Programming Environment

Cornell Center for Advanced Computing
January 23, 2017

Thanks to Dan Stanzione, Bill Barth, Lars Koesterke, Kent Milfeld, Doug James, and Robert McLay for their materials developed at TACC and XSEDE that were incorporated into this talk.
1. Accessing Stampede
2. Login Environment
3. Stampede Overview
4. Software
5. Compiling
6. Timing
7. Editing Files
8. Batch Job Submission: SLURM
9. Play Nice
10. Help
1. Accessing Stampede
Before you can access Stampede

2. Get an Allocation (computing hours)
   - PI must request allocation through appropriate portal
   - PI may use portal to assign active users to an allocation
   - Note your allocation’s “project name” (account code)
3. Activate your account on TACC resources (now automatic)
4. Decide how you’re going to log on:
   - Use XSEDE Single-Sign-On
     • Set up Multi-Factor Authentication (MFA) at XSEDE
   - SSH directly to TACC
     • Log into TACC portal, and reset your password
     • Set up Multi-Factor Authentication (MFA) at TACC
2. Get an Allocation -- View My Allocations on XUP

- Go to the XSEDE User Portal (XUP): portal.xsede.org
- Log in
- Go to ‘My XSEDE’ tab
- Choose ‘Accounts’ on the nav bar
- See which resources you have access to
- Note: you may not have the same username on all resources
4. Decide How You’re Going to Log On

SSH to the XUP Single Sign On (SSO) Login Hub

• Pros:
  – No need for a resource-specific username and password.
  – Easy way to switch between XSEDE resources

OR: SSH directly to TACC resources

• Pros:
  – Fewer clicks

**MFA is required to access TACC resources either way**
Either way you need SSH:

• SSH Secure Shell
  – SSH on the Command Line (Unix/Linux) or Terminal window (Mac)
  – SSH / telnet client for Windows – e.g. PuTTY
• Do not overwrite ~/.ssh/authorized_keys
• Do add stampede to the list of known hosts, if prompted

On linux: Open a command line window, then $ ssh …
On Mac: Open a terminal window, then $ ssh …
On Windows: Install and use a client like PuTTY (gui)
Single Sign On (SSO) Login Hub

Set up MFA  [https://portal.xsede.org/mfa/](https://portal.xsede.org/mfa/)

- Install the Duo app on your smartphone or other device
  Find [Duo](https://itunes.apple.com/app/duo-multifactor-authentication/id497030544) on iTunes or Google Play
  Duo at Cornell: [https://it.cornell.edu/twostep](https://it.cornell.edu/twostep)

- **Enroll your XSEDE Portal account in Duo**

- **Pair your Duo-enabled device with your XUP account**
Single Sign On (SSO) Login Hub

- `ssh` to `login.xsede.org` - single point-of-entry to XSEDE resources
- A 12 hour `proxy certificate` is automatically generated
- `gsissh` to any XSEDE compute resource
- Log in using your XUP username and password

Wait 2 slides if you’ve set up direct access

On Linux or Mac:
```
localhost$ ssh XUPusername@login.xsede.org
[shm7@sschub ~]$ gsissh stampede-knl
```

On Windows:
Start PuTTY, then enter Host Name `login.xsede.org`
```
[shm7@sschub ~]$ gsissh stampede-knl
```

[https://portal.xsede.org/web/xup/single-sign-on-hub](https://portal.xsede.org/web/xup/single-sign-on-hub)
Connect Directly to TACC resources

Log into TACC portal, and reset your password (first time)

Set up MFA
https://portal.tacc.utexas.edu/tutorials/multifactor-authentication

• Manage Profile on the TACC User Portal

• Select Pairing Method
  – TACC Token app – download & pair TACC Token app
  – SMS (text) messaging - pair
  – TACC Hard token – request physical token (fob)
Connect Directly to TACC resources

- `ssh` to `login-knl1.stampede.tacc.utexas.edu`
- Log in using your TACC username and password

On linux or Mac:

`localhost$ ssh TACCUsername@login-knl1.stampede.tacc.utexas.edu`

On Windows:

Start PuTTY, then enter Host Name

`login-knl1.stampede.tacc.utexas.edu`

*(Stay logged in)*

[https://portal.tacc.utexas.edu/user-guides/stampede#knl-system-access](https://portal.tacc.utexas.edu/user-guides/stampede#knl-system-access)
2. Login Environment
Account Info

Note your account number in the splash screen.

<table>
<thead>
<tr>
<th>Name</th>
<th>Avail SUs</th>
<th>Expires</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG-TRA140011</td>
<td>21016</td>
<td>2017-01-26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk</th>
<th>Usage (GB)</th>
<th>Limit</th>
<th>%Used</th>
<th>File Usage</th>
<th>Limit</th>
<th>%Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home1</td>
<td>0.0</td>
<td>5.0</td>
<td>0.07</td>
<td>231</td>
<td>150000</td>
<td>0.08</td>
</tr>
<tr>
<td>/work</td>
<td>0.0</td>
<td>1024.0</td>
<td>0.00</td>
<td>13</td>
<td>3000000</td>
<td>0.00</td>
</tr>
<tr>
<td>/scratch</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>2</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Get the Lab Files

- TAR = Tape ARchive. Just concatenates files.
- `tar <switches> <files>`
  - `z` = compress or decompress
  - `x` = extract
  - `c` = create
  - `v` = verbose
  - `t` = list files
  - `f` = next argument is the file to read or write
- `~username` is the home directory of that user
- For example, to create a tar: `tar cvf myfiles.tar dir1 dir2 README`

```
$ tar xvf ~tg459572/LABS/cornellcac_labs.tar  Get all lab files

$ cd envi  Change directory to the files for this session

$ ls -R  List the files in the current folder (envi)
```
Stampede’s Operating System: Linux

$ pwd
(Print the current directory)

$ ls -la
(List the content of the current directory)

$ cd $HOME
(Change the working directory to home directory)

$ cat .profile
(Print the file .profile to the screen)

$ mkdir testdir
(Create the directory, testdir)

$ touch test.txt
(touch renews a file’s timestamp, but here is used to create an empty file)

$ mv test.txt testdir
(Move test.txt into the directory testdir)

$ ls -la testdir
(See the files in folder testdir)

$ rm -r testdir
(Delete the directory and all subdirectories)

$ man ls
(Show the manual page for ls, q to quit)

$ env
(Show all environment/global variables)

$ export newgreeting="Hello World"
(Set an environment variable)

$ echo $newgreeting
(Print the variable newgreeting)
Shells and Startup Scripts on Stampede

Shells:

- bash is the default shell on Stampede
- `login1$ echo $SHELL` To determine your current login shell
- `login1$ cat /etc/shells` To see list of available shells
- To change your default shell, submit a ticket to TUP (chsh won’t work)

Startup Scripts:

- When you log in, system-level startup files execute to allow administrators to enhance and customize the environment
- Enhance your shell environment, not your account
- Don’t use “echo” in startup scripts, will break other tools
- TACC staff recommends that Bash shell users use `~/.profile` rather than `.bash_profile` or `.bash_login`

Bash Users' Startup Files: Quick Start Guide
3. Stampede Overview
Transition: Stampede => Stampede 2

2011: NSF award to TACC to acquire and deploy Stampede
2012: Stampede deployed
2016: NSF award to TACC to acquire and deploy Stampede 2
2017: Stampede 2 fully deployed (Fall)

Over the next 9 months:
• Stampede will remain in production until Stampede 2 is in full production, ~Fall 2017
• Stampede will gradually have fewer nodes
• Stampede has 508 Knights Landing (KNL) nodes for testing; they will later become part of Stampede 2; will gradually increase
• File transfer: During transition Stampede’s home and scratch file systems will be accessible from the Stampede 2 login nodes, and the work file system will be the same
## Transition: Stampede => Stampede 2

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Stampede (nodes)</th>
<th>Stampede-KNL (nodes)</th>
<th>Stampede 2 (nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td></td>
<td>6,400</td>
<td>508</td>
<td></td>
</tr>
<tr>
<td>Late Jan. 2017</td>
<td>Phase 1 start</td>
<td>3,800</td>
<td>508</td>
<td></td>
</tr>
<tr>
<td>Summer 2017</td>
<td>Phase 1 production; Phase 2 start</td>
<td>1,800 (migrate files now)</td>
<td></td>
<td>4,204 (+ new filesystem)</td>
</tr>
<tr>
<td>October 2017</td>
<td>Phase 2 production</td>
<td></td>
<td></td>
<td>5,940</td>
</tr>
</tbody>
</table>
Stampede’s Original Components (Sandy Bridge Cluster)

- login1 through login4 (Sandy Bridge)
- Infiniband network
- Sandy Bridge compute nodes with KNC MIC coprocessors
- Sandy Bridge largemem and GPU compute nodes

---

Stampede Upgrade (KNL Cluster)

- login-knl1 (Haswell)
- OmniPath network
- KNL compute nodes

Image: https://portal.tacc.utexas.edu/user-guides/stampede#intro
The Generic Environment
Available File Systems

File transfer options:
- **Globus Connect**
- **Linux command-line utilities** `scp` & `rsync`
- **Globus' globus-url-copy** command-line utility
- **GSI-OpenSSH**

**rcp/scp/ftp only**

**Home**

**Ranch**

**Scratch**

**Work**

All Nodes
## File System

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Purpose</th>
<th>User Access Limits</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>Source code</td>
<td>5 GB</td>
<td>Backups</td>
</tr>
<tr>
<td>$WORK</td>
<td>Large file storage</td>
<td>1.0 TB</td>
<td>No backup</td>
</tr>
<tr>
<td>$SCRATCH</td>
<td>Large files needed by compute jobs</td>
<td>~8.5PB total</td>
<td>Purged after 10 days</td>
</tr>
<tr>
<td>/tmp</td>
<td>Local disk on batch job node</td>
<td>~80 GB / node</td>
<td>Purged after job ends</td>
</tr>
<tr>
<td>${ARCHIVER}:${ARCHIVE}</td>
<td>Archival tape</td>
<td>Essentially unlimited</td>
<td>Project</td>
</tr>
</tbody>
</table>
Sharing Files with your Research Group
File sharing tutorial: https://portal.tacc.utexas.edu/tutorials/sharing-project-files

- All accounts have a default group when the account is created
- All usernames sharing an allocation should be in a common group
  (If they are not, submit a ticket)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ groups &lt;username&gt;</td>
<td>Display groups that username belongs to</td>
</tr>
<tr>
<td>$ groups</td>
<td>Display groups that you belong to</td>
</tr>
<tr>
<td>$ id –g –n</td>
<td>Display your default group</td>
</tr>
<tr>
<td>$ id &lt;username&gt;</td>
<td>Display username, default group, all groups, for that user</td>
</tr>
<tr>
<td>$ touch test.txt</td>
<td>Create a file</td>
</tr>
<tr>
<td>$ ls -la</td>
<td>Display your files, including group information. Note that the file you just created has your default group ownership</td>
</tr>
<tr>
<td>$ chgrp -v G-&lt;grp number&gt; test.txt</td>
<td>Change the group ownership of a file to a different group (verbose output)</td>
</tr>
<tr>
<td>$ chmod 644 test.txt</td>
<td>Modify permissions so everyone in that group has access</td>
</tr>
</tbody>
</table>
Sharing Files with your Research Group

Want to share files with your colleagues, but you have different default groups? You can

a) submit a ticket to get your default group changed, or

b) create a common folder with the proper settings:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ mkdir /scratch/&lt;grp 1&gt;/&lt;username&gt;/test</td>
<td>Create a directory for everybody to share</td>
</tr>
<tr>
<td>$ chmod g+rwx /scratch/&lt;grp 1&gt;/&lt;username&gt;/test</td>
<td>Set permissions allow group read/write/execute (also make sure the parent dir's permissions aren't too restrictive).</td>
</tr>
<tr>
<td>$ chgrp G-&lt;grp 2&gt; test</td>
<td>Change its group via 'chgrp' to the common group</td>
</tr>
<tr>
<td>$ chmod g+s test</td>
<td>Set the setgid bit, so that everything created underneath it will inherit its group</td>
</tr>
<tr>
<td>$ umask 177</td>
<td>Everyone in the group should use an appropriate umask such as 002 or 117, so that files they create are actually group readable and writable. (Put this into a login script!)</td>
</tr>
</tbody>
</table>
File System

$ lfs quota -u <username> $HOME  
see quota limits & usage
$ lfs quota -u <username> $WORK  
$ lfs quota -u <username> $SCRATCH
$ cd  
change directory to $HOME
$ pwd
$ cdw  
change directory to $WORK
$ pwd
$ cds  
change directory to $SCRATCH
$ pwd
$ du -sh  
see how much space is available in the 
current user-owned directory
$ df -k .  
see the amount of disk space used in a file 
system, “.” meaning in the current directory
4. Software
Software

Use the **module** utility on Stampede to provide a consistent, uniform method to access software

- Loads specific versions of libraries/executables
- Manages dependencies between multiple compilers & software stacks
- Works in your batch file, Makefile, and scripts
- Affects $PATH, $MANPATH, $LIBPATH
- Order matters! First choose compiler, then application software.
- Warning: the module system treats the Sandy Bridge and KNL clusters as separate systems.

**Software** search available on XSEDE

**Lmod** is TACC’s Module System
Setting your Default Software Environment

Set and save your personal default module environment:

$ module reset  # return to the default environment
$ module load ddt
$ module load fftw3
$ module save   # will load at login or restore

Create a named collection of modules for reliability and repeatability:

$ module save chemtools

... 

$ module restore chemtools
Module
This utility is used to set up your PATH and other environment variables:

$ module help
$ module avail
$ module list
$ module load boost
$ module unload boost
$ module help <module_name>
$ module spider
$ module spider petsc
$ module load impi
$ module list
$ module reset
5. Compiling
Compiling Serial Code

- The default compilers on Stampede are Intel C++ and Fortran (& support KNL)
- Don’t try to compile once to run on both KNC and KNL (Network stack incompatible)
- Use man or –help option, e.g. man icc
- Compilers are available on login and compute nodes, but--
- The login node is a Haswell, not KNL, processor; use the "-xMIC-AVX512" switch at both compile and link time for KNL:

```
knl-login1$ icc -xMIC-AVX512 -o mycode.exe mycode.c
knl-login1$ ifort -xMIC-AVX512 -o mycode.exe mycode.f90
```

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Language</th>
<th>File Extension</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>icc</td>
<td>C</td>
<td>.c</td>
<td>icc compiler_options prog.c</td>
</tr>
<tr>
<td>icpc</td>
<td>C++</td>
<td>.C, .cc, .cpp, .cxx</td>
<td>icpc compiler_options prog.cpp</td>
</tr>
<tr>
<td>ifort</td>
<td>F77</td>
<td>.f, .for, .ftn</td>
<td>ifort compiler_options prog.f</td>
</tr>
<tr>
<td>ifort</td>
<td>F90</td>
<td>.f90, .fpp</td>
<td>ifort compiler_options prog.f90</td>
</tr>
</tbody>
</table>
Compiler Options

• Use compiler options to achieve optimal performance.
• To obtain best results:
  – Select the appropriate optimization level
  – Target the architecture of the computer (CPU, cache, memory system)
  – Allow for interprocedural analysis (inlining, etc.)
• No single answer for all cases; test different combinations.

<table>
<thead>
<tr>
<th>Optimization Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O0</td>
<td>Fast compilation, full debugging support. Automatically enabled if using -g.</td>
</tr>
<tr>
<td>-O1, -O2</td>
<td>Low to moderate optimization, partial debugging support:</td>
</tr>
<tr>
<td>-O3</td>
<td>Aggressive optimization - compile time/space intensive and/or marginal effectiveness; may change code semantics and results (sometimes even breaks code!)</td>
</tr>
</tbody>
</table>

See the User Guide for additional compiler options.
Makefiles

$ cd $HOME/envi/using_makefiles

$ cat Makefile
Read over the Makefile

$ make
Compile the program, generate a.out

$ make
Reports “up to date”, i.e. not recompiled

$ touch suba.f90
Simulate changing a file

$ make
suba.f90 (and only suba.f90) is recompiled
6. Timing
Timers

• Time your code to see how long your program runs and estimate if it’s having gross difficulties. Gauge effectiveness of code and software changes.
• Wall-clock time in a dedicated environment is most accurate
• \texttt{/usr/bin/time -p} is preferred over the shell’s time command \texttt{(-p specifies traditional precision output in seconds)}

```
$ cd $HOME/envi/intro
$ make
g++ hello.c -o hello
$ /usr/bin/time -p ./hello
Hello world!
real 0.01
user 0.00
sys 0.01
$ 
```

You can also \texttt{time specific sections} of your code by inserting timer calls before and after important sections.
Profilers: gprof (GNU profiler)

- gprof reports a basic profile of time spent in each subroutine
- Find the most time-consuming routines, the hotspots
- As with all profiling tools, the code must be instrumented to collect the timing data and then executed to create a raw-date report file.
- Read the data file into an ASCII report or a graphic display.
- Instrument the code by recompiling using the -pg option (Intel)
- https://portal.tacc.utexas.edu/user-guides/stampede#tools-profilers
- Also available: PerfExpert

```
$ cd $HOME/envi/precision
$ ifort -pg precision.f90
$ a.out
$ gprof
```

Instrument code with -pg
produce gmon.out trace file
reads gmon.out (default args: a.out gmon.out) report sent to STDOUT
7. Editing Files
vi (short for “visual”)

- “vi filename” will open it or create it if it doesn’t exist.
- Command mode: keystrokes are commands
- Input mode: keystrokes are text you are adding to the file
- Last line mode: start with : end with <return>
- Examples:
  - i Insert characters before current position (use ESC to exit)
  - dd Delete current line
  - R Overwrite existing text (until ESC)
  - u Undo last operation
  - :wq Writes a file to disk and exit editor
  - :q! Quit without saving

http://www.tuxfiles.org/linuxhelp/vimcheat.html
nano

• The commands for all operations are preceded by the Control key:

• 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 (search with regular expressions, etc..)
emacs

• emacs is actually a lisp interpreter with extensions to use it as a text editor

• Can perform the same operations as in vi

• Uses Control or ESC followed by keystroke combinations to execute commands

• “Hard to learn, easy to use”

http://emacswiki.org/emacs/ReferenceCards
Files edited in Windows?  Warning:

- Beware line ending differences
- Do not use MS Word (e.g.) to create scripts or programs
- Do not copy/paste from PDF files
- Linux filenames are case-sensitive
- Blanks in filenames can be problematic
- Use Wordpad rather than Notepad if necessary
- dos2unix utility can convert text files with DOS or MAC line breaks to Unix line breaks
8. Batch Job Submission: SLURM
Getting to the Compute Nodes

Four ways to get to the back end (compute nodes):

• SLURM batch job: `sbatch <batchfilename>`
• Interactive session: `srun` (SLURM) or `idev` (TACC)
• Run special app 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`, `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
Queue: Job script waits for resources.
Master: Compute node that executes the job script, launches all MPI processes.
# Stampede Batch Environment Queues

## Sandy Bridge Queue

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Max Runtime</th>
<th>Max Nodes/Procs</th>
<th>Max Jobs in Queue</th>
<th>Queue Multiplier</th>
<th>Max Jobs in Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>development</td>
<td>2 hrs</td>
<td>16 / 256</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>largemem</td>
<td>48 hrs</td>
<td>3 / 96</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>normal</td>
<td>48 hrs</td>
<td>256 / 4K</td>
<td>50</td>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

## KNL Queue

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Max Runtime</th>
<th>Max Nodes and Associated Cores per Job</th>
<th>Max Jobs in Queue</th>
<th>Charge per node hour</th>
<th>Configuration (Memory-Cluster)</th>
</tr>
</thead>
<tbody>
<tr>
<td>development</td>
<td>2 hrs</td>
<td>4 nodes (272 cores)</td>
<td>1</td>
<td>16 SU</td>
<td>Cache-Quadrant</td>
</tr>
<tr>
<td>normal</td>
<td>48 hrs</td>
<td>80 nodes* (5440 cores)</td>
<td>10</td>
<td>16 SU</td>
<td>Cache-Quadrant</td>
</tr>
<tr>
<td>Flat-Quadrant</td>
<td>48 hrs</td>
<td>40 nodes* (2720 cores)</td>
<td>5</td>
<td>16 SU</td>
<td>Flat-Quadrant</td>
</tr>
<tr>
<td>Flat-All2All</td>
<td>12 hrs</td>
<td>2 nodes* (136 cores)</td>
<td>1</td>
<td>16 SU</td>
<td>Flat-All-to-All</td>
</tr>
<tr>
<td>Flat-SNC-4</td>
<td>12 hrs</td>
<td>2 nodes* (136 cores)</td>
<td>1</td>
<td>16 SU</td>
<td>Flat-SNC-4</td>
</tr>
</tbody>
</table>

[access by request](https://portal.tacc.utexas.edu/user-guides/stampede#knl-running)
Batch on Stampede: Select SLURM Commands

- **showq** - view summary of jobs in the batch system (not SLURM)
  
  ```
  showq | more
  showq -u <userid>
  ```

- **sacct** - report job or job step accounting information.

- **salloc** - allocate resources for a job in real time.

- **sbatch** - submit a job script for later execution.
  
  ```
  sbatch <batchfilename>
  ```

- **sbcast** - transfer a file from local disk to local disk on the job nodes.

- **scancel** - cancel a pending or running job or job step.

  ```
  scancel <jobid>
  ```

- **sinfo** - lists the availability and status of queues

  ```
  sinfo -o "%20P %5a %.10l %16F"
  ```

- **squeue** - reports the state of jobs or job steps.

  ```
  squeue | more
  squeue -u <userid>
  ```

- **srun** - submit an interactive job (this example: 1-node 16 core)

  ```
  srun --pty -n 16 -t 00:30:00 -p development -A <acct no> /bin/bash -l
  ```

- **ibrun** – run an MPI program (put this command in your batch script for MPI jobs)

Man pages exist for all SLURM daemons, commands, and API functions. The command option **--help** also provides a brief summary of options. Note that the command options are all case insensitive.
### squeue Options, Output, and Job State Codes

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Output Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i &lt;interval&gt;</td>
<td>Repeatedly report at intervals (in seconds)</td>
<td>- JOBID: job id assigned to the job</td>
</tr>
<tr>
<td>-j &lt;job_list&gt;</td>
<td>Displays information for specified job(s)</td>
<td>- USER: user that owns the job</td>
</tr>
<tr>
<td>-p &lt;part_list&gt;</td>
<td>Displays information for specified partitions (queues).</td>
<td>- STATE: current job status.</td>
</tr>
<tr>
<td>-t &lt;state_list&gt;</td>
<td>Shows jobs in the specified state(s)</td>
<td>- PD: Pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- R: Running</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- S: Suspended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CA: Configuring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CG: Completing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CD: Completed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CF: Cancelled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- F: Failed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TO: Timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- PR: Preempted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- NF: Node_fail</td>
</tr>
</tbody>
</table>

1/23/2017

www.cac.cornell.edu
Batch Job Script Example: MPI

#!/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=myemail@myuniv.edu
#SBATCH --mail-type=ALL

#SBATCH -A TG-TRA120006 # necessary if you have multiple project accounts

module load fftw3 # You can also load modules before launching job
module list

ibrun ./main.exe # Use ibrun for MPI codes. Don’t use mpirun or srun.
Batch Job Script Example: Serial

#!/bin/bash
# Don't miss this line!

#---------------------------------------------------
# Generic SLURM script
#---------------------------------------------------

#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 serial # queue
#SBATCH -N 1 -n 1 # one node and one task
#SBATCH -t 00:30:00 # max time

#SBATCH --mail-user=myemail@myuniv.edu
#SBATCH --mail-type=ALL

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

module load fftw3
module list

./main.exe
Batch on Stampede: SLURM Commands

1. Use `sinfo -o "%20P %5a %.10l %16F"` to list queues, nodes, and system state
2. Issue `showq` to show all queued jobs
3. `$ idev -r -A TG-TRA140011`
4. Issue `cat` to take one last look at the batch script
   
   ```bash
   #!/bin/bash
   #SBATCH -J myMPI  # Job name
   #SBATCH -o myjob.%j.out  # stdout file (%j expands to jobId)
   #SBATCH -p development  # Queue name
   #SBATCH -N 2  # Total number of nodes requested (16 cores/node)
   #SBATCH -n 32  # Total number of mpi tasks requested
   #SBATCH -t 01:30:00  # Run time (hh:mm:ss) - 1.5 hours
   #SBATCH --A TG-TRA140011  # Specify project/allocation number
   ibrun ./a.out
   ```
5. Compile: `mpicc -xMIC-AVX512 -O3 mpihello.c` OR `mpif90 -xMIC-AVX512 -O3 mpihello.f90`
6. Issue `sbatch` to submit a batch script: `sbatch job`
7. Issue `squeue -u <your username>` to see the job status
8. Run `scancel <jobid>` to cancel the job, or `cat myjob.###.out` to view your output
Resource Allocation on SLURM

• \(-N\) – Total nodes requested
• \(-n\) – Number of tasks per node
• Explicitly specify both "-N" (total nodes) and "-n" (tasks per node) in your Slurm job script; you’ll get 272 tasks per node if only \(n\) is specified.

Serial Job
#SBATCH -N 1
#SBATCH -n 1

2 Tasks
#SBATCH -N 1
#SBATCH -n 2

4 Tasks Parallel
#SBATCH -N 2
#SBATCH -n 4
9. Play Nice
TACC’s Acceptable Use Policy

A large, shared, resource like Stampede works best when everyone practices good citizenship

- **Respect the shared filesystems**
  - Run jobs should access files in $WORK or $SCRATCH, not in $HOME
  - Don’t exceed ~three concurrent file transfer sessions
  - Limit I/O intensive sessions (lots of reads and writes to disk)

- **Don't run programs on the login nodes**
  - Login nodes are for compiling, file management, managing batch jobs, modest post-processing
  - Login nodes are not for running programs; submit a batch job instead
10. Help
Questions?

Help Tickets
• CAC help@cac.cornell.edu Email to submit a ticket
• TACC https://portal.tacc.utexas.edu/ Log in, Open a ticket
• XSEDE https://portal.xsede.org/ Log in, Help, Help Desk

Training/Documentation
• https://cvw.cac.cornell.edu/Topics/ Online training
• https://portal.tacc.utexas.edu/user-guides/stampede User Guide
• https://portal.xsede.org/documentation-overview Getting Started

• For shell commands, try man <command> or man -k <command>
Appendix
The precision program computes and prints $sin(\pi)$. The $\pi$ constant uses “E” (double precision) format in one case and “D” (single) in the other.

```bash
$ cd $HOME/envi/precision
$ cat precision.f90
$ module load intel
$ ifort precision.f90
$ ./a.out
```