

Joint High Performance Computing Exchange (JHPCE) Cluster Orientation



JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

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

Version: 20240518

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



Who we are – Getting help:

- Beyond this class, when you have questions:
 - <http://jhpce.jhu.edu> – JHPCE Web Site
 - Lots of good FAQ info
 - Searchable
 - You can download a copy of these slides
 - <https://jhpce-app02.jhpce.jhu.edu> – JHPCE App Portal
 - User self service (password resets)
 - Portal to Web-based Apps (Rstudio, Jupyter Labs, Visual Studio)
 - bitsupport@lists.johnshopkins.edu – System Support Email
 - System issues (password resets/disk space)
 - Monitored by the 5 people above
 - bithelp@lists.johnshopkins.edu – App Support EMAIL
 - Application issues (R/SAS/perl...)
 - Monitored by dozens of application subject matter experts
 - All volunteers
 - Others in your lab
 - Web Search – google for your error message



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:

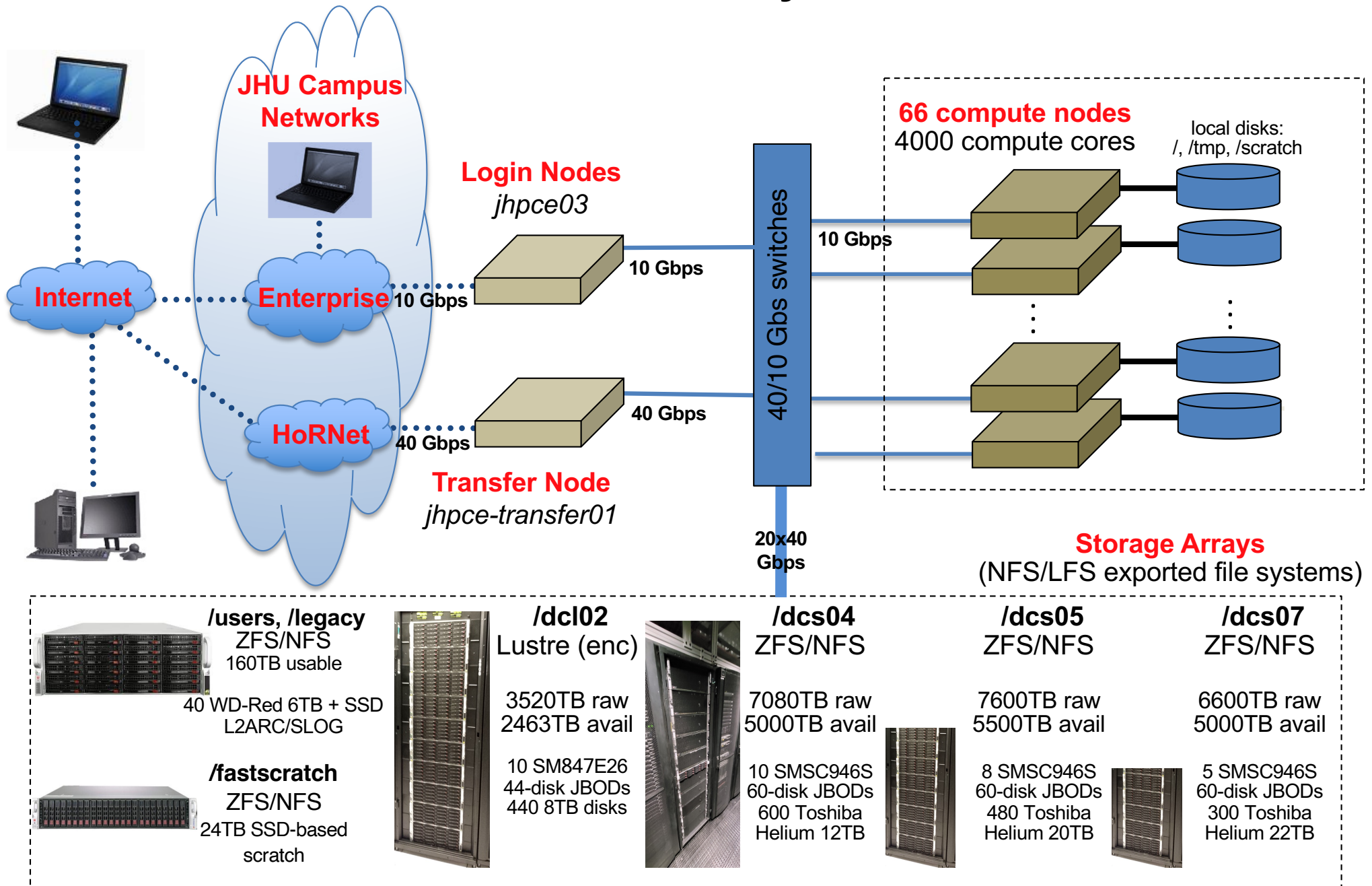
- 12 Racks of equipment – 5 compute, 6 storage, 1 infra.
- 70 Nodes – 66 compute, 2 transfer, 2 login
 - 4000 Cores - Nodes have 2 - 4 CPUs, 24 to 128 cores per node
 - 30 TB of RAM - Nodes ranges from 128 GB to 2048 GB RAM.
 - Range in size from a large pizza box to a long thin shoe box
- 21,000 TB of Disk space – 18,000 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 Rocky Linux 9
- Used for a wide range of Biostatistics – gene sequence analysis, population simulations, medical treatment.
- Common applications: R, SAS, Stata, python, Jupyter ...

Workstations, Desktops, Laptops

JHPCE System Architecture



SLURM Terminology

- **SLURM:** (Simple Linux Utility for Resource Management)
The scheduler for the cluster. Assigns jobs to nodes.
- **Job:** A program running under SLURM's control
- **Node:** One of the server that makes up the cluster.
- **Partition:** The queue that is used to schedule jobs to run on the nodes.
 - **shared** – The default partition
 - **sas** – The partition for running SAS
 - **gpu** – The partition to access
 - **transfer** – The partition for accessing the
 - **Various PI Partitions** – Special partitions for PIs that purchase nodes
- **CPUs:** Refers to number of cores used by a job, not physical CPUs



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 “`jhpce03.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 mm111116@jhpce03.jhsph.edu
```

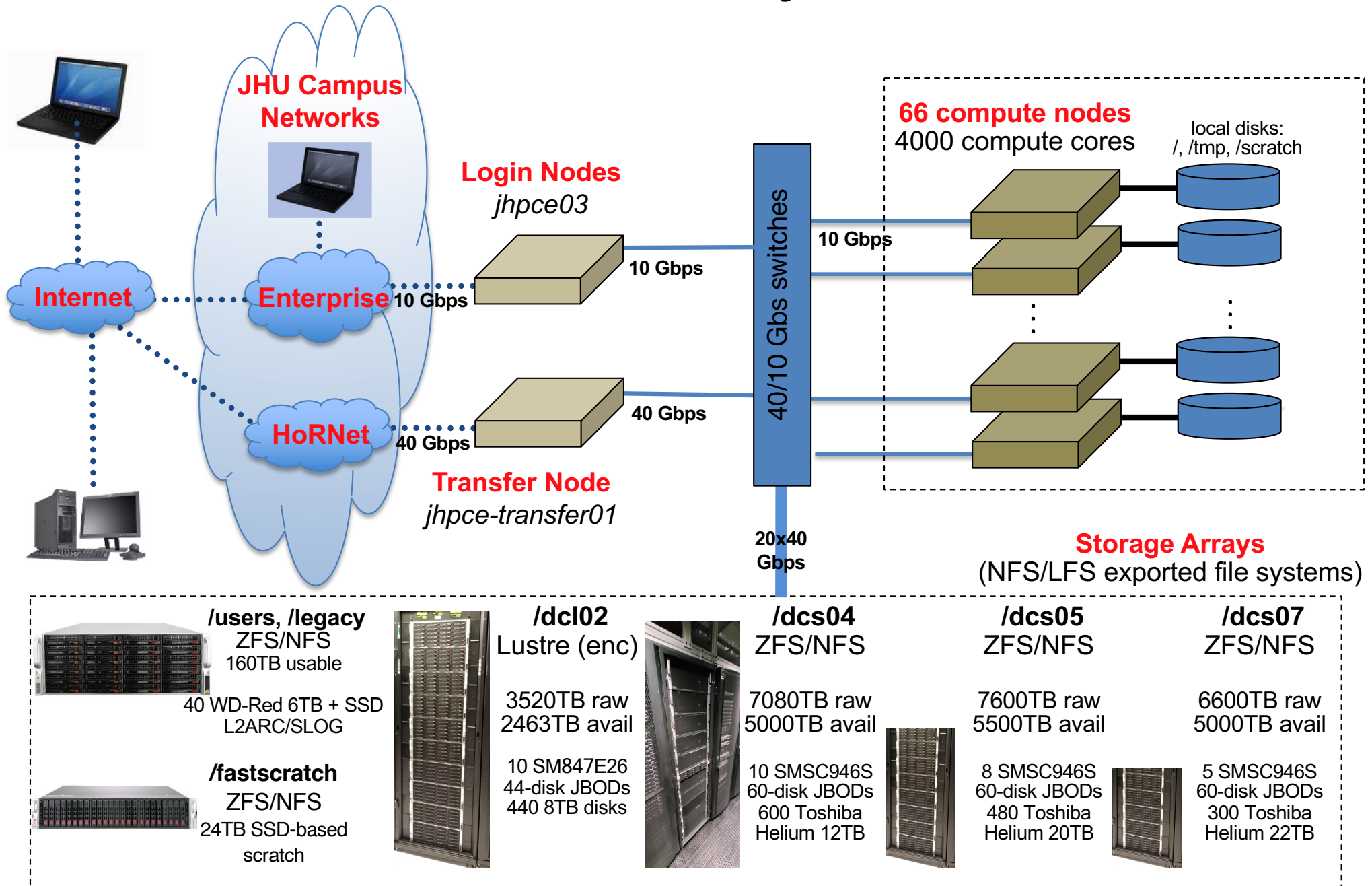


- For Linux laptops, you should already have an X11 server install. You will though need to add the -X option to ssh:

```
$ ssh -X mm111116@jhpce03.jhsph.edu
```


Workstations, Desktops, Laptops

JHPCE System Architecture



Example 1 – Logging in



- Bring up Terminal
- Run: `ssh -X USERID@jhpce03.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@jhpce03.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 “jhpce03.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
 - 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. You should see a 6 digit number with a "jhpce" title in the Google Authenticator app.

(steps continue on next slide)



Lab 1 - Logging In (cont)

- 5) Next, in "auth_util" use option 2 to display your Scratch Codes

You should record these Scratch Codes in a file for now, but print them out or store them in an encrypted file on your laptop. Don't keep them in an unencrypted file on your laptop.

Normally when logging in you will use the 6 digit number from the Google Authenticator app on your phone when prompted for "Verification Code:" during the login process. But if you don't have access to your phone, you can use one of these Scratch Codes. Each code is good for one login, so if you have these printed on a sheet of paper, you would scratch it off of the list once you use it.

The most frequent time we see these needed is when someone either breaks or trades in their phone, and they don't have Google Authenticator set up on their old phone. You would need to use one of these Scratch Codes to login to the cluster, and follow steps 1-4 above to set up the Google Authenticator app on your new phone.

- 6) In "auth_util", use option "6" to exit from "auth_util"

- Log out of the cluster by typing "exit".
- Log back into the cluster again using your own Password and the Google Authenticator code when prompted for "Verification Code"



Lab 1 - Logging In – other comments

- All users have a home directory, `/users/USERID`, that only the user has access to by default.
- In Linux, your home directory can be referred to by `~` or **\$HOME**
- You can use your home directory for storing programs, application, data....
- All home directories have a 100 GB limit.
- Home directories are backed up, but other storage areas are probably not.

- All users also have access to 1 TB of “fastscratch” storage for temporary data storage (less than 30 days)

<https://jhpcce.jhu.edu/storage/fastscratch/>

- (optional) Setup ssh keys

<https://jhpcce.jhu.edu/access/ssh/#ssh-keys>

<https://jhpcce.jhu.edu/access/mobaxterm/#optional-setting-up-ssh-keys-in-mobaxterm>



General Linux/Unix Commands



Navigating Unix:

- **ls**
- **ls -l**
- **ls -al**
- **pwd**
- **cd**
- **.** and **..**
- **~** and **\$HOME**

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
ls - directory listing ls -al - formatted listing with hidden files cd <i>dir</i> - change directory to <i>dir</i> cd - change to home pwd - show current directory mkdir <i>dir</i> - create a directory <i>dir</i> rm <i>file</i> - delete <i>file</i> rm -r <i>dir</i> - delete directory <i>dir</i> rm -f <i>file</i> - force remove <i>file</i> rm -rf <i>dir</i> - force remove directory <i>dir</i> * cp <i>file1 file2</i> - copy <i>file1</i> to <i>file2</i> cp -r <i>dir1 dir2</i> - copy <i>dir1</i> to <i>dir2</i> ; create <i>dir2</i> if it doesn't exist mv <i>file1 file2</i> - 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 <i>file link</i> - create symbolic link <i>link</i> to <i>file</i> touch <i>file</i> - create or update <i>file</i> cat > <i>file</i> - places standard input into <i>file</i> more <i>file</i> - output the contents of <i>file</i> head <i>file</i> - output the first 10 lines of <i>file</i> tail <i>file</i> - output the last 10 lines of <i>file</i> tail -f <i>file</i> - output the contents of <i>file</i> as it grows, starting with the last 10 lines	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 <i>user</i> - display information about <i>user</i> uname -a - show kernel information cat /proc/cpuinfo - cpu information cat /proc/meminfo - memory information man <i>command</i> - show the manual for <i>command</i> df - show disk usage du - show directory space usage free - show memory and swap usage whereis <i>app</i> - show possible locations of <i>app</i> which <i>app</i> - show which <i>app</i> will be run by default
Process Management	Compression
ps - display your currently active processes top - display all running processes kill <i>pid</i> - kill process id <i>pid</i> killall <i>proc</i> - 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 <i>n</i> - brings job <i>n</i> to the foreground	tar cf <i>file.tar files</i> - create a tar named <i>file.tar</i> containing <i>files</i> tar xf <i>file.tar</i> - extract the files from <i>file.tar</i> tar czf <i>file.tar.gz files</i> - create a tar with Gzip compression tar xzf <i>file.tar.gz</i> - extract a tar using Gzip tar cjf <i>file.tar.bz2</i> - create a tar with Bzip2 compression tar xjf <i>file.tar.bz2</i> - extract a tar using Bzip2 gzip <i>file</i> - compresses <i>file</i> and renames it to <i>file.gz</i> gzip -d <i>file.gz</i> - decompresses <i>file.gz</i> back to <i>file</i>
File Permissions	Network
chmod <i>octal file</i> - change the permissions of <i>file</i> to <i>octal</i> , which can be found separately for user, group, and world by adding: <ul style="list-style-type: none"> 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 .	ping <i>host</i> - ping <i>host</i> and output results whois <i>domain</i> - get whois information for <i>domain</i> dig <i>domain</i> - get DNS information for <i>domain</i> dig -x <i>host</i> - reverse lookup <i>host</i> wget <i>file</i> - download <i>file</i> wget -c <i>file</i> - continue a stopped download
SSH	Installation
ssh <i>user@host</i> - connect to <i>host</i> as <i>user</i> ssh -p <i>port user@host</i> - connect to <i>host</i> on port <i>port</i> as <i>user</i> ssh-copy-id <i>user@host</i> - add your key to <i>host</i> for <i>user</i> to enable a keyed or passwordless login	Install from source: ./configure make make install dpkg -i <i>pkg.deb</i> - install a package (Debian) rpm -Uvh <i>pkg.rpm</i> - install a package (RPM)
Searching	Shortcuts
grep <i>pattern files</i> - search for <i>pattern</i> in <i>files</i> grep -r <i>pattern dir</i> - search recursively for <i>pattern</i> in <i>dir</i> <i>command</i> grep <i>pattern</i> - search for <i>pattern</i> in the output of <i>command</i> locate <i>file</i> - find all instances of <i>file</i>	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. 



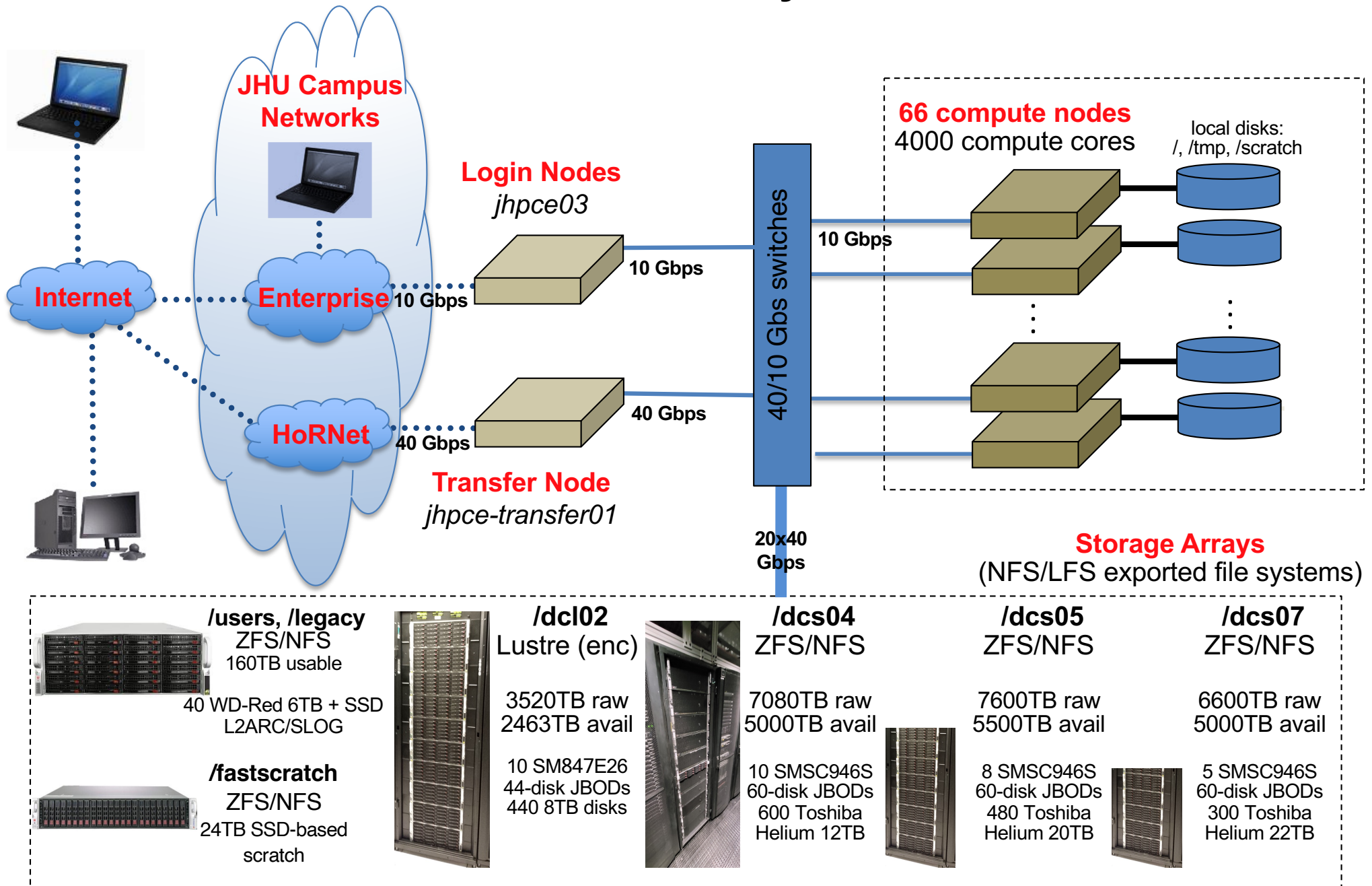
Schedule

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



Workstations, Desktops, Laptops

JHPCE System Architecture



Primary SLURM Commands

2 Main SLURM commands for utilizing the cluster:

- **sbatch** – submit a batch job to the cluster
- **srun --pty --x11 bash** – establish an interactive session

Other SLURM Commands for getting info on jobs & the cluster:

- **scancel** – cancel or pause a job
- **squeue** – see the status of running & pending jobs
- **sacct** – see the status of past (& running) jobs
- **sinfo** – see status of the compute nodes
- **scontrol show job** – see statistics from running jobs

JHPCE Written Scripts for getting info on jobs & the cluster:

- **qmem** – show summary of RAM usage for running jobs
- **slurmpic** – show table of compute node usage



Lab 2 - Using the SLURM cluster

Example 2a – using an **batch** session

```
cd class-scripts  
sbatch script1  
squeue --me  
sstat -j JOBID
```

examine results file with the **more** command
more slurm-JOBID.out

Example 2b – submitting a **interactive** job

```
srun --pty --x11 bash  
cd class-scripts  
bash script1
```

Note that output is displayed on the screen rather than saved in a file



Difference between Batch and Interactive

Batch Jobs:

- Running long-running jobs
- Batch Jobs continue to run even if you log off or get disconnected from the cluster

Interactive Sessions:

- Running short jobs
- Writing and debugging programs
- Running test on subset of data prior to submitting long-running batch job on full dataset
- Running graphical programs
- An interactive session ends and your jobs is terminated if your connection to the cluster is disconnected (laptop reboot, network drops)
 - any unsaved changes may be lost
 - programs will be terminated and may leave truncated results



Modules

Modules are sets of configuration information which change your environment to suite a particular software package.

We have modules for R, SAS, Mathematica, python, . . .

- module list
- module avail
- module avail stata
- module load
- module unload
- module describe



Never run a job on the login node!

Login nodes have many fewer resources than the compute nodes. They are a shared resource.

- Always use "**sbatch**" or "**srun**" 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 **srun**.
- Even something as simple as copying large files should be done via **srun** or **sbatch**



Other useful SLURM commands

squeue – shows information about running & pending jobs

squeue # defaults to all jobs for all users

squeue --me -t r,pd # just my running & pending jobs

scontrol show job – shows summary info about running jobs

scontrol show job JOBID

sacct – shows information about completed jobs

sacct -j JOBID

sacct --helpformat (lists avail info fields)

sacct --units=M -j JOBID -o JobID,JobName,MaxVMSize,Elapsed,TotalCPU

scancel – deletes your job (you can also can pause them)

scancel JOBID



Primary Commands – SLURM

(Simple Linux Utility for
Resource Management)

2 Main SLURM commands for utilizing the cluster:

- **sbatch** – submit a batch job to the cluster
- **srun --pty --x11 bash** – establish an interactive session

Other SLURM Commands for getting info on jobs & the cluster:

- **scancel** – cancel or pause a job
- **squeue** – see the status of running & pending jobs
- **sacct** – see the status of past (& running) jobs
- **sinfo** – see status of the compute nodes
- **scontrol show job** – see statistics from running jobs

JHPCE Written Scripts for getting info on jobs & the cluster:

- **qmem** – show summary of RAM usage for running jobs
- **slurmpic** – show table of compute node usage



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



Requesting additional Cores

- By default, when you submit a job with sbatch, or run srun, you are allotted 5GB of RAM and 1 core for your job.
- You can request more cores with the --cpus-per-task option to sbatch and srun.

- Examples:

```
sbatch --cpus-per-task=4 job1.sh
```

or

```
srun --mem=10G --cpus-per-task=6 --pty --x11 bash
```



Types of parallelism on JHPCE

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



Requesting additional RAM

- By default, when you submit a job with sbatch, or run srun, you are allotted 5GB of RAM and 1 core for your job.
- You can request more or less RAM via 2 options:
 - `--mem` : memory per node (for all cores used)
 - `--mem-per-cpu` : memory per core (harder to accurately estimate)
- Examples:
 - `sbatch --mem=10G job1.sh`**
 - This would give your batch job a total of 10GB of RAM
 - `srun --mem-per-cpu=5G --cpus-per-task=4 --pty --x11 bash`**
 - This would give your interactive session a total of 20GB of RAM and 4 cores



Estimating RAM usage

- There is no easy formula to know ahead of time how much RAM a job will need when working with large data.
- You can run a test job on a small subset of data.
- A good place to start is the size of the files you will be reading in. Add a bit extra, as a starting point.
- You can run `sacct` to gather info on a completed job:
`sacct -o JobID,JobName,ReqTRES%40,MaxVMSize,MAXRSS,State%20 -j JOBID`
- Try to make your RAM request slightly higher than your expected usage.
 - Too low and your job will get killed for exceeding your request
 - Too high and your job may take longer to get scheduled, plus you'll be squatting on RAM that others can use.
- Use `slurmpic` to see the current core and RAM availability, and plan accordingly.



Setting a time limit for your job

- By default, when you submit a job with sbatch, or run srun, you will have a 1 day time limit for your job.
- You can request more time with the --time option to sbatch and srun with the time in the format of DAYS-HH:MM:SS
- Your job will die at the end of the time limit!!

Examples:

- To set a 4 day limit for your job:
`sbatch --time=4-00:00:00 job1.sh # 4-00 also works`
- To set an 8 hour time limit for your interactive session:
`srun --time=08:00:00 --pty --x11 bash # 8:00 means 8 minutes`
- Shorter jobs are given higher priority via the “backfill scheduler”
- More information at: <https://jhpce.jhu.edu/slurm/time-limits/>



Email notification on sbatch job completion

- Rather than checking on your long-running job from time to time, you can add options to sbatch to receive an email when your job completes or fails by adding the mail-type and mail-user options:

```
$ sbatch --mail-type=FAIL,END --mail-user=john@jhu.edu script2
```

- Note that email notification 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 as it has appeared that they were getting spammed.



Supplying options to your sbatch job

- You can supply SLURM directives to sbatch in 4 ways:
- Order of precedence:

1. On the command line

```
$ sbatch --mem=10G --cpus-per-task=6 --mail-type=FAIL,END --mail-user=john@jhu.edu script2
```

2. Environment variable

3. Embedding them in your batch job script

Lines which start with “#SBATCH” are interpreted as options to **sbatch**. Such lines must:

- start at the very beginning of a line
- come after the interpreter line `#!/bin/bash`
- come before any commands
- There is an example in the class-scripts directory:

```
[login-31 /users/mmill1116/class-scripts]$ more script1-resource-request
#!/bin/bash
#
#SBATCH --mem=10G
#SBATCH --cpus-per-task=6
#SBATCH --mail-type=FAIL,END
#SBATCH --mail-user=marcus@jhu.edu

date
. . .
```



Supplying options to your sbatch job (cont'd)

4. In your `~/.slurm/defaults` file

Syntax is: `[<command>:][<cluster>:] <option> = <value>`

Where `[]` indicates an optional argument

Command can be one of (at least): `srun`, `sbatch`, `salloc`

(But perhaps other commands also refer to the file.)

You need to specify an asterisk in between colons

We have not tested blank or commented lines.

Example contents:

```
mem=2GB
```

```
mail-user=franksmith@jh.edu
```

```
srun:*:partition=debug
```

```
sbatch:*:mail-type=FAIL,END
```



Supplying options to your sbatch job (cont'd)

4. In your ~/.slurm/defaults file

Simple syntax is: `<directive> = <value>`

Full syntax is: `[<command>:<cluster>:] <directive> = <value>`

Where `[]` indicates an optional argument

Command can be one of (at least): `srun`, `sbatch`, `salloc` or `*`
(But perhaps other commands also refer to the file???)

You need to specify an asterisk in between colons (or a SLURM cluster name (ours are "jhpce3" & "cms"))

Blank and `#`-commented lines are allowed.

Example contents:

```
mem=2GB
```

```
mail-user=franksmith@jh.edu
```

```
*:time=5-00
```

```
srun*:partition=interactive
```

```
sbatch*:mail-type=FAIL,END
```



“How many jobs can I run?”

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 using array jobs.

More importantly, we 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 **100 cores per user and 1024GB of RAM per user**.

So, if a user submits 1,000 single-core jobs, the first 100 will begin immediately (assuming the cluster has 100 cores available on the shared queue), and the rest will remain in the 'PD' state until the first 100 jobs start to finish. As jobs complete, the cluster will start running 'PD' jobs, and keep the number of running jobs at 100.

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
 - `sbatch plot1.sh`
- Run R commands interactively
 - `srun --pty --x11 bash`
 - `module load R`
 - `R --no-save -f plot1.r`
- Look at pdf from example 4 via `xpdf`



Lab 4 – Transferring data

Data can be transferred to and from the cluster via the **transfer node**.

From outside of the JHPCE cluster, you can access the **transfer node** by using the address **jhpce-transfer01.jhsph.edu**. On a Mac or Linux system, you can use the text version of sftp to transfer data. You would start by running the following **from your local laptop or desktop**:

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

Login with the same credentials used for ssh. From within the `sftp` session, you can use `ls` and `cd` commands as well as the `get` command to get a file from the cluster and save it on your local system, and `put` to upload a file from your local system to the cluster.

```
sftp> cd class-scripts/R-demo
sftp> ls
plot1.r  plot1-R-results.pdf  plot1.r.Rout  plot1.sh  slurm-1344981.out
sftp> get plot1-R-results.pdf
```

Once you have the file on your laptop, you can look at it with the PDF viewer on your laptop. You can also use graphical sftp programs like Cyberduck on Mac or MobaXterm on Windows.

From within the JHPCE cluster, you can access the **transfer node** by running:

```
$ srun --pty --partition=transfer bash
```

This can be used if you need to download large files from the internet or some external source to the JHPCE cluster (wget, curl, git clone).

More info on transferring files on JHPCE can be found at:

<https://jhpce.jhu.edu/access/file-transfer/>

<https://jhpce.jhu.edu/access/mobaxterm/#configuring-sftp-sessions>

<https://www.digitalocean.com/community/tutorials/how-to-use-sftp-to-securely-transfer-files-with-a-remote-server>



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 yourusername@jhpce03.jhsph.edu
```

- Start up Rstudio

```
$ srun --pty --x11 --mem=10G bash  
$ module load R  
$ module load rstudio  
$ rstudio
```

Your `srun` session will be running, but you will not be able to run additional commands until you exit out of rstudio.



Lab 6 – Running Stata



Batch:

```
$ cd $HOME/class-scripts/stata-demo
$ ls
$ more stata-demo1.sh    # see contents of the batch script
$ more stata-demo1.do    # see contents of the stata program
$ sbatch stata-demo1.sh
$ cat stata-demo1.log    # see the output
```

Interactive:

```
$ srun --pty --x11 --cpus-per-task=4 bash
$ module load stata
$ stata-mp
```

or

```
$ xstata-mp # starts the GUI interface
```

Notes:

- The program and script do not need to be named the same, but it is good practice to keep them the same when possible.
- File extensions are sometimes meaningful in Linux. SAS doesn't care, but Stata programs need to have ".do" as the extension. It is good practice for human readability.
- By default "stata" runs a single thread. For faster results when running on real data, request 2 or more cores and use the command "stata-mp" instead of "stata"
- By default stata stores temporary files in /tmp. You may need to define an environmental variable to avoid job failure due to lack of space. `export STATATMP=$HOME`



Lab 7 – Running SAS



- SAS example:

The SAS module will only load when run on a compute node in the "sas" partition!!!!

Batch:

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

Interactive:

```
$ srun --pty --x11 --partition=sas bash # you may well want to ask for more RAM
$ module load sas
$ sas
$ exit      # log out of your srun session & return to the login node
```

To display a plot, SAS needs to send it to a web browser running on the compute node. To do this you would need to run:

```
$ sas -helpbrowser SAS -xrm "SAS.webBrowser: '/usr/bin/chromium-browser'" -xrm
"SAS.helpBrowser: '/usr/bin/chromium-browser'"
```

Since that's a lot to type, you can create a bash function by adding the following lines to your ~/.bashrc file.

```
csas ()
{
    sas -helpbrowser SAS -xrm "SAS.webBrowser: '/usr/bin/chromium-browser'" -xrm
    "SAS.helpBrowser: '/usr/bin/chromium-browser'" "$@" > /dev/null 2>&1
}
```

Once added, and after logging out and back in, you can start sas with its options by just running "csas".



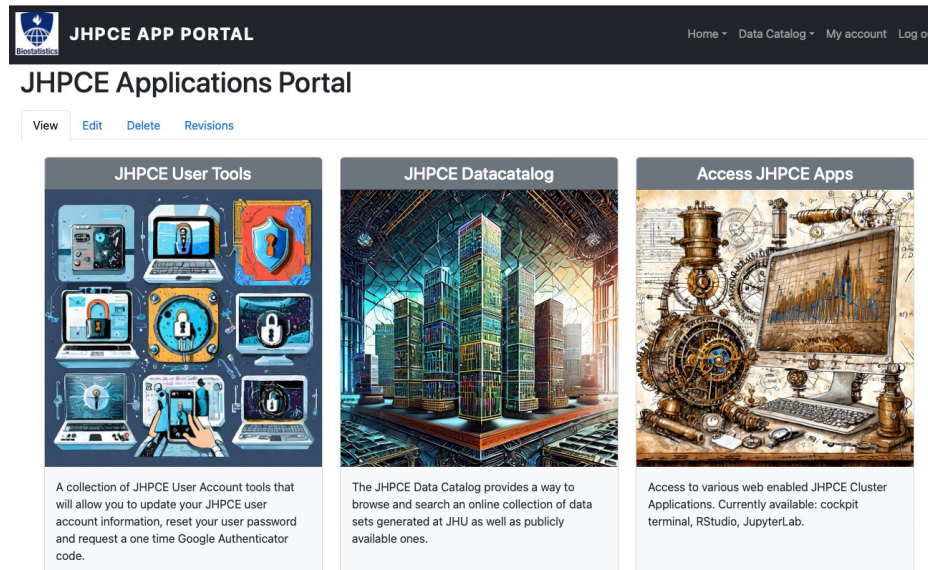
Other Queues/Partitions

- “shared” queue – default queue
- “dedicated” queues - Queues that are for PIs who have purchased nodes on the cluster.
- “transfer” partition
 - **srun --pty --x11 -p transfer bash**
(jhpce-transfer01.jhsph.edu)
- “sas” partition
 - **srun --pty --x11 -p sas bash**
- “gpu” partition
 - **srun --pty --x11 -p gpu bash**



JHPCE App Portal

You can access the JHPCE App Portal at <https://jhpce-app02.jhpce.jhu.edu>



The App Portal can be used for:

- User self service (password resets)
- Access to web-based Apps (Rstudio, Jupyter Labs, Visual Studio)
- Only accessible on campus or via VPN

More information can be found on our web site <https://jhpce.jhu.edu/portal/web-apps/>



Summary

- Review
 - Use ssh to connect to JHPCE cluster
 - Use rsun and sbatch to submit jobs
 - Never run jobs on the login nodes
 - Helpful resources
 - <https://jhpce.jhu.edu> (available anywhere)
 - <https://jhpce-app02.jhpce.jhu.edu> (available on campus or VPN)
 - bitsupport@lists.johnshopkins.edu - System issues
 - bithelp@lists.johnshopkins.edu - Application issues
- What to do next
 - Set up ssh keys if you will be accessing the cluster frequently
<https://jhpce.jhu.edu/knowledge-base/authentication/login/>
 - Get familiar with Linux
 - Play nice with others – multi-user, community-supported system.



Thanks for attending! Questions?

