Stampede
User Environment

Victor Eijkhout
September 2013
Overview

Effective users, good citizens

• Getting Started – Access to Stampede
• Getting Acquainted – A Tour of Stampede
• Getting Work Done – Using Stampede
• Getting Along – Good Citizenship

-------------------------------

• Lab 1 – Test Drive (including tour of the MIC)
• Supplemental Material – end of slide deck
Disclaimers

• Audience: users new to supercomputing
• Tone: breadth rather than depth
• Moving target: much is pending or evolving
Where to go for information
(after you’ve been through this training)

• Help with Unix in general
• Help with specific programs
• Help with the environment at TACC in general (batch system, tape robot, Xeon PHI, ....)
• Advice with computing in general
Unix help

- man ("manual") pages and help systems
  - Try "man" and "man –k" before command name
  - Space bar to advance within man page, "q" key to exit
  - Try command name with -h, --help, -help, help
  - Try command name with no argument
Help with specific programs

• For commands controlled by the “module” system (see later): “module help abaqus”

• Maybe someone at TACC knows more: submit a ticket

• See if there are webpages, user forums, etc.
Help with the environment at TACC

• Read the userguides!
• User Guide(s), Usage Policies, etc. and associated links
  http://www.tacc.utexas.edu/user-services
  – Note "Updates and Notices" at top of Stampede User Guide
• Or submit a ticket in the user portal
Help with computing in general

• Submit a ticket asking for advice.
• We love doing this stuff, so we’re happy to talk to you and think about your application
• Possibility of extended support through XSEDE
Getting Started

Access to Stampede and Other TACC Resources
Before you log in, you'll need...

• **Portal Account**
  – XSEDE users go to [https://portal.xsede.org/](https://portal.xsede.org/)
  – UT System users go to [www.portal.tacc.utexas.edu](http://www.portal.tacc.utexas.edu)

• **Allocation (computing hours)**
  – PI must request allocation through appropriate portal
  – PI may use portal to assign active users to an allocation
  – Allocation associated with "project name" (account code)

• **Activation on TACC resources**
  – Involves email handshake(s) with TACC user services
  – May take a few business days
  – Note that your TACC credentials (think ssh) may differ from XSEDE
  – TACC password resets can take 30+ minutes to propagate
Initial login with explicit ssh

• Start with a Linux-like terminal or equivalent* connected to internet
  – Linux command line
  – Mac terminal app
  – PuTTY, Secure Shell Client, GSI-SSH on XSEDE portal,…

• Connect to a login node with ssh or equivalent

  % ssh stampede.tacc.utexas.edu
  % ssh username@stampede.tacc.utexas.edu
  % ssh -X stampede.tacc.utexas.edu
  % ssh -Y stampede.tacc.utexas.edu

*many users will access Stampede through a special gateway designed and maintained for their research community; see e.g. xsede.org/gateways-overview
Shells and Startup Scripts

• OS is Linux

• bash is default shell, but TACC supports most major shells
  – bash, csh, tcsh, zsh, ...

• Submit ticket to change default shell (chsh will not work)

• System-level startup files execute before account-level

• It’s worth your trouble to understand startup files
  – e.g. .profile and .bashrc
  – Easy way to customize environment (e.g. prompt, aliases)
  – Caution: environment associated with shell (~ “window”), not acct
  – Caution: avoid using “echo” in startup scripts (will break scp et al!)
Text Editors

• Pick a favorite; become proficient
  – nano – simple
  – vi (vim) – terse
  – emacs – powerful

• Appreciate cross-platform issues
  – Win to Linux – dos2unix utility
  – Linux to Win – Wordpad rather than Notepad
  – Linux filenames are case sensitive
  – Blanks in filenames require some care

• Do NOT use MS Word (or so) to create job scripts and such. Also do not copy/paste from PDF files.
Getting Acquainted

A Tour of Stampede
Typical Stampede Node ( = blade )

Dell PowerEdge 8220
("DCS Zeus")
Compute Node
Typical Stampede Node (= blade)

CPU (Host) 
"Sandy Bridge"

16 cores 
32G RAM
Two Xeon E5 8-core processors

Coprocessor (MIC) 
"Knights Corner"

x16 PCIe

61 lightweight cores 
8G RAM
Xeon Phi Coprocessor
Each core has 4 hardware threads
MIC runs lightweight Linux-like OS (BusyBox)
Stampede Basic Specs

• ~6400 nodes in 160 racks

• Typical node
  – 16 cores on divided over 2 CPU sockets, 32G RAM
  – 61 cores on MIC coprocessor, 8G RAM

• Specialized nodes
  – 16 large-memory nodes (32 Xeon cores, 1T RAM) with Fermi-class GPUs for visualization (no CUDA, no MIC)
  – 128 GPU nodes, each with NVIDIA Kepler2 and a MIC
  – Login nodes don’t have MIC coprocessors
Nodes Have Personalities and Purposes

Internet

ssh

Login Node e.g. login1

sbatch job

Queue

Compute Nodes e.g. c426-601

“Front end” or “head node”

“Back end”
What does this mean?

• Do not run parallel programs on the login nodes
  – Either it’s simply not possible
  – Or you’ll annoy the sysadmin (and he may lock your account)

• Instead: submit a batch job. We’ll get to that.
• From any cpu host: the aliases `cdh`, `cdw` and `cds` change your working directory to your `$HOME`, `$WORK` and `$SCRATCH` directories respectively.

• From MIC coprocessor: file systems are visible, but cd aliases (e.g. `cdw`) and env variables (e.g. `$WORK`) are not yet avail (cd to full explicit path).

---

**File System Specs**

<table>
<thead>
<tr>
<th>Environmental Variable</th>
<th>User Size Limits</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$HOME</code></td>
<td>5.0 GB</td>
<td>Regular backups</td>
</tr>
<tr>
<td><code>$WORK</code></td>
<td>400 GB</td>
<td>Not purged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No backup</td>
</tr>
<tr>
<td><code>$SCRATCH</code></td>
<td>(~8.5PB total)</td>
<td>Subject to purge after 10 days</td>
</tr>
<tr>
<td><code>$ARCHIVER:$ARCHIVE</code></td>
<td>Essentially unlimited</td>
<td>Files staged to and from tape</td>
</tr>
<tr>
<td><code>(Ranch home directory)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>/tmp (local to node)</code></td>
<td>~80 GB per node</td>
<td>Purged after job</td>
</tr>
</tbody>
</table>
File Transfers

- We recommend starting with scp or rsync; other protocols possible.
- Avoid using recursive (-r) flag with large transfers; bundle files with tar utility.
- Avoid simultaneous transfers and tar jobs on login nodes.
- Compression and optimization are rarely necessary.
- On Ranch, staging from tape takes time.
- Beware of cross-platform issues: filenames (spaces, capitalization).

http://www.tacc.utexas.edu/user-services/user-guides/ranch-user-guide
Getting Work Done

Using Stampede
Lmod: TACC’s Module System

• Developed from an earlier open source project “software modules”
• Load a module to make certain software available
• Tricky: some software available both from the Linux installation and as module (svn, python). The module is likely more up to date
• Load modules
  – Interactively
  – In job scripts
  – In startup scripts
Why modules?

- Some software needs permission to use (Vasp, Abaqus)
- Dependency management:
  - Multiple compilers
  - Multiple MPI stacks (each dependent on compilers)
  - Using the wrong combination can give “interesting” errors.
- Modules can affect MIC operations, but Lmod not currently available on MICs themselves
Key Module Commands

%- module help {lists options}
%- module load <module> {add a module}
%- module avail {lists available modules}
%- module unload <module> {remove a module}
%- module swap <mod1> <mod2> {swap two modules}
%- module help <module> {module-specific help}
%- module spider {lists all modules}
%- module spider petsc {list all versions of petsc}
%- ml {abbrev for module list}
%- ml <module> {abbrev for module load}
%- module reset {return to system defaults}
(Personal) Default Modules

• Save/restore personal default module environment:
  $ module reset  # return to sys default
  $ module load ddt
  $ module load fftw3
  $ module save   # now loaded at login or restore

• Save/restore named collections of modules:
  $ module save chemtools
  ...
  $ module restore chemtools
  – Execute “module help” for more info

• This is a great way to achieve reliability and repeatability
Compilers

• Intel 13 is the compiler of choice for Stampede
  – The only compiler that supports Xeon Phi coprocessor
  – Currently three versions of gcc suite are also available
  – We also support other specialized compilers
    • E.g. cuda support (nvcc): module load cuda

• Compilers available from login nodes and compute node hosts
  – Compilers not visible from MIC coprocessors...
  – ...but you can compile for the MIC from a Sandy Bridge host

• Numerous math libraries available, but MKL's MIC support
  makes it especially important
  (www.intel.com/software/products/mkl)
MPI Compilation

<table>
<thead>
<tr>
<th>Command</th>
<th>Language</th>
<th>Type Suffix</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpicc</td>
<td>c</td>
<td>.c</td>
<td>mpicc &lt;options&gt; prog.c</td>
</tr>
<tr>
<td>mpicxx</td>
<td>C++</td>
<td>.C, .cc, .cpp, .cxx</td>
<td>mpicxx &lt;options&gt; prog.cpp</td>
</tr>
<tr>
<td>mpif77</td>
<td>F77</td>
<td>.f, .for, .ftn</td>
<td>mpif77 &lt;options&gt; prog.f</td>
</tr>
<tr>
<td>mpif90</td>
<td>F90</td>
<td>.f90, .fpp</td>
<td>mpif90 &lt;options&gt; prog.f90</td>
</tr>
</tbody>
</table>

- mvapich2 and impi (Intel) currently supported.
- The mpiXXX commands are shell scripts.
- They call the underlying C/C++/Fortran compiler.
Your Ticket to Compute Nodes

• Four ways to get to the back end (compute nodes):
  – SLURM batch job: `sbatch <batchfilename>`
  – SLURM interactive session: `srun <flags>`
  – Run special app that connects to back end: e.g. `ddt`
  – ssh to node on which you already have a job running
    -- once on compute node, `ssh mic0` gets you to its mic

• If you don’t use sbatch, srun, or equivalent, you’re running on the front end (login nodes) – don't do this!
  – Don’t launch exe ( e.g. `./a.out` ) on the command line
  – One of the easiest ways to get your account suspended
Key SLURM and Related Commands

• To launch a batch job
  ```
  sbatch <batchfilename>
  ```

• To launch a one-node, sixteen core interactive session in the development queue
  $$
  \texttt{srn \ --pty \ -n \ 16 \ -t \ 00:30:00 \ -p \ development \ -A \ 20130418HPC \ /bin/bash \ -l}
  \# \ last \ char \ is \ lower \ case \ "el" \ (launches \ bash \ as \ login \ shell)
  \# \ -A \ flag \ is \ optional \ unless \ you \ have \ multiple \ projects
  $$

• 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>`
• To view status of queues: `sinfo -o "\%20P \ %5a"
  – Ignore queue limits reported by this command; they are not the ones in force.
### General Use Stampede Queues*

<table>
<thead>
<tr>
<th>Queue</th>
<th>Max Runtime</th>
<th>Max Nodes (Cores)</th>
<th>Charge Rate</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>48 hrs</td>
<td>250 (4000)</td>
<td>1</td>
<td>normal production</td>
</tr>
<tr>
<td>development</td>
<td>4 hrs</td>
<td>16 (256)</td>
<td>1</td>
<td>development nodes</td>
</tr>
<tr>
<td>largemem</td>
<td>48 hrs</td>
<td>4 (128)</td>
<td>2</td>
<td>large memory nodes</td>
</tr>
<tr>
<td>serial</td>
<td>12 hrs</td>
<td>1 (16)</td>
<td>1</td>
<td>serial or shared memory</td>
</tr>
<tr>
<td>large</td>
<td>24 hrs</td>
<td>1000 (16000)</td>
<td>1</td>
<td>large core counts</td>
</tr>
<tr>
<td>normal-mic</td>
<td>24 hrs</td>
<td>250 (4000)</td>
<td>1</td>
<td>early production mic nodes</td>
</tr>
<tr>
<td>gpu</td>
<td>24 hrs</td>
<td>32 (512)</td>
<td>1</td>
<td>GPU nodes</td>
</tr>
<tr>
<td>gpudev</td>
<td>4 hrs</td>
<td>4 (64)</td>
<td>1</td>
<td>GPU development nodes</td>
</tr>
<tr>
<td>vis</td>
<td>8 hrs</td>
<td>32 (512)</td>
<td>1</td>
<td>GPU nodes + VNC service</td>
</tr>
</tbody>
</table>

*Queue properties subject to change
#!/bin/bash

# Don't miss this line!

# Generic SLURM script -- MPI
#
#SBATCH -J myjob            # Job name
#SBATCH -o myjob.%j.out    # stdout; %j expands to jobid
#SBATCH -e myjob.%j.err    # stderr; skip to combine stdout and stderr
#SBATCH -p development     # queue
#SBATCH -N 2               # Number of nodes, not cores (16 cores/node)
#SBATCH -n 32              # Total number of MPI tasks (if omitted, n=N)
#SBATCH -t 00:30:00        # max time

#SBATCH --mail-user=djames@tacc.utexas.edu
#SBATCH --mail-type=ALL

#SBATCH --A TG-01234       # necessary if you have multiple project accounts

module load fftw3
module list

ibrun ./main.exe          # Use ibrun for MPI codes. Don’t use mpirun or srun.

Additional Software

• Stack is always under construction
• Computation: e.g. R, Octave, PETSc, ...
• Python module gives you NumPy, SciPy, MatPlotLib, ...
• Analysis and Debugging: e.g. tau, papi, perfexpert, ddt, ...
• Parameter Studies: pylauncher and launcher
• High performance file i/o: hdf5, parallel hdf5, netcdf
• Build and install your own tools
  – We strongly recommend installing in $WORK
  – Download tar archive, not pre-packaged installer
  – Standard trick: ./configure --prefix=$WORK/myapps
• Submit a ticket asking TACC to install something
Getting Along

Good Citizenship
The Keys to Good Citizenship

Remember you are sharing resources
(login nodes, file systems, bandwidth)

Use components for intended purposes
Login nodes: appropriate use

- Building software
  - But Stampede compilers are also visible on compute nodes
- Managing files
  - Editing, transfers, tar/untar
- Submitting, monitoring, managing batch jobs
  - sbatch, showq, squeue, squeue -u username, scancel...
- Launching interactive sessions
  - srun, ddt, etc.
- Modest postprocessing (gnuplot and such)
Login nodes: inappropriate use

• Don’t do science on the front end
  – Access compute nodes with sbatch, srun, or equiv
  – Don’t launch exe directly

• Avoid simultaneous instances of demanding processes
  – Parallel builds (e.g. `make -j`), tar/untar, transfers
File System Citizenship

• Avoid running jobs from $HOME
• Run demanding jobs from $SCRATCH
• Avoid frequent i/o when possible
• Minimize simultaneous i/o from many processes
• Learn to recognize/avoid other stressors
  – e.g. under-the-table stat (du, default ls) on big dirs
• Know when it’s time to learn/use parallel i/o
Lab 1

Test Drive
Overview of Lab

- Part 0 – Grab the Lab Files
- Part 1 – Run an MPI Batch Job (sbatch)
- Part 2 – An Interactive Session (srun)
- Part 3 – Run MIC App from the Host
- Part 4 – Visit the MIC

- Secure Shell Client terminal program available on TACC laptops
- Slides contain supplemental info on editors
Supplemental Material
nano

- All operations/commands are preceded by the Control key:
  - ^G Get Help
  - ^O WriteOut
  - ^X Exit
  - ....

- If you have modified the file and try to exit (^X) without writing those changes (^O) you will be warned.

- Makes text editing simple, but it has less powerful options than vi and emacs (search with regular expressions, etc..)
vi/vim & emacs

• vi/vim command cheat sheet
  – http://www.tuxfiles.org/linuxhelp/vimcheat.html

• emacs command cheat sheet
  – http://emacswiki.org/emacs/ReferenceCards