Computing Environment

Ankit Bhagatwala
NERSC User Services Group

New User Training
March 21, 2016
Node Types

• Login nodes
  – Shared with other users
  – Code compilation, job preparation and submission

• Compute nodes
  – Not shared (except shared queue on Cori)

• Service nodes
  – File system access (lustre), data movement (HPSS), network connections to outside world, etc.
  – Not accessed directly
Login Node Configuration

- **Edison**
  - Twelve nodes
    - 16 cores, 2.0 GHz Intel Sandy Bridge, 512 GB

- **Cori**
  - Twelve nodes
    - 16 cores, 2.0 GHz Intel Sandy Bridge, 512 GB

- **Genepool**
  - Four nodes
    - 8 cores, 2.3 GHz Intel Sandy Bridge, 32 GB

- **PDSF**
  - Three nodes
    - 16 cores, 2.6 GHz Intel Sandy Bridge, 125 GB
Login Node Access

• Connect (via ssh) to load balancer
  % ssh edison.nersc.gov
  % ssh cori.nersc.gov
  % ssh genepool.nersc.gov
  % ssh pdsf.nersc.gov

• Load balancer selects login node based on:
  – Number of connections
  – Memory of previous connections from same IP
Login Node Usage

• Login nodes are shared by many users, all the time
• Edit files, compile programs, submit batch jobs
• Some post-processing/data analysis
  – IDL, MATLAB, NCL, python, etc.
• Some file transfers
  – Use data transfer nodes for large/long-running transfers (dtn[01-04].nersc.gov)
• Please use discretion
  – All users get frustrated by sluggish interactive response
  – If unsure, please write to consult@nersc.gov
Login Node Guidelines

• Use *no more* than 50% of available cores
• Use *no more* than 25% of available memory
• Limit use of parallel “make”
  \%
  `make -j 32 all`
• NERSC will kill user processes if login nodes become unacceptably slow or unresponsive
• Terminate idle sessions of licensed software
  – IDL
  – MATLAB
  – Mathematica
Shell Initialization Files

• Standard dot files are maintained by NERSC
  – .bashrc, .profile, .cshrc, .login, etc.
  – Symbolic links to read-only files

• Personal dot files
  – Aliases, environment variables, modules, etc.
  – Use .ext suffix (“.ext files”) .bashrc.ext, etc.

• Broken? Use “fixdots” to start over
  – Creates $HOME/KeepDots.<timestamp>
  – Restores all dot files to default state
  – If PATH corrupted:
    /usr/common/usg/bin/fixdots

• Use NIM to change default login shell
NERSC Supported Software

• NERSC provides a wide range of software
  – Scientific Applications
    • VASP, Amber, NAMD, ABySS, ...
  – Compilers
    • Intel, GCC, PGI, Cray
  – Scripting Languages
    • perl, python, R - including common packages for each
  – Software Libraries (some maintained by Cray)
    • blas/lapack (MKL), boost, hdf5, netcdf, ...
  – Development utilities
    • git, mercurial, cmake, ...
  – Debuggers and Profilers
    • CrayPat, DDT, TotalView, gdb, MAP, darshan, IPM, VTune
  – Visualization
    • Visit, ParaView, VMD, ...

• See complete list
  – $ module avail
  – http://www.nersc.gov/users/software/
Software is Managed by Modules

• Identify the software you need
  – Use the NERSC website
    http://www.nersc.gov/users/software/

• Load the module
  % which idl
  idl: Command not found.
  % module load idl
  % which idl
  /usr/common/usg/idl/idl82/bin/idl
Loading Modules

• Different module for each version of software
  – Syntax: `<name>/<version>`
  – Default provided if no `<version>` supplied

```bash
% module avail idl
idl/7.1   idl/8.0   idl/8.2 (default)
% module load idl/7.1
```

• Load modules in every batch script
  – Ensure correct run-time environment
Other Useful Module Commands

module unload <modulename>
  – Remove the module from your environment

module swap <module1> <module2>
  – Unload one module and replace it with another
    module swap PrgEnv-intel PrgEnv-gnu

module list  -- Very useful!
  – See what modules you have loaded right now

module show <modulename>  -- Very useful!
  – See what the module actually does

module help <modulename>
  – Get more information about the software
Default Modules

• When you login, many *default* modules are loaded automatically
  
  – Usually foundational modules which are required to get proper function from the system
    • Build environment, required libraries and applications, batch environment
  
  – Use caution in unloading these

• Swapping to functionally equivalent module may be OK

  `edison% module swap PrgEnv-intel PrgEnv-gnu`
Types of Modules

• Applications
  – VASP, amber, blast, ...
  – Usually only set PATH, LD_LIBRARY_PATH

• Libraries
  – Set LD_LIBRARY_PATH
  – Set “helper” environment variable for building software
    • Header/include file search paths (e.g. C_INCLUDE_DIR)
    • Library search paths (e.g. PETSC_DIR, HDF5, PYTHONPATH)
    • module show command useful here
      % module load hdf5
      % mpicc mycode.f $HDF5
Module pitfalls

- **Output from** `module <command>` **is piped to** STDERR, **and not** STDOUT

- **Need to re-direct output** if you need to pipe to another command
  - `module list 2>&1 | grep dmapp`

- **Need to be re-loaded** for each newly spawned shell (unless specified in setup scripts)

- No inverse mapping from library name to module

- Most popular packages are pre-packaged by Cray and have “cray-” prepended to their names
  - `cray-petsc` instead of `petsc`
  - `cray-netcdf` instead of `netcdf`
  - `cray-hdf5-parallel` instead of `hdf5-parallel`
Cray Programming Environment

• Compiler specific
  PrgEnv-intel, PrgEnv-cray, PrgEnv-gnu (modules)
  – Intel is default on Edison and Cori

• Swapping Programming Environments
  module swap PrgEnv-gnu PrgEnv-intel
  – swaps compiler
  – no need to swap libraries!
Compiler Wrappers

• On Cori / Edison:
  – Defined by PrgEnv-* modules
  – ftn (Fortran), cc (C), CC (C++)
  – Provides include header and library search paths for MPI, common math libraries (e.g., cray-libsci), Cray system software
  – Provides consistent level of optimization across compilers

• Seldom need native compilers!
  (ifort, icc, gcc, g++ etc)
Resources

http://www.nersc.gov/users/software/nersc-user-environment/

http://www.nersc.gov/users/software/nersc-user-environment/modules/

http://www.nersc.gov/users/computational-systems/edison/programming

http://www.nersc.gov/users/computational-systems/cori/programming/
Thank you.
CHOS Environment

- Provides different OS environments
  - Often different third-party software
    - Some software packages have specific OS requirements
      - Possibly due to validation requirements
- Used on Genepool and PDSF
- Transparent
  - Default configuration for most users
  - Alternate configurations for some users
- Details on website
Login Node Monitoring

• Determine number of available cores
  \% grep processor /proc/cpuinfo | wc -l

• Determine amount of physical memory
  \% grep MemTotal /proc/meminfo

• Use “top” command to view process activity
Software is Managed by Modules

- NERSC provides many versions of many software packages
- Maintaining all these separate software installations on heterogeneous systems is a major challenge!
  - Software can’t just be installed in the base operating system
    - How many copies of /usr/bin/vasp could be supported?
  - Each software package installed in its own directory
    /usr/common/usg/vasp/5.3.5

Modules is the user interface to software at NERSC
Carver “Programming Environment”

• Not as sophisticated as Cray PrgEnv
• Separate compiler and OpenMPI modules

<table>
<thead>
<tr>
<th>Compiler module</th>
<th>OpenMPI module</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgi</td>
<td>openmpi</td>
</tr>
<tr>
<td>intel</td>
<td>openmpi-intel</td>
</tr>
<tr>
<td>gcc</td>
<td>openmpi-gcc</td>
</tr>
</tbody>
</table>

• Must keep libraries consistent with compiler!