Joint High Performance Computing Exchange (JHPCE) Cluster Orientation

JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

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

Version: 20260218

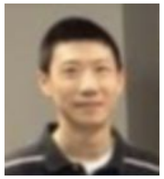
Schedule

- **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.jhsph.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

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.
- Also referred to as High Performance Computing (HPC)

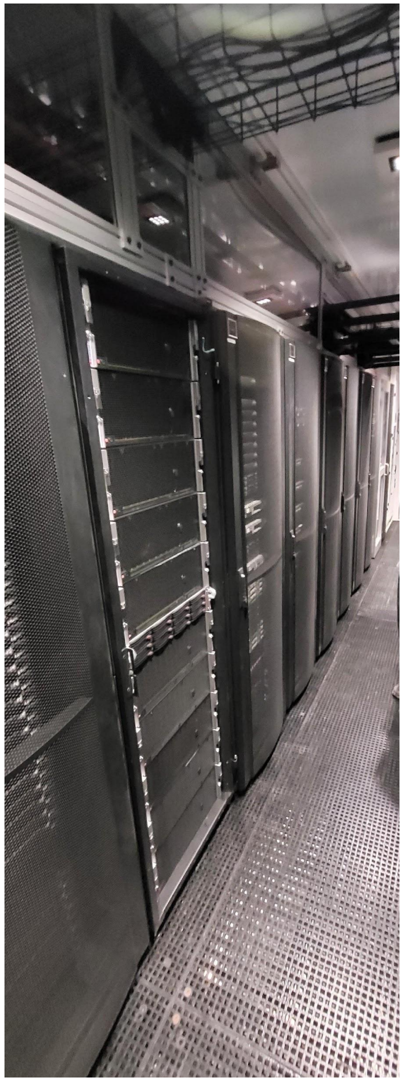
Why would you use a cluster?

- 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**)

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)
- Located at Bayview Colocation Facility (ARCH)

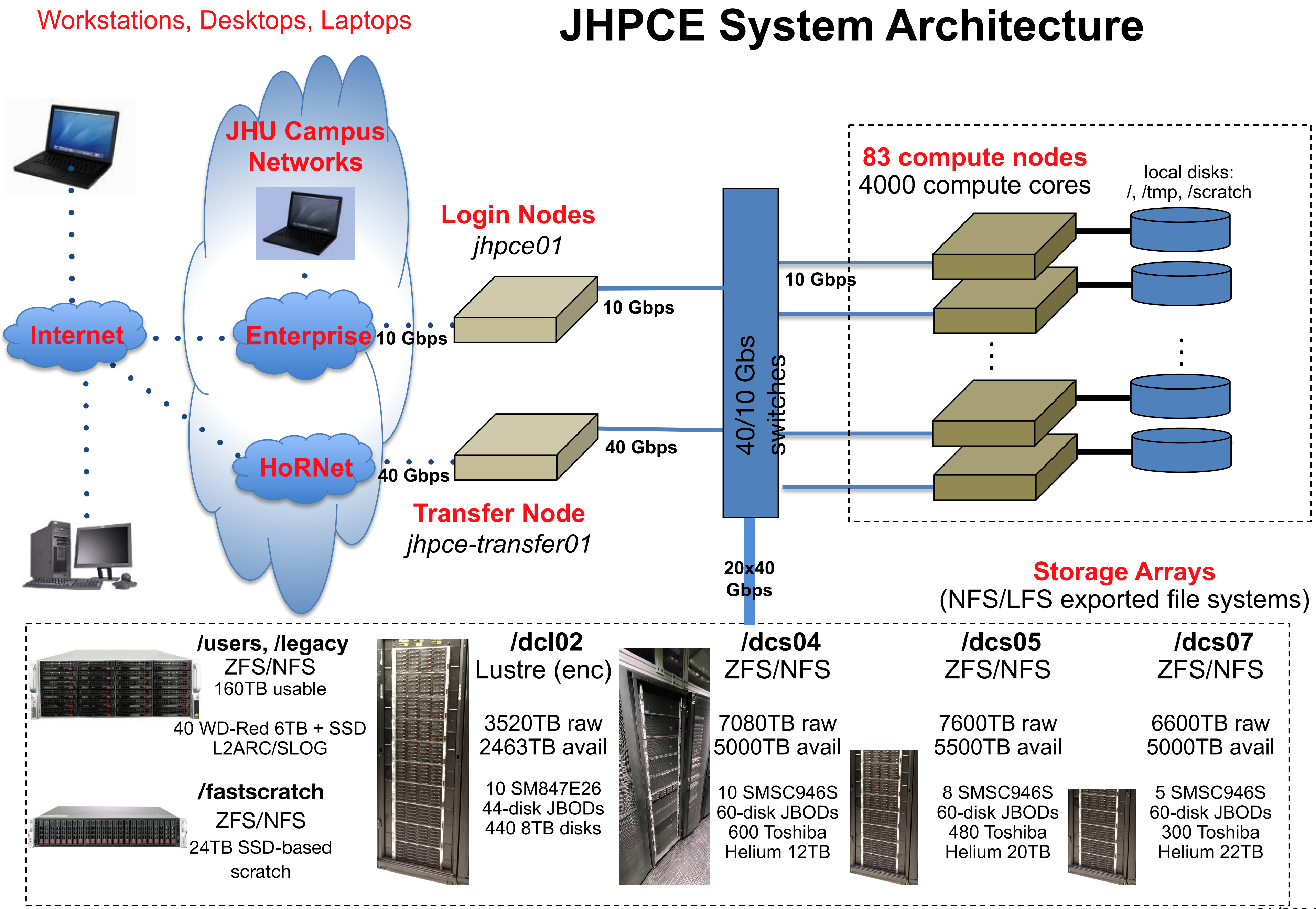
Hardware:

- 12 Racks of equipment – 5 compute, 6 storage, 1 infra.
- 86 Nodes – 83 compute, 1 transfer, 2 login
  - 4000 Cores - Nodes have 2 - 4 CPUs, 24 to 128 cores per node
  - 46 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
- 10 GPU nodes: 37 Nvidia V100, A100, H100, L40S GPUs
- 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 ...

JHPCE System Architecture



01/2024

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

- **Partition:** The queue that is used to schedule jobs to run on the nodes.
    - **shared –** The default partition

- **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 "`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 X11 server built into it.
- For Windows, if you are using Putty, you will need to install an X server such as Cygwin.

- For Macs:

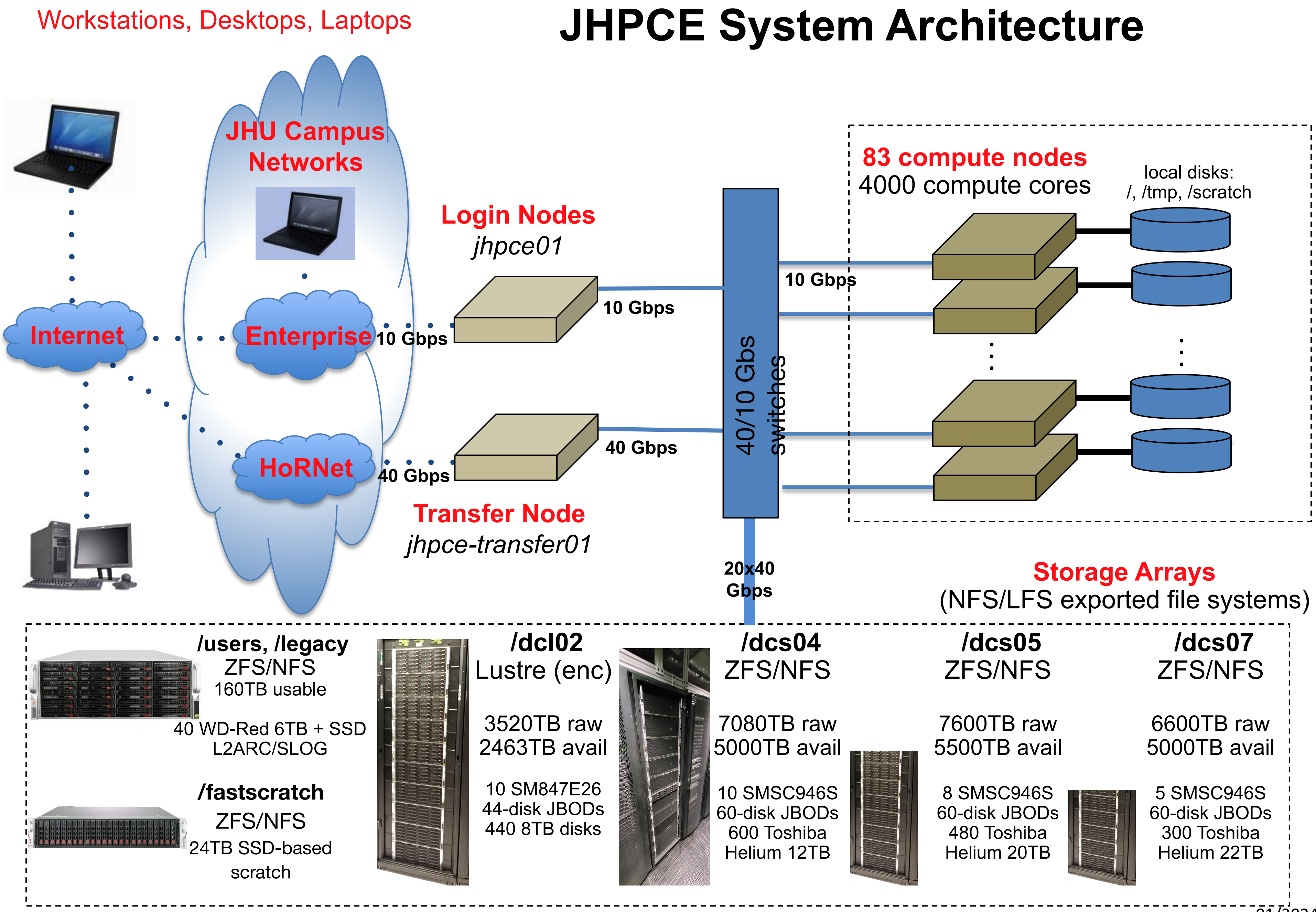
1) You need to have the Xquartz program installed on your laptop. This software is a free download from Apple, and does require you to reboot your laptop http://xquartz.macosforge.org/landing/
2) You need to add the "-X" option to your ssh command:

```
$ 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



01/2024

Example 1 – Logging in

- Bring up Terminal
- Run: `ssh -X USERID@jhpce01.jhsph.edu`
    - Your JHPCE userID is not the same as your JHED ID.
- 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"
- Linux Shell prompt

Lab 1 - Logging In

- For Mac/Linux laptop Users:

- Bring up a Terminal
- Run: **`ssh 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 Session 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.
  - Your new password must be at least 8 characters long
  - Your new password must have at least 3 of:
    - Upper case character
    - Lower case character
    - Digit
    - Special Character, such as: %$!@

- 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://jhpce.jhu.edu/storage/fastscratch/

- (optional) Setup ssh keys

https://jhpce.jhu.edu/access/ssh/#ssh-keys
https://jhpce.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`

Looking at files:

`-more/less`

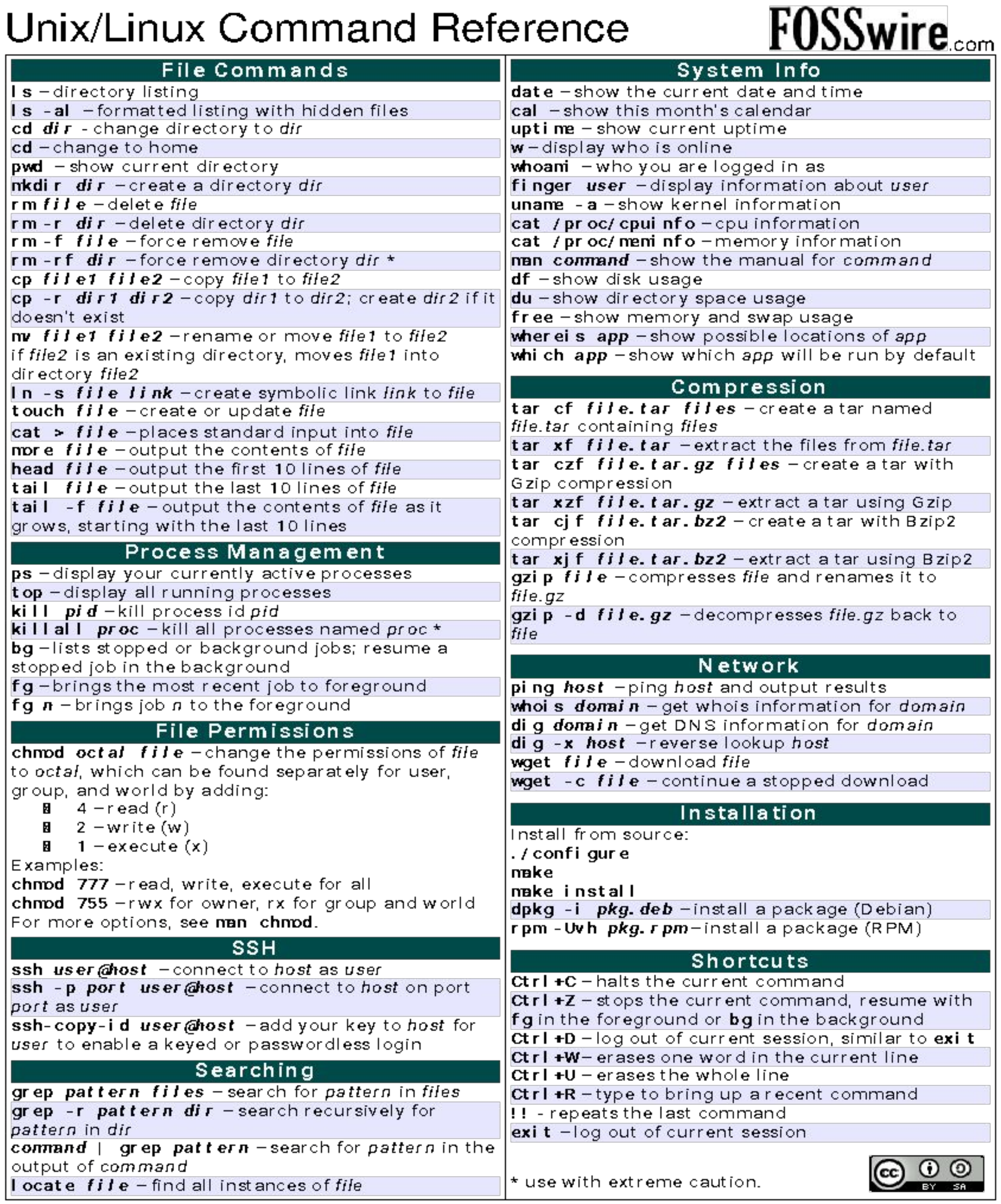
Commands in example script:

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

Changing files with editors:

`- nano`
`- vi/emacs`
`- gedit (with X11)`

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

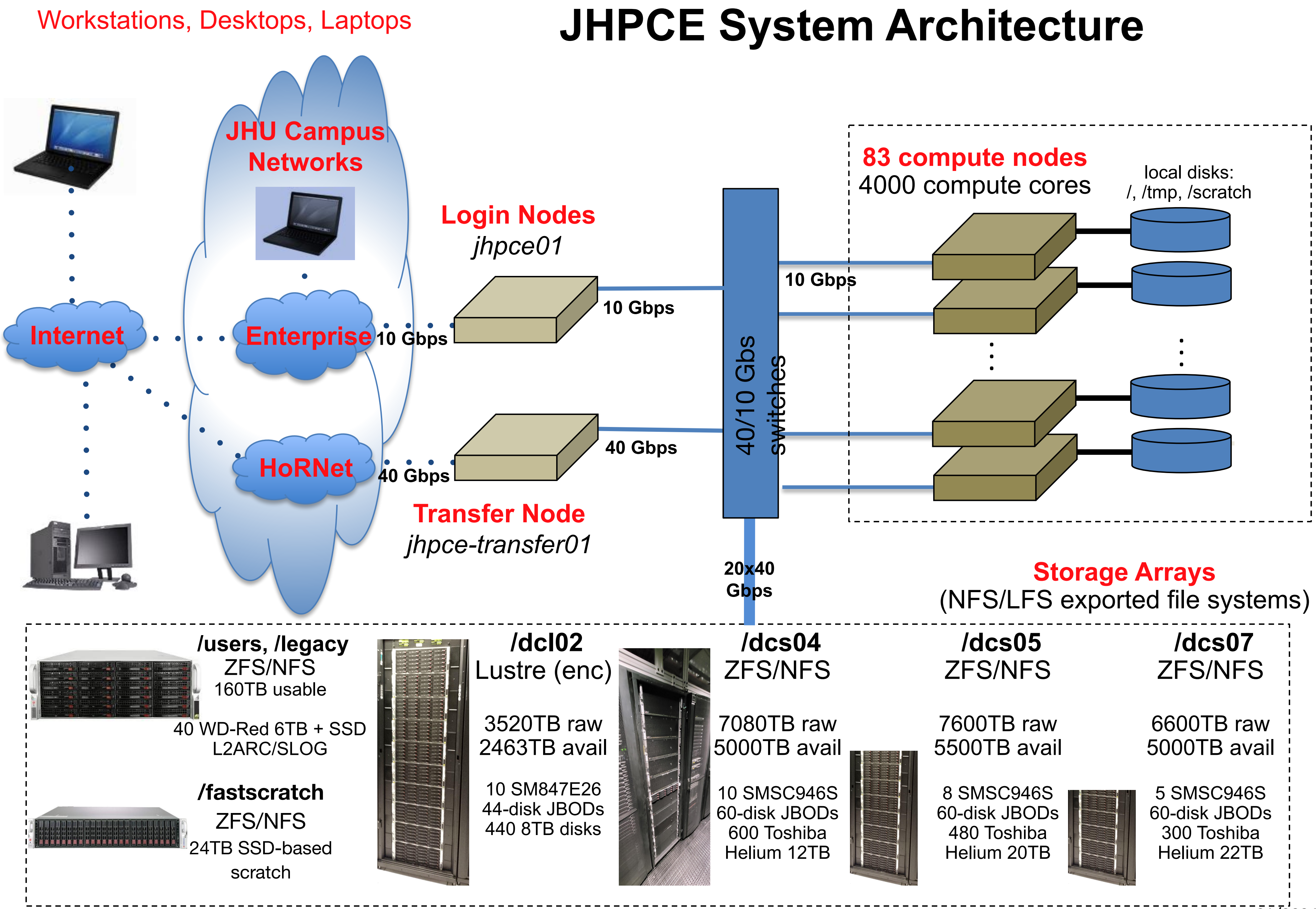
Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent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

JHPCE System Architecture



01/2024

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:

- `smem` – show summary of RAM usage for running jobs
- `slurmpic` – show table of compute node usage

Lab 2 - Using the SLURM cluster

Example 2a – submitting a **batch** job

```
cd class-scripts
sbatch script1
squeue --me
```

examine results file with the `more` command

```
more slurm-JOBID.out
```

Example 2b – establish an **interactive** session

```
srun --pty --x11 bash
ls
bash script1
```

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

Modules

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

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

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

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
- You can submit multiple jobs

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

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
```

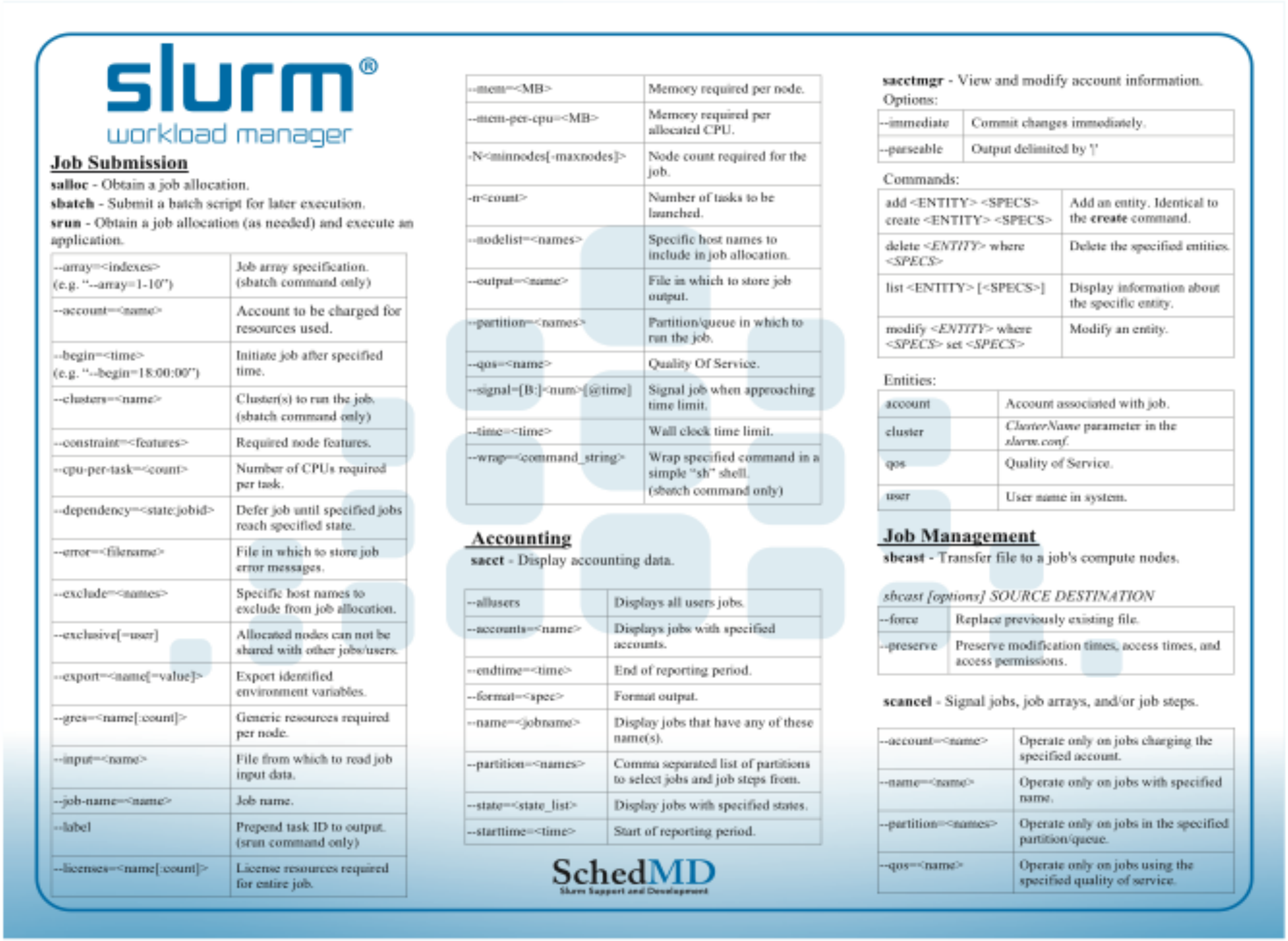
Primary Commands – SLURM

(Simple Linux Utility for Resource Management)

2 Main SLURM commands for utilizing the cluster:
- **`sbatch`** – submit a <u>batch</u> job to the cluster
- **`srun --pty --x11 bash`** – establish an <u>interactive</u> 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:
- **`smem`** – show summary of RAM usage for running jobs
- **`slurmpic`** – show table of compute node usage
- jobson



slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

--array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
--account=<name>	Account to be charged for resources used.
--begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
--clusters=<name>	Cluster(s) to run the job. (sbatch command only)
--constraint=<features>	Required node features.
--cpu-per-task=<count>	Number of CPUs required per task.
--dependency=<state:jobid>	Defer job until specified jobs reach specified state.
--error=<filename>	File in which to store job error messages.
--exclude=<names>	Specific host names to exclude from job allocation.
--exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
--export=<name[=value]>	Export identified environment variables.
--gres=<name[:count]>	Generic resources required per node.
--input=<name>	File from which to read job input data.
--job-name=<name>	Job name.
--label	Prepend task ID to output. (srun command only)
--licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by '|'

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete *<ENTITY>* where *<SPECS>*	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify *<ENTITY>* where *<SPECS>* set *<SPECS>*	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

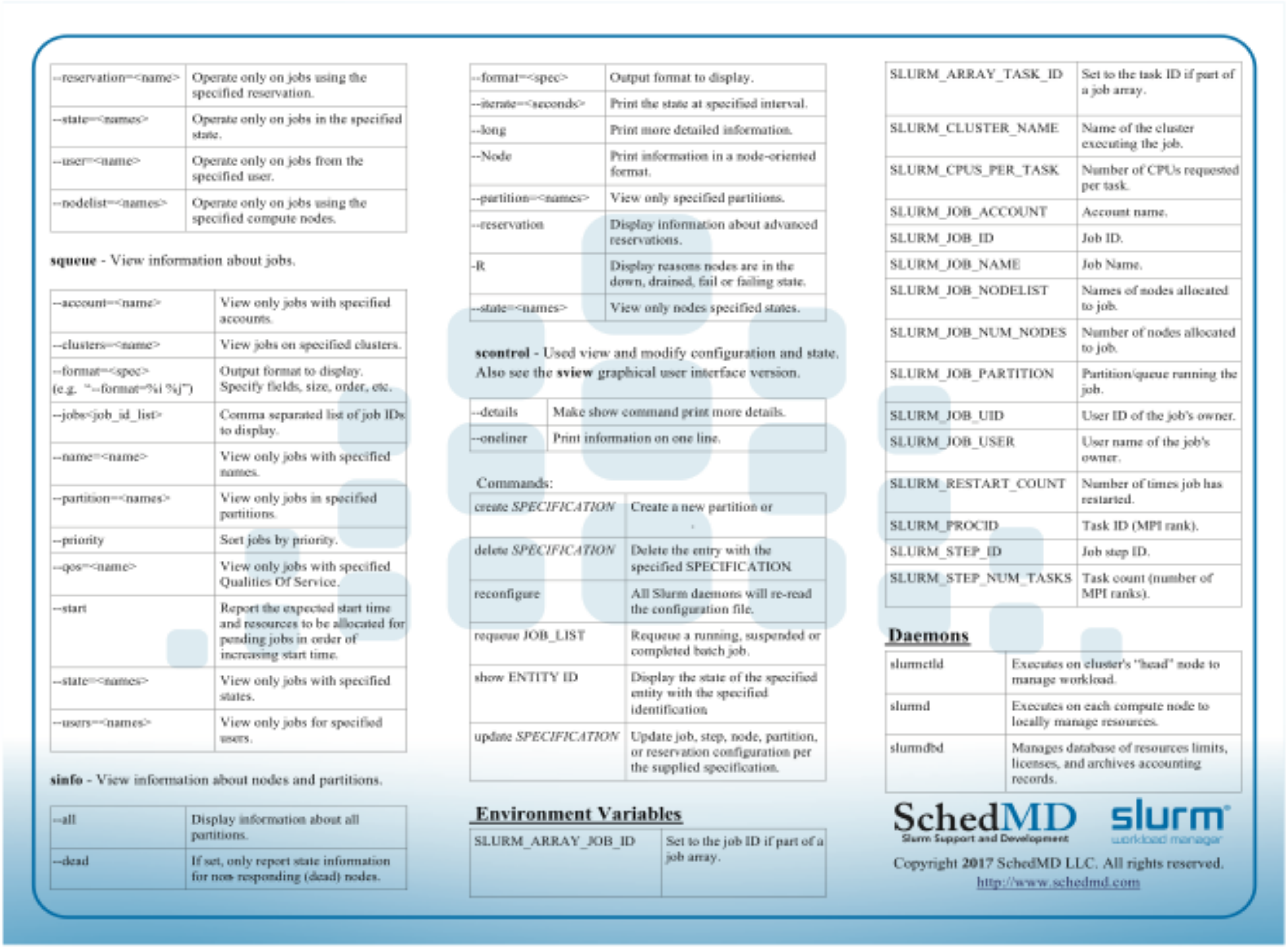
sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.



--reservation=<name>	Operate only on jobs using the specified reservation.
--state=<names>	Operate only on jobs in the specified state.
--user=<name>	Operate only on jobs from the specified user.
--nodelist=<names>	Operate only on jobs using the specified compute nodes.

squeue - View information about jobs.

--account=<name>	View only jobs with specified accounts.
--clusters=<name>	View jobs on specified clusters.
--format=<spec> (e.g. "--format=%i %j")	Output format to display. Specify fields, size, order, etc.
--jobs<job_id_list>	Comma separated list of job IDs to display.
--name=<name>	View only jobs with specified names.
--partition=<names>	View only jobs in specified partitions.
--priority	Sort jobs by priority.
--qos=<name>	View only jobs with specified Qualities Of Service.
--start	Report the expected start time and resources to be allocated for pending jobs in order of increasing start time.
--state=<names>	View only jobs with specified states.
--users=<names>	View only jobs for specified users.

sinfo - View information about nodes and partitions.

--all	Display information about all partitions.
--dead	If set, only report state information for non-responding (dead) nodes.
--format=<spec>	Output format to display.
--iterate=<seconds>	Print the state at specified interval.
--long	Print more detailed information.
--Node	Print information in a node-oriented format.
--partition=<names>	View only specified partitions.
--reservation	Display information about advanced reservations.
-R	Display reasons nodes are in the down, drained, fail or failing state.
--state=<names>	View only nodes specified states.

scontrol - Used view and modify configuration and state. Also see the **sview** graphical user interface version.

--details	Make show command print more details.
--oneliner	Print information on one line.

Commands:

create *SPECIFICATION*	Create a new partition or
delete *SPECIFICATION*	Delete the entry with the specified SPECIFICATION
reconfigure	All Slurm daemons will re-read the configuration file.
requeue JOB_LIST	Requeue a running, suspended or completed batch job.
show ENTITY ID	Display the state of the specified entity with the specified identification
update *SPECIFICATION*	Update job, step, node, partition, or reservation configuration per the supplied specification.

Environment Variables

SLURM_ARRAY_JOB_ID	Set to the job ID if part of a job array.
SLURM_ARRAY_TASK_ID	Set to the task ID if part of a job array.
SLURM_CLUSTER_NAME	Name of the cluster executing the job.
SLURM_CPUS_PER_TASK	Number of CPUs requested per task.
SLURM_JOB_ACCOUNT	Account name.
SLURM_JOB_ID	Job ID.
SLURM_JOB_NAME	Job Name.
SLURM_JOB_NODELIST	Names of nodes allocated to job.
SLURM_JOB_NUM_NODES	Number of nodes allocated to job.
SLURM_JOB_PARTITION	Partition/queue running the job.
SLURM_JOB_UID	User ID of the job's owner.
SLURM_JOB_USER	User name of the job's owner.
SLURM_RESTART_COUNT	Number of times job has restarted.
SLURM_PROCID	Task ID (MPI rank).
SLURM_STEP_ID	Job step ID.
SLURM_STEP_NUM_TASKS	Task count (number of MPI ranks).

Daemons

slurmctld	Executes on cluster's "head" node to manage workload.
slurmd	Executes on each compute node to locally manage resources.
slurmdbd	Manages database of resources limits, licenses, and archives accounting records.

SchedMD Slurm Support and Development

slurm workload manager

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 --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 are 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.

Examples:

- To set a 4 day limit for your job:

```
sbatch --time=4-00:00:00 job1.sh
```

- To set an 8 hour time limit for your interactive session:

```
srun --time=08:00:00 --pty --x11 bash
```

- 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, 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/mmill116/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
```

“How many jobs can I run?”

There is a limit of 10,000 pending or running jobs than any user can have. 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 **400 cores per user and 2.5TB of RAM per user.**

So, if a user submits 1,000 single-core jobs, the first 400 will begin immediately (assuming the cluster has 400 cores available on the shared queue), and the rest will remain in the ‘PD’ state until the first 400 jobs start to finish. As jobs complete, the cluster will start running ‘PD' jobs, and keep the number of running jobs at 400.

Similarly, if a user's job requests 100GB of RAM to run, the user would only be able to run 25 jobs before hitting their 2.5TB limit, and subsequent jobs would remain in “PD” 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 conda_R`
  - `R --no-save -f plot1.r`
  - 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
sffp> 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.

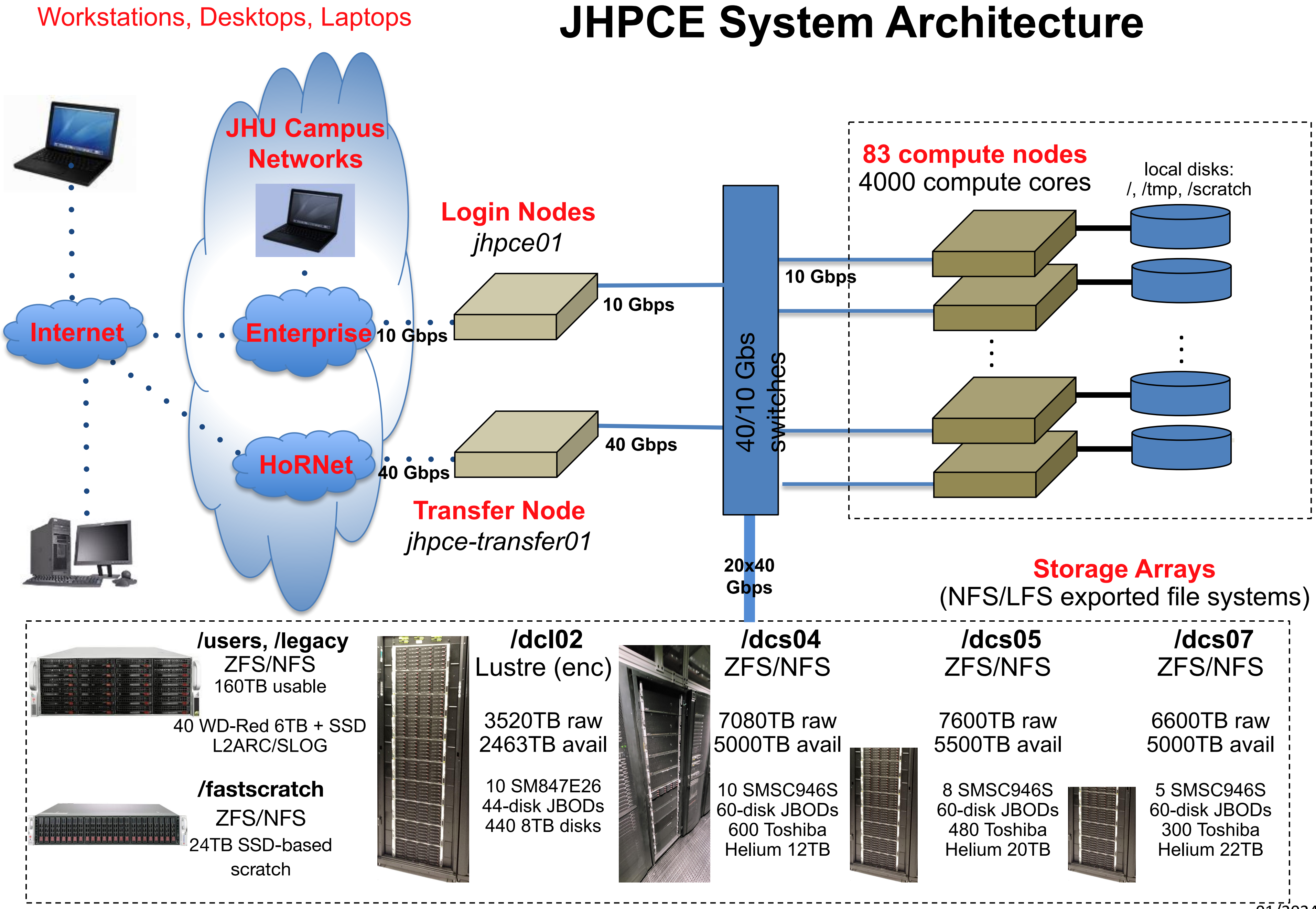
From within the JHPCE cluster, you can access the **transfer node** by running:

```
$ 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

JHPCE System Architecture



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@jhpce01.jhsph.edu
```

- Start up Rstudio

```
$ srun --pty --x11 --mem=10G bash
$ module load conda_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:

Batch:

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

Interactive:

```
  $ srun --pty --x11 --partition=sas bash
  $ module load sas
  $ sas
```

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

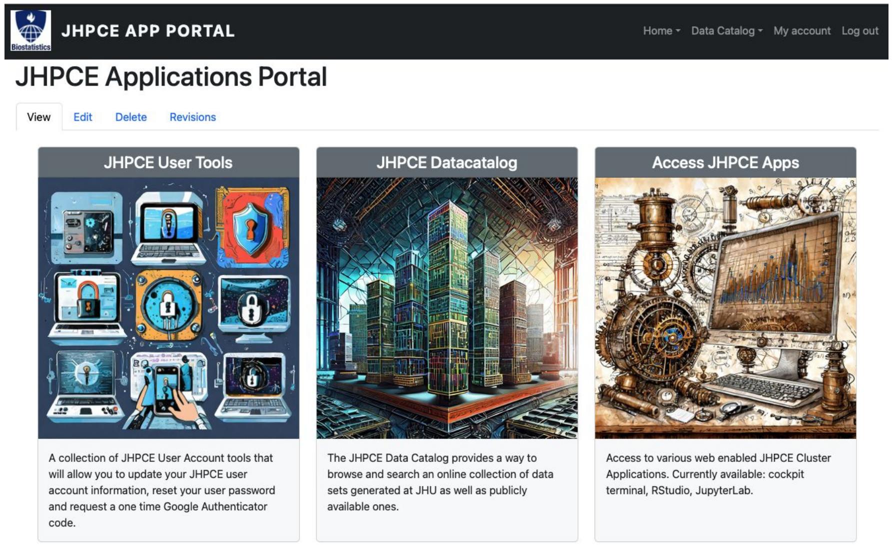
- "shared" partition – default queue
- "dedicated" partition - Queues that are for PIs who have purchased nodes on the cluster.

- "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 --gpus=1 bash`

JHPCE App Portal

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



JHPCE Applications Portal

JHPCE User Tools

A collection of JHPCE User Account tools that will allow you to update your JHPCE user account information, reset your user password and request a one time Google Authenticator code.

JHPCE Datacatalog

The JHPCE Data Catalog provides a way to browse and search an online collection of data sets generated at JHU as well as publicly available ones.

Access JHPCE Apps

Access to various web enabled JHPCE Cluster Applications. Currently available: cockpit terminal, RStudio, JupyterLab.

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 srun and sbatch to submit jobs or run an interactive session
    - Never run jobs on the login nodes
    - Helpful resources
        - https://jhpce.jhu.edu (available anywhere)
        - https://jhpce-app02.jhsph.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/access/ssh/
    - Get familiar with Linux
    - Play nice with others – multi-user, community-supported system.

Thanks for attending! Questions?