HARVARD
Faculty of Arts and Sciences
RESEARCH COMPUTING

Introduction to Odyssey

Bob Freeman, PhD
RC Facilitator
XSEDE Campus Champion

robert_freeman@harvard.edu
@DevBioInfoGuy

Goals

HARVARD
Faculty of Arts and Sciences
RESEARCH COMPUTING

Objectives

- Use compute resources efficiently, to get your results faster
  - Quicker to present at conferences
  - Better positioned when securing funding
  - More competitive when applying for positions
- Submit jobs without affecting the work of 500+ other active users
- To think about the science, not the technology logistics

To enable you to be successful with your research!

Overview

HARVARD
Faculty of Arts and Sciences
RESEARCH COMPUTING

1. All About Odyssey
2. Typical Workflow
   a. Login & Access
   b. Filesystems & Storage
   c. Transferring Files
   d. Loading Software
   e. Login/Interactive Nodes
   f. Choosing Appropriate Resources
   g. Submitting/Controlling Jobs
3. Troubleshooting
4. Common Pitfalls
5. Getting Help

What is Odyssey?

HARVARD Faculty of Arts and Sciences RESEARCH COMPUTING

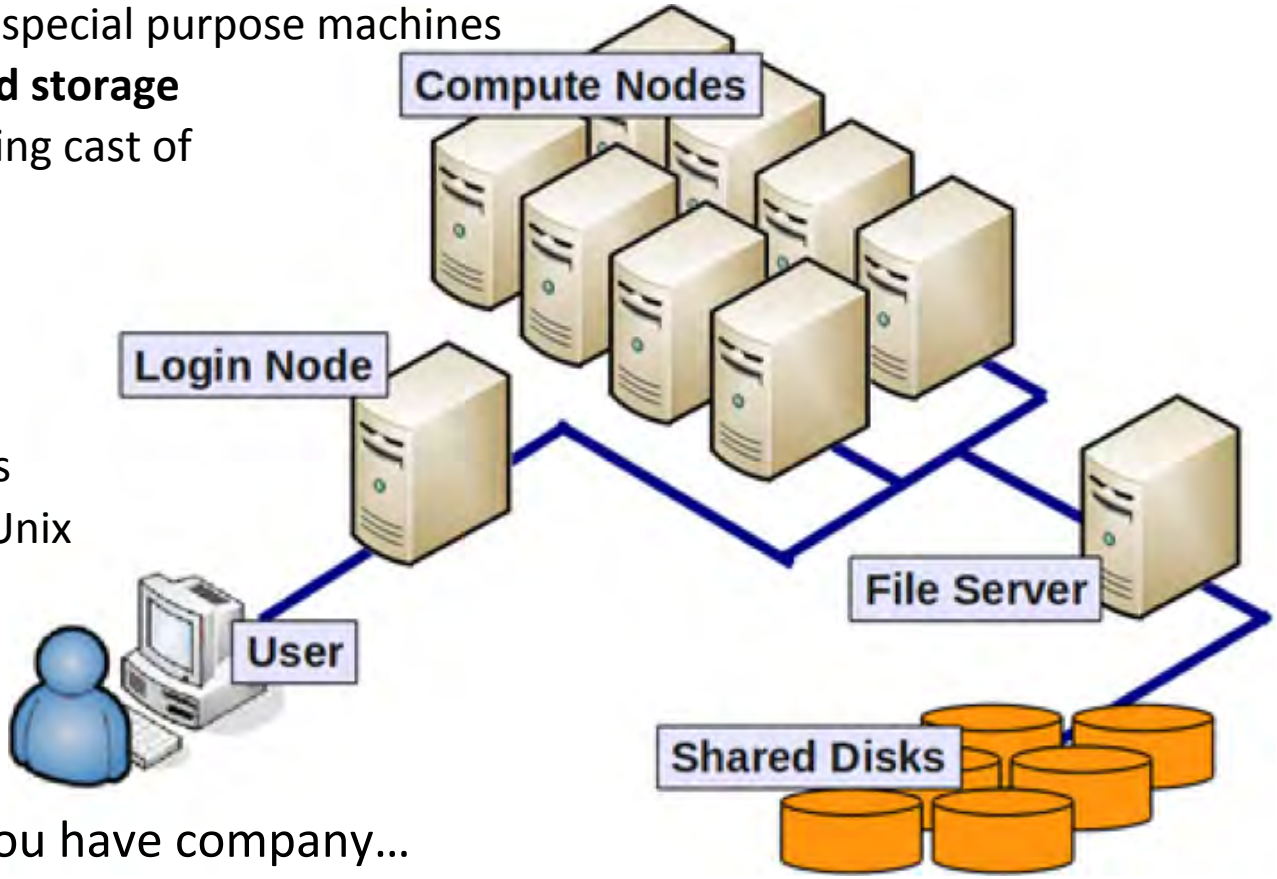
RC's premier resource is the Odyssey cluster. What is a cluster?

A collection of various types of hardware:

- A cluster of tightly interconnected **machines with identical hardware**
- Several high-powered, special purpose machines
- Large amount of **shared storage**
- Miscellaneous supporting cast of other servers

And some software:

- **User and group management:** separation of resources
- **SLURM** (Simple Linux/Unix Resource Manager)
- Linux OS (CentOS 6)



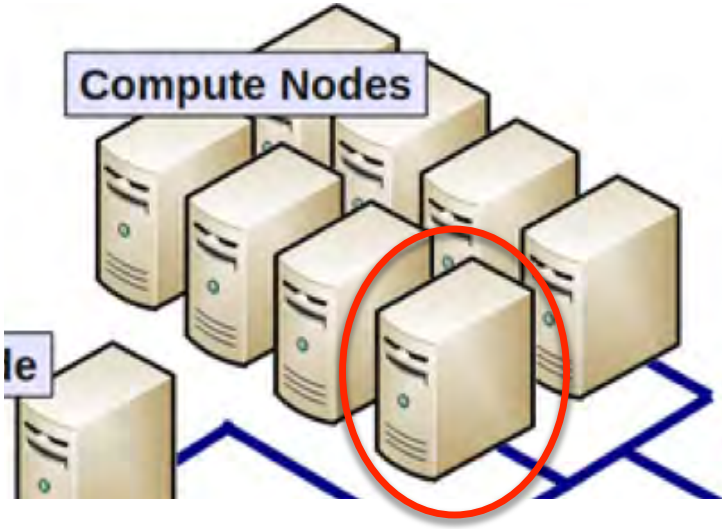
It's a **shared** system -- you have company…

http://fasrc.us/whatisodyssey

Key definitions...

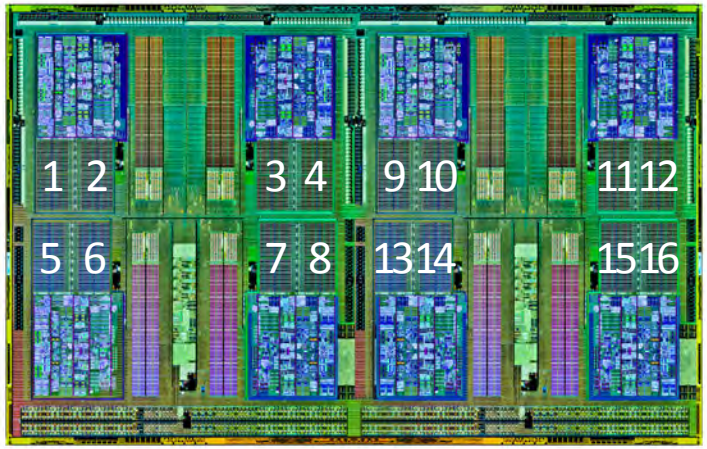
The typical hardware unit is called a **node**

- Same stuff that's in a desktop/laptop: CPUs, Memory, Hard drive, Network cards
- But more powerful and more of them compared to a typical desktop
- Nodes are individual hosts with distinct names. E.g...
  - rclogin03: one of the login nodes, located in Boston area
  - holy2a18208: one of the compute nodes, located in Holyoke, MA



The basic computational unit in a cluster is a **CPU core**

- Each core runs one process, a *average* job
- Most compute nodes have 64 cores arranged on 4 CPUs (16 cores/CPU)
- Thus, most nodes run 64 batch job processes

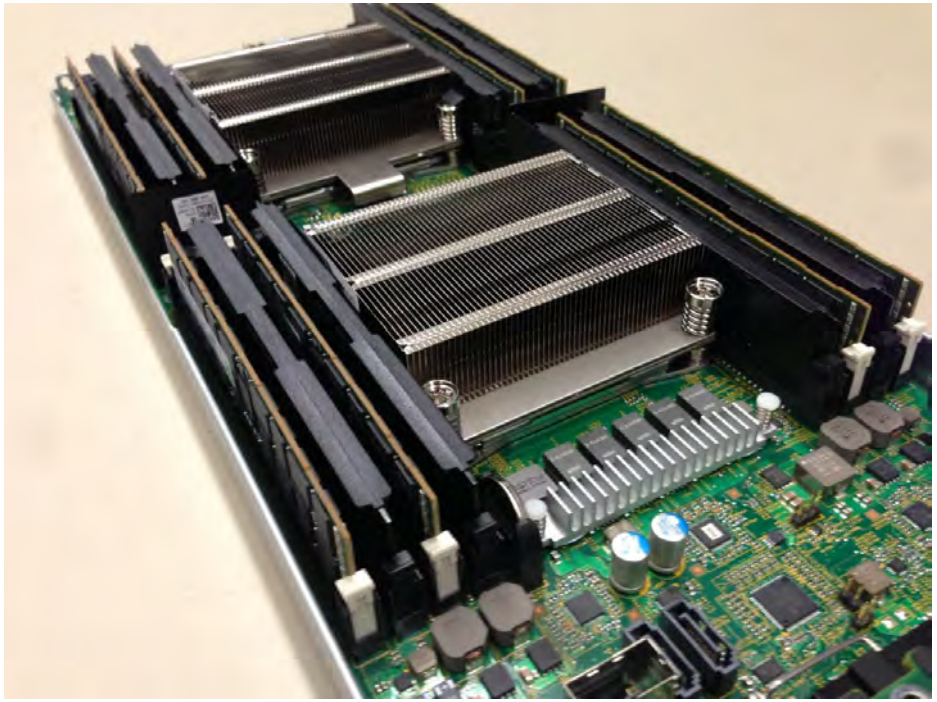


Key definitions...

HARVARD
Faculty of Arts and Sciences
RESEARCH COMPUTING

A **typical compute node** is configured:

- 64 cores
- 256 GB RAM, or ~4 GB RAM/core
- 2 network cards:
  Infiniband (intraconnect)
  & xGb connections (interconnect)
- Small, local hard disk/SSD
  for boot and local /scratch



All cores on a node share all other resources of the node: memory, network bandwidth, etc.

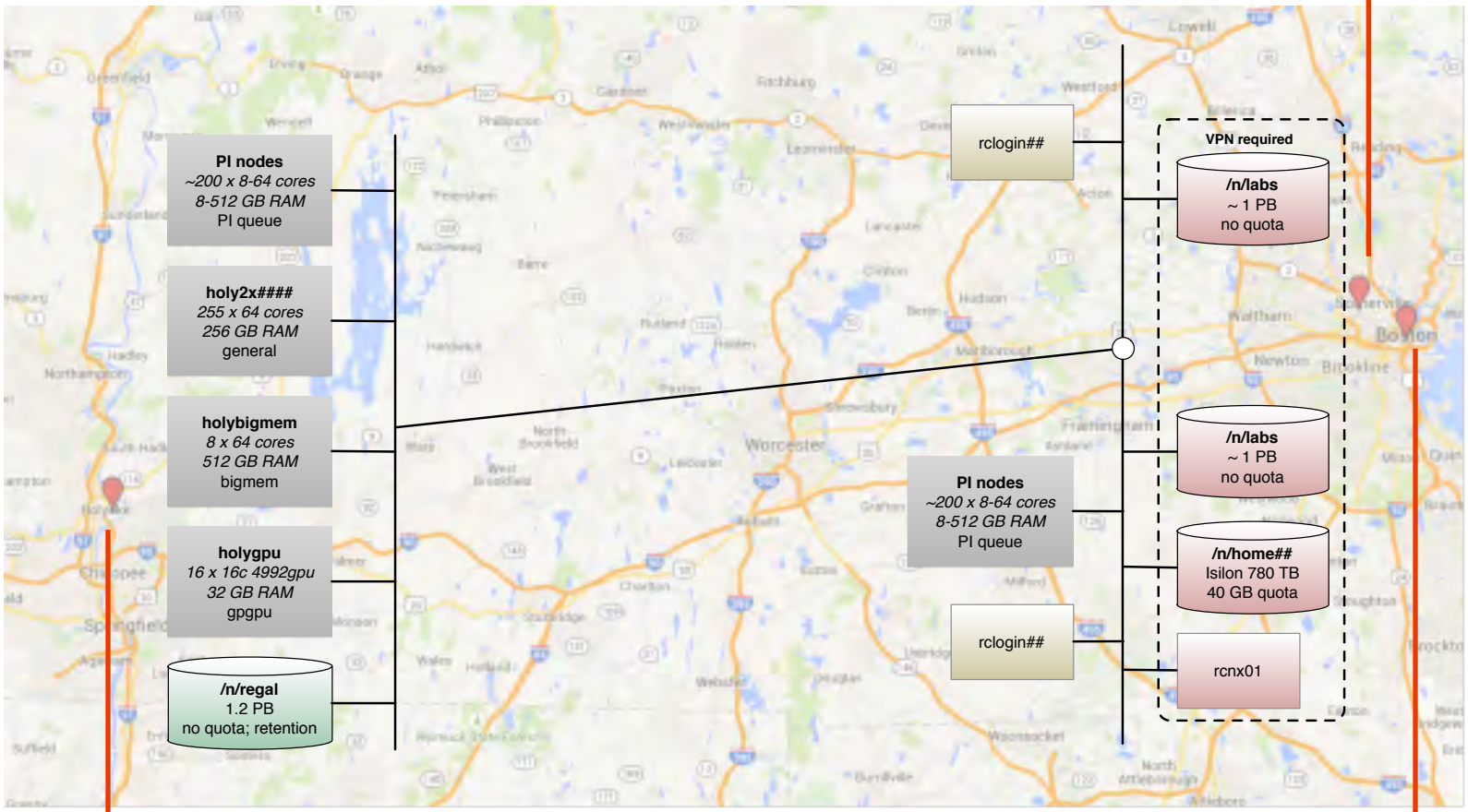
Thus, how you use these resources affects the other 63 jobs on that compute node

What is Odyssey?

HARVARD Faculty of Arts and Sciences RESEARCH COMPUTING

Compute nodes/disk are located in 3 data centers:

60 Oxford Street



MGHPCC, Holyoke MA

1 Summer Street

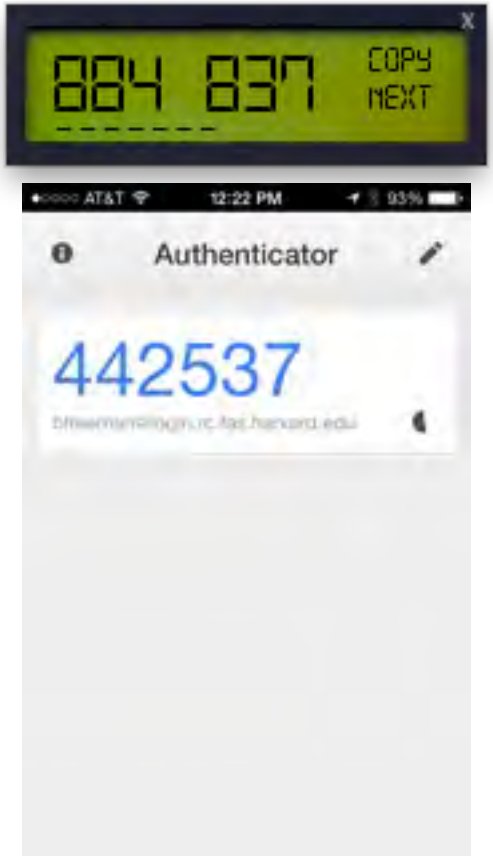
Topology may effect the efficiency of work

Typical Workflow

1. Login in to Odyssey
   a. Land on a login (head) node, appropriate for light work only
2. Copy/upload/download some files
3. Get interactive session
4. Load appropriate software
5. Test your program/script interactively to ensure it runs properly
6. Test run in batch: create batch file & submit to SLURM
   a. Continue working in the foreground while waiting for results
7. Scale up as necessary (10s, 100s, 1000s)
   a. With *caveats*: proper file placement, # cores, etc.

Login & Access

- Use your RC credentials for:
  - Instrument Windows machines
  - VPN
  - File transfer clients
  - Mounting disk shares
  - Terminal sessions to Odyssey
- OpenAuth 2-factor authentication (2FA) required for VPN & Odyssey sessions (file transfer & terminal)
- Account locks automatically if 5 failed login attempts, and auto unlocks after 10 minutes
- **Reset your own password** on RC portal
- If you are switching labs, please let us know, as we'll need to change your access groups



Account credentials should not be shared!
Using RC services in an explicit acceptance of the University Security Policy
http://security.harvard.edu/book/information-security-policy

Login & Access

Recommended SSH clients:

- Terminal on Mac/Linux
- Putty on PC
- X11: Xquartz (Mac) or Xming (PC)

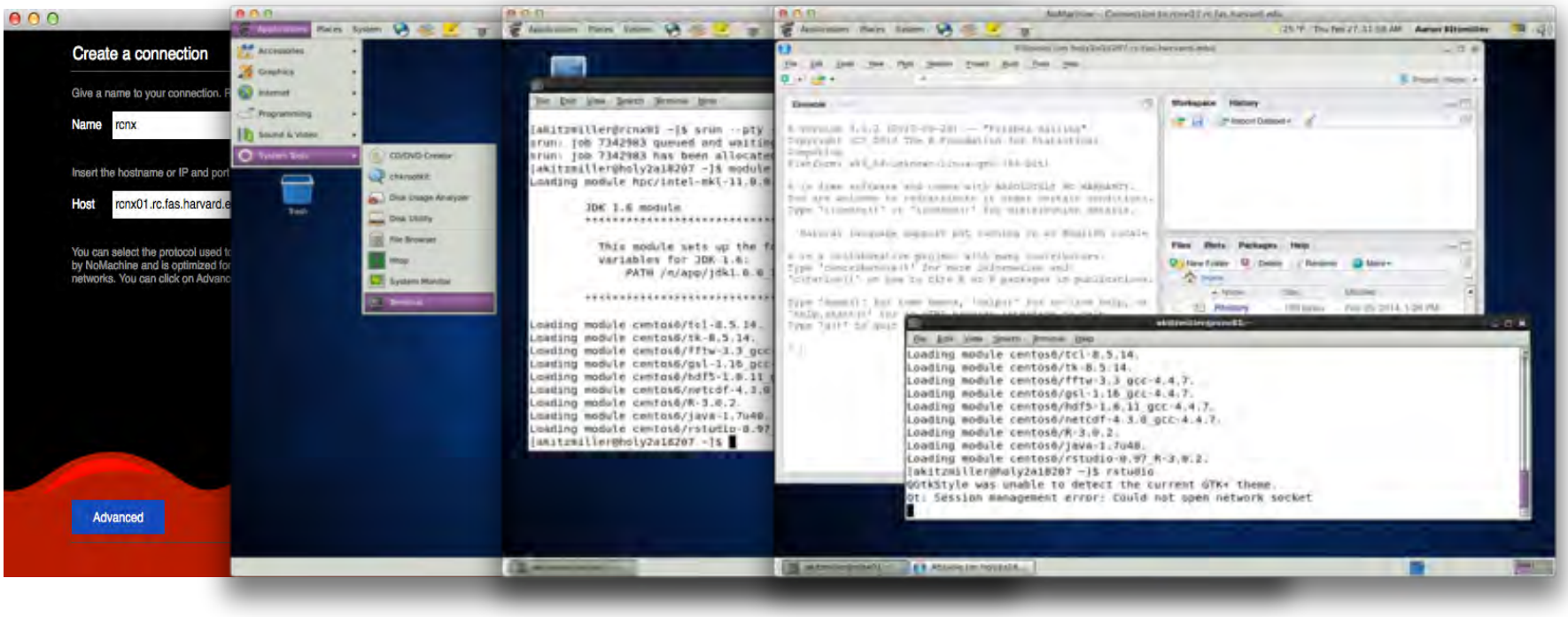
```
ssh rcusername@login.rc.fas.harvard.edu*

# Or, if X11 forwarding is required...
ssh -YC username@login.rc.fas.harvard.edu
```

We expect you to have good Unix skills before submitting jobs on the cluster. See http://fasrc.us/unixfoo

*unless in sensitive information setups

GUI Login

- Some apps require GUI/X11 interface: MATLAB, RStudio, CLCBio, etc..
- Use NoMachineX instead, as X11 performance can sluggish
- *VPN is required* (vpn.rc with username@odyssey* + 2FA)
- As this is a login node, *request an interactive* session to do any work



* exception is for sensitive information setups

Transferring files to/from Odyssey

GUI client FileZilla for all platforms

- Configure according to http://fasrc.us/configfilezilla to avoid 2FA problems

Command line tools `scp` or `rsync`

- `rsync` is best for resuming transfers or transferring only changed file parts

Download data using `curl` or `wget`

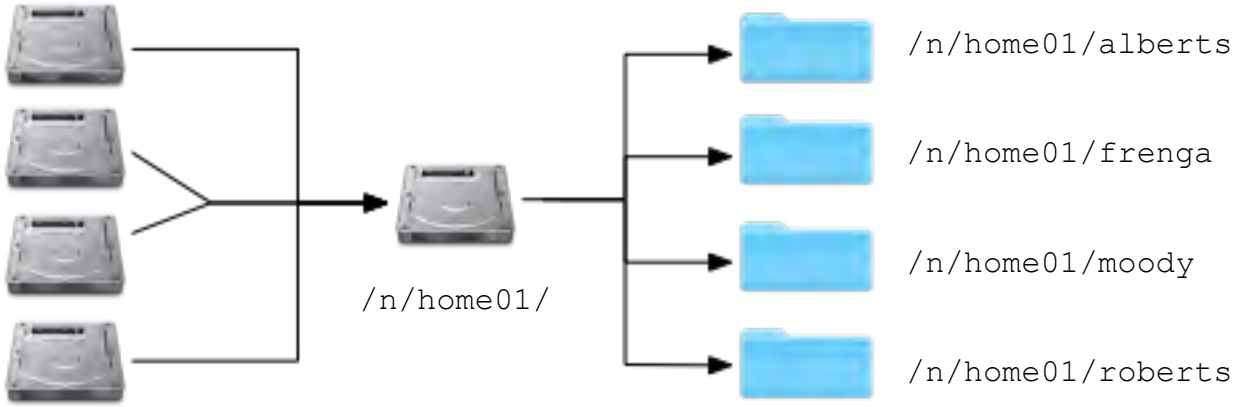
- Both are available on all nodes, though web proxy needed for HRCI setups

Or by mountings disk shares. Please see http://fasrc.us/mountdisks

Examples:

```
# copy file in current directory to Odyssey home folder
scp somefile.txt rcuser@login.rc.fas.harvard.edu:~

# copy folder in current directory & contents to Odyssey home folder
rsync -av myfolder rcuser@login.rc.fas.harvard.edu:~

# download FASTA sequence from NCBI
wget "http://www.ncbi.nlm.nih.gov/nuccore/L03535.1?report=fasta&log$=seqview&format=text"
```

Filesystems & Data Storage



Storage on Odyssey is not what and where you think it is…

- Created by bundling together a group of disks to form a virtual volume
- The virtual volume is sliced up into one or more filesystems to hold files & folders
- These are accessed transparently over the network through *mount points* (e.g. `/n`)

Running large #s of jobs out of home or lab directories will negatively affect all other persons sharing those physical disks

Take home message: Ensure that you use the proper filesystem for your work

Common Filesystems

	Type	Size	Avail?	Mount Desktops?	Backup?[1]	Retention?	I/O profile
`/n/home##`	NFS	40 GB (hard limit)	all nodes	Y	Y	N	low
`/n/labfs#`	NFS	1 TB free (new labs) contact for costs	all nodes	Y	Y [2]	N	low
`/scratch`	local	250 GB/node (~4 GB/core)	all nodes	N	N	Y[3]	high
`/n/regal`	Lustre	1.2 PB	all nodes	N[4]	N	90-days[5]	high

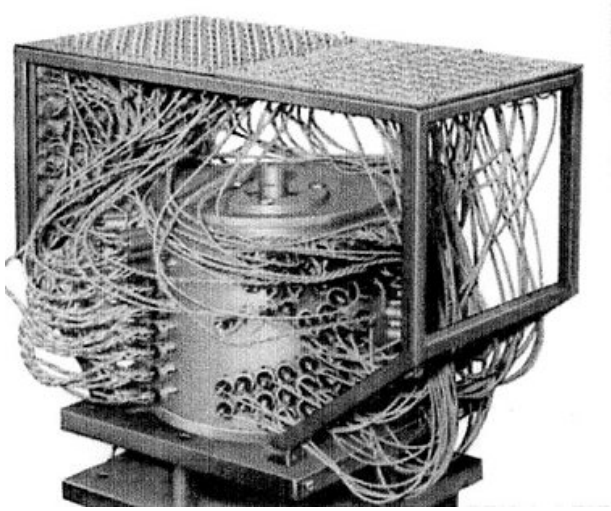
[1]Backup methods differ. See http://fasrc.us/faqrecovery for more information.
[2]Lab disks shares are typically backed up unless noted.
[3]Files usually deleted when job finished. Please clean up your own mess, though.
[4]Can use file transfer methods to stage data.
[5]Retention is typically run at maintenance times. Areas can be exempted for common data (e.g. NCBI Genbank at `/n/regal/informatics_public`). Contacts us.

Common Filesystems: /scratch

Using local `/scratch`:

- 250 GB slice on each compute node, so there's about 4 GB disk space/job
- Is currently underutilized, so more space may be available (check `sbatch` options)
- Can see speedup of 2x – 3x, depending on pattern of file read/writes
- Since is local to each node, must use it *during* your job:

```
...
start_dir=$PWD
mkdir -p /scratch/$USER/$SLURM_JOBID
cd /scratch/$USER/$SLURM_JOBID

# do your work while writing temp files here
...

# copy files back and return from whence we came
cp –r results/ $start_dir/
cd $start_dir

# now cleanup
rm -rf /scratch/$USER/$SLURM_JOBID
```



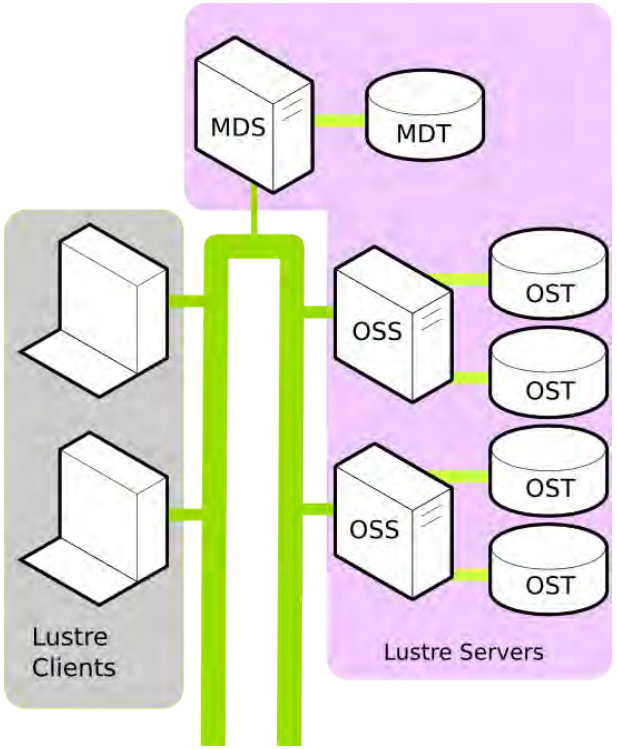
Common Filesystems: /n/regal

Using `/n/regal`:

- Most work should be done here, especially for ~≥ 10 simultaneous jobs
- No space restrictions, but files > 90 days old deleted (usually at maintenance)
- Can stage files prior to job by typical copy/rsync commands or FileZilla
- Remember to copy results back to home or lab shares for permanent storage

A couple more things to remember:

- Shared lab areas can be exempt from retention. Contact us.
- Public data sets can also be staged here – no need to keep your own copy
- NCBI, EMBL, UCSC data is stored at `/n/regal/informatics_public`:
  - FASTA data, BLAST databases, Bowtie2 indexes
- Contact us if you'd like to add more to this location



Load/Installing software

We have ~3000 applications/libraries in chemistry, biology, statistics, social sciences, and more available for use, but not at all the same time

Software is loaded incrementally using modules, to set up your shell environment

Rolling out a new module system `Lmod`:

- New system is opt-in for old accounts, but will be default soon (if not already)
- Strongly suggested reading: http://fasrc.us/rclmod

```
source new-modules.sh                  # for opt-in folks
module load fastqc                     # most recent version

module load fastqc/1.0.0-fasrc01       # specific version

module spider fastqc                   # find details on software
module avail 2>&1 | grep -i fastqc     # find software titles
```

Load/Installing software

Old (legacy) modules system still available…

```
module load legacy                      # get to old modules
module load centos6/bowtie-1.2.1        # old can be used with new
```

If not using `Lmod` , the old (legacy) module system still available, but will be retired soon

These modules are at http://fasrc.us/modulelist, or

```
module avail 2>&1 | grep -i 'fastqc'    # find software

module load centos6/fastqc-0.10.0
```

Module loads best placed in SLURM batch scripts (vs. in your `.bashrc` login script):

- 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

Load/Installing software

Please only request software installs if the program will be used by multiple labs or if standard in your domain.

For all other software, please install software yourself

- Follow software instructions for 'local', non-root, or 'home' directory installation
- Consult our web site for instructions on using appropriate compilers
- Please don't use the `sudo` command

For Perl / Python modules or R packages

- Installation can be in home folder (personal) or lab folder (shared)

Consider using home folder for personal or for code under development; and lab folder for code in shared projects or production code

Details at http://fasrc.us/installsw

Login vs Interactive Nodes

Terminal sessions to `login.rc` puts you on one of several login nodes

- This gateway to the cluster has limited entry points, so..
- Only non-CPU-intensive work is appropriate: `cp`, `mv`, `nano`, `rsync`, etc.
- *Reminder*: rcnx01 and holynx01 are login nodes

Don't compute here, instead

- Submit a batch job (background task) to SLURM, or
- Request an interactive session (foreground task) on a compute node:

```
srun --pty --x11=first --mem 1000 -p interact -t 0-6:00 -n 1 -N 1 /bin/bash
```

srun: foreground
sbatch: background

Resources that you wish to request from SLURM

Script or program
`/bin/bash` == shell

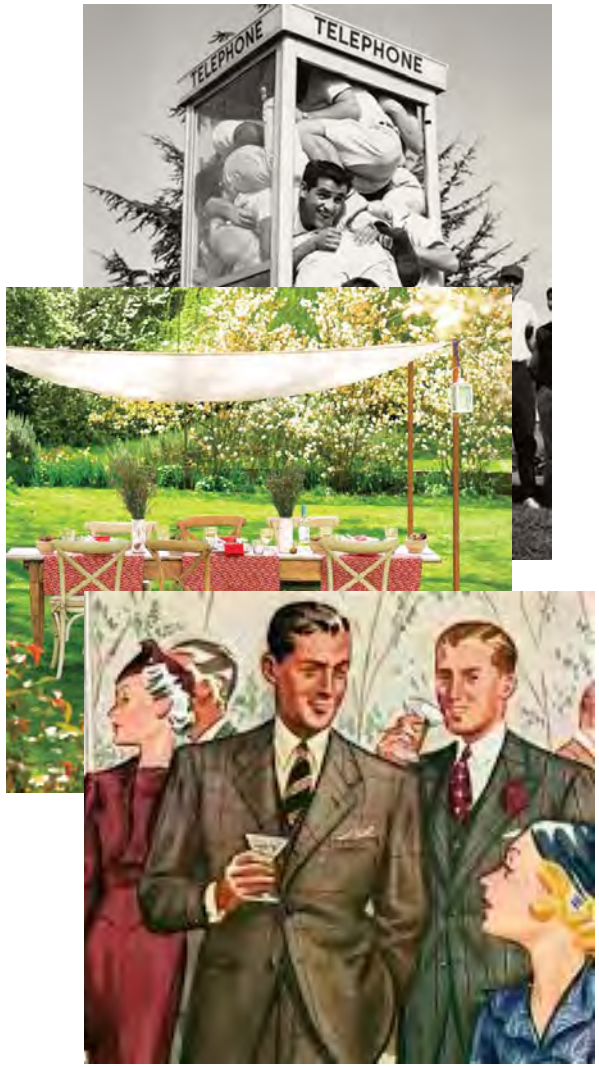
Choosing Resources: How?

Choosing resources is like attending a party:

- You need to RSVP the number of guests you intend to bring
  Request the resources you intend to use
- Extra guests: there's not enough food and drink for everyone
  CPU/disk overage: all jobs including your will run more slowly
  RAM/time overage: your job will be killed
- Too few: an unhappy host and wasted $$ / effort
  CPU/RAM: resources are wasted as they cannot be used by anyone else
  All: your job becomes harder to schedule

You also want to be polite:

- Stay the appropriate amount of time…
  Try to approximate your resource use with some padding for safety
- Don't slip in, drink & eat, and leave within minutes
  Try to avoid jobs that start and complete within minutes; especially in large numbers



Choosing Resources: Time & Memory

Time:

- Determined by your test runs during an interactive session
- Or if trying in batch, over-ask first, then reduce time on later runs
- Due to scheduler overhead, jobs should do at least 5 - 10 min of work

Memory:

- Check software docs for memory requirements
- If none stated, over-ask and do a trial run (via `srun` or `sbatch`)
- use `sacct` command to get post-run job info:

```
# RAM                                    requested/used!!
sacct -j JOBID --format=JobID,Elapsed,ReqMem,MaxRSS
```

"Never use a piece of bioinformatics software for the first time without looking to see what command-line options are available and what default parameters are being used"

-- acgt.me · by Keith Bradnam

Choosing Resources: Partitions

Name	Length	Size (cores)	Memory/node	Usage
interact	3 days	512 (8 nodes)	256 GB	all interactive work
serial_requeue	7 days*	30K+	varies (512 GB max)	best for single core jobs; or small numbers of cores for short durations; schedules best as hits all parts of the cluster
general	7 days	~14K	256 GB	large # of cores; MPI jobs; jobs sensitive to pre-emption
unrestricted	no limit	512	256 GB	all jobs with no time limit
bigmem	7 days	512	512 GB	jobs requiring >256 GB RAM (restricted access)
(private)	no limit	varies	256 GB typical	lab-specific partitions

Note: SLURM can schedule to quickest of two partitions with `-p partion1,partion2`

Choosing Resources: Partitions

interact

- Use for foreground, interactive sessions.
- You can request multiple cores or large RAM
- Limit the number of active, interactive session to 5 or less

(private)

- PI-specific partitions, usually named after the lab
- Access is automatic, by group membership

gigmem

- For work where each job requires > 256 GB RAM
- Can be accessed only by request

general

- For all large core #, long, or MPI jobs, or jobs sensitive to pre-emption
- When busy, typically will take tens of minutes or hours to schedule
- Requesting full nodes may take >1 day for your job to schedule

Choosing Resources: Partitions

serial_requeue

- Recommended partition for single-core jobs; or jobs using up to 8 cores lasting up to approx. 6 – 12 hrs
- Most 'powerful' as hits every core on the cluster, including private compute
- Dispatches within seconds to minutes

But the downside…

- Jobs may be pre-empted (killed) and automatically rescheduled if originally scheduled on a private node and the node owner submits work

To mitigate this…

- If you append output, ensure that you zero your data files at the start of the job, to ensure that any files left over from a previous, partial run are removed.
- Structure your command flow so that you skip over any work already done. This allows your re-run job to pick up from where the pre-empted one left off.

Submitting jobs

Two methods for submitting jobs via `sbatch`…

For simple, one line commands, submit with `sbatch`:

```
sbatch -p serial_requeue -n 1 -N 1 -t 0-1:00 \
--mem 100 -o hostname.out -e hostname.err \
--mail-type=ALL --mail-user=rmf@123.com \
--wrap="rsync -av ~/test/some_set_of_files \
        /n/regal/my_lab/bfreeman/high_IO_destination/"
```

Required
Recommended
Optional

The flags are your resource requests and command to run is enclosed by `--wrap=""`

After you enter your `sbatch` command, SLURM will return..

```
Submitted batch job 29484165
```

This jobID is your way of tracking the job, controlling it, or obtaining info about it

Submitting jobs

The other is to create a SLURM script file using a recommended text editor:

- TextWranger/BBEdit on Mac
- GEdit/NotePad+ on PC
- nano, vi, emacs on Linux

The script files will contain resource requests (and other directives) and your code. And submit to SLURM via same `sbatch` command:

```
sbatch fastqc.slurm
```

OK, so we're going to get rather technical…

We will briefly highlight template SLURM script files for four types of jobs:

- *Single core (serial)*: sequentially-executing code that typically runs on one core
- *Multicore (multithreaded)*: code that is structured to allow multiple parts to run concurrently (in parallel) across multiple cores on one compute node
- *Multicore (openMP)*: a special type of multithreaded code
- *Multinode (MPI)*: code designed to run in parallel, but across multiple compute nodes and communicate with one another through a Message Passing Interface

Submitting Jobs - Single Core

`fastqc.slurm` file contents:

Required

```
#!/bin/bash
#
#SBATCH -p serial_requeue          # Partition to submit to (comma separated)
```

Recommended

```
#SBATCH -J frog_fastqc             # Job name
#SBATCH -n 1                       # Number of cores
#SBATCH -N 1                       # Ensure that all cores are on one machine
#SBATCH -t 0-1:00                  # Runtime in D-HH:MM (or use minutes)
#SBATCH --mem 100                  # Memory in MB
#SBATCH -o fastqc.out              # File to which standard out will be written
#SBATCH -e fastqc.err              # File to which standard err will be written
```

Start

```
#SBATCH --mail-type=ALL            # Type of email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=rmf@123.com    # Email to which notifications will be sent
```

Required

```
source new-modules.sh; module load fastqc

cd my_output_directory
fastqc --casava -o fastqc_reports A01_R1.pair.fastq.gz
...
... do more processing here...
```

Submitting Jobs - Multicore

`bowtie.slurm` file contents:

Required

```
#!/bin/bash
#
#SBATCH -p serial_requeue,general # Partition to submit to (comma separated)
```

Recommended

```
#SBATCH -J frog_bowtie            # Job name
#SBATCH -n 8                      # Number of cores
#SBATCH -N 1                      # Ensure that all cores are on one machine
#SBATCH -t 0-6:00                 # Runtime in D-HH:MM (or use minutes)
#SBATCH --mem 8000                # Memory pool in MB for all cores
#SBATCH -o bowtie.out             # File to which standard out will be written
#SBATCH -e bowtie.err             # File to which standard err will be written
```

Start

```
#SBATCH --mail-type=ALL           # Type of email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=rmf0123.com   # Email to which notifications will be sent
```

Required

```
source new-modules.sh; module load bowtie

cd my_output_directory
bowtie -q -p $SLURM_NTASKS -1 A01_R1.pair.fastq.gz -2 A01_R1.pair.fastq.gz
...
... do more processing here...
```

Submitting Jobs - Multicore #2

`omp_test.slurm` file contents:

Required

```
#!/bin/bash
#
#SBATCH -p general                  # Partition
```

Recommended

```
#SBATCH -J omp_test                 # Job name
#SBATCH -n 8                        # Number of cores
#SBATCH -N 1                        # Make sure all cores are on a single node
#SBATCH -t 0-1:00                   # Runtime limit
#SBATCH --mem 8000                  # Memory pool in MB for all cores
#SBATCH -o omp_test.out             # Standard output
#SBATCH -e omp_test.err             # Run-time errors
```

Start

```
#SBATCH --mail-type=ALL             # Type of email notification
#SBATCH --mail-user=rmf@123.com     # Email to which notifications will be sent
```

Required

```
export OMP_NUM_THREADS=$SLURM_NTASKS # Specify number of OMP threads

source new-modules.sh; module load intel
./omp_itest.x
...
... do more processing here...
```

Submitting Jobs - Multinode

`mpi_test.slurm` file contents:

Required

```
#!/bin/bash
#
#SBATCH -p general                  # Partition
```

Recommended

```
#SBATCH -J mpi_test                 # Job name
#SBATCH -n 128                      # Number of cores
#SBATCH -N 2-10                     # # nodes min (-max optional) (-N param optional)
#SBATCH -t 0-1:00                   # Runtime limit
#SBATCH --mem-per-cpu=4000          # Memory in MB per core
#SBATCH -o mpi_test.out             # Standard output
#SBATCH -e mpi_test.err             # Run-time errors
```

Start

```
#SBATCH --mail-type=ALL             # Type of email notification
#SBATCH --mail-user=rmf@123.com     # Email to which notifications will be sent
```

Required

```
source new-modules.sh; module load intel
module load openmpi

mpirun -np $SLURM_NTASKS ./mpi_test.x
...
... do more processing here...
```

Controlling Jobs & Getting Job Info

`scancel` may become your best friend

```
scancel JOBID                                  # specific job
scancel -u bfreeman                            # ALL my jobs
scancel -u bfreeman -J many_blast_jobs         # named jobs
scancel -u bfreeman -p bigmem                  # ALL in partition
```

`squeue` gives info on currently running jobs

```
squeue -u bfreeman                             # jobs for bfreeman
squeue -u bfreeman --states=R | wc -l          # # of Running jobs
```

`sacct` gives current and historical information

```
sacct -u bfreeman                              # jobs for bfreeman
sacct -u bfreeman -p bigmem --starttime=9/1/14 # same+bigmem partition
sacct -j JOBID --format=JobID,JobName,ReqMem,MaxRSS,Elapsed # RAM requested & used!!
```

Check out Common SLURM Commands: http://fasrc.us/easyslurm

SLURM, LSF, SGE, PBS/Torque rosetta stone: http://fasrc.us/move2slurm

Advanced Topics on Odyssey

HARVARD Faculty of Arts and Sciences RESEARCH COMPUTING

Not enough time to cover, but look on our website for...

- *FairShare*:
  - Assigned priority based on past job count and CPU/time usage
  - Same score for all members of the lab
- *Job dependencies*:
  - JobA & B submitted at the same time, but JobB starts when JobA has finished
- *Job arrays*:
  - Large bundle of jobs run individually but handled as one unit
- Pleasantly parallel jobs
- OpenMP (multicore) & MPI (multinode)
- Parallel IO, R, MATLAB, Python

Check out documentation at: http://fasrc.us/fasrcdocs

Example scripts at: http://fasrc.us/slurmutils

Problems, Pitfalls, and Prevention

HARVARD Faculty of Arts and Sciences RESEARCH COMPUTING

This is a shared resource, so everyone has skin in the game. And you can help us and yourself…

- Node and cluster problems are not unusual, esp. as large as system as Odyssey. Report any usual problems: I/O errors, node failures, memory errors, etc.
- Review our Usage & Responsibilities guidelines: http://fasrc.us/hpccustoms
- Review our Common Pitfalls, lest you fall victim: http://fasrc.us/hpcpitfalls

PEND for >48 hrs.

Asking for very large resource requests (cores/memory);very low Fairshare score

Quick run and FAIL...Not including -t parameter

no -t means shortest possible in all partitions == 10 min

Asking for multiple cores but forgetting to specify one node

-n 4 -N 1 is very different from -n 4

Not specifying enough cores

prog1 | prog2 | prog3 > outfile should run with 3 cores

Causing massive disk I/O on home folders/lab disk shares

your work & others on the same filesystem slows to a crawl; simple commands like ls take forever

Getting Help

RC Website & Documentation -- only authoritative source
https://rc.fas.harvard.edu/

Best way to help us to help you? Give us...

Description of problem
Additional info (login/batch? partition? JobIDs?)
Steps to Reproduce (1., 2., 3...)
Actual results
Expected results

OdyBot, for quick-fix problems http://odybot.org/

Submit a ticket on the portal https://portal.rc.fas.harvard.edu/

RC & Informatics Training

HARVARD Faculty of Arts and Sciences RESEARCH COMPUTING

Training opportunities: New & Evolving…

- Office Hours: every Wed 12 - 3 pm @ RC conference room
- Tips@12: Highlighted topics in the first ½ hour of Office Hours
  - Troubleshooting Jobs, iPython Notebooks, Parameter Sweeps, Unix Tricks
- *Intro to Odyssey & RC Services*
- Using HPC Resources Efficiently series (early 2016)
- Guest lectures in courses
- Lab-specific, customized training, including optimizing workflows
- MATLAB workshops (Fall '14, Spring '15)
- Software Carpentry (http://software-carpentry.org/ Fall '14, Summer '15)
- Data Carpentry (http://datacarpentry.org/ Summer '15)

All of the training materials can be found at http://fasrc.us/fasrcmaterials

Take-home Message…

If you can mind these Top 5 items, you'll be an Odyssey rock star!

1. Use the appropriate partition for your job
2. Don't run large numbers of jobs out of home or lab directories; use `/scratch` or `/n/regal` instead
3. Lower your RAM usage and use `--mem` where possible
4. Pass along to your code the number of cores you requested from SLURM (usually $SLURM_NTASKS) if using more then 1 core; and use `-N 1` unless you know what you're doing
5. Ensure your jobs run for at least 5 - 10 minutes and keep job counts in a reasonable range (≤ 1000)

Research Computing

Please talk to your peers, and …
We wish you success in your research!

http://rc.fas.harvard.edu
https://portal.rc.fas.harvard.edu
@fasrc

Harvard Informatics
@harvardifx