

Joint High Performance Computing Exchange (JHPCE) Cluster Orientation



JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

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

Schedule

- **Introductions – who are we, who are you?**
- Terminology
- Logging in and account setup
- Basics of running programs on the cluster
- Details – limits and resources
- 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
 - bitsupport@lists.johnshopkins.edu
 - System issues (password resets/disk space)
 - Monitored by the 5 people above
 - bithelp@lists.johnshopkins.edu
 - Application issues (R/SAS/perl...)
 - Monitored by dozens of application SMEs
 - All volunteers
 - Others in your lab
 - Web Search



Who are you?

- Name
- Department
- How do you plan on using the cluster? What data or applications will you be using?
- Will you be accessing the cluster from a Mac or a Windows system?
- What is your experience with Unix?
- Any 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
- Details – limits and resources
- Examples



Clusters – what and why?

What is a cluster?

- A collection of many powerful computers (**nodes**) that can be shared with many users.

Why would you use a cluster?

- Need resources not available on your local laptop
- Need to run a program (**job**) that will run for a long time
- Need to run a job that can make use of multiple computers simultaneously (**parallel computing**)
- Want to queue multiple jobs so they run ASAP without needing your attention to launch them (using a **job scheduler**)



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 connected display/keyboard/mouse – they are used over a network, often from a **command line interface (CLI)** known as a “**shell**”.
- **Graphical user interface (GUI)** programs can be run, displaying on your desktop/laptop.



The JHPCE cluster components



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

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 thin 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 cluster nodes.

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, python, Jupyter ...



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.



ORACLE®



Schedule

- Introductions – who are we, who are you?
- Terminology
- **Logging in and account setup**
- Basics of running programs on the cluster
- Details – limits and resources
- 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 a Terminal application window.
- For MS Windows users, you need to install an ssh client – such as MobaXterm (strongly recommended) or Cygwin, Putty and Winscp :



<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 mmill1116@jhpce01.jhsph.edu
```



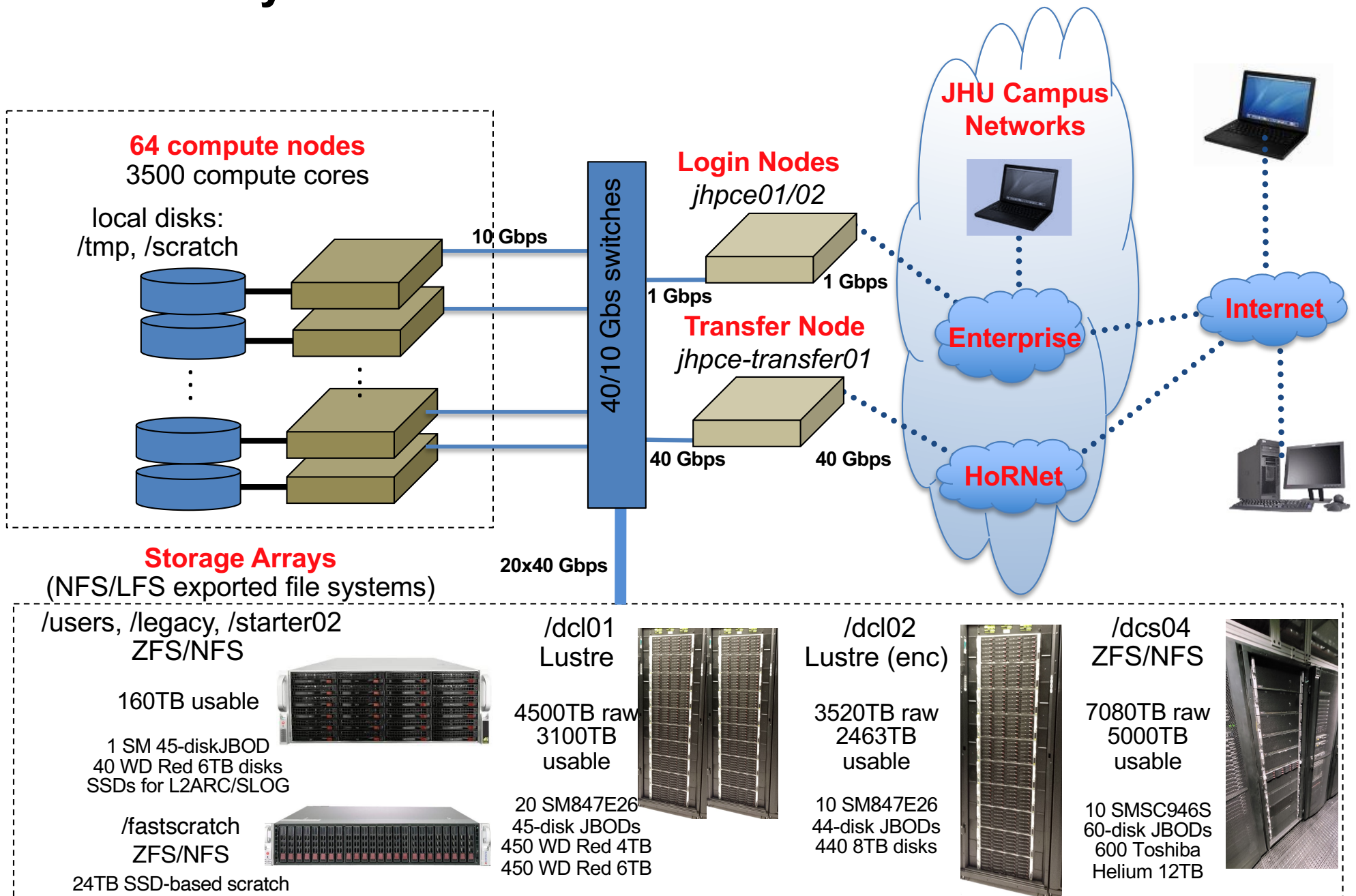
- For Linux laptops, you should already have an X11 server install. You will though need to add the `-X` option to ssh:

```
$ ssh -X mmill1116@jhpce01.jhsph.edu
```



JHPCE System Architecture

Workstations, Desktops, Laptops



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

- For Mac/Linux laptop Users:




- Bring up a Terminal
- Run: **ssh -X USERID@jhpce01.jhsph.edu**



- Login with the initial Verification Code and Password that were sent to you

- For PC Users:



- Launch MobaXterm
- click on the “Sessions” icon  in the upper left corner
- On the “Session settings” screen, click on “SSH”
- Enter “jhpce01.jhsph.edu” as the “Remote host”. Click on the “Specify username” checkbox, and enter your JHPCE username in the next field. Then click the “OK” button.
- Login with the initial Verification Code and Password that were sent to you.
- If dialog windows pop up, click "Cancel" when prompted for another Verification Code, or click "No" or “Do not ask this again” when prompted to save your password.



Lab 1 - Logging In - cont

- Change your password with the “`kpasswd`” command. You will be prompted for your current (initial) password, and then prompted for a new password.
- Setup 2 factor authentication
 - <http://jhpce.jhu.edu/knowledge-base/how-to/2-factor-authentication/>
 - 1) On your smartphone, bring up the "Google Authenticator" app
 - 2) On the JHPCE cluster, run "auth_util"
 - 3) In "auth_util", use option "5" to display the QR code (you may need to resize your ssh window - "view->terminal unzoom" in MobaXterm)
 - 4) Scan the QR code with the Google Authenticator app
 - 5) Next, in "auth_util" use option 2 to display your scratch codes – record these
 - 6) In "auth_util", use option "6" to exit from "auth_util"
- Log out of the cluster by typing "exit".
- Log into the cluster again with 2 factor authentication



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](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>



File Commands	System Info
<p>ls - directory listing ls -al - formatted listing with hidden files cd dir - change directory to <i>dir</i> cd - change to home pwd - show current directory mkdir dir - create a directory <i>dir</i> rm file - delete <i>file</i> rm -r dir - delete directory <i>dir</i> rm -f file - force remove <i>file</i> rm -rf dir - force remove directory <i>dir</i> * cp file1 file2 - copy <i>file1</i> to <i>file2</i> cp -r dir1 dir2 - copy <i>dir1</i> to <i>dir2</i>; create <i>dir2</i> if it doesn't exist mv file1 file2 - rename or move <i>file1</i> to <i>file2</i> if <i>file2</i> is an existing directory, moves <i>file1</i> into directory <i>file2</i> ln -s file link - create symbolic link <i>link</i> to <i>file</i> touch file - create or update <i>file</i> cat > file - places standard input into <i>file</i> more file - output the contents of <i>file</i> head file - output the first 10 lines of <i>file</i> tail file - output the last 10 lines of <i>file</i> tail -f file - output the contents of <i>file</i> as it grows, starting with the last 10 lines</p>	<p>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 <i>user</i> uname -a - show kernel information cat /proc/cpuinfo - cpu information cat /proc/meminfo - memory information man command - show the manual for <i>command</i> df - show disk usage du - show directory space usage free - show memory and swap usage whereis app - show possible locations of <i>app</i> which app - show which <i>app</i> will be run by default</p>
Process Management	Compression
<p>ps - display your currently active processes top - display all running processes kill pid - kill process id <i>pid</i> killall proc - kill all processes named <i>proc</i> * 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 <i>n</i> to the foreground</p>	<p>tar cf file.tar files - create a tar named <i>file.tar</i> containing <i>files</i> tar xf file.tar - extract the files from <i>file.tar</i> 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 <i>file</i> and renames it to <i>file.gz</i> gzip -d file.gz - decompresses <i>file.gz</i> back to <i>file</i></p>
File Permissions	Network
<p>chmod octal file - change the permissions of <i>file</i> to <i>octal</i>, which can be found separately for user, group, and world by adding:</p> <ul style="list-style-type: none"> ● 4 - read (r) ● 2 - write (w) ● 1 - execute (x) <p>Examples: chmod 777 - read, write, execute for all chmod 755 - rwx for owner, rx for group and world For more options, see man chmod.</p>	<p>ping host - ping <i>host</i> and output results whois domain - get whois information for <i>domain</i> dig domain - get DNS information for <i>domain</i> dig -x host - reverse lookup <i>host</i> wget file - download <i>file</i> wget -c file - continue a stopped download</p>
SSH	Installation
<p>ssh user@host - connect to <i>host</i> as <i>user</i> ssh -p port user@host - connect to <i>host</i> on port <i>port</i> as <i>user</i> ssh-copy-id user@host - add your key to <i>host</i> for <i>user</i> to enable a keyed or passwordless login</p>	<p>Install from source: ./configure make make install dpkg -i pkg.deb - install a package (Debian) rpm -Uvh pkg.rpm - install a package (RPM)</p>
Searching	Shortcuts
<p>grep pattern files - search for <i>pattern</i> in <i>files</i> grep -r pattern dir - search recursively for <i>pattern</i> in <i>dir</i> command grep pattern - search for <i>pattern</i> in the output of <i>command</i> locate file - find all instances of <i>file</i></p>	<p>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</p>
	<p>* use with extreme caution.</p>



Schedule

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



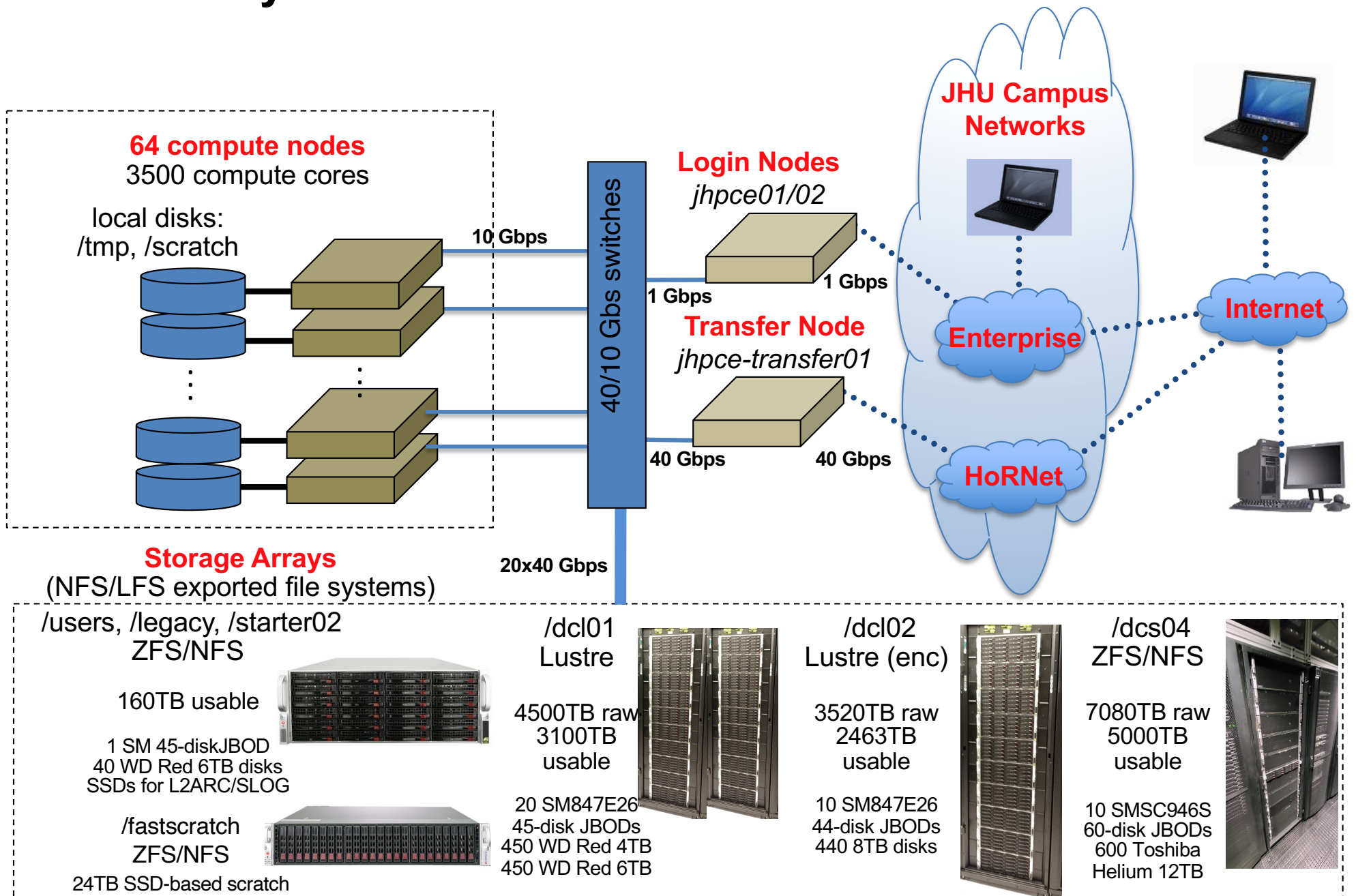
Submitting jobs to the queue with Sun Grid Engine (SGE)

- `qsub` – allows you to submit a batch job to the cluster
- `qssh` – allows you establish an interactive session
- `qstat` – allows you to see the status of your jobs



JHPCE System Architecture

Workstations, Desktops, Laptops



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

```
qssh
```

Note current directory



Modules

Modules for R, SAS, Mathematica . . .

- module list
- module avail
- module load
- module unload



Never run a job on the login node!

- Always use “**qsub**” or **qssh**” to make use of the compute nodes
- Jobs that are found running on the login node may be killed at will
- If you are going to be compiling programs, do so on a compute node via **qssh**.
- Even something as simple as copying large files should be done via **qssh**. The compute nodes have 10Gbps connections to the storage systems and jhpce01 only has a 1Gbps connection.



Other useful SGE commands

qstat – shows information about running jobs you are running

```
qstat
```

```
qstat -u \*
```

qacct – shows information about completed jobs

```
qacct -j JOBID
```

qhost – shows information about the nodes

```
qhost
```

```
qpics -q shared
```

qdel – deletes your job

```
qdel JOBID
```



Schedule

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



A few options for “qsub”

- To run the job in the current working directory (where qsub was executed) rather than the default (home directory)

```
$ qsub -cwd
```

- To send standard output (error) stream to a different file or directory

```
-o path/filename
```

```
-e path/filename
```

- To receive notification via email when your job completes

```
-m e -M john@jhu.edu
```

This is a great option for a handful of long running jobs. This is a horrible option for 1000s of jobs, and has caused users to have their email accounts suspended.



A few options for “qrsh”

- By default, qrsh will timeout after 5 seconds if it can't find a compute node to use. To cause qrsh to wait indefinitely for a connection, you can add:

-now no



File size limitations

- There is a default file size limitation of 10GB for all jobs.
- If you are going to be creating files larger than 10GB you need to use the “-l h_fsize” option.
- The “-l” option is used to set "limits" for `qsub/qrsh` and will be used quite a bit in the next several slides.

- Example:

```
$ qsub -l h_fsize=50G myscript.sh
```



Embedding options in your qsub script

- You can supply options to qsub in 2 ways:

- On the command line

```
qsub -cwd -l h_fsize=100G -m e -M john@jhu.edu script1.sh
```

- Setting them up in your batch job script. Lines that start with “#\$” are interpreted as options to **qsub**

```
$ head script1-resource-request.sh
```

```
#$ -l h_fsize=100G
```

```
#$ -cwd
```

```
#$ -m e
```

```
#$ -M john@jhu.edu
```

```
$ qsub script1-resource-request.sh
```



Requesting resources in qsub or qrsh

- By default, when you submit a job, or run qrsh, you are allotted 5GB of RAM and 1 core for your job.
- These default settings can be adjusted by modifying the `.sge_request` file in your home directory.



Requesting additional RAM

- You can request more RAM by setting `mem_free` and `h_vmem`.
 - `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
```



Estimating RAM usage

- No easy formula until you've actually run something
- A good place to start is the size of the files you will be reading in
- If you are going to be running many of the same type of job, run one job, and use "`qacct -j JOBID`" to look at "`maxvmem`".



Requesting additional cores

To request multiple cores, the "-pe local N" option (where N is the number of cores) needs to be supplied to the qsub or qrsh command. For example:

- A job that will use 4 cores:

```
qsub -cwd -pe local 4 myscript.sh
```

- A qrsh session that will use 6 cores:

```
qrsh -pe local 6
```

IMPORTANT NOTE: The `mem_free` and `h_vmem` RAM limits are per-core values, so the total RAM requested needs to be divided by the number of cores.

Examples:

– A job which will use 10 cores and need 120GB of RAM:

```
qsub -cwd -pe local 10 -l mem_free=12G,h_vmem=12G myscript2.sh
```



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.



“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 1,000 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 1,000 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.



Schedule

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



Lab 3



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`



Lab 4

Transferring files to the cluster

- Transfer results back

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

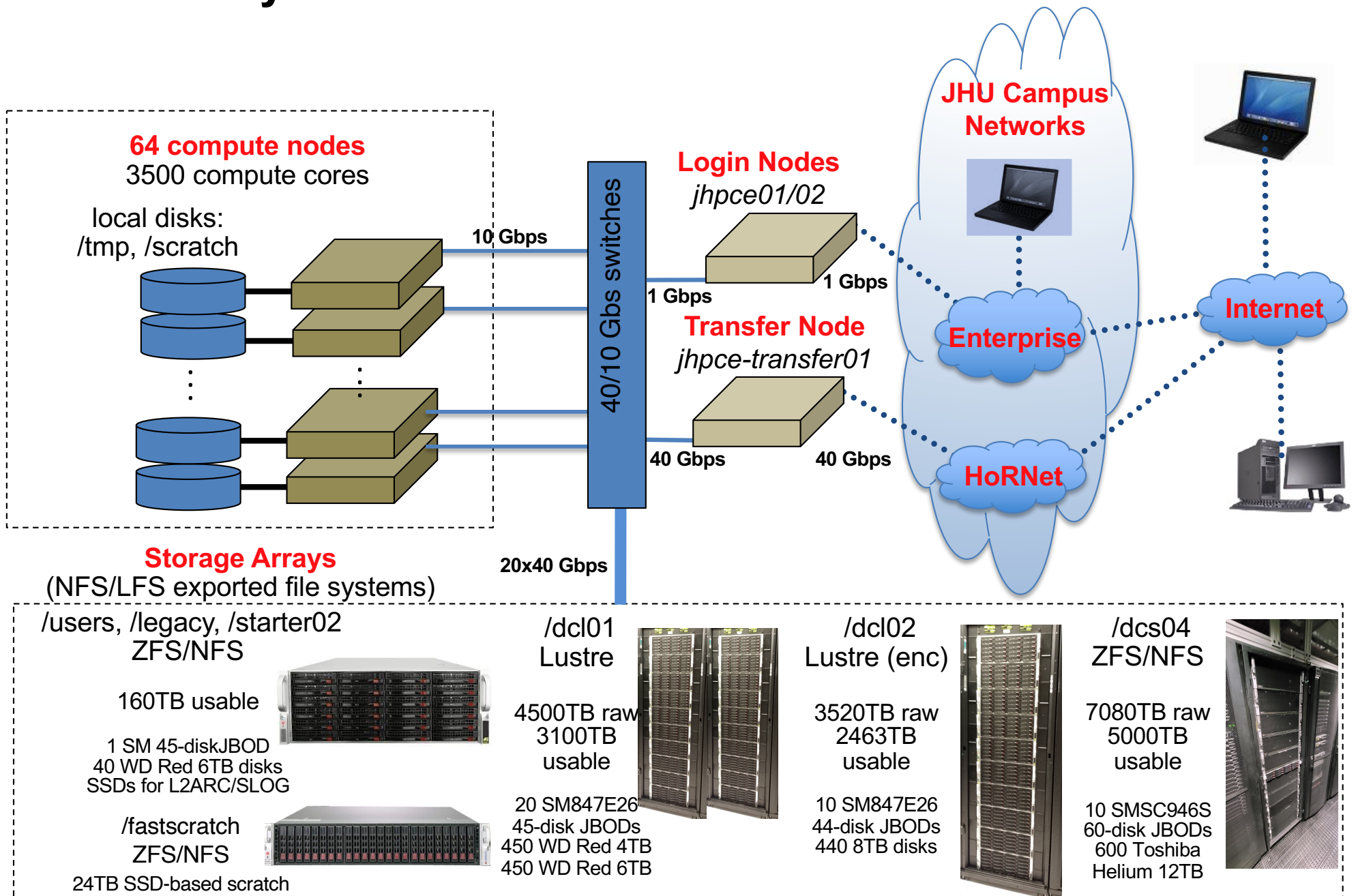
Or use WinSCP, Filezilla, Globus, MobaXterm...

- Look at resulting PDF file using viewer on laptop
- Look at pdf from example 4 via `xpdf`



JHPCE System Architecture

Workstations, Desktops, Laptops



Lab 5

Running RStudio

- X Windows Setup

- For Windows, MobaXterm has an 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 mmill1116@jhpce01.jhsph.edu
```

- Start up Rstudio

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



Other queues

- “shared” queue – default queue
- “download” queues
 - **qrsh -l rnet**
 - (`jhpce-transfer01.jhsph.edu`)
 - be sure to increase your `h_fsize` to be larger than the size of the largest file you’ll be transferring to or from the cluster, e.g. **qrsh -l rnet -l h_fsize=50G**
- “math” queue
 - **qrsh -l math**
- “sas” queue
 - **qrsh -l sas**
- “gpu” queue
 - **qrsh -l gpu**
- “dedicated” queues - Queues that are for PIs who have purchased nodes on the cluster.



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:

```
$ qssh
$ 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:

```
$ qssh -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.



Summary

- Review
 - Get familiar with Linux
 - Use ssh to connect to JHPCE cluster
 - Use qsub and qrsh to submit jobs
 - Never run jobs on the login nodes
 - Helpful resources
 - <http://www.jhpce.jhu.edu/>
 - bitsupport@lists.johnshopkins.edu - System issues
 - bithelp@lists.johnshopkins.edu - Application issues
- What to do next
 - Make note of your Google Authenticator scratch codes (option 2 in "auth_util")
 - Set up ssh keys if you will be accessing the cluster frequently
<https://jhpce.jhu.edu/knowledge-base/authentication/login/>
 - Play nice with others – this is a shared community-supported system.



Thanks for attending! Questions?

