Intro to Linux on the HPC cluster

Harry Mangalam

harry.mangalam@uci.edu
Some remarks..

• You've heard of Linux...? (~ Darwin/BSD, very unlike Windows, but .. Cygwin!)
• You're interested in it, because ...?
• Google, and read further by yourself.
• Questions, **ASK THEM**, but I may defer answering them immediately.
• “You don’t know what you don’t know”
Computing Philosophy

- Unlike your Science...
- Be lazy.
- Copy others.
- Don't invent anything you don't have to.
- Re-USE, re-CYCLE, DON'T re-invent.
- Don't be afraid to ask others.
- Resort to new code only when absolutely necessary.
Philosophy

• You're not CS, not programmers

• Don't try to be them

• But! Try to think like them, at least a bit

• Google is your friend
Getting Help

- Fix IT Yourself with Google
  http://goo.gl/05MnTi
- Listservs, forums, IRCs are VERY useful for more involved questions.
- The HPC Web page http://hpc.oit.uci.edu/ and HOWTOs there.
- Us – Harry, Joulien, Joseph, Francisco
- BUT!! Unless you ask questions intelligently, you will get nothing but grief.
How to Ask Questions

• Reverse the situation: if you were answering the question, what information would you need?
• Not Science, but it is Logic.
• Include enough info to recreate the problem.
• Exclude what's not helpful or ginormous (use http://pastie.org or http://tny.cz)
• Use text, not screenshots if possible.
Bad Question

Why doesn’t “X” work?

or

“X” doesn't work anymore.
I tried running the new podunk/2.8.3 module this morning and it looks like I can't get it to launch on the Free64 queue. My output files aren't helping me figure out what is wrong.

I am working out of the /bio/joeuser/RNA_Seq_Data/ directory and the qsub script is 'job12.sh'. The output should be in /bio/joeuser/RNA_Seq_Data/output.

I tested it in my login shell with a small data set and it worked OK, but when I submit the job, it appears to go thru the scheduler but then dies immediately when it hits the execution node.

I can't find any output to tell me what's wrong, but the error messages suggest that there's a problem finding libgorp.so.3
HELP US HELP YOU

We Need:

- the directory in which you're working (pwd)
- the machine you're working on (hostname)
- modules loaded (module list)
- computer / OS you're connecting from
- the command you used and the error it caused (in /text/, not screenshot)
- much of this info is shown by your prompt
- try the mayday script from anywhere on HPC

see http://goo.gl/6eZORd
On to HPC

What is the High Performance Computing Cluster?

and...

Why do I need HPC?
On to HPC

- Pod of large (multicore) general purpose computers that..
- run the Linux Operating System
- are linked by some form of networking
- have access to networked storage
- can work in concert to address large problems (altho each core is pretty slow) ..
- ..by *scheduling* jobs very efficiently
HPC in more detail

- ~10K 64b Cores – Mostly AMD, few Intel
- ~55TB aggregate RAM
- ~2PB of storage
- Connected by 1Gb ethernet (100MB/s)
- and by QDR IB (4000MB/s)
- **Grid Engine** scheduler to handle Queues
- > 1600 active users, 100+ are online at anytime
- ~2K-20K job in the Qs
What HPC is NOT

- **NOT**: your personal machine
- It is a shared resource.
- What you do affects all the other users, so think before you hit that 'Enter' key.
- Well secured from mischief and disasters – not an invitation.
Selective Backup only.

You have to choose which data to back up or not.

Most data is stored on RAID6

BUT! Any of that can disappear at any moment

IF ITS VALUABLE, back it up elsewhere --- or the code that generated it.
HPC FileSystem Layout

Orange – Cluster Wide
Black – Node Specific

/ 
├── data/  
│   ├── apps  
│       All Programs are installed here  
│   └── users  
│       Users home directory  
│           – 50GB LIMIT PER USER  
│   └── pub/  
│       Public scratch space, overflow  
│       - 2 TB limit (but only active data)  
│   └── bio/  
│       Space for BIO group → /dfs1  
│   └── som/  
│       Space for SOM group → /dfs1  
│   └── cbcl/  
│       Space for CBCL group → /dfs1  
│   └── dfs1/  
│       BeeGFS Distributed File System  
│           ~620TB  
│   └── dfs2/  
│       BeeGFS Distributed File System  
│           ~403TB  
│   └── dfs3/  
│       BeeGFS Distributed File System  
│           ~500TB  
├── scratch  
│       Node-specific temporary storage per job (faster than all above)  
│       ~1TB – 14TB  
├── fast-scratch  
│       High Speed FileSystem for temporary storage  
│           - 13TB  
├── ssd-scratch  
│       Very High IOPS for DB, other jobs.  
│           ~2TB  
└── /tmp  
    Same as scratch
• You can only have so much space
• 50GB for /data/ (home directory)
• if 6months or older without use – please remove from cluster or tarchive it.
• More for Condo owners or Groups who have bought extra disk space.
• We now have a Selective Backup system, but it's no replacement for your own backups.
Backup Possibilities

• Your UCI 'Google Drive' can be connected to HPC to back up small files

• You can buy a USB drive to use as a backup →

• Use 'rsync' to do incremental backups to it:
  
  rsync -av this_dir MyMac:/this_dir

• Your lab can buy a NAS device and mount it on HPC as an NFS mount

• Your lab can rent space on an HPC filesystem.
• SSH is an encrypted protocol so that info over the connection can't be deciphered by others.

• You MUST use SSH to connect to HPC, using the **command line**....

• ssh underlies 'scp' (secure copy), sftp

• Also 'sshfs' which allows you to attach your filesystem to HPC (or vice versa).
Here vs There

• Your laptop is **HERE** (and HERE is often dynamic)
• (How do you find out your IP #?)
• HPC is **THERE** (and THERE is always static)
• Files have to get from HERE to THERE (so it's always easier to push data from HERE to THERE, but ......)
• Displays are generated THERE but are seen HERE. (both Text and Graphics).
• The point above can be exploited to make life easier. [*byobu* and *x2go*]
• Make sure of where you are (what machine you're logged into) and in which direction the bytes are going.
• Especially when you issue commands like 'rm'.
Commandline Hell
Command Line Cons

• The tyranny of the blank page

• No visual clues

• Type vs click

• Have to know what to type

• HOW DO YOU KNOW WHAT TO TYPE???
Command Line Pros

• It doesn't get much worse than this

• When you do learn it, you'll know it and it probably won't change for the rest of your life

• It's a very efficient way of interacting with the computer (which is why it's survived for 50+yrs).

• You can use it to create simple, but very effective pipelines and workflows.
Keeping SSH Session Alive

• If you need to maintain a live connection for some reason, use **byobu** or **screen**.

• It allows you to multiplex and maintain connections in a single terminal window.

• Somewhat unintuitive interface but very powerful.

• *Find out how to use them via cheatsheets* (Google!!)
Byobu / Screen
Graphics Apps on HPC

- Linux uses X11 for graphics
- X11 is very chatty, high bandwidth, sensitive to network hops/latency.
- If you need graphics programs on HPC, use x2go vs native X11, which does for graphics what byobu does for terminal screens.
- x2go is described in the Tutorial & HOWTO, also … GOOGLE
Network Considerations

$ traceroute hpc.oit.uci.edu
traceroute to hpc.oit.uci.edu (128.200.84.34), 30 hops max, 60 byte packets
1  415-vl110.ucinet.uci.edu (128.200.34.1)  0.434 ms  0.524 ms  0.586 ms
2  cs1-core--415.ucinet.uci.edu (128.195.249.233)  0.376 ms  0.380 ms  0.416 ms
3  dca--cs1-core.ucinet.uci.edu (128.195.239.182)  0.488 ms  0.594 ms  0.736 ms
4  hpc-login-4.oit.uci.edu (128.200.84.34)  0.313 ms  0.300 ms  0.283 ms
Network Considerations

$ traceroute moo.nac.uci.edu
traceroute to moo.nac.uci.edu (128.200.34.95), 30 hops max, 60 byte packets
1  haggis.net (192.168.1.1)  0.694 ms  0.940 ms  1.134 ms
2  172.27.35.1 (172.27.35.1)  2.232 ms  2.301 ms  2.394 ms
3  10.75.151.1 (10.75.151.1)  11.647 ms  11.766 ms  11.855 ms
4  ip68-4-13-176.oc.oc.cox.net (68.4.13.176)  12.249 ms  16.099 ms  16.845 ms
5  ip68-4-11-12.oc.oc.cox.net (68.4.11.12)  17.661 ms  18.192 ms  18.181 ms
6  68.1.1.171 (68.1.1.171)  18.989 ms  23.355 ms  13.053 ms
7  xe-5-1-1.edge2.LosAngeles9.Level3.net (4.53.230.93)  16.391 ms xe-5-0-
    1.edge2.LosAngeles9.Level3.net (4.53.230.85)  16.392 ms xe-9-0-
8  * ** *
9  CENIC.ear1.LosAngeles1.Level3.net (4.35.156.66)  20.376 ms  20.806 ms  20.817 ms
10 dc-uci-uci1--dc-lax-agg6-egm.cenic.net (137.164.24.42)  23.856 ms  24.259 ms  24.261 ms
12 msd-core--cpl-core.ucinet.uci.edu (128.195.248.250)  18.776 ms  18.656 ms  18.152 ms
14 moo.nac.uci.edu (128.200.34.95)  19.151 ms  19.084 ms *
SSH & The Shell

- Once logged in to HPC via SSH you are now using the Shell, which is..
- A program that intercepts and translates what you type, to tell the computer what to do.
- It is what you will be interacting with mostly.
- HPC shell is 'bash', altho there are others.
- **bash** is a full programming language, but a very poor one. You'll need to learn Perl or Python.
Learn the bash shell.

• If you don't learn bash, life will be difficult on Linux. Google is your friend, even if bash is not. Before you submit anything to the cluster via `qsub`, get it going in your login bash shell with a small amount of data..

• You're welcome to start big jobs in on the IO node, type: `qrsh`. But:

**DO NOT RUN JOBS ON THE LOGIN NODES!!**
How to know if I am on Login Node?

Look at your shell prompt!

- Mon Mar 28 21:05:29 [0.02 0.18 0.23] user@hpc-login-1-2:~
  1 $

- Can also use the command `hostname`
  Mon Mar 28 21:05:29 [0.02 0.18 0.23] user@hpc-login-1-2:~
  1 $ hostname
  hpc-login-1-2.local
Now the Practical Stuff.

- From here on, we'll be covering actual, useful commands and approaches.
- These will be demonstrated by me.
- The tutorial will repeat these with examples, so don't get upset if you don't understand them the first time. I didn't.
- Ask me to clarify if you don't understand a concept. The concepts are important at this time, not the commands.
Command Line Editing

- Since you'll be spending a lot of time fighting with the cmd line, make it easy on yourself.
- Use cmdline editing to edit previous cmds.
- Up/Down arrow keys scroll thru cmd history.
- L/R arrow keys scroll by 1 char
- ^ means [hold down the CONTROL Key]
- ^→ & ^← makes the cursor jump by word (usually)
- **Home, End, Insert, Delete** keys work (except Macs lack **Delete** keys (because ... Steve Jobs)
- ^u kills from cursor left; ^k kills from cursor to right
- Use **Tab** for *auto complete*. 
• **THIS IS IMPORTANT**
• STDIN is usually the keyboard, but...
• STDOUT is usually the screen, but...
• STDERR is *also* usually the screen, but...
• All can be redirected all over the place
• to files, to pipes, to FIFOs to network sockets
• can be combined, split (by 'tee'), spawned into subshells to make simple workflows
• More on this later.
• Files & Directories much like on Mac & Win
• Except...
• Names are case-sensitive, 256 char long
• 'Folders' → 'Directories', separated by '/'
• No spaces in names*
• [...] means 'in this dir'
• [...] means parent dir
• [...] means 'home dir'
• A leading '/' means 'from the root dir'
Foreground & Background Jobs

- Foreground (fg) jobs are connected to the terminal. You kill a fg job with ^C.
- Background (bg) jobs have been disconnected from the terminal and are running in the bg.
- Send a job to the bg immed. by appending &
- Recall a job to the fg with fg.
- Send a fg job to the bg with ^z (suspend), then 'bg'.
- All jobs started in the terminal are killed when you log out. (usually)
Pipe |

- Works with STDIN/OUT/ERR to create 'pipelines'
- Very similar to plumbing; can add 'tee's to introduce splits.
  
  $ ls | tee 1file 2file 3file | wc

- STDOUT of one program goes to the STDIN of another command whose STDOUT goes to the STDIN of another program ad infinitum.
- Soooooo......
Pipe Example

```
w|cut -f1 -d' '|egrep -v "(^$|USER)"|sort|uniq -c|wc
```

- `w` spits out who is on the system right now
- `cut -f1 -d' '` chops out the 1st field (the user), based on the space token
- `egrep -v "(^$|USER)"` filters out both blank lines and lines with 'USER'
- `sort` sorts the usernames alphabetically
- `uniq -c` counts the unique lines
- `wc -l` word-counts that output.

Example: Now on HPC!
Help on Commands

- `cmd -h`
- `cmd --help` or `cmd --help`
- `man cmd`
- And ..... Google...
• Use specific terms to narrow search
• ‘Linux’ ‘bash’ ‘nano’ ‘Perl’
• include ‘examples’ (!)
• StackOverflow, ServerFault
• BioStars, SeqAnswers
Some Useful Commands

- `ls [many options]` = list files
- `cd [up or down]` = change directory
- `find [from] -name [name]` = find files
- `locate [name]` = where is this file?*
- `tree [options]` = show the dir tree
- `file [name(s)]` = what is this?
- `du -h` = disk usage
- `df -h` = disk free
- `less [names]` = view files read-only
- `cols [file]` = view file in columns
Creative / Destructive Commands

- `mkdir [name]` – make a dir
- `rmdir [name]` – remove a dir
- `mv [from] [to]` = move or rename
- `cp [from] [to]` = copy file(s)
- `rm [file]` = delete file(s)
- `wget [URL]` = get a file from the Internet
- `curl -O [URL]` = ditto, but on steroids
More Useful Commands

• `mc` = Midnight Commander
• `[ah]top` = top CPU-using processes
• `time [command]` = how long does it take?
• `[aef]grep [regex] [files]` = find regex* in files
• `cat [files]` = print the files to STDOUT
• `head/tail [files]` = dump the top / bottom of files
Regular Expressions

• Among the most powerful concepts in pattern matching
• Simple in concept, NASTY in implementation
• Among the ugliest / most confusing things to learn well
• But pretty easy to learn the simple parts.
• You will NEED to learn it – it's central to computers and especially biology
Simplest form is called *globbing*: `a*`
Barely more complicated: `a*.txt`
A bit more: `a*th.txt`
Can be MUCH more complex:
- `[aeiou]` = any of 'aeiou'
- `F{3,5}` = 3-5 'F's
- `H+` = 1 or more 'H's
- `.` = any character
Also classes of characters (#s, alphabetic, words)
Archiving / Compression

- tar = std archive format for Linux [example]
- zip = common archive format, from Windows
- gzip/ungzip = common compressed format
- bzip2/bunzip2 = another compressed format
- pigz = parallel gzip (for large files)
- pbzip – parallel bzip2 (ditto)
Customize Your Environment

• (or don't.. your choice)
• Set aliases *(alias nu="ls -lt |head -22")*
• Set Environment Variables *(export EDITOR=vim)*
• Change your bash behavior via **shopt** (Google for how)
• Make these permanent via **.bash_profile** & **.bashrc** files in your **home directory (~)**
Editing Files: simple & complex

- Text-based:
  - nano, joe, vi/vim, emacs

- GUI-based:
  - gedit, nedit, kate, jedit, emacs

(choose one and learn it, well!)
Move Data to / from HPC

- Covered in detail in HPC USER HOWTO, which references: [http://goo.gl/XKFEp](http://goo.gl/XKFEp)

- `scp` on Mac, Linux

- **WinSCP, Filezilla, CyberDuck, FDT** on Win

- Everyone should know how to use `rsync`. Not the easiest to learn, but very powerful & scriptable.

- rsync GUIs for Linux, Windows, MacOSX
Checksums

- They work. Choose one and use it.
- md5sum / hashdeep / shasum
- Use **MANIFEST** files & copy them along with the data files.
- See Checksum example
- http://goo.gl/uvB5Fy
3 main sets of programs

• Your personal set (typically in ~/bin)

• The default system utilities
  cut, grep, ls, mv, cp, rm, cd, etc...

• The module system programs
The Module System

- `[module avail]` shows all installed software
- `[module load R/3.01]` loads program R version 3.01 (but doesn't start it)
- `[module unload]` unloads the specified program
- `[module purge]` removes all loaded modules
- `[module list]` lists all the currently loaded modules
How to Find Software

- na\(\text{tab}\)\(\text{tab}\) → name
- `yum search <search term>`  # CentOS
- `module avail` (will dump all modules)
- `searchmodules` (modules & Perl, Python, R)
- Google
- Ask us.
The Grid Engine Scheduler

- Just another program that juggles requests for resources
- Make sure the submitted program is working on a small set of test data on an interactive shell.
- Need a short bash script (aka `qsub script`) to tell the GE what your program needs to run.
- Can improve the performance of your program in a variety of ways (staging data, running in parallel, using array jobs, etc)
The Scheduler: qsub vs qrsh

- *qrsh* will log you into an Interactive Node (IO Node) where you can test out your scripts.
- IO Nodes are useful for GUI programs (X11 & x2go needed) or testing/running SMALL and FAST data sets.
- A *qsub* script is just a series of bash commands that sets up your resource requirements, PATHs, executes your jobs, and does the post-processing.
- **NO USER INVOLVEMENT** during the process.
GE Useful Commands

• `qstat` - Queue Status
• `queue / q` – What queues you have access to
• `qdel` – Delete/Stop your job
• `qhost` – Show all nodes and their status

• Use `man cmd` to find out more information on above

Ref:
• [http://hpc.oit.uci.edu/running-jobs](http://hpc.oit.uci.edu/running-jobs)
Controlling SGE

- Visit:

- Ref:

- Some useful SGE Parameters:
  - http://goo.gl/hrcXBg
SGE: Request Node Resources

• Use Case: You know your program requires at least
  24GB Memory
  16 CPU Cores
You need to tell the scheduler
#$ -l mem_free=24G
#$ -pe openmp 16
This does not make your program run faster or use all cores – you simply reserve this amount
SGE: Queues

- As you noticed, the scheduler uses queues to slot your job into available queues.
- Some queues have higher priority than others.
- Type `queue` or `q` to see what you have access to.
- You specify these Qs in your qsub script with:
  
  ```bash
  #SBATCH -q som,asom,free*
  ```
  
  (note that you can use globs (free*) to specify Qs)
• The free* queue allows anyone to use CPU cycles when they are not in use on any queue/node cluster wide

• When owners want to use their nodes, free* jobs are suspended

• Similar to the free* queue, the 'all' queue is group-specific: abio, asom, etc.
Follow Along

• Take a few moments to login to cluster and follow along if you want.

• After logging in, follow me on screen

• Ref:
  • http://moo.nac.uci.edu/~hjm/biolinux/Linux_Tutorial_12.html