FASRC

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?

HARVARD
UNIVERSITY

Research Computing

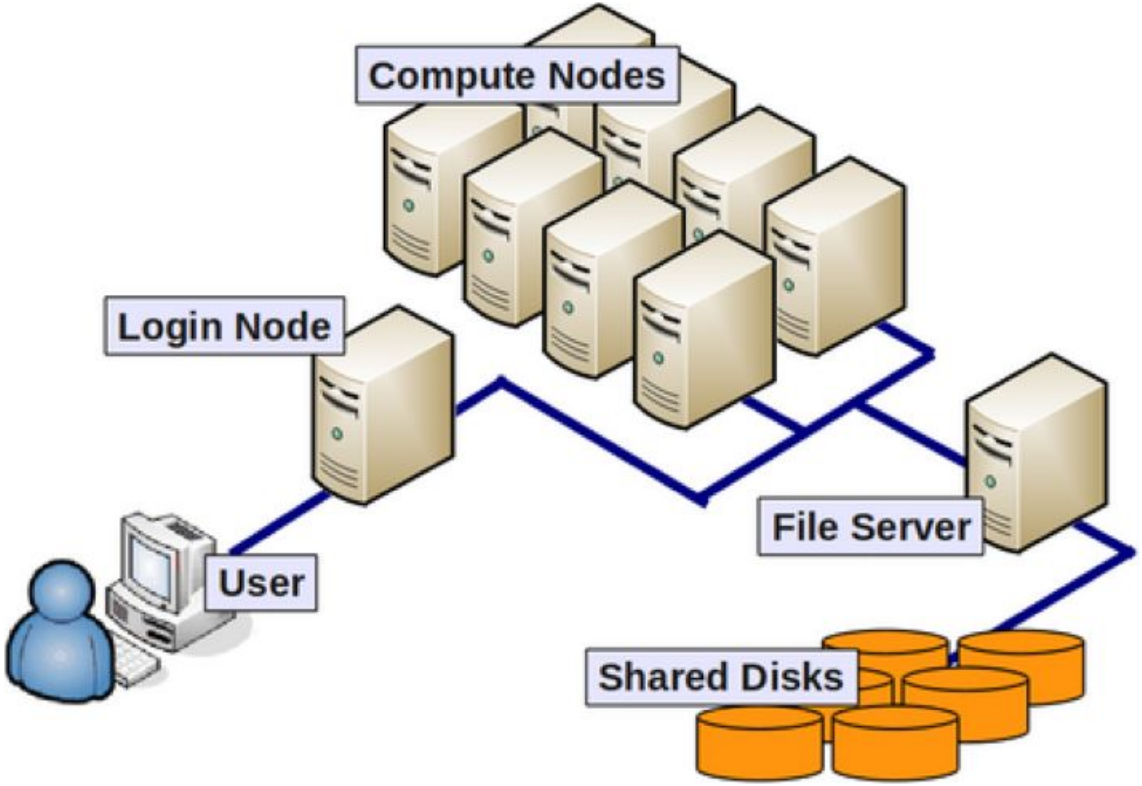
Faculty of Arts and Sciences (FAS) department that handles non-enterprise IT requests from researchers.

- **RC Primary Services:**
  - Cannon 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 5500+ 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



Cannon Components

Compute:

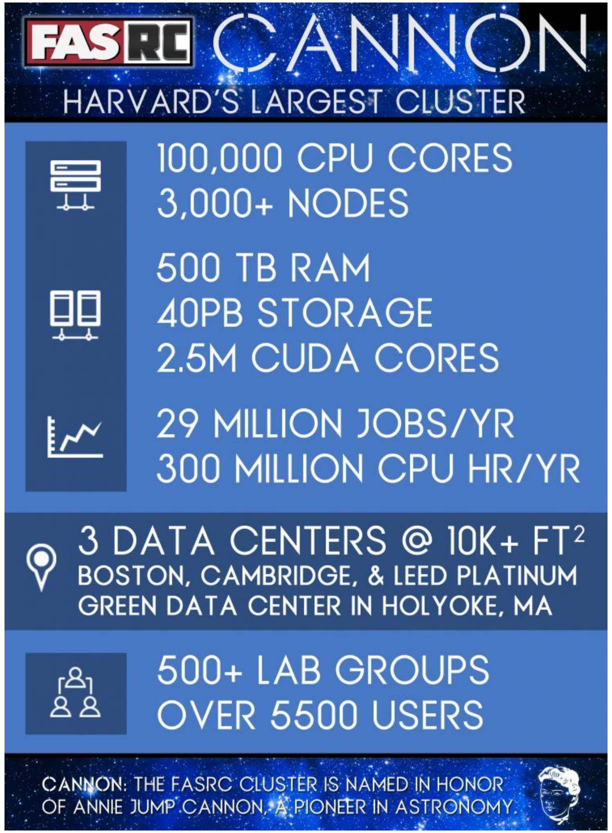
- 100,000 compute cores
- Cores/node: 8 to 64
- Memory/node: 12GB to 512GB (4GB/core)
- 2,500,000 NVIDIA GPU cores

Software:

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

Interconnect:

- 2 underlying networks connecting 3 data centers
- TCP/IP network
- Low-latency 200 GB/s HDR InfiniBand (IB) and 56 GB/s FDR IB network:
    - inter-node parallel computing
    - fast access to Lustre mounted storage



FASRC CANNON

HARVARD'S LARGEST CLUSTER

100,000 CPU CORES
3,000+ NODES

500 TB RAM
40PB STORAGE
2.5M CUDA CORES

29 MILLION JOBS/YR
300 MILLION CPU HR/YR

3 DATA CENTERS @ 10K+ FT^2
BOSTON, CAMBRIDGE, & LEED PLATINUM GREEN DATA CENTER IN HOLYOKE, MA

500+ LAB GROUPS
OVER 5500 USERS

CANNON: THE FASRC CLUSTER IS NAMED IN HONOR OF ANNIE JUMP CANNON, A PIONEER IN ASTRONOMY.

HARVARD UNIVERSITY

Storage Grid

FASRC

	Home Directories	Lab Storage	Local Scratch	Global Scratch	Persistent Research Data
Mount Point	/n/home#/ $USER	/n/pi_lab	/scratch	/n/scratchlfs	/n/holylfs
Size Limit	100GB	4TB+	70GB/node	2.4PB total	3PB
Availability	All cluster nodes + Desktop/laptop	All cluster nodes + Desktop/laptop	Local compute node only.	All cluster nodes	All cluster nodes
Backup	Hourly snapshot + Daily Offsite	Daily Offsite	No backup	No backup	External Repos No backup
Retention Policy	Indefinite	Indefinite	Job duration	90 days	3-9 mo
Performance	Moderate. Not suitable for high I/O	Moderate. Not suitable for high I/O	Suited for small file I/O intensive jobs	Appropriate for large file I/O intensive jobs	Appropriate for large I/O intensive jobs
Cost	Free	4TB Free + Expansion at $50/TB/yr	Free	Free	Free

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/
- Running Jobs : https://www.rc.fas.harvard.edu/resources/running-jobs/
- Software modules available : https://portal.rc.fas.harvard.edu/apps/modules
- Cannon Storage: https://www.rc.fas.harvard.edu/resources/cluster-storage/
- Interactive Computing Portal https://www.rc.fas.harvard.edu/resources/documentation/virtual-desktop/
- Singularity Containers: https://www.rc.fas.harvard.edu/resources/documentation/software/singularity-on-the-cluster/
- gpu computing https://www.rc.fas.harvard.edu/resources/documentation/gpgpu-computing-on-the-cluster/
- How to get help : https://www.rc.fas.harvard.edu/resources/support/

Login & Access

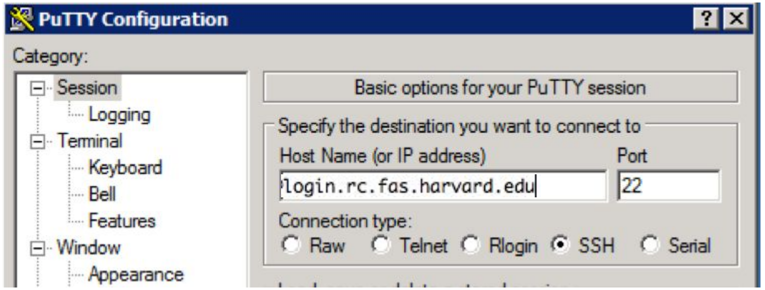
- Terminal application is needed to connect via secure shell (SSH)
  - Mac: Terminal.app on Mac/Linux
  - Linux: Xterm or Terminal

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

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

Password:
Verification code:
```

- Windows: Putty


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): login.rc.fas.harvard.edu

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

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/
- 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 5 minutes after 3 failed login attempts in a row.
- Password Reset: 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

```
module load R/3.2.0

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
```

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:
  - **sinfo**: Partitions you can use
  - **sbatch**: submit a batch job script
  - **srun**: submit an interactive test job
  - **squeue**: contact slurmctld 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

Partitions:	shared	gpu	test	gpu_test	serial_requeue	gpu_requeue	bigmem	unrestricted	**pi_lab**
Time Limit	7 days	7 days	8 hrs	1 hrs	7 days	7 days	no limit	no limit	**varies**
# Nodes	606	15	16	1	1930	155	6	8	**varies**
# Cores / Node	48	32 + 4 V100	48	32 + 4 V100	varies	varies	64	64	**varies**
Memory / Node (GB)	196	375	196	375	varies	varies	512	256	**varies**

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 Cannon
- **Shares**: How much resources a group is allocated on Cannon
- **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
test_lab user4 parent        0.001363   18901027  0.000572   0.747627
test_lab user5 parent        0.001363   18050039  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

Slurm Job Script

```
#!/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/3.2.0-fasrc01

input=M2.R

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

JOB SCRIPT HEADER

Load Module

Call the program

Slurm Scheduler

- 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

HARVARD UNIVERSITY

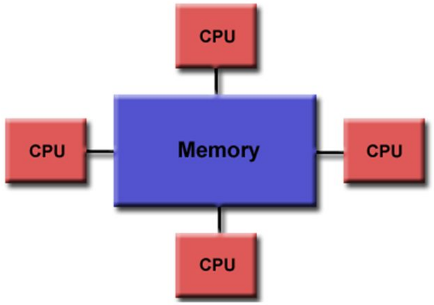
FASRC

Slurm Scheduler

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:

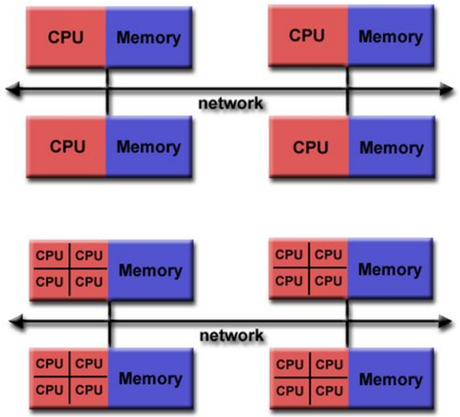
```
srun -t 0-02:00 -c 4 -N 1 -p test --pty OTHER_JOB_OPTIONS /bin/bash
```

Slurm Scheduler

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

srun -n $SLURM_NTASKS --mpi=pmix code PROGRAM_OPTIONS
```

Interactive jobs:

```
srun -t 0-02:00 -n 4 -p test --pty OTHER_JOB_OPTIONS /bin/bash
```

Slurm Scheduler

GPU jobs

Batch jobs:

```
#SBATCH -p gpu_requeue                # Partition
#SBATCH -t 0-02:00                    # Runtime in D:HH:MM
#SBATCH -n 1                    # Number of cores/tasks
#SBATCH --gres=gpu:1            # Number of GPUs
#SBATCH --constraint=”v100”     # GPU type
#SBATCH --gpu-freq=high            # GPU Frequency
```

Interactive jobs:

srun -t 0-02:00 -n 1 --gres=gpu:1 -p gpu_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
```

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
```

Slurm Job Arrays Example

```
#!/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/3.2.0-fasrc01

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 strderr

```
#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/scratchlfs/pi_lab/$USER/${Slurm_JOB_ID}.${input}
```

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 a jobs on the cluster and thus use fairshare but also can run on any partition

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

Data Type: Fortran / C	Bytes
integer*4 / int	4
integer*8 / long	8
real*4 / float	4
real*8 / double	8
complex*8 / float complex	8
complex*16 / double complex	16

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>
  ```

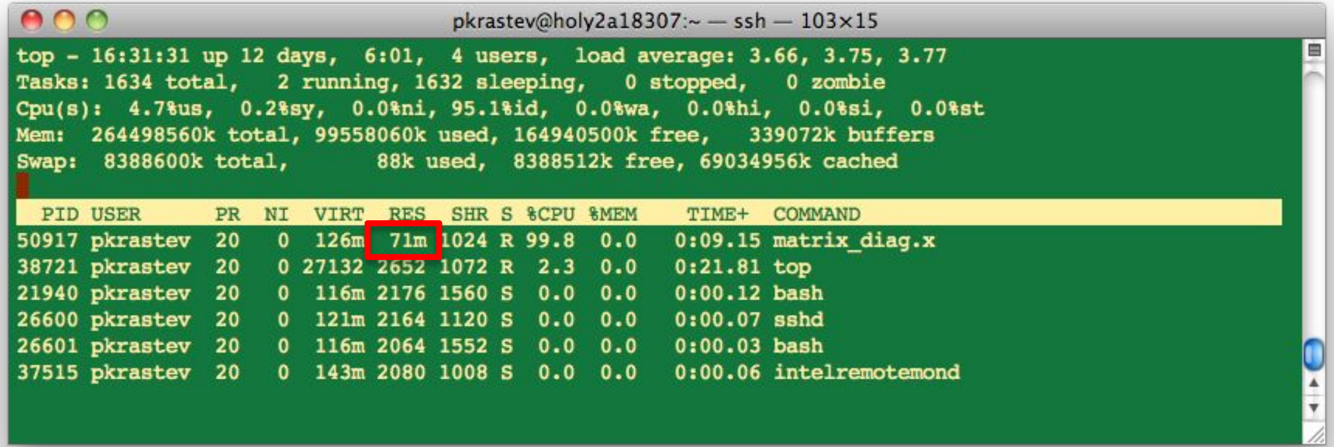
- **In the new shell terminal run top, e.g.,**
  ```
  top -u <username>
  ```

Memory Usage

Run 1:

Matrix dimension = 3000 X 3000 (real*8)

Needs 3,000 X 3000 X 8 / 1000000 = ~72 MB of RAM



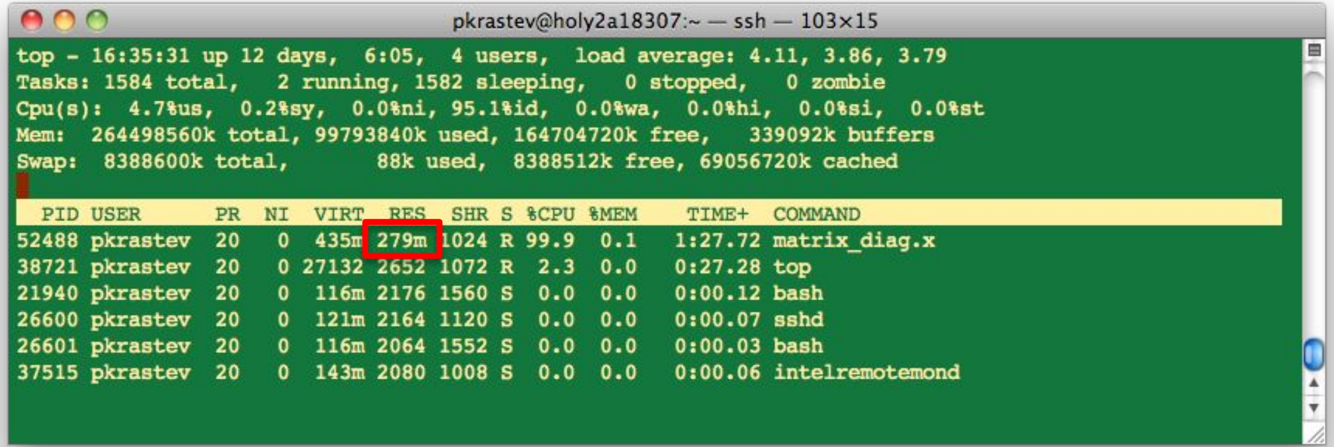
pkrastev@holy2a18307:~ — ssh — 103×15

```
top - 16:31:31 up 12 days,  6:01,  4 users,  load average: 3.66, 3.75, 3.77
Tasks: 1634 total,   2 running, 1632 sleeping,   0 stopped,   0 zombie
Cpu(s):  4.7%us,  0.2%sy,  0.0%ni, 95.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  264498560k total, 99558060k used, 164940500k free,   339072k buffers
Swap:  8388600k total,       88k used,  8388512k free, 69034956k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
50917 pkrastev  20   0  126m  71m 1024 R 99.8  0.0   0:09.15 matrix_diag.x
38721 pkrastev  20   0 27132 2652 1072 R  2.3  0.0   0:21.81 top
21940 pkrastev  20   0  116m 2176 1560 S  0.0  0.0   0:00.12 bash
26600 pkrastev  20   0  121m 2164 1120 S  0.0  0.0   0:00.07 sshd
26601 pkrastev  20   0  116m 2064 1552 S  0.0  0.0   0:00.03 bash
37515 pkrastev  20   0  143m 2080 1008 S  0.0  0.0   0:00.06 intelremotemond
```

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


pkrastev@holy2a18307:~ — ssh — 103×15

```
top - 16:35:31 up 12 days,  6:05,  4 users,  load average: 4.11, 3.86, 3.79
Tasks: 1584 total,   2 running, 1582 sleeping,   0 stopped,   0 zombie
Cpu(s):  4.7%us,  0.2%sy,  0.0%ni, 95.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  264498560k total, 99793840k used, 164704720k free,   339092k buffers
Swap:  8388600k total,       88k used,  8388512k free, 69056720k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
52488 pkrastev  20   0  435m 279m 1024 R 99.9  0.1   1:27.72 matrix_diag.x
38721 pkrastev  20   0 27132 2652 1072 R  2.3  0.0   0:27.28 top
21940 pkrastev  20   0  116m 2176 1560 S  0.0  0.0   0:00.12 bash
26600 pkrastev  20   0  121m 2164 1120 S  0.0  0.0   0:00.07 sshd
26601 pkrastev  20   0  116m 2064 1552 S  0.0  0.0   0:00.03 bash
37515 pkrastev  20   0  143m 2080 1008 S  0.0  0.0   0:00.06 intelremotemond
```

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 *ouput_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

    ReqMem     MaxRSS
    ----------        ----------
    1000Mn
    1000Mn      286712K
```

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 – slurmctld process may be overwhelmed with work

Types of Errors - Syntax

- Syntax
  - job script

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

```
sbatch: error: Invalid argument: odyssey
```

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

echo $input
```

  - input files or data files

```
/var/slurmd/spool/slurmd/job70807187/slurm_script: line 8:     input: command not found
```

Types of Errors - Memory

- Memory
  - out of memory

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

  - 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.

```
forrtl: severe (174): SIGSEGV, segmentation fault occurred
```

  - physical memory issue

Types of Error - Storage

- Storage
  - out of space on device

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

  - out of space on filesystem quota

```
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

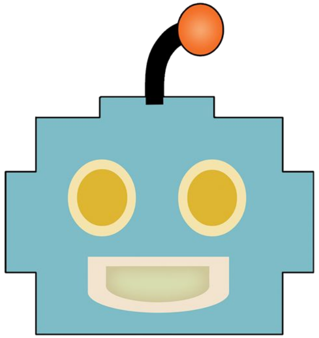
```
/var/slurmd/spool/slurmd/job70844124/slurm_script: line 16: mpirun:       command not found
```

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
    - Wednesday noon-3pm 38 Oxford - 100
  - Consulting Calendar
    - https://www.rc.fas.harvard.edu/consulting-calendar/
  - Training
    - https://www.rc.fas.harvard.edu/upcoming-training/



- RC Staff are here to help you and your colleagues effectively and efficiently use Cannon resources to expedite your research endeavors.
- Please acknowledge our efforts:
  - "The computations in this paper were run on the Cannon cluster supported by the FAS Division of Science, Research Computing Group at Harvard University."
  - https://www.rc.fas.harvard.edu/about/attribution/

Backup Slides

Memory Example 2

- Do another example where the algorithm changes the complexity. See: https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations