Intro to Linux on the HPC cluster

Harry Mangalam

harry.mangalam@uci.edu
Some talking points..

• You've heard of Linux...? (~ Darwin/BSD, very unlike Windows, but .. Cygwin!)
• You're interested & bright
• You can Google, and read further by yourself.
• Questions, **ASK THEM**, but I may not answer 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 – Take Away

- 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 <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, Garr, Joseph, Edward
• 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 <pastie.org> or <tny.cz>)
- Use text, not screenshots if possible.
Bad Question

Why doesn’t “X” work?
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 /bio/joeuser/RNA_Seq_Data/ 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
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

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

What is the High Performance Computing Cluster?

and...

Why do I need HPC?
What is a Cluster?

- Pod of large general purpose computers
- run the Linux Operating System
- linked by some form of networking
- have access to networked storage
- can work in concert to address large problems
- by scheduling jobs very efficiently
What HPC is

- ~7400 64b Cores – Mostly AMD, few Intel
- ~44TB aggregate RAM
- ~1.3PB of storage
- Connected by 1Gb ethernet (100MB/s)
- Connected by QDR IB (4000MB/s)
- Grid Engine Scheduler to handle Queues
- > 1950 users, 100+ are online at anytime
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
DATA IS NOT BACKED UP

• NO DATA IS BACKED UP – WHAT. SO. EVER. -- Lobby your PIs to get us more $ if you want this.

• Most data is stored on RAID6

• BUT! Any of that can disappear at any moment. It has.

• IF IT'S VALUABLE, back it up elsewhere --- or the code that generated it.
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 another filesystem.
HPC FileSystem Layout

Orange – Cluster Wide
Black    – Node Specific

/
├── data/    NFS Mount
│   ├── apps  All Programs are installed here
│   +── users Users home directory
│      └── pub/ Public scratch space, overflow
│         └── bio/ Space for BIO group → /dfs1
│         └── som/ Space for SOM group → /dfs1
│         └── cbcl/ Space for CBCL group → /dfs1
│      └── dfs1/ BeeGFS Distributed File System ~460TB
│      └── dfs2/ BeeGFS Distributed File System ~190TB
│      └── scratch Node-specific temporary storage per job (faster than all above) ~1TB – 14TB
│      └── fast-scratch High Speed Fraunhofer FileSystem for temporary storage - 13TB
│      └── ssd-scratch Very High IOPS for DB, other jobs.
│      └── /tmp Same as scratch
Disk Space / Quotes / Policies

- You can only have so much space
- 50GB for /data/ (home directory)
- if 6 months or older without use – please remove from cluster
- More for Condo owners or Groups who have bought extra disk space.
- Regardless, NO DATA IS BACKED UP
Data Sizes

• Your data will be BIG – “BigData”
• BigData is somewhat 'dangerous' due to its bigness.
• Think before you start. You can't predict everything, but you can predict a lot of things – more on this later
Network Considerations

From Cox.net ↔ UCI

$ 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
20.660 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 *
Network Considerations

Inside UCI (moo ↔ HPC)

$ 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
How to: Login with SSH

• 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**....

• 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 and in which direction the bytes are going.
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.
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.

You know about 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
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.

What you will be interacting with mostly.

HPC shell is *bash*, altho there are others.
Learn the shell, or else.

• If you don't get along with the shell, life will be hard.

• Before you submit anything to the cluster via qsub, get it going in your login shell.

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

• “DO NOT RUN JOBS ON THE LOGIN NODE”
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
Command Line Editing

- Since you'll be spending a lot of time fighting with the cmd line, make it easy on yourself.
- Learn cmd line editing to edit previous cmds
- Up/Down arrow keys scroll thru cmd history
- L/R arrow keys scroll by 1 char
- $^\text{u}$ means CONTROL Key
- $^\text{ }$ makes L/R arrow jump by a word (usually)
- Home, End, Insert, Delete keys work (except Macs lack 'Delete' keys (because … Steve Jobs))
- $^\text{u}$ kills from cursor left; $^\text{k}$ kills from cursor to right
- Tab for auto complete
STDIN, STDOUT, STDERR

- 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'), to make simple workflows
  - More on this later.
File & Directories

- 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'
Forefront & 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`
- `info cmd` (but you hope not)
- And ..... Google...
Some Useful Commands

- `ls [many options] = list file`
- `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 = disk usage`
- `df = disk free`
- `less [names] = view files`
- `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 toSTDOUT
- `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
Regexes

- 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
- zip = common archive format, from Windows
- gzip/unzip = common compressed format
- bzip2/bunzip2 = another compressed format
- pigz = parallel gzip (for large files)
- pbzip – parallel bzip2 (ditto)
Customize Your Environment

- 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: goo.gl/XKFEp

- scp, bbcp, netcat/tar 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
Using Software on HPC

• 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 programs
- **[module list]** lists all the currently loaded ones
How to Find Software

- **locate** `<partial search term>`
- **apropos** `<search term>`
- **na<tab><tab>** → name
- **yum search** `<search term>`  # CentOS
- **module avail** (will dump all modules)
- Google
- Ask us.
The Scheduler (GE)

- Just another program that juggles requests for resources
- Make sure a 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
- [http://hpc.oit.uci.edu/PHPQstat](http://hpc.oit.uci.edu/PHPQstat)

- Ref:
  - [http://hpc.oit.uci.edu/running-jobs](http://hpc.oit.uci.edu/running-jobs)
  - [http://hpc.oit.uci.edu/PHPQstat/](http://hpc.oit.uci.edu/PHPQstat/)
Sample QSUB Script

- Visit:

- Ref:
  - <http://goo.gl/hrcXBg>
GE – 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
GE – Queues

• As you noticed, the scheduler uses queues to handle your job.

• 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
  #$ -q som,free*
  ```
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.
Big Data

- Volume
- Scary sizes, and getting bigger

- Velocity
- Special approaches to speed analysis

- Variety
  - Domain-specific standards (HDF5/netCDF, bam/sam, FITS), but often aggregations of unstructured data

- BigData Hints for Newbies
  <http://goo.gl/aPj4az>
# Big Data – How Big is Big?

<table>
<thead>
<tr>
<th># Bytes</th>
<th>Byte name / Abbrev’n</th>
<th>Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>bit (b)</td>
<td>0 or 1: the smallest amount of information.</td>
</tr>
<tr>
<td>1</td>
<td>Byte (B)</td>
<td>8 bits, the smallest chunk normally represented in a programming language.</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>1,024 B (1 KB)</td>
<td>a short email is a few KBs</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>1,048,576 B (1 MB)</td>
<td>a PhD Thesis; Human Chr 1 is ~250 MB</td>
</tr>
<tr>
<td>$2^{30}$</td>
<td>1,073,741,824 B (1 GB)</td>
<td>the Human Genome is 3,095,693,981 B (optimized, ~780 Mb @ 2b/base); a BluRay DVD holds 25GB per layer (most movie BluRays are dual-layer = 50GB); a Genomic bam file is ~150GB</td>
</tr>
<tr>
<td>$2^{32}$</td>
<td>4,294,967,296 (4GB)</td>
<td>fuzzy border between SmallData (32b) and BigData (64b)</td>
</tr>
<tr>
<td>$2^{40}$</td>
<td>1,099,511,627,776 B (1 TB)</td>
<td>1/10th Library of Congress (LoC); the primary data fr. an Illumina HiSeq2K is ~5 TB</td>
</tr>
<tr>
<td>$2^{50}$</td>
<td>1,125,899,906,842,624 B (1 PB)</td>
<td>100X LoC; ~HPC’s aggregate storage; ~100 PB is the yearly storage requirements of YouTube.</td>
</tr>
<tr>
<td>$2^{60}$</td>
<td>1,152,921,504,606,846,976 B (1 EB)</td>
<td>the est. capacity of the NSA’s data facility is ~12 EB</td>
</tr>
</tbody>
</table>
## Integer Byte Sizes

<table>
<thead>
<tr>
<th>word size</th>
<th>#bits</th>
<th>range of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte or char</td>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>int</td>
<td>16</td>
<td>65,536</td>
</tr>
<tr>
<td>long int</td>
<td>32</td>
<td>4,294,967,296</td>
</tr>
<tr>
<td>long long int</td>
<td>64</td>
<td>1.84467440737e+19</td>
</tr>
</tbody>
</table>
Inodes and ZOT Files

- Inodes contain the metadata for files and dirs
- Inodes are pointers to the data
- Regardless of size, a file needs at least one inode to locate it.
- A file of 1 byte takes up the same minimum inode count as a file of 1TB
- DO NOT USE ZOTFILES!! – Zillions of Tiny Files
• Use format-specific utilities to view such files and hash values to check if they’re identical to what they should be.

• Try not to be the person who tries to open a 200GB compressed data file with nano/vim/joe/emacs, etc.
[De]Compression

- If your applications can deal with compressed data, KEEP IT COMPRESSED.
- If they can't, try to use pipes (|) to decompress in memory and feed the decompressed stream to the app. Many popular apps now allow this.
- Use native utilities to examine the compressed data (zcat/unzip/gunzip, grep, archivemount, Vitables, ncview, etc.)
Moving BigData

• 1st: Don't.
• Otherwise, plan where your data will live for the life of the analysis, have it land there, and don't move it across filesystems.
• Don't DUPLICATE DUPLICATE DUPLICATE DUPLICATE BigData
• See: <http://goo.gl/2iaHqD>
  • rsync for modified data
  • bbcp for new transfers of large single files, regardless of network
  • tar/netcat for deep/large dir structures over LANs
  • tar/gzip/bbcp to copy deep/large dir structures over WANs
They work. Choose one and use it.

- `md5sum / jacksum / shasum`

- Use MANIFEST files & copy them along with the data files.

- See Checksum example

- `http://goo.gl/uvB5Fy`
Processing BigData

- Files (HDF5, bam/sam) and specialized utilities (nco/ncview, [Py/Vi]tables, R, Matlab)
- Relational Dbs (SQLite, Postgres, MySQL)
- NoSQLs (MongoDB, CouchDB)
- Binary Dumps (Perl's Data::Dumper, Python's pickle)
- Non-Storage (pipes, named pipes/FIFOs, sockets)
- Keep it RAM-resident.
• HPC is not backed-up.

• Cannot tolerate old, unused BigData.

• RobinHood is looking for your old BigData.

• Please help us by doing your own data triage.

• Ask your PIs to bug our boss to provide more resources so we can provide more resources.
Remember...

Good Judgment comes from Experience

Experience comes from Bad Judgment
Follow Along

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

- After logging in, follow me on screen

- Ref: