High-Performance Computing

Mesfin Diro Chaka
Computational Science
Addis Ababa University
mesfin.diro@aau.edu.et

Addis Ababa, Ethiopia

May 20, 2019

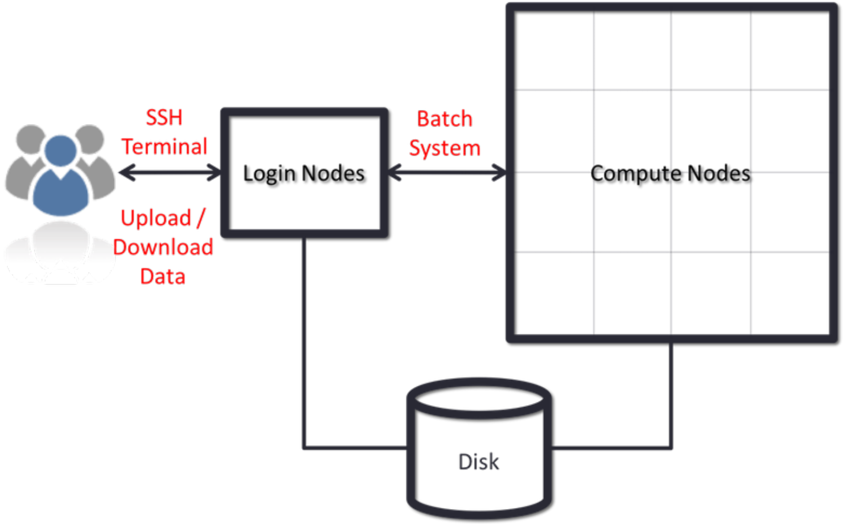
What is High Performance Computing (HPC)?

- High Performance Computing (HPC) is the name given to the use of computers with capabilities well beyond the scope of standard desktop computers.
- HPC typically involves connecting to very large computing systems elsewhere in the world.
- Using HPC systems often involves the use of a shell and either specialized software or programming techniques.
- Interacting with the shell is done via a command line interface (CLI) on most HPC systems.
- Typing-based interfaces are often called a command-line interface, or CLI, to distinguish it from a graphical user interface, or GUI, which most people now use.
- The heart of a CLI is a read-evaluate-print loop, or REP

What is High Performance Computing . . .

- HPC systems are generally constructed:
    - from many individual computers
    - similar in capability to many personal computers
    - connected together by some type of network
- HPC systems often include several different types of nodes:
    - Master(or front-end or login) nodes
    - Compute nodes

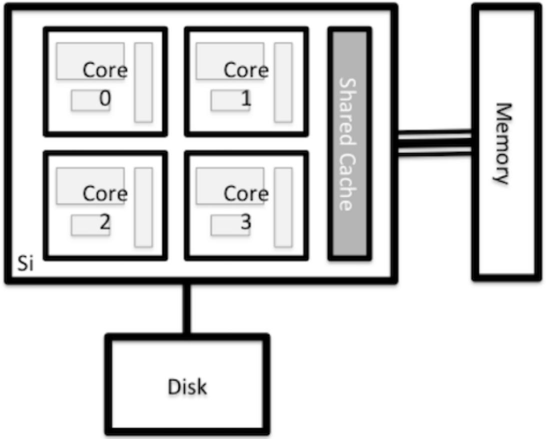
What is High Performance Computing. . .



What is High Performance Computing. . .

Nodes

Each node on an HPC system is essentially an individual computer



What is High Performance Computing. . .

Parallel programming models

- The two main programming models are:
  - Shared memory program (or multithreading) runs on only one node because, like the name says, all the memory has to be accessible to all the processes.
  - Message passing programming (e.g. MPI, message passing interface) can run on multiple nodes interconnected with the network via passing data through MPI software libraries.
  - OpenMP is a standard de facto for the multithreading implementations.

What is High Performance Computing. . .

Nodes

- The processor contains multiple compute cores (usually shortened to core.)
- Each core contains a floating point unit (FPU) which is responsible for actually performning the computations.
- The compute power of a processor generally depends on three things:
  - The speed of the processor (2-3 GHz are common speeds on modern processors)
  - The power of the floating point unit(FPU)
  - The number of cores available
- Each node also has a certain amount of memory available
- each node also has access to storage (file system) for persistent storage of data.

What is High Performance Computing. . .

Scheduler

- To manage the sharing of the compute nodes among all of the jobs, HPC systems use a batch system(scheduler).
- The batch system usually has commands for:
  - submitting jobs
  - inquiring about their status
  - and modifying them
- a typical HPC workflow could look something like this:
  1. Transfer input datasets to the HPC system (via the master nodes)
  2. Create a job submission script to perform your computation (on Master nodes)
  3. Submit your job submission script to the scheduler (on Master nodes)
  4. Scheduler runs your computation (on the compute nodes)
  5. Analyse results from your computation (on the login or compute nodes)

What is High Performance Computing...

Storage and File Systems

- HPC systems often involves very large files, and/or many of them.
- HPC systems have specialized file systems that are designed to meet different needs.
- most HPC systems often have several different file systems available

What is High Performance Computing. . .

Accessing Software on HPC Systems

- HPC systems often provide a lot of different software packages
- HPC provides ways of selecting and configuring them to get the environment you need.
- It has multiple versions of commonly used software packages installed called **Environmental Modules**

Why Use a Cluster(HPC)?

The benefits of using HPC systems for research often far outweigh the cost of learning to use a Shell and include:

- **Speed.** With many more CPU cores, often with higher performance specs, than the computers most people have access to HPC systems can offer significant speed up.
- **Volume.** Many HPC systems have both the processing memory (RAM) and disk storage to handle very large amounts of data.
- **Efficiency.** Many HPC systems operate a pool of resources that are drawn on by a many users.
- **Cost.** Bulk purchasing and government funding mean that the cost to the research community for using these systems in significantly less that it would be otherwise.
- **Keep personal resources free.** By using an HPC system when required your own personal computer can be used for other things to which it is better suited, like email and spreadsheets.

Cloud vs Cluster

- High Performance Computing is not the same as cloud computing.
- HPC targets extremely large sets of data and crunching the information in parallel while sharing the data between compute nodes
- In HPC one application can be run across a variable number of nodes. We call this **vertical scalability**.
- Cloud computing on the other hand targets “embarrassingly parallel problems” (EPP) with little or no effort is required to separate the problem into a number of parallel tasks.
- In a cloud several applications (or, copies of the same application) run on several nodes. We call this **horizontal scalability**.

Connecting to the HPC system

Workflow

The workflow for using HPC typically consists of the following steps:

1. Login to HPC login/head node.
2. Organize workspace.
3. Transfer data and files.
4. Install/run software on HPC.
5. Test your job interactively on a compute node.
6. Submit your job to the batch processor, to run it remotely on a compute node.
7. Monitor your job and check your results when it has completed.

Connecting to the HPC system . . .

Logging onto HPC

- Connecting to an HPC system is most often done through a tool known as “SSH” (Secure SHell)
- To begin using an HPC system we need to begin by opening a terminal.
- SSH allows us to connect to Linux computers remotely, and use them as if they were our own.
- Let’s attempt to connect to the HPC system now:

```
ssh yourUsername@10.4.17.30
```

Connecting to the HPC system . . .

Examining the nodes

- Now we can log into the Entoto HPC system we will look at the nodes.
- There are at least two types of node on the system: **login nodes** and **compute nodes**.
- We can use the `lscpu` command to print information on the processors on the login nodes to the terminal:

```
[remote]$ lscpu
```

- us the total amount of memory available so we use the `head -1` command:

```
[remote]$ head -1 /proc/meminfo
```

- The `qsub` command is used to submit a job to the scheduler.

Accessing Software

- Multiple versions of software are available on HPC systems
- The three biggest factors to have multiple versions of software are :
  - software incompatibilities;
  - versioning;
  - dependencies.
- Environment modules are the solution to these problems.
- A module is a self-contained description of a software package
- There are a number of different environment module implementations commonly used on HPC systems: the two most common are **TCL** modules(Tmod) and **Lmod**.
- One major difference between the two tools is that Lmod is written in Lua and not TCL.
- Lmod: An Environment Module System based on Lua, Reads TCL Modules, Supports a Software Hierarchy
- Lmod has to translate TCL into Lua

Accessing Software . . .

- You can use the **module list** command to see which modules you currently have loaded in your environment.
- If you have no modules loaded, you will see a message telling you so

```
[remote]$ module list
```

```
    No Modulefiles Currently Loaded.
```

- To see available modules, use `module avail`

```
[remote]$ module avail
```

- To load a software module, use `module load`. Hence We can load the `openmpi-x86_64` command with `module load`:

```
[remote]$ module load openmpi-x86_64
[remote]$ which openmpi-x86_64
```

Transferring files

Grabbing files from the internet

- the easiest tool to download files from the internet is `wget`.
- The syntax is relatively straightforward:

```
[remote]$ wget https://mesfind.github.io/hpc/files/cfd.tar.gz
```

Transferring files

Transferring with scp

- To copy a single file to or from the remote system, we can use scp.

```
[local]$ scp local-file.txt yourUsername@remote.computer.address:
```

- To transfer *to* another computer:

```
[local]$ scp file.txt yourUsername@remote.computer.address: path
```

Transferring files . . .

1. How to Connect to SFTP
   - sftp is an interactive way of downloading and uploading files.
   - Let's connect to a remote system using sftp

```
[local]$ sftp yourUsername@10.4.17.30
```

Transferring files . . .

2. Getting Help

- We can see which commands are available with `?` or `help`:

```
sftp> ?
sftp> help
```

Transferring files . . .

3. Check Present Working Directory

- To show our remote working directory:

```
sftp> pwd
```

- To show our local working directory, we add an `l` in front of the command:

```
sftp> lpwd
```

Transferring files . . .

4. Listing Files

- Listing files and directories in local as well as remote system.

```
sftp> ls
```

-
  - Listing files and directories in local as well on local system.

```
sftp> lls
```

Transferring files . . .

5. Upload a File

- we can put single or multiple files in remote system.
- To upload a file, we type `put some-file.txt`

```
sftp> put input.dat
```

Transferring files . . .

6. Upload multiple Files

```
sftp> mput *.c
```

Transferring files . . .

7. Download File

- To download a file we type `get some-file.txt`:

```
sftp> get input.dat
```

Transferring files . . .

8. Download multiple Files

- Get multiple files on a local system

```
sftp> mget input.dat
```

Transferring files . . .

9. Switching Directories

- Switching from one directory to another directory in local and remote locations.
- on a remote system

  ```
  sftp> cd test
  ```
- on a local system

  ```
  sftp> lcd Desktop
  ```

Transferring files . . .

10. Create Directories

- Creating new directories on local and remote locations.

```
sftp> mkdir test
sftp> lmkdir openmpi
```

How does parallel computing work

Types of parallelism

- There are different parallel strategies available to get performance on HPC systems.
- Many HPC users will encounter two forms of parallelism:
  - Shared Memory Parallelism
  - Distributed Memory Parallelism(MPI)
- Common terms to look for are **OpenMP and Threading** to help identify Shared Memory type of parallelism.
- The typical term you will see that identifies tDistributed Memory type of parallelism is **MPI**.
- Most HPC systems allow you to use different parallel strategies in combination.

Benchmarking

- Benchmarking provides you with insight into how well something performs in a controlled environment.
- Being able to estimate the average runtime for a single job will allow you to:
    - right size your jobs and reduce the wait time for you jobs to start.
    - scope out your computational research timelines.
    - prepare better applications for access to large HPC centres

Benchmarking checklist

- To start you will need:
  - know how your application(s) is parallelized shared memory or distributed memory (or a combination of the two known as hybrid)
  - a representative test case, ideally something you have run before!
  - a method to time the runs, here a couple of examples