HPC Job Submission and Load Balancing

Andrew Monaghan
andrew.monaghan@Colorado.edu
https://www.rc.colorado.edu

Slides available for download at
https://github.com/rctraining/HPC_Short_Course_Spring_2018

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

Outline

- General Info
- Examples of submitting jobs to the supercomputer!
  - Simple Batch job
  - Advanced Batch job: running programs or scripts
  - Interactive job
  - Load balancing

Hardware: Summit Supercomputer

475 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 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

Phi Nodes
- 20 nodes
- Intel Xeon Phi

RC Access: Logging in

- For this tutorial, we will be using accounts on RC resources
- In a terminal or Git Bash window, type the following:

```
ssh –X userNNNN@tutorial-login.rc.colorado.edu

Password:
```

***Note that `userNNNN` is a temporary account that I have assigned to you. If you don’t have one, let me know.*

Working on RC Resources

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

```
[monaghaa@login04 ~]
```

- 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':

```
[monaghaa@login04 ~]ssh scompile
[monaghaa@shas0137 ~]$
```

- Now go to the directory for this workshop:

```
$ cd ~/Jobs_HPC18
```

Useful Slurm Commands: sbatch

- **sbatch**: submit a batch script to slurm
- You can use a bunch of flag options in a batch script or on the command line
- Useful to put in script so have for future use
- Example:

```
 sbatch test.sh
```

OR

```
  sbatch --partition=shas test.sh
```

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

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>`

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

Available Partitions *(--partition)*

Partition	Description	# of nodes	cores/node	GPUs/node
shas	General Compute (Haswell)	380	24	0
sgpu	GPU-enabled nodes	10	24	effectively 4
smem	High-memory nodes	5	48	0
sknl	Phi (Knights Landing) nodes	20	68	0

Quality of Service *(--qos)*

QoS	Description	Max wall time	Max jobs/user	Max nodes/user
normal	Default QoS	Derived from partition	n/a	256
debug	For quick turnaround when testing	1 H	1	32
long	For jobs needing longer wall times	7 D	n/a	20
condo	For groups contributing to the Summit condo	7 D	n/a	n/a

Practice Examples

Submit Your First Job!

- Submit a slurm job with the following instructions:

1. The job should run the Linux “hostname” command
2. The job will be submitted from a bash script named submit_hostname.sh
3. The job will run on 1 core of 1 node
4. We will request a 1 minute wall time
5. Run from the debug QOS
6. Run on the shas partition
7. Use the ‘tutorial1’ reservation
   - *This is only for this workshop*

submit_hostname.sh

```
#!/bin/bash
#SBATCH --nodes=1                  # Number of requested nodes
#SBATCH --ntasks=1                 # Number of requested nodes
#SBATCH --time=0:01:00             # Max wall time
#SBATCH --qos=debug                # Specify QOS
#SBATCH --partition=shas           # Specify Summit haswell nodes
#SBATCH --output=hostname_%j.out # Rename standard output file
#SBATCH --reservation=tutorial1  # Reservation (workshop only)

# Written by:  Shelley Knuth, 15 July 2016
# Updated by:  Andy Monaghan, 8 March 2018
# Purpose:     Demonstrate how to run a batch job on RC resources

# purge all existing modules
module purge

hostname
```

Running the script

Submit the job:

```
$ sbatch submit_hostname.sh
```

Check the status of the job:

```
$ sacct
```

…or

```
$ squeue
```

Look at the job output:

```
$ more sbatch hostname_NNNNNN.out
```

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

Slurm command: squeue

- **squeue**
  - View information about jobs located in the slurm scheduling queue
- OPTIONS:
  - User: `-u <user_list>`
  - Queues: `-q <qos_list>` or `--qos=<qos_list>`

- EXAMPLE:

```
squeue --qos=debug
```

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

Your turn

- 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 run first the whoami command, then the Linux “sleep” command for 30 seconds, then the hostname command
   - Syntax for these Linux commands is:

```
whoami
sleep 30
hostname
```

Submit_sleep.sh

```
#!/bin/bash
#SBATCH --nodes=1                          # Number of requested nodes
#SBATCH --ntasks=1                         # Number of requested tasks
#SBATCH --time=0:01:00.                    # Max walltime
#SBATCH --qos=debug                        # Specify QOS
#SBATCH --partition=shas                   # Specify Summit haswell nodes
#SBATCH --output=sleep_%j.out              # Rename standard output file
#SBATCH --job-name=sleep                   # Job submission name
#SBATCH --mail-type=end.                   # Email you when the job ends
#SBATCH --mail-user=<user>@colorado.edu # Email address to send to
#SBATCH --reservation=tutorial1            # Reservation (workshop only)


# purge all existing modules
 module purge

  whoami
  sleep 30
  hostname
```

Running an external program or script

- Let's run a Matlab program
- We will run the batch script submit_matlab.sh
- This script calls and runs matlab_tic.m

Running the script

- Submit the job:

```
sbatch submit_matlab.sh
```

- Check output

submit_matlab.sh

```
#!/bin/bash
#SBATCH --nodes=1                       # Number of requested nodes
#SBATCH --ntasks=1                      # Number of requested tasks
#SBATCH --time=0:02:00                  # Max walltime
#SBATCH --qos=debug                     # Specify debug QOS
#SBATCH --partition=shas                # Specify Summit haswell nodes
#SBATCH --output=matlab_%j.out          # Output file name
#SBATCH --reservation=tutorial1         # Reservation name

# purge all existing modules
module purge

# Load Matlab module
module load matlab

# Run matlab without a GUI
 cd progs
 matlab -nodisplay -nodesktop -r "clear; matlab_tic;"
```

Your turn

- Submit a slurm job with the following instructions:

1. Create an R program called `R_program.R` that creates a vector called "planets" and then list the planets in the vector. Syntax:
    - `planets -> planets <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto")`
2. Print off the vector. Syntax:
    - `Planets`
3. Create a bash script called `submit_R.sh` that runs the R script (hint: Copy and modify the matlab job script you just ran)
4. In the script, you'll run R with the following syntax:
    - `Rscript R_program.R`
5. Don't forget to load the R module before the `Rscript` command!

Solution: submit_R.sh

```
#!/bin/bash
#SBATCH --nodes=1                           # Number of requested nodes
#SBATCH --ntasks=1                          # Number of requested tasks
#SBATCH --time=0:02:00                      # Max walltime
#SBATCH --qos=debug                         # Specify debug QOS
#SBATCH --partition=shas                    # Specify Summit haswell nodes
#SBATCH --output=R_code_%j.out              # Output file name
#SBATCH --reservation=tutorial1             # Reservation name

# purge all existing modules
module purge

# Load the R module
module load R

# Run R script
Rscript progs/R_program.R
```

Solution: R_program.R

```
#Simple R code example by (your name/email)

# Create vector
planets <- c("Mercury", "Venus", "Earth",  "Mars", "Jupiter",
"Saturn", "Uranus", "Neptune", "Pluto")

# Print off vector
planets
```

Interactive jobs!

- Sometimes we want our job to run in the background
- Sometimes we want to work in program in real time
- For example, R
- Let's run an interactive R job

Interactive job

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

```
sinteractive --ntasks=1 --reservation=tutorial1
```

- Once we receive a prompt, then:

```
module load R
Rscript R_program.R
```

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

CURC Load Balancer

- Useful when you have 100s to 100,000s of “tiny” serial jobs (<5 min)
- Instead of running them all as separate jobs, wrap into one job
- This will save you time and SUs, as it reduces start-up overhead
- To use (within batch job script): `module load loadbalance`
- Let’s take a look at the example in the documentation:
  - *https://github.com/ResearchComputing/Research-Computing-User-Tutorials/wiki/The-Load-Balancer-Tool*

Thank you!

Please fill out the survey:
http://tinyurl.com/curc-survey16

My contact information:
Andrew.Monaghan@Colorado.edu

Additional learning resources:
Slides and Examples from this course:
https://github.com/rctraining/HPC_Short_Course_Spring_2018

Slurm Commands:
https://slurm.schedmd.com/quickstart.html

Load Balancer Tool:
https://github.com/ResearchComputing/Research-Computing-User-Tutorials/wiki/The-Load-Balancer-Tool