UNIX/Linux for remote access, TACC systems

SDS335/398 Scientific/Technical Computing

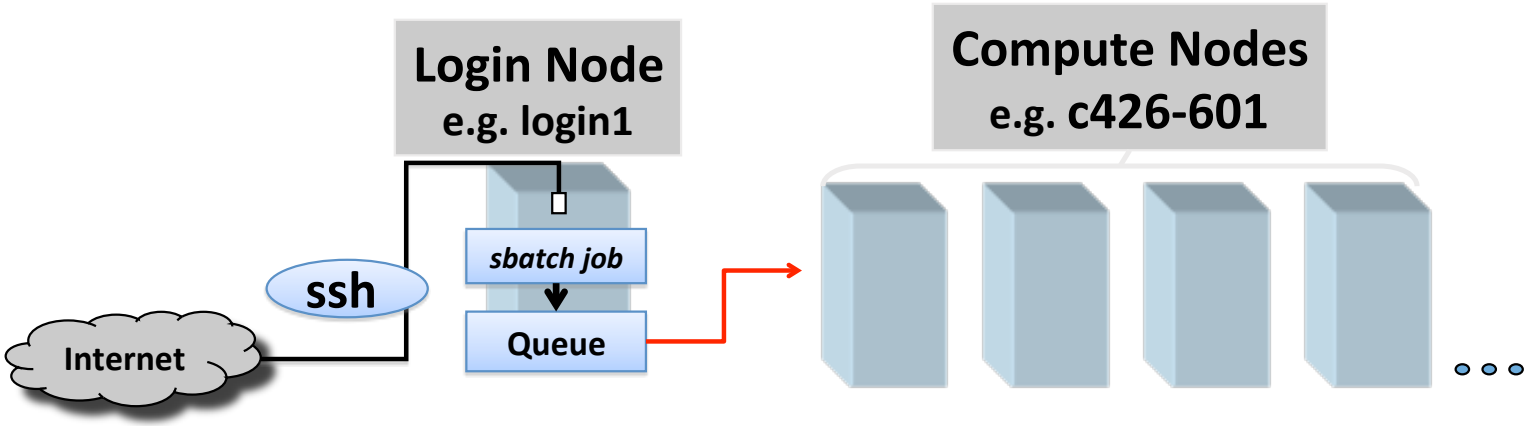
September 5, 2019

Lars Koesterke
Slides by Anne Bowen

stampede2

- Note there is a user guide: https://portal.tacc.utexas.edu/user-guides/stampede2
- and a ticket system: https://portal.tacc.utexas.edu/tacc-consulting

Login Nodes and Compute Nodes



- "Front end" or "head node" "Back end" Compute nodes

Do not run parallel programs on the login nodes!

Good Citizenship

- The Stampede cluster is a shared resource. Hundreds of users may be logged on at one time accessing the filesystem, hundreds of jobs may be running on all compute nodes, with a hundred more jobs queued up. All users must practice good citizenship and limit activities that may impact the system for other users. Stampede's four login nodes as well as the three Lustre file systems ($HOME, $WORK, and $SCRATCH) are shared among all users. Good citizenship can be boiled down to two items:

- Do not run programs on the login nodes

- Do not abuse the shared filesystem:

    - Avoid running jobs in the $HOME directory. Run jobs in $WORK or $SCRATCH.

    - Avoid too many simultaneous file transfers. Three concurrent scp or globus-url-copy sessions (see Transferring Files) is probably fine. One hundred concurrent file sessions is not.

    - Limit I/O intensive sessions (lots of reads and writes to disk), particularly opening/closing a lot of files

stampede2 filesystem

- Most likely you'll be only using the HOME file system

Environmental Variable	User Size Limits	Characteristics
$HOME	10.0 GB	Not intended for parallel of high-intensity file operations Regular back ups
$WORK	1.0 TB	Not intended for parallel of high-intensity file operations Not purged, Not backed up
$SCRATCH	(30PB total)	Subject to purge after 10 days (admin discretion)

Hands-on: ssh to stampede2

- `ssh username@stampede2.tacc.utexas.edu`
- Note the project balances, disk quotas and tips.
- Note your current directory `(pwd)`
- `cd $WORK`
  `cd $SCRATCH`
- `‘cdw’ and ‘cds’`

TACC's module system

- Developed from an earlier open source project “software modules”
- Load a module to make certain software available (including setting paths and environment variables)
  - Note: some software available both from the Linux installation and as module (e.g. cmake, python). The module is likely more up to date

Hands-on: module commands

- Give these a whirl

$ module list	lists currently loaded modules
$ module spider python	lists all modules with text "python"
$ module help	lists options
$ module avail	lists available modules
$ module load <module>	add a module
$ module help <module>	module-specific help
$ module unload <module>	remove a module
$ module swap <mod1> <mod2>	swap two modules
$ module spider	lists all modules
$ module reset	restore "factory settings"

Your ticket to Compute Nodes:

- Three ways to get to the back end (compute nodes):
  SLURM batch job: sbatch <batchfilename>
  ‘idev’
  Run special application that connects to back end: e.g. ddt
  ssh to node on which you already have a job running

- If you don’t use sbatch, idev (or something equivalent) you’re executing on the front end (login nodes) – don't do this!

- Don’t launch exe (e.g. ./a.out) on the command line (aka the login node)
  One of the easiest ways to get your account suspended

-

Hands-on: idev

- Can execute `idev` on the login node to start an interactive job
- Can customise your job:
  - idev -p development -N 1 -m 30 # development queue, 1 node, 30 minutes
- We do **not** have a reservation of nodes for this class!
- Your bash prompt will change from `login1$` to something like `c448-004$`. You are now on a compute node. You can now run your executable or run a script.
- Let's do some python scripting on the compute node.
- You can use srun too (but you'll see later why idev is better)

Hands-on: idev

- Let's spin up a python interpreter and calculate pi:

```
damon@c401-092$ ipython
In [1]: import numpy as np

In [2]: points = np.random.random((10000, 2))

In [3]: points2 = np.square(points)

In [4]: norm2 = np.sum(points2, axis=1)

In [5]: num_inside = np.sum(norm2 <= 1.0)

In [6]: 4.0 * num_inside / 10000.0
Out[6]: 3.1256
```

Hands-on: slurm and bash

- Let’s do some bash scripting.
- Bash scripting is easy: you just put bash commands in a file and add `#!/bin/bash` at the top of the file
- You’ll probably need to make your script executable (recall chmod)
- Bash scripts are better for recording a set of instructions to execute
- slurm takes a bash script as input to run on the compute node

Hands-on: slurm

- To launch a batch job
- sbatch <batchfilename>
- To launch a one-node, sixteen core interactive session in the development queue
- $ idev –n 16 –t 00:30:00 –p development –A STC19
- To view all jobs in the queues: squeue | more or showq | more
- To view status of your own jobs: squeue –u <userid> or showq -u <userid>
- To delete a job: scancel <jobid>

Exercises with Unix commands (1)

- All users that are
  - Logged on
  - Have the letter ‘a’ in their user name
- Use these commands: ‘who’ and ‘grep’

Exercises with Unix commands (2)

- **First** user that is
  - Logged on
  - Have the letter ‘a’ in their user name
- Use these commands: ‘who’, ’grep’, and ‘head'
- What does ‘head’ do?
  - Why could the option ‘-n <number>’

Exercises with Unix commands (3)

- **Second** user that is
  - Logged on
  - Have the letter ‘a’ in their user name
- Use these commands: ‘who’, ’grep’, ‘head’, and ‘tail’
- What does ‘tail’ do?

Exercises with Unix commands (4)

- **Second** user that is
  - Logged on
  - Have the letter ‘p’ in their user name
- Use these commands: ‘who’, ’grep’, ‘head’, ‘tail’, and ‘cut’
- What does ‘cut’ do?
  - Maybe ‘**cut -d” “ -f 1**’ would help in the pipeline

Exercises with Unix commands (4)

- **First** users that are
  - Logged on
  - Have the letter ‘a’ in their user name
- Use these commands: ‘who’, ’grep’, ‘head’, and ‘tail’
- What does ‘tail’ do?

vis portal and vnc access (will cover this later)