -q edu_shared
#PBS -m abe
#PBS -M netid@uic.edu
#PBS -e mpitest.err
#PBS -o mpitest.out
#PBS -d /export/home/netid/MPIsample

module load compilers/intel

mpirun -machinefile $PBS_NODEFILE -np $PBS_NP ./mpitest
```

Building and Running MPI (Example)

0. Copy (**R**ecursively) the directory containing the sample code and job script to your home directory.

```
cp -R /export/share/classes/cs-ece566/MPIsample ~
```

1. Load Intel MPImodule before compiling the program.

```
module load compilers/intel
```

2. Compile the program

```
cd ~/MPIsample
mpicc -o mpitest mpitest.c
```

3. Modify and submit the job script to the queue:

```
qsub mpitest.pbs
```

Resources

- **UIC Account Requests**
  - http://extreme.uic.edu/requests-policies/adding-a-new-userclass/
- **Software Request**
  - http://extreme.uic.edu/requests-policies/software-request/
- **Available Software**
  - http://extreme.uic.edu/knowledge-base/available-software/
- **Service Alerts**
  - http://extreme.uic.edu/news-annoucements/service-alerts/

Additional Resources

- **General Resources**
  - http://software-carpentry.org/lessons.html
  - https://www.coursera.org
- **vi Editor Commands**
  - http://www.lagmonster.org/docs/vi.html
- **PBS Command Resource**
  - https://www.osc.edu/supercomputing/batch-processing-at-osc/pbs-directives-summary
  - http://www.nas.nasa.gov/hecc/support/kb/Commonly-Used-PBS-Commands_174.html
- **PuTTY Tutorials**
  - http://www.jfitz.com/tips/putty_config.html
  - http://www.gamexe.net/other/beginner-guide-ssh/

Contact Us

- Questions, comments, and issues should be sent to **extreme@uic.edu**
- Additional resources can be found at **http://extreme.uic.edu**
- Follow **@UIC_Extreme** on Twitter

UIC EXTREME COMPUTING