Introduction to HPC Resources and Linux

Scientific Computing Consultant
Sharon Solis
Enterprise Technology Services &
Center for Scientific Computing
Elings 3229
swsolis@ucsb.edu

Paul Weakliem
California Nanosystems Institute &
Center for Scientific Computing
e-mail: weakliem@cnsi.ucsb.edu
Elings 3231

Fuzzy Rogers
Materials Research Laboratory &
Center for Scientific Computing
e-mail: fuz@mrl.ucsb.edu
MRL

http://www.ets.ucsb.edu/services/supercomputing

http://csc.cnsi.ucsb.edu

Overview

Most research now involves some form of computing

Often you're solving equations, or analyzing data/doing statistics ('data science').
Engineers often will model a device.

Some specific examples:

- Protein Folding
- Structure of a crystal
- How a molecule behaves on the surface
- How well will an antenna of a certain size and shape work

- Searching for patterns in DNA
- Predicting the spread of wildfires
- Weather prediction

- What factors have a strong influence on poverty

Like many parts of research, you often start small, with a simple idea, but it grows beyond what you (or your computer) can do yourself!

Solutions:

- Better computer
- Multiple computers (HPC)
- Cloud (Can be both of above, with arbitrary size) – somebody else's computer!

Overview

- What is High Performance Computing (HPC)?

  High Performance Computing (HPC) allows scientists and engineers to solve complex science, engineering, and business problems using applications that require **high bandwidth**, **enhanced networking**, and very **high compute** capabilities. From: https://aws.amazon.com/hpc/

- Multiple computer nodes connected by a very fast interconnect
- Each node contains many CPU cores (around 12-40) and 4-6GB/core
- Allows many users to run calculations simultaneously on nodes
- Allows a single user to use many CPU cores incorporating multiple nodes
- Often has high end (64 bit/high memory) GPUs

UCSB provide access and support for multiple HPC resources and educational/training/research support.

HPC is not always the right solution!!!!

- Sometimes you need a fast desktop workstation (if there's lots of interaction with the program, and single runs)
- Sometimes 'Cloud' is the right solution (need 1,000 nodes, but only once every 3 months)
- Sometimes you might even need your own cluster.....
- HIIPPA data

What resources are available at UCSB?

- UCSB Center for Scientific Computing (CSC) HPC clusters
  - Access to all UCSB staff, free (knot) and condo clusters (braid)
  - ***Knot cluster (2011) and Pod Cluster (2018)***
- Nautilus cluster (consumer GPUs) - https://nautilus.optiputer.net/
- Extreme Science and Engineering Discovery Environment (XSEDE) Project funded by NSF. Access to national resources. Free*
- Triton Shared Computing Cluster (TSCC) at San Diego Supercomputing Center (SDSC)
  Mostly used for education/training and class support
- Aristotle Cloud (LSIT) www.aristotle.ucsb.edu
  Local cloud resource, e.g. jupyter hub instances
- Library Collaboratory (2'nd Floor Library/new section)
  Home for data-centric research support (e.g. humanities)
- SCRE
  Secure Compute Research Env.
- Other discipline specific UCSB resources,
  NCEAS, ERI, ECI, your local department

http://csc.cnsi.ucsb.edu/resources

HPC systems at CSC

Campus available cluster Knot (CentOS/RH 6):

110 node, ~1400 core system

4 'fat nodes'(1TB RAM)

GPU nodes (12 M2050's) (now too old)

Campus available cluster Pod (CentOS/RH7):

70 node, ~2600 core system

4 'fat nodes' (1TB RAM)

GPU nodes (3) (Quad NVIDIA V100/32 GB with NVLINK)

GPU Development node (P100, 1080Ti, Titan V)

Published papers should acknowledge CSC

Request access: http://csc.cnsi.ucsb.edu/acct

Condo clusters: (PI's buy compute nodes)

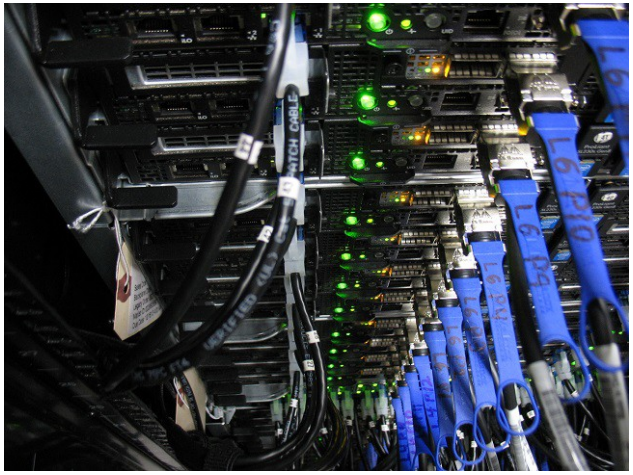
- Guild (60 nodes)
- Braid (120 nodes, also has GPUs)

PIs buy nodes in the clusters, CSC handles infrastructure

Using HPC Clusters and Basic Linux (examples on Pod / Knot)

http://csc.cnsi.ucsb.edu/docs/getting-started

Reminder - What is a ‘cluster’?



It’s just a bunch of computers – but with high speed networks.

Login node (‘pod’) is just a single computer – all the jobs can’t run there!

Your ‘home’ directory is visible on all nodes (XSEDE often uses ‘workspace’)
(to your jobs, all nodes look alike)

Biggest change is you’re using the command line!!!

Access (terminal)

For Linux, Mac, Win10, open a terminal window

Mac: Applications/Utilities/Terminal

Windows 10: PowerShell

$ ssh username@pod.cnsi.ucsb.edu

Access (Graphics) – not usually needed

Mac, Need XQuartz

Windows, XMing

$ ssh –X username@pod.cnsi.ucsb.edu

Or x2go

Important: Remote (non UCSB) login via VPN client:
http://www.ets.ucsb.edu/services/campus-vpn/get-connected

Let’s open a connection to Pod, or Knot.

We'll concentrate on Pod, rather than knot in this class

Some basic commands

pwd (what directory (Folder) are we in?) ls (list files)

more (show file, one page at a time) tail (show

end of file)
head

‘nano’ (editor) (also ‘emacs’ or ‘vi’)

File Transfer

For Linux , Mac, Win10, open a terminal/powershell, use `scp` or `rsync` commands:
E.g. Copy file.txt from your computer to your home directory on Knot

```
scp file.txt user@knot.cnsi.ucsb.edu:file_copy.txt
```

GUI based (Windows 7 has no command line scp) :

https://filezilla-project.org/ (Both Windows and Mac)

https://winscp.net/eng/download.php (WinSCP for Windows)

https://cyberduck.io (Cyberduck, for Mac and Windows 10)

Globus is another option (all operating systems). Preferred for large files transfers.

http://csc.cnsi.ucsb.edu/docs/globus-online

Storage

Not unlimited - Each dollar spent on storage is one not spent on compute

/home – active working files (Pod - /scratch – high speed, temp files)

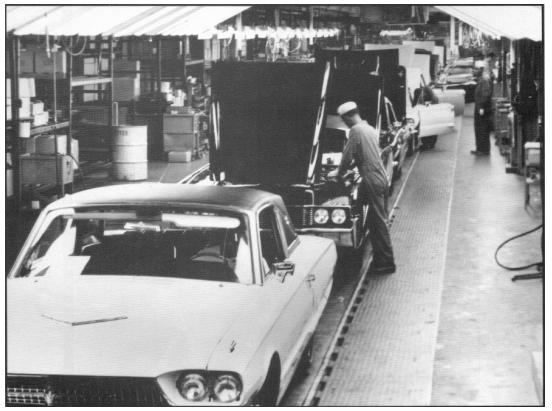
/csc/central – files that aren't immediately needed, but want close by (not visible to compute nodes).

You can move files up to Google (unlimited storage!) at a rate of about 0.5TB/day, if you make them into archives (order of a TB is good size).
https://csc.cnsi.ucsb.edu/docs/copying-files-google-google-drive
Example: Have some directory 'finished-data'

tar czf – finished-data > finished-data.tgz
rclone copy finished-data.tgz Google: (and make sure PI is co-owner)

For NSF archival requirements, either public repositories (PDB, DataOne), or locally, the library Data Collective.

Types of processing



Serial – data is dependent on previous operation, so single thread

- *parameter sweeps (need to do 1,000 runs at different values of T)*

Parallel – problem can be broken up into lots of pieces
(Tom Sawyer and painting the fence)

There are different kinds of parallel,

- Embarrassingly parallel – independent runs (e.g. Monte Carlo)
- Grids – problem broken down into nearby areas that interact

Speed of communication between processes (bandwidth, and latency)

- Single node (up to 24 or 40 cores, low latency)
- Multiple nodes (essentially infinite cores, higher latency)

Running Jobs

- When you login to Pod (or any other cluster), you are on the login node This node is NOT for running calculations!
- All jobs must be submitted to the queue – it just allocates nodes (slurm, PBS/Torque)
- Submission to the queue requires a script to be written

Example Slurm job submission script - (submit.job):

```
#!/bin/bash -l
#SLURM --nodes=1 –ntasks-per-node=10
#SLURM  --time=0:30:00
#SLURM --mail-type=start,end
#SLURM --mail-user=weakliem@ucsb.edu
cd   $SLURM_SUBMIT_DIR

./a.out
#  or, more complex…
mpirun   -np $SLURM_NTASKS    ./run.x
```

sbatch submit.job (or to test 'sbatch –p short submit.job)

Let’s run a couple of jobs (on Pod (Slurm))

cd to the directory I need.

cat the job batch.job

Submit it, ‘sbatch -q short batch.job’

showq

squeue –u $USER

What’s the output?

What's a typical (easy!) workflow?

- You're most familiar with your own computer
- Much of what you do is manipulating the data you generated on the cluster
- Soooo..... Let's mainly use your own computer!

Working on computer at home...save file.

Use file transfer program (drag and drop) to move files to cluster

Submit job/monitor job on cluster.

Use file transfer program to drag back results, and analyze locally

Repeat!!

Running Jobs on Pod (Slurm)

Start a job: $ sbatch filename.job

Check status of the running jobs:
$ showq
$ squeue -u $USER

Delete a running job: $ scancel job_id

More options for Slurm:

https://www.rc.fas.harvard.edu/resources/documentation/convenient-slurm-commands/

Available queues:

- Short queue: $ sbatch –p short submit.job
- Large memory queues : $ sbatch –p largemem submit.job
- GPU queue: $ sbatch –p gpu submit.job

Note one big changes from Torque to Slurm is that in Torque it's "queues" and in slurm it's "partition", so your submit should be with –p, not –q, in slurm.

Running Jobs on Knot (Torque)

Start a job: $ qsub filename.job

Check status of the running jobs:
$ showq
$ qstat -u $USER

Delete a running job: $ qdel job_id

More options for PBS:

https://www.olcf.ornl.gov/kb_articles/common-batch-options-to-pbs/

Available queues:

- Short queue: $ qsub -q short submit.job
- Large memory queues : $ qsub -q (x)largemem submit.job
- GPU queue: $ qsub -q gpuq submit.job

Be aware on Knot – the queue allocates you the nodes and the cores, but you need to make sure you are using the correct number of cores!

e.g. don't ask for ***ppn=2*** and then ***mpirun –np 12*** since you may share the node

Fairshare/Resource sharing

You can't monopolize cluster - limit jobs/cores

Where is Package X??? Modules

module avail
module load lumerical
module load intel/18

If no module (knot) - most software is stored in /sw

e.g. /sw/Ansys, or /sw/chem
Add to path in .bashrc or .bash_profile

Or use full path,
/sw/cnsi/lumerical/fdtd/bin/fdtd-solutions

Software, continued

You can install your own software too, e.g. download/configure/make just install in /home/$USER/somedirectory

Common ways to get software are: 'github', 'wget'
Revision control – github (e.g. https://github.com/ArcticaProject/nx-libs)
Campus now has site license for it (private repos, etc.) see https://github.com/ucsb/github-guide

git clone https://github.com/pweakliem/BookSample.git

Command line - 'subversion' if you want local copies, e.g.

svnadmin create /home/pcw/svn/myproj-1
svn import /home/pcw/SiGe-code file:///home/pcw/svn/myproj-1

Now can check out copies/edit/check in and have all revisions
svn checkout file:////home/pcw/svn/myproj-1

Extracting Information

Power of command line is that you can quickly look at info, even while job is running

```
[pcw@pod-login1 class]$ wc don-big.log
  3497 22462 222558 don-big.log

[pcw@pod-login1 class]$ grep SCF don-big.log
 SCF Done:  E(RPBE1PBE) =  -1299.71733732     A.U. after   18 cycles
          Population analysis using the SCF density.
 SCF Done:  E(RPBE1PBE) =  -1299.73486337     A.U. after   15 cycles
 SCF Done:  E(RPBE1PBE) =  -1299.75383101     A.U. after   15 cycles
 SCF Done:  E(RPBE1PBE) =  -1299.76854482     A.U. after   15 cycles

[pcw@pod-login1 class]$ grep "SCF Done" don-big.log
 SCF Done:  E(RPBE1PBE) =  -1299.71733732     A.U. after   18 cycles
 SCF Done:  E(RPBE1PBE) =  -1299.73486337     A.U. after   15 cycles
 SCF Done:  E(RPBE1PBE) =  -1299.75383101     A.U. after   15 cycles
 SCF Done:  E(RPBE1PBE) =  -1299.76854482     A.U. after   15 cycles

[pcw@pod-login1 class]$ grep "SCF Done" don-big.log | awk '{print $5;}'
-1299.71733732
-1299.73486337
-1299.75383101
-1299.76854482

[pcw@pod-login1 class]$ grep "SCF Done" don-big.log | awk '{sum+=$5} END {print sum};'
-5198.97
```

More example usage of Linux Commands

Pipes and redirects…

[pcw@pod-login1 class]$ ***grep "SCF Done" don-big.log | awk '{print $5;}' > mydata.dat***

Can use GUI to analyze run data, e.g. MatLab

mkdir : make directory
head/tail : Display beginning/end of file
cd : Change directory
cat [file] : view file
grep [pattern] [file] : Find matching patterns in a file
cut : Get a piece of string
| : Pipe, connecting commands
> and >> : Redirect and append

Scripting

Scripts string a bunch of commands together.

Example: Search for final energy in a file

```
#!/bin/bash
for i in $(ls data.??); do
     echo $i >> summary.file
      grep "SCF Done" $i >> summary.file
done
```

Torque/Slurm job files are just scripts…

```
#!/bin/bash
for i in $(seq 1 12); do
./serial-executable < inputfile.$i  &
done
wait
```

Appendix

Some useful linux commands

ls [-option] : list files
mkdir : make directory
cd : change directory
man : display manual for a command
mv : mv file/folder
rm [-r] : remove file. -r to remove folders
pwd : present working directory
cat [file] : view file
less /more : view file, one screen at a time
grep [pattern] [file] : Find matching patterns in a file

Pipes and redirection

command > file : Redirect output of command to file

command >> file : Append output of command to file

command < file1 > file2 : Get input from file1, write output to file2

command1 | command2 : Join command1 & command2

You will come across this a lot with job files, e.g. to run a python script, or Matlab, or ….

e.g. python < myinput.py

Common shortcuts

* : Wildcard

~ : Home directory

. : Current directory

.. :One directory up

TAB key: Finish commands, good for typing fast

Up arrow key – previous commands

Creating/Extracting Archives

Suppose you have an archive: package.tar.gz

Extract: $ tar -xzvf package.tar.gz

Suppose you have files you want to collect together: file1, …, file10

$ tar czf file1 file2 .. file10 package.tar.gz

Questions?

What else should we have covered?

Other ideas for a class?