Introduction to HPC Resources and Linux

Burak Himmetoglu
Enterprise Technology Services &
Center for Scientific Computing
e-mail: bhimmetoglu@ucsb.edu

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

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

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

http://csc.cnsi.ucsb.edu

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 12-32)

- Allows many users to run calculations simultaneously on nodes
- Allows a single user to use many CPU cores incorporating multiple nodes

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

Which resources do we have?

- UCSB Center for Scientific Computing (CSC) clusters
  - Access to all UCSB staff, free and condo clusters
- 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

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

HPC systems at CSC

Campus available cluster Knot:

- 110 node, ~1400 core system
- 4 ‘fat nodes’ (256/512 GB RAM)
- GPU nodes (12)
- Published papers should acknowledge CNSI and MRL

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

Condo clusters:

- Lattice (62 nodes)
- Guild (60 nodes)
- Braid (60 nodes, also has GPUs)

PIs buy nodes in the clusters, CSC handles infrastructure

XSEDE

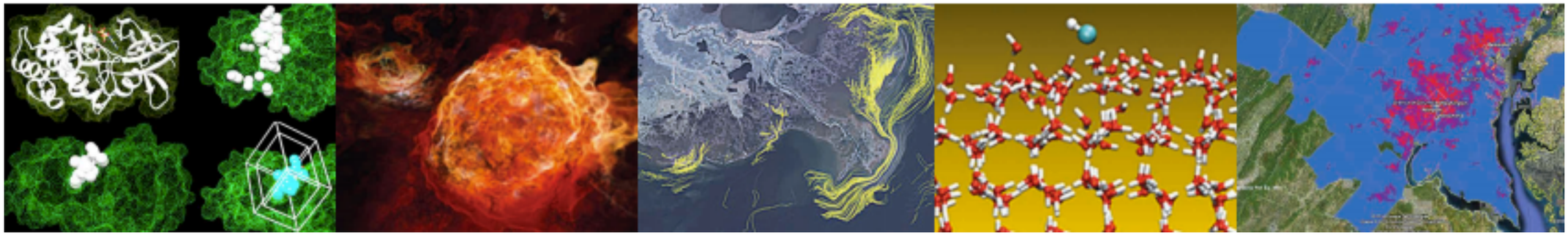
Extreme Science and Engineering Discovery Environment

XSEDE is an NSF sponsored service organization that provides access to computing resources.

https://portal.xsede.org

www.xsede.org

Currently XSEDE supports more than a dozen supercomputers and high-end visualization and data analysis resources.



Summary of XSEDE systems

Top 500

General purpose
Big data

GPU

Small,
general purpose

General purpose
Big Data

Cloud (on demand)

General purpose
Big data

High I/O
data intensive

XSEDE Compute Resources — Detail View

Name	Status	Load	Jobs
Stampede | UT Austin	Healthy	97%	R: 677 Q: 624 O: 186
Comet | SDSC	Healthy	87%	R: 2327 Q: 217 O: 297
XStream | Stanford U	Healthy	71%	R: 50 Q: 286 O: 947
SuperMIC | LSU CCT	Healthy		
Bridges Regular Memory | PSC	Healthy		
Bridges Large Memory | PSC	Healthy		
Jetstream | UT Austin	Healthy		
Gordon Compute Cluster | SDSC	Healthy	58%	R: 179 Q: 0 O: 6
Wrangler | UT Austin	Healthy		

XSEDE Campus Champions Program

Campus Champion: Represents XSEDE on the campus

As a campus champion, I (Burak) will help you:

- Understand the capabilities of HPC and get to include it as a part of your research and educational work.
- Get access to local, regional and national resources.
- Maintenance of accounts, allocations of computing time and technical expertise.
- Connect with the broader community of HPC users in your field

How to get access?

https://portal.xsede.org/allocations-overview

Campus Champion Trial Allocation

- Immediately available
- ~100-200 SUs (CPU hours)

Startup

- Available: ~1 week
- Up to 200k SUs
- Valid for 1 year, **not** renewable

HPC@UC

- Available: ~1 week
- Up to 1M SUs
- Valid for 1 year, **not** renewable

Only: Gordon & Comet

Research Allocation

- Allocation requests needed
- 4 times a year
- As much SUs as you need
- Valid for 1 year, renewable

XSEDE Portal

Get access:

- Get a username from the portal
- Send me your username
- You will get limited trial time from any resource!

On the portal you will find:

- Documentation
- Help desk
- Training sessions broadcasted from the web
- Forums

Using HPC Clusters and Basic Linux (examples on Knot)

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

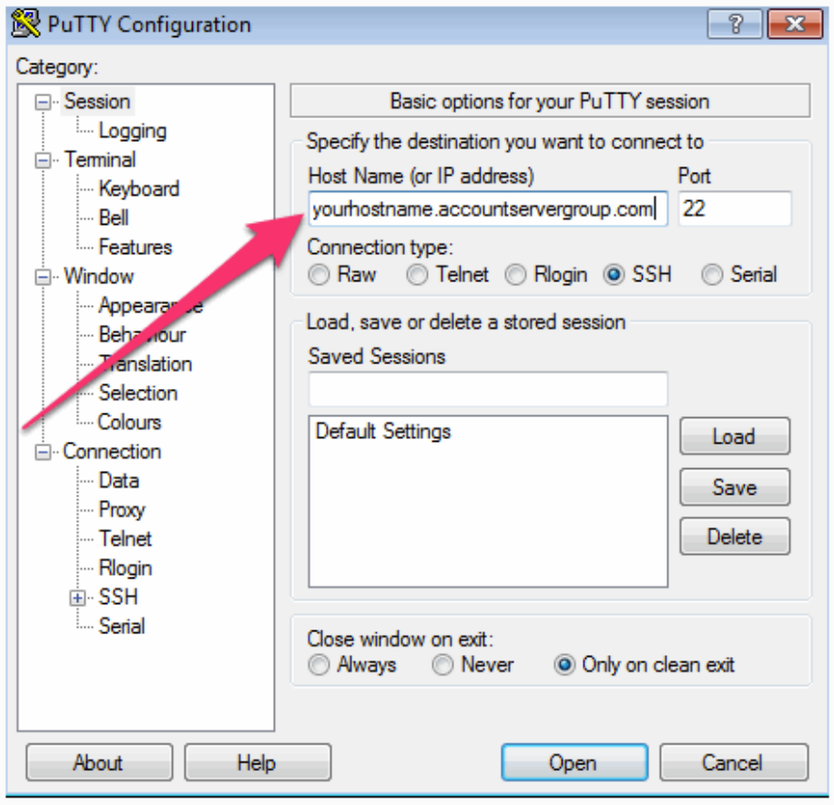
Access

For Linux and Mac, open a terminal:

$ ssh username@knot.cnsi.ucsb.edu

For Windows, you will need a client (e.g. Putty, www.putty.org)

knot.cnsi.ucsb.edu



X2go is another option (good for graphics):

http://csc.cnsi.ucsb.edu/docs/using-x2go-windows-knot

Remote (non UCSB) login via VPN client:

http://www.ets.ucsb.edu/services/campus-vpn/get-connected

File Transfer

For Linux and Mac, open a terminal, 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**

Windows users: Need a client to copy files. Usage is similar to Putty.

https://filezilla-project.org/

https://winscp.net/eng/download.php

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

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

Running Jobs

- When you login to Knot (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
- Submission to the queue requires a script to be written

Example job submission script (submit.job):

```
#!/bin/bash
#PBS -l nodes=2:ppn=12
#PBS -l walltime=2:00:00
#PBS -N test

cd $PBS_O_WORKDIR

mpirun -machninefile $PBS_NODEFILE -np 24 ./run.x
```

$ qsub submit.job

http://csc.cnsi.ucsb.edu/docs/example-scripts-running-jobs

Running Jobs

Check status of the running jobs:

```
$ showq -u $USER
$ 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

Extracting Information

For this example, we investigate the OUTCAR file generated from the software VASP (http://csc.cnsi.ucsb.edu/docs/vasp-compilation)

```
## Print direct and reciprocal lattice vectors
grep -A 3 "direct lattice vectors" OUTCAR | tail -4 > cell.info

## Print information about Forces in the unit cell
grep -A 15 "FORCE on cell" OUTCAR > force.info

## Print the calculated energy at each iteration
nIter=`grep TOTEN OUTCAR | wc -l`

for index in `seq 1 $nIter` # Grab value of energy at each iteration
do

  En=`grep TOTEN OUTCAR | head -$index | tail -1 | awk '{print $5}'`
  echo $index $En >> E_vs_iter.dat # Note the use of >> here instead of >

done
```

Example usage of Linux Commands

For this example, we investigate example.sh.

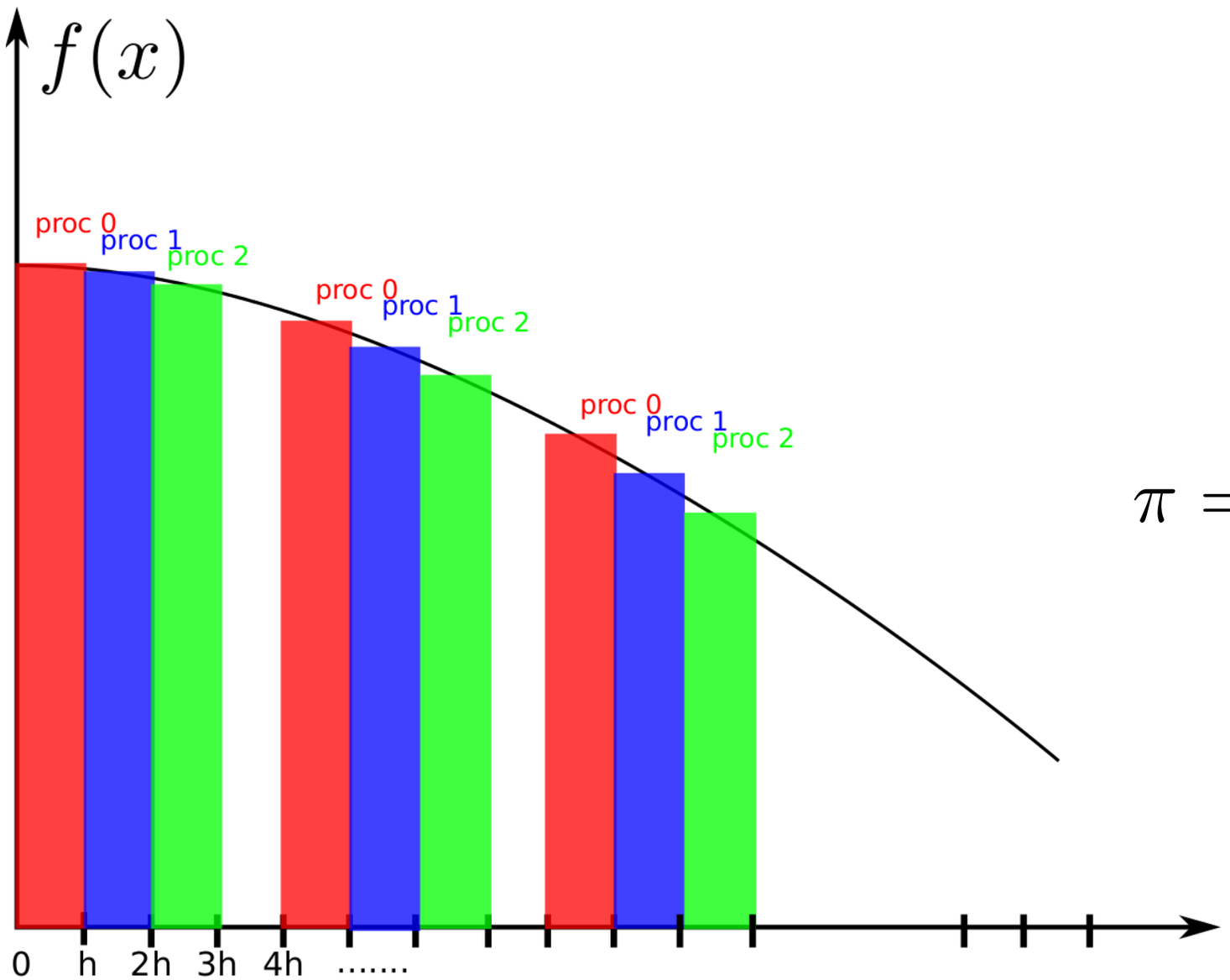
A simple shell script can be used to combine Linux commands.

The following commands are demonstrated in this script:

mkdir : <ake 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

Computing Pi in parallel

For this example, look at mpi_pi.c.



$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

```
for (i=iProc+1; i <= nDivisions; i+=nProcs){
    x = h * (i-0.5)
    sum += f(x)
}
```

Some Linux Commands

Some useful 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

Common shortcuts

* : Wildcard

~ : Home directory

. : Current directory

.. :One directory up

TAB key: Finish commands, good for typing fast

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
```