Joint High Performance Computing Exchange (JHPCE) Cluster Overview

2023-01-30

Biostatistics Journal/Computing Club

JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

http://www.jhpce.jhu.edu/

Schedule

- **Introductions – who are we?**
- Terminology
- Logging in and account setup
- Basics of running programs on the cluster
- Examples

Who we are:

- JHPCE – Joint High Performance Computing Exchange
  - Co-Director: Brian Caffo
  - Co-Director: Mark Miller
  - Systems Engineer: Jiong Yang
  - Systems Engineer: Jeffrey Tunison
  - Application Developer: Adi Gherman
- Beyond this class, when you have questions:
  - http://www.jhpce.jhu.edu
    - lots of good FAQ info
    - these slides (full version)
  - bitsupport@lists.jh.edu
    - System issues (password resets/disk space)
    - Monitored by the 5 people above
  - bithelp@lists.jh.edu
    - Application issues (R/SAS/perl...)
    - Monitored by dozens of application SMEs
    - All volunteers
  - Others in your lab
  - Web Search

Quick Survey

- Who has experience using Unix command line, Terminal, Shell?
- Who has experience using other clusters?

Schedule

- Introductions – who are we, who are you?
- **Terminology**
- Logging in and account setup
- Basics of running programs on the cluster
- Examples

What is a cluster?

- A collection of many powerful computers that can be shared with many users.

Why would you use a cluster?

- You need resources not available on your local laptop
- You need to run a program (job) that will run for a very long time
- You need to run a job that can make use of parallel computing

Types of parallelism

1. Embarrassingly (obviously) parallel …
http://en.wikipedia.org/wiki/Embarrassingly_parallel

2. Multi-core (or multi-threaded) – a single job using multiple CPU cores via program threads on a single machine (cluster node). Also see discussion of fine-grained vs coarse-grained parallelism at
http://en.wikipedia.org/wiki/Parallel_computing

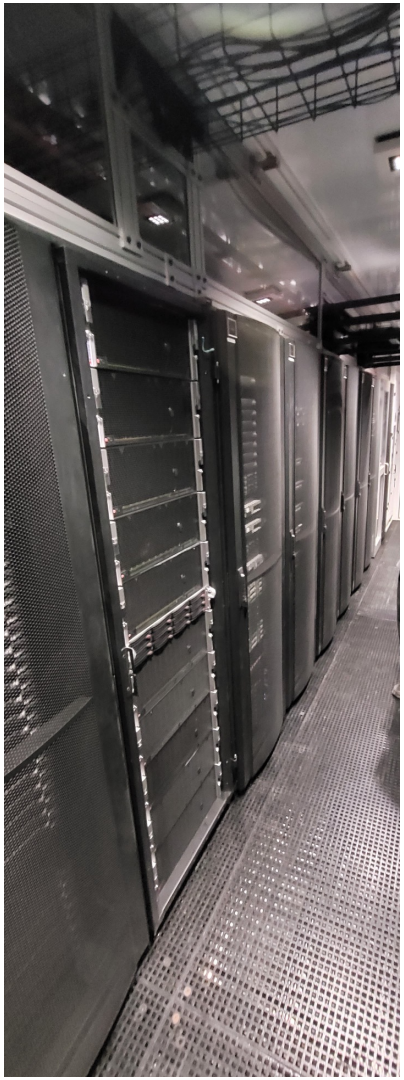
3. Many CPU cores on many nodes using a Message Passing Interface (MPI) environment. Not used much on the JHPCE Cluster.

Node (Computer) Components

- Each computer is called a "**Node**"
- Each node, just like a desktop/laptop has:
  - RAM
  - Intel/AMD CPUs
  - Disk space

- Unlike desktop/laptop systems, nodes do not make use of a display/mouse – they are used from a command line interface known as a "**shell**".

The JHPCE cluster components



- Joint High Performance Computing Exchange (JHPCE)
- Fee for service – nodes purchased by various PIs.
- Located at Bayview Colocation Facility

Hardware:

- 12 Racks of equipment – 5 compute, 6 storage, 1 infra.
- 76 Nodes – 72 compute, 2 transfer, 2 login
  - 4000 Cores - Nodes have 2 - 4 CPUs, 24 to 128 cores per node
  - 30 TB of RAM - Nodes ranges from 128 GB to 2048 GB RAM.
  - Range in size from a large pizza box to a long shoe box
- 14,000 TB of Disk space – 11,500 TB of project storage, 2000 TB of backup, 500TB of scratch/home/other storage.
  - Storage is network attached-available to all nodes of the cluster.

Software:

- Based on Centos 7 Linux
- Used for a wide range of Biostatistics – gene sequence analysis, population simulations, medical treatment.
- Common applications: R, SAS, Stata, perl, python ...

How do programs get run on the compute nodes?

- We use a product called “Sun Grid Engine” (**SGE**) that schedules programs (jobs). Other clusters may use other schedulers such as Slurm or Torque.
- History:
  - 1990s - Developed by Gridware
  - 2000 - Gridware purchased by Sun Microsystems
  - 2001 - Sun makes source code open source
  - 2010 - Oracle buys Sun and discontinues support for SGE
  - 2013 - Univa picks up support for Sun customers
- Jobs are assigned to slots as they become available and meet the resource requirement of the job
- Jobs are submitted to **queues**
- The cluster nodes can also be used interactively.

Schedule

- Introductions – who are we, who are you?
- Terminology
- **Logging in**
- Basics of running programs on the cluster
- Examples

How do you use the cluster?

- The JHPCE cluster is accessed using SSH (Secure SHell), so you will need an ssh client.
- Use **`ssh`** to login to "**`jhpce01.jhsph.edu`**"

- For Mac and Linux users, you can use **`ssh`** from Terminal Window.

- For MS Windows users, you need to install an ssh client – such as MobaXterm (recommended) or Cygwin, Putty and Winscp, or WSL :

http://mobaxterm.mobatek.net/

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

http://www.cygwin.com

http://winscp.net

Quick note about graphical programs

To run graphical programs on the JHPCE cluster, you will need to have an X11 server running on your laptop.

- For Microsoft Windows, Mobaxterm has an X server built into it.
- For Windows, if you are using Putty, you will need to install an X server such as Cygwin.

- For Macs:

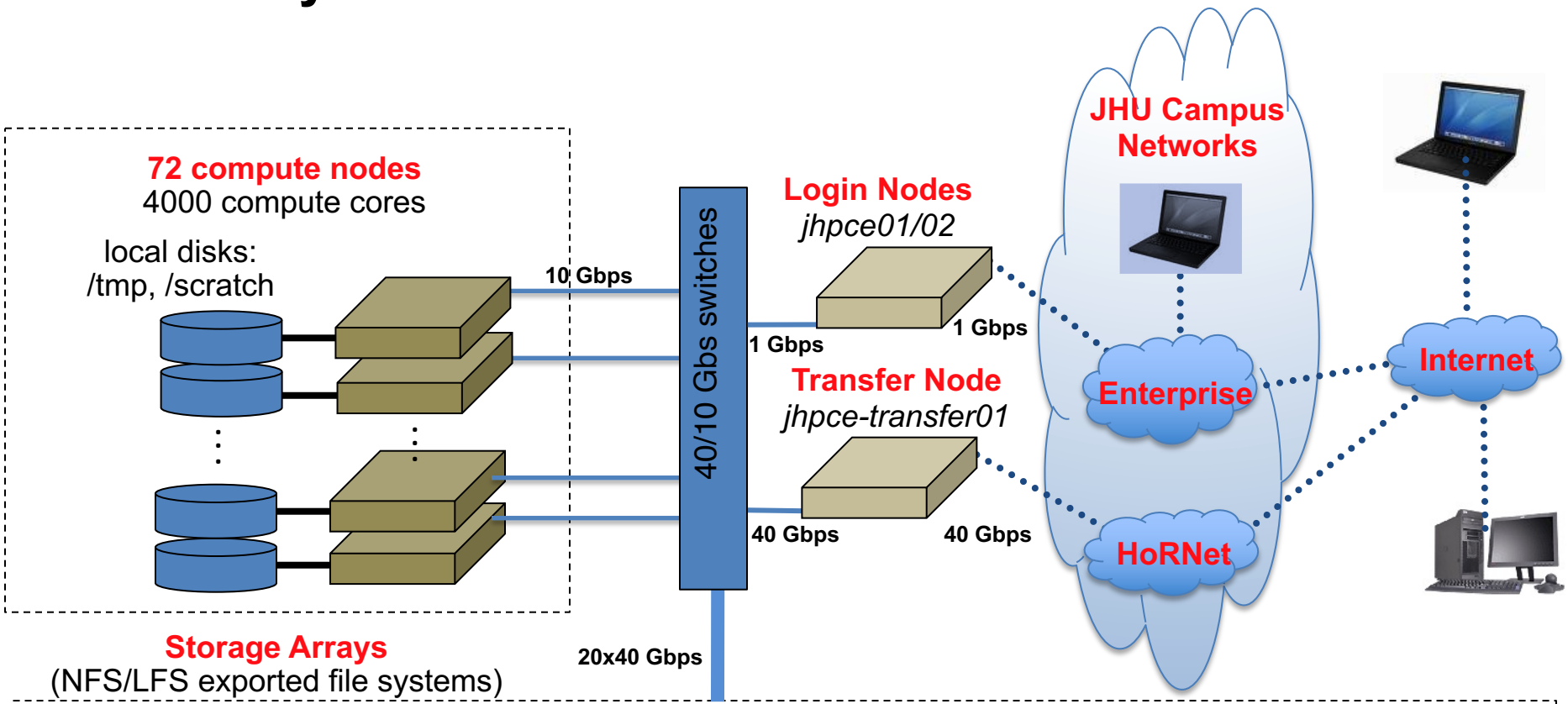
1) You need to have the Xquartz program installed on your laptop. This software is a free download from Apple, and does require you to reboot your laptop http://xquartz.macosforge.org/landing/
2) You need to add the "-X" option to your ssh command:

```
$ ssh -X mmill116@jhpce01.jhsph.edu
```

- For Linux laptops, you should already have an X11 server install. You will though need to add the –X option to ssh:

```
$ ssh -X mmill116@jhpce01.jhsph.edu
```

JHPCE System Architecture



Example 1 – Logging in

- Bring up Terminal
- Run: `ssh -X USERID@jhpce01.jhsph.edu`
- 2 Factor authentication
  - When you type your password, the cursor will not move. This is a security mechanism so that someone looking over your shoulder won't be able to see your password.
  - The first time you login, you will use the Initial Verification Code and Initial Password sent to you.
  - Google Authenticator will be set up after you login the first time
  - Going forward you'll use Google Authenticator when prompted for "Verification Code"
- Shell prompt

Lab 1 - Logging In - cont

- Note "Emergency Scratch Codes"

- 100 GB limit on home directory. Home directories are backed up, but other storage areas are probably not.

- 1 TB of intermediate “fastscratch” storage for temporary storage (less than 30 days)

https://jhpce.jhu.edu/knowledge-base/fastscratch-space-on-jhpce
https://jhpce.jhu.edu/policies/current-storage-offerings

- (optional) Setup ssh keys

https://jhpce.jhu.edu/knowledge-base/authentication/ssh-key-setup
https://jhpce.jhu.edu/knowledge-base/mobaxterm-configuration

General Linux/Unix Commands

Navigating Unix:

- ls
- ls -l
- ls -al
- pwd
- cd
- . and ..

Commands in example script:

- date
- echo
- hostname
- sleep
- control-C

Looking at files:

- more/less

Changing files with editors:

- nano
- vi/emacs

Good resources for learning Linux:
http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v3.1.1.html
https://www.digitalocean.com/community/tutorials/a-linux-command-line-primer
https://files.fosswire.com/2007/08/fwunixref.pdf

Unix/Linux Command Reference

FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:
./configure
make
make install
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.

Schedule

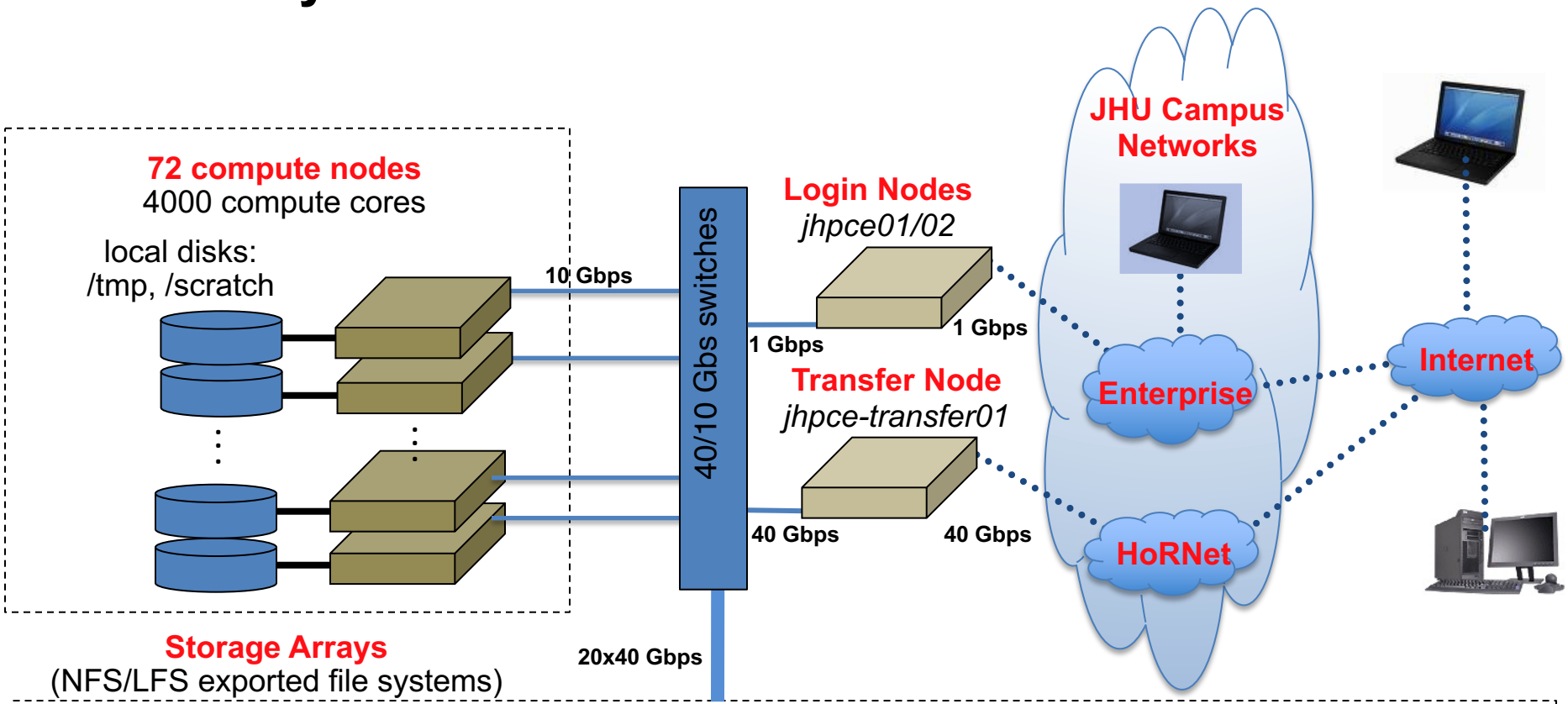
- Introductions – who are we, who are you?
- Terminology
- Logging in and account setup
- **Basics of running programs on the cluster**
- Examples

Submitting jobs to the queue with Sun Grid Engine (SGE)

- **`qsub`** – allows you to submit a batch job to the cluster
- **`qrsh`** – allows you establish an interactive session

- **`qstat`** – allows you to see the status of your jobs

JHPCE System Architecture



Lab 2 - Using the cluster

Example 2a – submitting a batch job

```
cd class-scripts
qsub -cwd script1
qstat
```

examine results files

Example 2b – using an interactive session

```
qrsh
```

Note current directory

Modules

Modules for R, SAS, Mathematica . . .

- module list
- module avail
- module avail python
- module avail conda_R
- module load
- module unload

Lab 3 - Running R

Running R on the cluster:

- In `$HOME/class-scripts/R-demo`, note 2 files – Script file and R file

- Submit Script file
  - `qsub -cwd plot1.sh`

- Run R commands interactively
  - `qrsh`
  - `module load conda_R`
  - `R`

Requesting additional RAM

- By default you are given 5GB of RAM to work in
- You can request more RAM by using the **`mem_free`** and **`h_vmem`** options to **`qrsh`.**
  - **`mem_free`** – is used to set the amount of memory your job will need. SGE will place your job on a node that has at least **`mem_free`** RAM available.
  - **`h_vmem`** – is used to set a high water mark for your job. If your job uses more than **`h_vmem`** RAM, your job will be killed. This is typically set to be the same as mem_free.

- Examples:

```
qsub -l mem_free=10G,h_vmem=10G job1.sh
```

or

```
qrsh -l mem_free=10G,h_vmem=10G
```

Lab 4

Running RStudio

- X Windows Setup
  - For Windows, Mobaxterm has and X server built into it
  - For Mac, you need to have the Xquartz program installed (which requires a reboot), and you need to add the "-X" option to ssh:

```
$ ssh -X mmill116@jhpce01.jhsph.edu
```

- Start up Rstudio

```
$ qrsh -l mem_free=10G,h_vmem=10G
$ module load conda_R
$ module load rstudio
$ rstudio
```

Lab 5 - Running Python

- The default system version of python is version 2

```
[jhpce01 /users/mmill116]$ qrsh -now n
Last login: Wed Jan 25 12:51:14 2023 from jhpce01.cm.cluster
[compute-076 /users/mmill116]$ which python
/usr/bin/python
[compute-076 /users/mmill116]$ python -V
Python 2.7.5
```

- To use Python 3, you'll need to load the module for the version you want.

```
[compute-076 /users/mmill116]$ module avail python


---------------------- /jhpce/shared/jhpce/modulefiles/core ----------------------
   python/2.7.9    python/3.7.3 (D)    python/3.9.10
   python/3.6.9    python/3.8.3


. . .

[compute-076 /users/mmill116]$ module load python
[compute-076 /users/mmill116]$ which python
/jhpce/shared/jhpce/core/python/3.7.3/bin/python
[compute-076 /users/mmill116]$ python -V
Python 3.7.3
```

Lab 5 (cont) - Running a Python program

- Use a text editor like "nano" to create a program then run it via python.

```
[compute-076 /users/mmill116]$ nano program1.py
[compute-076 /users/mmill116]$ cat program1.py
print("Hello World")
[compute-076 /users/mmill116]$ python program1.py
Hello World
```

Lab 6

Running Stata and SAS

- Stata example:

Batch:

```
$ cd $HOME/class-scripts/stata-demo
$ ls
$ more stata-demo1.sh
$ more stata-demo1.do
$ qsub stata-demo1.sh
```

Interactive:

```
$ qrsh
$ stata
```

or

```
$ xstata
```

- SAS example:

Batch:

```
$ cd $HOME/class-scripts/SAS-demo
$ ls
$ more sas-demo1.sh
$ more class-info.sas
$ qsub sas-demo1.sh
```

Interactive:

```
$ qrsh -l sas
$ sas
```

Note – The name of the program and script need not be the same, but it is good practice to keep them the same when possible.

Note – Extensions sometimes are meaningful. SAS doesn't care, but Stata programs need to have ".do" as the extension. It is good practice for human readability.

BETA TESTING JUPYTER LAB

We are starting to test using Jupyter Lab on the JHPCE cluster.

https://jhpce-app02.jhsph.edu

- Only accessible on campus or via VPN
- Login with your JHED ID and password
- Go to "Access JHPCE Apps" and select "Jupyter Lab"
- After 5 minutes you'll receive an email with a link for your Jupyter Lab session
- You can ignore the "Your Connection is not Private" message. In Chrome, you may need to type "thisisunsafe"

It's best to rely on the command line access to Python.

Lab 5 - Transferring files to the cluster

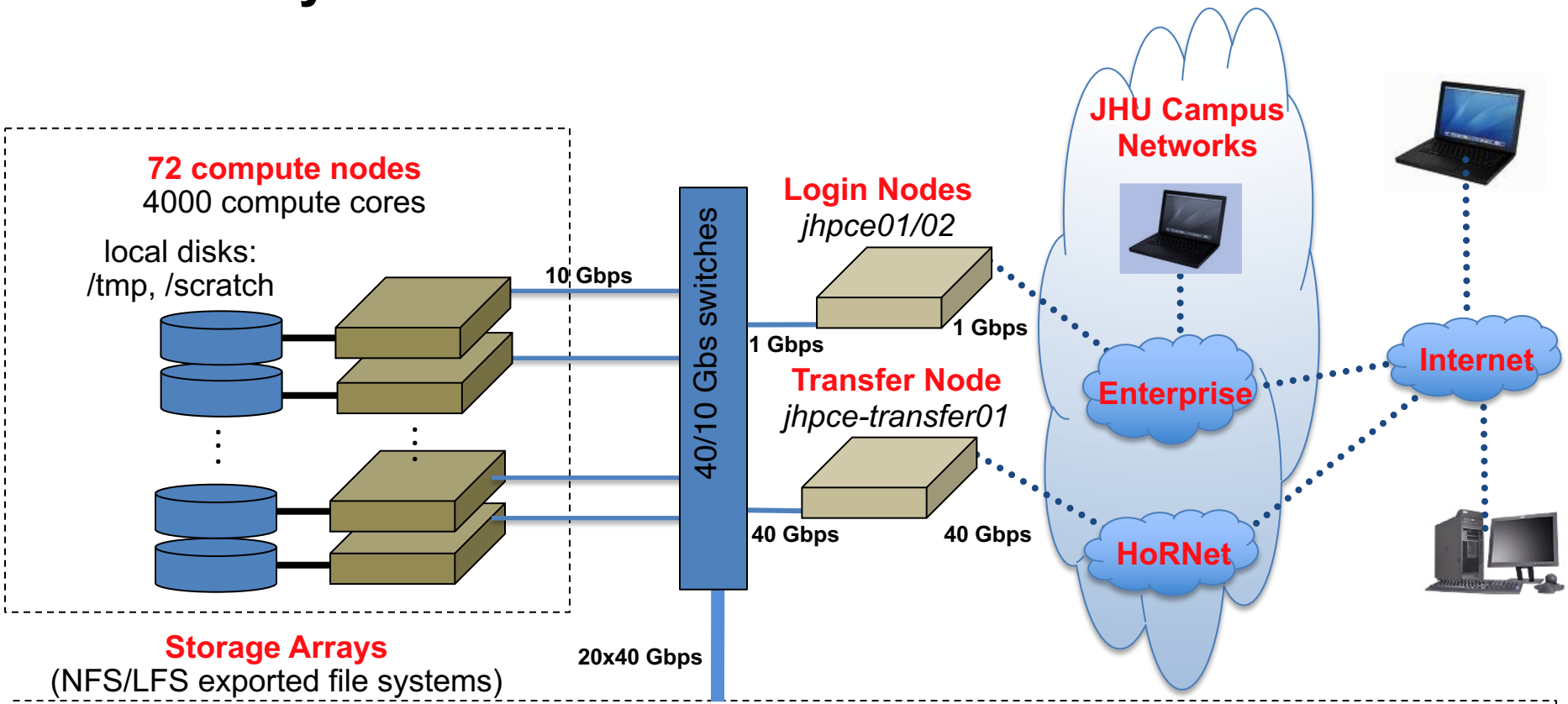
- Transfer results back

```
$ sftp mmill116@jhpce-transfer01.jhsph.edu
```

- Within sftp, you can use "ls" and "cd" to navigate.
- You can also use:
  - "get" to get a file from the cluster
  - "put" to put a file on the cluster

- Or use a graphical sftp program like WinSCP, Filezilla, Globus, mobaxterm…

JHPCE System Architecture



Thanks for attending! Questions?

“How many jobs can I submit?”

Users frequently submit 1000s of jobs on the cluster, but we do impose a limit of 10,000 **submitted** jobs per user. If you anticipate the need to submit more than 1000 jobs, please email us at bitsupport@lists.jhu.edu as there are mechanisms and strategies for efficiently handling 1000s of jobs.

More importantly, we also impose a per-user limit on the number of cores and RAM for **running** jobs on the shared queue. Currently, the limit is set to **200 cores per user and 1024GB of RAM per user.**

So, if a user submits 1000 single-core jobs, the first 200 will begin immediately (assuming the cluster has 200 cores available on the shared queue), and the rest will remain in the ‘qw’ state until the first 200 jobs start to finish. As jobs complete, the cluster will start running 'qw' jobs, and keep the number of running jobs at 200.

Similarly, if a user's job requests 100GB of RAM to run, the user would only be able to run 10 jobs before hitting their 1024 GB limit, and subsequent jobs would remain in 'qw' state until running jobs completed.

The maximum number of slots per user may be temporarily increased by submitting a request to bitsupport@lists.jhu.edu. We will increase the limit, depending on the availability of cluster resources. There are also dedicated queues for stakeholders which may have custom configurations and limits.