Thanks to Dan Stanzione, Bill Barth, and Robert McLay for their materials developed at TACC and XSEDE that were incorporated into this talk.
1. Orientation
2. Allocations
3. Accessing Ranger
4. Login Environment
5. Ranger Overview
6. Software
7. Timing
8. Editing Files
9. Batch Job Submission
10. Miscellaneous
1. Orientation
Orientation

• **XSEDE** – Extreme Science and Engineering Discovery Environment  
  – Cyber infrastructure funded by NSF; a single virtual system  
  – 9 supercomputers, 3 visualization systems, and 9 storage systems  
  – 16 partner institutions  

• **TACC** – Texas Advanced Computing Center  
  – **Ranger** – Sun Constellation Linux Cluster, 90% dedicated to XSEDE  
  – **Longhorn** – 256-Node Dell Visualization Cluster  

• **CAC** – Cornell Center for Advanced Computing  
  – **HPC Systems** – general use and private clusters  
  – **Red Cloud** – on-demand research computing service
XSEDE

Extreme Science and Engineering Discovery Environment
2. Allocations
XSEDE Allocations

Allocations provide computing, storage, and support services.

- **Startup** – for testing and preparing allocation request
  - Up to 200,000 core-hours, for 1 year
  - Submit Abstract, Awarded every 2 weeks

- **Research** – usually for funded research project
  - Unlimited core-hours, for 1 year
  - 10 page Request, Awarded quarterly

- **Education** – for classroom instruction and training sessions
  - Up to 200,000 core-hours, for 1 year
  - Submit CV and abstract, Awarded/2 weeks

https://portal.xsede.org/web/guest/allocations
Campus Champions

- The Campus Champions program supports campus representatives as a local source of knowledge on XSEDE resources.
- Contact your Campus Champion for
  - Trial allocation
  - Information on XSEDE and cyberinfrastructure resources

https://www.xsede.org/campus-champions

Contact Susan Mehringer at shm7@cornell.edu
4. Accessing Ranger
Login with SSH

- **Putty** for Windows
- Built-in as “ssh” for Linux or Mac

- You will be connected to login3.ranger.tacc.utexas.edu or login4
- **Do not** overwrite ~/.ssh/authorized_keys

Login now to ranger.tacc.utexas.edu:

% ssh [username]@ranger.tacc.utexas.edu

-or-

All Programs | ClassFiles | putty
use Host Name: ranger.tacc.utexas.edu
Login with SSO

- Log into the XSEDE User Portal
- Go to ‘My XSEDE’ tab
- Go to the ‘Accounts’ link
- Use the appropriate ‘login’ link

Login using the XSEDE portal
Single Sign On (SSO)

• SSO is the default method; you’ll need to file a ticket to request a direct access password to the machine.
• SSO allows you to use just one username and password (your User Portal one) to log into every digital service on which you have an account.
• The easiest way to use SSO is via the XSEDE User Portal, but you can also use SSO via a desktop client or with an X.509 certificate.
• After you authenticate using SSO with your User Portal username and password, you will be recognized by all XSEDE services on which you have account, without having to enter your login information again for each resource.
VNC

- **VNCServer**
  - used to start a VNC (Virtual Network Computing) desktop.
  - a Perl script which simplifies the process of starting an Xvnc server.
  - can be run with no options at all. In this case it will choose the first available display number
- VNCServer copies a bitmap of the X-Windows screen across.
- Can be much less chatty than X-Windows.
- Good for remote graphics.
- VNCServer screen 4 uses TCP/IP port 5904.
Connect with VNC

• Start VNC on Ranger
  – First ssh normally.
  – Type “vncserver” and look for screen number, for example. “4”.

• Connect with a client
  – RealVNC or TightVNC on Windows
  – On Linux, vinagre or vncviewer
  – Connect to “ranger.tacc.utexas.edu:4” or your port number

• Be sure to kill it when you are done
  – vncserver –kill 4
5. Login Environment
Note your account number at bottom of splash screen.

<table>
<thead>
<tr>
<th>Name</th>
<th>Avail SUs</th>
<th>Expires</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG-TRA120006</td>
<td>5000</td>
<td>2013-01-04</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>/share</td>
<td>1.1</td>
<td>6.0</td>
<td>17.75</td>
<td>10535</td>
<td>100000</td>
<td>10.54</td>
</tr>
<tr>
<td>/work</td>
<td>0.0</td>
<td>200.0</td>
<td>0.00</td>
<td>1</td>
<td>200000</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`

Get the lab files:
% `tar xvf ~tg459572/LABS/envi.tar`

Change directory to the envi directory:
% `cd envi`
List the lab files:
% `ls -la`
% echo $SHELL
% env   (show environment variables – persists)
% set   (show shell variables – current shell only)
% pwd
% ls -la
% df -h
% uname -a
% echo $WORK
% cd $HOME
% cat .login
% cat /usr/local/etc/login
% cat .login_user (create, then edit this one to personalize)
% chsh -l
% man chsh (q to quit)
Startup Scripts & Modules

- Login shell is set with "chsh -s <login shell>"
  - Takes some time to propagate (~1 hour)
- "chsh -l" will list available login shells.
- Each shell reads a set of configuration scripts.
- Bourne-type shells (Bourne, Korn, and Bash Shells)

System-wide config scripts:
Bash: /etc/tacc/profile
     /etc/tacc/bashrc
     /etc/profile.d/<xxx>.sh
Tcsh: /etc/tacc/csh.cshrc
     /etc/tacc/csh.login
     /etc/profile.d/<xxx>.csh

User-customizable config script:
Bash: ~/.bashrc, ~/.profile
Tcsh: ~/.cshrc, ~/.login
3. Ranger Overview
The Generic Environment

SSH to Head Node

Scheduler

Head Node
Available File Systems (Ranger)

- **Home**
- **Ranch**
- **Scratch**
- **Work**

**Lustre**

**SunBlade x6420**
4 Quad-Core CPUs

**rcp/scp/ftp only**

**All Nodes**
## File System on Ranger

<table>
<thead>
<tr>
<th>Environmental Variable</th>
<th>User Access Limits</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>6 GB quota</td>
<td>Project</td>
</tr>
<tr>
<td>$WORK</td>
<td>350 GB quota</td>
<td>Project</td>
</tr>
<tr>
<td>$SCRATCH</td>
<td>~400 TB</td>
<td>10 Days</td>
</tr>
</tbody>
</table>

```bash
%lfs quota -u <username> $HOME
%lfs quota -u <username> $WORK
%lfs quota -u <username> $SCRATCH
%cd     change directory to $HOME
%pwd    
%cdw    change directory to $WORK
%pwd    
%c当地 change directory to $SCRATCH
%pwd    
```
# TACC HPC/DATA Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Ranger</th>
<th>Lonestar</th>
<th>Longhorn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>HPC</td>
<td>HPC</td>
<td>Data Analysis</td>
</tr>
<tr>
<td>Nodes</td>
<td>3,936</td>
<td>1,888</td>
<td>256</td>
</tr>
<tr>
<td>CPUS/node x cores/CPUS</td>
<td>4 x 4</td>
<td>2 x 6</td>
<td>2 x 4 + 2GPUs</td>
</tr>
<tr>
<td>Total cores</td>
<td>62,976</td>
<td>22,656</td>
<td>2,048</td>
</tr>
<tr>
<td>CPUS</td>
<td>AMD Barcelona 2.3GHz</td>
<td>Intel Westmere 3.3GHz</td>
<td>Intel Nehalem +NVIDIA 2.5 GHz +Quadro Plex S4s</td>
</tr>
<tr>
<td>Memory</td>
<td>2GB/core</td>
<td>2GB/core</td>
<td>6GB/core (240 nodes) 18GB/core (16 nodes)</td>
</tr>
<tr>
<td>Interconnect</td>
<td>SDR IB</td>
<td>QDR IB</td>
<td>QDR IB</td>
</tr>
<tr>
<td>Disk</td>
<td>1.7PB Lustre (IB)</td>
<td>1PB Lustre (IB)</td>
<td>0.2PB Lustre (10GigE)</td>
</tr>
</tbody>
</table>
Storage Systems

High Speed Disk -- Corral

- 1 PB Data Direct Disk
- 800TB Lustre File System
- 200TB Data Collections
- InfiniBand interconnect
- Access: as /corral file system on ranger, lonestar and longhorn; ssh/scp; requires allocation

Tape Storage -- Ranch

- 10PB capacity
- 70 TB cache
- 10Gb Ethernet interconnect
- Access: scp/bbcp to ranch.tacc.utexas.edu; or rsh/ssh
6. Software
Software

Software section in User Guide

Software list available on Ranger

Software list available on XSEDE

The module utility is used to provide a consistent, uniform method to access software.
MODULE Command (Ranger-only)

- Affects $PATH, $MANPATH, $LIBPATH
- Load specific versions of libraries/executables
- Works in your batch file
- Define environment variables:
  - TACC_MKL_LIB, TACC_MKL_INC, TACC_GOTOBLAS_LIB
- Order matters! First choose compiler, then application software.
Module
This utility is used to set up your PATH and other environment variables:

% module help
% module list
% module avail
% module load pgi
% module load intel
% module swap pgi intel
% module load boost
% module unload boost
% module help <mod1>
% module spider
% module spider petsc

{lists options}
{lists loaded modules}
{lists available modules}
{add a module}
{try to load intel}
{swap two modules}
{add a module}
{remove a module}
{module-specific help}
{lists all modules}
{list all version of petsc}
More Module Notes:

- Create your own initial default setup
  
  % module purge; module load TACC
  % module load git boost petsc
  % module setdefault

- Family
  TACC supports two Families: Compilers and MPI implementations. You can only have one member of the family.

- Only one compiler, one MPI stack.
- Env. Var: TACC_FAMILY_COMPILER: intel, pgi, gcc
- Env. Var: TACC_FAMILY_MPI: mvapich, mvapich2, openmpi
- Can be used in Makefiles, and Scripts.
Two Time Commands

- Used to see how long your program runs and estimate if it’s having gross difficulties
- /usr/bin/time generally gives more information

```
login3% cd $HOME/envi/intro
login3% make
  g++ hello.c -o hello
login3% time ./hello
Hello world!
  0.000u 0.004s 0:00.01 0.0%  0+0k 0+0io 0pf+0w

login3% /usr/bin/time ./hello
Hello world!
  0.00user 0.00system 0:00.00elapsed 133%CPU (0avgtext+0avgdata
  3120maxresident)k
  0inputs+0outputs (0major+238minor)pagefaults 0swaps
```
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
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 (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 series of multiple keystroke combinations to execute commands
- “Hard to learn, easy to use”
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 ranger.tacc.utexas.edu
In the left pane, navigate to the desktop.
Drag files between panes to copy.
Again with X-Windows

- Start X-Windows server on local machine.

```
> echo $DISPLAY localhost:39.0
> emacs README&
```

```
> jobs
> kill %1
```
Login with X-Windows

- Start Exceed->Exceed on Windows Startup menu (Already started on Mac and Linux)
- ssh –X on Linux, Mac. For Windows, select in Putty Connection->SSH->X11, and check “X11 Forwarding”
- Type in username and password.
- echo $DISPLAY
- emacs README& # This runs emacs in the background.
- At the command prompt, type “jobs” to see that you have a job running in the background.
- Try Emacs for a while, then kill it with
- kill %1
8. Batch Job Submission with Sun Grid Engine (SGE)
**Batch Submission Process**

**Queue:** Job script waits for resources.

**Master:** Compute node that executes the job script, launches all MPI processes.

**Login Node**

- `ssh`

**Queue**

- `qsub job`

**Master Node**

- `mpirun -np # ./a.out`

**Compute Nodes**

- C1
- C2
- C3

**Ibrun ./a.out**
Submit a Job

1. Write a script

```bash
#!/bin/sh
echo Starting job
date
/usr/bin/time ./hello
date
echo Ending job
```

2. Add batch instructions

```bash
#!/bin/sh
#$ -N hello
#$ -cwd
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
#$ -q development
#$ -pe 1way 16
#$ -V
#$ -l h_rt=00:2:00
echo Starting job
date
/usr/bin/time ./hello
date
echo Ending job
```

3. Submit it to the scheduler

```
qsub -A 20101208HPC job.sge
```
Queue Examples

login3% qconf -sql
clean
development
large
long
normal
request
reservation
Serial
stci
sysdebug
systest
vis

login3% qconf -sq development
qname
devment
qtype
BATCH INTERACTIVE
pe_list
16way 15way 14way 12way 8way 4way 2way 1way
slots
16
	/tmp

tmpdir
...

Slots = number of cores, 16 per node
pe = wayness, how many cores per node
Job is killed if over time limit.

Why 15way?
Submit a Job Example

cd $HOME/envi/intro
ls -la
cat Makefile # Review the makefile
make # Compile hello.c
ls –la # Take a look at what compiled
./hello # Run compiled program
less job.sge # View the script (q)
qsub –A TG-TRA120006 job.sge # Submit the job
showq –u # Look at your job(s)
States

- Unscheduled – Likely not good
- DepWait – You can ask that one job run after another finishes.
- w(aiting) – Queued, waiting for resources to run.
- r(unning) – As far as SGE is concerned, it’s going.
- h(old)
- s(uspended)
- E(rrror)
- d(eletion)
SGE: Basic MPI Job Script

```bash
#!/bin/bash
#$ -pe 16way 32       # Wayness and total core number
#$ -N hello           # Job name
#$ -o $JOB_ID.out     # stdout file name (%J = jobID)
#$ -e $JOB_ID.err     # stderr file name
#$ -q normal          # Submission queue
#$ -A A-ccsc          # Your Project Name
#$ -l h_rt=00:15:00    # Max Run Time (15 minutes)
ibrun ./hello         # Execution command
```
Parallel Environment

- Each node has 16 cores and is used by one person at a time
- `#$ -pe 1way 16` Run one task on a node with 16 cores
- `#$ -q serial`
- `./hello`
- `#$ -pe 8way 64` Run 8 tasks/node on 4 nodes
- `#$ -q normal`
- `export MY_NSLOTS=31` Launch 31 tasks
- `lbrun ./a.out` Run with mpi wrapper
SGE: Memory Limits

- Default parallel job submission allocates all 16 compute cores per node.
- If you need more memory per MPI task, you can request fewer cores per node by using one of the ‘Nway’ environments below.
- Even if you only launch 1 task/node, you will still be charged for all 16!

<table>
<thead>
<tr>
<th>Parallel environment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16way</td>
<td>16 tasks/node, 1.9GB/task</td>
</tr>
<tr>
<td>8way</td>
<td>8 tasks/node, 3.8GB/task</td>
</tr>
<tr>
<td>4way</td>
<td>4 tasks/node, 7.6GB/task</td>
</tr>
<tr>
<td>2way</td>
<td>2 tasks/node, 15.2 GB/task</td>
</tr>
<tr>
<td>1way</td>
<td>1 task/node, 30.4 GB/task</td>
</tr>
</tbody>
</table>
SGE Batch

% cd $HOME/envi/batch
% ls -la
% mpif90 -O3 mpihello.f90 -o mpihello

OR
% mpicc -O3 mpihello.c -o mpihello
% cat job (edit account?)
% qsub job
% watch showq -u -l (Ctrl-C to quit watching)
% vi job (add “sleep 60”)
% qsub job (observe the returned jobid)
% qdel jobid
10. Miscellaneous
The precision program computes 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.f
% module load intel
% ifort -FR precision.f
(or)
% ifort precision.f90
% ./a.out

( The ifc compiler regards “.f” files as F77 fixed format programs. The –FR option specifies that the file is free format.)
% 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
Questions?

• CAC
  help@cac.cornell.edu
• XSEDE
  – portal.xsede.org -> Help
  – portal.xsede.org -> My XSEDE -> Tickets
  – portal.xsede.org -> Documentation -> Knowledge Base