USC HPC Cluster

What is HPC?

- HPC is USC's Center for High-Performance Computing.
- HPC resources are free to any USC researcher.
- Requires Unix

Why do we sometimes call an HPC platform a cluster?

- Because it is basically a giant building on campus that houses thousands of computers. This building is connected to the internet and made available exclusively to the USC research community for running computationally heavy tasks that are beyond the capabilities of standard computers. We think of this platform as a cluster of compute nodes where each node consists of several cpus or gpus.

What makes it so great?

- Through HPC you can request compute resources according to the needs of your application.
- You can access HPC from anywhere in the world.
- HPC has support for storing protected health data.
- Easily shareable to anyone on the USC HPC network which is great for collaboration between labs on campus.

HPC Accounts

Accounts are project-based

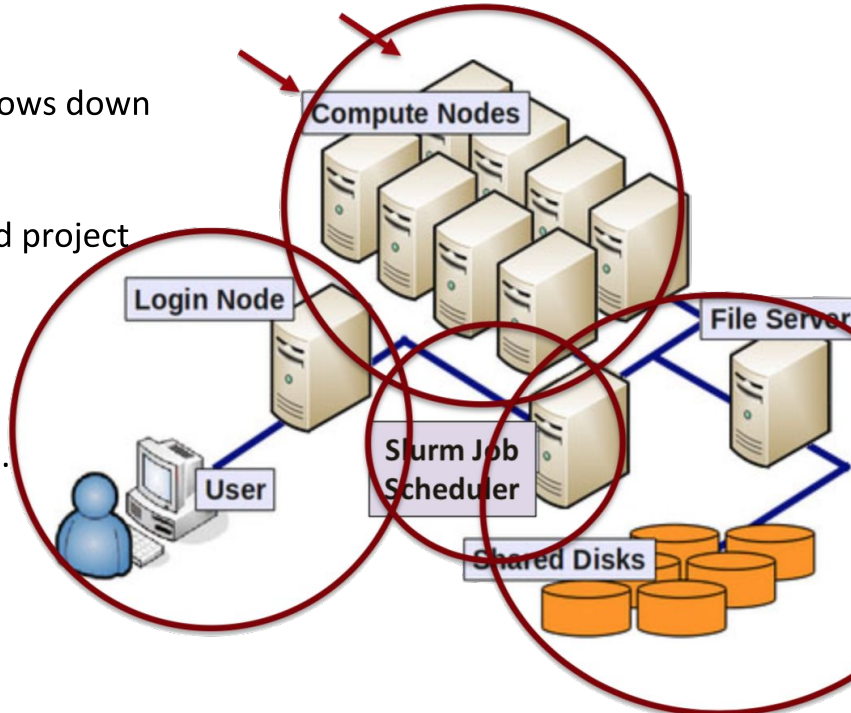
- Project “PI”s “own” accounts
- A PI can add members or be the sole member
- Members can belong to multiple projects
- Project members **share** quotas
- PI can request increases on **project website**
  - *https://hpc-web.usc.edu/project*

** If you do not yet have an account, please consult with your PI. The rest of this presentation assumes you have been registered for an HPC account with USC.*

HPC Architecture

All nodes run a linux based operating system

- **Login Node**
  - 1gb storage
  - Shared for all users
  - Minimal compute resources
  - **Do not** compute or submit jobs here. Slows down login node for everyone!
- **Compute Nodes**
  - Standard 5 Terabytes of Data per shared project
  - Dedicated CPUs
  - Compute here
  - Submit Jobs here
  - Long term/permanent storage
  - For storing code and software packages.
- **Scratch**
  - 10 Terabytes per user
  - Temporary storage lasts 6 weeks.
  - For storing large datasets and results.
  - Submit jobs here.



Connecting to the Cluster

```
~ :> ssh username@hpc-login3.usc.edu
username@hpc-login3.usc.edu's password: your_password

Duo two-factor login for username

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-1232
 2. Phone call to XXX-XXX-1232
 3. SMS passcodes to XXX-XXX-1232

Passcode or option (1-3): 1
Success. Logging you in…

-bash-4.2$
```

* Duo authentication is required so please install the app and connect your account to it.

Navigating Your HPC Account

Login node: (This is only a login node. Switch to scratch or project directories for storage or computation)

```
-bash-4.2$ pwd
/home/rcf-15/username
-bash-4.2$ ls
```

Compute/Project node: (software packages here)

```
-bash-4.2$ cd /home/rcf-proj/sm3/username
-bash-4.2$ ls
anaconda2 genomics.USC.HPC R_packages
```

scratch node: (Temporary 6 week storage: 10TB per person)

```
-bash-4.2$ cd /scratch/username
-bash-4.2$ ls
STAGING_README RNA_seq_data
```

Transfer Files To/From The Cluster

Use the transfer node for transferring large files to or from the cluster: hpc-transfer.usc.edu

scp [from] [to]

Local to cluster:

```
~:> scp filename username@hpc-transfer.usc.edu:/home/rcf-15/username/path
```

Cluster to local:

```
~:> scp username@hpc-transfer.usc.edu:/home/rcf-15/username/filename .
```

Note:

1) **.** represents the current working directory on your local machine
2) To move a folder, first compress into a zip-file. Then unzip once transferred.

```
zip -r foldername.zip foldername
```

Available Software Packages

The USC HPC comes preinstalled with many commonly used software packages.

1) Navigate to available packages directory and list them.

```
~:> cd /usr/usc
~:> ls
```

2) Identify the package and version you wish to use. Source the setup.sh file. We will use python version 3.7.4.

```
~:> cd python
~:> cd 3.7.4
~:> source setup.sh
```

3) Python 3.7.4 is now the current active version of python on your hpc account. When you logout, the default will return to hpc's default version.

Custom software packages

- Any changes you make to the libraries available within these packages will be erased when you logout and will not persist in new sessions you access.

- If you spend a lot of time fine tuning the packages in your environment, then it is better to download the software from the internet and store it in your project directory. This way you can fine tune your own packages and their state will persist each time you log in.

- To ensure that you use the version of software that you downloaded and not the default version available on hpc, make sure to specify the full path to your software package when using it. In the example below, you installed your own version of anaconda3 in your project directory. Specify the full path to run the python version that came with your anaconda3 installation on your unique python script called my_code.py

```
~:> /home/rcf-proj/sm3/username/anaconda3/bin/conda my_code.py
```

Conda Environments

Batch Jobs (1)

Make a .sh file that invokes the scripts you wish to execute. At the top of your .sh file, insert the line: **#!/bin/bash**

1)
   ~:> vi my_job.sh
2)
   Type: i
3)
   ~:> #!/bin/bash
4)
   Type: echo "This echo command will run after I submit this my_job.sh
   file to the cluster and sufficient compute resources are available"

   sleep 3000

5)
   Type: esc :wq

Batch Jobs (2)

Run the following command in the cluster to submit your job.

```
~:> sbatch --job-name=A --mem-per-cpu=B --time=C my_job.sh
```

A = name of your choosing for the job Ie. Big_Data

B = memory per cpu required for job. Ie. 2G

C = computation time required for your job to complete. Ie. 1:00:00

Note that the more resources you request, the longer it will take for your job to begin as there are limited resources available at any given time.

Monitor/Cancel Job

Show my jobs on cluster:

```
~:> myqueue /my jobs
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
3176	quick	Big_Data	username	PD	0:00	1	(Resources)

Cancel job

```
~:> scancel job_id
```

Batch Jobs (3)

Try it out

1) Submit the job using the script we created earlier.

```
~:> sbatch --job-name=BigData --mem-per-cpu=1GB --time=1:00:00 my_job.sh
```

2) When the job completes, look at the slurm-7332691.out file to see the expected console text output and job stats.

```
~:> cat slurm-7332691.out
```

Interactive Session

1) To request an interactive session, use the command salloc. In the example below, four processors with two gigabytes per processor are requested for one hour.

```
~:> salloc --time=1:00:00 --mem-per-cpu=2GB
```

2) Test out your .sh file before submitting job to catch syntax errors early on.

```
~:> ./my_job.sh
```

Useful Cluster Commands

Show disk usage of account:

```
~:> myquota
```

Percent of CPU that is being used compared to what was requested:

```
~:> htop
```

Check how busy cluster is:

```
~:> sinfo --partition main
```

Cluster Resources

1) **Unless your PI has purchased dedicated nodes, your jobs will be submitted to the nodes made available to the general usc hpc community. Thus your wait times for job execution will depend on the community's usage of the cluster at any given time.**

2) **For community nodes, the max number of jobs that can run simultaneously per user is 10 with up to 99 compute nodes. Max 1000 jobs in the queue at any given time.**

3) **For more information on limits and more complex usage of the cluster please visit,** https://hpcc.usc.edu/gettingstarted/