HPC Overview

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2017

Outline

http://pj.freefaculty.org/guides/Computing-HOWTO

Outline

This is an Overview

- Prepare you to work on the “compute cluster”, the KU Community Cluster, a network of Linux computers (“nodes”).
- Linux will be unfamiliar to some new students, but it is vital to overcome fear/anxiety about the “command line”

What is the Purpose? I

- Launch “long running” programs
- Launch programs that divide their work among many separate computers
- Run simulations that might take months or years in your PC.

Outline

Get some Linux Practice I

- Get some practice with Linux.
  - Become familiar with the command line
  - If you have a Mac, you are already running a Unix system :). Utilities -> Terminal
  - In Windows, install Git BASH or Ubuntu BASH Shell for windows
  - Log in on somebody else's Linux system, which probably has a Graphical User Interface.
- Try some of the simple commands. See the "linux-help" page on the CRMDA website.

Jump in with both feet

- Consider installing Linux on your system, or in an external disk drive
  - If you are indifferent between Linux distributions, choose Centos or Fedora Linux because they are most similar to the Cluster.
  - Perhaps Ubuntu Linux is more "user friendly." It is *certainly* easier to configure proprietary video drives in Ubuntu.
- Or consider Virtual Machine. We have good experience with Oracle VirtualBox

Outline

Cluster Transition

- CRMDA had its own cluster, 60 compute nodes, 8 cores each
- In 2015, we transitioned into the Advanced Computing Facility cluster at KU
- In February, 2017, KU re-organized that into the Center for Research Computing cluster

Instructions needed re-writing

- `http://crmda.ku.edu/computing` under revision
- CRMDA timeline blog has updates that are not in the Web pages yet
- If our pages tell you to do something, and it does not work, there's a reasonably good chance we are telling you the wrong thing☹

Outline

The Cluster "login" node

- Secure Shell (SSH) is required.
- In Linux or Mac OSX open a terminal and run

```
ssh username@hpc.crc.ku.edu
```

- On Windows, install an “ssh capable” terminal program, such as “Putty”.
- These will put you into “the command line” on the “login node” (AKA “head node”) of the cluster.
- Tip: `hpc.crc.ku.edu` is a “load balancing” login that points your connection at either of 2 login nodes, `submit1.hpc.crc.ku.edu` and `submit2.hpc.crc.ku.edu`.

I Logged In. See? I

```
$ ssh hpc.crc.ku.edu
Last login: Tue Aug 15 12:24:28 2017
*****************************************************************
                                NOTICE TO USERS

Access to electronic resources at the University of
Kansas is restricted to employees, students, or
individuals authorized by the University or its
affiliates. Use of this system is subject to all
policies and procedures set forth by the University
located at www.policy.ku.edu. Unauthorized use is
prohibited and may result in administrative or
legal action. The University may monitor the use of
this system for purposes related to security management,
system operations, and intellectual property compliance.

*****************************************************************

    Welcome to the University of Kansas Community Cluster
      Send questions, problems, etc to crchelp@ku.edu
```

I Logged In. See? II

```
  Check out the Center for Research Computing website for
      more information on usage of the cluster

      https://crc.ku.edu

*****************************************************

    Upcoming Maintenance (all time CDT):

Start                End                  Reason       Affects
______________       ______________       ________
    _____________

08:00 September 19   08:00 September 20   maintenance   all
    nodes

*****************************************************

Storage groups you have access to:
____________________________________

crmda (Primary)
```

I Logged In. See? III

```
Queues you may submit to:
-------------------------

crmda (Default)
sixhour

Your Storage variables and Quotas:
----------------------------------

$HOME = /home/pauljohn

    <GB> <soft> <hard> : <files> <soft> <hard> : <path to
        volume> <pan_identity(name)>
   33.46   85.00 100.00 :    98538   85000 100000 : /home/
       pauljohn uid:142424(pauljohn)

$WORK = /panfs/pfs.local/work/crmda/pauljohn
Filesystem
Size Used Avail Use% Mounted on panfs://pfs.local/work
14T  6.3T  7.4T  47% /panfs/pfs.local/work/crmda/pauljohn

$SCRATCH = /panfs/pfs.local/scratch/crmda/pauljohn
```

I Logged In. See? IV

```
Filesystem
Size  Used  Avail  Use% Mounted on panfs://pfs.local/scratch
55T   35T    20T   65% /panfs/pfs.local/scratch/crmda/pauljohn

$CRMDA = /panfs/pfs.local/work/crmda
```

Memorize some things, Look up others. I

I have memorized these, further discussed in Terminal-1 lecture notes

```
$ pwd                       # print working dir
$ ls                        # list files
$ mkdir                     # make a directory
$ cd                        # change directory
$ rm fn                     # erase fn
$ mv fn somewhere           # move fn somewhere
$ cat fn                    # print out fn on screen
$ nano fn                   # use "nano" editor
$ grep "poop" *             # find "poop" in *
```

The Module system

- Software is provided in modules: choose what you need.
  module avail: modules you are allowed to load
  module list: currently loaded modules
  module spider: search for modules you want but are not (yet) allowed to load
  module load: load a module
- In my opinion, this should be easy for novices, we should apply a default set of modules.
- We are developing a system to make that work, but some user effort is required. See my Timeline blog post "R modules: Super Exciting New Updates" (July 24, 2017).
- This will eventually find its way into `http://crmda.ku.edu/acf-modules`, which needs updating

NFS: Networked File System

- Your $HOME folder is shared by all the compute nodes and the head node
- Other folders are shared as well, such as $WORK and $SCRATCH
- So the files you see on one "node" in the system are the same ones you see on any other system
- That's part of the magic of parallel programming–60 computers can read and write in the same area.
- That's also part of the danger: Different nodes writing on same file might cause failure

HOME, WORK, SCRATCH I

- The new CRC cluster changed some policies about file storage
- User home folders are not backed up, it is not recommended to do important work there.
- Simulations and other "real work" should be done in the $WORK folder assigned for each user.
- Computations that generate a lot of temporary files can be run in the $SCRATCH folder.

If you have X11. . . I

- X11 displays "windows" launched by remote system
  - If your workstation has X11, the cluster can display windows "onto" your desktop
- Easier on Linux and Macintosh (get Xquartz), tougher (but possible) in Windows
- If you have X11, then log in by telling your ssh program that you want to use X11 to "relay" GUI to your desktop:

```
$ ssh -X username@hpc.crc.ku.edu
```

Changing Remote Desktop Options I

- We enjoyed using NoMachine with the cluster, but the CRC elected to discontinue that service
- A new service, called Viewpoint, is a Web browser portal `https://crc.ku.edu/using-hpc`

Fear the Head Node

- You log in to hpc.crc.ku.edu, but
  - **don't** work there
  - **do** go a "compute node" for an "interactive session" `http://crmda.ku.edu/interactive-session`
- Interactive sessions can be requested in various ways, by typing out a long "msub" command or by using the
  - SHORTCUT: `qxlogin`
- The "x" in there is for "X11 forwarding"

There is a Graphical Interface for Linux

- My Linux Laptop uses the XFCE graphical "desktop environment".
- Cluster nodes
  - don't have the whole desktop environment installed
  - do have individual programs that can "make" windows
- At the current time, the 2 alternatives for displaying those windows are
  - X11 forwarding, and
  - RemoteViz in the View Web portal

Outline

Don't Let an Important Program Be Your First Program

- Everybody makes mistakes. Nothing works the first time.
- Please practice with small programs
  - Don't throw together a 1000 line sequence of commands and expect it to work.
  - Keep separate pieces separate
- Be clear in your purpose in every step

Why Working Examples?

This is what I learned from programmers at Santa Fe Institute and Los Alamos Labs:

- A working program may knit together 5 or 10 separate pieces.
- Failures can occur in any of those separate pieces.
- A WorkingExample is focused effort to explore one piece or the connection between pieces.

`http://pj.freefaculty.org/R/WorkingExamples`
Self explanatory, single purpose R examples.

We are on a Leading Edge

- "High Performance Computing" is in rapid development
- "We" are all learning how to run programs in that environment.
- If things don't work, don't panic, we can often find solutions
- If things don't work, please read error messages and file good bug reports.

Working Examples For Batch Computing

We recently relocated our repository from an SVN server to GitLab, which offers a web view of the project:
`https://gitlab.crmda.ku.edu/crmda/hpcexample`

Don't copy individual files in your browser.

Use a Git client to retrieve everything.

More about Git `https://crmda.ku.edu/git-help`

Step 2: Get to a "compute node"

1. log in hpc.crc.ku.edu.
2. Run either “qlogin” or “qxlogin”, depending if your workstation has X11.
3. qxlogin example

```
$ qxlogin
qsub: waiting for job 41452764.sched to
    start
qsub: job 41452764.sched ready
```

Step 3: Create a folder for a Working Copy ("Sandbox")

1. At the command prompt, make a practice directory

```
$ mkdir GIT
```

2. Make sure that new directory got created

```
$ ls -la
```

3. Change into that directory (so it becomes your "working directory")

```
$ cd GIT
```

4. Make sure you really are there!

```
$ pwd
$ ls -la
```

Step 4: Get a "Sandbox" (Working Copy) of the hpcexample Repository I

1. Run this command to download a copy of the repository

```
$ git clone https://gitlab.crmda.ku.edu/
    crmda/hpcexample.git
Cloning into 'hpcexample'...
remote: Counting objects: 1559, done.
remote: Total 1559 (delta 0), reused 0 (
    delta 0)
Receiving objects: 100% (1559/1559), 2
    .46 MiB, done.
Resolving deltas: 100% (896/896), done.
```

Overcome your fears I

1. change into the directory

```
$ cd hpcexample
```

2. List what you have:

```
$ ls
```

I see this:

Overcome your fears II

```
$ ls
00-README-ROADMAP/
      Ex56-MISimulation-ManySerial/
Ex01-ShellScript/                   Ex57-MISimulation-RMPI/
Ex05-Mplus-1/                       Ex60-HelloWorldSnow/
Ex08-MplusRunall-1/                 Ex61-HelloWorldSnowFT/
Ex09-MplusRunall-2/                 Ex65-R-parallel/
Ex11-Mplus-ManyInpFiles/            Ex67-SOCK-Cluster/
Ex20-SAS-1/                         Ex70-doMPI-sinc/
Ex30-BashScriptinParallel/          Ex75-corrSim-doMpi/
Ex41-HelloWorldinParallelC/         Ex76-corrSim-notParallel/
Ex50-R-serial/                      Ex80-PrevSci2007/
Ex51-R-ManySerialJobs/              Ex81-ParallelSeedPrototype
      /
Ex52-R-JobArray/                    README.md
Ex53-HelloWorldRmpi/
```

3. Obliterate a directory. Pretend there is a user error.

```
$ rm -rf Ex50-R-serial
```

Overcome your fears III

4. Make sure Ex50-R-serial is gone

```
$ ls
00-README-ROADMAP                    Ex57-MISimulation-RMPI
Ex01-ShellScript                     Ex60-HelloWorldSnow
Ex08-MplusRunall-1                   Ex61-HelloWorldSnowFT
Ex09-MplusRunall-2                   Ex65-R-parallel
Ex15-Mplus-ManyInpFiles
     Ex66-ParallelSeedPrototype
Ex30-BashScriptinParallel            Ex70-doMPI-sinc
Ex41-HelloWorldinParallelC           Ex75-corrSim-doMpi
Ex51-R-ManySerialJobs                Ex76-corrSim-notParallel
Ex53-HelloWorldRmpi                  Ex80-PrevSci2007
Ex56-MISimulation-ManySerial
```

5. To recover that directory from Git, do this

```
$ git checkout -- Ex50-R-serial
```

After that, “voila”, everything is back:

Overcome your fears IV

```
$ ls Ex50-R-serial/
README.md                  Rsimple.o-example
    sub-serial.sh
Rsimple.e-example   r-serial.R
    testGraph-example.pdf
```

Git is a "distributed file system". There is a hidden directory, .git, in which all of the contents of the repository, and its history, are kept:

Overcome your fears V

```
$ ls -la .git
total 896
drwxrwxr-x  8 pauljohn pauljohn_g  4096
   Aug 21 17:57 ./
drwxr-xr-x 27 pauljohn pauljohn_g  4096
   Aug 21 17:57 ../
-rw-rw-r--  1 pauljohn pauljohn_g   346
   Aug 21 17:56 FETCH_HEAD
-rw-rw-r--  1 pauljohn pauljohn_g    23
   Aug 21 17:54 HEAD
-rw-rw-r--  1 pauljohn pauljohn_g    41
   Aug 21 17:56 ORIG_HEAD
drwxrwxr-x  2 pauljohn pauljohn_g  4096
   Aug 21 17:54 branches/
```

Overcome your fears VI

```
-rw-rw-r--  1 pauljohn pauljohn_g   273
   Aug 21 17:54 config
-rw-rw-r--  1 pauljohn pauljohn_g    73
   Aug 21 17:54 description
drwxrwxr-x  2 pauljohn pauljohn_g  4096
   Aug 21 17:54 hooks/
-rw-rw-r--  1 pauljohn pauljohn_g 16928
   Aug 21 17:57 index
drwxrwxr-x  2 pauljohn pauljohn_g  4096
   Aug 21 17:54 info/
drwxrwxr-x  3 pauljohn pauljohn_g  4096
   Aug 21 17:54 logs/
drwxrwxr-x  4 pauljohn pauljohn_g  4096
   Aug 21 17:54 objects/
```

Overcome your fears VII

```
-rw-rw-r--  1 pauljohn pauljohn_g   232
   Aug 21 17:54 packed-refs
drwxrwxr-x  5 pauljohn pauljohn_g  4096
   Aug 21 17:54 refs/
```

What Are You Supposed To Find In There?

- Each folder is a more-or-less self contained small example of something
- In each, there SHOULD be
  README
  Example output
- The "public facing" README.md document, which shows at
  https://gitlab.crmda.ku.edu/crmda/hpcexample,
  is a replacement for 00-README-ROADMAP.

Consider Ex65-R-parallel

```
$ ls -la
total 464
drwxrwxr-x  2 pauljohn pauljohn_g  4096 Aug 21 17:54 ./
drwxr-xr-x 27 pauljohn pauljohn_g  4096 Aug 21 17:57 ../
-rw-rw-r--  1 pauljohn pauljohn_g  1625 Aug 21 17:54
    README.md
-rw-rw-r--  1 pauljohn pauljohn_g    33 Aug 21 17:54
    RParallelHelloWorld.e-example
-rw-rw-r--  1 pauljohn pauljohn_g 40115 Aug 21 17:54
    RParallelHelloWorld.o-example
-rw-rw-r--  1 pauljohn pauljohn_g  5298 Aug 21 17:54
    parallel-hello.R
-rw-rw-r--  1 pauljohn pauljohn_g   850 Aug 21 17:54
    sub-hello.sh
```

Consider Ex65-R-parallel

- To go into the cluster's Queue, a program needs 2 things.
  1. A submission script. My habit is to call those "sub-???.sh", here "sub-hello.sh"
  2. The program that the script refers to: `parallel-hello.R`

The Submission Script

```
#!/bin/sh
#
#
#MSUB -M your-name_here@ku.edu
#MSUB -N RParallelHelloWorld
#MSUB -q sixhour
#MSUB -l nodes=1:ppn=11:ib
#MSUB -l walltime=00:05:00
#MSUB -m bea

cd $PBS_O_WORKDIR

## Please check your ~/.bash_profile to make sure
## the correct modules will be loaded with new shells.
## See discussion:
## http://www.crmda.dept.ku.edu/timeline/archives/184
## In a terminal on a compute node, you can test this
## by running
## $ module list
##
## Currently Loaded Modules:
##  1) legacy
```

How to Submit that job

- The cluster administrators prefer if we submit jobs while on the login node
- We requested a special exception so that we can put jobs into the "queue" while we are in the CRMDA computer nodes.
- The program that can put jobs into the queue is now called `msub` (not `qsub` anymore)

```
$ msub sub-hello.sh
```

- After that, there are a number of ways to find out if the job is waiting, or running, or finished. The new program for that is

```
$ showq
```

- However we notice that the older program, `qstat`, also

When that's finished

Output files created are created from every submission, one suffixed "e" and one "o". They will have names like RParallelHelloWorld.e234234 and RParallelHelloWorld.o234234.

Example copies of the files are saved with the directory, "RParallelHelloWorld.e-example" "RParallelHelloWorld.o-example".

- The "e" file: standard error (stderr)
- The "o" file: standard output (stdout)

Before you launch these examples, please:

- Edit the submission script and change the email address to YOUR email:

```
#MSUB -M your-name_here@ku.edu
```

- The default will send you an email when the job starts and ends.
- Maybe you don't want all of that email. It is OK to change this:

```
#MSUB -m bea
```

to this:

```
#MSUB -m ea
```

So that email is sent only when the job `ends` ("e") or `aborts` ("a"), but not when it begins ("b")

Layers of Jargon

- Serial: Run One Self Contained Program at a Time
  - Ex15-Mplus-ManyInpFiles
  - Ex09-MplusRunall-2
  - Ex50-R-serial
  - Ex51-R-ManySerialJobs
- Simpler, easier to write, don't involve parallel programming
- Parallel
  - MPI: Message Passing Interface library. We use "OpenMPI"
  - RMPI: R package accessing the MPI
  - Snow: Simple Network of Workstations: A simplification "layer" that sits "on top" of RMPI.
  - R parallel package: Introduced R-2.14.1.

Outline

How Do You Work On Your R Code?

- You can edit code "on" hpc if you want to
- Especially if edits are simple (config files), this is workable.
- If an X11 server is available, I *do* use Emacs on compute nodes
- For bigger edits, I usually edit in my Workstation
  - Use Git to track files and exchange between systems.
  - Use rsync to transfer files between my system and the cluster
- Sometimes I mount the server file share on my workstation. In my opinion, that's a nice way, but the system administrators do not look kindly upon it.

We have a new suggestion for R parallel

- In hpcexample Ex67-SOCK-Cluster, we have this suggestion:

  Write code in your PC "as if" your PC has a small cluster inside it

- Transfer your R program to the cluster and make "just a few" minor changes so your program interacts with the real cluster.

Outline

Use resources carefully I

- Before launching a large calculation that takes days on many compute nodes, carefully test smaller cases to make sure
  1. the results will be useful
  2. the code is efficient
- Efficiency analysis involves
  - "profiling" to find out where the program spends too much time,
  - conducting test runs and monitoring their resource use, especially memory

Use resources carefully II

- Be careful in your assumptions about the cluster. Some things are SLOW that you expect might be fast (writing lots of small files) and we can suggest ways to do this efficiently.

Learn to Report Bugs

- Nothing works right the first time, but before you report a bug, try to make sure you are not making an obvious mistake.
  - Log out, relax. Perhaps logging in, and re-setting your user environment, will fix it.
- When you can't figure this out, please make good bug reports.
  - What are you trying to do?
  - What commands were run to cause the error?
    - not "something like" what you did. Exactly what you did.
    - a transcript of the full session often helps!
  - What exactly were the errors? (not "something like" ...)
  - Send text, not screenshots, if possible.