Introduction to SLURM and Modules at CHPC

Albert Lund
CHPC User Services

Overview

- CHPC user environment (modules)
- Running interactive jobs
- Running batch jobs

Slides: home.chpc.utah.edu/~u0403692/SLURM-modules.pdf

vi Refresher/Exercise

- A few commands will get you started:
  - Press 'i' for insert! (Insert mode, Replace mode)
  - Press 'Esc' to get back to command mode!

  - :w - 'write'
  - :wq! - 'write and quit'
  - :q! - 'quit without saving (good for mistakes)
  - Press 'u' to undo in command mode
- Exercise: write something in vi and save it!
  - Try it with 'vim' too

Modules

- Modules are a way of managing the user environment
  - Most programs can be run through full path, e.g.

    /bin/ls
  - How does the OS know where 'ls' is? The PATH variable: "echo $PATH"
- Modules lets users modify the environment on the fly

Getting the dot files

- CHPC module page:
  https://www.chpc.utah.edu/documentation/software/modules.php
- New users do not need to do the following steps:
- To enable modules, the user will need to copy the CHPC bashrc and tcshrc to their home directory:

  cp /uufs/chpc.utah.edu/sys/modulefiles/templates/bashrc ~/.bashrc
  cp /uufs/chpc.utah.edu/sys/modulefiles/templates/tcshrc ~/.tcshrc

  – To set up the default CHPC environment, users will also need to copy the custom.sh and custom.csh scripts

  cp /uufs/chpc.utah.edu/sys/modulefiles/templates/custom.sh ~/.custom.sh
  cp /uufs/chpc.utah.edu/sys/modulefiles/templates/custom.csh ~/.custom.csh

- Make sure to save a copy of your old .bashrc/.tcshrc!

Basic module commands

- module - shows the list of module commands
- module avail - shows a list of "available" modules
- module list - shows a list of loaded modules
- module load <name> - loads a module
- module unload <name> - unloads a module
- module help <name> - prints help for a module
- module whatis <name> - prints info about the module
- module purge - unload all modules
- module swap <name1> <name2> - swap two modules

Module spider

- Some modules are dependent on other modules
  - these modules are not immediately available
  - example: Amber14 depends on the intel compiler and an MPI module in order to run (because of library dependencies).
- Use "module spider" to see a list of all modules, including modules that aren't loaded
- Use "module spider <name>" to see a subset of modules, or how to load a specific module

Exercise 1

- Explore the module load, unload, list, and avail commands to try loading and unloading modules
- Using "module spider", find the LAMMPS module and load it. (Remember that spider gives you instructions!)

Other module things

- Collections of modules:
  - Sometimes you'll have a set of modules you want to save so you don't have to reload them all the time
  - "module save" saves a default list of module
  - "module restore" restores the default list
  - "module save <name>" creates a named module list ("module restore <name>" reloads it)
  - "module savelist" shows a list of modules

Advanced modules

- Users can write and use their own modules
  - See our webpage on this

    https://www.chpc.utah.edu/documentation/software/modules-advanced.php
- Contact CHPC if you want/need help with this
- Modules can be loaded automatically in the .custom.sh/.custom.csh files provided.

Questions about modules?

SLURM

- CHPC SLURM page:

  https://www.chpc.utah.edu/documentation/software/slurm.php

- SLURM is a batch scheduling software
  - CHPC replaced PBS with SLURM in Spring 2015
- SLURM is almost the same, but completely different
  - Batch scheduling is handled identically (no change to policies)
  - Commands and scheduler directives are all different

Basic SLURM commands

- squeue - shows all jobs in the queue
  - squeue -u <username> - shows only your jobs
- sinfo - shows partition/node state
- sbatch <scriptname> - launches a batch script
- scancel <jobid> - cancels a job (find the jobid with squeue!)

Accounts and Partitions

- You need to specify an account and partition to run jobs
- You can see a list of partitions using the sinfo command
- Your account is assigned to you based on your group PI (e.g. if my PI is Baggins, I use the "baggins" account)
- Private node accounts are the same name as the partition for the private nodes:
  - e.g. baggins-kp, baggins-em, etc
- Private nodes can be used as a guest using the "owner-guest" account and the cluster-guest partition
- More info on the chpc slurm page:
  https://www.chpc.utah.edu/documentation/software/slurm.php#submit

Running interactive jobs

- An interactive command is launched through the srun command
  - Example:

```
srun --time=1:00:00 --ntasks 2 --nodes=1 --account=chpc
--partition=ember --pty /bin/tcsh -l
```

- Launching an interactive job automaticaqlly forwards environment information, including X11 forwarding (unless you tell SLURM not to inherit).
- "--pty" may be /bin/tcsh or /bin/bash

Exercise 2

- Start an interactive job on one node for ten minutes using the shell of your choice (use "echo $SHELL" to find out what your shell is)

```
srun --time=10:00 --ntasks 2 --nodes=1                --
account=chpc --partition=ember --pty /bin/tcsh -l
```

- In the session try running a few different commands:
  - srun hostname | sort | uniq (useful for making a machinefile)
  - echo $SLURM_JOBID
  - echo $SLURM_NNODES
  - echo $SLURM_NTASKS
  - echo $SLURM_SUBMIT_DIR

Template for the Basic SLURM Script

1. Set up the #SBATCH directives for the scheduler
2. Set up the working environment by loading appropriate modules
3. Add any additional libraries or programs to $PATH and $LD_LIBRARY_PATH
4. Set up temporary/scratch directories if needed
5. Switch to the working directory
6. Run the program with your input
7. Clean up any temporary files or directories

SLURM Batch directives

- #SBATCH -t 1:00:00 - time of a job
- #SBATCH --nodes=2 - number of nodes
- #SBATCH --ntasks 12 - total number of slurm tasks requested
- #SBATCH --ntasks-per-node=12 - tasks per node
- #SBATCH --mail-type=FAIL,BEGIN,END - send an email on events
- #SBATCH --mail-user=name@example.com - user email address
- #SBATCH --partition=name - partition to use
- #SBATCH --account=name - account to use
- #SBATCH -o filename - standard output file
- #SBATCH -e filename - standard input file
- #SBATCH -o filename.%j - "%j" when used in the filename will print the job number as part of the filename

The Basic PBS Script – batchscript.sh

```
!/bin/bash
#SBATCH --time=02:00
#SBATCH --nodes=1
#SBATCH -o slurmjob-%j
#SBATCH --ntasks=16
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-guest

#Set up whatever package we need to run with
module load somemodule

#set up the temporary directory
TMPDIR=/scratch/local/u0123456/data
mkdir -P $TMPDIR

#Set up the path to the working directory
WORKDIR=/uufs/chpc.utah.edu/common/home/u0123456/data
cd $WORKDIR

#Run the program with our input
BINDIR=/uufs/chpc.utah.edu/sys/pkg/mypackage/bin
$BINDIR/myprogram < $WORKDIR/input > $WORKDIR/output

rm -rf $TMPDIR
```

The Basic PBS Script – batchscript.csh

```
!/bin/tcsh
#SBATCH --time=02:00
#SBATCH --nodes=1
#SBATCH -o slurmjob-%j
#SBATCH --ntasks=16
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-guest

#Set up whatever package we need to run with
module load somemodule

#set up the temporary directory
set TMPDIR="/scratch/local/u0123456/data"
mkdir -P $TMPDIR

#Set up the path to the working directory
set WORKDIR="/uufs/chpc.utah.edu/common/home/u0123456/data"
cd $WORKDIR

#Run the program with our input
set BINDIR="/uufs/chpc.utah.edu/sys/pkg/mypackage/bin"
$BINDIR/myprogram < $WORKDIR/input > $WORKDIR/output

rm -rf $TMPDIR
```

srun vs mpirun (Parallel execution)

- The probable most significant change for parallel execution in moving from PBS to SLURM is that the PBS_NODEFILE is not generated by SLURM
- All of the MPI we use at CHPC are SLURM aware though, so mpirun will work without a machinefile unless you are manipulating the machinefile in your scripts
- Alternatively, you can use the srun command instead, which also hooks into most of our MPI libraries.
- Mileage may vary, and for different MPI distributions, srun or mpirun may be preffered (check our slurm page on the website for more info or email us)

Exercise 3

- Write a slurm batch script from scratch that does the following things and writes all output to a slurm output file:
  1. Reports the start time (hint: use the "echo" and "date" command")
  2. Purges all modules
  3. Loads two or three modules
  4. Lists the module loaded
  5. Executes "srun hostname"
  6. Executes "sleep 60"
  7. Reports the time at the end of the job (using "date")
- Once you have the script written, launch it on a single node using "sbatch" with ntasks equal to the number of cores on the node
- While the job is running use squeue to watch your job change states.
- Try running in different partitions to see the effects.

Solution to Exercise 1

```
#!/bin/bash
#SBATCH --time=02:00
#SBATCH --nodes=1
#SBATCH -o slurmjob-%j
#SBATCH --ntasks=16
#SBATCH --account=chpc
#SBATCH --partition=kingspeak

echo "Job started at "`date`

module purge
module load intel impi qe
module list

srun hostname

sleep 60

echo "Job ended at "`date`
```

- You should have an output file with a name similar to what you used for #SBATCH -o
  - e.g. slurmjob-123456
- If you were lucky you should have been able to see your job in the PD, R, and CG running states.

End of session!

Questions?

Email issues@chpc.utah.edu