High-Performance Computing 101

Health Data Working Group

Osvaldo Espin-Garcia

January 24, 2022

Learning objectives

1. Introduce scope, utility and capabilities of HPC systems
2. Present relevant topics on HPC:
   - job scheduling and monitoring
   - software and libraries usage
   - efficient data creation and management
3. Provide some resources for HPC learning at UofT (and beyond)

Audience interests

If you are here for this workshop you may fall into one of these categories

1. Completely new to HPC
2. Modest experience in HPC and looking to get useful pointers
3. Seem like an interesting thing to do today

What is HPC? - Expectation

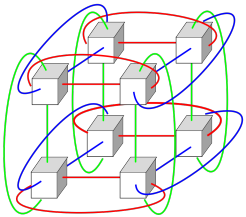
Figure 1: Advertising a supercomputer facility

What is HPC? - Reality

Figure 2: Niagara supercomputer at UofT

HPC - in short

- Practice of aggregating computing power (measured in FLOPS)
- Achieved by 'clustering' many smaller computers and processors
  - Each of these computers is called a 'node'
  - Nodes communicate using a dedicated network
- Deliver higher performance than a typical workstation
- Aims to solve large problems in science, engineering, or business



HPC - some applications

- Quantum mechanics
- Weather forecasting
- Molecular modelling
- Astrophysics

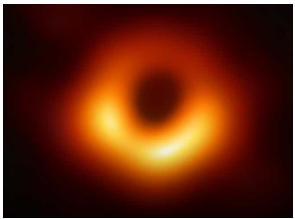
Why using HPC?

Pros:

- Reduce computational time
- Increase CPU capacity
  - Model resolution
  - Number of scenarios
- Increase memory and/or storage
  - Aggregate larger data
  - Solve bigger problems
- Can tackle otherwise unfeasible projects

Cons:

- A relatively steep learning curve
- Largely DIY
- Shared systems $\rightarrow$ Planning needed



Getting started - SSH

Secure SHell Protocol

- cryptographic network protocol for operating network services securely over an unsecured network

Windows:

- MobaXterm
- PuTTY

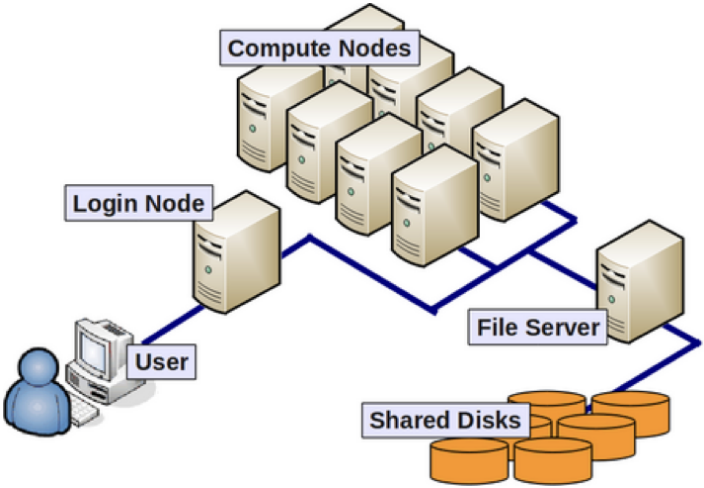
MacOS/Linux

- Terminal 

Getting started - login

- Clusters can be accessed over the internet / local network
- `ssh username@hostname`
- Other commonly used syntax: `ssh -Y ...` or `ssh -X ...`
- `man ssh` (MacOS/Linux) for many more options

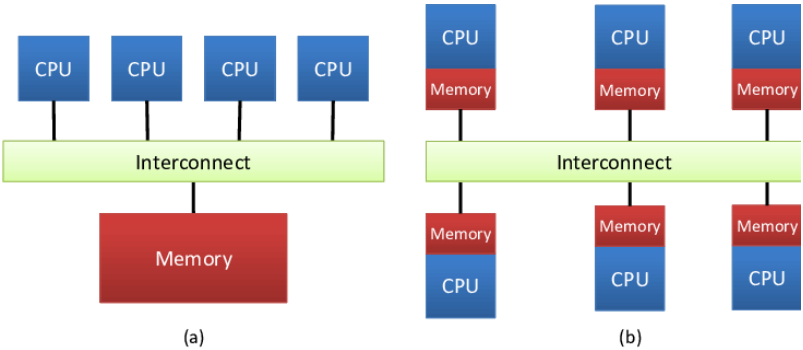
Getting started - basic environment



Shell scripting

- This is the most widely used interface in HPC
- For the most part one gets by with basic commands
  - `ls`, `cd`, `mv`, `cp`, `pwd`
- Shell scripting intros/tutorials
  - shellscript.sh
  - The Unix shell
  - Intro to the Linux shell (account needed)

Shared vs. distributed memory



Job schedulling

- Login nodes are not designed to carry out large computations
- As a shared resource, fair use is encouraged / enforced
- Multiple schedulling tools available:
  - `Slurm`
  - `PBS/Torque`
  - `SGE`
  - `LSF`
  - Different syntax similar purpose (rosetta stone of workload managers)

Job schedulling - in a nutshell

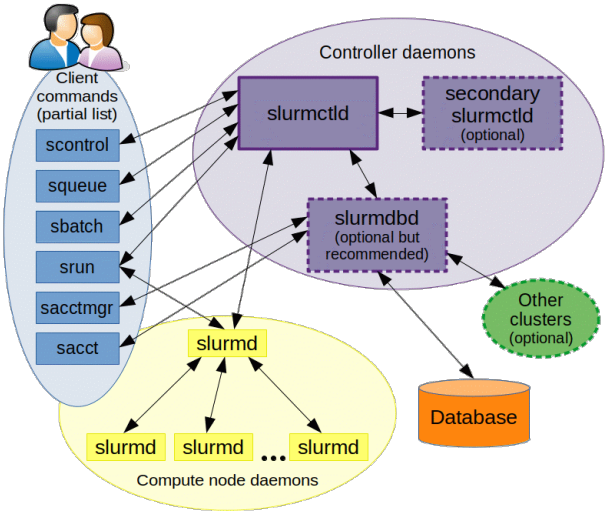
- A tool that tells the HPC system what to run, for how long, and how much resources it will need - At all times, for all users

- A job scheduler performs the following tasks
  - Assigns users to compute nodes
  - Provides a framework for initiating, performing, and monitoring work on the assigned nodes
  - Manages the queue of pending work and determines which job will be assigned to the node next

Job schedulling - Slurm

Basic commands:

- `sbatch`
- `squeue`
- `scancel`
- `srun`
- `salloc`



Slurm documentation

Job schedulling - a simple example (1)

The following lines are part of a file called `firstjob.sh`

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --time=1:00:00
#SBATCH --job-name=Myfirstjob
#SBATCH --output=job_output_%j.out

cd $SLURM_SUBMIT_DIR

# ... do something in this job using 80 cores ...  #
```

Submit the job as

```
login-node:$ sbatch firstjob.sh
```

Job schedulling - a simple example (2)

Alternatively, we can do the following:

File `secondjob.sh`

```
#!/bin/bash

cd $SLURM_SUBMIT_DIR

# ... do something in this job using 80 cores ...  #
```

Submit the job as

```
login-node:$ sbatch -N 2 -ntasks-per-node=40 -t 1:00:00 \
                    -J Mysecondjob -o job_output_%j.out secondjob.sh
```

Job schedulling - a simple example (3)

If successful, the system will show something like:

```
login-node:$ sbatch ... secondjob.sh
Submitted batch job 34987
```

where 34987 is the job id.

(Or give an error otherwise)

Common errors that may make the job submission fail:

- The bash script is not executable (solution: `chmod 700 firstjob.sh`)
- The script is launched from a read-only path (solution `cd /writable/path`)
- Requesting resources, e.g., wall time, memory, cores

Job schedulling - partitions

Some systems configure different ‘queues’ or ‘partitions’

These are set with option `-p` or `--partition`

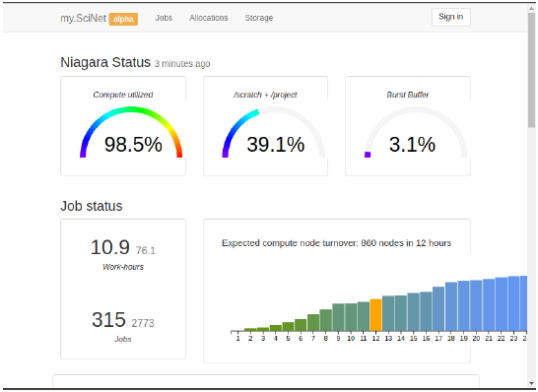
- walltime
- memory available
- number of cores per node

Job monitoring

- `squeue` to show the job queue (`squeue --me` for just your jobs)
- `squeue -j JOBID` information on a specific job (alternatively, `scontrol show job JOBID`, which is more verbose)
- `squeue --start -j JOBID` estimate for when a job will run
- `scancel -i JOBID` cancel the job
- `scancel -u USERID` cancel all your jobs
- `sinfo -p partition` look at available nodes in `partition`
- `sacct` info on your recent jobs

Job monitoring - my.SciNet

https://my.scinet.utoronto.ca

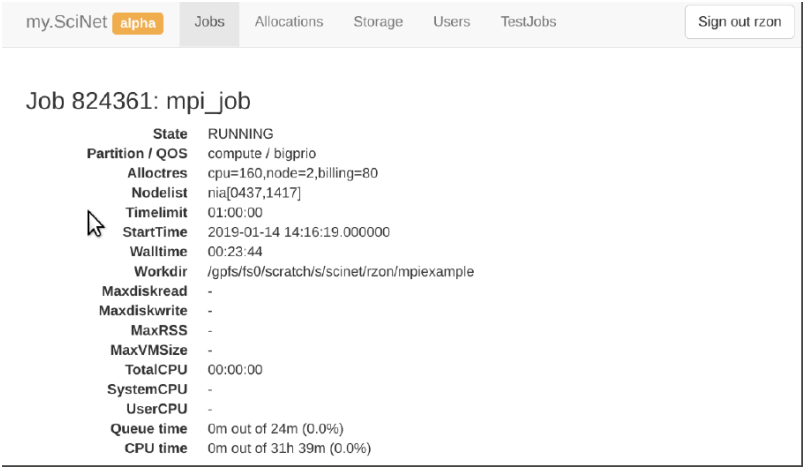


Features

- Niagara cpu and storage utilization
- Status of the login nodes
- Job history
- Per job:
  - jobscript
  - environment
  - wall time
  - *memory usage*
  - *CPU usage*
  - *GFlops/s*
  - *disk I/O*

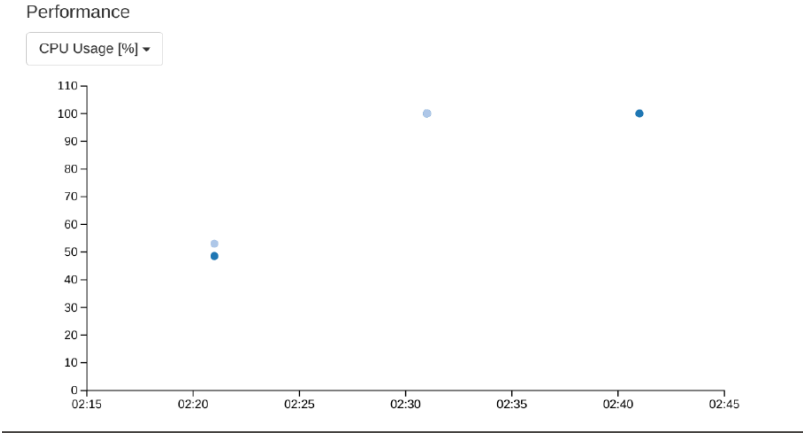
Job monitoring - my.SciNet

https://my.scinet.utoronto.ca



Job monitoring - my.SciNet

https://my.scinet.utoronto.ca



Software in HPC

- Given the wide range of users and applications different software is needed
- Robust software management is necessary
- Tools like `lmod` come to the rescue

Software in HPC - module loading

- Usually, multiple software is installed, all is made available via `module` commands
- `module` sets appropriate environment variables (`PATH`, etc.)
- `module` takes care of conflicting versions of a given software to be available
- relevant for reproducibility purposes

Software in HPC - useful commands

- `module load <module-name>` use particular software
- `module purge` remove currently loaded modules
- `module spider` (or `module spider <module-name>`) list available software packages
- `module avail` list loadable software packages that require no other modules to be loaded first
- `module list` list loaded modules

Software in HPC - available software (SciNet's Niagara)

```
login-node:$ module spider
-------------------------------------------------------------------------
The following is a list of the modules and extensions
currently available:
-------------------------------------------------------------------------
CCEnv: CCEnv
  Compute Canada software modules. Must be loaded to see
  Compute Canada modules in 'module spider'.

NiaEnv: NiaEnv/2018a, NiaEnv/2019b
  Software modules for Niagara. Must be loaded to see
  Niagara modules in 'module spider' (loaded by default).

antlr: antlr/2.7.7
  ANTLR, ANother Tool for Language Recognition, ...

...
```

Software in HPC - software loading details (1)

```
login-node:$ module spider r
----------------------------------------------------------------------
  r:
----------------------------------------------------------------------
Description:
  R is a language and environment for statistical computing
  and graphics
Versions:
        r/3.5.3
        ...
        r/4.1.2
Other possible modules matches:
    .singularity  antlr  arm-forge  ...
----------------------------------------------------------------------
To find other possible module matches execute:
      $ module -r spider '.*r.*'
----------------------------------------------------------------------
For detailed information about a specific "r" package
(including how to load the modules) use the module full name ...

For example:
     $ module spider r/4.1.2
----------------------------------------------------------------------
```

Software in HPC - software loading details (2)

```
login-node:$ module load r/4.1.2

Lmod has detected the following error:

These module(s) or extension(s) exist but cannot be
loaded as requested: "r/4.1.2"

   Try: "module spider r/4.1.2" to see how to load the module(s).
```

Software in HPC - software loading details (3)

```
login-node:$ module spider r/4.1.2
----------------------------------------------------------------------
  r: r/4.1.2
----------------------------------------------------------------------
Description:
  R is a language and environment for statistical computing
  and graphics

  You will need to load all module(s) on any one of the lines
  below before the "r/4.1.2" module is available to load.

      gcc/8.3.0
      intel/2019u4

  Help:
    R is 'GNU S', a freely available language and environment
    for statistical computing and graphics which provides a wide
    variety of statistical and graphical techniques: linear and
    nonlinear modelling, statistical tests, time series analysis,
    classification, clustering, etc. Please consult the R project
    homepage for further information.

    Homepage: https://www.r-project.org
```

Software in HPC - software loading details (4)

```
login-node:$ module load intel/2019u4 r/4.1.2
login-node:$ R

R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Installing libraries/packages

- For the most part library/package installation works as in any workstation
  - i.e., `install.packages(...)` in `R`
- There are some exceptions:
  - cluster has no internet access (usually, only `login` nodes can reach the www)
  - installation requires special dependencies

Installing libraries/packages - special dependencies

One example: install `data.table` package with OpenMP support

```
login-node:$ ml NiaEnv/2019b ml gcc/8.3.0 openmpi/3.1.3 r/4.1.2
login-node:$ R
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
....

> install.packages("data.table",
  configure.args = c(paste0("--with-data.table-include=",
  Sys.getenv("SCINET_OPENMPI_ROOT"),"/include"),
  paste0("--with-data.table-libpath=",
  Sys.getenv("SCINET_OPENMPI_ROOT"),"/lib")))
...
* DONE (data.table)

> library(data.table)
data.table 1.14.2 using 20 threads (see ?getDTthreads).
Latest news: r-datatable.com
```

Data creation

We are generating data at unprecedented rates!

General guidelines for (efficient) data creation in HPC:

- File I/O (Input/Output) is slow! Avoid it as much as you can!
  (CPU operations $\approx$ 1 ns, disk access times $\approx$ 5 ms.)
- Do not create lots of little files! They are an inefficient use of space and time (slow to create)
- Instead, save your data in big files which contain all the information you need
- Do not have multiple processes writing to files in the same directory (unless you're using parallel I/O)
- Write data out in binary. Faster and takes less space

Data management

How to organize files on disk?

- Human-interpretable filenames lose their charm after a few dozen files (or after a few months pass)
- Don't use filenames to store run information
- Avoid using a flat directory structure (i.e., no sub-directories). Organize your data in a sensible directory tree
- If you're doing many runs with many varied parameters, consider using a database to store the filenames of your runs, with associated run metadata
- Rigorously maintained meta-data (data about the data) is essential
- Back up your data, especially your metadata or database

Take-home messages

- HPC can greatly accelerate your work
- Understanding how to operate HPC systems is crucial
- Make full and efficient use of available resources

Acknowledgements and additional resources

- Some slides thanks to the SciNet team:
  - Introduction to SciNet, Niagara & Mist by Mike Nolta (Dec 2021)
  - Storage and I/O in Large Scale Scientific Projects by Ramses van Zon and Marcelo Ponce (Sep 2016)
- Introduction to Shell and Cluster computing
- Niagara Quickstart
- Education @ SciNet (amply recommended!)

Other clusters

- Compute Canada
- CCM @ Sickkids
- h4h @ UHN
- Galen @ Mount Sinai
- HPC4Health

Related topics of interest (R)

- Intro to parallel computing in R
- A guide to parallelism in R
- Scalable data analysis in R
- `foreach` package vignette

Thanks for listening!