Introduction to HPC @ RCC
September 11, 2018
Research Computing Center
What is HPC

“High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business”
How to allocate resources?
Job scheduler
Partitions?

- Collection of nodes
- Public (general access) and owner nodes
- Access is granted through a unix group
- Partitions spawn different architectures
  - Owner has bought nodes from different years
  - Jobs can not spawn different architectures
1. Login at https://acct.rcc.fsu.edu/account
## Request access to Partition

### 2. Request membership to partition

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Queue</th>
<th>Start Time</th>
<th>End Time</th>
<th>Users</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC4M Partition</td>
<td>sec4m_q</td>
<td>2160:00:00</td>
<td>336:00:00</td>
<td>0</td>
<td>500</td>
</tr>
<tr>
<td>Engineering Partition</td>
<td>engineering_q</td>
<td>06:00:00</td>
<td>06:00:00</td>
<td>0</td>
<td>512</td>
</tr>
<tr>
<td>Engineering Long Partition</td>
<td>engineering_long</td>
<td>48:00:00</td>
<td>48:00:00</td>
<td>0</td>
<td>280</td>
</tr>
<tr>
<td>RCC Internal Partition</td>
<td>rcc_internal</td>
<td>72:00:00</td>
<td>36:00:00</td>
<td>0</td>
<td>512</td>
</tr>
<tr>
<td>Bleiholder Queue</td>
<td>bleiholder_q</td>
<td>2160:00:00</td>
<td>336:00:00</td>
<td>0</td>
<td>144</td>
</tr>
<tr>
<td>Deprince Queue</td>
<td>deprince_q</td>
<td>2160:00:00</td>
<td>336:00:00</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>Statistcs Queue</td>
<td>statistics_q</td>
<td>2160:00:00</td>
<td>336:00:00</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>Shangchao Lin Queue (AME)</td>
<td>lin_q</td>
<td>2160:00:00</td>
<td>336:00:00</td>
<td>0</td>
<td>208</td>
</tr>
</tbody>
</table>
How to submit a job

1. Command line
   ▪ Ssh to hpc-login.rcc.fsu.edu
   ▪ Use srun/sbatch

2. Web interface (script generator)
   ▪ Currently only generates scripts and you have to use (1) to submit the script
How to submit a job (ssh)

1. **sbatch**
   non-interactive batch submission
   schedules job in background

2. **srun & salloc**
   interactive submission
   srun/salloc run program in foreground
   srun can also be used in batch script!
Submit jobs: sbatch

sbatch \{flags\} myscript

- man sbatch
- sbatch -p myqueue -n 10 myscript
  - request 10 cores from the myqueue queue and run \textit{myscript} job script
- sbatch myscript
  - request 1 core from my default queue
- sbatch -D myproject/workdir myscript
  - start job in \$HOME/myproject/workdir folder
srun to submit a job

- man srun
- srun from a submit node will start a new job
  - srun -p myqueue myprogram
- will not run in the background (unless &)
- srun -n x myscript.sh will start x instances of myscript.sh
  - srun will not “interpret” scripts: ignore #SBATCH flags
- The salloc is similar to srun, but be careful!
slurm enabled replacement of mpirun
- mpirun is no longer supported (mvapich2)
- srun myprogram
  - will run myprogram on requested number of cores (sbatch -n x)
- srun -n y myprogram
  - will run myprogram on y number of cores
  - error if y>x (sbatch -n x)
- be careful when you use srun in a script submitted by srun
Interactive jobs --pty

sr
un --pty someprogram
sr
un --pty /bin/bash
sr
un --pty R
sr
un --pty gdb myprogram

- sr
un -n x --pty program will start 1 instance
- sr
un will start from your submit directory
<table>
<thead>
<tr>
<th><strong>SLURM</strong></th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbatch</td>
<td>sbatch -p myqueue myjobscript.sh</td>
<td>Submit a batch script</td>
</tr>
<tr>
<td>srust &amp; s alloc</td>
<td>srun myprogram.exe s alloc myprogram.exe</td>
<td>Submit an interactive program</td>
</tr>
<tr>
<td>s queue</td>
<td>s queue -p mypartition</td>
<td>Show jobs in a mypartition</td>
</tr>
<tr>
<td>s queue</td>
<td>s queue -j 1251</td>
<td>Inspect a specific job</td>
</tr>
<tr>
<td></td>
<td>s control show job 1251</td>
<td></td>
</tr>
<tr>
<td>s queue</td>
<td>s queue -j 1252 --start</td>
<td>Show start time of job</td>
</tr>
<tr>
<td>scancel</td>
<td>scancel 1251</td>
<td>Cancel a job</td>
</tr>
<tr>
<td>s info</td>
<td>s info -p mypartition</td>
<td>Shows nodes in my partition</td>
</tr>
</tbody>
</table>

https://rcc.fsu.edu/docs/hpc-cheat-sheet
s* caveats

- Jobs will start in the current working directory (unless -D flag was used)
- Job environment is a copy of your working environment (except for limits)
  - environment variables
  - be careful what modules you autoload in your ~/.bashrc
- sbatch is not for interactive jobs
Common flags for s*

- `-n number` : Request *number* of cores
- `-p partition` : Run a job on this queue
- `-C feature` : Restrict job to nodes with this feature
- `--exclusive` : Do not share nodes with other jobs
- `-J jobname` : job name (not outputfile)
- `-o outputfile` : output file (default slurm)
- `--mail-type=X` : receive this type of notifications
  (ALL, BEGIN, END, FAIL)
Less Common flags

- `--begin=time` : Start a job at time `time`
- `--output=slurm.%N.%j.out` : output log
- `--input=inputfile.txt` : use text from file for std input
- `--pty` : interactive job, only for srun!
Memory

- Slurm takes memory in consideration
- Default is 4GB per core (2GB backfill{2})
- `--mem-per-cpu=<MB>` or `--mem=<MB>`
- Under the hood: memory is “mapped” to cores:
  - `-n 1 --mem=5GB` will reserve 2 cores on a node.
- Memory limit is enforced
#!/bin/bash

#SBATCH -J MYJOBNAME
#SBATCH -n 10

module load gnu-openmpi

pwd

srun myprogram
Script Generator

https://rcc.fsu.edu/submit-script-generator

- Interactively generate a slurm script
- Limited syntax checking
- Templates available for some software
- Submit jobs directly from website (future)
Script Generator Demo

Job Title
MyProgram

Create a name for your job (alphanumeric, dashes, and spaces allowed)

Executable Call
test

Enter the program you wish to run for your job. If you pipe input or pass arguments, include those.

Separate Results and Verbose Output
- Separate Verbose Output from Results

Email Notifications
- On Job Start
- On Job End
- If Job Fails
- If Job Requeues

Please select the type of email notifications about your job you would like to receive.

HPC Partition
Backfill 2 Queue (backfill2) – General Access

Number of Cores
4

Select the number of processor cores your job will run on.

Number of Nodes
2

Select the number of compute nodes your job will run on. Note that adjusting this does not guarantee that processes will be evenly distributed across all nodes. The default is "No Preference" and the Number of Processes is adjusted instead. If Number of Nodes is set to "No Preference" then the numbers specified above are ignored.

SLURM Submission Script

#!/bin/bash
#SBATCH --job-name=MyProgram

#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH -n 4

#SBATCH -N 2
#SBATCH -p backfill2
#SBATCH -t 04:00:00

module load gnu-openmpi/2.1.0

## Submit Script Generator automatically addresses
srun test

To use this script:
Why is my job not running?

- Partition does not have enough cores available?
- You ask for too much memory?

```
squeue -u $(whoami)
squeue -p mypartition
scontrol show job jobid
```
Submit a request to support@rcc.fsu.edu
- Include the path to your job script and output files
- Include the error you received
- If possible, include job id.
Job arrays are a way to efficiently submit large numbers of jobs.

Single program with a lot of different datasets

`sbatch --array=1-10 program.sh`

- `$SLURM_ARRAY_TASK_ID`
Job dependencies

- Scheduling of job is conditional
- For example, a job can only run when another job has finished
  - `#sbatch --dependency=afterok:otherjobid`
- For example, job can only run when no job of the same “type” (name) runs
  - `#sbatch --dependency=singleton`
  - `#sbatch --job-name=jobname`
Shared MPI space

- Share MPI communicator space with multiple programs
- Define cpu mapping in `layoutfile`
  
  0-7  ./prog1
  8-15 ./prog2

- `srun -n16 --multi-prog layoutfile`