BioLinux on HPC

Bio: Jenny Wu
<jiew5@uci.edu>

Linux: Harry Mangalam
<harry.mangalam@uci.edu>
Good Judgement comes from Experience

Experience comes from Bad Judgement
Good Judgement comes from Experience.

Experience comes from Bad Judgement.
Or...

Some good comes from all catastrophes.

Learn & Move on.
Some comments

- I assume you're a novice at Linux..
- .. & Bioinformatics
- We want to address both
- Without making you a mess.
- I speak too fast; let me know when I do.
- The **Unknown Unknowns** problem.
- Questions, please, but I may not answer them immediately
Philosophy about computing

- Be lazy
- Copy others
- Don't invent anything you don't have to
- Re-use, re-cycle, DON'T re-invent
- Resort to new code only when absolutely necessary.
For Biologists

• You're not CS, not programmers
• Don't try to be them
• But! Try to think like them, at least a bit
• And some concepts and abilities are important
Useful Concepts

- LEARN HOW TO GOOGLE (see *Fix IT Yourself with Google* in the resources).
- Listservs, forums, IRCs are VERY useful for more involved questions
- 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 (learn to use pastie.org)

• Use text, not screenshots if possible.
This is a bad question:

Why doesn't 'X' work?
A good question

I tried running this new module this morning and it looks like I can't get it to launch on HPC and my output files aren't helping me figure out what is wrong.

I am working out of
/bio/abriscoe/RNA_Seq_Data/M_sexta_RNAseq
And the qsub script is 'job12.sh'

When I submit the job, it appears to go thru the scheduler but then dies immediately.

I can't find any output to tell me what's wrong.
Linux Resources

- GOOGLE → Forums, Lists & List Archives, IRCs
- The HPC HOWTO <goo.gl/kzlql>
- Software Carpentry
- Showmedo.com
- Us – Jenny & Harry
- Please ask questions that are answerable.
What is a cluster?

- bunch of big general purpose computers
- running Linux
- linked by some form of networking
- have access to networked storage
- that can work in concert to address large problems
- by scheduling jobs very efficiently
HPC Specifically

- ~2500 64bit compute cores
- ~14TB aggregate RAM (fast silicon memory)
- 1/2 PB of storage (1000x slower than RAM)
- connected by 1Gb ethernet (100MB/s), DDR (400MB/s), QDR Infiniband (800MB/s) (per channel)
- uses the Grid Engine scheduler to handle Queueing
- >600 users, of whom 20-100 are online at any time
HPC is NOT

• Your personal machine.
• It's a shared resource
• Pretty well protected against mischief and disaster
• But don't take that as a challenge
• !! → Data is NOT backed up. ← !!
Data Sizes

- Especially with NGS techniques, you'll be crossing the line into BIG DATA.
- Big Data is somewhat dangerous due to its bigness.
- Think before you start. You can't predict everything, but you can predict a lot of things.
NO BACKUPS

- Data on HPC is **not backed up**
- Most data is stored on RAID6 storage.
- **BUT!** Any of that data can disappear at any moment.
- So if it's valuable to you, back it up elsewhere.
Commandline Cons

- The tyranny of the blank page
- No visual clues
- Type vs click
- Have to know what to type
Commandline 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, unless they perfect mind control.
- It's an efficient way of interacting with the computer (which is why it's survived for 50yrs).
- You can use it to create simple, but very effective pipelines and workflows.
The Shell

• 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' (also dash, csh, tcsh zsh)

• A `qsub script` is just a series of bash commands that sets up your resource requirements
Know the shell, Embrace the shell

- 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 login shell, but don't let them run long.
- 'Ctrl+C (^C) kills the job you just started.'
The scheduler (GE)

- Just another program that juggles requests for resources
- Make sure a program is working on a small set of test data.
- 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)
A simple qsub script

#!/bin/bash
# Usage: sleeper.sh [seconds]
#        default for time is 60 seconds
#$ -N Sleeper1
#$ -S /bin/bash
# Make sure that the .e and .o file arrive in the working directory
#$ -cwd
# Merge the standard out and standard error to one file
#$ -j y
/bin/echo Here I am: `hostname`. Sleeping now at: `date`
/bin/echo Running on host: `hostname`.
/bin/echo In directory: `pwd`
/bin/echo Starting on: `date`
#$ -m be
#$ -M hmangala@uci.edu
time=60
if [ $# -ge 1 ]; then
time=$1
fi
sleep $time
echo Now it is: `date`
Solving Problems

- Reduce the scope of the problem
- What in particular is failing?
- Debug in the *login shell* rather in qsub shell as long as possible.
- Things will start faster and fail faster in the login shell.
- (almost) anything in a qsub script can be pasted into a bash shell and have the same effect.
- Think of your login shell as your home and the cluster as a slightly sketchy bar.
Foreground & background jobs

- Foreground (fg) jobs are connected to the terminal
- Background (bg) jobs have been disconnected from the terminal.
- Send a job to the bg by appending '=&'
- Recall a job to the fg with 'fg'.
- Send a fg job to the bg with '^z', then 'bg'
Screen & Byobu

- If you need to maintain a live connection for some reason, use 'byobu'.
- It calls 'screen' and allows you to multiplex and maintain connections.
- Somewhat unintuitive interface but very powerful.
x2go

- 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.
- x2go is described in the Tutorial & HOWTO.
Commandline Editing

• Since you'll be spending a lot of time fighting with the cmdline, make it easy on yourself.
• Learn cmdline editing to edit previous cmds
• Up/Down arrow keys scroll thru cmd history
• L/R arrow keys scroll by 1 char
• ^ makes L/R arrow jump by a word
• Home, End, Insert, Delete keys work (except Macs lack 'Delete' keys)
• ^u kills from cursor left; ^k kills from cursor to right
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, combined, split (by tee), etc
• More on this later.
Pipe = |

• Works with STDIN/OUT/ERR to create 'pipelines'

• Very similar to plumbing; can add 'tee's to introduce splits

• 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 ' ' | sort | egrep -v "(^$|USER)" | 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

.sort sorts the usernames alphabetically

.egrep -v "(^$|USER)" filters out both blank lines and lines with 'USER'

.uniq -c counts the unique lines

.wc word-counts that output.
Here's another

```
$ qhost | scut -f=7 | tr -d /G/ | stats
```

- **qhost**: Grid Engine utility that shows host status
- **scut -f=7**: extracts the 8th col*, separated by whitespace
- **tr -d /G/**: deletes all instances of 'G'
- **stats**: calculates descriptive stats of all STDIN

* computers (often) annoyingly count from 0, so the 8th column is #7
Files & Directories

• Files & folders 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 'home dir'
• A leading '/' means 'from the root dir'
/  
├── bin  critical executables  
├── boot  kernel image and init files  
├── dev  device file  
├── etc  config files  
├── home  usually where your files live  
├── lib  critical library files  
├── lib32  32bit libs  
├── lib64  64bit libs  
├── lost+found  what it sounds like  
├── media  where removable disks get mounted  
├── mnt  where temporary other devices devices get mounted  
├── opt  optional package installs  
├── proc  process tracking dir, system config files  
├── root  home for the root user  
├── run  keeps track of running processes (locks, IDs)  
├── sbin  system binaries  
├── selinux  ugh. Secure linux config (usually empty on a usable system)  
├── srv  service-specific files (some distros)  
├── sys  system-specific files (some distros)  
├── tmp  where anyone can write temporay files  
├── usr  most of the system files live here  
└── var  'varying' files for keeping track of various system processes.
How to use commands

- 'cmd -h'
- 'cmd --help'
- 'man cmd'
- 'info cmd' (but you hope not)
- And ..... Google...
Finally, 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 directory 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 informational cmds

- `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
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)
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.
• But you NEED to learn it – it's central to computers and especially biology
Regexes

- Simplest form is called globbing (a*)
- Mix it up (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)
Editors: simple $\rightarrow$ complex

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

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

(choose one and learn it)
Customize Your Environment

• Change your prompt to something useful to you (and to us):
  • `PS1="\n\t\u@\h:\w\n! \$ "`

• Set aliases (`alias nu="ls -lt |head -22"`)  

• Set Environment Variables (`export EDITOR=joe`)
Disk Quotas

- Unlike BDUC, HPC enforces disk quotas
- You can only have so much space.
- 20GB for most users
- More for Condo owners or groups who have bought extra disk space.
- AGAIN: the fact that you are allowed 20 GB or 200GB does not mean that it's SAFE. It is not.
Moving 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.
- rsync GUls for Linux, Windows, MacOSX
Programs, finally

- 3 main sets of programs
- Your personal set (later)
- The default system utilities
- The module system programs
How to find them

- `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.
When (not if) it fails

- prog -h
- prog --help
- prog -?
- man prog
- info prog
- Google
Resources

• Please see the Resource List at the end of the tutorial.