Ranger Environment Documentation

Drew Dolgert

May 18, 2010

Contents

Contents:

1 Introduction

This document accompanies a talk introducing the working environment on the Ranger cluster at the Texas Advanced Computing Center. The exercises inside are enough for someone already familiar with the Linux command line to submit jobs to the Ranger cluster. There is also some reference information, mostly a convenient copy of the Ranger user guide.

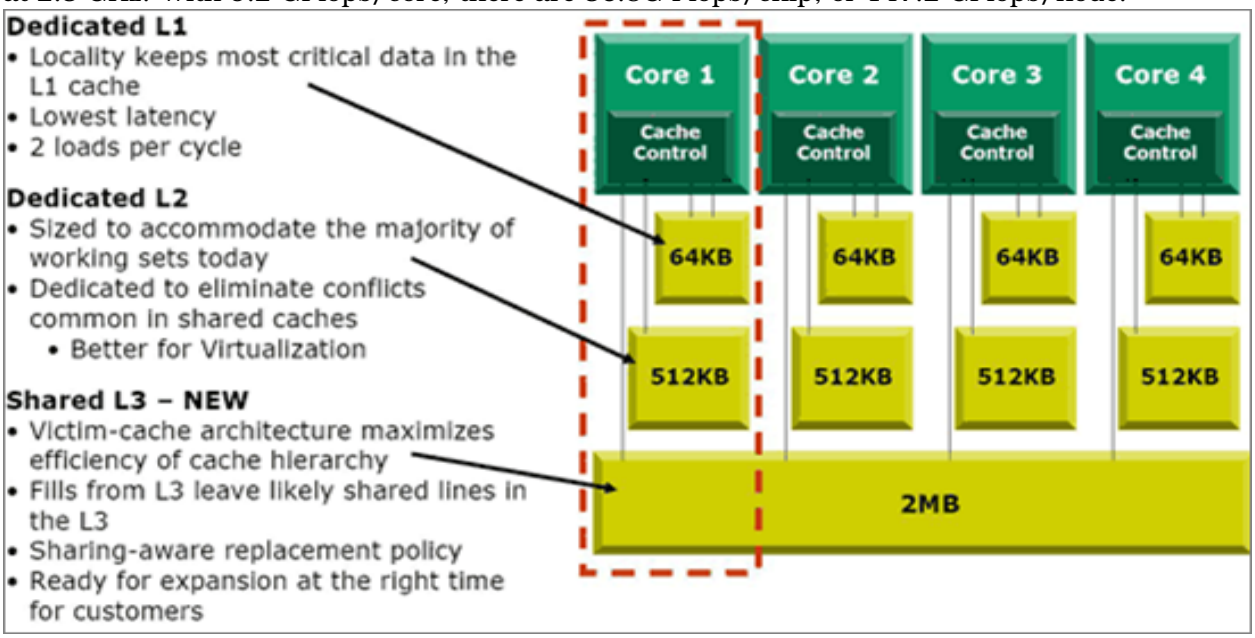
Learning to use the Linux command line for the first time is significantly more difficult than learning to submit simple jobs to Ranger. Finding a separate guide on C Shell will greatly increase your comfort.

References:

- Ranger User Guide - http://services.tacc.utexas.edu/index.php/ranger-user-guide
- Beginning Unix - http://info.ee.surrey.ac.uk/Teaching/Unix

1.1 Ranger Hardware

Each node on Ranger has four processors of four cores each. It's a 64-bit AMD Barcelona chipset at 2.3 GHz. With 9.2 GFlops/core, there are 36.8G Flops/chip, or 147.2 GFlops/node.



There are 3,936 nodes, which makes 62976 cores. Peak performance is 579 TFLOPS.

Check out the Ranger User Guide for more.

2 Connect

2.1 Login Nodes

You can connect to any of Ranger's login nodes using Secure Shell, or SSH.

Ranger Login Address	Usage
`ranger.tacc.utexas.edu`	Puts you on one of the main login nodes.
`tg-login.ranger.teragrid.org`	The same login nodes, through a TeraGrid alias.
`login4.ranger.tacc.utexas.edu`	Chooses a specific login node.

The login nodes share a filesystem, so there is rarely a reason to return to the same node unless you are using a VNC connection.

Each node is a 16-core Sun Linux system. You can edit, compile, and manage data on these systems, but they aren't a good place to run any code because they are shared among many users. If you want to run code quickly, use the development queue.

2.2 Exercise: Use SSH to Connect

Goal: Connect to the login computer where we will work.

Skip to the section for either Mac, Linux, or Windows, depending on what kind of computer you are using.

2.2.1 SSH from Linux or Mac

Secure Shell (SSH) is already installed on almost all Linux and Mac OS X systems. Telnet is no longer used because SSH is more secure.

Open a terminal window by right-clicking on the desktop. Type the part that follows the dollar sign:

```
$ ssh username@ranger.tacc.utexas.edu
```

The Ranger login node will respond with its status and the status of your account.

2.2.2 SSH from Windows

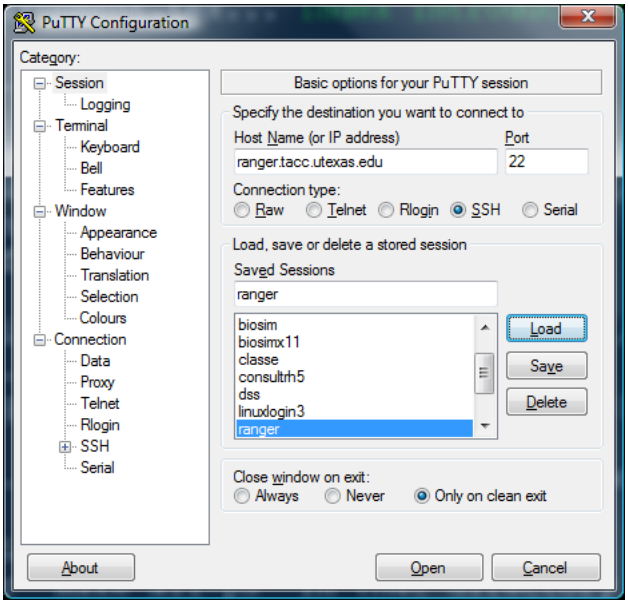
If you are working on a lab computer, skip the part about installation. Putty is installed on lab computers.

Installation of Secure Shell

Telnet is disabled for security reasons, but Secure Shell (ssh) clients work nicely as long as they support the SSH2 protocol. A popular client for Windows is the free Putty client, http://www.chiark.greenend.org.uk/~sgtatham/putty/. The simplest installation is to download the Windows installer, called putty-0.60-installer.exe, and run it to install putty into your Start menu.

Starting a Client Session

1. Start Putty from the Start Menu.
2. Type a host name, `ranger.tacc.utexas.edu`, where you see the red circle. and hit Enter.



The Ranger login node will respond with its status and the status of your account.

2.3 Starting: Read Examples of Sessions

Goal: Familiarize with general command line use.

The Linux command line may look like an unfair game because there is no indication what to type. The great news is that there are just a few commands that account for almost everything you might want to do. In this exercise, **just read** transcripts to see how work is done.

Following are transcripts of Csh sessions at a command line. The hash, #, is a comment character. The part you would type, but which you are **not going to type right now,** is after the dollar sign. Any responses from Csh follow.

Create a New Workspace

```
# Comments like this start with hash marks.
# First, where am I?
$ pwd
/home/ajd27
# Make a directory called intro.
$ mkdir intro
# Check that the directories were made
$ ls
bin course intro
# Move into intro.
$ cd intro
# Run a command from another user's directory
$ ~ajd27/bin/lu.S
NAS Parallel Benchmarks (NPB3.2-SER) - LU Benchmark
Size:  12x 12x 12
```

```
Iterations:  50
...
# Copy that command to your own bin directory
$ cp ~ajd27/bin/lu.S ~/bin
# See that it worked by listing the bin subdirectory of the parent (..) directory.
$ ls ../bin
farm lu.A lu.C     lu.omp.B lu.omp.S lu.S
htop lu.B lu.omp.A lu.omp.C lu.omp.W lu.W
# Save the command results to a file.
~/bin/lu.S > s0.txt
# Check that you got the file and see its size in bytes.
$ ls -la
total 3428
drwxr-xr-x  5 ajd27 Domain Users    4096 Apr 20 17:05 .
drwxrwxrwx 35 ajd27 Domain Users    4096 Apr 14 15:17 ..
-rw-r--r--  1 ajd27 Domain Users    1932 Apr 20 17:05 s0.txt
# Read the file carefully, using a text reader called less.
# In less: 'b' for back. 'q' to quit. '/' lets you enter a search string.
less s0.txt
```

2.4 Further Exercise: Using X-Windows to Connect

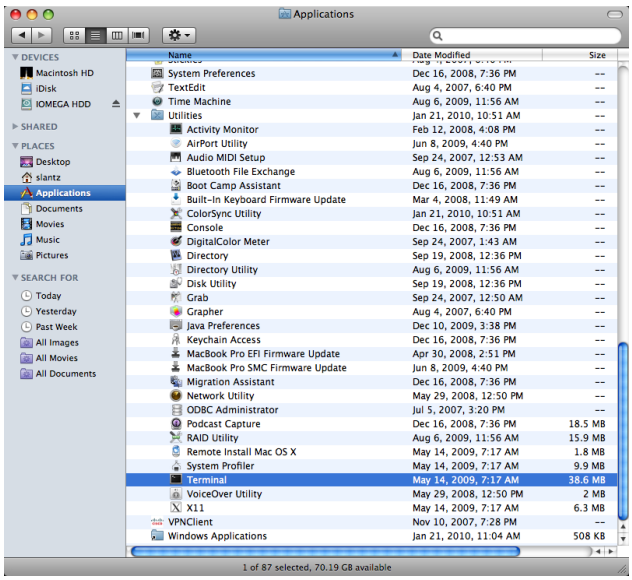
Goal: Use the mouse and menus with remote applications.

X-Windows can be sluggish over long distances. This section shows how to start X-Windows, and you can make it faster later by increasing the compression level on your SSH connection.

Skip to the section for either Mac, Linux, or Windows, depending on what kind of computer you are using.

2.4.1 Opening a Connection From Mac

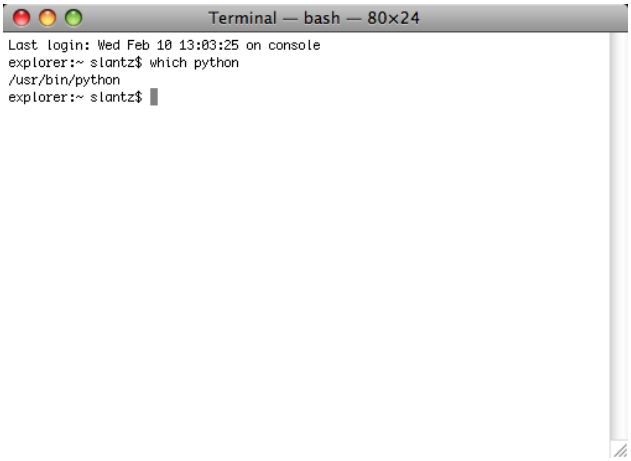
If you are on a Mac, the big question is whether you have X11 installed. Navigate in the Finder to the Applications folder and Utilities sub-folder. Do you see an application here called X11, as shown in this picture?



If not, then you need your installation disc. X11 is in the Applications package, and the installer is in the Optional Installs folder.

If you do have X11 installed, then first open a terminal window as shown below. Then follow directions in *Opening a Connection From Linux*.

Navigate in the Finder to the Applications folder and Utilities sub-folder. Double-click on the Terminal application to see a Bash command-line.



Continue with the next section, *Opening a Connection From Linux*.

2.4.2 Opening a Connection From Linux

In order to use X-Windows applications on Ranger, you have to turn on *X11 Forwarding* when you make an SSH connection. This is as simple as adding `-X` to the SSH command:

```
$ ssh -X username@ranger.tacc.utexas.edu
```

Now that you are at a prompt at the login node, open an X-Windows application. The dollar sign is called the *prompt.* Type the part that follows it:

```
$ xclock
```

You should see it appear in the corner of your screen if X-Windows is working.

If ever you see a message that "trusted X11 forwarding has failed," connect again with `-Y` instead of `-X`.

2.4.3 Opening a Connection From Windows

If you are working on a lab computer, skip the part about installation. Putty and Exceed (not Xming) are installed on lab computers.

Installation of X-Windows

You will need to install an X-Windows server on your Windows machine. Common choices include Xming (free), Cygwin/X (free), XFree86 (free), and Exceed.

Xming, available at http://www.straightrunning.com/XmingNotes/, is open source, with an available shareware version that has improved graphics. There are two pieces to download.

Links and release status

Website Releases	Version	State/Notes	Released	MD5 signatures	Size MB
Xming	7.5.0.17	Website Release	7 Mar 2010	MD5 signatures	3.77
Xming-portablePuTTY	7.5.0.18	Website Release	7 Mar 2010	MD5 signatures	1.62
See Donations for how to obtain a Donor Password.					

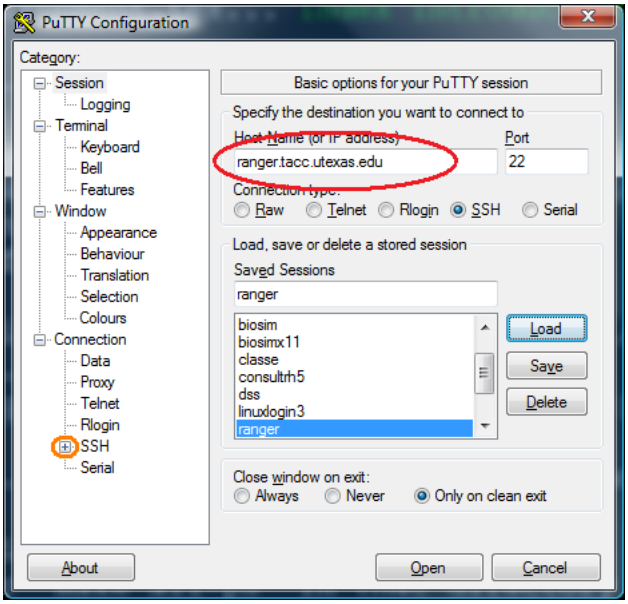
Public Domain Releases	Version	State/Notes	Released	MD5 signature	Size MB
Xming-fonts	7.5.0.11	Public Domain	20 Oct 2009	01843baadd75105b962baafa2d08942e	30.6
Xming Xming-mesa	6.9.0.31	Public Domain	4 May 2007	4cd12b9bec0ae19b95584650bbaf534a e580debbf6110cfc4d8fcd20beb541c1	2.10 2.50

Figure 1: Download and install both Xming-mesa and Xming-fonts.

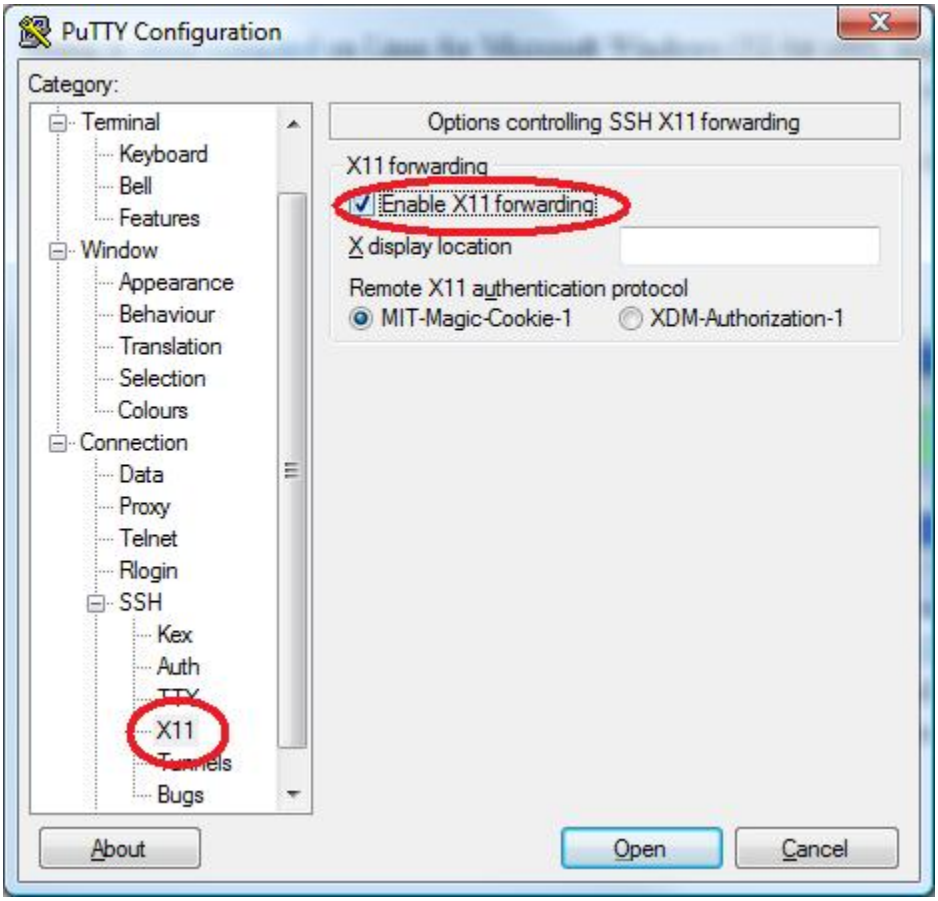
- Xming-mesa (public domain release). There are two links together, one for Xming, one for Xming-mesa. Either will work for this tutorial, but Xming-mesa has some newer features that might come in handy some time.
- Xming-fonts (public domain release) If you purchase the website release of Xming, remember to install the Xming-fonts, as well.

Starting a Client Session

1. Before you connect to the login node, start the X-Windows server on your machine. In the Start menu, either double-click the Xming icon. The application will briefly appear, then disappear into the background, waiting for windows to show.
2. Start Putty.
3. Type a host name, `ranger.tacc.utexas.edu`, where you see the red circle.



4. Open the SSH menu, in orange on the left of the picture above. Choose X11 settings, shown on the left below, to find the button to "Enable X11 forwarding."



1. Now use the menu on the left to navigate back to the Session screen. Enter "ranger" under Saved Sessions, and click the Save button. You can use this short cut the next time you connect.

2. Click Open, and it will connect to the head node.

Now that you are at a prompt at the login node, open an X-Windows application. The dollar sign is called the *prompt.* Type the part that follows it:

```
$ xclock
```

You should see it appear in the corner of your screen if X-Windows is working.

2.5 Exercise: Connect with VNC

VNC is a protocol that gives you a shows you a complete desktop, as though Ranger's login node were your a local computer. The great advantage of VNC is that it can be fast enough to do remote visualization because it transfers only mouse information and bitmap images of the desktop.

Like SSH and X-Windows, a VNC session requires a client and a server. In this case, you have to start a server manually on the Ranger login node. That server will define a *screen* to which you can connect the VNC client, which is installed on your local computer. Each VNC server listens on a TCP/IP port designated by its screen number, so screen 1 listens at port 5901, screen 27 at port 5927.

There are many good VNC clients. On Windows and the Mac, try RealVNC or TightVNC. On Linux, try `vinagre` or `vncviewer`.

Login to Ranger. First set your vnc password, which you will need when your client connects to the server. Then start the server:

```
$ vncpasswd
$ vncserver
New 'login3.ranger.tacc.utexas.edu:1 (train200)' desktop is
login3.ranger.tacc.utexas.edu:1
Starting applications specified in /share/home/0002/train200/.vnc/xstartup
Log file is /share/home/0002/train200/.vnc/login3.ranger.tacc.utexas.edu:1.log
```

The message above indicates that this VNC session is on screen 1 of `login3`. We can connect to that from our local VNC client using the address `login3.ranger.tacc.utexas.edu:1`.

When you are done with the session, be sure to kill the VNC server with:

```
% vncserver -kill :1
Killing Xvnc process ID 11406
```

Note that `:1` represents screen 1.

It is possible to tunnel VNC through Secure Shell for a safer connection. If you are using TightVNC over long distances, you can greatly speed the connection by increasing compression. Another trick is to offer your VNC server address and password to a colleague in order to share a display.

2.6 Further Exercise: Choose Your Shell

The default shell is `/bin/csh`. If you prefer another shell, now is a good time to change it. List available shells with:

```
$ chsh -l
```

and then change your shell, for instance to tcsh, with:

```
$ chsh /bin/tcsh
```

Every shell reads configuration files on login. The two main types of shells, the Csh family and Bash family, are configured on Ranger so that your personal configurations belong in `$HOME/.cshrc_user` or `$HOME/.login_user`, respectively.

2.7 Advanced: Make Login Faster

There are two things you can do to make it easier to login to ranger. One is to make a shortcut, or alias, to the login node. The other is to create SSH keys that will allow you to login to the node without typing a password.

2.7.1 Making Shortcuts on Windows

Goal: Open SSH terminal window with a single click.

If you are working on your laptop, make a shortcut or alias to login to `ranger.tacc.utexas.edu`.

For Windows and Putty, first create a saved session. When you first open Putty, it gives you a list of "saved sessions." Enter a name in the box above that list, and click the save button. Then make a shortcut with the -load option, where *ranger* is the whatever you called your session:

```
C:\util\putty.exe -load ranger
```

2.7.2 Setup SSH Keys for No Password Login

Goal: Make login not require typing a password.

This can take a few minutes if you have not done it before.

Start a key agent on your local computer which then logs in for you. You have to type your pass phrase once when you start your local computer. There is an excellent tutorial already at http://www.mtu.net/~engstrom/ssh-agent.php which covers Linux, the Mac, and Windows.

3 Using Module

Goal: Compile a parallel application on Ranger.

3.1 About the Module Command

The `module` command changes variables in your shell in order to choose which applications you use. For instance, Ranger has about twenty different versions of MPI because there are MPI builds for two compilers, three versions of MPI, and for past and development builds.

To load or unload a module, the `module` command changes environment variables the shell uses to choose what it runs.

- **$PATH** - list of directories where executables are found.
- **$MANPATH** - list of locations to find help files.

- **$LD_LIBRARY_PATH** - list of directories where applications can find shared libraries.

Each loaded module may define a new set of environment variables. For instance, loading the Intel Math Kernel Libaries (MKL) defines variables you can use in a makefile.

- **$TACC_MKL_DIR** - The base directory of the MKL installation.
- **$TACC_MKL_DOC** - Location of documentation.
- **$TACC_MKL_LIB** - Library directory.
- **$TACC_MKL_INC** - Include directory for headers.

The `module` command tells you about possible conflicts. For instance, you will have the Intel compilers loaded when you start. If you add the PGI compilers, you will see a message that Intel is already loaded. Heed that advice, and delete the Intel compilers when you add the PGI compiler.

The basic `module` commands are

- **module avail** - List available modules
- **module avail string** - Search for the module "string".
- **module list** - Show currently-loaded modules.
- **module add** - Add a module to the environment.
- **module del** - Remove a module from the current environment.
- **module help string** - Get help on a module named "string".
- **module purge; module load TACC** - Return to original state.

Some versions of `module` require you to load them in a particular order. The version installed (and developed) at TACC does not.

3.2 Discussion Exercise: Learn Modules

Goal: Find out what is installed.

Scan the list of modules in the next section for ones you recognize and think about the following questions, then discuss with the person beside you.

1. How many do you recognize as free software and which cost money?
2. Which scientific disciplines are well-represented?
3. How do you think applications and libraries are chosen?

If you need for work an application not already installed, you will be able to install it in your home directory.

3.3 Modules List

This is a partial list of modules. Some entries that appear more than once on Ranger are here only in one section.

- /opt/apps/pgi7_2/mvapich1_1_0_1/modulefiles
  - amber - Molecular dynamics of proteins and nucleic acids
  - arpack - Fortran subroutines to solve large scale eigenvalue problems.

  - charm++ - Object-oriented language in C++ with adaptive MPI (NAMD, cosmology, QCD).
  - espresso - electronic-structure calculations and materials modeling at the nanoscale.
  - fftw2 - A fast, free C FFT library; includes real-complex, multidimensional, and parallel transforms.
  - fftw3 - A fast, free C FFT library; includes real-complex, multidimensional, and parallel transforms.
  - gamess - ab inition quantum chemistry.
  - gromacs - molecular dynamics for biomolecular systems.
  - gulp - General utility lattice program for 3D periodic solids, gas phase clusters, and more.
  - hypre - library for solving large, sparse linear systems of equations
  - ipm - integrated performance monitoring.
  - kojak - performance-analysis tool for parallel applications supporting the programming models MPI, OpenMP, SHMEM, and combinations thereof
  - mpiP - lightweight profiling library for MPI applications.
  - pdtoolkit - Program database toolkit for analyzing source code.
  - petsc - data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
  - phdf5 - Parallel data model, library, and file format.
  - plapack - Parallel linear algebra matrix manipulations.
  - pmetis - parallel graph partitioning.
  - pnetcdf - parallel interface, library, and format for data.
  - scalapack - linear algebra routines using MPI.
  - scalasca - open-source analysis of parallel applications.
  - slepc - Scalable library for eigenvalue problems.
  - sprng - scalable parallel psuedo random number generation.
  - tao - toolkit for advanced optimization.
  - trilinos - algorithms for large-scale, complex multi-physics engineering and scientific problems.
- /opt/apps/pgi7_2/modulefiles
  - acml - AMD Core Math Library, lapack.
  - autodoc - docking tools for small molecule binding to known 3D structure.
  - glpk - GNU linear programming kit.
  - gotoblas - Hand-tuned Basic linear algebra subroutines.
  - hdf5 - data model, library, and file format.
  - hecura - MPI-2 implementation for Infiniband
  - mvapich - MPI-1 implementation for InfiniBand

  - nco - Programs for manipulating and analyzing NetCDF files.
  - netcdf - interface, library, and format for data.
  - openmpi - MPI-2 implementation for Infiniband
- /opt/apps/teragrid/modulefiles
  - ctssv4 - Coordinated TeraGrid software and services.
  - globus-4.0 - Toolkit for building grids.
  - teragrid - presumably TeraGrid tools.
  - apache-ant - A Java build tool.
  - condor-g - Grid version of Condor scheduler.
  - condor - Methods for high throughput computing.
  - gridshell - login shell for grid computing.
  - gsissh - SSH for TeraGrid.
  - gx-map - Utilities to maintain grid security directory.
  - koomie - Internal for testing software builds.
  - mycluster - Virtual clusters on demand.
  - pacman - Pacman lets you define, install, configure and setup software.
  - srb-client - Storage request broker client.
  - tg-policy - Displays TACC teragrid policy.
  - tginfo - TeraGrid Information Service Discovery Client
  - tgproxy - Helps you work with TeraGrid credentials.
  - tgresid - TeraGrid resource identification
  - tgusage - query TeraGrid allocations.
  - uberftp - Grid-enabled FTP client.
- /opt/apps/modulefiles
  - R - functional language for statistical computing.
  - autotools - configuration of software.
  - beta - unknown
  - binutils-amd - Tools to work with binaries.
  - cmake - cross-platform, open-source build system.
  - ddt - large-scale parallel debugger
  - gcc - GNU compiler collection.
  - git - version control system
  - gmake - GNU make program.
  - gmp - Gnu Multiple Precision Library
  - gnuplot - 2D and 3D plotting.
  - gsl - GNU scientific library.

  - gzip - Open-source compression.
  - intel - Intel compilers
  - irods - Integrated Rule-oriented Data System.
  - launcher - TACC Parametric Job Launcher
  - lua - Programming language.
  - mkl - Intel Math Kernel Library.
  - mpfr - C library for multiple-precision floating point.
  - mysql - Open-source database.
  - ncl_ncarg - NCL/NCAR graphics library
  - numpy - Tools for numerical computing on Python.
  - papi - Interface for to use hardware performance counters.
  - perfexpert - Peformance evaluation tool developed at TACC.
  - pgi - Portland Group compilers for C and Fortran.
  - plplot - Library to create scientific plots.
  - postgres - Open Source database.
  - python - Scripting language.
  - star-ccm- CAD preparation, meshing, and model setup.
  - subversion - version control system.
  - sun - Loads the Sun compiler environment.
  - tar - Archive creation and manipulation.
  - vis - Makes all of the visualization modules visible.
  - zlib - compression library.
- /opt/modulefiles
  - java - programming language.
- /opt/apps/vis/modulefiles
  - VTSSV3 - TeraGrid visualization tools?
  - amira - Visualization stressing life sciences.
  - blender - Open Source 3D creation suite.
  - chromium - scalable rendering for clusters of workstations.
  - cuda-sdk - Software development kit for GPU programming.
  - ensight - The “extreme” visualization post-processor.
  - glew - OpenGL extension wrangler library.
  - glui - OpenGL User Interface library.
  - glut - OpenGL utility kit.
  - gmt - 60 unix tools to manuplate (x,y) and (x,y,z) data sets.
  - idl - Visualization programming language.

- imagemagick - Convert, edit, compose images.
- mesa - OpenGL offscreen rendering.
- mplayer - Movie player.
- netpbm - command-line utilities for image manipulation.
- paraview - Open source scientific visualization.
- qt - Application windowing toolkit.
- sdl - Simple DirectMedia Layer for graphics and audio.
- silo - file format used by VisIt.
- vapor - Visualization package for ocean, atmosphere, solar.
- visit - Open source, 3D visualization tool.
- vtk - Visualization library, open-source, scriptable from C++, perl, python, java, tcl.

3.4 Exercise: Compilation with Modules

Goal: Use modules to choose tools.

Imagine you brought your code to this computer. You know your code needs the following kinds of libraries.

- C Compiler - Three compilers are installed on this system, two of which are listed by the module command.
- BLAS Library - Basic Linear Algebra Subroutines, which our program will use to invert large matrices.
- MPI - The message passing interface, which our program will use to run on many cores at the same time.

We have, in this case, the HPL benchmarks, which are sometimes used to test supercomputers.

1. Use `module` to list what is currently loaded. (Section *About the Module Command* might be helpful here.)
2. Add `intel`, for the Intel compilers, and `mkl`, which are the Intel Math Kernel Libraries. It may hint you need to unload another module. Take the hint.
3. Copy code with `cd && tar zxf ~train400/envi.tgz`.
4. Make that code with `cd hpl-2.0; make arch=icc_mkl`. This may take a few minutes. If there are problems, they would indicate you don't have the modules loaded.

3.5 Further Exercise: Module Dependencies

First, start fresh by purging all modules. (You can always get the original set back by loading the TACC package.) Then take a look at what is loaded:

```
$ module purge
$ module avail
```

This list includes the `intel` and `pgi` compilers. Load the `intel` compilers and look to see how your choice of modules has changed:

```
$ module add intel
$ module avail
```

Do you see that some of the section headers now refer to `intel`? These options were not present until you loaded the compiler because these libraries and applications depend on which compiler you load. The last major choice you would make is which version of MPI you will use. Load `mvapich`:

```
$ module add mvapich
$ module avail
```

You will see that modules are listed in sections by `module avail`:

```
/opt/apps/intel10_1/mvapich1_1_0_1/modulefiles
/opt/apps/intel10_1/modulefiles
/opt/apps/teragrid/modulefiles
/opt/apps/modulefiles
/opt/modulefiles
```

The section headings are the names of files in /opt subdirectories where the applications are installed. The first two sections are interesting because they involve your choices of compiler and MPI library. See what happens if you switch to a different compiler:

```
$ module del intel
$ module add pgi
```

Now look at module avail again. The sections have changed:

```
/opt/apps/pgi7_2/mvapich1_1_0_1/modulefiles
/opt/apps/pgi7_2/modulefiles
/opt/apps/teragrid/modulefiles
/opt/apps/modulefiles
/opt/modulefiles
```

The good news is that when you switch from the Intel compilers to the PGI compilers the MPI module, which depends on them, is automatically reloaded for you. You can see this by checking the MPICH_HOME variable:

```
login4% env | grep -i MPI
MPICH_HOME=/opt/apps/intel10_1/mvapich/1.0.1
login4% module del intel
login4% module add pgi
Due to MODULEPATH changes the follow modules have been reloaded:
  1) mvapich
login4% env | grep -i MPI
MPICH_HOME=/opt/apps/pgi7_2/mvapich/1.0.1
```

Most applications that depend on the compiler or MPI version are installed for all versions of the compiler and MPI. A few are not, which can make finding them a challenge. For instance, if you want to run `lammps` or `nwchem`, the `module avail` command will not find it, even with its search function, unless you have already loaded the Intel compiler and MVAPICH. You have to ensure `intel` and `mvapich` are loaded, then use `module avail lammps` to find it. If we started with `pgi` loaded, we see:

```
login4% module avail lammps
login4% module swap pgi intel
Due to MODULEPATH changes the follow modules have been reloaded:
  1) mvapich
login4% module avail lammps
---------------- /opt/apps/intel10_1/mvapich1_1_0_1/modulefiles ----------------
lammps/25Aug09 (default)
```

4 Running in Batch

4.1 Text Editors

There are several text editors available on Ranger.

- `nano` - Small and quick. Use Ctrl-o to save, Ctrl-x to quit.
- `vi` - Quick, powerful. You can get by in this with a cheat sheet below.
- `emacs` - Less quick, also powerful.

4.1.1 VI Cheat Sheet

Vi has two modes, *insertion mode* for editing and *command mode* for saving. The editor starts in command mode.

In command mode,

- `:w` writes a file to disk
- `:q` quits
- `:q!` quits even if you didn't save changes
- `i` puts you in insertion mode.

In insertion mode,

- Cursors and typing work as expected.
- Escape key returns to command mode.

Ctrl-c will not quit this program.

For more, ask the Google about Vi cheat sheets.

4.2 Constructing a Batch File

Ranger uses the Sun Grid Engine scheduler. Submit jobs with qsub:

```
$ qsub batch.sh
```

You can put arguments to qsub on the command line, but they usually go in the batch files as comments at the start of the file where the scheduler searches for all lines that begin with `#$`.

Argument	Description
#$ -N	The job name, available to the script as $JOB_NAME
#$ -cwd	Start the job in the same current working directory.
#$ -o	The name of the output file.
#$ -j y	Join the stdout and stderr streams in the output file.
#$ -A	Your account name, listed at login.
#$ -q	Which queue to use, found in the Ranger User Guide.
#$ -pe	The wayness of the job, as in 16way, and core count.
#$ -V	Set job variables as though it were a child of this shell.
#$ -l	Specify resources, such as h_rt=00:10:00 for run time.

For example, a job to run on 64 cores, which is 4 nodes, for 10 minutes might look like:

```
#$ -N ht3d-hyb
#$ -cwd
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
#$ -A 20100519HPC
#$ -q development
#$ -pe 16way 64
#$ -V
#$ -l h_rt=00:10:00
set -x
ibrun $HOME/bin/xhpl
```

The `ibrun` command at the bottom starts a job under MPI using settings from the `-pe` line. The second argument on that line is the number of cores for your job. Given sixteen cores per node on all Ranger nodes, the number of nodes is the number of cores divided by sixteen.

Wayness tells ibrun how many processes to start on each node you have requested. You might request `1way` if you have a distributed, multi-threaded code. You might try `15way` or `14way` if you want to leave a free core to handle intensive I/O on a node.

The scheduler stores all stdout and stderr from your batch script when it executes on the head node of the job. Those files are stored, by default, in the directory where you executed `qsub`, unless you specify otherwise with the `-o` option.

4.3 Queues

The first place to learn about queues is the Ranger User Guide section called "Ranger Queue Structure." I'll recopy that information here for convenience.

Queue Name	Max Runtime	Max Procs	Purpose
normal	24 hours	4096	The standard place to submit.
long	48 hours	1024	Longer, smaller jobs.
large	24 hours	16384	Very large jobs.
development	2 hours	256	Quick test jobs.
serial	12 hours	16	Jobs which use only one node.
request	–	–	Special requests.
vis	24 hours	32	For visualization on Spur cluster.

4.4 Charges

When you log on to Ranger, the splash screen shows what accounts you have and how many service units (SUs) you have in your accounts. You can see that information at any time with by

executing:

```
$ /usr/local/etc/taccinfo
```

or by going to the TeraGrid User Portal.

A service unit, or SU, is defined as one core-hour on a TeraGrid Phase-1 DTF cluster on TeraGrid's Getting Started. Ranger charges one SU per core-hour on all of its queues.

4.5 Monitor a Job

When you submit a job, the scheduler assigns it a *state* and changes that state over time. The four basic states are

- Unscheduled - Likely a sign something is wrong.
- DepWait - This job is set to run after another finishes.
- Waiting - It's in the queue and waiting for resources to become available.
- Running - As far as the scheduler is concerned, everything is working.

The main command for monitoring your own jobs is `qstat`.

qstat argument	description
qstat	Show status of only your jobs.
qstat -j <jobid>	Show details about one job.
qstat -ext	Extended job information.
qstat -r	Resource requirements.

The `showq` command will show jobs from all users.

showq argument	description
showq	Show summary of jobs in queue.
showq -sfb +priority	Sort by increasing priority.
showq -sfa -queuetime	Sort by decresasing queue time.

The possible sorting arguments are `priority`, `starttime`, `queuetime`, `wclimit`, `jobname`, and `remaining`.

To cancel a failing job, use `qdel`.

qdel argument	description
qdel <jobid>	Cancels a waiting or pending job.
qdel -f <jobid>	Cancels a running job.

4.6 Exercise: Run in Batch

Goal: Monitor a job and evaluate its success.

You compiled code earlier. Now it's time to run that code.

1. Go to the same directory as the HPL-2.0 binary with "cd ~/hpl-2.0/bin/icc_mkl".
2. Compare the sample batch file, ranger.sh, with that of Section *Constructing a Batch File*. You want your job to use 16 cores, and your account for this class is 20100519HPC. Edit it so that it has the right account.
3. Submit `batch.sh` using the `qsub` command. Remember its job id.

4. Can you find detailed information on what directory it will run in using `qstat`? (Use Section *Monitor a Job*.
5. Can you tell when it will run? Mostly, not, so can you tell how many jobs are waiting?
6. On what node did it run?
7. When the job finishes, do you see its output file in the same directory from which you submitted the file?

4.7 Discussion: Why the Long Queue?

You can get a list of queues with `qconf -sql` and look at details of a particular queue with `qconf -sq queuename`.

1. Look at the `hostlist` and `user_lists` of development, normal, and large queues. Why make different host lists or user lists? What behavior are they trying to encourage?
2. Do any queues have higher or lower priority according to `qconf`? The `qstat -g c` command shows how many slots each queue is permitted to use. What incentives are there to prefer certain queues, and what are those incentives?
3. What are the consequences of a 24 hour max queue time?

4.8 Further Exercise: Using Scheduler-Defined Variables

The scheduler defines variables for the job to use when it runs on the node. The simplest are `$JOB_ID`, which is the number the scheduler assigns to the job and `$JOB_NAME`, which is the name you give with the `-N` switch.

You can see what variables are defined by submitting a script whose body contains the following:

```
env|sort>variables.txt
```

This will show many variables which begin with `$PBS`. Those that start with `$PBS_O` refer to the state of the shell when you submitted the job. So try submitting a job and take a look.

5 File Management

Now you have run your job and begun to generate lots of data. Where do you put it? How do you get it home?

5.1 Filesystems on Ranger

Ranger has three main filesystems, available from both login nodes and compute nodes.

Filesystem	Total Size	Per User Quota	Shortcut	Retention Policy
$HOME	~100 TB	6 GB	`cd`	Nightly backup
$WORK	~200 TB	350 GB	`cdw`	No backup
$SCRATCH	~800 TB	400 TB	`cds`	Purged every 10 days

These filesystems reside on 73 Sun x4500 I/O servers and 4 metadata servers, served by Lustre. They are highly parallel and very fast.

You can query quota information about individual directories with the lfs command. For instance, to see your quota on $WORK:

```
$ lfs quota -u $USER $WORK
```

Find your usage either from the login splash screen or with the disk usage command. This finds the number of megabytes under $HOME:

```
$ du -sm $HOME
1316    /share/home/00692/train00
```

5.2 Tape Archive

There is a 1.7 PB tape archiver, called Ranch, attached to Ranger. It is available for you to use as needed. To access it, use secure copy or the bbcp command to copy files to your directory under the machine. The syntax for copying to your archive directory on Ranch is:

```
$ scp filename ${ARCHIVER}:${ARCHIVE}
```

For more on this, look at the Ranch User Guide.

5.3 File Transfer with BBCP

You could do this exercise on your own computer later. It requires an installation of BBCP on the local machine.

BBCP is a program that functions much like secure copy, or SCP, except that it has several enhancements to transfer files much more quickly. In order to use this to transfer files to or from Ranger, you would have to install it on your local machine, and it seems to install best on Linux or other Unixes. Darwin builds for Mac are possible, too.

Download the source code from http://www.slac.stanford.edu/~abh/bbcp/bbcp.tar.Z. To copy a file to ranger, use it like scp:

```
[local]$ bbcp filename login3.ranger.tacc.utexas.edu:
```

More information is in the Ranger user guide, under High speed transfers.

5.4 File Transfer with GridFTP

You could do this exercise on your own computer later. It requires an installation of GridFTP on the local machine.

GridFTP is a protocol designed to transfer files using grid credentials. With GridFTP, you can

- Login once with your TeraGrid credentials and transfer files many times without passwords.
- From one machine, execute a command that transfers files between two other machines.
- Encrypt files as they are sent for extra security.
- Transfer files quickly using many other features.

There are several applications that can do GridFTP, all described at https://www.teragrid.org/web/user-support/gridftp. You would have to install one on your local computer in order to exchange files with Ranger. Ranger's gridftp address is `gridftp.ranger.tacc.utexas.edu`.

5.5 Exercise: Retrieve Files with Secure Copy

This is a first experience with Secure Copy to show how easy it is to transfer files back and forth. An alternative, also easy, method is to use Secure FTP, or SFTP. On Windows, try WinSCP. On the Mac, try Fetch.

Find the Linux and Mac or Windows sections, as appropriate.

5.5.1 From Linux or Mac

1. Open a terminal window with a right-click on the desktop. You should now have two windows open, one that is on the local machine and one that is logged into Ranger.
2. On Ranger, create a 1 GB file with `cd && dd if=/dev/zero of=gig bs=1G count=1`.
3. Do an `ls` to find out how large this file is.
4. On your local machine, copy that file from Ranger with `scp`. Put your username where the example shows `train2xx`. The single period at the end represents the current directory on the local drive as the place to store the retrieved file.

```
[local]$ scp train2xx@ranger.tacc.utexas.edu:gig .
```

About how long did that take? Now copy the file back.

```
[local]$ scp gig train2xx@ranger.tacc.utexas.edu:gig
```

That should transfer files from the local machine to the remote one.

5.5.2 From Windows

On Windows, you need to install a program called Pscp in order to transfer files with Secure Copy. This program is part of the suite with Putty, the SSH client.

1. You already have one Putty window open and logged into Ranger. Now open a local command line by going to the Start Menu, looking under All Programs, Accessories, and then open "Command Prompt". In what directory is this command prompt?
2. Download `pscp` into the same directory as the local command prompt. Do this by right-clicking on this link, http://tartarus.org/~simon/putty-snapshots/x86/pscp.exe, and choosing "save as..".
3. On Ranger, create a 1 GB file with `cd && dd if=/dev/zero of=gig bs=1G count=1`.
4. Do an `ls` to find out how large this file is.
5. Run pscp in the local command prompt.

```
[local]$ pscp train2xx@ranger.tacc.utexas.edu:gig .
```

About how long did that take? Now copy the file back.

```
[local]$ pscp gig train2xx@ranger.tacc.utexas.edu:gig
```

That should transfer files from the local machine to the remote one.

5.6 Further Exercise: File Transfer Competition

There are two faster ways to transfer files, `bbcp` and `gsiftp`, but we can't test them from any of the lab computers because they require installation of other programs. If you have your own machine, you might try this after class.

First, make a nice big file. On what filesystem would you make a large temporary file? Look above to choose one, and we'll call it "chosen_directory" below. Now create a file called gig and see where it's located:

```
$ cd chosen_directory
$ dd if=/dev/zero of=gig bs=1G count=1
$ pwd
```

Transfer that file with `bbcp` and compare with the time of `scp`:

```
[local]$ bbcp train2xx@login3.ranger.tacc.utexas.edu:/path/to/gig .
[local]$ scp  train2xx@login3.ranger.tacc.utexas.edu:/path/to/gig .
```

If you have a TeraGrid account, you can try the `gsiftp` program. It has many possible parameters. Try the timing of with these commonly-used options:

```
[local]$ grid-proxy-init
[local]$ globus-url-copy -stripe -tcp-bs 11M -vb \
gsiftp://gridftp.ranger.tacc.teragrid.org//path/to/gig file:///gig
```

For more information, look at the TeraGrid GridFTP page.

5.7 Discussion: Transfer Techniques

- Which transfer method is fastest?
- How long would it take to transfer a terabyte (a thousand gigabytes) with the fastest method?
- Do you know of faster ways to transfer data?
- How could you avoid transfer of data?