What is a Supercomputer?

Shelley Knuth
shelley.knuth@colorado.edu

www.rc.colorado.edu Questions? #RC_BasicSC

Link to survey on this topic: http://tinyurl.com/rcpresurvey

Slides:
https://github.com/ResearchComputing/Final_Tutorials/tree/master/Basics_Supercomputing/2017_January

General Information

What Is a Supercomputer?

- A supercomputer is one large computer made up of many smaller computers and processors
- Each different computer is called a node
- Each node has processors/cores
  - Carry out the instructions of the computer
- With a supercomputer, all these different computers talk to each other through a communications network
  - Example – InfiniBand or Omni-Path

Computers and Cars - Analogy

Computers and Cars - Analogy

Why Use a Supercomputer?

- Supercomputers give you the opportunity to solve problems that are too complex for the desktop
  - Might take hours, days, weeks, months, years
  - If you use a supercomputer, might only take minutes, hours, days, or weeks

- Useful for problems that require large amounts of memory

World's Fastest Supercomputers

www.top500.org November 2016

Rank	Site	Name	TeraFlops
1	National Supercomputing Center (Wuxi, China)	Sunway	93,014.6
2	National Super Computer Center (Guangzhou, China)	Tianhe-2	33,862.7
3	DOE/SC/Oak Ridge National Laboratory (United States)	Titan	17,590.0
4	DOE/NNSA/LLNL (United States)	Sequoia	17,173.2
5	DOE/SC/LBNL/NERSC (United States)	Cori	14,014.7
6	Joint Center for Advanced High Performance Computing (Japan)	Oakforest-PACS	13,554.6
7	RIKEN Advanced Institute for Computational Science (Japan)	K	10,510.0
8	Swiss National Supercomputing Centre (Switzerland)	Piz Daint	9,779.0
9	DOE/SC/Argonne National Lab (United States)	Mira	8,586.6
10	DOE/NNSA/LANL/SNL (United States)	Trinity	8,100.9

What Does It Mean to Be Fast?

- Titan can do 17 thousand trillion calculations per second
- A regular PC can perform 17 billion per second
- Researchers can get access to some of these systems through XSEDE (The Extreme Science and Engineering Discovery Environment)

Supercomputer Details

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

Additional Compute Resources

- 10 Graphics Processing Unit (GPU) Nodes
  - Visualization of data
  - NVIDIA Tesla K80 (2/node)
- 5 High Memory Nodes
  - 2 TB of memory/node, 48 cores/node
- Phi Nodes (planned summer 2017)
  - 20 nodes
  - Intel Xeon Phi

Different Node Types

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

Storage Spaces

- System variations
- **Home Directories**
  - Store source code
  - Not for direct computation
  - Small quota (2 GB)
  - Backed up
- **$PROJECT Space**
  - Mid level quota (250 GB)
  - Large file storage
  - Backed up

- **Scratch Directory**
  - Much larger – depends on system
  - Output from running jobs should go here
  - Files purged around 90 days

Jobs

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

What is Job Scheduling

- Supercomputers usually consist of many nodes
- Users submit jobs that may run on one or multiple nodes
- Sometimes these jobs are very large; sometimes there are many small jobs
- Need software that will distribute the jobs appropriately
  - Make sure the job requirements are met
    - Reserve nodes until enough are available to run a job
    - Account for offline nodes
- Also need software to manage the resources
- Integrated with scheduler

Job Scheduling

- On a supercomputer, jobs are scheduled rather than just run instantly at the command line
  - Shared system
  - People are allocated a certain amount of time on the system
  - Based on your need
    - Proposal
    - Request the amount of resources needed and for how long
  - Jobs are put in a queue until resources are available

Job Schedulers - Slurm

- Jobs on supercomputers are managed and run by different software
- Simple Linux Utility for Resource Management (Slurm)
  - Open source software package
- Slurm is a resource manager
  - Keeps track of what nodes are busy/available, and what jobs are queued or running
- Slurm is a scheduler
  - Tells the resource manager when to run which job on the available resources
- No longer need to load the Slurm module!

Useful Slurm Commands - sbatch

- **sbatch**: submit a batch script to slurm
  - Standard input (keyboard)
  - File name
    - Options preceded with #SBATCH
- sbatch exits immediately after receiving a slurm job ID
- By default, standard output and errors go to file named slurm-%j.out (job allocation number)
- Slurm runs a single copy of the script on the first node in the set of allocated nodes

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

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

SBATCH Options

In batch script put:
`#SBATCH <options>`

OR on command line
`sbatch <options>`

- Allocation: `-A=<account_no>`
- Partition: `-p=<partition_names>`
- Sending emails: `--mail-type=<type>`
- Email address: `--mail-user=<user>`
- Number of nodes: `-N <nodes>` or `--nodes=<nodes>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall time>`
- Job Name: `-J <jobname>` or `-job-name=<jobname>`

Allocations

- Moving to a fair-share allocation
- Users should no longer think of allocations as a bank account
  - No longer get a certain number of hours
- Now get a share of a computer at any given time
  - Averaged over a 3-4 week period

- Motivation for fair share:
  - More universities involved, so rather than just allocate a percentage gives everyone a target
  - Also prevents the system from being idle
  - No more fuss of worrying if ran out of allocation time

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 and by QOS plus how long job has been in the queue
  - Each is weighted differently

- System will first look at weighted average utilization of user over last 3-4 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 your target percentage (based on a 3-4 week average) your priority is increased
- If you are over your target percentage (based on a 3-4 week average) your priority is decreased

Fair Share Allocation (Cont)

- Once the system determines the priority based on historical use of the user as well as target percentage then it looks at QOS utilization
- Job priority is adjusted accordingly

- 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

Fair Share Example - Under Utilization

- User has not run a job in the last 4 weeks and is trying to run in the `normal` QOS
- System discovers that the user has a 0% utilization
  - They are therefore under their target percentage, and priority gets bumped up
- The system then looks at the `normal` QOS and discovers that it is underutilized
  - Priority gets bumped up

Fair Share Example - Over Utilization

- User has been running over their target percentage for the last four weeks
  - Maybe it's Christmas and the system is idle
- Requests to run on the `normal` QOS
- System discovers that the user is over their target percentage, so their priority gets bumped down
- The system then looks at the `normal` QOS and discovers that it is over utilized
  - Priority gets bumped further down

Fair Share Example - Over Utilization (Continued)

- Priority is constantly getting re-evaluated based on new information
  - Jobs will get a higher priority the longer they wait
  - Can continue to put in jobs but priority is lower
  - This will also have you be penalized for running jobs in spurts

Wall Times

- The maximum amount of time your job will be allowed to run
- How do I know how much time that will be?
- What happens if I select too much time?
- What happens if I select too little time?

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

Partitions

- By default, jobs will run on the general compute (Haswell) nodes
- Recommend specifying a partition
- Do this using the –p partition flag

Available Partitions

Partition	Description	# of nodes	cores/node	GPUs/node	Maxwall
shas	General Compute (Haswell)	380	24	0	24 H
sgpu	GPU-enabled nodes	10	24	effectively 4	24 H
smem	High-memory nodes	5	48	0	7 D
sknl	Phi (Knights Landing) nodes - [not currently available]	20	64	0	24 H

Quality of Service = Queues

QoS	Description	Maxwall	Max jobs/user	Max nodes/user	Initial priority
normal	Default QoS	Derived from partition	n/a	300	0
debug	For quick turnaround when testing	1 H	1	32	Equiv of 3-day queue wait time
long	For jobs needing longer wall times	7 D	n/a	20	
condo	For groups who have purchased Summit nodes	7 D	n/a	n/a	Equiv of 1-day wait time

Job Scheduling - Priority

- What jobs receive priority?
  - Fair share allocation
- Certain QOSs might have priority over others
  - Might have different queues based on different job needs
- If you reach job limits associated with QOS will cause priority changes
- Can set aside nodes for a job by creating a reservation

Software

- Common software is available to everyone on the systems
- Can install your own software
- To find out what software is available, you can type **ml avail**
- To set up your environment to use a software package, type **ml <package>/<version>**

Software

- Things to be aware of
  - Some software packages require you to first load a compiler module
    - Intel, gcc
    - Will fail to load the module
  - Some packages require you to be specify a version – for example R
    - `ml R/3.2.0`
    - Use `module spider` to find more information

Initial Steps to Use RC Systems

- Apply for an RC account
  - https://portals.rc.colorado.edu/account/request
- Get registered with Duo
  - Duo invitation
  - Smart phone app
  - Push notifications
- Apply for a computing allocation
  - Startup allocation
  - Additional SU require a proposal

Janus/Summit Access

- To access RC's computing, in general:

```
ssh login.rc.colorado.edu –l <username>
Password:  duo:<identikey>
```

- Need the Duo application

What's Next?

- So far we've introduced you to the basics of supercomputing
- Next, learn to:
  - Use the command line
  - Submit jobs!
  - Learn Linux!
  - Load up some software!

Questions?

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

- Link to survey on this topic:

http://tinyurl.com/curc-survey16

- Slides: https://github.com/ResearchComputing/Final_Tutorials/tree/master/Basics_Supercomputing/2017_January