Joint High Performance Computing Exchange (JHPCE) Cluster SLURM Overview

2023-10-17

Biostatistics Journal/Computing Club

JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

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

Schedule

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

Who we are:

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

Schedule

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

What is a cluster?

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

Why would you use a cluster?

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

Types of parallelism

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

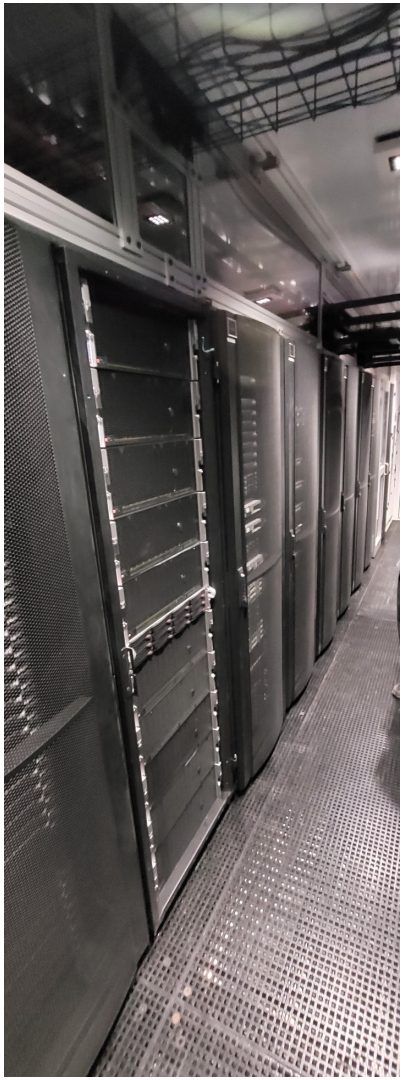
2. Multi-core (or multi-threaded) – a single job using multiple CPU cores via program threads on a single machine (cluster node). Also see discussion of fine-grained vs coarse-grained parallelism at
http://en.wikipedia.org/wiki/Parallel_computing

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

Node (Computer) Components

- Each computer is called a "**Node**"
- Each node, just like a desktop/laptop has:
  - RAM
  - Intel/AMD CPUs
  - Disk space
- Unlike desktop/laptop systems, nodes do not make use of a display/mouse – they are used from a command line interface known as a "**shell**".

The JHPCE cluster components



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

Hardware:

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

Software:

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

How do programs get run on the compute nodes?

- We use a product called “SLURM” that schedules programs (jobs). We have been migrating from SGE to SLURM this summer.
- Jobs are assigned to slots as they become available and meet the resource requirement of the job
- Jobs are submitted to partitions (formerly **queues)**
- The cluster nodes can also be used interactively.

Motivation: JHPCE 3.0

We are implementing an upgrade of the cluster
- from CentOS 7.9 (based on RHEL 7.9)
- to Rocky 9.2 (based on RHEL 9.2).

Internally we refer to these as JHPCE 2.0 and JHPCE 3.0, or in shorthand, J2 and J3.

Part of this upgrade is a switch in the choice of **job scheduler.**
- J2 uses SGE (the Sun Grid Engine).
- J3 will use “SLURM” (Simple Linux Utility for Resource Management)

- As of 2023-10-17, 50 nodes migrated, 7 “draining” on SGE, and 10 still on SGE. 4 old nodes decommissioned.

Schedule

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

How do you use the cluster?

- The JHPCE cluster is accessed using SSH (Secure SHell), so you will need an ssh client.
- Use **`ssh`** to login to "**`jhpce03.jhsph.edu`**"

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

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

http://mobaxterm.mobatek.net/

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

http://www.cygwin.com

http://winscp.net

Quick note about graphical programs

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

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

- For Macs:

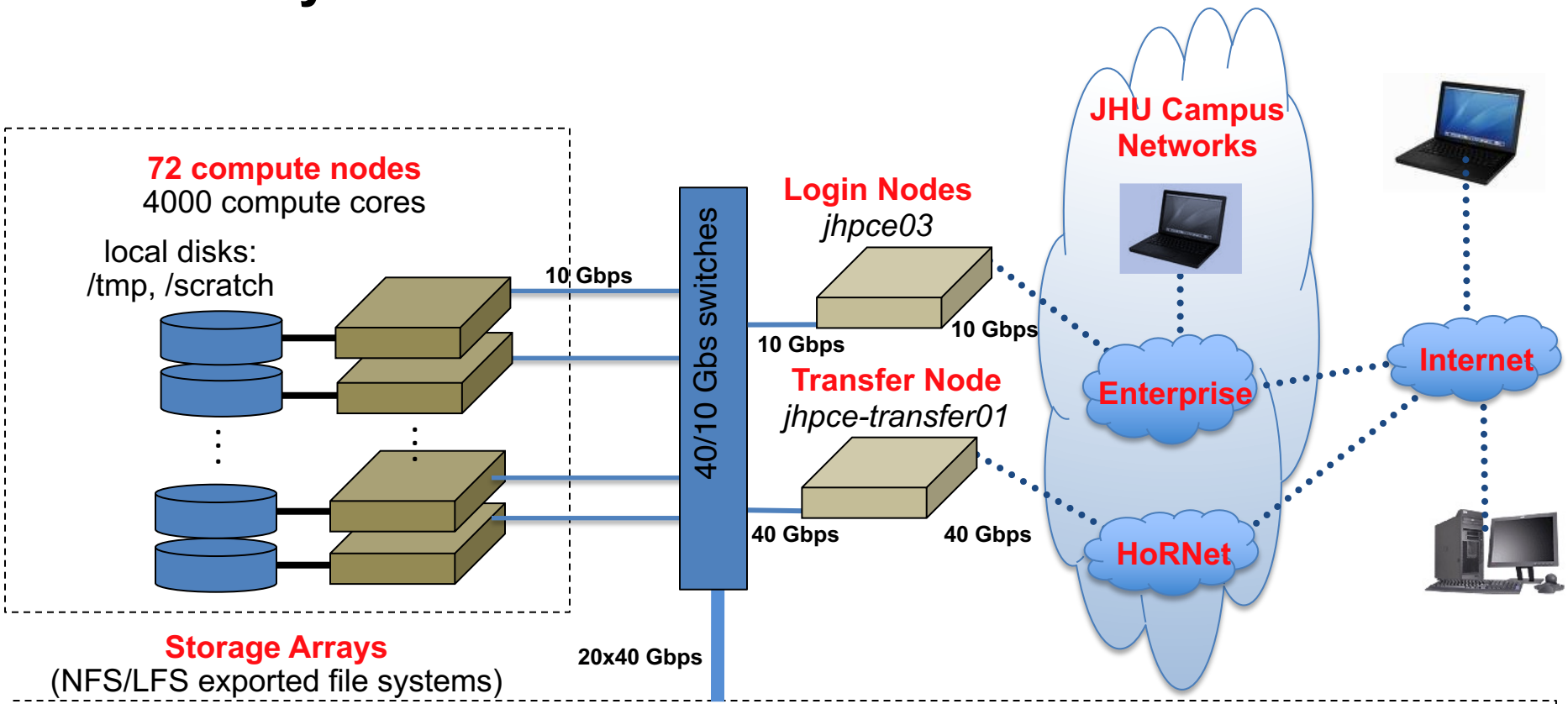
1) You need to have the Xquartz program installed on your laptop. This software is a free download from Apple, and does require you to reboot your laptop http://xquartz.macosforge.org/landing/
2) You need to add the "-X" option to your ssh command:

```
$ ssh -X mmill116@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 mmill116@jhpce03.jhsph.edu
```

JHPCE System Architecture



General Linux/Unix Commands

Navigating Unix:

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

Commands in example script:

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

Looking at files:

- more/less

Changing files with editors:

- nano
- vi/emacs

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

Unix/Linux Command Reference

FOSSwire.com

File Commands

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

Process Management

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

File Permissions

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

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

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

SSH

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

Searching

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

System Info

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

Compression

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

Network

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

Installation

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

Shortcuts

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

* use with extreme caution.

SGE vs SLURM

SGE	SLURM	Notes
qsub	sbatch	SLURM requires an interpreter first line
qrsh	srun --pty --x11 bash	Interactive login
qhost –q	sinfo	Queue status, by queue
qhost	sinfo –N	Queue status, by node.
qstat –u *	squeue	See everyone's jobs
qdel	scancel	Cancel job
qhold	scontrol hold	Pause job
qrls	scontrol release	Release job

Table-based command comparisons:

- https://hpcsupport.utsa.edu/foswiki/pub/Main/SampleSlurmSubmitScripts/SGEtoSLURMconversion.pdf
- https://docs.mpcdf.mpg.de/doc/computing/clusters/aux/migration-from-sge-to-slurm

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

http://www.schedmd.com

SLURM Documentation

Key links:

- https://slurm.schedmd.com/documentation.html
- https://slurm.schedmd.com/quickstart.html
- https://slurm.schedmd.com/man_index.html

The latest SLURM is 23.02.

We have installed SLURM version 22.05.09.

In a few cases, you might want to consult our specific version's documentation:
https://slurm.schedmd.com/archive/slurm-22.05.9/

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
```

sacct – shows information about completed jobs

```
sacct -aj JOBID
sacct -S=2023-06-1514:30 # started after 2:30pm
sacct -S=noon # also can use: today, midnight, now
sacct --helpformat # lists avail info fields
sacct --units=M -j 130.batch -o
JobID,MaxVMSizeNode,MaxVMSize,AveVMSize,MaxRSS,AveRSS,MaxDiskRead,MaxDiskWrit
e,AveCPUFreq,TRESUsageInMax%-20 # that is all on one line; capitalization does not matter
```

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

```
scancel JOBID
scancel -u <username> # cancels all of that user's jobs
```

Useful Slurm commands (cont'd)

`scontrol` – shows information about many types of things

```
scontrol show job <jobid>
```

```
login31:~% scontrol show job 22800

JobId=22800 JobName=scr120
   UserId=tunison(42629) GroupId=users(100)
   Priority=1 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)

   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:49 TimeLimit=90-00:00:00 TimeMin=N/A
   SubmitTime=2023-08-15T21:09:02 EligibleTime=2023-08-15T21:09:02
   AccrueTime=2023-08-15T21:09:02
   StartTime=2023-08-15T21:09:03 EndTime=2023-11-13T20:09:03 Deadline=N/A
   Scheduler=Backfill

   Partition=shared AllocNode:Sid=login31:3264028
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=compute-094
   BatchHost=compute-094

   NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=2,mem=2G,node=1,billing=2
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=2G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Command=/users/tunison/scr120
   WorkDir=/users/tunison
   StdErr=/users/tunison/slurm-22800.out
   StdIn=/dev/null
   StdOut=/users/tunison/slurm-22800.out
```

Useful Slurm commands (cont'd)

`scontrol` – shows information about many types of things

`scontrol show node [nodename]` # defaults to all nodes

```
NodeName=compute-153 Arch=x86_64 CoresPerSocket=32
   CPUAlloc=56 CPUEfctv=128 CPUTot=128 CPULoad=7.00

   NodeAddr=compute-153 NodeHostName=compute-153 Version=22.05.9
   OS=Linux 5.14.0-284.11.1.el9_2.x86_64 #1 SMP PREEMPT_DYNAMIC Tue May 9 17:09:15 UTC 2023

   RealMemory=515415 AllocMem=458752 FreeMem=281975 Sockets=2 Boards=1
   State=MIXED ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=debug,interactive,shared

   CfgTRES=cpu=128,mem=515415M,billing=128
   AllocTRES=cpu=56,mem=448G
```

TRES means Trackable RESources

GRES means General RESources (seen on e.g. GPU nodes)

Requesting additional RAM and CORES

- By default, when you submit a job with sbatch, or run srun, you are allotted 10GB of RAM and 2 cores for your job.

- You can request more RAM by setting the "--mem” option
- You can request more CORES by setting the "--cpus-per-task” option

- Examples:

```
sbatch --mem=20G job1.sh
```

or

```
srun --mem=40G --cpus-per-task=4 --pty --x11 bash
```

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

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.

2. Environment variables

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

`$ less /jhpce/shared/jhpce/slurm/class-scripts/script3.annotated`

This file contains many examples!!

Modules

Modules for R, SAS, Mathematica . . .

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

Some useful scripts that we've written

- smem – shows memory usage for your jobs

```
[compute-147 /users/mmill116/class-scripts]$ smem
520605     mmill116   compute-147     mem=7.80 MB      vmem=8.63 MB elapsed=4:30         bash
[compute-147 /users/mmill116/class-scripts]$ smem  520606
520606 was completed and used 40.00 KB memory
```

- slurmpic – shows status of nodes

```
[compute-147 /users/mmill116/class-scripts]$ slurmpic
NODENAME    NODESTATE  PARTITION CPUS_A/T CPU_LOAD  TOT_MEM FREEMEM SYSFREEMEM
compute-057 down+not_    shared*    0/ 64     0.00   488 GB  488 GB       0 GB
compute-062     mixed    shared*   38/ 64    21.79   488 GB    8 GB     407 GB
compute-063     mixed    shared*    2/ 64    11.45   503 GB  103 GB     282 GB
compute-065     mixed    shared*   20/ 64    20.17   488 GB   88 GB     102 GB
compute-068     mixed    shared*    4/ 64     1.16   488 GB   88 GB     447 GB
. . .
```

Schedule

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

Basics of using SLURM

You can copy a SLURM version of class scripts to your home directory.

rsync -a /jhpce/shared/jhpce/slurm/class-scripts class-scripts-slurm

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

```
srun --pty --x11 bash
cd class-scripts-slurm
./script1 # script is executable and first line is #!/bin/bash
exit
```

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

```
cd class-scripts
sbatch script1  # note script2 doesn't need to be executable
squeue
sstat –j JOBID        # use the number of your job
```

examine results files with the **`cat`** or **`less`** commands

```
slurm-JOBID.out
```

Note: Your script or interactive shell run in the same directory in which you ran sbatch or srun, unless the --chdir argument is used.

Running a Python program

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

```
[compute-076 /users/mmill116/class-scripts]$ nano script2.py
[compute-076 /users/mmill116/class-scripts]$ cat script2.py
#!/usr/bin/python
print("Hello World")
[compute-076 /users/mmill116/class-scripts]$ python script2.py
Hello World

[compute-076 /users/mmill116/class-scripts]$ sbatch script2.py
Submitted batch job 520604
[compute-076 /users/mmill116/class-scripts]$ ls
R-demo    script1                  script2.py  slurm-520604.out
SAS-demo  script1-resource-request  sequence1   stata-demo
[compute-076 /users/mmill116/class-scripts]$ cat slurm-520604.out
Hello World
```

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`
  - `Open and run plot1.r`

```
[compute-151 slurm/class-scripts-slurm/R-demo]$ cat plot1.sh
#!/bin/bash
# Run the "R" program to read in the "plot1.r" script.

echo "`date`: Loading R Module"
module load R

echo "`date`: Running R Job"
R CMD BATCH plot1.r

echo "`date`: R job complete"
exit 0
[compute-151 slurm/class-scripts-slurm/R-demo]$
```

plot1.r creates plot1-R-results.pdf which you can view with xpdf or a web browser (firefox or chromium-browser)

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
$ exit                    # log out of your srun session
```

- Can also us “jhpce-rstudio-server-R4.3.0”

Accessing GPUs on SLURM

We have 1 GPU node currently under SLURM with 3 GPUs on the “gpu” partition. We will be adding more soon.

You can use “slurmpic –p gpu” to see current usage:

```
$ slurmpic -p gpu
NODENAME    NODESTATE  PARTITION CPUS_A/T CPU_LOAD  TOT_MEM FREEMEM SYSFREEMEM
compute-117      idle        gpu     0/ 48     0.06   376 GB  376 GB       80 GB
```

To access the GPU node and request any available GPU, you would run:

```
$ srun --pty --x11 --partition gpu --gpus=1 bash
```

If you need 2 GPUs for an interactive session, you would use the command:

```
$ srun --pty --x11 --partition gpu --gpus=2 bash
```

We have 2 GPUs types on this gpu node, and you can request a specific GPU with the “gres” option. To request an Nvidia V100 GPU, you would run:

```
$ srun --pty --x11 --partition sysadmin1 --gres=gpu:tesv100:1 bash
```

… and to request the Nvidia Titan V GPU, you would run:

```
$ srun --pty --x11 --partition sysadmin1 --gres=gpu:titanv:1 bash
```

Transferring files to the cluster

- Transfer results back

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

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

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

“How many jobs can I submit?”

Currently we don’t impose a limit on the number of jobs you can submit.

We do 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 1000 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 “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.

BETA TESTING JUPYTER LAB

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

https://jhpce-app02.jhsph.edu

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

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

Never run a job on the login node!

This has not changed in the new cluster. 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`**

Summary

- Review
  - Use **`ssh jhpce03.jhsph.edu`** to connect to new JHPCE cluster
  - Use sbatch and srun to submit jobs
  - Never run jobs on the login node
  - Helpful resources
    - https://slurm.schedmd.com/documentation.html
    - http://www.jhpce.jhu.edu/
    - bitsupport@lists.johnshopkins.edu - System issues
    - bithelp@lists.johnshopkins.edu - Application issues

Thanks for attending! Questions?