Hands-On Practice Session

HPC User Services

LSU HPC LONI

sys-help@loni.org

Louisiana State University

Baton Rouge

March 07, 2018

Outline

- **Things to be covered in the training**
  - Introduction to Linux
    - Basic commands for files/directories and text processing
    - Text editor
    - File permission
  - HPC software environment 1
    - User portal login
    - File transfer
    - The software management tools: softenv and modules
  - HPC software environment 2
  - Introduction to Bash script

Linux, Windows or MAC OS?

- **What operating system (OS) do the LSU HPC/LONI clusters have? To use HPC resources, what kind of OS I must install on my PC?**
  a) All LSU HPC/LONI clusters have Linux OS only, and my PC must have Linux OS.
  b) All LSU HPC/LONI clusters have Linux OS only, and my PC can have either Linux, Windows or MAC OS. ☺
  c) I can use either Linux, Windows or MAC OS on the cluster, and my PC must have Linux OS.
  d) I can use either Linux, Windows or MAC OS on the cluster, and my PC can have either Linux, Windows or MAC OS.
  e) None of the above

Linux - Exercise

- **Accessing cluster via SSH**

```
$ ssh ychen64@mike.hpc.lsu.edu
```

- **pwd**
  - Print working directory - i.e. Where are we currently.

```
$ pwd
```

- **ls**
  - List the contents of a directory.

```
$ ls
```

  - List all contents of a directory with a long listing format.

```
$ ls -la
```

- **mkdir**
  - Create a directory “testdir”.

```
$ mkdir testdir
```

- **cp**

```
$ cp /etc/shells testdir/
$ cp /proc/cpuinfo testdir/
```

  - Copy /etc/shells and /proc/cpuinfo to the directory “testdir”.
  - The above two can be combined into one line.

Linux - Exercise

- **cd**
  - Change directory

```
$ cd testdir
```

  - Change to the upper directory (one level up)

```
$ cd ..
```

  - Change to the previous working directory

```
$ cd -
```

  - Change to the home(login) directory, which is /home/your_username

```
$ cd
$ cd $HOME
$ cd /home/ychen64
$ cd ~
```

  - Change to your own /work directory, which is /work/your_username

```
$ cd /work/ychen64
```

Relative path vs. Absolute path

- **Relative path**
  - A file or directory location relative to the current in the file system.
- **Absolute path**
  - A file or directory location in relation to the root of the file system.
- **Identify the following file/directory path: relative or absolute?**

```
$ mkdir testdir

$ cp /etc/shells testdir/

$ cd testdir

$ cd /work/ychen64
```

Linux - Exercise

- **cat**
  - Display the file contents.

```
$ cat shells
```

  - Not a good option to display a long file.

```
$ cat cpuinfo
```

- **head**
  - Display the beginning part of text file

```
$ head cpuinfo
$ head -20 cpuinfo
$ head -n 20 cpuinfo
```

  - What will happen to this one?

```
$ head 20 cpuinfo
```

- **less**
  - View the file contents without actually opening it.

```
$ less cpuinfo
```

Vim editor - Exercise

```
$ vi cpuinfo
```

- **Command mode**

Type	Function
dd	Delete one line
3dd	Delete 3 lines
yy	Copy one line
3yy	Copy 3 lines
p	Paste below the line of the cursor
u	Undo the last operation
G	Move to the last line
gg	Move to the first line
/	Search strings
:(some number)	Move through file to row #
:set nu	Display the line number

Vim editor - Exercise

➢ **Insert mode**

Type	Function
i	Enter insert mode (-- INSERT -- shows in the bottom left corner)
o	Enter insert mode but start a new line
Esc (Escape key)	Exit insert mode, back to the command mode

➢ **Save and/or quit in Command mode**

Type	Function
:w	Save and continue editing
:wq	Save and quit
:w filename	Save the file to another name (save as..)
:q!	Quit without saving

File permission

```
$ls -l cpuinfo
-r--r--r-- 1 ychen64 Admins 14764 Feb 28 13:41 cpuinfo
```

- **Linux files and directories have three permission groups:**
  - owner, group, and other
- **Three basic permission types:**
  - read(r), write(w), and execute(x).
- **Octal notation (base-8) for file and directory permissions:**
  - r: 4 w: 2 x: 1 if not at all: 0
- **chmod**
  - change permissions

```
$chmod 644 cpuinfo
$chmod u+w cpuinfo
```

- **chown**
  - change the owner to you.

```
$chown ychen64:Admins cpuinfo
```

Account Management
- LSU HPC and LONI User Portals

- **Both portals can be found at the top of http://www.hpc.lsu.edu/**
- **LONI account**
  - https://allocations.loni.org
- **LSU HPC account**
  - https://accounts.hpc.lsu.edu
- **View/Update profile**
  - Change Login Shell at the profile page
- **Search/join your PI's allocation**
- **Check your allocation situation**

File transfer - Exercise

- **Download this slide from the HPC website to your home directory on the cluster.**
  - wget
  - scp
  - Windows SSH client

Cluster Environment - Exercise

- **Useful commands on the head node**
  - check your personal disk quota and usage

```
$ showquota
```

  - check who is on the node

```
$ who
```

  - check allocation balance

```
$ balance
```

Application Software

- **Installed Software**
  - Mathematical and utility libraries
    - FFTW, HDF5, NetCDF, PETSc...
  - Applications
    - Amber, CPMD, NWChem, NAMD, Gromacs, R, LAMMPS...
  - Visualization
    - VisIt, VMD, GaussView
  - Programming Tools
    - Totalview, DDT, TAU...
- **List of software**
  - `http://www.hpc.lsu.edu/resources/software/index.php`
- **Installed under** `/usr/local/packages`
- **User requested packages**
  - Usually installed in user home directory, unless request by a group of users, in which case it will be installed under `/project` or `/usr/local/packages`

Software Environment: Module and Softenv

- **Environment variables**
  - `PATH`: where to look for executables
  - `LD_LIBRARY_PATH`: where to look for shared libraries
  - `LD_INCLUDE_PATH`: where to look for header and include files
- **Other environment variables sometimes needed by various software**
  - `LIBRARY_PATH`, `C_LIBRARY_PATH`
  - `LDFLAGS`, `LDLIBS`
- **SoftEnv**
  - A software that helps users set up environment variables properly to use other software package. Much more convenient than setting variables in .bashrc
  - SuperMike2
- **Modules**
  - Another software that helps users set up their environment. Most supercomputing sites (including XSEDE) use modules.
  - SuperMIC, Philip and QB2

Softenv - Exercise

- **List all packaged with softenv**
  - full list

```
$ softenv
```

  - concise list

```
$ softenv | grep +
$ softenv | less
```

- **Add gromacs-4.5.5**
  - Find the key for gromacs-4.5.5

```
$ softenv -k gromacs
```

  - Set up your environment to use gromacs-4.5.5 (one time change)

```
$ soft add +gromacs-4.5.5-Intel-13.0.0-openmpi-1.6.2
```

  - Check if the variables are correctly set by “which mdrun”

```
$ which mdrun
```

  - delete the key

```
$ soft delete +gromacs-4.5.5-Intel-13.0.0-openmpi-1.6.2
```

  - Set up your environment to permanently use gromacs-4.5.5:

  Add **+gromacs-4.5.5-Intel-13.0.0-openmpi-1.6.2** to the .soft file, and then use command “resoft”, or relogin to the cluster to take it effective.

Modules - Exercise

- **List software packages currently available in the Environment Modules system**
  - list all packages

```
$ module av
```

  - list certain package (e.g. Python)

```
$ module av python
```

- **List all software packages loaded into the user environment**

```
$ module list
```

- **Load/unload software packages into the user environment**

```
$ module load python/2.7.10-mkl-mic
$ module unload python/2.7.10-mkl-mic
```

- **Display the module changes**

```
$ module disp python/2.7.10-mkl-mic
```

- **Load automatically on login**
  - Add **module load python/2.7.10-mkl-mic** to the .modules file. Source the .modules file, or relogin to the cluster to take it effective.

Creating Your Own Module File

- **An example of a simple module file** `(~/my_module/gitkey):`

```
#%Module
proc ModulesHelp { } {
    puts stderr { my compiled version of git.
    }
}
module-whatis {version control using git}
set GIT_HOME /home/fchen14/packages/git-master/install
prepend-path    PATH            $GIT_HOME/bin
```

- **Add the path to the key to the MODULEPATH environment variable:**

```
$ export MODULEPATH=~/my_module:$MODULEPATH
```

- **Then try to use:**

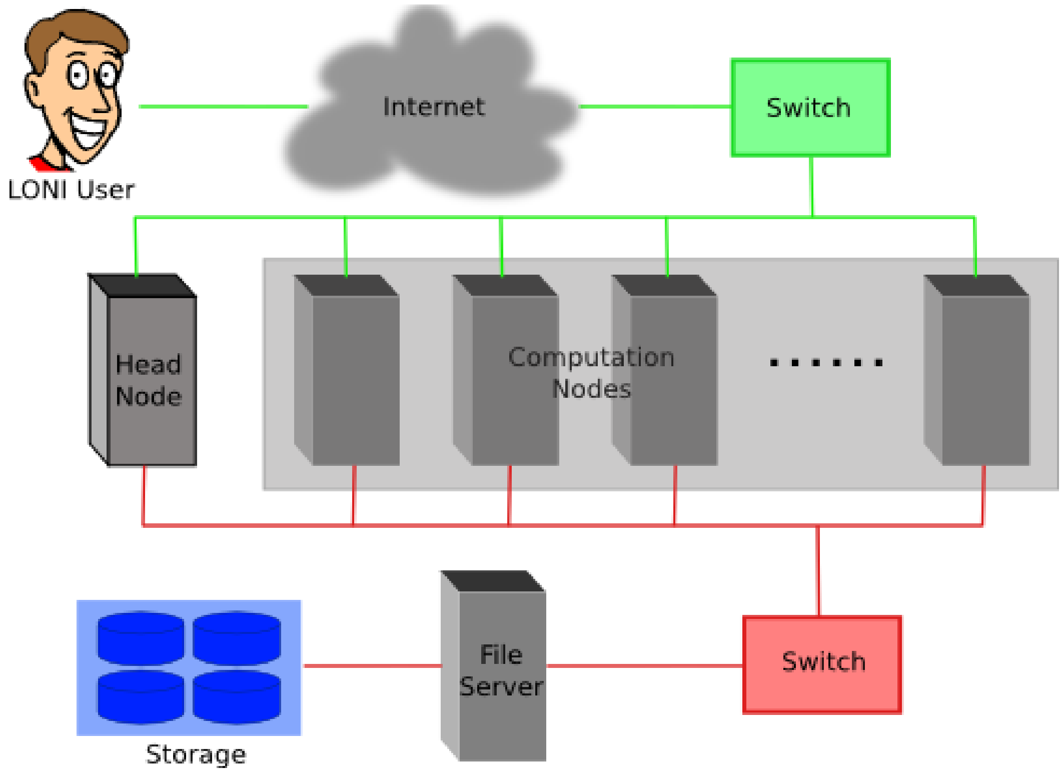
```
$ module load gitkey
$ which git
$ module unload gitkey
$ which git
```

Exercise

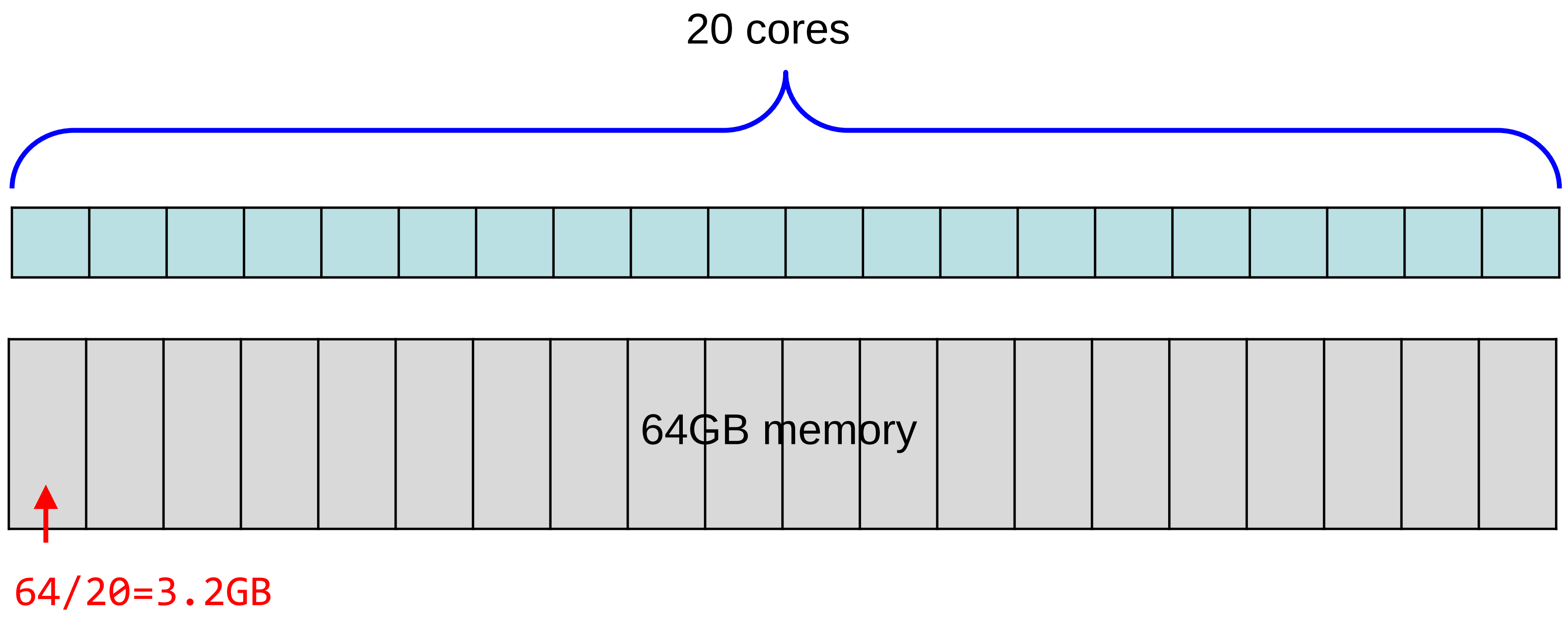
Job Submission

Cluster Environment

- Multiple compute nodes
- Multiple users
- Each user may have multiple jobs running simultaneously
- Multiple users may share the same node



Core and Memory in Single queue



Question:
On QB2, if my job needs 7GB memory, what ppn value should I use?
On SuperMike2, if my job needs 7GB memory, what ppn value should I use?

Exercise (1)

- **Run an interactive job session for 30 min, using `nodes=1:ppn=16 (on SuperMike2), nodes=1:ppn=20 (on SuperMIC/QB2)`**
  - Verify using `hostname` that you are not on the headnode
  - Check available PBS variables and print them
  - Run your favorite code on this interactive session and monitor the usage of memory and CPU cores

- **Submit a batch job to single queue, using `nodes=1:ppn=1`, run the python script `calc_pi.py` (in `/home/fchen14/userenv/pbs_script`) to calculate the value of pi**
  - You can use the sample file in example directory, modify it to your environment:

```
/home/fchen14/userenv/pbs_script/single.pbs
```

Exercise (2)

- **Submit a small job to run “sleep 180”and “print PBS variables”**
  - Create a script to submit a 5 min job and print from within the job script PBS variables $PBS_NODEFILE, $PBS_WORKDIR. Also run “sleep 180” to give you a few minutes to verify status.
  - Once the job is running, find out the Mother Superior node and other slave nodes assigned to your job.
  - Log into MS node and verify that your job is running
  - Modify your script to print hello from each of your assigned nodes*

Exercise 3

- Run the molecular dynamics code lammps using different number of cores/nodes, the input file is already provided in:

  ```
  /home/fchen14/userenv/pbs_script/lj.txt
  ```

  You can copy this file to your directory:

  ```
  cp /home/fchen14/userenv/pbs_script/lj.txt /your/own/dir
  ```

- On SuperMike2, this can be achieved by, for example:

  ```
  $  cd /your/own/dir
  $  soft add +lammps-06Dec12-Intel-13.0.0-openmpi-1.6.2
  $  # run lammps on one SuperMike2 node with 16 cores
  $  mpirun -np 16 -machinefile $PBS_NODEFILE lmp_openmpi -in lj.txt
  ```

- Write a pbs job script in order to:
  - Using 1 node and then monitor the memory usage using the "`top`" command.
  - Using 2 nodes and then monitor the memory usage using the "`top`" command.
  - Using 4 nodes and then monitor the memory usage using "`qshow`"

Basic Shell Scripting Practice

HPC User Services

LSU HPC & LON

sys-help@loni.org

March 2018

Quotation Exercise

1. Print out your $LOGNAME
2. Print date
3. Print `who am i`
4. Print your current directory

Quotation Exercise

```
#!/bin/bash

echo "Hello, $LOGNAME"
echo "Current date is `date`"
echo "User is `who i am`"
echo "Current directory `pwd`”
```

Number Exercise

1. a=5.66; b=8.67
2. Print out sum of a+b
3. z=5
4. Print out result of z+5

Number Exercise

```
#!/bin/bash

a=5.66
b=8.67
c=`echo $a + $b | bc`
echo "$a + $b = $c"

z=5
z=`expr $z + 3`
echo "z=$z"
z=$(($z+5))
echo "z=$z”
```

Loop Exercise

1. Loop through a list of planets (Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune Pluto)
2. Print out list element one by one

Loop Exercise

```
#!/bin/bash

for planet in Mercury Venus Earth Mars
Jupiter Saturn Uranus Neptune Pluto
do
  echo $planet
done
```

Loop Exercise

1. Print out the contents of the current directory

Loop Exercise

```
#!/bin/bash

for file in `ls`
do

echo $file

done
```

Function Exercise

1. Create a “hello” function requiring a name as parameter and print out “hello name”
2. Call the function twice with different parameters

Function Exercise

```
#!/bin/bash
# Passing arguments to a function
hello () {
  echo Hello $1
}
hello John
hello James
```

grep & egrep

- **`grep:`** Unix utility that searches through either information piped to it or files.

- Usage: `grep <options> <search pattern> <files>`
- Options:

`-i`	ignore case during search
`-r,-R`	search recursively
`-v`	invert match i.e. match everything except *`pattern`*
`-l`	list files that match *`pattern`*
`-L`	list files that do not match *`pattern`*
`-n`	prefix each line of output with the line number within its input file.
`-A num`	print `num` lines of trailing context after matching lines.
`-B num`	print `num` lines of leading context before matching lines.

grep Examples

- Search files containing the word `bash` in current directory

```
grep bash *
```

- Search files NOT containing the word `bash` in current directory

```
grep -v bash *
```

- Repeat above search using a case insensitive pattern match and print line number that matches the search pattern

```
grep -in bash *
```

grep Examples

```
100  Thomas  Manager   Sales        $5,000
200  Jason     Developer  Technology  $5,500
300  Raj         Sysadmin   Technology  $7,000
500  Randy    Manager   Sales          $6,000
```

- grep OR : find people either manager or in sales dept

```
grep  'Manager\|Sales'  employee.txt
->  100  Thomas  Manager   Sales     $5,000
    500  Randy   Manager   Sales     $6,000
```

- grep AND: find people who is both sysadmin and in Tech dept

```
grep –i 'sysadmin.*Technology' employee.txt
->  100300  Raj   Sysadmin  Technology  $7,000
```

`sed` commands and flags

Flags	Operation	Command	Operation
`-e`	combine multiple commands	`s`	substitution
`-f`	read commands from file	`g`	global replacement
`-h`	print help info	`p`	print
`-n`	disable print	`i`	ignore case
`-V`	print version info	`d`	delete
`-r`	use extended regex	`G`	add newline
		`w`	write to file
		`x`	exchange pattern with hold buffer
		`h`	copy pattern to hold buffer
		`;`	separate commands

sed Examples

```
#!/bin/bash

# My First Script

echo "Hello World!”
```

sed Examples (1)

- Add flag -e to carry out multiple matches.
- Replace bash with tcsh and Frist with Second

```
cat hello.sh | sed -e ’s/bash/tcsh/g’ -e ’s/First/
Second/g’
```

- Alternate form with ; instead of -e

```
sed ’s/bash/tcsh/g; s/First/Second/g’ hello.sh
```

- The default delimiter is slash (/), try : in place of /

```
sed ’s:/bin/bash:/bin/tcsh:g’ hello.sh
```

sed Examples (2)

- Delete blank lines from a file

```
sed '/^$/d' hello.sh
```

- Delete line `n` through `m` in a file

```
sed '2,4d' hello.sh
```

awk Syntax

```
awk pattern {action}
```

`pattern` decides when `action` is performed

`Actions:`

- Most common action: `print`
- Print file dosum.sh:

```
 awk '{print $0}' dosum.sh
```

- Print line matching files in all `.sh` files in current directory:

```
awk '/bash/{print $0}' *.sh
```

- `$0` Print the entire line, use.
- `NR` #records (lines)
- `NF` #fields or columns in the current line.
- By default the field delimiter is space or tab. To change the field delimiter use the `-F<delimiter>` command.

Awk Examples

```
uptime
11:18am up 14 days 0:40, 5 users, load average: 
0.15, 0.11, 0.17
```

Use awk to print out the entire fields of uptime results

```
uptime | awk '{print $0}'
```

```
Print out current time and #fields
uptime | awk '{print $1,NF}'
11:18am 12
```

Print out # of users

```
uptime | awk -F, '{print $1}'
5 users
```