Introduction to Cluster Computing
Intro Objectives

- What is RC?
- Do you speak Supercomputer?
- What resources are available?
- How do I access resources?
- How do I submit calculations?
- Am I using the resources effectively?
- What could go wrong?
- How do I get help?
Research Computing

Faculty of Arts and Sciences (FAS) department that handles non-enterprise IT requests from researchers.

**RC Primary Services:**
- Odyssey Supercomputing Environment
- Lab Storage
- Instrument Computing Support
- Hosted Machines (virtual or physical)

**RC Staff:**
- 20 staff with backgrounds ranging from systems administration to development-operations to Ph.D. research scientists.
- Supporting 600 research groups and 3000+ users across FAS, SEAS, HSPH, HBS, GSE.
- For Bioinformatics researchers the Harvard Informatics group is closely tied to RC and is there to support the specific problems for that domain.
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.
- **Server, Node, Blade, Box, Machine**: An individual motherboard with CPU, memory, network, and local hard drive.
- **CPU (Socket)**: Central Processing Unit, a single silicon die that can contain multiple computational cores.
- **Core**: Basic unit of compute that runs a single instruction of code.
- **GPGPU/GPU**: General Purpose Graphics Processing Unit, a GPU designed for supercomputing.
- **InfiniBand (IB)**: A near zero latency high bandwidth interconnect used in Supercomputing.
- **Serial**: Doing tasks/instructions in sequence on a single core.
- **Parallel**: Doing tasks/instructions on multiple cores simultaneously.
- **I/O**: Input/Output, a general term for reading and writing files to/from storage whether local or remote.
Cluster Basics
Odyssey Components

**Compute:**
- 78,000+ compute cores
- Cores/node: 8 to 64
- Memory/node: 12GB to 512GB (4GB/core)
- 1,000,000+ NVIDIA GPU cores

**Software:**
- Operating System CentOS 7
- Slurm job manager
- 1,000+ scientific tools and programs
  - [https://portal.rc.fas.harvard.edu/apps/modules](https://portal.rc.fas.harvard.edu/apps/modules)

**Interconnect:**
- 2 underlying networks connecting 3 data centers
- TCP/IP network
- Low-latency 56 GB/s InfiniBand network:
  - inter-node parallel computing
  - fast access to Lustre mounted storage
# Storage Grid

<table>
<thead>
<tr>
<th></th>
<th>Home Directories</th>
<th>Lab Storage</th>
<th>Local Scratch</th>
<th>Global Scratch</th>
<th>Persistent Research Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mount Point</strong></td>
<td>/n/home#$USER</td>
<td>/n/pi_lab</td>
<td>/scratch</td>
<td>/n/scratchlfs</td>
<td>/n/holylfs</td>
</tr>
<tr>
<td><strong>Size Limit</strong></td>
<td>100GB</td>
<td>4TB+</td>
<td>270GB/node</td>
<td>2.4PB total</td>
<td>3PB</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>All cluster nodes + Desktop/laptop</td>
<td>All cluster nodes + Desktop/laptop</td>
<td>Local compute node only.</td>
<td>All cluster nodes</td>
<td>Only IB connected cluster nodes</td>
</tr>
<tr>
<td><strong>Backup</strong></td>
<td>Hourly snapshot + Daily Offsite</td>
<td>Daily Offsite</td>
<td>No backup</td>
<td>No backup</td>
<td>External Repos No backup</td>
</tr>
<tr>
<td><strong>Retention Policy</strong></td>
<td>Indefinite</td>
<td>Indefinite</td>
<td>Job duration</td>
<td>90 days</td>
<td>3-9 mo</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>Appropriate for large I/O intensive jobs</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Free</td>
<td>4TB Free + Expansion at $50/TB/yr</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
</tr>
</tbody>
</table>
Intro Objectives

▪ What is RC?
▪ Do you speak Supercomputer?
▪ What resources are available?
▪ How do I access resources?
▪ How do I submit calculations?
▪ Am I using the resources effectively?
▪ What could go wrong?
▪ How do I get help?
Documentation: www.rc.fas.harvard.edu

Here you will find all our user documentation.

Of particular interest:

• Access and Login:
  [https://www.rc.fas.harvard.edu/resources/access-and-login/](https://www.rc.fas.harvard.edu/resources/access-and-login/)

• Running Jobs:
  [https://www.rc.fas.harvard.edu/resources/running-jobs/](https://www.rc.fas.harvard.edu/resources/running-jobs/)

• Software modules available:
  [https://portal.rc.fas.harvard.edu/apps/modules](https://portal.rc.fas.harvard.edu/apps/modules)

• Odyssey Storage:
  [https://www.rc.fas.harvard.edu/resources/odyssey-storage/](https://www.rc.fas.harvard.edu/resources/odyssey-storage/)

• Interactive Computing Portal
  [https://www.rc.fas.harvard.edu/resources/documentation/virtual-desktop/](https://www.rc.fas.harvard.edu/resources/documentation/virtual-desktop/)

• Singularity Containers:
  [https://www.rc.fas.harvard.edu/resources/documentation/software/singularity-on-odyssey/](https://www.rc.fas.harvard.edu/resources/documentation/software/singularity-on-odyssey/)

• gpu computing
  [https://www.rc.fas.harvard.edu/resources/documentation/gpgpu-computing-on-odyssey/](https://www.rc.fas.harvard.edu/resources/documentation/gpgpu-computing-on-odyssey/)

• How to get help:
  [https://www.rc.fas.harvard.edu/resources/support/](https://www.rc.fas.harvard.edu/resources/support/)
Login & Access

- Terminal application is needed to connect via secure shell (SSH)
  - Mac: Terminal.app on Mac/Linux
  - Linux: Xterm

```
> ssh username@login.rc.fas.harvard.edu
```

Odyssey2
Login issues? See https://rc.fas.harvard.edu/resources/support/

Password:
Verification code:

- Windows: Putty
Verification Code?

- OpenAuth is 2-factor authentication separate from HarvardKey and updates the token every 30 seconds
- Download OpenAuth from: [https://software.rc.fas.harvard.edu/oa/](https://software.rc.fas.harvard.edu/oa/)
- NOTE: OpenAuth token requires that your computer time be correct. If you have problems logging in this is one of the first things you should check.

Access Issues?

- Accounts are locked for 15 minutes after 5 failed login attempts in a row.
- Password Reset: [https://portal.rc.fas.harvard.edu/pwreset/](https://portal.rc.fas.harvard.edu/pwreset/)
Transfer Files

• Secure File Transfer: SFTP Client
  • GUI client FileZilla for all platforms
  • Configure according to http://fasrc.us/configfilezilla to avoid 2FA problems

• command-line from local terminal application
  – scp: secure copy
    
    ```
    scp file1 username@login.rc.fas.harvard.edu:directory2/
    ```

  ```
  scp -r directory1 username@login.rc.fas.harvard.edu:directory2/
  ```

  – rsync: remote sync
    
    ```
    rsync -av --progress directory1/ username@login.rc.fas.harvard.edu:directory2/
    ```
Working with Environment

• In Linux, only the Unix core utilities are in your command-`PATH` by default.

• In Linux, only the default system libraries are in your `LD_LIBRARY_PATH`.

• The **module** system allows users to easily update their working environment, to include specific codes, versions, compilers, and libraries.

• List of installed modules: [https://portal.rc.fas.harvard.edu/apps/modules](https://portal.rc.fas.harvard.edu/apps/modules)

```bash
module load R
module list
Currently Loaded Modules:
  1) R_core/3.2.0-fasrc01   2) R_packages/3.2.0-fasrc01
  3) R/3.2.0-fasrc01
module show R
```
Intro Objectives

▪ What is RC?
▪ Do you speak Supercomputer?
▪ What resources are available?
▪ How do I access resources?
▪ How do I transfer files?
▪ How do I submit calculations?
▪ Am I using the resources effectively?
▪ What could go wrong?
▪ How do I get help?
What is Slurm?

• Simple Linux Utility for Resource Management
  – User tasks (jobs) on the cluster are containerized so that users cannot interfere with other jobs or exceed their resource request (cores, memory, time)

• Basic Slurm commands:
  – sbatch: submit a batch job script
  – srun: submit an interactive test job
  – squeue: contact slurmd for currently running jobs
  – sacct: contact slurmdb for accounting stats after job ends
  – scancel: cancel a job(s)

SLURM Docs: https://slurm.schedmd.com/
## Slurm Scheduler

<table>
<thead>
<tr>
<th>Partitions:</th>
<th>shared</th>
<th>serial_requeue</th>
<th>test</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>no limit</td>
<td>no limit</td>
<td>varies</td>
</tr>
<tr>
<td><strong># Nodes</strong></td>
<td>456</td>
<td>1930</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>varies</td>
</tr>
<tr>
<td><strong># Cores / Node</strong></td>
<td>32</td>
<td>varies</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td>varies</td>
</tr>
<tr>
<td><strong>Memory / Node (GB)</strong></td>
<td>128</td>
<td>varies</td>
<td>128</td>
<td>512</td>
<td>256</td>
<td>varies</td>
</tr>
</tbody>
</table>

### Batch jobs:
#SBATCH -p shared  # Partition name

### Interactive or Test jobs:
srun -p test OTHER_OPTIONS
Slurm Scheduler

- **Fairshare**: Adjudicates what priority different groups get on Odyssey
- **Shares**: How much resources a group is allocated on Odyssey
- **TRES**: How Slurm charges back based on resources that are used
- **sshare**: A tool that can be used to see your current fairshare.

```
[root@holyitc01 ~]# sshare --account=test_lab -a
Account    User  RawShares  NormShares  RawUsage  EffectvUsage  FairShare
----------------- ---------- ---------- ----------- ----------- -------------
test_lab      244  0.001363   45566082  0.000572   0.747627
test_lab user1 parent  0.001363   8202875   0.000572   0.747627
test_lab user2 parent  0.001363   248820    0.000572   0.747627
test_lab user3 parent  0.001363   163318    0.000572   0.747627
```
Slurm Scheduler

- How long does my code take to run?

**Batch jobs:**
- `#SBATCH -p serial_requeue` # Partition
- `#SBATCH -t 0-02:00` # Runtime in D:HH:MM

**Interactive jobs:**
- `srun -t 0-02:00 -p test --pty OTHER_JOB_OPTIONS /bin/bash`
#!/bin/bash
#SBATCH -J Rjob1
#SBATCH -p shared
#SBATCH -n 1
#SBATCH -t 00:30:00
#SBATCH --mem=500M
#SBATCH -o %j.o
#SBATCH -e %j.e

## LOAD SOFTWARE ENV ##
module load R

input=M2.R

## EXECUTE CODE ##
R CMD BATCH $input $input.out
Is my code serial or parallel?

Serial (single-core) jobs

Batch jobs:
#SBATCH -p serial_requeue # Partition
#SBATCH -t 0-02:00 # Runtime in D:HH:MM
#SBATCH -n 1 # Number of cores/tasks

Interactive jobs:
srun -t 0-02:00 -n 1 -p test --pty OTHER_JOB_OPTIONS /bin/bash

Other Job Options: --x11=first # to start an interactive job with X11 forwarding
Parallel shared memory (single node) jobs

Examples:
- OpenMP (Fortran, C/C++)
- MATLAB Parallel Computing Toolbox (PCT)
- Python (e.g., threading, multiprocessing)
- R (e.g., multicore)

Batch jobs:
#SBATCH -p shared  # Partition
#SBATCH -t 0-02:00  # Runtime in D:HH:MM
#SBATCH -c 4  # Number of cores/tasks
#SBATCH -N 1  # Number of nodes

srun -c $Slurm_CPUS_PER_TASK code PROGRAM_OPTIONS

Interactive jobs:
srun -t 0-02:00 -c 4 -N 1 -p test --pty OTHER_JOB_OPTIONS /bin/bash
Parallel distributed memory (multi-node) jobs

Examples:
- MPI (openmpi, impi, mvapich) with Fortran or C/C++ code
- MATLAB Distributed Computing Server (DCS)
- Python (e.g., mpi4py)
- R (e.g., Rmpi, snow)

Batch jobs:
#SBATCH -p shared  # Partition
#SBATCH -t 0-02:00  # Runtime in D:HH:MM
#SBATCH -n 4  # Number of cores/tasks

Interactive jobs:
srun -t 0-02:00 -n 4 -p test --pty OTHER_JOB_OPTIONS /bin/bash
Slurm Scheduler

Serial and parallel shared memory (single node) jobs

**Batch jobs:**

```
#SBATCH -p shared  # Partition
#SBATCH -t 0-02:00  # Runtime in D:HH:MM
#SBATCH -c 4  # Number of cores/tasks for a Multi-threading jobs
#SBATCH -N 1  # Number of nodes
#SBATCH --mem=2000  # MB Memory per node
srun -c $Slurm_CPUS_PER_TASK code PROGRAM_OPTIONS
```

**Interactive jobs:**

```
srun -t 0-02:00 -c 4 -N 1 --mem=2000 -p test --pty OTHER_JOB_OPTIONS /bin/bash
```

Parallel distributed memory (multi-node) jobs

**Batch jobs:**

```
#SBATCH -p shared  # Partition
#SBATCH -t 0-02:00  # Runtime in D:HH:MM
#SBATCH -n 4  # Number of cores/tasks
#SBATCH --mem-per-cpu=4000  # Memory / core in MB
```

**Interactive jobs:**

```
srun -t 0-02:00 -n 4 --mem-per-cpu=4000 -p test --pty JOB_OPTIONS /bin/bash
```
#!/bin/bash
#SBATCH -p shared
#SBATCH -n 1
#SBATCH -t 00:10:00
#SBATCH --mem=500M
#SBATCH -o %A-%a.o
#SBATCH -e %A-%a.e
#SBATCH --array=2-20:2

## LOAD SOFTWARE ENV ##
module load R

input=M2.R

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

This is per array task resource needs
Job Script - Best Practices

• Keep unique copies of the stdout and stderr

  #SBATCH -o jobname.%j.o
  #SBATCH -e jobname.%j.e

• echo commands back

  #!/bin/bash -x
  set -x

• print statements

  input=file1.inp
  echo $input

• print runtime environment

  env

• make unique scratch directories

  mkdir -pv /n/regal/pi_lab/$USER/${Slurm_JOB_ID}.${input}
Intro Objectives

- What is RC?
- Do you speak Supercomputer?
- What resources are available?
- How do I access resources?
- How do I transfer files?
- How do I submit calculations?
- Am I using the resources effectively?
- What could go wrong?
- How do I get help?
Memory Requirements

• How much memory does my code require?

  • Understand your code and how the algorithms scale analytically (e.g. $X= [R]$ and $x^2$ vs $x^3$)

  • Run an interactive job and monitor memory usage (with the “top” Unix command)

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

Know your code

Example:

A real*8 (Fortran), or double (C/C++), matrix of dimension 100,000 X 100,000 requires ~80GB of RAM

<table>
<thead>
<tr>
<th>Data Type: Fortran / C</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer*4 / int</td>
<td>4</td>
</tr>
<tr>
<td>integer*8 / long</td>
<td>8</td>
</tr>
<tr>
<td>real*4 / float</td>
<td>4</td>
</tr>
<tr>
<td>real*8 / double</td>
<td>8</td>
</tr>
<tr>
<td>complex*8 / float complex</td>
<td>8</td>
</tr>
<tr>
<td>complex*16 / double complex</td>
<td>16</td>
</tr>
</tbody>
</table>
Memory Usage

Run an interactive job and monitor memory usage (with the “top” Unix command)

Example: Check the memory usage of a matrix diagonalization code

- Request an interactive bash shell session:
  ```
  srun -p test -n 1 -t 0-02:00 --pty --mem=4000 /bin/bash
  ```

- Run the code, e.g.,
  ```
  ./matrix_diag.x
  ```

- Open a new shell terminal and **ssh** to the compute node where the interactive job dispatched, e.g.,
  ```
  ssh <nodeName>
  ```

- In the new shell terminal run **top**, e.g.,
  ```
  top -u <username>
  ```
Memory Usage

Run 1:
Matrix dimension = 3000 X 3000 (real*8)
Needs 3,000 X 3000 X 8 / 1000000 = ~72 MB of RAM
Memory Usage

**Run 2: Input size changed**

Double matrix dimension, Quadrupole required memory
Matrix dimension = 6000 X 6000 (real*8)
Needs 6,000 X 6000 X 8 / 1000000 = ~288MB of RAM
Memory Example 2

- Do another example where the algorithm changes the complexity. See:
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 slurmdb.

• 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:

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

https://rc.fas.harvard.edu/resources/faq/how-to-know-what-memory-limit-to-put-on-my-job
Intro Objectives

- What is RC?
- Do you speak Supercomputer?
- What resources are available?
- How do I access resources?
- How do I transfer files?
- How do I submit calculations?
- Am I using the resources effectively?
- What could go wrong?
- How do I get help?
Test first

• Before diving right into submitting 100s or 1000s of research jobs. ALWAYS test a few first.
  – ensure the job will finish to completion without error
  – ensure you understand the resources needs and how they scale with different data sizes and input options
Types of Errors - Overview

- Scheduler (Slurm)
- Syntax
- Memory
- Storage
- File access
- Network
- Parallel communication
Types of Errors - Slurm

• Scheduler (Slurm)
  – errors executing commands (sbatch, squeue)

  `sbatch: error: Batch job submission failed: Unable to contact slurm controller`

  `squeue: error: slurm_receive_msg: Socket timed out on send/recv operation`
  `slurm_load_jobs error: Socket timed out on send/recv operation`

Don’t worry, try again – slurmtld process may be overwhelmed with work
Types of Errors - Syntax

• Syntax
  – job script

```bash
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -t 1:00:00
#SBATCH --mem=4000
#SBATCH -partition odyssey

# This is a Job Script for Syntax Errors
input file1.txt

echo $input
```

`sbatch: error: Invalid argument: odyssey`

`/var/slurmd/spool/slurmd/job70807187/slurm_script: line 8:   input: command not found`
Types of Errors - Memory

- Memory
  - out of memory
- malloc failure
  - C function that allocates bytes of memory and returns a pointer to the allocated memory
  - SIGSEGV, segfault or segmentation violation
    - arise primarily due to errors in use of pointers for virtual memory addressing, particularly illegal access.
- physical memory issue

**slurmstepd**: error: Exceeded step memory limit at some point.

**fortrl**: severe (174): SIGSEGV, segmentation fault occurred
Types of Error - Storage

• Storage
  – out of space on device
  – out of space on filesystem quota
  – out of inodes / file descriptors

*cp: closing `mtbd_water_tmd2_restart.namd`: No space left on device*

*cp: cannot create regular file `fastq.sh`: Disk quota exceeded*
Types of Errors – File Access

# This is a Job Script for Syntax Errors
input=/n/home_rc/pedmon/a.out

```
cat $input
mpirun a.out
```

• File access
  – no permission to read/write
    ```
    /n/home_rc/pedmon/a.out: Permission denied.
    ```
  – file or library not found
    ```
    /n/home_rc/pedmon/a.out: error while loading shared libraries: libquadmath.so.0: cannot open shared object file: No such file or directory
    ```
  – command not found
    ```
    /var/slurmd/spool/slurmd/job70844124/slurm_script: line 16: mpirun: command not found
    ```
Intro Objectives

- What is RC?
- Do you speak Supercomputer?
- What resources are available?
- How do I access resources?
- How do I transfer files?
- How do I submit calculations?
- Am I using the resources effectively?
- What could go wrong?
- How do I get help?
Request Help - Resources

- **https://rc.fas.harvard.edu/resources/support/**
  - Documentation
    - https://rc.fas.harvard.edu/resources/documentation/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@rc.fas.harvard.edu
  - Office Hours
    - Tuesday 2pm-3.30pm HSPH Kresge 204
    - Wednesday noon-3pm 38 Oxford - 206
  - Training
    - https://www.rc.fas.harvard.edu/upcoming-training/
RC Staff are here to help you and your colleagues effectively and efficiently use Odyssey resources to expedite your research endeavors.

Please acknowledge our efforts:

- "The computations in this paper were run on the Odyssey cluster supported by the FAS Division of Science, Research Computing Group at Harvard University."
- https://rc.fas.harvard.edu/odyssey/publications-citing-odyssey/