Statistical computing on campus cluster

Xiaohui Chen & Annie Qu

Statistics Brown Bag Series

September 19, 2014

Agenda

- Illinois campus cluster resources
- Run jobs on clusters

Campus cluster

- Illinois campus cluster program: https://campuscluster.illinois.edu/

Campus Cluster Usage Overview

Researcher

Login

Command Line

When you connect to the Campus Cluster, you are on a head node shared by many users.

Batch Script

Commands for code execution, copy input files to scratch,... Specify number/type of nodes, length of run, output directory, ...

Data

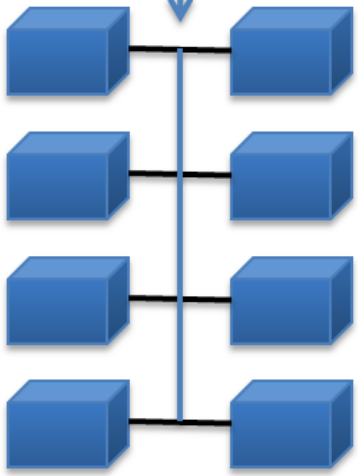
Job

Head Nodes

Use for tasks such as file editing, code compilation, data backup, and job submission.

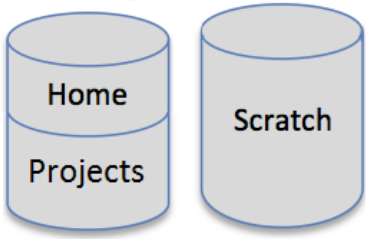
*Store project files such as source code, scripts, and **small** input data sets to your Home directory.*

Run jobs by submitting your batch script to the compute nodes using the "qsub" command.



Compute Nodes

Your job is submitted to a queue and will wait in line until nodes are available. Queues are managed by a job scheduler that enables jobs to run efficiently.



File System

Data

Read/write data from compute nodes to your Scratch directory.

Campus cluster overview

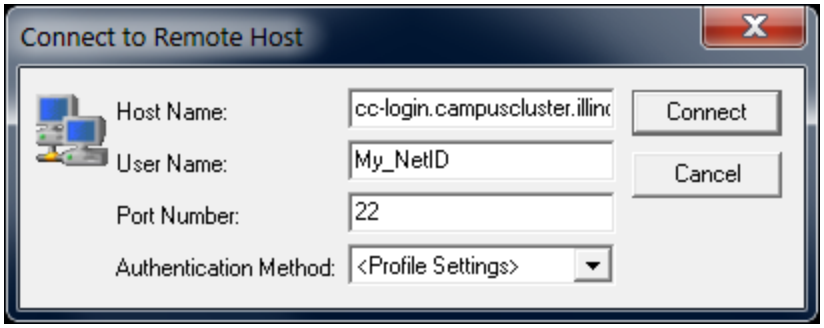
CC user	Head nodes	Compute nodes
Connect to a head node from your local computer using an ssh client.	• Learn Linux basics • Transfer files to and from the campus cluster • Use a text editor • Build applications • Submit batch jobs for scheduling on the compute nodes	• All batch jobs run on the compute nodes. • Each investor group has a dedicated *primary* queue. • There is also a shared *secondary* queue for opportunistic access to idle nodes, and a very short time *test* queue that provides quick turnaround time.

- How to access?

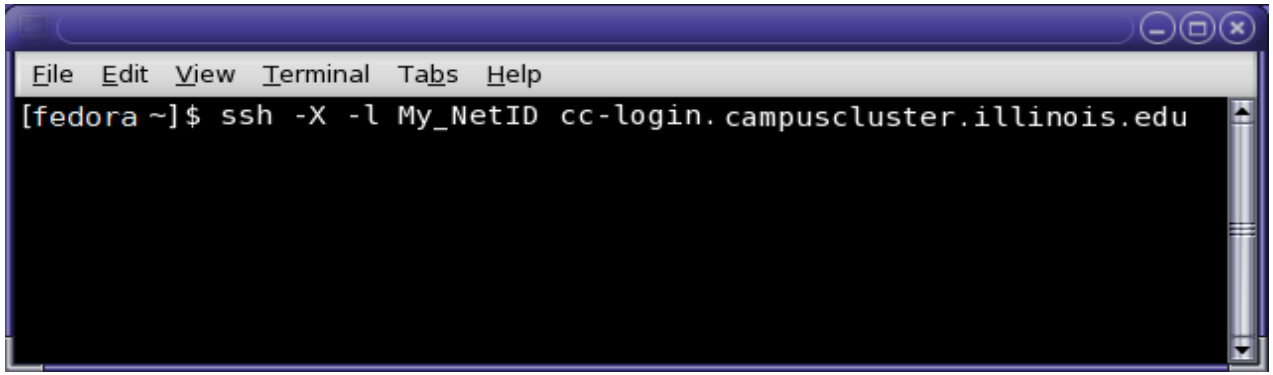
Access method	Hostnames	Head nodes
SSH	• cc-login.campuscluster.illinois.edu • taub.campuscluster.illinois.edu • golub.campuscluster.illinois.edu	Taub & Golub Taub (12-core per node) Golub (16-core per node)

SSH: remote access

- Windows: Graphical User Interface (GUI)



- Linux/unix, Mac OS: Command Line Interface (CLI)



Campus cluster is Linux based

Useful Unix/Linux Commands

Basic Commands

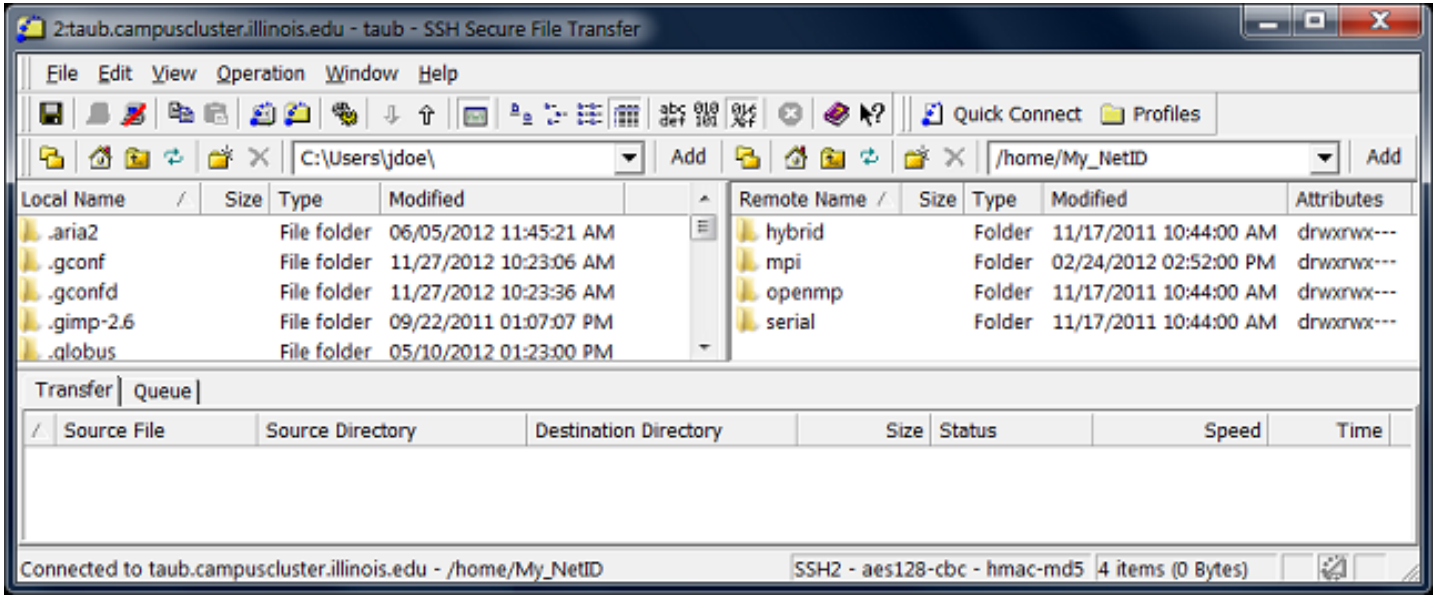
Command	Description
ls **ls -l**	List directory contents Detailed listing of directory contents
pwd	Display the path of the current/working directory
man *command_name*	Display online help (manual page) for*command_name*
quota	Display your home directory disk usage
ps -u	Display detailed information about your running processes
exit *or* **logout**	Log out of your current session
history	Display a list of the commands you've recently run
date	Display the system date and time

Working With Files

Command	Description
nano *myfile*	Create a new (or edit an existing) file named *myfile* with a simple text editor
grep *string myfile*	Display the lines in *myfile* that contain a matching pattern(*string*)
cat *myfile*	Display the entire contents of the file*myfile*
more *myfile*	Display the contents of the file *myfile*, one page at a time
cp *myfile1 myfile2*	Copy the file *myfile1* to *myfile2*
mv *myfile1 myfile2*	Rename the file *myfile1* to *myfile2*
mv *myfile mydir*	Move the file *myfile* into the directory*mydir*
rm *myfile*	Delete the file *myfile*
mkdir *mydir*	Create a directory named *mydir*
cd *mydir*	Change the current directory to *mydir*
rmdir *mydir*	Remove the directory *mydir* (if empty)

Data transfer

- GUI data transfer; e.g. SSH Secure Shell SFTP client
- Drag and drop interface: recommended



- CLI data transfer: scp

Software list

- GCC
- MATLAB
- R
- Python
- Mathematica
- OpenMPI
- Octave
- …

User environment: modules

- To load a software into your programming environment, use command
  module load R
  module load matlab
  module load gcc
- Then, use command
  R
  matlab
- Use **no display** option
  matlab -nodisplay

Where to run your programs?

- Primary queues
  - Each investor group has unrestricted access to a dedicated *primary* queue with concurrent access to the number and type of nodes in which they invested.
- Secondary queues (default!)
  - A shared *secondary* queue will allow users access to any idle nodes in the cluster. Users must have access to a primary queue to be eligible to use the secondary queue.
- Test queues
  - A *test* queue is available for providing very short jobs with quick turnaround time. Jobs in the test queue currently only run on Taub nodes.

Comparison of the three queues

Queue	Max Walltime	Max # Nodes
Primary	Unrestricted	Unrestricted*
Secondary	4 hours	208
Test	5 minutes	2
Stat	Unrestricted	4

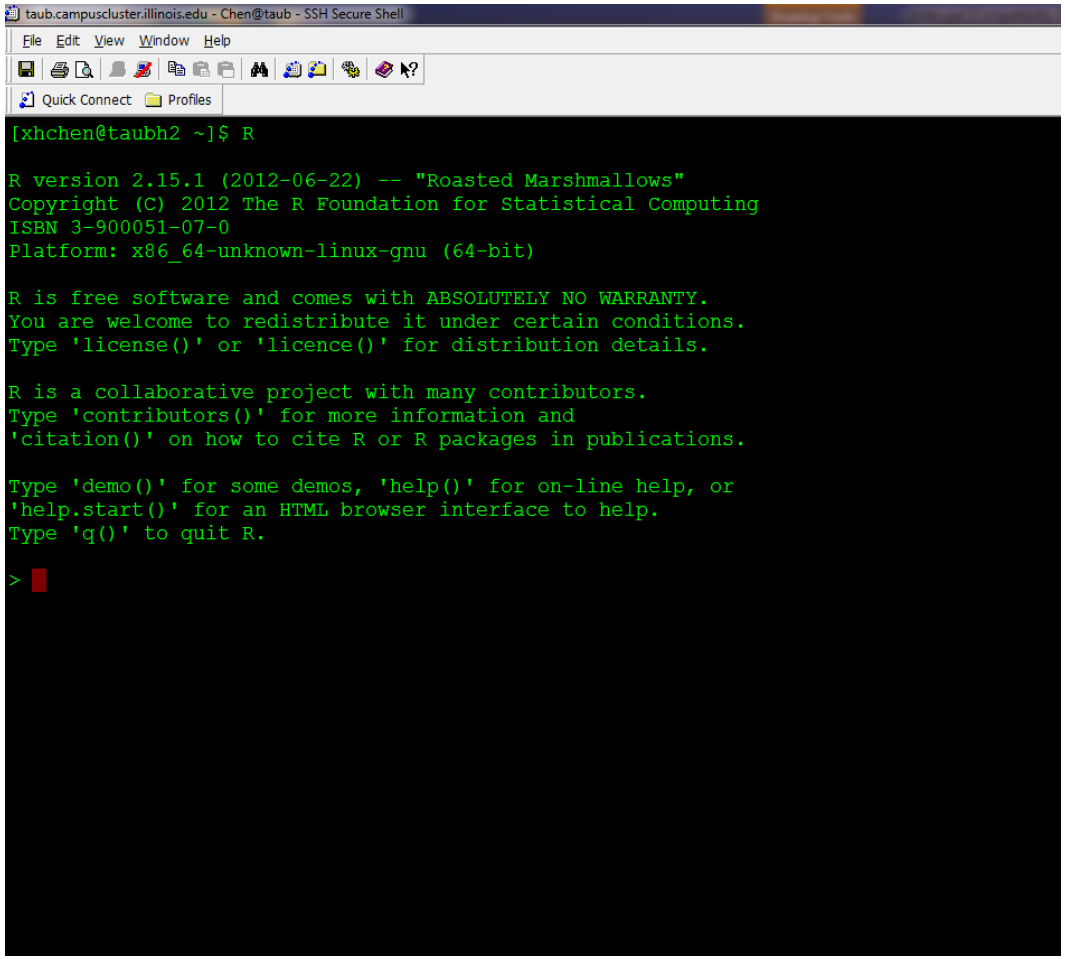
- The primary queue for Statistics Department, is called *stat* queue with 4 nodes.

***: depend on the invested # nodes**

Agenda

- Illinois campus cluster resources
- Run jobs on clusters

Run R on a PC



Warning: do NOT run R program this way on clusters!!

Run jobs on clusters: batch commands

- Batch jobs are submitted through a *job script* using the qsub command.
- But you can**not** submit a job on the command line.
- What do you need to do?
  1. Write a PBS script file.
  2. Submit the PBS file using qsub.

PBS file

- PBS is a job resource manager. A job is defined as a computational task such as computational simulation or data analysis. PBS provides job queuing and execution services in a batch cluster environment.
- **Example**: lasso.pbs

```
#!/bin/bash                                  Type of shell

#PBS -l nodes=1:ppn=6                        Resources request via
#PBS -l walltime=04:00:00                    PBS queuing system: 1
                                             node, 6 cores/node, max
                                             4 hours to kill

cd $PBS_O_WORKDIR                            Directory to get job run

module load R                                Software needed
R CMD BATCH sim_lasso_p=100_n=100.R          Execute the program!
```

Job execution: qsub

- Submit your job (to the default queue):
  `qsub lasso.pbs`

- Check status of your jobs:
  `qstat –u xhchen`

- Delete your job:
  `qdel JobID`

- Peek your job:
  `qpeek JobID`

More options and some caveats

- Submit your job to the stat queue
  `qsub -W group_list=stat -q stat  lasso.pbs`

- Revisit: lasso.pbs

```
#!/bin/bash

#PBS -l nodes=1:ppn=6
#PBS -l walltime=04:00:00,nodes=1:ppn=6:m96G

cd $PBS_O_WORKDIR

module load R
R CMD BATCH sim_lasso_p=100_n=100.R
```

Note: Do not use the memory specification unless absolutely required since it could delay scheduling of the job; also, if nodes with the specified memory are unavailable for the specified queue the job will never run.

A toy example to run MATLAB

- **Step 1: make a MATLAB script**
- Example: generate a 1000*1000 random matrix and then compute its singular value decomposition (svd)
- Name the file: `randmat.m`

```
% randmat.m
% M = U * S * V'

p = 1000;
M = randn(p);
[U S V] = svd(M);

save('randmat.mat');
```

A toy example to run MATLAB

- **Step 2: make a PBS script file**
- Name this file: `randmat.pbs`

```
#!/bin/bash
#PBS -l nodes=1:ppn=6
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module load matlab
matlab -nodisplay < randmat.m > randmat.mout
```

A toy example to run MATLAB

- **Step 3: submit the job via qsub**
- Run the following command line:

  - On the secondary queue
    `qsub randmat.pbs`

  - On the stat queue
    `qsub -W group_list=stat -q stat  randmat.pbs`

You can put as many programs as you want.

Randomization

```
% randmat.m
% M = U * S * V'

p = 1000;
M = randn(p);
[U S V] = svd(M);

save('randmat.mat');
```

- Run twice independently, see what you get:

Run 1: norm(M) = 63.0942

Run 2: norm(M) = 63.0942

- **Need randomization of the seed!**
- 1. use the random system noise
- 2. use high-precision system time (mixed with the job ID):

as.integer((as.double(Sys.time())*1000+Sys.getpid()) %% 2^31)

Summary

- Three key ingredients:
  1. Debug your programs on your computer (remember to **randomize** and add a save line in the end)
  2. Write and customize your PBS file, according to the size of your problems
  3. Learn various options of qsub (man qsub)

- Now, just enjoy ☺

THANK YOU!