HCC Kickstart
Introduction to HCC
University of Nebraska
Holland Computing Center

October 15-16, 2018

Carrie Brown, David Swanson, Emelie Fuchs,
Adam Caprez, Caughlin Bohn, Huijun Zhu
Schedule

09:00 – 09:30 Overview of HCC

09:30 – 12:00 Introduction to HCC

12:00 – 1:00 Lunch / Ask the Users Panel

1:00 – 1:30 Using Globus for Data Management

1:30 – 3:00 Overview of Anvil

3:00 – 3:30 Using Jupyter Notebook on HCC

3:30 – 4:00 Introduction to OSG

4:00 – 5:00 One on One Consultations / Tour of HCC / Resource Planning Team Meeting

Upcoming Workshops

• UNL: Software Carpentry
  • Bash, Git & Python
  • December

• UNO: Data Carpentry
  • Genomics Data Analysis
  • January 2019
Intro to HCC: Teaching Objectives

• Login to HCC clusters using PuTTY(Windows)/Terminal(Mac).

• Create files on the cluster. Transfer files between your laptop and the cluster.

• Utilize modules (software) installed on the clusters.

• Submit jobs using SLURM job scheduler.
Logistics

- Location: Nebraska East Union – Arbor Suite
- Name tags, sign-in sheet
- Sticky notes: Red = need help, Green = all good
- Lunch is provided
- Link to attendee portal (with command history): [http://hcc.unl.edu/swc-history/index.html](http://hcc.unl.edu/swc-history/index.html)
- Link to schedule and lessons/slides: [http://hcc.unl.edu/hcc-kickstart-2018](http://hcc.unl.edu/hcc-kickstart-2018)
Intro to HCC: Teaching Objectives

- Login to HCC clusters using PuTTY (Windows)/Terminal (Mac).
- Create files on the cluster. Transfer files between your laptop and the cluster.
- Utilize modules (software) installed on the clusters.
- Submit jobs using SLURM job schedule.
What is a Cluster?
Crane
- 548 node Linux cluster
- 452 nodes with 64 GB RAM
  - 2.6 GHz 2 CPU/16 cores
- 59 nodes with 256 GB RAM
  - 2 CPU/36 cores per node
- 37 nodes with 512 GB RAM
  - 2 CPU/36 cores per node
- 1452 TB shared storage
  - 4 TB scratch per node

Sandhills
- 108 node “Condominium” Linux cluster
- 62 nodes with 192 GB RAM
  - 2.3 GHz and 64 cores/node
- 44 nodes with 128 GB RAM
  - 2.0 GHz and 32 cores/node
- 2 nodes with 156 GB RAM
  - 1.9 GHz and 48 cores/node
- 175 TB shared storage
  - 1.5 TB scratch per node

Tusker
- 82 node Linux cluster
- 452 nodes with 64 GB RAM
  - 2 CPU/16 cores
- 59 nodes with 256 GB RAM
  - 2 CPU/36 cores per node
- 37 nodes with 512 GB RAM
  - 2 CPU/36 cores per node
- 1452 TB shared storage
  - 4 TB scratch per node

Red
- 337 node Linux cluster
- 5888 cores
- 6.67 PB raw storage
- Used by HCC staff scientists to analyze LHC data
Anvil

- HCC’s Cloud
- Customizable virtual machines based on the OpenStack Software
- For projects that are not well served by a traditional Linux environment:
  - Highly interactive environments
  - Projects that require root access
  - Alternate operating systems
  - Test cluster environments
  - Cluster applications that require dedicated resources

Attic

- Near-line data archive
- Backed up in both Lincoln and Omaha for disaster tolerance
- 10 GB/s transfer speed to and from the clusters when using Globus Connect
- Cost per TB is significantly lower than commercial cloud services.
- No transfer limits or charges
Connecting to the Clusters
How to connect

Mac OS / Linux
• Open Terminal
• Type in the following command and press Enter:
  `ssh <user_name>@tusker.unl.edu`
  (Replace `<user_name>` with your HCC login)

Windows
• Open PuTTY
• Type `tusker.unl.edu` for Host Name and click Open

• On the second screen, click Yes

Once you connect, you will be prompted for your password. Nothing will appear on your screen as you type your password. This is normal.
Two-Factor Authentication

If you are using a cell phone:

• Press 1 to receive a push notification on your device
• Press 2 to receive an automated phone call. Then enter the passcode provided.
• Press 3 to receive a list of passcodes via text message. Enter the passcode which starts with the number indicated

If you are using a hardware token (Yubikey):

• Insert the Yubikey into the computer’s USB slot on your computer. A green light should come on inside the gold circle.
• Press your finger against the gold circle and a passcode will be generated.

The Duo Mobile app can also be used to generate passcodes, even when cell service is unavailable.
<table>
<thead>
<tr>
<th>Command</th>
<th>What it does</th>
<th>Example Usage</th>
</tr>
</thead>
</table>
| **ls**  | list: Lists the files and directories located in the current directory | **ls**  
  - Lists all files and directories in the current directory  
  **ls**  
  *.*.txt  
  - Lists all files which end with .txt in the current directory |
| **cd**  | change directory: this allows users to navigate in or out of file directories | **cd**  
  *.*.txt  
  - Navigates into the directory “dir_name”  
  **cd**  
  .  
  - Navigates out of the current directory and into the parent directory  
  **cd**  
  $WORK$  
  - Navigates to your default /work directory |
| **nano** | nano text editor: opens the nano text editor  
  Note: To access the menu options, ^ indicates the control (CTRL) key. | **nano**  
  - opens the text editor in a blank file  
  **nano**  
  *.*.txt  
  - opens the text editor with “file_name” open. If “file_name” does not exist, it will be created if the file is saved |
| **less** | less: opens an extended view of a file  
  Note: scroll using up and down arrows. To exit, press 'q' | **less**  
  *.*.txt  
  - opens an extended view of the file "file_name" |
Exercises

1. Type `exit` to logoff of the Tusker cluster.

2. Connect to the Crane cluster (hostname: `crane.unl.edu`)
   - Try using a different Two-factor authentication method if you have another option.

3. Change to your `$WORK` directory

4. Clone the tutorial files by entering the command:
   - `git clone https://github.com/unlhcc/job-examples.git`
     - Contributions to this repository are welcome!

   **Once you have finished, put up your green sticky note.**
   **If you have issues, put up your red sticky note and one of the helpers will be around to assist.**
Intro to HCC: Teaching Objectives

• Login to HCC clusters using PuTTY (Windows)/Terminal (Mac).

• Create files on the cluster. Transfer files between your laptop and the cluster.

• Utilize modules (software) installed on the clusters.

• Submit jobs using SLURM job schedule.
# Home vs Work vs Common

**$HOME - /home/group/user**
- Not intended for I/O to active jobs
- Quota-limited to 20GB *per user*
- Backed up for best-effort disaster recovery
- Meant for items that take up relatively small amounts of space.
  - Source code
  - Program binaries
  - Configuration files
- *Read-Only to worker nodes*

**$WORK - /work/group/user**
- Designed for I/O for running jobs
- 50TB quota *per group*
- Not backed up
- To check group usage:
  - `hcc-du`
- **Purge policy**
  - Files will be deleted after 6 months inactivity
  - Files eligible for purge can be checked using `hcc-purge`
$COMMON- /common/group/user

- Mounted on all HCC clusters
- Not intended for **heavy** I/O to active jobs
- Quota-limited to 30TB *per group*
- Not backed up
- Not purged
- Meant for items that read to be accessed from multiple clusters.
  - Shared data files
  - Reference databases
  - etc.
Home vs Work vs Common vs Attic

- Faster File Access (higher bandwidth / IOPS)
- Ease of Access
- Diminishing Redundancy (backups)

Attic
- Globus Access
- Additional storage for fee
- Not mounted to clusters
- Off-site Backups

Home
- Medium Speed Access
- Individual quotas (20 GB)
- Mounted on single cluster
- Daily backups

Common
- Fast Access
- Group quotas (30 TB)
- Additional storage for fee
- Mounted on all clusters
- Not backed up

Work
- Fastest Access
- No paid access
- Group quotas (50 TB)
- Mounted on single cluster
- Not backed up
- Subject to purge
$WORK/$COMMON Usage Best Practices

• **Avoid large numbers of files**
  • Having hundreds of files in your directories can drastically degrade the performance of the file system

• **Storage resources are finite**
  • Be judicious about the data you store.
  • Periodically review your files and carefully consider what you need; remove what is no longer needed.

• **Use Globus for file transfers**
  • Globus Connect is a fast and robust file transfer service that allows users to quickly move large amounts of data between computer clusters and to and from personal workstations

• **Backup valuable data**
  • Users are responsible to backup their valuable research data on $WORK periodically
Transferring Files

• **Transfer files using an SCP client**
  • WinSCP ([http://winscp.net](http://winscp.net)) – Windows
  • Cyberduck ([http://cyberduck.io](http://cyberduck.io)) – MacOS
  • Filezilla ([http://filezilla-project.org](http://filezilla-project.org)) – Windows, MacOS and Linux

• **Globus Connect**
  • [http://globus.org](http://globus.org)
  • Fast, secure and robust transfers with user-friendly web interface
  • Uses the High-Speed transfer nodes by default
  • Can transfer directly between clusters, Attic and personal machine

• **scp**
  • Usage: `scp user@host:source_file user@host:target_file`
  • Example: `scp my_file.txt demo01@crane.unl.edu:$WORK/my_file.txt`
Exercises

1. In your $HOME directory create a file called ‘bio.txt’ – edit this file to include your name and department. (type command ‘cd’ to switch to $HOME)

2. If you are using a Windows laptop, download and setup WinSCP. Or if you are using a Mac, download and setup Cyberduck.
   • Windows instructions: https://hcc-docs.unl.edu/x/2IEr
   • Mac instructions: https://hcc-docs.unl.edu/x/2oEr

3. Copy the ‘bio.txt’ file from Crane $HOME to your local laptop.

4. Copy the ‘bio.txt’ file from your local laptop to Crane $WORK.

Once you have finished, put up your green sticky note.
If you have issues, put up your red sticky note and one of the helpers will be around to assist.
Intro to HCC: Teaching Objectives

• Login to HCC clusters using PuTTY (Windows)/Terminal (Mac).

• Create files on the cluster. Transfer files between your laptop and the cluster.

• Utilize modules (software) installed on the clusters.

• Submit jobs using SLURM job schedule.
Running Applications

• All applications installed on HCC clusters are loaded in individual modules implemented with Lmod
  • Lua based module system

• Modules dynamically change the user environment
  • $PATH
  • $LD_LIBRARY_PATH

• Hierarchical structure
  • If module A is dependent on module B, then module B must be loaded first to load module A

• Typically follows the naming convention <software>/<version>
  • Example: python/2.7

• Load using the module command
<table>
<thead>
<tr>
<th>Command</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>module avail</td>
<td>Lists all modules available to be loaded</td>
</tr>
<tr>
<td>module spider &lt;name&gt;</td>
<td>Information about a specific module – can also be used to search</td>
</tr>
<tr>
<td>module load &lt;module_name&gt;</td>
<td>Load module(s) – can load a list of space delimitated modules</td>
</tr>
<tr>
<td>module unload &lt;module_name&gt;</td>
<td>Unload module(s) – can unload a list of space delimitated modules</td>
</tr>
<tr>
<td>module purge</td>
<td>Unloads all currently loaded modules</td>
</tr>
<tr>
<td>module list</td>
<td>Lists all currently loaded modules</td>
</tr>
</tbody>
</table>

For more information:
- module --help

- Available software lists for each cluster:
  - Crane: [https://hcc-docs.unl.edu/x/OwBq](https://hcc-docs.unl.edu/x/OwBq)
  - Tusker: [https://hcc-docs.unl.edu/x/LQF9](https://hcc-docs.unl.edu/x/LQF9)
  - Sandhills: [https://hcc-docs.unl.edu/x/LQF9](https://hcc-docs.unl.edu/x/LQF9)
Exercises

1. Find a list of ‘Core’ modules that do not have any dependences.

2. Load one or more of the modules from the list you just found. Now unload them. Try doing it two different ways.

3. Lookup information about the MATLAB module.
   - How many different versions are available?
   - Load a specific version using the `<module_name>/<version>` name

4. Try loading the lammps/10Feb2017 module. Did it work?
   - Now load these modules first
     - compiler/intel/15
     - openmpi/1.10
   - Now try loading lammps/10Feb2017 again. Does it work now? Why?
   - Unload all modules with only one command.

Once you have finished, put up your green sticky note.

If you have issues, put up your red sticky note and one of the helpers will be around to assist.
Intro to HCC: Teaching Objectives

• Login to HCC clusters using PuTTY (Windows)/Terminal (Mac).

• Create files on the cluster. Transfer files between your laptop and the cluster.

• Utilize modules (software) installed on the clusters.

• Submit jobs using SLURM job schedule.
Running Jobs

• All software and analysis must be done on the worker nodes
• Processes started on the login node will be killed
  • Limit usage to brief, non-intensive tasks like file management and editing text files
• HCC uses the SLURM scheduler to manage and allocate resources
  • Resources are allocated based on the Fair Use Algorithm
• Jobs can be run in interactive or batch format
Interactive vs Batch Jobs

Batch Jobs

• The user creates a **submit file** which contains all commands and job information and add its to the queue
  • Similar to running Bash scripts
• Holds job information until resources become available
  • User can disconnect and the job will remain queued
• Uses the **sbatch** command

Interactive Jobs

• Allows the user to type in commands and run programs **interactively**
• Must remain connected and wait for resources to be allocated
• Job can be interrupted
• Uses the **srun** command
Batch Jobs

• To create a batch job, the user must first make a submit script
  • Submit scripts include all job resource information:
    • number of nodes/cores
    • required memory
    • Runtime
  • If the job exceeds the requested memory or time, it will be killed.

• Submit script is then added to the job queue using the `sbatch` command

• `squeue` will show queued and running jobs

• `sacct` can be used to find information about completed jobs
# Common SBATCH Options

<table>
<thead>
<tr>
<th>Command</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>--nodes</td>
<td>Number of nodes requested</td>
</tr>
<tr>
<td>--time</td>
<td>Maximum walltime for the job – in DD-HHH:MM:SS format – maximum of 7 days on batch partition</td>
</tr>
<tr>
<td>--mem</td>
<td>Real memory (RAM) required per node - can use KB, MB, and GB units – default is MB Request less memory than total available on the node - The maximum available on a 512 GB RAM node is 500, for 256 GB RAM node is 250</td>
</tr>
<tr>
<td>--ntasks-per-node</td>
<td>Number of tasks per node – used to request a specific number of cores</td>
</tr>
<tr>
<td>--mem-per-cpu</td>
<td>Minimum of memory required per allocated CPU – default is 1 GB</td>
</tr>
<tr>
<td>--output</td>
<td>Filename where all STDOUT will be directed – default is slurm-&lt;jobid&gt;.out</td>
</tr>
<tr>
<td>--error</td>
<td>Filename where all STDERR will be directed – default is slurm-&lt;jobid&gt;.out</td>
</tr>
<tr>
<td>--job-name</td>
<td>How the job will show up in the queue</td>
</tr>
</tbody>
</table>

For more information:
- `sbatch -help`
- SLURM Documentation: [https://slurm.schedmd.com/sbatch.html](https://slurm.schedmd.com/sbatch.html)
Submit Scripts

Shebang
The shebang tells Slurm what interpreter to use for this file. This one is for the shell (Bash).

Name of the submit file
This can be anything. Here we are using “invert_single.slurm” the .slurm makes it easy to recognize that this is a submit file.

SBATCH options
These must be immediately after the shebang and before any commands.

Commands
Any commands after the SBATCH lines will be executed by the interpreter specified in the shebang – similar to what would happen if you were to type the commands interactively.

The only required SBATCH options are time, nodes and mem, but there are many that you can use to fully customize your allocation.
Determining Parameters

How many nodes/memory/time should I request?

• **Short answer:** We don’t know.

• **Long answer:** The amount of time and memory required is highly dependent on the application you are using, the input file sizes and the parameters you select.
  
  • Sometimes it can help to speak with someone else who has used the software before.
  
  • Ultimately, it comes down to trial and error
    
    • Check the output and utilization of each job will help you determine what parameters you will need in the future.
    
    • Trying different combinations and seeing what works and what doesn’t.
Submit Files Best Practices

• Put all module loads immediately after SBATCH lines
  • Quickly locate what modules and versions were used.

• Specify versions on module loads
  • Allows you to see what versions were used during the analysis

• Use a separate submit file for each job
  • Instead of editing and resubmitting a submit files – copy a previous one and make changes to it
  • Keep a running record of your analysis

• Redirect output and error to separate files
  • Allows you to see quickly whether a job completes with errors or not

• Separate individual workflow steps into individual jobs
  • Avoid putting too many steps into a single job
Exercises

1. Navigate into the matlab directory of the job-examples directory and locate the invertRand.submit file
   • This submit file runs a MATLAB script which inverts a 10,000 x 10,000 randomly generated matrix and outputs the length of time it took to perform the inversion.
   • Look at the contents of invertRand.submit - How many nodes this will run on? How many cores? How much memory and time is requested?
   • Submit the invertRand.submit job. Check the output to see how long it took to invert the matrix.

2. Copy the invertRand.submit file to a new file using the `cp` command.
   • Edit the invertRand.submit file to use 4 cores.
     • If you need help, look at the InvertRandParallel.submit file which is already configured to run on 4 cores
   • Submit the job again and compare the time to your initial run. How much faster or slower is it?
   • Compare times with your neighbor. Did you see the same amount of improvement?

3. If there’s time, try different combinations of SBATCH commands and see how the running time changes.

Once you have finished, put up your green sticky note.
If you have issues, put up your red sticky note and one of the helpers will be around to assist.
Interactive Jobs

- Interactive jobs work very similarly to batch jobs

- Once resources are allocated, commands can be input interactively
  - All output is directed to the screen
Exercises

1. Request an interactive job for certain amount of resources
   • If you can’t think of a setup, use one of these:
     • 1 node, 1 core with 2 GB RAM each core
     • 2 nodes, 1 core per node with 2 GB RAM each core
   • How long did it take to allocate resources? Compare your set up and wait time with your neighbor. Did they get resources faster or slower than you did?

2. Using your interactive job from above, load the module for your favorite programming language (Python, R, MATLAB, etc.)
   • Run the program interactively

   Once you have finished, put up your green sticky note.

   If you have issues, put up your red sticky note and one of the helpers will be around to assist.
Asking for help

• **Read the Documentation**
  • [http://hcc-docs.unl.edu](http://hcc-docs.unl.edu)
    • If the documentation doesn’t answer your question, leave a comment or email to let us know!

• **Consult Google**
  • Useful for application specific errors or general usage

• **Contact Us**
  • Drop in our offices
    • UNL: Room 118 in the Schorr Center on City Campus
    • UNO: Room 158 in the Peter Kewit Institute on Center Campus
  • Email [hcc-support@unl.edu](mailto:hcc-support@unl.edu)