Introduction to High Performance Computing

Ping Luo

Third Annual Texas A&M Research Computing Symposium

Prerequisites

- This workshop is designed for
  - No prior experience with HPC
  - No prior knowledge about Linux
  - Needs to use HPC for their work
- A valid account on Ada/Terra

Schedule

https://luop0812.github.io/HPRC_IntroHPC/

	Setup	Download files required for the lesson
00:00	1. Introduction to HPC	What is High Performance Computing? Why should I be using High Performance Computing for my research? Don't I need to know how to program to use High Performance Computing?
00:10	2. Connect to the HPC	How do I connect to an HPC system?
00:25	3. Using a cluster: Introduction	What is a cluster? How does a cluster work? How do I log on to a cluster?
00:45	4. Using a cluster: Scheduling jobs	What is a scheduler and why are they used? How do we submit a job?
01:45	5. Using a cluster: Accessing software	How do we load and unload software packages?
02:20	6. Using a cluster: Using resources effectively	How do we monitor our jobs? How can I get my jobs scheduled more easily?
02:50	7. Basic UNIX Commands	What is the syntax of UNIX commands? How do I navigate the file system? How do I transfer files to HPC? How do I interact with files on the HPC?
03:20	Finish	

Teaching Methodology

- Use the tutorial that was developed based on the Software Carpentry method
  - Overview
    - Questions
    - Objectives
  - Main content
  - Key points
  - Expected time for teaching and exercise
  - Also suitable for self-study
- Content
  - Linux Command lines
  - With the OnDemand web portal
- A lot of hands-on

1: Introduction to HPC

Questions

- What is High Performance Computing?
- Why should I be using High Performance Computing for my research?
- Don’t I need to know how to program to use High Performance Computing?

Objectives

- Understand what an HPC is.
- Understand how HPC can improve your research efficiency.
- Understand the ‘barrier to entry’ is lower than you’d think.

What is High Performance Computing (HPC)

- A name given to the use of computers with capabilities beyond the scope of standard desktop computers
- HPC systems usually have
  - More powerful Central Processing Units (CPUs)
  - More CPUs
  - More memory
  - More storage
  - Faster connection to other computers
  - Special hardware: Nvidia GPU, Intel MIC, FPGA, etc

How to interact with HPC

- Through Shell
  - A program that executes other programs in the Operating System
  - Provides a command line interface to the Operating System
  - Bash is a popular Shell program used with HPC
    - The default shell used on Ada/Terra
- Through batch jobs
  - Most common way
  - A batch job is a script to finish some work:
    - Resource specifications
    - Commands
  - A batch scheduler is needed to submit and schedule jobs

Why HPC

- Large problems
  - Computational intensive
    - Too many small tasks
    - A very large task that must be done with many CPUs at once
  - Data intensive
- Benefits
  - Speed
  - Volume
  - Free
  - Convenience
- Ways to access HPC
  - Traditional command line: Unix/Linux commands, Batch jobs, modules
  - The Open OnDemand web portal
    - Provides a single access point with a web browser
    - No need to install any client
    - Intuitive user interface makes it easier and more convenient to access and interact with HPC

2: Connect to HPC

Questions

- How do I connect to an HPC system?

Objectives

- Be able to connect to a remote HPC system

Connect to HPC

- Unlike working with a PC, we work with HPC remotely
- SSH (**S**ecure **SH**ell)
  - For remote login to an HPC system
  - Linux/MacOS: ssh is available through a terminal
  - Windows: putty or MobaXterm
- Parameters
  - Server address
  - Port number (default is 22)
  - User ID
- Connect with OnDemand portal

3: Using a cluster: Introduction

Questions

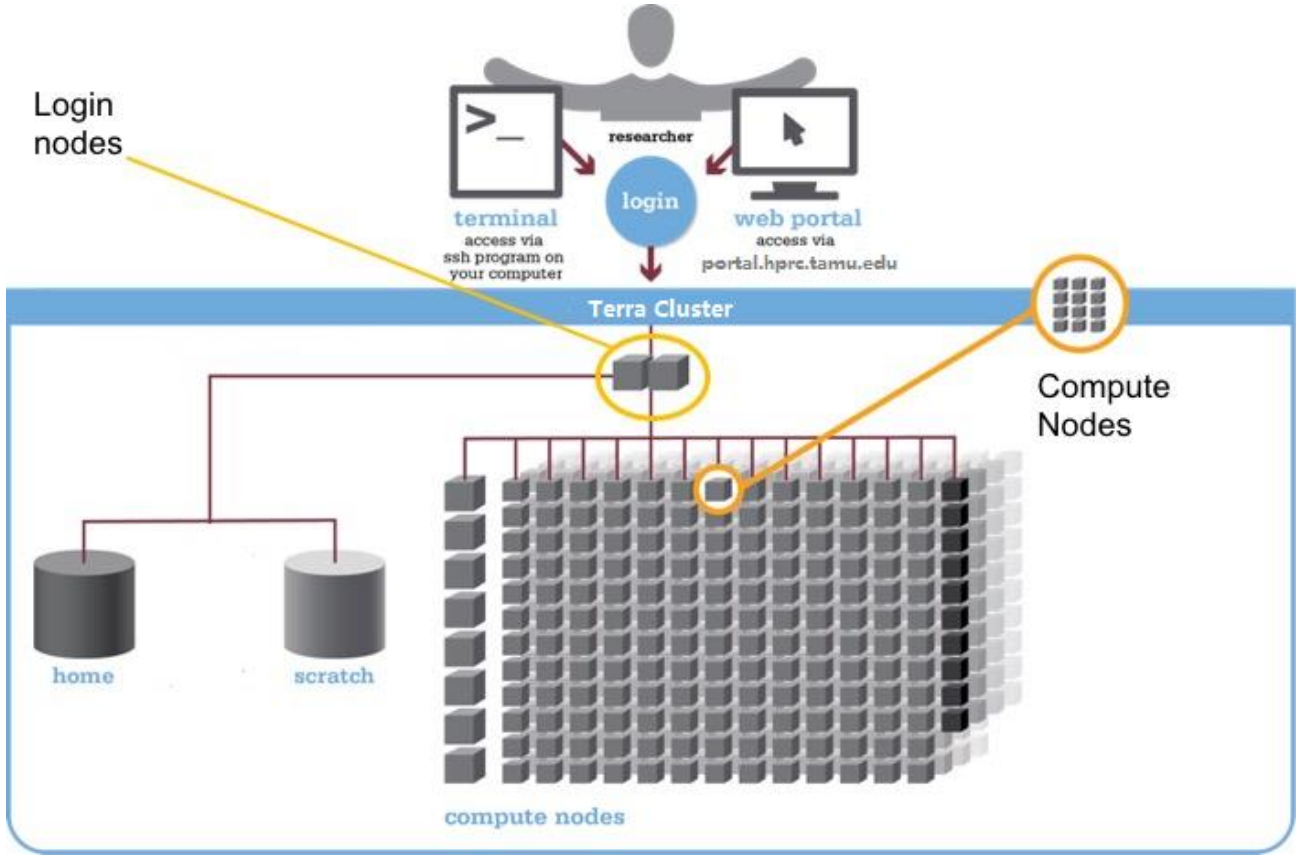
- What is a cluster?
- How does a cluster work?
- How do I log on to a cluster?

Objectives

- Connect to a cluster.
- Understand the general cluster architecture.

HPC cluster

- A network of computers



When do you need to use a cluster

- You need access to large numbers of CPUs.
- You need to run a large number of jobs.
- Your jobs are running out of memory.
- Perhaps you need to store tons and tons of data.
- You require an exceptionally high-bandwidth internet connection for data transfer.
- You need a safe archival site for your data.
- Your compute jobs require specialized GPU or FPGA hardware.
- Maybe your jobs just take a long time to run.

Hands-on

- Check host name
- List all hosts
- File systems
- Check quota
- Transfer files
  - scp, winscp, filezilla
  - OnDemand

4: Using a cluster: Scheduling jobs

Questions

- What is a scheduler and why are they used?
- How do we submit a job?

Objectives

- Submit a job and have it complete successfully.
- Understand how to make resource requests.
- Submit an interactive job.

Why batch scheduler

- A cluster contains hundreds and thousands of interconnected computers (nodes)
- Shared by a large number of users
- Challenges
  - Who gets what and when
  - Fair and efficient
- Batch scheduler
  - Special software designed to meet the challenges
- Slurm, LSF, PBS, etc
  - Same concepts, similar syntax, different key words
  - If you learned one, it is easy to pick up the others
  - Luckily, it is not so difficult to learn how to use a batch scheduler
    - Job script (Job specifications), batch commands, environment variables, queues

Job specifications

```
#!/bin/bash
#SBATCH --job-name=test_script
#SBATCH --time=00:03:00
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=2
#SBATCH --mem=2560M
#SBATCH --output=test.out

echo 'This script is running on:'
hostname
echo 'The date is :'
date
sleep 120
```

- Start with a special directive
  - #SBATCH
  - #BSUB
- To specify resource requirements
  - Walltime
  - Node, CPU, processor, core, task
  - Memory
- And other things
  - Job name
  - Account
  - Email
  - Output file

Batch commands

- Job submission
- Job monitoring
- Job operation
  - Delete
  - modify
- Accounting and history

Environment variables

- Input environment variables
  - Upon start, sbatch will read and handle the options set in these variables
  - Users seldom need to mess with these environment variables
- Output environment variables
  - Slrum will set these environment variables in the batch job
  - A few of them are useful
    - identify nodes the job is working on
    - Get the job id

Queues

- Software structure used by the job scheduler to group jobs
  - Resource requirements
  - Special requests
- Queues from Terra

Queue	Job Max Cores / Nodes	Job Max Walltime	Compute Node Types	Per-User Limits Across Queues
short	448 cores / 16 nodes	30 min / 2 hr	64 GB nodes (256) 128 GB nodes with GPUs (48)	1800 Cores per User
medium	1792 cores / 64 nodes	1 day		
long	896 cores / 32 nodes	7 days	64 GB nodes (256)	
xlong	448 cores / 16 nodes	21 days	64 GB nodes (256)	448 cores per User
gpu	1344 cores / 48 nodes	2 days	128 GB nodes with GPUs (48)	

5: Using a cluster: Accessing Software

Questions

- How do we load and unload software packages?

Objectives

- Understand how to load and use a software package.

What is module and why do we need it

- Software on HPC is not loaded by default
  - You cannot use it immediately after you log onto an HPC
- It is related to software incompatibility and versioning
- The solution is environment modules
  - Each module is a self-contained software packages
    - Contains all the files required to run the software package
    - Loads dependent modules
  - LMOD is the software that provides management of the modules
- All you need is to remember a few commands
  - Check available modules
  - Load/unload a module
  - List loaded module
  - Search a module

Install software of your own

- There are situations you need to build software
  - We have a LOT of pre-installed software packages, but no one can provide all
  - You need to add your own code into a software package
- Steps
  - Download the software
  - Read the instruction (very important)
  - Install dependency
  - Compile
  - Start using the software
- We can help if you get stuck

6: Using a cluster: Using resources efficiently

Questions

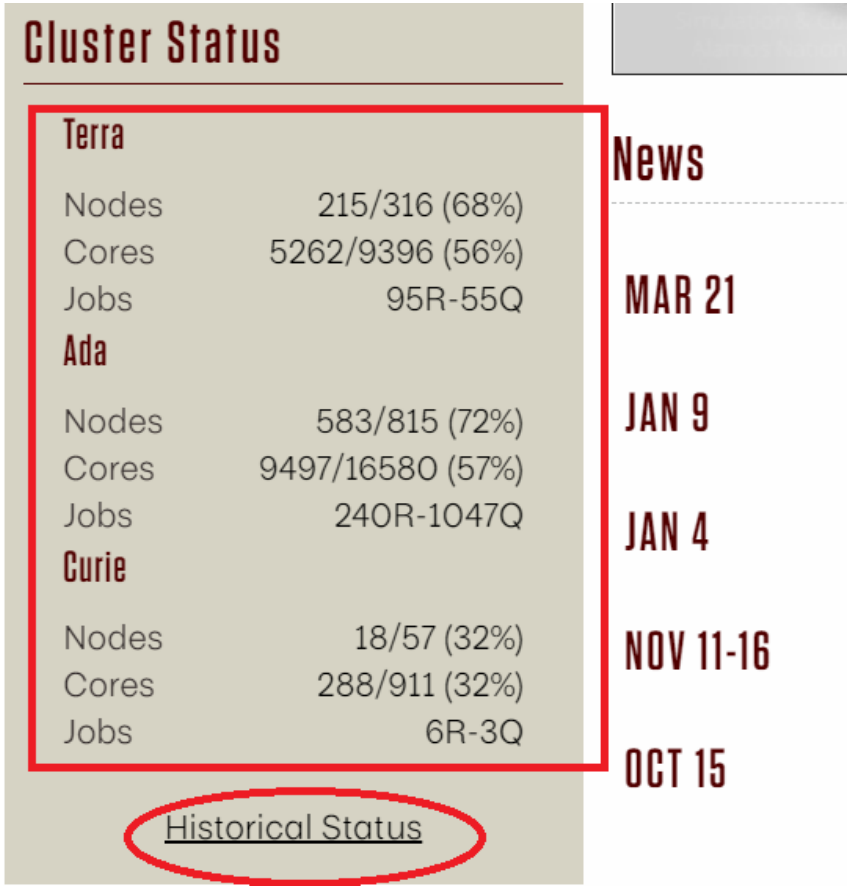
- How do we monitor our jobs?
- How can I get my jobs scheduled more easily?

Objectives

- Understand how to look up job statistics and profile code.
- Understand job size implications.
- Be a good person and be nice to other users.

Which cluster to choose?

- Ada, Terra, Curie, Lonestar
- Curie is Power7+, the rest are X86-64
- Lonestar is maintained by TACC
  - Follow TACC user guide
- Ada and Terra
  - Most of our users use one of these two or both systems
  - Ada is larger but older; Terra is smaller but newer
  - Ada uses LSF; Terra uses Slurm
  - Ada has 20 cores / node; Terra has 28 cores / node
  - Software environment is similar
  - Easy to switch between the two systems
    - Need to convert the batch script
    - May need to recompile your code
  - Choose the one that is least busy
- Some times fixing one cluster is easier. Ex:
  - large data files need be transferred



Estimate required resources

- Can we know how much resource we need? (walltime, memory, number of cores)
  - Not until you run it
- Best way to find out is to submit a test job

Rule of Thumb

- Don’t run large jobs on the login nodes
- Login nodes are for something small and quick
- Compress files before transferring
- Use some VCS system like ‘git’ to keep track of your code
- Backup your code by yourself
- Before submitting a run for jobs, submit one as a test to find out how much resources are needed
- Less resources you ask for, the faster your job can find enough resources to run
- You can install software by yourself in your scratch. If it is something of common interest by many users, we will install it in a shared location
- Always use the default compilers first, unless your code breaks

Acknowledgement

- The online course material in the Software Carpentry format is adapted from Ohio Supercomputing Center’s Introduction to HPC.