

Introduction to the HPC Cluster

Rollins School of Public Health, Computer Club
October 2019

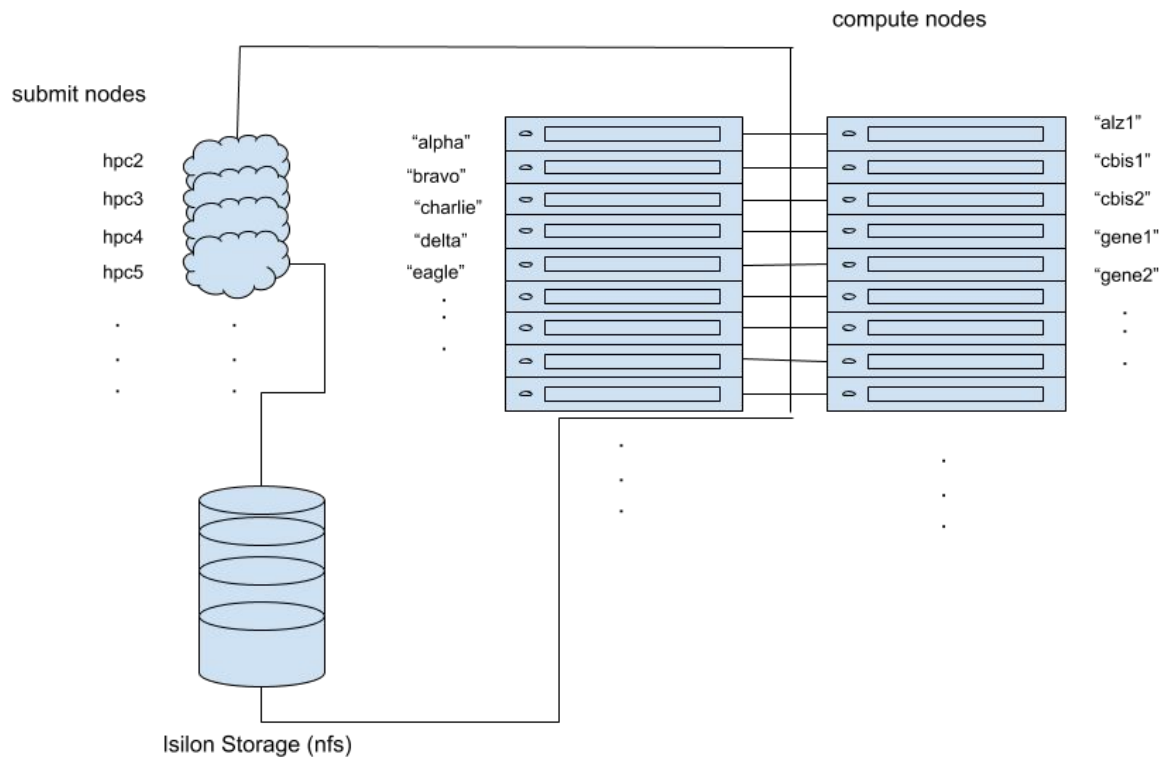


What is the (an) HPC Cluster?

The HPC Cluster Layout

- Compute nodes
- Submit hosts
- Storage service
- Scheduler
- License Manager





Submit hosts and Compute Nodes

Login/Submit hosts:

- Your “login host”. You **ssh**-in to this machine: `ssh <userid>@<submit_host>`
- You most likely will need VPN access to reach these (some campus wired connections are accepted)
- Firewall timeout: 15 minutes
- You copy data onto these systems, which have your home and project directories available (which are mounted everywhere)
- Limited application availability - don't run jobs on these systems
- Named “hpc3”, “hpc4”, “hpc5”, etc.

Compute nodes:

- These are where the jobs are run
- Applications readily available here
- No direct login access
- Many cores, large amounts of RAM
- Physical servers (not VMs)
- Named after fruit (“apple”, “banana”, “cherry”) or research group (“gene1”, “gene2”, etc.) or randomly (“dynareg1”)
- There are 32 compute nodes
- 664 cores -> 1328 threads total
- Running RHEL 7

Preliminaries: Changing your \$SHELL

- A legacy “feature” of central IT at Emory: everyone’s Unix shell is set to ‘ksh’ in LDAP.
- Literally no one uses **ksh** anymore. Default shell on Linux, Mac OS X and even Solaris is **bash** now, and has been for years.
- In process of getting it changed, but until then, all new Emory users have to change it themselves:
<https://mynetid.emory.edu/IDMProv/portal/cn/DefaultContainerPage/WelcometoMyNetID>
- Call 7-7777 (Central IT help line) for assistance.



Logging in to the Cluster

- We use **ssh** (Secure Shell) to log into the submit hosts
- The easiest operating systems: Linux and Mac OS X clients. Use the provided terminal program, and **ssh** is already included.
- Windows: more complicated. You'll need an **ssh** client. There are at least three options:
 - Install Putty. Go to putty.org, download 64-bit .msi file and install. It works on older Windows systems
 - On Windows 10, use Windows Subsystem for Linux. Download Ubuntu from Windows store
 - Install Linux via a virtual machine: e.g., VirtualBox or VMWare
 - Excellent how-to video at www.lynda.com: *Learning Linux Command Line*

Using SSH

- Use command `ssh <username>@<submit_host>.sph.emory.edu`, e.g., “`ssh fred@hpc6.sph.emory.edu”`”
- Accept new key the first time you login
- Best practice: password-less key exchange and `ssh-agent`
- **ssh-copy-id** works well from any Linux/Mac OS X terminal.
- **.ssh/config** can be used to save aliases to further reduce typing.



Connecting to the Cluster: VPN, firewalls, **screen**

- If you connect to the cluster from an off-site location, including Emory Unplugged (Emory's wireless network), you'll need access to the Emory VPN. This is granted by individual request by central IT (*LITS*):
<https://it.emory.edu/security/vpn.html>
- Once logged in, both the VPN and the network firewall settings can "time out".
- As a work-around, use the Unix command **screen**.
- After logging in, simply type 'screen'. It will return a prompt to you, as if nothing happened. Type "<ctrl>-a d", to exit a screen.
- If your connection gets dropped (while in a screen), you can "reattach" to the session via **screen -r**. (Type '<ctrl>-a ?' in **screen**, for command list.)

Unix: a few helpful commands

- **du** : shows disk usage. Ex: **du -hs /home/<netid>** shows how large your home directory is (in gigabytes). (*This command may take a few minutes to complete.*)
- **ls** : list files in current working directory
- **pwd** : print working directory
- **more** : read contents of a text file, with paging
- **chown** : change user or group ownership of a file
- **vim** : a lightweight text file editor (worth learning)
- **man** : read manual pages. Ex. **man chown**
- **scp** : secure copy, part of **ssh**
- There is an entire free course at www.lynda.com:
Learning Linux Command Line

Using the Cluster: Grid Engine q-commands

Work on the cluster is submitted in the form of “jobs”, or programming tasks, that are sent to the compute nodes from the submit nodes. The *q-commands* are Grid Engine commands that perform various job functions:

Submit jobs	qsub, qlogin
-------------	---------------------

Check job status	qstat
------------------	--------------

Delete jobs	qdel
-------------	-------------

Cluster/queue state	qhost, qstat
---------------------	---------------------



Cluster status: **qhost** - show available hosts

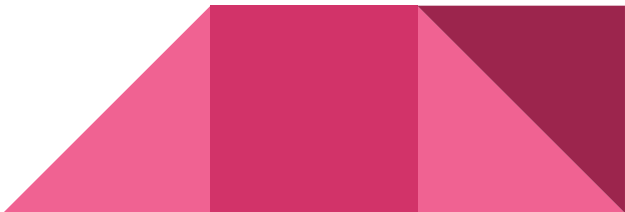
```
khaynes@hpc1 ~->qhost
```

HOSTNAME	ARCH	NCPU	NSOC	NCOR	NTHR	NLOAD	MEMTOT	MEMUSE	SWAPTO	SWAPUS
global	-	-	-	-	-	-	-	-	-	-
alpha	lx-amd64	72	2	36	72	0.04	220.3G	5.8G	4.0G	1.4G
alz1	lx-amd64	56	2	28	56	0.00	251.8G	5.0G	128.0G	595.5M
apple	lx-amd64	40	2	20	40	0.01	251.9G	2.8G	128.0G	2.5G
banana	lx-amd64	40	2	20	40	0.01	251.9G	3.0G	128.0G	2.6G
bravo	lx-amd64	24	2	12	24	0.08	62.9G	5.5G	1024.0M	1023.7M
cbis1	lx-amd64	40	2	20	40	0.03	125.7G	2.4G	4.0G	1.1G
cbis2	lx-amd64	40	2	20	40	0.00	125.7G	2.5G	4.0G	1.2G
charlie	lx-amd64	24	2	12	24	0.04	62.9G	1.4G	13.6G	1.2G
cherry	lx-amd64	32	2	16	32	0.06	125.9G	3.4G	128.0G	654.4M
condor	lx-amd64	40	2	20	40	0.09	125.8G	59.7G	96.0G	176.7M
delta	lx-amd64	24	2	12	24	0.13	62.9G	6.1G	13.6G	1.1G
dynareg1	lx-amd64	48	2	24	48	1.01	251.6G	14.6G	4.0G	264.2M
eagle	lx-amd64	8	2	4	8	0.17	31.4G	1.1G	64.0G	924.7M
echo	lx-amd64	24	2	12	24	0.13	62.9G	13.5G	64.0G	1.5G
foxtrot	lx-amd64	32	2	16	32	0.07	125.9G	2.9G	128.0G	1.1G
gene1	lx-amd64	40	2	20	40	0.02	125.8G	3.4G	128.0G	1.0G
gene10	lx-amd64	72	2	36	72	0.00	251.8G	4.3G	128.0G	485.8M

Cluster status: qstat -f

```
khaynes@hpc1 ~->qstat -f -q long.q
queue name      qtype resv/used/tot. np_load  arch      states
-----
long.q@alpha    BP    0/2/72        0.04    lx-amd64
long.q@bravo    BP    0/2/24        0.08    lx-amd64
long.q@charlie  BP    0/1/24        0.04    lx-amd64
long.q@delta    BP    0/2/24        0.13    lx-amd64
long.q@echo     BP    0/3/24        0.13    lx-amd64
long.q@foxtrot  BP    0/1/32        0.07    lx-amd64
long.q@hotel    BP    0/1/32        0.05    lx-amd64
long.q@india.sph.emory.edu BP    0/1/48        0.02    lx-amd64
long.q@juliett.sph.emory.edu BP    0/2/48        0.04    lx-amd64
long.q@kilo.sph.emory.edu BP    0/2/48        0.04    lx-amd64
```

Submitting jobs: **qsub**

- Batch job submission is done via **qsub**
 - **qsub <program_name>** submits a job running <program_name> with default attributes, after which the job will appear in the output of **qstat**
 - Many attributes can be specified, such as run queue, input, output, job name, memory usage, etc.
 - Ex. **qsub -cwd -i /data/example.in -o /results/example.out -N testrun1 -q long.q -l mem_free=2GB R_wrapper_script.sh**
 - That's a lot of typing! Most people put those attributes in a job submission script.
- 

File Edit View Search Terminal Help

```
#!/bin/bash
```

```
# set the name
```

```
#$ -N testrun1
```

```
# Set the input file
```

```
#$ -i /data/example.in
```

```
# Set the output, both stdout and stderr
```

```
#$ -o /results/example.out -o /results/example.err
```

```
# other submit options
```

```
#$ -l mem_free=2GB
```

```
#$ -q long.q
```

```
some_R_program.R
```

```
# END
```

A sample job submission script.

```

File Edit View Search Terminal Help
guest@hpc5~> cat GE_TEST/bin/simple.sh
#!/bin/bash

# This is a simple example of a SGE batch script

# request Bourne shell as shell for job
#$ -S /bin/bash

# print date and time
date
# Sleep for 20 seconds
sleep 20
# print date and time again
date

guest@hpc5~> qsub GE_TEST/bin/simple.sh
Your job 1606884 ("simple.sh") has been submitted
guest@hpc5 ~>qstat

```

job-ID	prior	name	user	state	submit/start at	queue	jclass	slots	ja-task-ID
1606884	0.55476	simple.sh	guest	r	02/07/2019 15:34:02	short.q@gene10.sph.emory.edu		1	

```

guest@hpc5~> ls -l simple.sh.*
-rw-r--r-- 1 guest RSPH 0 Feb  7 15:34 simple.sh.e1606884
-rw-r--r-- 1 guest RSPH 29 Feb  7 15:34 simple.sh.o1606884

guest@hpc5~> more simple.sh.o1606884
Thu Feb  7 15:34:02 EST 2019
Thu Feb  7 15:34:22 EST 2019

```

A **qsub** workflow example.

Checking on Jobs: **qstat** and **qacct**

- **qstat** (without arguments) shows the status all of your currently queued jobs
- Jobs usually have status of **r** (“running”), **qw** (“queue wait” or pending), or **Eqw** (in and “error” state)
- Use **qacct -j <job_id>** to get detailed information about any job.



Killing jobs: **qdel**

- Use **qdel** *<job_id>* to kill a submitted job



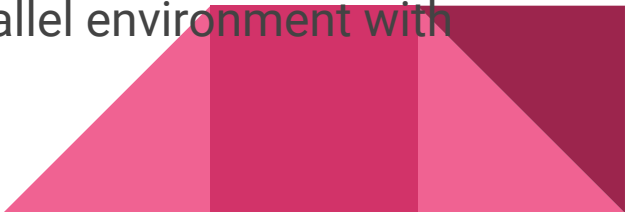
```

File Edit View Search Terminal Help
user@hpc5> ./sleeper.sh 30 1
Here I am. Sleeping now at: Thu Feb 7 16:06:48 EST 2019
Now it is: Thu Feb 7 16:07:18 EST 2019
user@hpc5> qsub sleeper.sh 30 1
Your job 1606886 ("Sleeper") has been submitted
user@hpc5> qsub sleeper.sh 30 1
Your job 1606887 ("Sleeper") has been submitted
user@hpc5> qsub sleeper.sh 30 1
Your job 1606888 ("Sleeper") has been submitted
user@hpc5> qstat
job-ID      prior    name         user          state submit/start at   queue                                jclass      slots ja-task-ID
-----
    1606886  0.55476  Sleeper      user           r      02/07/2019 16:07:31 short.q@gene9.sph.emory.edu          1
    1606887  0.55476  Sleeper      user           r      02/07/2019 16:07:35 short.q@gene10.sph.emory.edu         1
    1606888  0.55476  Sleeper      user           r      02/07/2019 16:07:35 short.q@gene3                        1
user@hpc5> qdel 1606887
user has registered the job 1606887 for deletion
user@hpc5> qstat
job-ID      prior    name         user          state submit/start at   queue                                jclass      slots ja-task-ID
-----
    1606886  0.55476  Sleeper      user           r      02/07/2019 16:07:31 short.q@gene9.sph.emory.edu          1
    1606887  0.55476  Sleeper      user           dr     02/07/2019 16:07:35 short.q@gene10.sph.emory.edu         1
    1606888  0.55476  Sleeper      user           r      02/07/2019 16:07:35 short.q@gene3                        1
user@hpc5> qstat
job-ID      prior    name         user          state submit/start at   queue                                jclass      slots ja-task-ID
-----
    1606886  0.55476  Sleeper      user           r      02/07/2019 16:07:31 short.q@gene9.sph.emory.edu          1
    1606888  0.55476  Sleeper      user           r      02/07/2019 16:07:35 short.q@gene3                        1
khaynes@hpc1 ~>

```

A **qdel** workflow example.

Cluster queues

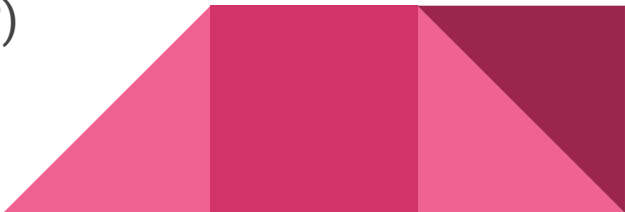
- *qlogin.q* - interactive sessions; dedicated 64 cores; timeout after 24 hours
 - *long.q* - general use; no time limit; 14 jobs/user max.
 - *short.q* - general use subordinate queue; 72 hour limit; 240 cores; limited to 40 jobs/user
 - Restricted queues: *fruit.q*, *gene.q*, *sunlab.q*, *cbis.q*, *alz.q*, *mh.q*
 - Restricted queues have priority on owned systems, and have no per user job limit or run-time limit
 - All queues have corresponding parallel queue (*long*, *short*, *gene*, etc.)
 - Ex. **qsub -pe long 4 <program_name>** requests a parallel environment with four processors
- 

A Quick Primer on Parallel Computing

- Some computations lend themselves very easily to parallelism - called “Embarrassingly Parallel”, distinguished from “concurrent” or “distributed” computation. Distributed-parallel (IPC) programming is really difficult.
- Platforms are called “tightly coupled” (e.g., a multiprocessor system, or low-latency infiniband) or “loosely coupled” (e.g. our HPC Cluster w/ethernet)
- Parallelism happens at different levels on the Cluster:
 - Grid Engine parallelizes across cluster
 - Unix systems run processes concurrently internally
 - Code can execute multiple threads or fork copies



Parallel code on the Cluster

- In order for processes to be properly load-balanced across the cluster, we need to let GE know of our multi-threaded or processing intent. Parallel jobs need to be submitted to the *parallel queues*. The scheduler can then take into consideration the load of submitted parallel jobs on compute nodes.
 - Our cluster has limited ability to handle proper parallel jobs. In general, if your code is embarrassingly parallel, you should “serialize” (i.e., not use internal parallelization) your code and submit them, as arrays if necessary ('-t')
 - You reserve the amount of job slots (threads?, cores?) you need for each job using the '-pe' flag, up to a maximum of 12. (Why 12?)
 - Parallel jobs have to “fit” on a compute node.
- 

Understanding Parallelism: Restaurant Analogy

Imagine the HPC cluster is a restaurant, and the Grid Engine scheduler is the very accommodating maitre d'. Compute nodes are tables, and submitted jobs are dining patrons wishing to be seated for dinner. (Assume we are considering one queue at this time...)

- If a single patron ask for a seat, the maitre d' will seat them at any table with an available seat, but trying to distribute the load amongst the wait staff.
- A party of two or more (a parallel job!) requires a table with two or more seats free. The larger the party, the harder it is to find an available table.
- If you tell the maitre d' you are by yourself, but you are really seating five of your friends also, your table will be overcrowded ("oversubscribed").

Available Software

- C/C++, Fortran compilers
- Python*
- R**
- MATLAB
- SAS
- Openmpi
- Others, including user-installed



Storage on Cluster

- Each account is granted a home directory in /home/<NetID>, which is mounted across all nodes of the cluster (i.e., software installed there is available everywhere). (*“Isilon” storage, maintained by LITS*)
- Initial quota is set to 16 GB. Quota can be increased once, no questions asked. Users are responsible for deleting/off-loading and compressing files.
- For large data installations, request a project directory. There is a cost associated with project directories, depending on type. (*The latest pricing change has not been announced at time of slide creation.*)



More info

- For technical help with the cluster (including new account requests), send email to “help@sph.emory.edu”
- Excellent how-to videos are available to Emory community (free!) via www.lynda.com. Click on sign in, then enter “emory.edu” for the domain, and you will be redirected to Emory for sign in. Search for “Learning Linux Command Line” for video instructions on how to install Windows Subsystem for Linux or Linux in a virtual machine (plus, basic Linux usage).
- The old HPC Cluster documentation is still here:
<https://intranet.sph.emory.edu/services/it/environment/cluster/index.html>

(It is in need of updating!)

