Introduction to HPC @ RCC

September 14, 2016

Research Computing Center
“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”
Typical HPC Workflow

High-throughput Instruments

Tier 1 Storage
X200

Tier 2 Storage
IQ NL-Series

Health Records
Experimental Data
Compound Libraries

HPC Compute Cluster

Tape Library

Local Researchers and Analysts

Remote Researchers and Analysts

WAN
How to allocate resources?
Job scheduler
Partitions?

- Collection of nodes
- Access is granted through an account
- Users can run jobs on “their” account
- Spawn different architectures (eg AMD)
  - Jobs can not spawn different architectures
- Similar to Queues
  - Direct mapping account - partition (RCC)
Nuts and Bolts

- Submit job to a partition.
  - partitions are managed by RCC staff
- Membership to accounts determines who can submit to which partition.
  - accounts are managed by ‘owners’
- Feature (--constraint) determines where the job will run. Default: any.
# Commands

<table>
<thead>
<tr>
<th>SLURM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbatch</td>
<td>Submit a batch script</td>
</tr>
<tr>
<td></td>
<td>sbatch -p myqueue myjobscript.sh</td>
</tr>
<tr>
<td>srun &amp; salloc</td>
<td>srun myprogram.exe</td>
</tr>
<tr>
<td></td>
<td>salloc myprogram.exe</td>
</tr>
<tr>
<td>squeue</td>
<td>Submit an interactive program</td>
</tr>
<tr>
<td></td>
<td>squeue -p mypartition</td>
</tr>
<tr>
<td>squeue</td>
<td>Show jobs in a mypartition</td>
</tr>
<tr>
<td></td>
<td>squeue -j 1251</td>
</tr>
<tr>
<td></td>
<td>scontrol show job 1251</td>
</tr>
<tr>
<td>squeue</td>
<td>Inspect a specific job</td>
</tr>
<tr>
<td></td>
<td>squeue -j 1252 --start</td>
</tr>
<tr>
<td>scancel</td>
<td>Show start time of job</td>
</tr>
<tr>
<td></td>
<td>scancel 1251</td>
</tr>
<tr>
<td>sinfo</td>
<td>Cancel a job</td>
</tr>
<tr>
<td></td>
<td>sinfo -p mypartition</td>
</tr>
<tr>
<td></td>
<td>Shows nodes in mypartition</td>
</tr>
<tr>
<td></td>
<td>sinfo -p mypartition</td>
</tr>
</tbody>
</table>

[https://rcc.fsu.edu/docs/hpc-cheat-sheet](https://rcc.fsu.edu/docs/hpc-cheat-sheet)
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
How to submit a job

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 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 vs salloc for submission

srun -n X program
salloc -n X program

- both will allocate X cores
  - srun will start program X times
  - salloc will start 1 instance program
srn to submit a job

- man srn
- srn from a submit node will start a new job
  - srn -p myqueue myprogram
- will not run in the background (unless &)
- srn -n x myscript.sh will start x instances of myscript.sh
  - srn will not “interpret” scripts: ignore #SBATCH flags
sr

slurm enabled replacement of mpirun
mp

mpirun is no longer supported (mvapich2)
srun myprogram
  will run myprogram on requested number of cores
    (sbatch -n x)
srun -n y mypro
    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
s* caveats

- Jobs will start in the current working directory (unless -D flag was used)
  - moab: job will always start in home directory
- 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!
Submit a job

sbatch -p bio_q mywrf.sh
srun -p cob_q --constraint=AMD XYZ.exe
sbatch -p yang_q,bio_q job.sh
sbatch -o myjob.%j.out myjob.sh
srun --pty /bin/bash
Interactive jobs --pty

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

- sr
  -n x --pty program will start 1 instance
- sr
will start from your submit directory
#!/bin/bash

#SBATCH -J MYJOBNAME
#SBATCH -n 10

module load gnu-openmpi

pwd

srun myprogram
Run a sequential program

```bash
#!/bin/bash

#SBATCH -J MY-R-CODE
#SBATCH --input myRinput.txt

pwd
module load R
R --no-save
```
Job Array

- 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`
Disclaimer

- There are 2 sites about slurm. One is outdated:
  - computing.llnl.gov: original project site
    - refers to version 2.3
  - http://www.schedmd.com: correct website

- Don’t use mpirun, use srun
  - mpirun still available for openmpi
  - both openmpi and mvapich2 support srun
#SBATCH -n 4
srun -n 5 myprogram

srun: error: Unable to create job step: More processors requested than permitted
Errors

srun --constraint "X&Y" myprogram

srun: error: Unable to allocate resources: Requested node configuration is not available