Research Computing New User Seminar

Shelley Knuth
shelley.knuth@colorado.edu

www.rc.colorado.edu

Slides: https://github.com/ResearchComputing/New_User_Seminar

Outline

- What is RC?
- Steps to get access to our systems
  - Accounts
  - Two-factor authentication
  - Allocations
  - Logging in
- Navigating our systems
  - Storage spaces
  - Data transfer - Globus
  - Software
- Running jobs

What is Research Computing?

- Provide services for researchers that include:
  - Large scale computing
  - Data storage
  - High speed data transfer
  - Data management support
  - Consulting
  - Training

- We are likely best known for:
  - Janus/Summit
  - PetaLibrary

What Would I Use Summit For?

- Research Computing is more than just Summit
- But it is what we are most known for

- So what would you use Summit For?
  - Solving large problems that require more:
    - Memory than you have on your personal computer
    - Cores/nodes/power than you have on your personal computer
  - Large visualization jobs
  - High memory jobs
- Not a place for:
  - Large data storage

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

How To Access RC Resources?

1. Get an account
2. Set up two factor authentication
3. Set up an allocation
   - Don’t need for Blanca or PetaLibrary
4. Log in
5. Create greatness

- After you login, you will need to do many additional things that we will discuss today

Getting an RC Account

- CU Boulder users and affiliates:
- Request an account through the RC Account request portal
  - https://portals.rc.colorado.edu/accounts/account-request/create

- CSU Users:
  - Request an CSU eID if you don't have one
  - Fill out account application form
  - Duo authentication
  - Then get an RC user account
  - https://www.acns.colostate.edu/hpc/summit-get-started/

- RMACC Users:
  - Login through XSEDE
    - Need XSEDE account and Duo access through XSEDE
  - Contact us to start process

Setting up Two-Factor Authentication

- Two factor authentication is required to access our system
- Require this to provide an extra level of authentication
- Two methods for achieving this:
  - Duo
    - Access through a smart phone app
  - Vasco OTP (one time password)
    - CU only
    - Physical device

Duo Authentication

- Once you get an account, contact rc-help@colorado.edu to request a Duo invitation
- Once you get the invitation, you'll get a series of steps to complete Duo enrollment
- RC supports Duo "push" and "phone call" for authentication
- Greatly prefer "push"

Vasco Authentication

- Physical device that generates a new password every 30 seconds
- To get the device, go to the IT Service Center or request a time to pick one up from Research Computing
- You will need to show an ID to get a device

- First, register the device
  - Otp.colorado.edu
- You will set a four digit password

Allocations

- You will need a compute allocation to use any of our resources
- Currently, to request an allocation please email rc-help@colorado.edu and ask for a General allocation
  - Need to provide a few sentences on your project
- In the future, we will have a place on our website to submit a more formal request
- Once you have some benchmarks, you will want to move to a project allocation

Why Do I Need An Allocation?

- I have an account – why do I need an allocation?
  - An account validates you are eligible to use RC resources
  - An allocation allows us to keep track of your use of the system
  - This is important because:
    - We need to make sure we have enough resources to accommodate all of our users
    - Helps for reporting to NSF and the CU Research & Innovation Office
  - Applying for an allocation beyond a general allocation:
    - Gives you higher priority in the system

What is Fair Share?

- Fair share scheduling uses a complex formula to determine priority in queue
- Looks at load for each user and each QOS and balances utilization to fairly share resources
  - Involves historical use by user plus how long job has been in the queue

- System will first look at weighted average utilization of user over last 4 weeks
- Then compare it to the fair share target percentage of a user

Fair Share Target Percentage

- The target percentage depends on your priority based on your project proposal
- Everyone not associated with a project shares a target percentage of 13% (20% of the CU fraction)
  - No guaranteed level per user
- If you are under (over) your target percentage (based on a 4 week average) your priority is increased (decreased)
- Reminder this all only impacts pending jobs
- If no other pending jobs and enough resources are available then your job will run regardless of your previous usage

Allocations

- Need an allocation? Plan to run on Summit?
- Make a request now!
- Include 2-3 sentences describing your proposed usage
- Email rc-help@colorado.edu

Logging In

- It's important to note that you are NOT logging into any specific resource
  - Summit, etc
- When you log in, you land on our login nodes
- From there, you can access our other resources

RC Resource Access

- To login to an RC login node:

```
ssh username@login.rc.colorado.edu
```

- If logging in with Duo, your enter your password as:

```
duo:identikey_password
```

- If logging in with Vasco, your enter your password as:

```
Pin+six-digit number on Vasco
```

Navigating our Systems

- Now that you've logged in, now what?
  - What are the different node types we have?
  - What are the different storage spaces?
    - What should I be putting in these storage spaces?
  - How do I transfer data around?
  - How do I deal with software?

Different Node Types

- Login nodes
  - Four virtual machines
  - This is where you are when you log in
  - No heavy computation, interactive jobs, or long running processes
  - Script or code editing
  - Job submission
- Compile nodes
  - Where you compile code
- Compute/batch nodes
  - This is where jobs that are submitted through the scheduler run
  - Intended for heavy computation

Storage Spaces

- **Home Directories**
  - /home/$USER
  - Not for direct computation
  - Small quota (2 GB)
  - Backed up
- **$PROJECT Space**
  - /projects/$USER
  - Mid level quota (250 GB)
  - Large file storage
  - Backed up

- **Scratch Directory**
  - /scratch/summit/$USER
  - 10 TB
    - Can ask for more if needed
  - Files purged around 90 days

What Belongs Where?

- /home
  - Scripts
  - Code
  - Very small files
  - Inappropriate for sharing files with others
  - Inappropriate for job output
- /projects
  - Code/files/libraries relevant for any software you are installing (if you want to share files with others)
  - Mid-level size input files
  - Appropriate for sharing files with others
  - Inappropriate for job output
- /scratch/summit
  - Output from running jobs
  - Large files
  - Appropriate for sharing files with others
  - THIS IS NOT APPROPRIATE FOR LONG TERM STORAGE

Transferring Data

- Globus is Research Computing's preferred method of data transfer
- Designed with researchers in mind
- End points between computers make for efficient data transfer with an easy to use interface
  - Endpoints are different locations that data can be moved to/from
  - Personal or multi-user
- Rsync and sftp through the login nodes is good for small transfers

Setting Up Globus

- Create an account at Globus.org
- Make your personal computer an endpoint
- Transfer data
- www.globus.org

Software

- Common software is available to everyone on the systems
- Can install your own software
  - But you are responsible for support
  - We are happy to assist
- Research Computing uses modules to manage software
  - You can load modules to prepare your environment for using software
    - Set any environment variables
    - Set environment so application can find appropriate libraries, etc.

Important Things to Know About Modules

- Some modules might require a specific hierarchy to load
  - For some modules, you may need to specify a specific version
    - For example, **`module load R/3.3.0`**
  - For other modules, you may be able to be more generic
    - For example, **`module load matlab`**
- Some modules may require you to first load other modules that they depend on
- To find dependencies for a module, type `module spider <package>`
- To find out what software is available, you can type **`module avail`**
- To set up your environment to use a software package, type **`module load <package>/<version>`**

Job Submission

Running Jobs

- What is a “job”?
- Interactive jobs
  - Work interactively at the command line of a compute node
- Batch jobs
  - Submit job that will be executed when resources are available
  - Create a text file containing information about the job
  - Submit the job file to a queue

Job Scheduling

- On a supercomputer, jobs are scheduled rather than just run instantly at the command line
  - Shared system
  - Jobs are put in a queue until resources are available
- Need software that will distribute the jobs appropriately and manage the resources
  - Simple Linux Utility for Resource Management (Slurm)
    - Keeps track of what nodes are busy/available, and what jobs are queued or running
    - Tells the resource manager when to run which job on the available resources

Partitions and ‘Quality of Services’

- There are several ways to define where your job will run
- Partitions (basically a queue):
  - Resources/hardware
- QoS:
  - Tells what the limits or characteristics of a job should be
    - Maximum wall time
    - Number of nodes
- One partition might have multiple QoS
- A QoS might exist on multiple partitions

Available Partitions

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	Description	Maxwall	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 who have contributed to the Summit condo	7 D	n/a	n/a

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

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

`#SBATCH <options>` `sbatch <options>`

- Allocation: `--account=<account_no>`
- Partition: `--partition=<partition_name>`
- Sending emails: `--mail-type=<type>`
- Email address: `--mail-user=<user>`
- Number of nodes: `--nodes=<nodes>`
- Number of tasks: `--ntasks=<processes>`
- 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

Working on Summit

- Make sure you load the appropriate slurm module

```
module load slurm/summit
```

- After you run this command you can run sbatch to submit jobs

Blanca

- If you are a Blanca user, you need an RC account, but not an allocation
- To run jobs as a Blanca user, once you've logged into a login node, load the Blanca slurm module

```
module load slurm/blanca
```

- Only certain users have access to Blanca – paid service
- If you are unsure, you can ask your advisor or RC
  - But likely if you are unsure you don't have access

PetaLibrary

- To access the PetaLibrary, you login in to one our RC's login nodes as normal
- Then you cd to either /work/<groupname> or /archive/<groupname>, depending on your PetaLibrary service
  - <groupname> is the name set for your group when you set up the PetaLibrary service
  - You do not include the <>
- Only certain users have access to PetaLibrary – paid service
- If you are unsure, you can ask your advisor or RC
  - But likely if you are unsure you don't have access

Practice Examples

Submit Your First Job!

- Submit a slurm job with the following instructions:

1. The job should run the Unix “hostname” command
2. The job will be submitted from a bash script named hostname_summit.sh
3. The job will run on 1 node
4. We will request 1 minute wall time
5. Run from the debug QOS
6. Run on the shas partition

Hostname_summit.sh

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


# purge all existing modules
module purge


hostname
```

Running the script

- Load up the slurm module

```
module load slurm/summit
```

- Submit the job:

```
sbatch hostname_summit.sh
```

- Check output

Another slurm command

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

- EXAMPLE:

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

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

Your turn

- Submit a slurm job with the following instructions:

1. The job should run first the whoami command, then the Unix "sleep" command for 30 seconds, then the hostname command
   - Syntax for these Unix commands are below:

```
whoami
sleep 30
hostname
```

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

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

Your turn

- Submit a slurm job with the following instructions:

1. The job will be submitted from a bash script named sleep.sh
2. The job will run on 1 node
3. Request a 1 minute wall time
4. Run the job from the normal QOS
5. Run the job from the Summit haswell partition
6. Name your job `sleep`
7. Email yourself the results at the end of the job run
   - Hint: Requires two SBATCH options to do this – see link at top of this slide

Sleep.sh

```
#!/bin/bash
#SBATCH --nodes=1                                  # Number of requested nodes
#SBATCH --time=0:01:00                             # Max walltime
#SBATCH --qos=normal                               # Specify normal QOS
#SBATCH --partition=shas                           # Specify Summit GPU 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


# purge all existing modules
module purge


whoami
sleep 30
hostname
```

Running an external script

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

Running the script

- Submit the job:

```
sbatch matlab.sh
```

- Check output

Matlab.sh

```
#!/bin/bash
#SBATCH --nodes=1                          # Number of requested nodes
#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

# purge all existing modules
module purge


# Load Matlab module
module load matlab


# Run matlab without a GUI
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 `R_code.sh` that runs the R script
   - Syntax: `Rscript R_program.R`
4. The job will run on 1 node
5. We will request a 1 minute wall time
6. Specify the debug QOS
7. Specify the shas partition
8. The output will be put in a file called R_code_%j.out
9. Don’t forget to load the R module!

Solution – R_code.sh

```
#!/bin/bash
#SBATCH --nodes=1                    # Number of requested nodes
#SBATCH --time=0:01:00               # Max walltime
#SBATCH --qos=debug                  # Specify debug QOS
#SBATCH --partition=shas             # Specify Summit haswell nodes
#SBATCH --output=R_code_%j.out       # Output file name


# purge all existing modules
module purge


# Load the R module
module load R/3.3.0


# Run R Script
Rscript R_program.R
```

Solution – R_program.R

```
#Simple R code example by Shelley Knuth (shelley.knuth@colorado.edu)

# 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, Matlab
- Let’s run an interactive Matlab job

Interactive job

- To do this, we are going to log out and log back in
  - Only necessary for demo
  - Need to add something to the sign in process

- For Mac Users:

```
ssh -X username@login.rc.colorado.edu
```

- For Windows Users, must set up X-forwarding through your SSH client program

- Also must have an X-server package on your laptop
  - Xming for Windows or XQuartz for Mac

Interactive job

- To work with Matlab interactively, we're going to request some time from the supercomputer
- When the resources become available then we will start up Matlab
- Commands to run:

```
module load slurm/summit
sinteractive --qos=debug --time=00:05:00
```

Once we receive a prompt, then:

```
module load matlab
matlab
```

- Once we finish we must exit!

Questions?

- Email rc-help@colorado.edu
- Twitter: @CUBoulderRC

- Link to survey on this topic:

http://tinyurl.com/curc-survey16

- Slides: https://github.com/ResearchComputing/New_User_Seminar