HPC Job Submission

HPC Job Submission

Andy Monaghan, Shelley Knuth, Dan Trahan, Joel Frahm

rc-help@colorado.edu
https://www.colorado.edu/rc

Sign in! http://tinyurl.com/curc-names

Slides available for download at
https://github.com/ResearchComputing/CU_DENVER_HPC_2019

Adapted from presentations by RC members Andrew Monaghan, Aaron Holt and John Blaas: 1, 2, 3, 4.

Outline

- General Info
- Examples of submitting jobs to the supercomputer!
  - Simple batch jobs
  - Advanced batch jobs: running programs, mpi
  - Interactive jobs

Hardware: RMACC Summit Supercomputer

- 450 compute nodes (Intel Xeon Haswell)
- 24 cores per node
- 11,400 total cores
- Omni-Path network
- 1.2 PB scratch storage
- GPFS File system

- 67% CU, 23% CSU, 10% RMACC

Additional Types of RMACC Summit Compute Nodes

- 10 Graphics Processing Unit (GPU) Nodes
  - NVIDIA Tesla K80 (2/node)
- 5 High Memory Nodes
  - 2 TB of memory/node, 48 cores/node
- 20 Phi Nodes
  - Intel Xeon Phi
  - 68 cores/node, 4x threads/core

RC Access: Logging in

- If you have an RMACC account already, login as follows from a terminal:
  - `$ ssh janedoe@login.xsede.org` *["janedoe"= your xsede username]*
    - *[enter xsede passcode and accept duo push or call to phone]*
    - `[janedoe@ssohub ~]$ gsissh rmacc-summit`
- If you do not have an RMACC account use one of our temporary accounts:
  - `$ ssh userNNNN@tlogin1.rc.colorado.edu`
    - *[enter temporary password provided by me]*

Working on RC Resources

- When you first log in, you will be on a login node. Your prompt will look like this (e.g.):

```
[user0049@tlogin1 ~]$
```

- The login nodes are lightweight virtual machines primarily intended to serve as ‘gateways’ to RC resources. If you plan to work on Summit (most will), your first step should always be to move to a Summit ‘scompile node’:

```
[user0049@tlogin1 ~]$ ssh scompile
```

- Now go to your working directory and download the material for this workshop:

```
[user0049@shas0137 ~]$ cd /scratch/summit/$USER

[user0049@shas0137 ~]$ git clone
https://github.com/ResearchComputing/CU_DENVER_HPC_2019
```

Useful Slurm Commands: sbatch

- **sbatch**: submit a batch job to slurm
- Submit your first job! :
- `$ sbatch submit_test.sh`

http://slurm.schedmd.com/sbatch.html

Anatomy of a job script (submit_test.sh)

```
#!/bin/bash
#SBATCH --ntasks=1                          # Number of requested tasks
#SBATCH --time=0:01:00                      # Max wall time
#SBATCH --partition=shas-testing            # Specify Summit Haswell nodes
#SBATCH --output=test_%j.out                # Rename standard output file

# Updated by:   Shelley Knuth, 17 May 2019
# Purpose:   To demonstrate how to run a batch job on RC resources

# Purge all existing modules
module purge

# Run commands
echo "This is a test of user $USER"
```

SBATCH Options

Specified at command line or in job script as…

`#SBATCH <options>` …where options include:

- Allocation: `--account=<account_name>`
- Partition: `--partition=<partition_name>`
- Sending emails: `--mail-type=<type>`
- Email address: `--mail-user=<user>`
- Number of nodes: `--nodes=<nodes>`
- Number of Tasks `--ntasks=<number-of-tasks>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall time>`
- Job Name: `--job-name=<jobname>`

More on slurm commands: https://slurm.schedmd.com/quickstart.html

FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job

Available Partitions (--partition)

Partition (sub-partition)	Description	# of nodes	cores/node	GPUs/node
shas (shas-testing) (shas-interactive)	General Compute (Haswell)	~450	24	0
sgpu (sgpu-testing)	GPU-enabled nodes	10	24	effectively 4
smem	High-memory nodes	5	48	0
sknl (sknl-testing)	Phi (Knights Landing) nodes	20	68	0

Sub-Partitions

Partition	Description	Max wall time	Max jobs/user	Max nodes/user
shas-testing sgpu-testing smem-testing	For quick turnaround when testing	30 M	1	2 12 cores/node
shas-interactive	For interactive jobs (command or GUI)	4 H	1	1 core

Quality of Service (--qos)

QoS	Description	Max wall time	Max jobs/user	Max nodes/user
normal	Default QoS	Derived from partition	n/a	256
long	For jobs needing longer wall times	7 D	n/a	20

Practice Job Submission Examples

Write your first job script!

- Create a Slurm job script and submit it as a job, with the following instructions:

1. Name it 'submit_sleep.sh'

2. The job should contain the following commands:

```
echo "Running on host" `hostname`
echo "Starting Sleep"
sleep 30
echo "Ending Sleep. Exiting Job!"
```

Details on job script parameters are in the next slide

Details of submit_sleep.sh

1. The job will run on 1 core of 1 node
2. We will request a 1 minute wall time
3. Run on the shas-testing partition
4. Set the output file to be named “sleep_ID.out”
5. Name your job “sleep”
6. Bonus: Email yourself when the job ends
7. Contains the following commands →
8. Submit using the ‘tutorial1’ reservation *(This is only for this workshop):*

   $ sbatch --reservation=tutorial1 submit_sleep.sh

```
echo "Running on host" `hostname`
echo "Starting Sleep"
sleep 30
echo "Ending Sleep. Exiting Job!"
```

Solution can be found in “./solutions” subdirectory

Running and monitoring jobs

Submit the job:

```
$ sbatch submit_hostname.sh
```

Check the status of the job:

```
$ squeue / $ squeue -u <user> / $ squeue -q <qos>
```

…or

```
$ sacct / $ sacct --format=<options>
```

…or

```
$ scontrol show job <job number>
```

Look at the job output:

```
$ more hostname_NNNNNN.out
```

*(*note that NNNNNN is your job number)*

More on slurm commands: https://slurm.schedmd.com/quickstart.html

Running an external program

- Let's run R on an R script
- This script calls and runs the script *R_program.R*
- Let's examine the batch script submit_R.sh
  - Note how R is loaded
- Go ahead and submit the batch script submit_R.sh

Running an external program as an mpi job

- For cases where you have a code that is parallelized, meaning it can run across multiple cores.
- Number of tasks always > 1. E.g.,

```
--ntasks=4
```

- Will always need to load a compiler and mpi. E.g.,

```
module load intel impi
```

- Executable preceded with mpirun, srun, or mpiexec. E.g.,

```
mpirun -np 4 python yourscript.py
```

- Examine and run the example 'submit_python_mpi.sh'

```
$ sbatch --reservation=tutorial1 submit_python_mpi.sh
```

Interactive jobs

- Sometimes we want our job to run in the background
- Sometimes we want to work on program in real time
    - Great for testing, debugging
- For example, let's run the R job we previously ran as a batch job, but this time let's do it interactively…

Running an interactive job

- To work with R interactively, we request time from Summit
- When the resources become available the job starts
- Commands to run:

```
$ sinteractive –-time=00:10:00 --reservation=tutorial1
```

- Once we receive a prompt, then:

```
$ module load R
$ cd ./progs
$ Rscript R_program.R
```

- Once we finish we must exit! (job will time out eventually)

```
$ exit
```

Tools for submitting 100s or 1000s of “tiny” jobs all-at-once

- GNU Parallel
  - https://curc.readthedocs.io/en/latest/software/GNUParallel.html
- CURC Load Balancer
  - https://curc.readthedocs.io/en/latest/software/loadbalancer.html
- Slurm job arrays
  - https://slurm.schedmd.com/job_array.html

Thank you!

- Please fill out the survey: http://tinyurl.com/curc-survey18
- Sign in! http://tinyurl.com/curc-names
- Contact information: rc-help@Colorado.edu
- Slides and Examples from this course: https://github.com/rctraining/HPC_Short_Course_Fall_2018
- Slurm Commands: https://slurm.schedmd.com/quickstart.html
- Load Balancer Tool: https://curc.readthedocs.io/en/latest/software/loadbalancer.html