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. Ranger Overview
4. Accessing Ranger
5. Login Environment
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

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

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

• Education – for classes and training sessions
  – Up to 200,000 core-hrs, for 1 year
  – Submit Abstract, Awarded/2 weeks

https://portal.xsede.org/allocations-overview
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
3. Ranger Overview
The Generic Environment

Scheduler

Head Node

SSH to Head Node

www.cac.cornell.edu
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>

- `%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`
- `%cds` 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
4. Accessing Ranger
SSH Clients

- Windows: Putty
- Linux or Mac: built-in as “ssh”

Login now to ranger.tacc.utexas.edu:

% ssh username@ranger.tacc.utexas.edu  
-or-
Start putty
use Host Name: ranger.tacc.utexas.edu

- **Do not** overwrite ~/.ssh/authorized_keys
  - Feel free to add to it if you know what it’s for
Accessing XSEDE Resources

• Several methods are possible:
  – Direct login access
  – Single Sign On (SSO) through portal
  – SSO between resources
  – Through Science Gateways

• Your choice of method may vary with:
  – How many resources you use
  – How much you want to automate file transfers, job submission, etc.
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.
• Stand-alone client: http://grid.ncsa.uiuc.edu/gsi-sshterm/
• 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.
SSO thru user portal

• Make sure you are logged into the XSEDE User Portal

• Go to ‘My XSEDE’ tab

• Go to the ‘Accounts’ link

• Resources you have access to will be indicated by a ‘login’ link

• Click on the ‘login’ link of the resource you would like to access.

Login now to ranger.tacc.utexas.edu using the XSEDE portal
SSO Thru User Portal

- A Java Applet will talk... you may be asked permission to allow it to run.
- After the applet starts, a blank terminal window will appear in your web browser.
- The window will fill with text indicating that you have been successfully logged into the resource of your choice.
- You can now work on this machine, and connect to other machines from this terminal, using the command `gsissh machine-name`
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.
- SSH to ranger. Start it. Connect with VNC Client. Kill it.
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
## Account Info

Note your account number at bottom of splash screen.

<table>
<thead>
<tr>
<th>Project balances for user train100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>TG-TRA120006</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk quotas for user train100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
</tr>
<tr>
<td>/share</td>
</tr>
<tr>
<td>/work</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

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

• `~username` is the home directory of that user
• For example, to create a tar: `tar cvf myfiles.tar dir1 dir2 README`

  Change directory to the envi directory:
  % `cd envi`
  List the lab files:
  % `ls -la`
% echo $SHELL
% chsh -l
% man chsh
% 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)
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
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
Try MODULE

module load fftw2
module del fftw2

# note the response
# so delete pgi

Order matters! Unload MPI, then choose a compiler, then load the MPI version.
Module
This utility is used to set up your PATH and other environment variables:

% module help  {lists options}
% module list   {lists loaded modules}
% module avail  {lists available modules}
% module load pgi {add a module}
% module load intel  {try to load intel}
% module swap pgi intel  {swap two modules}
% module load boost  {add a module}
% module unload boost  {remove a module}
% module help <mod1>  {module-specific help}
% module spider  {lists all modules}
% module spider petsc  {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
make: `hello' is up to date.
login3% time ./hello
Hello world!
0.000u 0.00s 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
To Edit A File in VI (short for “visual”)

- “vi filename” will open it or create it if it doesn’t exist.
- Command mode and Insert mode. You start in command mode.
- Command mode. Cursors work here, too.
  - :w  Writes a file to disk.
  - :q  Quits
  - :q! Quits even if there are changes to a file
  - i  Takes you to insert mode
- Insert Mode
  - Cursors, typing characters, and deleting work here.
  - Escape key takes you to command mode.
- Ctrl-c will get you nowhere.
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”
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 backgrounded job.
- 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.

ssh

Login Node

qsub job

Queue

Master Node

Compute Nodes

C1
C2
C3

mpirun -np # ./a.out

ibrun ./a.out

Internet
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

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

<table>
<thead>
<tr>
<th>login3% qconf -sql clean development large long normal request reservation serial sysdebug systest vis</th>
</tr>
</thead>
</table>

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

```
login3% qconf -sq development
qname development
qtype BATCH INTERACTIVE
pe_list 16way 15way 14way 12way 8way 4way 2way 1way
slots 16
tmpdir /tmp
shell /bin/csh
prolog /share/sge/default/pe_scripts/prologWrapper
epilog /share/sge/default/pe_scripts/tacc_epilog_new.sh
shell_start_mode unix_behavior
s_rt 07:58:00
h_rt 08:00:00
```

Why 15way?

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

- `qconf -sql`  # List available queues
- `qconf -sq <queue name>`  # Soft and hard wall clock limits
- `cat /share/sge6.2/default/tacc/sge_esub_control`  # Queue core limit
- `showq`
- `showq -u`
- `qdel or qdel -f`  # Delete job
Submit a Job Example

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
qsub –A 20101208HPC job.sge  # Submit the job
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(upended)
- E(rror)
- d(eletion)
SGE: Basic MPI Job Script

#!/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
% 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
Precision

• Look over the precision.f program in the precision directory.
  \% cd $HOME/lab1/precision

• The program computes prints $\sin(\pi)$. The $\pi$ constant uses “E” (double precision) format in one case and “D” (single) in the other.

\% 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.)
### Makefiles

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>% cd $HOME/envi/using_makefiles</code></td>
<td>Change directory to the Makefiles directory.</td>
</tr>
<tr>
<td><code>% cat Makefile</code></td>
<td>Read over the Makefile</td>
</tr>
<tr>
<td><code>% make</code></td>
<td>Compile the program, generate a.out</td>
</tr>
<tr>
<td><code>% make</code></td>
<td>Reports “up to date”, i.e. not recompiled</td>
</tr>
<tr>
<td><code>% touch suba.f</code></td>
<td>Simulate changing a file</td>
</tr>
<tr>
<td><code>% make</code></td>
<td>suba.f (and only suba.f) is recompiled</td>
</tr>
</tbody>
</table>
References

• TACC
  – Ranger (http://services.tacc.utexas.edu/index.php/ranger-user-guide)
  – Spur (http://services.tacc.utexas.edu/index.php/spur-user-guide)

• CAC

• Tutorials
  – Beginners Unix (http://info.ee.surrey.ac.uk/Teaching/Unix/)
Questions?

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