User Environment

Doug James
Overview

Effective users, good citizens

• Getting Started – Access to Lonestar
• Getting Acquainted – A Tour of “The Cluster”
• Getting Work Done – Using Lonestar
• Getting Along – Good Citizenship

• Lab 1 – Test Drive
• Supplemental Material – end of slide deck
Disclaimers

• Tone: breadth rather than depth
• Hardware: Lonestar only
• Pacing and Interactivity: lecture vs lab
References: Not an Afterthought!

- TACC User Guides, Usage Policies, etc.
  [http://www.tacc.utexas.edu/user-services](http://www.tacc.utexas.edu/user-services)
- man pages and help systems
  - Try “man” and “man -k” before command name
  - Try command name with -h, --help, -help, help
  - Try command name with no argument
- Web: tutorials, cheat sheets, forums
  - Include error messages in search strings
- TACC and XSEDE ticket systems
Getting Started

Access to Lonestar and Other TACC Resources
Before you log in...

• Step 1 -- Portal Account
  – [www.portal.tacc.utexas.edu](http://www.portal.tacc.utexas.edu)
  – Click on “Account Request”

• Step 2 -- Portal Activation
  – Click on the email confirmation
    • Check junk/spam mail if you don’t see the message

• Step 3 – Allocation for specific machine(s)
  – PI must request allocation or tie you to one
Initial Login

• Login with ssh or equivalent
  % ssh lonestar.tacc.utexas.edu
  % ssh username@lonestar.tacc.utexas.edu
  % ssh -X lonestar.tacc.utexas.edu
  % ssh -Y lonestar.tacc.utexas.edu

• Connects you to login1.lonestar.tacc.utexas.edu or login2.lonestar.tacc.utexas.edu

• Don’t overwrite ~/.ssh/authorized_keys
  – Feel free to add to it if you know what it’s for
  – Mistakes can prevent jobs from running
Shells and Startup Scripts

• OS is Linux
• TACC supports major shells
  – bash, csh, sh (Bourne), ksh (Korn), tcsh, zcsh, ...
• Change default shell via “chsh -s <login shell>”
  – Can take an hour or two to take effect
• System-level startup files execute before account-level
• It’s worth your trouble to understand startup files
  – e.g. .bash_profile (or .profile) and .bashrc
  – Easiest way to customize environment (prompt, modules, aliases)
  – Caution: environment assoc’d with shell (~ “window”), not acct
  – Caution: avoid using “echo” in startup scripts (will break scp et al!)
Text Editors

• Pick your favorite; become proficient
  – nano – simple
  – vi (vim) – terse
  – emacs – powerful

• Appreciate cross-platform issues
  – Win to Linux – dos2unix utility
  – Linux to Win – Wordpad rather than Notepad
Getting Acquainted

A Tour of “The Cluster”
Generic Cluster Architecture

Internet

Server

PC+

File

Server

PC+

Switch

PC

PC

PC

PC

Lonestar

- Ethernet
- Myrinet, IB
- FCAL, SCSI,...
Nodes Have Personalities and Purposes

Internet

Login Node
eg. login1$

ssh

qsub job

Queue

Compute Nodes
eg. c101-423

“Front end”
or “head node”

“Back end”
Lonestar Basic Specs
(an excuse to cover terminology)

• 1888 nodes (blades, PCs)
• Each node has 12 cores (two six-core processors) for a total of 22,656 cores
• Most nodes have 24G of RAM
• We also have five 24 core, 1T largemem nodes and eight GPU nodes
Available File Systems (Lonestar)

2 Hex-Core CPUs

- Home
- Ranch
- Scratch
- Work

Available File Systems:
- NFS
- Local
- Lustre

All Nodes
Use the aliases `cd`, `cdw` and `cds` to change directory to `$HOME`, `$WORK` and `$SCRATCH` respectively.

http://www.tacc.utexas.edu/user-services/user-guides/lonestar-user-guide#overview:filesystems
File Transfers

- scp or rsync recommended; other protocols possible
- Avoid recursive (-r) transfers; bundle files with tar
- Compression and optimization rarely necessary
- On Ranch, understand staging
- Avoid simultaneous transfers and tar jobs

Cross-platform issue: filenames (spaces, capitalization)

http://www.tacc.utexas.edu/user-services/user-guides/ranch-user-guide
Getting Work Done

Using Lonestar
Lmod: TACC’s Module System

• “Sets the table” by loading tools you need
• Prevents errors by managing dependencies
• Why this is so important
  – Multiple compilers
  – Multiple MPI stacks (each dependent on compilers)
  – Varied user apps, libraries, tools (often dependent on compiler and MPI stack)
Key Module Commands

% module help          {lists options}
% module load <module> {add a module}
% module avail         {lists available modules}
% module unload <module> {remove a module}
% module swap <mod1> <mod2> {swap two modules}
% module help <mod1>  {module-specific help}
% module spider        {lists all modules}
% module spider petsc  {list all versions of petsc}
% ml                    {abbrev for module list}
% ml <module>           {abbrev for module load}
% module reset         {return to system defaults}
(Personal) Default Modules

- Establish a personal default module environment
  
  ```bash
  $ module reset
  $ module load ddt
  $ module load fftw3
  $ module setdefault
  ```

- Also possible: named collections of modules, e.g.
  
  ```bash
  $ module setdefault sequencing
  ```
  - Execute "module help" for more info

- This is one way to achieve reliability and repeatability
Compilers

• Three major compiler families
  – Intel 11.1 (default): icc, icpc, ifort
  – PGI 12.5: pgcc, pgCC, pgf77, pgf90
  – Gnu 4.2.0 through 4.4.5 (and 3.4.6): gcc, g++, gfortran

• Additional specialized compiler technologies
  – E.g. cuda support (nvcc): module load cuda

• Numerous math libraries available, but MKL especially important (module load mkl)
  – LAPACK, BLAS, and extended BLAS (sparse), FFTs (single- and double-precision, real and complex data types).
  – APIs for both Fortran and C
MPI Compilation

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Language</th>
<th>Type Suffix</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpicc</td>
<td>c</td>
<td>.c</td>
<td>mpicc prog.c</td>
</tr>
<tr>
<td>mpicxx</td>
<td>C++</td>
<td>.C, .cc, .cpp, .cxx</td>
<td>mpicxx prog.cc</td>
</tr>
<tr>
<td>mpif77</td>
<td>F77</td>
<td>.f, .for, .ftn</td>
<td>mpif77 -Vaxlib prog.f</td>
</tr>
<tr>
<td>mpif90</td>
<td>F90</td>
<td>.f90, .fpp</td>
<td>mpif90 -Vaxlib prog.f90</td>
</tr>
</tbody>
</table>

• The mpiXXX commands are shell scripts.
• They call the underlying C/C++/Fortran compiler.
• This depends on the currently-loaded compiler module.
SGE: Your Ticket to Compute Nodes

• Four ways to get to the back end (compute nodes):
  – Batch job: `qsub <batchfilename>`
  – Interactive session: `idev <optional flags>`
  – Run special app: e.g. `ddt`
  – `ssh` to node on which you’re already running

• If you launch your simulation directly (without `qsub` or its equivalent) you’re running on the front end (login nodes)
  – One of the easiest ways to get your account suspended
Batch Submission Process

- Internet
- ssh
- Login Node eg. login1$
  - qsub job
  - Queue
- Compute Nodes eg. c101-423
- Master Node
  - mpirun –np # ./a.out
  - ibrun ./a.out

Queue: Job script waits for resources.
Master: Compute node that executes the job script, launches all MPI processes.
Lonestar Queues

<table>
<thead>
<tr>
<th>Queue</th>
<th>Max Runtime</th>
<th>Max Cores</th>
<th>SU Charge Rate</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>24 hours</td>
<td>2052</td>
<td>1.0</td>
<td>Normal usage</td>
</tr>
<tr>
<td>development</td>
<td>1 hour</td>
<td>264</td>
<td>1.0</td>
<td>Debugging, allows interactive jobs</td>
</tr>
<tr>
<td>serial</td>
<td>12 hours</td>
<td>1</td>
<td>1.0</td>
<td>Uniprocessor jobs</td>
</tr>
</tbody>
</table>

- Other specialized queues exist
  - e.g. largemem, gpu
- Scheduling/priority system considers many factors
  - “fair share” algorithm factors in other recent/scheduled activity
SGE: Basic MPI Job Script

#!/bin/bash

#$ -V     # Inherit the submission environment
#$ -c wd  # Start job in submission directory
#$ -N myMPI  # Job Name
#$ -A 20121022HPC2Day  # Project account name
#$ -j y  # combine stderr & stdout into stdout
#$ -o $JOB_NAME.o$JOB_ID  # Name of the output file
#$ -pe 12way 24  # Requests 12 cores/node, 24 cores total
#$ -q development  # Queue name
#$ -l h_rt=01:30:00  # Run time (hh:mm:ss) - 1.5 hours

module load fftw3
module list

ibrun ./a.out  # Run the MPI executable named "a.out"
### Wayness

<table>
<thead>
<tr>
<th>No. Processes</th>
<th>Stampede (16 way)</th>
<th>Lonestar (12 way)</th>
<th>Longhorn (8 way)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1way 16</td>
<td>1way 12</td>
<td>1way 8</td>
</tr>
<tr>
<td>4</td>
<td>4way 16</td>
<td>4way 12</td>
<td>4way 8</td>
</tr>
<tr>
<td>8</td>
<td>8way 16</td>
<td>8way 12</td>
<td>8way 8</td>
</tr>
<tr>
<td>12</td>
<td>12way 16</td>
<td>12way 12</td>
<td>6way 16</td>
</tr>
<tr>
<td>16</td>
<td>16way 16</td>
<td>8way 24</td>
<td>8way 16</td>
</tr>
<tr>
<td>32</td>
<td>16way 32</td>
<td><strong>8way 48</strong>*</td>
<td>8way 32</td>
</tr>
<tr>
<td>48</td>
<td>16way 48</td>
<td>12way 48</td>
<td>8way 48</td>
</tr>
<tr>
<td>64</td>
<td>16way 64</td>
<td>8way 96</td>
<td>8way 64</td>
</tr>
</tbody>
</table>

*This means 8 MPI processes per node, and 48 total cores in job. Since Lonestar has 12 cores per node, this job uses 48/12 = 4 nodes. With 8 processes on each of 4 nodes we get 8 x 4 = 32 MPI processes."
Additional Software

• Computation: NumPy, SciPy, Octave, R,…
• Performance, Analysis, Debugging: TAU, PAPI, PerfExpert, ddt,…
• Parameter Studies: launcher and pylauncher
• File i/o: hdf5, parallel hdf5, netcdf
• Build and install your own tools
  – We strongly recommend installing in $WORK
  – Standard trick: ./configure --prefix=$WORK/mydir
Getting Along

Good Citizenship
The Keys to Good Citizenship

Remember you are sharing resources
(login nodes, file systems, bandwidth)

Use components for intended purposes
Login nodes: appropriate use

• Building software
  – Compilers generally not visible on back end
• Managing files
  – Editing, transfers, tar/untar
• Submitting, monitoring, managing batch jobs
  – qsub, qstat, qdel, showq
• Launching interactive sessions
  – idev, ddt, etc.
Login nodes: inappropriate use

• Don’t do science on the front end
  – Access compute nodes with qsub, idev, or equiv
  – Don’t launch exe directly

• Avoid simultaneous instances of demanding processes
  – Parallel builds (e.g. make –j), tar/untar, transfers
File System Citizenship

• Avoid running jobs from $HOME
• Run demanding jobs from $SCRATCH
• Avoid frequent i/o when possible
• Minimize simultaneous i/o from many processes
• Learn to recognize/avoid other stressors
  – e.g. under-the-table stat (du, default ls) on big dirs
• Know when it’s time to learn/use parallel i/o
Lab 1

Test Drive
Supplemental Material
Modules

• Available modules depend on:
  – The compiler (eg. PGI, Intel, GCC) and
  – The MPI stack selected

• On Lonestar, default is Intel/Mvapich2

• To unload all grid-related (gsi, *** ) modules:

  login3$ module unload CTSSV4

• To return to _our_ system default modules:

  login3$ module purge; module load TACC
Modules – User View

• module spider phdf5
• module spider hdf5
• module spider hdf5/1.8.5
• module spider phdf5/1.8.5
• module spider .*hdf5 ← this is a regular expression NOT a wildcard
Modules – System View

• Modules available before selecting compiler (example):

```
login3$ module purge; module avail

------------------------------------- /opt/apps/modulefiles -------------------------------------
beta                                gzip/1.3.12                 papi/3.6.0 (default)
bunutils-amd/070220                 intel/10.1 (default)         pgi/7.1
ddt/2.3.1                           intel/9.1                      pgi/7.2-5 (default)
gcc/4.2.0                            intel/9.1                      pgi/8.0-6
GCC/4.3.2                           irods/2.1                        postgres/8.3.6
GCC/4.4.0                           launcher/1.3                     python/2.5.2
GCC/4.4.1 (default)                 lua/5.1.4                          *              star-ccm/4.04.011
   *                                  mkl/10.0 (default)                  *
   *                                  *
   *                                  *
```
## Modules – System View

- Additional modules become available after choosing a compiler...

```bash
login3$ module load intel; module avail
```

```
+--------------------------------------------+----------------+----------------+
| acml/4.1.0                                | hecura-debug   | mvapich/1.0    |
| autodock/4.0.1                            | hecura/0.1     | mvapich/1.0.1  |
| boost/1.37.0                              | hecura/1.4rc2  | mvapich2-new/1.2|
| boost/1.39.0 (default)                    | hecura/trunk_2009_09_20 (default) | mvapich2/1.2 |
| fftw3/3.1.2                               | hmmer/2.3.2    | nco/3.9.5      |
| gotoblas/1.26 (default)                   | metis/4.0      | netcdf/3.6.2   |
| gotoblas/1.30                             | mvapich-devel/1.0 | openmpi/1.2.4 |
| gotoblas2/1.00                            | mvapich-old/1.0.1 | openmpi/1.2.6 |
| hdf5/1.6.5                                | mvapich-ud/1.0 | openmpi/1.3 (default) |
```
• And even more after selecting an MPI stack.

```bash
login3$ module load mvapich; module avail
```

```
-------------------------- /opt/apps/intel10_1/mvapich1_1_0_1/modulefiles --------------------------
amber/9.0
arpack/2.1
charm++/6.0
desmond/2.0.4
espresso/4.1-scalapack
fftw2-test/2.1.5
fftw2/2.1.5
gamess/03_2007
gromacs/4.0.5 (default)
hypre/2.0.0-SmallScale (default)
ipm/0.922
kojak/2.2 (default)
lammps/25Aug09 (default)
mpiP/3.1.1
mpiblast/1.5.0
namd/2.6
nwchem/5.0
nwchem/5.1.1 (default)
pdtoolkit/3.12 (default)
petsc/2.3.3 (default)
petsc/2.3.3-complex
petsc/2.3.3-complexdebug
petsc/2.3.3-complexdebugdynamic
petsc/2.3.3-complexdynamic
petsc/2.3.3-debug
petsc/2.3.3-dynamic
petsc/3.0.0-complexdebug
petsc/3.0.0-complexdebugdynamic
petsc/3.0.0-complexdynamic
petsc/3.0.0-cxx
petsc/3.0.0-cxxdebug
petsc/3.0.0-debug
petsc/3.0.0-debugdynamic
petsc/3.0.0-uni
tao/1.9 (default)
tao/1.9-debug
```

```
petsc/3.0.0-debugdynamic
tao/2.17 (default)
thrp/2.1.5

petsc/3.0.0-dynamic
petsc/3.0.0-uni
tao/2.17 (default)
```

```
plapack/3.2 (default)
phdf5/1.8.2
pmetis/3.1
scalapack/1.8.0
slepc/2.3.3 (default)
slepc/2.3.3-complex
slepc/2.3.3-complexdebug
tao/1.9 (default)
tao/1.9-debug
```

```
trilinos/9.0.0
```

---

**TACC**
**TEXAS ADVANCED COMPUTING CENTER**
The default modules are suitable for most users.
Module Hierarchy

• On Lonestar we have three compiler families (multiple versions)
  – Intel, GCC, PGI

• On Lonestar we have two MPI implementations (multiple versions)
  – Mvapich2, Openmpi

• Packages like Petsc or FFTW2 could have 6 or more versions (not all combinations exist)
$ module spider

Tells what packages are on the system across all compiler/mpi pairings.

$ module avail

Tells what packages are available now (with currently loaded modules).
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 and emacs (search with regular expressions, etc..)
Vim & Emacs

• Vim command cheat sheet
  – http://www.tuxfiles.org/linuxhelp/vimcheat.html

• Emacs command cheat sheet
  – http://emacswiki.org/emacs/ReferenceCards
Job Sizing with SGE on Lonestar

• You must always put a multiple of 12 next to the name of the parallel environment
  $$ -pe \text{ 12way 48} \{48 \text{ cores, 12 cores/node, 4 nodes, 48 MPI tasks}\} $$
  $$ -pe \text{ 6way 48} \{48 \text{ cores, 6 cores/node, 4 nodes, 24 MPI tasks}\} $$

• Reduced wayness allows
  – more memory per task (4GiB/core in this example)
  – Threaded programming (OpenMP, Pthreads)
  – Hybrid programming (MPI/OpenMP)

• SGE doesn’t automatically handle the case where the number of tasks you want is not a multiple of 12

• If you want a non-multiple of 12, you may set
  $$ -pe \text{ 12way 24} $$
  $$ \ldots $$
  $$ \text{export MY_NSLOTS}=23 $$
  $$ \ldots $$
  $$ \text{ibrun ./mycode} $$
SGE: Memory Limits on Lonestar

- Default parallel job submission allocates all 12 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 12!

<table>
<thead>
<tr>
<th>Parallel environment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12way</td>
<td>12 tasks/node, 1.9GB/task</td>
</tr>
<tr>
<td>1way</td>
<td>1 task/node, 24 GB/task</td>
</tr>
</tbody>
</table>

Available wayness on Lonestar:
12way 11way 10way 8way 6way 4way 2way 1way