Hyak Training Session

October 25, 2019

Andrew Wildman

Outline

I. **Hyak overview**
   A. Hyak architecture
   B. Logging on to Hyak

II. **Navigating Hyak**
   A. Basic shell commands
   B. Transferring files
   C. Important locations

III. **Loading software**
   A. Modules
   B. Containers

IV. **Slurm**
   A. Commands
   B. Examples

V. **Other resources**
   A. Topics outside of this tutorial
   B. Places to get help

Hyak overview

Hyak is a “condominium” supercomputing cluster:

- Groups own nodes in partitions.
- In addition to our partitions, groups have access to the `build` and `ckpt` partitions
- ~10,000 cores in total

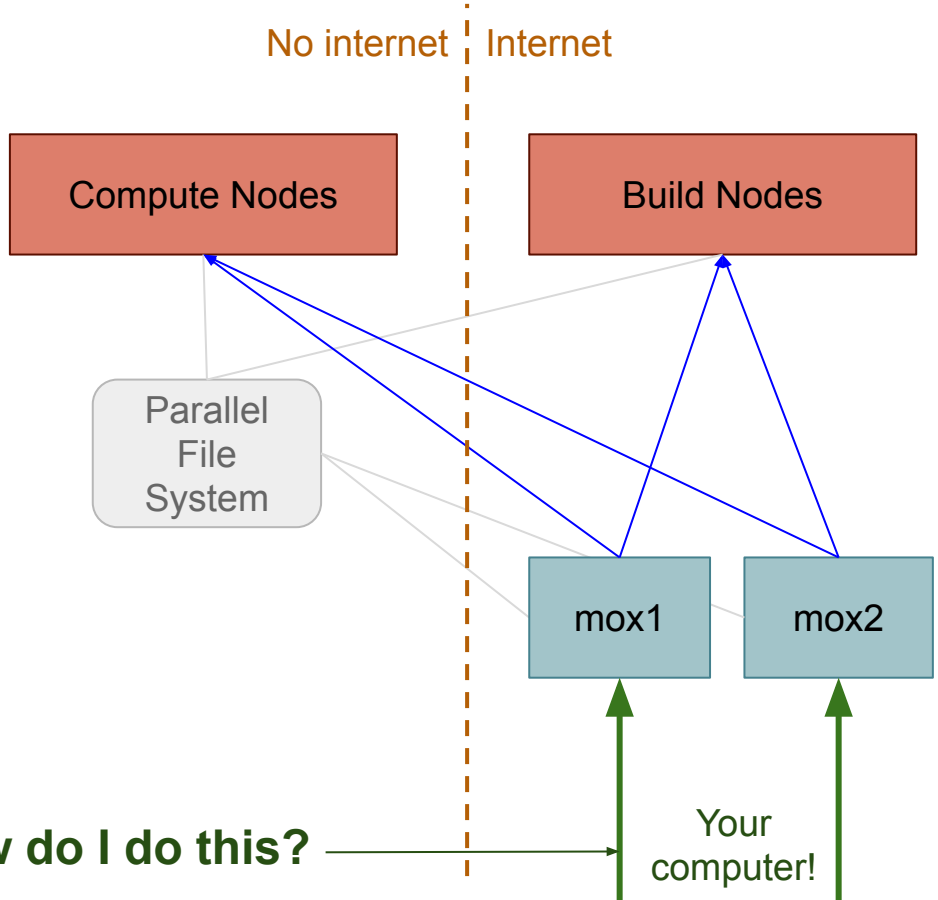
Two clusters: ikt (**retiring**) and mox (current)

Mox nodes:

- 28 cores
- 128 GB RAM
- 92 regular nodes - `stf` partition
- 2 interactive regular nodes - `stf-int` partition
- 9 GPU nodes - `stf-gpu` partition
- 1 interactive GPU node - `stf-int-gpu` partition

Hyak architecture

- All nodes share the same filesystem (except `/tmp`)
- Login nodes
  - Transfer files
  - Submit jobs
  - **NOT** for heavy processing
- Compute nodes
  - High performance
  - Interactive or batch
- Build nodes
  - Can't take a whole node
  - Access internet
  - Just for compiling software



Logging on to Hyak

Mac/Linux:

ssh:

```
ssh <uwnetid>@mox.hyak.uw.edu
```

Windows:

- PuTTY
- GitBash
- Windows Subsytem for Linux
- WinSCP (just for transfering files)
- cmder
- … and probably more!

Browser (still in beta):

http://ondemand.hyak.uw.edu/

- Need to be on campus (or Husky OnNet VPN)
- Terminal emulator
- Interactive apps
  - Jupyter notebooks
  - R Studio
  - Xfce for GUI programs

I’m in

… Now what?

Basic shell (bash) commands

File system manipulation:

- `ls`
  - “List” files in current directory (folder)
- `cd`
  - “Change directory”
- `pwd`
  - “Print working (current) directory”
- `mkdir`
  - “Make directory”
- `mv`
  - Move (rename) file or directory
- `cp`
  - “Copy” files and/or directories (-r)
- `rm`
  - “Remove” files and/or directories (-r)

File editing and compression:

- `nano`
  - Edit files
  - Other editors: vim, emacs, etc.
- `tar`
  - Compress for a “tape archive”
- `zip` (and `unzip`)
  - Compress via zip algorithm
  - Windows friendly

Many, many more

- **`man`**
- `find`
- `top`
- `kill`
- `chmod`
- `curl`
- `grep`
- `sed`

`...`

Transferring files

- Mac/Linux
  - To: `scp <path/to/file> <username>@mox.hyak.uw.edu:<path/to/dest>`
  - From: `scp <username>@mox.hyak.uw.edu:<path/to/file> <path/to/dest>`
- Windows
  - WinSCP: https://winscp.net/eng/index.php

Lolo:

- Magnetic tape archive (lolo archive)
- For long term storage - **only store compressed large files!**
- STF location: `/archive/hyak/stf`
- Transfer files the same as between local and Hyak

Important locations on Hyak

- `/gscratch/stf`
  - Main work location for stf users
  - **Any files untouched for >30 days will be scrubbed!!**
- `/usr/lusers/<username>`
  - Home directory
  - **Only 10 GB of storage per user**
- `/tmp`
  - Node local storage
- `/sw`
  - All software installs
- `/sw/contrib`
  - User installed software
- `/sw/modules-1.775/modulefiles/contrib`
  - User added modulefiles

Loading software: the `modules` system

- `module avail`
  - Show all available modules
- `module load <module>` | `module unload <module>`
  - (Un)load a given module
  - Must be full name up to non-ambiguity (e.g. `contrib/git` or `contrib/git/2.19.1`)
- `module list`
  - List loaded modules
- `module purge`
  - Unload all modules
- `module help`
  - Print help with commands

Advanced user's note:

The modules system works by keeping track of and modifying environment variables (e.g. PATH, LD_LIBRARY_PATH, CPATH, etc.)

https://modules.readthedocs.io/en/latest/

Loading software: Singularity containers

Hyak's containerization system is Singularity (**not Docker**)

https://sylabs.io/guides/3.4/user-guide/

- Can create singularity images from docker images
- Permissions are same as for user (not root)
- Containers are "high performance"
- Same benefits as other container tools (reproducibility, mobility, etc.)

Slurm: Hyak's job scheduler

https://slurm.schedmd.com/

- Ensures fair share between users
- Run interactive or batch jobs
- Allows for running on the `ckpt` queue

Advanced users:

- Manages cluster locality for multi-node jobs
- Handles MPI and other communication protocols through `srun`

Slurm: Commands

- `srun`
  - Submits job for interactive use or initiate job steps inside batch script
- `sbatch <script>`
  - Submits a script for non-interactive use
- `squeue`
  - Get status of jobs in batch queue
- `scancel <jobid>`
  - Cancels an unfinished job

You'll typically use these most often

- `sinfo`
  - Get state of partitions and nodes
- `sacct`
  - Gets accounting information about active and completed jobs

Slurm docs and `man <slurm-command>` are very useful!

Slurm: Example - Interactive job

```
srun -p stf-int -A stf --ntasks=8 --mem=20G --pty /bin/bash -l
```

- Partition
- Account
- Number of processes (*)
- Amount of RAM
- Command

Be aware of difference between:

- Number of tasks (processes, MPI)
  - `--ntasks (-n)`
  - `--ntasks-per-node`
- Number of cpus per task (threads, OpenMP)
  - `--cpus-per-task (-c)`

https://slurm.schedmd.com/srun.html

`man srun`

Slurm: Example - Batch job

```
sbatch [options] test.sh
```

Examples of options:

```
-p <partition>

-A <account>

-n <ntasks>

-J <job name>

etc…
```

GPU option:

```
--gres=gpu:P100:1
```

All sbatch options have options that can be specified in ***batch script***

Anatomy of a batch script

1. SBATCH directives
   - All options have a directive
   - Directives start with #SBATCH
   - Comments are ## lines
2. Set up environment (modules, containers)
3. Compute!
4. Clean up

Notes:

- Directive order doesn't matter
- Bash scripts don't exit on command failure - always check!

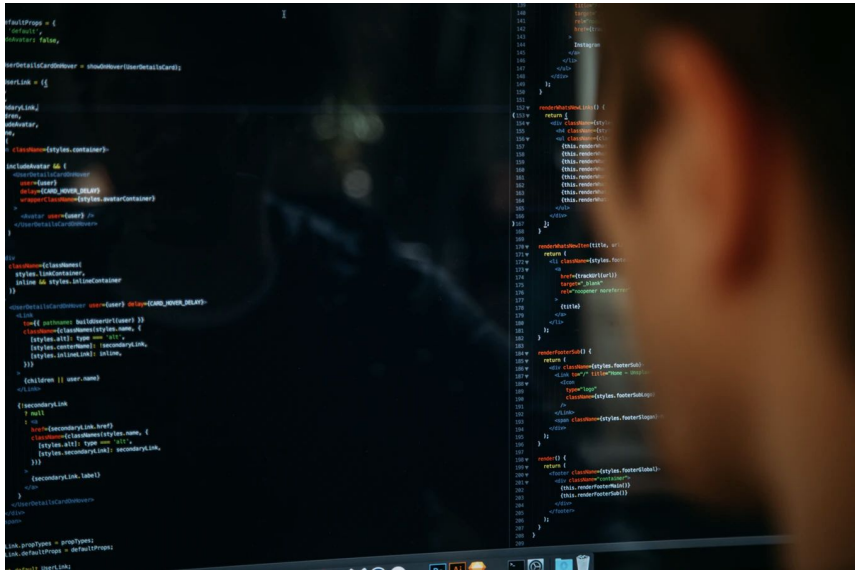
```
#!/bin/bash
## First line has to be this to be a bash script

##
## **** sbatch options *****
##

##    Accounting information
#SBATCH --partition=stf-gpu
#SBATCH --account=stf
#SBATCH --job-name=multi_param_en2_0

##    Location / output options
#SBATCH --verbose
#SBATCH --chdir=/gscratch/stf/<USERNAME>/path/to/directory

##    Requested resources
#SBATCH --nodes=1
#SBATCH --time=8:00:00
#SBATCH --mem=120G
#SBATCH --gres=gpu:P100:1

##
## **** Running begins here ****
##

## Load modules and environment
module load contrib/cuda-9.0_cudnn-7.0.5
module load anaconda3_4.3.1
source activate anaconda_environment || exit 1

## Do work
python gpu_script.py

## Clean up
source deactivate
exit 0
```

Checkpoint queue - special case

The checkpoint queue allows any user to run on other groups' unused nodes!

- Partition: `ckpt`
- Account: `<group>-ckpt` (`stf-ckpt`)
- Jobs can be interrupted at any time
- Jobs *will* be interrupted after 4 contiguous hours
- Jobs will be resubmitted after interruption (if under total requested time limit)
- **Your code must be checkpointed to take advantage of this**
  - Save a binary file containing state of some objects, restart from logs, etc.
  - Your script should also account for any checkpointing that is done

Topics not covered

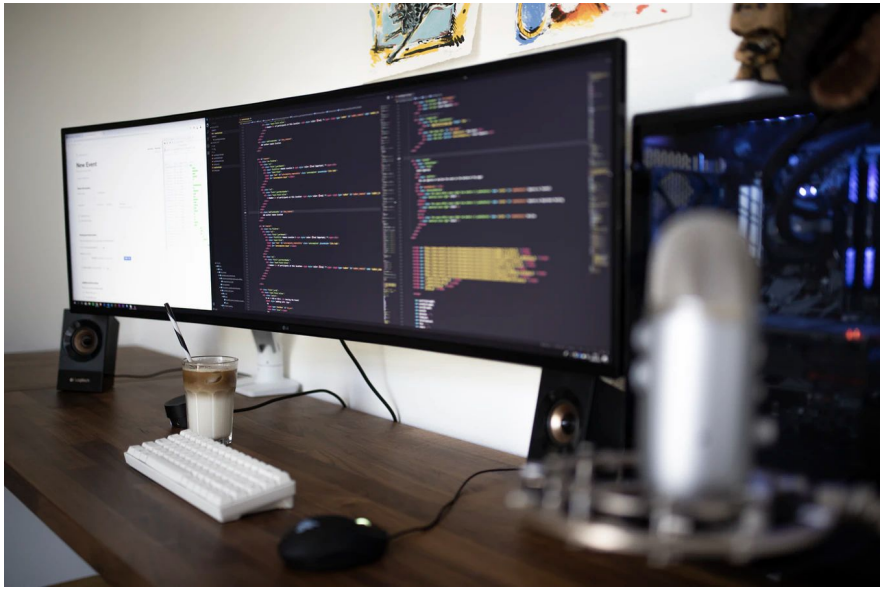


- How to install new software
  - Installation and build systems
  - Writing your own modulefiles
- How to parallelize your code
  - Different paradigms for parallelism
  - Data locality and contiguity
  - GPU parallelization
  - Parallel patterns
- Anything with the cloud (This is the Hyak training session!)
- General architecture of HPC systems
  - Interconnectivity and node locality
  - Physical infrastructure

Where to get help

- Documentation (`man` or webpages)
- Hyak wiki: https://wiki.cac.washington.edu/display/hyakusers/WIKI+for+Hyak+users
- Slack channel: https://uw-rcc.slack.com/
- Website: https://depts.washington.edu/uwrcc/
- Emails: hpcc@uw.edu or uwrcc@uw.edu
- Office hours: Alternating Tuesdays and Fridays from 1-3 pm

Happy computing!



Endorse STF Proposal: https://uwstf.org/proposals/2020/21