Getting started on the FASRC Cannon Cluster
Learning Objectives

▪ Describe the structure of a compute cluster
▪ Log in to Cannon
▪ Demonstrate how to start an interactive session and a batch job with the SLURM job scheduler
▪ Query job metadata
▪ Cluster storage
▪ Cluster software modules
▪ VDI - Open OnDemand
Cluster Architecture
Cluster Terminology

- **Supercomputer/High Performance Computing (HPC) cluster**: A collection of similar computers connected by a high speed interconnect that can act in concert with each other.
- **Node**: A computer in the cluster, an individual motherboard with CPU, memory, local hard drive.
- **CPU**: Central Processing Unit, it can contain multiple computational cores (processors).
- **Core**: Basic unit of compute that runs a single instruction of code (a single process).
- **GPGPU/GPU**: General Purpose Graphics Processing Unit, a GPU designed for supercomputing.
Login & Access

https://docs.rc.fas.harvard.edu/kb/quickstart-guide/

Cluster Quick Start Guide

This guide will provide you with the basic information needed to get up and running on the FASRC cluster for simple command line access. If you’d like more detailed information, each section has a link to fuller documentation

PREREQUISITES

1. Get a FASRC account using the account request tool.

Before you can access the cluster you need to request a Research Computing account.

See How Do I Get a Research Computing Account for instructions if you do not yet have an account.

See the account confirmation email for instructions on setting your password and getting started.
Login & Access

Once you have an account you can use the Terminal to connect to Cannon

- Mac: Terminal
- Linux: Xterm or Terminal
- Windows: SSH client - Putty or Bash Emulator - Git Bash

$ ssh username@login.rc.fas.harvard.edu

- ssh stands for Secure SHell
- ssh is a protocol for data transfer that is secure, i.e the data is encrypted as it travels between your computer and the cluster (remote computer)
- Commonly used commands that use the ssh protocol for data transfer are, scp and sftp
Once you have an account you can use the Terminal to connect to Cannon

- Mac: Terminal
- Linux: Xterm or Terminal
- Windows: SSH client - Putty or Bash Emulator - Git Bash

$ ssh username@login.rc.fas.harvard.edu

Cannon
Login issues? See https://rc.fas.harvard.edu/resources/support/

Password:
Verification code:
Once you have run the ssh command:

- Enter your password *(cursor won’t move!)*
- Add a verification code *(2-Factor Authentication)*

2. Setup OpenAuth for two factor authentication

Once you have your new FASRC account, you will need to set up our OpenAuth tool for two-factor authentication.

See the OpenAuth Guide for instructions if you have not yet set up OpenAuth.

For troubleshooting issues you might have, please see our troubleshooting page.

OpenAuth is 2-factor authentication separate from HarvardKey and updates the token every 30 seconds.
You have logged into the login node!

[joesmith@holylogin03 ~]$ 

Name of the login node assigned to you
Access to resources on a compute node

• Login node:
  – not designed for analysis
  – not anything compute- or memory-intensive
  – best practice is to request a compute node as soon as you log in

• Interactive session:
  – work on a compute node “interactively”
  – request resources from SLURM using the `salloc` command
  – session will only last as long as the remote connection is active
Access to resources on a compute node

**Simple Linux Utility for Resource Management - SLURM job scheduler:**

- Fairly allocates access to resources to users on compute nodes
- Manages a queue of pending jobs; ensures that no single user or group monopolizes the cluster
- Ensures users do not exceed their resource requests
- Provides a framework for starting, executing, and monitoring batch jobs
Access to resources on a compute node

Requesting an interactive session:

```
joesmith@holylogin03 ~]$ salloc -p test --mem 100 -t 0-01:00
```

- `salloc` - is how interactive sessions are started with SLURM
- `-p test` - requesting a compute node in a specific partition*
- `--mem 100` - memory requested in MB
- `-t 0-1:00` - time requested (1 hour)

```
joesmith@holy7c26602 ~]$  
```

* Partitions are groups of computers that are designated to perform specific types of computing. More on next slide

Name of the compute node assigned to you
#!/bin/bash

#SBATCH -J Rjob1   # Job name
#SBATCH -p shared  # Partition(s) (separate with commas if using multiple)
#SBATCH -n 1       # Number of cores
#SBATCH -t 0-00:30:00 # Time (D-HH:MM:SS)
#SBATCH --mem=500M  # Memory
#SBATCH -o %j.o    # Name of standard output file
#SBATCH -e %j.e    # Name of standard error file

## LOAD SOFTWARE ENV ##
module load R/3.5.1-fasrc01

input=M2.R

## EXECUTE CODE ##
R CMD BATCH $input $input.out

More information: https://docs.rc.fas.harvard.edu/kb/running-jobs/
ALWAYS test the job submission script first:

- To ensure the job will complete without any errors
- To ensure you understand the resource needs and have requested them appropriately

Submitting a batch job:

[joesmith@boslogin01 ~]$ sbatch runscript.sh
Submitted batch job 20801712
[joesmith@boslogin01 ~]$
# Partitions on Cannon

<table>
<thead>
<tr>
<th>Partitions:</th>
<th>shared</th>
<th>gpu</th>
<th>test</th>
<th>gpu_test</th>
<th>serial_requeue</th>
<th>gpu_requeue</th>
<th>bigmem</th>
<th>unrestricted</th>
<th>pi_lab</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Limit</strong></td>
<td>7 days</td>
<td>7 days</td>
<td>8 hrs</td>
<td>1 hrs</td>
<td>7 days</td>
<td>7 days</td>
<td>no limit</td>
<td>no limit</td>
<td>varies</td>
</tr>
<tr>
<td><strong># Nodes</strong></td>
<td>530</td>
<td>15</td>
<td>16</td>
<td>1</td>
<td>1930</td>
<td>155</td>
<td>6</td>
<td>8</td>
<td>varies</td>
</tr>
<tr>
<td><strong># Cores / Node</strong></td>
<td>48</td>
<td>32 + 4 V100</td>
<td>48</td>
<td>32 + 4 V100</td>
<td>varies</td>
<td>varies</td>
<td>64</td>
<td>64</td>
<td>varies</td>
</tr>
<tr>
<td><strong>Memory / Node (GB)</strong></td>
<td>196</td>
<td>375</td>
<td>196</td>
<td>375</td>
<td>varies</td>
<td>varies</td>
<td>512</td>
<td>256</td>
<td>varies</td>
</tr>
</tbody>
</table>

Learn more about a partition:

```
$ sinfo -p shared
$ scontrol show partition shared
```
**sacct overview**

- **sacct** = Slurm accounting database
  - every 30 sec the node collects the amount of cpu and memory usage that all of the process ID are using for the given job. After the job ends this data is set to slurmd.

- **Common flags**
  - `-j jobid` or `--name=jobname`
  - `-S YYYY-MM-DD` and `-E YYYY-MM-DD`
  - `-o output_options`

```
JobID,JobName,NCPUS,Nnodes,Submit,Start,End,CPUTime,TotalCPU,ReqMem,MaxRSS,MaxVMSize,State,Exit,Node
```
Memory Usage

Run a test batch job and check memory usage after the job has completed (with the “sacct” Slurm command)

Example:

[joesmith@boslogin01 ~]$ sacct -j 3937435 -o ReqMem,MaxRSS

<table>
<thead>
<tr>
<th>ReqMem</th>
<th>MaxRSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000Mn</td>
<td>286712K</td>
</tr>
</tbody>
</table>

or

286712KB = 286.712MB
seff overview

[user@boslogin01 home]# seff 1234567
Job ID: 1234567
Cluster: odyssey
User/Group: user/user_lab
State: COMPLETED (exit code 0)
Nodes: 8
Cores per node: 64
CPU Utilized: 37-06:17:33
CPU Efficiency: 23.94% of 155-16:02:08 core-walltime
Job Wall-clock time: 07:17:49
Memory Utilized: 1.53 TB (estimated maximum)
Memory Efficiency: 100.03% of 1.53 TB (195.31 GB/node)
A Fairshare score

- determines what priority a user/group has to run their jobs
- is calculated for a group using various factors, including what resources/partition of the cluster groups have access.
- goes from 1 to 0 with a middle point of 0.5

More information: https://docs.rc.fas.harvard.edu/kb/fairshare/
A Fairshare score

- determines what priority a user/group has to run their jobs
- is calculated for a group using various factors, including what resources/partition of the cluster groups have access.
- goes from 1 to 0 with a middle point of 0.5

- 1.0: Unused. The account has not run any jobs recently.
- 1.0 > f > 0.5: Under-utilization. The account is underutilizing their granted Share.
- 0.5: Average utilization. The account on average is using exactly as much as their granted Share.
- 0.5 > f > 0: Over-utilization. The account has overused their granted Share.
- 0: No share left. The account has vastly overused their granted Share.

More information: https://docs.rc.fas.harvard.edu/kb/fairshare/
A Fairshare score

- determines what priority a user/group has to run their jobs
- is calculated for a group using various factors, including what resources/partition of the cluster groups have access.
- goes from 1 to 0 with a middle point of 0.5
- dynamically updated based on usage
- ensures that no single user or group monopolizes the cluster resources

More information: https://docs.rc.fas.harvard.edu/kb/fairshare/
Fairshare score

- Accounts on the cluster are assigned to a primary lab “group” based on their affiliation.

```
[user1@holyitc01 ~]$: groups
test_lab  cluster_users
```
Fairshare score

- Accounts on the cluster are assigned to a primary lab “group” based on their affiliation.

```bash
[user1@holyitc01 ~]$ groups
test_lab  cluster_users
```

- `sshare` can be used to check the current fairshare for a whole group or a single user.

```bash
[user1@holyitc01 ~]$ sshare --account=test_lab -a
```

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>RawShares</th>
<th>NormShares</th>
<th>RawUsage</th>
<th>EffectvUsage</th>
<th>FairShare</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_lab</td>
<td></td>
<td>244</td>
<td>0.001363</td>
<td>45566082</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
<tr>
<td>test_lab</td>
<td>user1</td>
<td>parent</td>
<td>0.001363</td>
<td>8202875</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
<tr>
<td>test_lab</td>
<td>user2</td>
<td>parent</td>
<td>0.001363</td>
<td>248820</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
<tr>
<td>test_lab</td>
<td>user3</td>
<td>parent</td>
<td>0.001363</td>
<td>163318</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
<tr>
<td>test_lab</td>
<td>user4</td>
<td>parent</td>
<td>0.001363</td>
<td>18901027</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
<tr>
<td>test_lab</td>
<td>user5</td>
<td>parent</td>
<td>0.001363</td>
<td>18050039</td>
<td>0.000572</td>
<td>0.747627</td>
</tr>
</tbody>
</table>

More information: https://docs.rc.fas.harvard.edu/kb/fairshare/
### Storage Grid

<table>
<thead>
<tr>
<th></th>
<th>Home Directories</th>
<th>Lab Directory (Startup)</th>
<th>Local Scratch</th>
<th>Global Scratch</th>
<th>Tier Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mount Point</strong></td>
<td>/n/home#$USER</td>
<td>/n/holylabs/pi_lab</td>
<td>/scratch</td>
<td>/n/$SCRATCH</td>
<td>/n/pi_lab</td>
</tr>
<tr>
<td><strong>Size Limit</strong></td>
<td>100GB</td>
<td>1- 4TB</td>
<td>70GB/node</td>
<td>2.4PB total</td>
<td>Based on Tier</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>All cluster nodes + Desktop/laptop</td>
<td>All cluster nodes</td>
<td>Local compute node only</td>
<td>All cluster nodes</td>
<td>All cluster nodes/ mountable</td>
</tr>
<tr>
<td><strong>Retention Policy</strong></td>
<td>Indefinite</td>
<td>Indefinite</td>
<td>Job duration</td>
<td>90 days</td>
<td>Indefinite</td>
</tr>
<tr>
<td><strong>Backup</strong></td>
<td>Hourly snapshot + Daily Offsite</td>
<td>No backup</td>
<td>No backup</td>
<td>No backup</td>
<td>Depending on Tier</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Moderate. Not suitable for high I/O</td>
<td>Moderate. Not suitable for high I/O</td>
<td>Suited for small file I/O intensive jobs</td>
<td>Appropriate for large file I/O intensive jobs</td>
<td>Depending on Tier</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Free</td>
<td>Free max of 4TB</td>
<td>Free</td>
<td>Free</td>
<td>Paid</td>
</tr>
</tbody>
</table>

Tier Storage: https://www.rc.fas.harvard.edu/services/data-storage/
Software is loaded incrementally using modules, to set up your shell environment (e.g., PATH, LD_LIBRARY_PATH, and other environment variables)

Module loads best placed in SLURM batch scripts:

- Keeps your interactive working environment simple
- Is a record of your research workflow (reproducible research!)
- Keep .bashrc module loads sparse, lest you run into software and library conflicts

Software search capabilities similar to module-query are also available on the RC Portal: https://portal.rc.fas.harvard.edu/apps/modules

module load matlab/R2016a-fasrc01  # recommended
module load matlab                 # most recent version
module list                       # show loaded modules
module purge                      # unload all modules
VDI - Open OnDemand

For applications that need a GUI: https://vdi.rc.fas.harvard.edu

Supports:
- Remote Desktop
- Jupyter Notebooks
- Rstudio
- Matlab

Notes:
- Need to be on the RC VPN to use
- Sessions are submitted as jobs on the cluster and thus use fairshare but also can run on any partition
Request Help - Resources

- **https://docs.rc.fas.harvard.edu/kb/support/**
  - Documentation
    - https://docs.rc.fas.harvard.edu/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@rc.fas.harvard.edu
  - Office Hours
    - Wednesday noon-3pm https://harvard.zoom.us/j/255102481
  - Consulting Calendar
    - https://www.rc.fas.harvard.edu/consulting-calendar/
  - Training
    - https://www.rc.fas.harvard.edu/upcoming-training/
The FAS Secure Environment (FASSE) is a secure multi-tenant cluster environment to provide Harvard researchers access to a secure enclave for analysis of sensitive datasets with DUA’s and IRB’s classified as Level 3.

https://policy.security.harvard.edu/
https://docs.rc.fas.harvard.edu/kb/data-use-agreements/

https://security.harvard.edu/
https://docs.rc.fas.harvard.edu/kb/fasse/