Running programs on HARDAC
Goals of this Class

• Understand
  • A cluster provides CPU, RAM, and Disk space.
  • Terms: Cluster, Partition, Node, Job, and Job Step
  • Job life-cycle

• Be able to
  • Run a job on the cluster
  • Run batch and **batch array jobs**
  • Monitor/cancel jobs
Laptop Not Powerful Enough

**Symptoms**

- projects **TOO BIG** for your hard drive
- processing many files takes **FOREVER**
- high RAM commands **CRASH**

**The Problem**

- 128 GB hard drive
- CPUs
- RAM
Cluster Has More Power

500,000 GB

CPUs

RAM

Cluster
Just a bunch of computers (nodes)
Slurm manages our cluster

Login Node

Worker Nodes
(Partition: all)

Worker Nodes
(Partition: interactive)

ssh

ssh

ssh
ssh to the Login Node

$ ssh <netid>@hardac-login.genome.duke.edu
...
...password:XXXXX
...
...hardac-login ~]$
hostname
what machine am I on?

Run hostname command

```bash
..login ~]$ hostname
hardac-login.genome.duke.edu
```

This command prints out the name of the machine we are running it on. In this case the login node.

**NOTE:** Do not run intensive commands on the login node.
**srun**

Slurm run a command in the foreground

Ask slurm to run the hostname command on a worker node

```bash
..login ~]$
$ srun hostname
srun: job 51 queued and waiting for resources
srun: job 51 has been allocated resources
c1-10-4.genome.duke.edu
..login ~]$
```
srun

Specify memory requirements

By default HARDAC allocates 2G memory per job.
Run hostname command specifying 4G of RAM (memory)

```
..login ~]$ srun --mem=4G hostname
srun: job 51 queued and waiting for resources
srun: job 51 has been allocated resources
c1-10-3.genome.duke.edu
..login ~]$ 
```

- Slurm will stop your job if you use more than the requested memory

- If you allocate too much memory it can take longer to get your job scheduled and wastes resources
Cluster nodes are grouped into partitions based. To specify a partition with `srun` use the `-p` flag. The default partition is named all.

Explicitly run `hostname` on a node in the all partition:

```
..login ~]$
```

```
...$ srun -p all hostname
```
Interactive Job

typing srun and waiting is tedious

Steps

1. Connect to Login Node

2. Start interactive job using `srun` on the interactive partition

3. Run whatever commands you want

4. Type `exit` to quit interactive job

```
...login ~]$ srun -p interactive --pty bash
<workernode> ~]$ hostname
```
Getting code onto the Cluster

Works just like on your laptop!


Change into this directory

$ cd scicomp-hpc

See the files we downloaded

$ ls
Foreground vs Background Jobs

Foreground Job - srun

- Useful for testing but not for long running commands
- Actively monitored through terminal output
- Canceled by pressing Ctrl-C or closing your terminal window

Background Job

- Useful for long running commands
- Monitored via log files, slurm commands, and email messages
- Canceled by using a slurm command
**sbatch**

Run command(s) in the background

Make a file called `countgc.sh` using nano:

```
#!/bin/bash
echo "Starting GC counter"
python fasta_gc.py data/E2f1_dna.fasta
```

Run it by using the `sbatch` command:

```
$ sbatch countgc.sh
Submitted batch job 26651766
```

When done Slurm will create an output file(s) based on jobid.

```
$ cat slurm-* .out
```
1. Slurm creates a Job in the Job Queue with status Pending when a user submits a request.

2. When resources are available Slurm will run Pending jobs. The job state is changed to Running.

3. When a job is finished Slurm removes it from the Job Queue. Slurm records the job in the Accounting List with the final state.

### Job Queue - squeue

<table>
<thead>
<tr>
<th>User</th>
<th>Cmd</th>
<th>State</th>
<th>Job ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Star Aligner</td>
<td>Running</td>
<td>123</td>
</tr>
<tr>
<td>John</td>
<td>fastqc…</td>
<td>Running</td>
<td>411</td>
</tr>
</tbody>
</table>

### Accounting List - sacct

<table>
<thead>
<tr>
<th>User</th>
<th>Cmd</th>
<th>State</th>
<th>Job ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>kalign…</td>
<td>Error</td>
<td>112</td>
</tr>
<tr>
<td>John</td>
<td>fastqc…</td>
<td>Complete</td>
<td>411</td>
</tr>
</tbody>
</table>
squeue shows active job status

Look at your active jobs.

```
$ squeue -u <netid>

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>6335778</td>
<td>all</td>
<td>long_ru...</td>
<td>jpb67</td>
<td>R</td>
<td>0:05</td>
</tr>
</tbody>
</table>

...```

**Job Status Column**

- R - Running
- P - Pending

Start a long running job the repeat the above command.

```
$ sbatch long_running.sh
```
scancel

Terminate a running Job

Find the job id of that long_running job.

```
$ squeue -u <netid>
```

Stop a single job

```
$ scancel <JOBID>
```

Or stop all jobs for your user

```
$ scancel -u <netid>
```

⚠️ will cancel interactive jobs
$ sacct

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>...</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>26705496</td>
<td>countgc.sh</td>
<td></td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>26705496.ba+</td>
<td>batch</td>
<td></td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>26705566</td>
<td>countgc.sh</td>
<td></td>
<td>FAILED</td>
<td>1:0</td>
</tr>
<tr>
<td>26705566.ba+</td>
<td>batch</td>
<td></td>
<td>FAILED</td>
<td>1:0</td>
</tr>
<tr>
<td>26706541</td>
<td>countgc.sh</td>
<td></td>
<td>CANCELED</td>
<td>0:0</td>
</tr>
<tr>
<td>26706541.ba+</td>
<td>batch</td>
<td></td>
<td>CANCELED</td>
<td>0:15</td>
</tr>
</tbody>
</table>

Only shows results from current day by default. Checkout `starttime` flag to see a better date range.
How much memory did that use?

```bash
$ sacct -o JobName,State,MaxRSS,ReqMem,Elapsed

<table>
<thead>
<tr>
<th>JobName</th>
<th>State</th>
<th>MaxRSS</th>
<th>ReqMem</th>
</tr>
</thead>
<tbody>
<tr>
<td>countgc.sh</td>
<td>COMPLETED</td>
<td>2Gc</td>
<td></td>
</tr>
<tr>
<td>batch</td>
<td>COMPLETED</td>
<td>4960K</td>
<td>2Gc</td>
</tr>
<tr>
<td>countgc.sh</td>
<td>COMPLETED</td>
<td>400Mn</td>
<td></td>
</tr>
<tr>
<td>batch</td>
<td>COMPLETED</td>
<td>4936K</td>
<td>400Mn</td>
</tr>
</tbody>
</table>
```

- MaxRss / 1024 = MB for use with `sbatch --mem`
- See all options sacct can show: `sacct -e`
sbatch
memory requirements

Change countgc.sh using nano:

```
#!/bin/bash
#SBATCH --mem=400M
python fasta_gc.py data/E2f1_dna.fasta
```

The `#SBATCH` comment tells sbatch to pretend that the following flag was passed along the command line. This is preferable to typing the flags again and again.

`srun` and `sbatch` commands share many of the same arguments.
### sbatch

Email when job completes

Add two lines `countgc.sh` using nano:

```bash
#!/bin/bash
#SBATCH --mail-type=END
#SBATCH --mail-user=<your_email_address>
#SBATCH --mem=400M
echo "Starting GC counter"
python fasta_gc.py data/E2f1_dna.fasta
```

Run it with `sbatch`:

```
$ sbatch countgc.sh
```
job steps
break job into steps

Create jobsteps.sh using nano:

```bash
#!/bin/bash
FILENAME=data/E2f1_dna.fasta
srun cksum $FILENAME
srun python fasta_gc.py $FILENAME
```

Run our sbatch script

```
$ sbatch jobsteps.sh
```

Once it finishes look at

```
$ sacct
```
sbatch --array
make a bunch of jobs

Create arraytest.sh using nano:

```bash
#!/bin/bash
#SBATCH --mem=400M
#SBATCH --array=1-5%2
echo $SLURM_ARRAY_TASK_ID
```

The **1-5** part says to run array_test.sh script 5 times with **SLURM_ARRAY_TASK_ID** filled with a number 1-5. The **%2** part says to only run 2 at a time.

```bash
$ sbatch arraytest.sh
```
sbatch --array
use task id to find a filename

Change arraytest.sh using nano:

```bash
#!/bin/bash
#SBATCH --mem=400M
#SBATCH --array=1-5%
IDX=$SLURM_ARRAY_TASK_ID
FILENAME=$(ls data/*.fasta | awk NR==$IDX)
echo $FILENAME
```

Run your array job

```
$ sbatch arraytest.sh
```
Change arraytest.sh using nano:

```bash
#!/bin/bash
#SBATCH --mem=400M
#SBATCH --array=1-5%
IDX=$SLURM_ARRAY_TASK_ID
FILENAME=$(ls data/*.fasta | sed -n ${IDX}p)
python fasta_gc.py $FILENAME
```

This script will determine GC of 5 files in the `data` directory storing result into separate `slurm*.out` files.

```
$ sbatch arraytest.sh
```
How busy is the cluster?

Show status of nodes in the "all" partition

```
sinfo -p all
PARTITION AVAIL  TIMELIMIT  NODES  STATE  NODELIST
all    up    90-00:00:0  2    idle   c1-02-[1-4]
all    up    90-00:00:0  1    down   c1-09-3
all    up    90-00:00:0  1    mix     c1-09-[1-2],c1-10-4
all    up    90-00:00:0  3    alloc  x2-01-3,x2-07-3
```
Proper Job Allocations

Only use up to 2 nodes worth of compute resources at a time

Total Memory: 512GB
Total CPUs: 64
Do not run jobs on the login node!

Login Node

Worker Nodes

Your jobs run here slowly 😞

Other users can't start their jobs 😞

Instead of here which would be fast
Helpful Resources

HARDAC WIKI
https://wiki.duke.edu/display/HAR/

Requesting Software/Help
gcb-help@duke.edu