



# Programming Environment

Cornell Center for Advanced Computing  
October 23, 2013

*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. Help



# 1. Accessing Stampede



## Before you Start

- Get an XSEDE Portal Account : <https://portal.xsede.org/>
- 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)
- Activate your account 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
  - To activate, log into TACC portal, activation is immediate.
- Reset password on TACC portal. Takes 30 minutes to propagate.



## Logging into XSEDE Resources:

- Command Line (Unix/Linux, or Mac Terminal window) – ssh
- SSH / telnet client – e.g. Putty or Secure Shell Client
- Single Sign On (SSO) from the XSEDE User Portal
- ... [and more](#)



## Login with SSH:

- SSH Secure Shell - client for Windows
- You will be connected to `login#.stampede.tacc.utexas.edu`
- **Do not** overwrite `~/.ssh/authorized_keys`

Using the SSH Secure Shell client, login to `stampede.tacc.utexas.edu`:

All Programs | ClassFiles | SSH Secure Shell | Secure Shell Client  
use Host Name: `stampede.tacc.utexas.edu`



## Login with SSO

- Go to the XSEDE User Portal: [portal.xsede.org](http://portal.xsede.org)
- Log in
- Go to 'My XSEDE' tab
- Choose 'Accounts' on the nav bar
- Click on 'Login' in the row for Stampede
- Note: you may not have the same username on all resources

### Login using the XSEDE portal

The screenshot shows the XSEDE User Portal interface. The 'MY XSEDE' tab is selected in the navigation bar, and the 'Accounts' sub-tab is also selected. Below the navigation bar, there is a table of resources with columns for Resource Name, Login Name, Institution, Username, and Connect. The 'Stampede' row is highlighted, and the 'Login' link in the 'Connect' column is circled in red.

RESOURCE NAME	LOGIN NAME	INSTITUTION	USERNAME	CONNECT
Blacklight	<a href="http://blacklight.psc.xsede.org">blacklight.psc.xsede.org</a>	PSC	mehringe	<a href="#">Login</a>
Gordon Compute Cluster	<a href="http://gordon.sdsc.xsede.org">gordon.sdsc.xsede.org</a>	SDSC	mehringe	<a href="#">Login</a>
Gordon ION	<a href="http://gordon.sdsc.xsede.org">gordon.sdsc.xsede.org</a>	SDSC		
Keeneland	<a href="http://gsissh.keeneland.gatech.xsede.org">gsissh.keeneland.gatech.xsede.org</a>	NICS		
Keeneland-KIDS	<a href="http://gsissh.kids.gatech.xsede.org">gsissh.kids.gatech.xsede.org</a>	NICS		
Kraken	<a href="http://kraken-gsi.nics.xsede.org">kraken-gsi.nics.xsede.org</a>	NICS	shm7	<a href="#">Login</a>
Lonestar	<a href="http://lonestar.tacc.xsede.org">lonestar.tacc.xsede.org</a>	TACC	tg459571	<a href="#">Login</a>
Longhorn	<a href="http://longhorn.tacc.xsede.org">longhorn.tacc.xsede.org</a>	TACC	tg459571	<a href="#">Login</a>
Mason	<a href="http://mason.iu.xsede.org">mason.iu.xsede.org</a>	IU		
Open Science Grid	<a href="http://osg-xsede.grid.iu.edu">osg-xsede.grid.iu.edu</a>	OSG	shm7	<a href="#">Login</a>
Stampede	<a href="http://stampede.tacc.xsede.org">stampede.tacc.xsede.org</a>	TACC	tg459571	<a href="#">Login</a>
Trestles	<a href="http://trestles.sdsc.xsede.org">trestles.sdsc.xsede.org</a>	SDSC	mehringe	<a href="#">Login</a>



## 2. Login Environment





## Account Info

Note your account number in the splash screen.

```
----- Project balances for user tg459571 -----
| Name           Avail SUs   Expires |
| TG-TRA120006   49998                |
-----
----- Disk quotas for user tg459571 -----
| Disk           Usage (GB)   Limit   %Used   File Usage   Limit   %Used |
| /home1         0.0         5.0     0.06    43          150000  0.03 |
| /work          0.0        400.0   0.00    3          30000000 0.00 |
-----
```



## 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`

Get the lab files:

```
$ tar xvf ~tg459572/LABS/envi.tar
```

Change directory to the envi directory:

```
$ cd envi
```

List the lab files:

```
$ ls -la
```



## Experiment with Linux commands on Stampede

\$ pwd	(Print the current directory)
\$ ls -la	(List the content of the current directory)
\$ cd \$HOME	(Change the working directory to your home directory)
\$ cat .login	(Print the file <i>.login</i> to the screen)
\$ mkdir testdir	(Create the directory, <i>testdir</i> )
\$ 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)
\$ rm -r testdir	(Delete the folders and all subfolders)
\$ man ls	(Show the manual page for ls, <b>q</b> to quit)
\$ env	(Show all environment/global variables)
\$ export newgreeting="Hello World"	(Set an environment variable)
\$ echo \$newgreeting	( Print the variable <i>newgreeting</i> )

### [Command comparison – DOS to Unix](#)



## Shells and Startup Scripts on Stampede

### Shells:

- bash is the default shell on Stampede
- TACC supports most major shells, e.g. csh, tcsh, zsh ...
- To change your default shell, submit a ticket (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
- Put your personal customization in `.login_user`

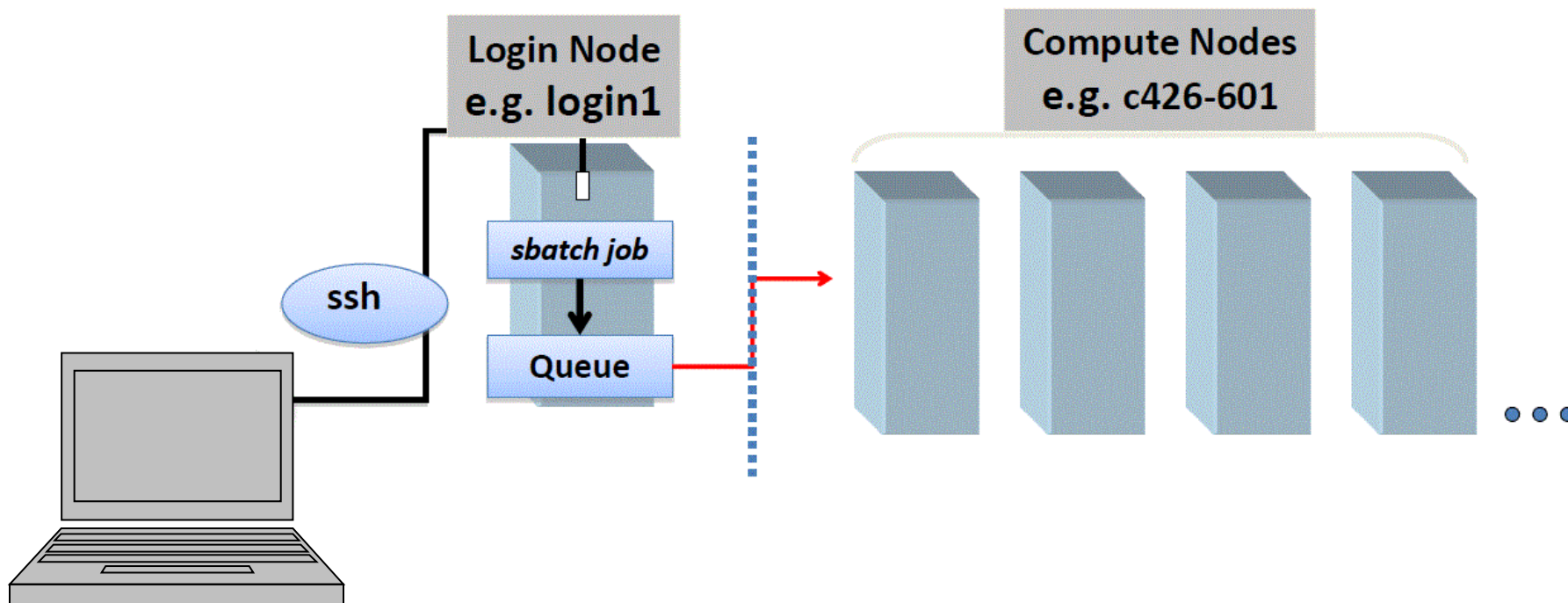
<http://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide#compenv-startup>



## 3. Stampede Overview

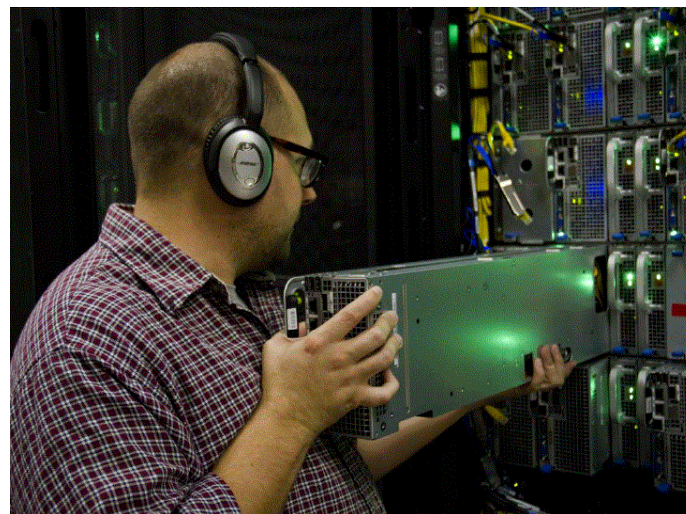
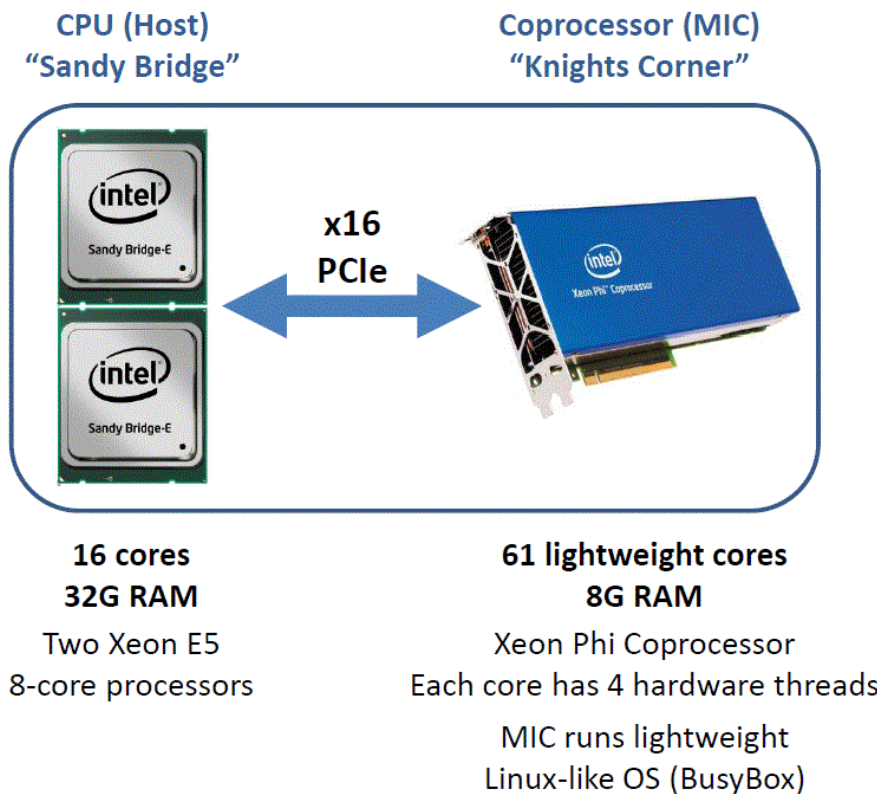


# The Generic Environment





## Typical Stampede Node ( = blade )

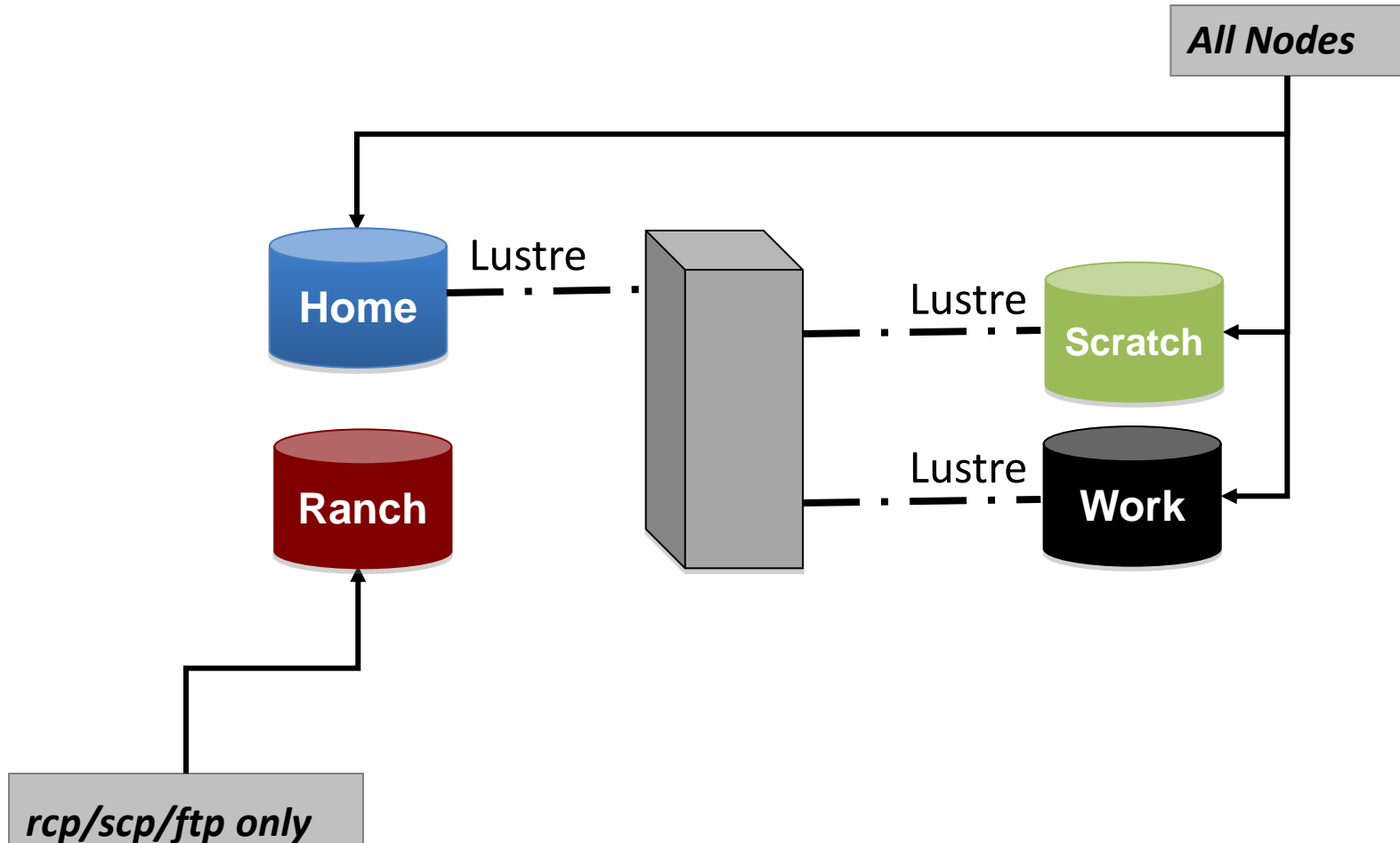




System	Stampede	Memory
Nodes	~6400 (in 160 racks) → 96,000+ total cores	
Typical Node	16 cores: 2 cpus/node x 8 cores/cpu	32GB RAM
	61 cores on MIC coprocessor	8G RAM
Special Nodes	16 large memory nodes (32 Xeon cores)	1TB/node RAM
	128 GPU nodes (w/ NVIDIA Kepler 2 & MIC)	2GB/core
	Login nodes (don't have MIC)	
CPUs	Intel Sandy Bridge Intel Xeon Phi coprocessor	
Interconnect	56Gb FDR IB	
Disk	14PB Lustre (IB)	



# Available File Systems





# File System

Environment Variable	Purpose	User Access Limits	Lifetime
\$HOME	Source code	5 GB	Backups
\$WORK	Large file storage	400 GB	No backup
\$SCRATCH	Large files needed by compute jobs	~8.5PB total	Purged after 10 days
/tmp	Local disk on batch job node	~80 GB / node	Purged after job ends
\${ARCHIVER}:%ARCHIVE	Archival tape	Essentially unlimited	Project



## Sharing Files with your Research Group

- 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)

<code>\$ groups &lt;username&gt;</code>	Display groups that username belongs to
<code>\$ groups</code>	Display groups that you belong to
<code>\$ id -g -n</code>	Display your default group
<code>\$ id &lt;username&gt;</code>	Display username, default group, all groups, for that user
<code>\$ touch test.txt</code>	Create a file
<code>\$ ls -la</code>	Display your files, including group information. Note that the file you just created has your default group ownership
<code>\$ chgrp -v G-803077 test.txt</code>	Change the group ownership of a file to a different group (verbose output)
<code>\$ chmod 644 test.txt</code>	Modify permissions so everyone in that group has access



## 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:

<code>\$ mkdir /scratch/01871/apb18/test</code>	Create a directory for everybody to share
<code>\$ chmod g+rwx /scratch/01871/apb18/test</code>	Set permissions allow group read/write/execute (also make sure the parent dir's permissions aren't too restrictive).
<code>\$ chgrp G-803077 test</code>	Change its group via 'chgrp' to the common group
<code>\$ chmod g+s test</code>	Set the setgid bit, so that everything created underneath it will inherit its group.
<code>\$ umask 177</code>	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!)



# File System

```
$ ls quota -u <username> $HOME           see quota limits & usage
$ ls quota -u <username> $WORK
$ ls 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 used 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 and software stacks
- Works in your batch file, Makefile, and scripts, but not on MICs
- Affects \$PATH, \$MANPATH, \$LIBPATH
- Order matters! First choose compiler, then application software.

[Software](#) available on Stampede

[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	{lists options}
\$ module avail	{lists available modules}
\$ module list	{lists loaded modules}
\$ module load boost	{add a module}
\$ module unload boost	{remove a module}
\$ module help <module_name>	{module-specific help}
\$ module spider	{lists all modules}
\$ module spider petsc	{list all versions of petsc}



## 5. Compiling



## Compiling Serial Code

- The default compilers on Stampede are Intel C++ and Fortran
  - These are the only compilers that support the Phi coprocessors
- Compilers are available on login and compute nodes
  - But not on MIC coprocessors; compile from a Sandy Bridge host
- Use **man** or **-help** option, e.g. **man icc**.

Compiler	Language	File Extension	Example
icc	C	.c	<code>icc compiler_options prog.c</code>
icpc	C++	.C, .cc, .cpp, .cxx	<code>icpc compiler_options prog.cpp</code>
ifort	F77	.f, .for, .ftn	<code>ifort compiler_options prog.f</code>
ifort	F90	.f90, .fpp	<code>ifort compiler_options prog.f90</code>

- Use the **module** command to list modules & versions & to change the default compiler.
- Three versions of gcc suite are also available
- Other specialized compilers also supported, e.g. cuda support (nvcc):  
**module load cuda**



## 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.

### Optimization Level Description

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

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.f**                      Simulate changing a file

**\$ make**                              suba.f (and only suba.f) 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
- **/usr/bin/time -p** is preferred over the shell's time command ( -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 [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-data report file.
- Read the data file into an ASCII report or a graphic display.
- Instrument the code by recompiling using the `-pg` option (Intel)
- More detail can be found in the [Profiling and Debugging](#) Virtual Workshop module.

```
$ cd $HOME/envi/precision
$ ifort -pg precision.f90      instrument code with -pg
$ a.out                      produce gmon.out trace file
$ gprof                       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:
  - ^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 (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>



## Use Your Computer's Editor

Copying the file to your computer might be quicker than learning a new editor. Use a simple file transfer client:

Start menu

All Programs

Class Files

SSH Secure Shell

Secure File Transfer Client ← Right click, "Pin to Start Menu"

Start Secure File Transfer Client

Use Quick Connect, specify hostname `stampede.tacc.utexas.edu`

In the left pane, navigate to the desktop.

Drag files between panes to copy.

\*\* Beware line ending differences!



## 8. Batch Job Submission: SLURM



## Getting to the Compute Nodes

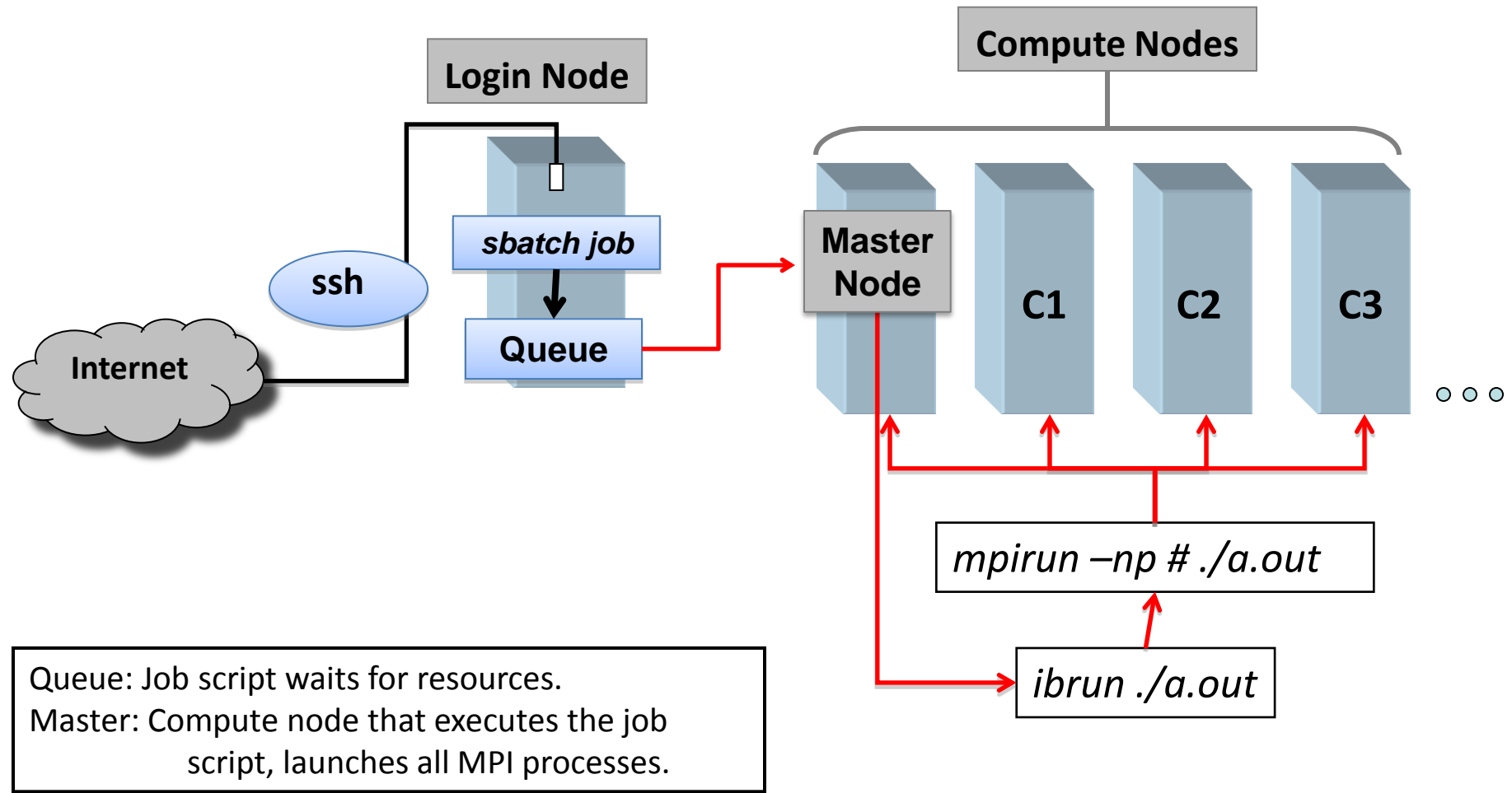
Four ways to get to the back end (compute nodes):

- SLURM batch job: **sbatch <batchfilename>**
- 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

# Batch Submission Process







## Stampede Batch Environment Queues

Queue Name	Max Runtime	Max Nodes/Procs	SU Charge Rate	Purpose
normal	48 hrs	256 / 4K	1	normal production
development	4 hrs	16 / 256	1	development nodes
largemem	24 hrs	4 / 128	2	large memory 32 cores/node
serial	12 hrs	1 / 16	1	serial/shared_memory
large	24 hrs	1024 / 16K	1	large core counts **
request	24 hrs	--	1	special requests
normal-2mic	24 hrs	128 / 2k	1	Production MIC nodes w/ 2 co-processors
normal-mic	48 hrs	256 / 4k	1	production MIC nodes
gpu	24 hrs	32 / 512	1	GPU nodes
gpudev	4 hrs	4 / 64	1	GPU development nodes
vis	8 hrs	32 / 512	1	GPU nodes + VNC service
visdev	4 hrs	4 / 64	1	Vis development nodes (GPUs + VNC)

<http://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide#running-slurm-queue>



## 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** - reports the state of partitions and nodes managed by SLURM.  
sinfo -o "%20P %5a" *ignore queue limits reported*
- **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 20130418HPC /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

-i <interval>	Repeatedly report at intervals (in seconds).
-j <job_list>	Displays information for specified job(s)
-p <part_list>	Displays information for specified partitions (queues).
-t <state_list>	Shows jobs in the specified state(s)

JOBID	job id assigned to the job
USER	user that owns the job
STATE	current job status.

PD	Pending
R	Running
S	Suspended
CA	Configuring
CG	Completing
CD	Completed
CF	Cancelled
F	Failed
TO	Timeout
PR	Preempted
NF	Node_fail



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

./main.exe
```



## Batch on Stampede: SLURM Commands

1. Use **sinfo -o "%20P %5a"** to list queues, nodes, and system state
2. Issue **showq** to show all queued jobs
3. Issue **srun** to run simple commands (e.g. an interactive shell) (ctrl-D to exit)  
\$ srun --pty -A TG-TRA120006 -p serial -t 00:10:00 -n 1 -N 1 /bin/bash -l
4. Issue **cat** to take one last look at the batch script  
\$ cd \$HOME/envi/batch  
\$ cat job  
#!/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  
ibrun ./a.out
5. Compile: **mpicc -O3 mpihello.c -OR- mpif90 -O3 mpihello.f90**
6. Issue **sbatch** to submit a batch script  
\$ sbatch job  
sbatch: Submitted batch job 469
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

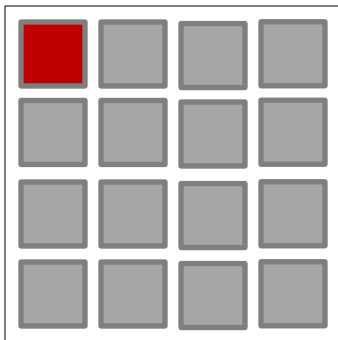


## Resource Allocation on SLURM

- **-N** – Number of node requested
- **-n** – Number of tasks to run

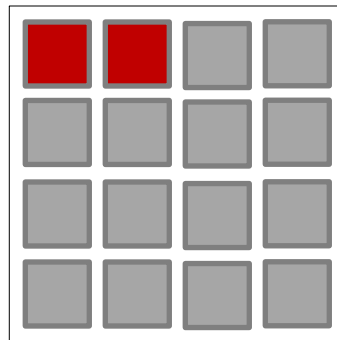
Serial Job

```
#SBATCH -N 1  
#SBATCH -n 1
```



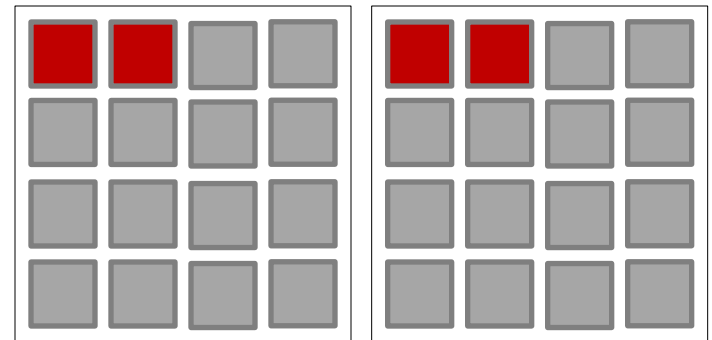
2 Tasks

```
#SBATCH -N 1  
#SBATCH -n 2
```



4 Tasks Parallel

```
#SBATCH -N 2  
#SBATCH -n 4
```





## 9. Help





## Questions?

- CAC [help@cac.cornell.edu](mailto:help@cac.cornell.edu)
- portal.xsede.org -> **Help** (in the navigation bar)
- portal.xsede.org -> My XSEDE -> **Tickets**
- portal.xsede.org -> Documentation -> **Knowledge Base**
- User Guide(s), Usage Policies, etc. and associated links:  
<http://www.tacc.utexas.edu/user-services>
- Try `man <command>` or `man -k <command>` or `<command> -h`  
or `<command> -help`



# Appendix

# Precision

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.

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