Introduction to High Performance Computing Cluster at CUIMC

Zilan Chai

March 2022

Overview

- Access to HPC
  - Login
  - Linux command
  - Transferring files between local and HPC
- R version and installation of R packages
- Submitting jobs
  - `qsub` command
  - Example of a shell script
  - Parallel computing via multiple tasks under a single job
- Check Cluster Usage

Access to HPC

- Contact HPC team to set up an account

  https://www.mailman.columbia.edu/information-for/information-technology/high-performance-computing-hpc

- Login through terminal

```
ssh uni@login.c2b2.columbia.edu
```

Basic Linux Commands

`cd`	cite directory
`cd .`	current directory
`cd ..`	go back to last folder
`cd ~`	go back to root folder
`ls`	list everything under this directory
`ls -l`	list long
`man ls`	help with command
`q`	quit
`pwd`	pass working directory
`nano`	create file
`cat`	print content of file
`clear`	clear the screen
`mkdir`	create folder
`cp`	copy file
`vi`	view (exit the view mode: esc : q! enter)
`mv a.txt b.txt`	change file name from "a" to "b"

Examples:

```
cd /ifs/scratch/msph/biostat/zc2326
cd /ifs/home/msph/biostat/zc2326
```

Note: all the jobs should be submitted under the scratch folder, not home.

Transferring files between local and HPC

- 1. scp command
- 2. Mac OS – cyberduck
- 3. Windows – Winscp

1. scp command

- Transfer a file from your computer to the cluster

```
scp –r /local directory/filename.text youruni@login.c2b2.columbia.edu:/ifs/scratch/msph/biostat/youruni
```

- Copying files from cluster

```
scp youruni@login.c2b2.columbia.edu:/ifs/scratch/msph/biostat/youruni/JOBNAME*  /local directory
```

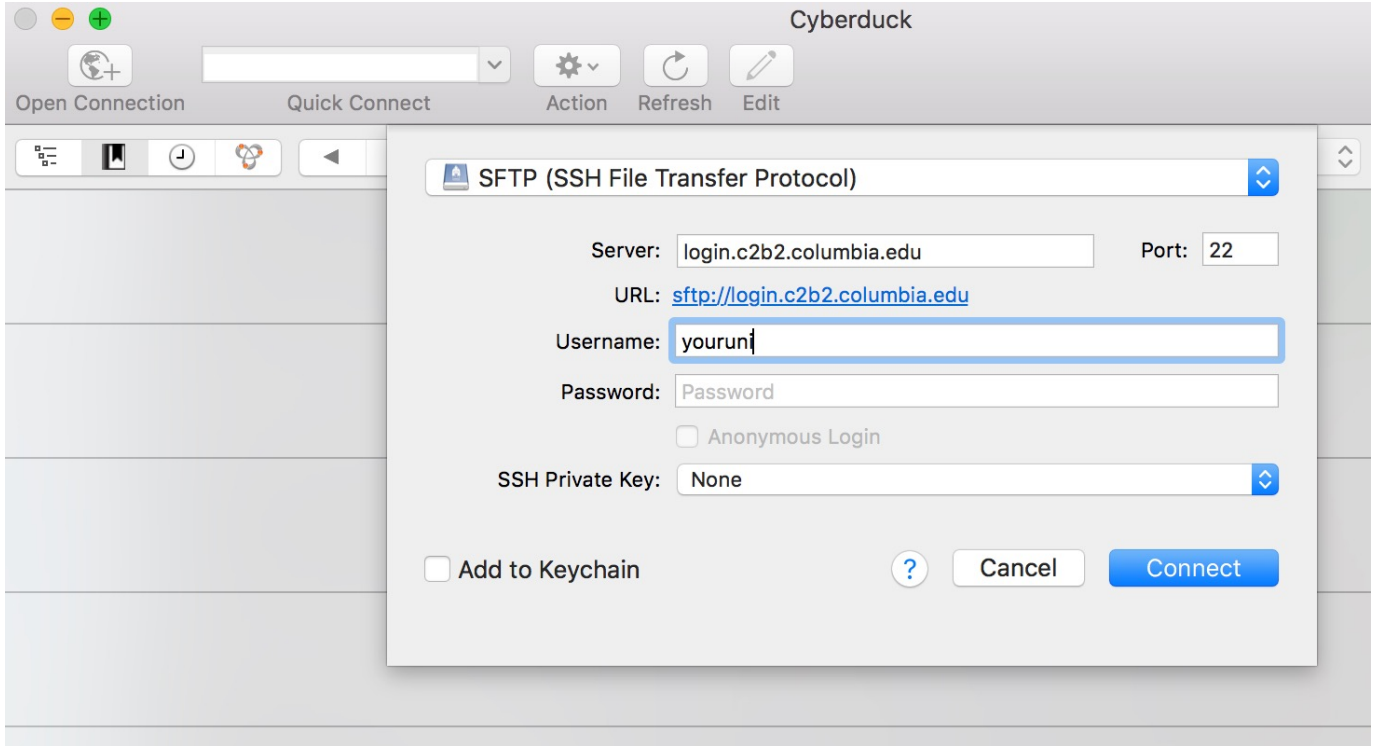
```
-r : Transfer Entire Folder
```

Mac OS – cyberduck

Open Connection, select “SFTP (SSH File Transfer Protocol)”

Server: login.c2b2.columbia.edu

Use uni and password to login.



R version and installation of R packages

- Use R in cluster:

```
qrsh
/nfs/apps/R/3.6.0/bin/R
```

- Create a version specific directory in home directory to save R libraries.

- Install R packages:

```
install.packages("Package_Name", "/ifs/home/msph/biostat/youruni/R_LIB",
dependencies = TRUE, repos = "http://cran.rstudio.com/")
```

```
-bash-4.1$ qrsh
-bash-4.1$ /nfs/apps/R/3.4.1/bin/R

R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
> q()
```

```
-bash-4.1$ cd /ifs/home/msph/LeeLab/zc2326/R_LIB
-bash-4.1$ ls
BH       bindrcpp  crayon  dplyr      glue  MCMCpack  pkgconfig  RAMP  rlang     tibble
bindr    cli       dfcrm   glinternet MASS  pillar    plogr      Rcpp  rstiefel  utf8
```

Submitting Jobs

```
qsub runR.sh refund.R   # passing this R file to $1 in the shell file.
```

```
qdel                    delete a job
qstat                   check job status    # r: running , t: transfering
qdel -u username    kill all jobs submitted by user username
del -j  jobID
qacct -j 4964255  # tells every things about the job, memory/time it takes;
```

Submitting Jobs - Write a Shell file

-S /bin/bash	telling it this is a bash script
#! /bin/sh	indicating this is a shell doc
-cwd	change working directory results go under this directory
-l mem =1G	memory you require in the cluster
-l time = 01:10:00	time in hour:minute:second
-N R-code_simu	the name of the result file will be this
-j y	error file & result file, generate those two files into one file.
SCRIPT = $1	place holder for my R code

Submitting Jobs - Shell file example

```
#!/bin/sh
#$ -cwd -S /bin/bash
#$ -l mem=1G
#$ -l time=:5:
#$ -N R-code_simu -j y

SCRIPT=$1

R=/nfs/apps/R/3.1.1/bin/R
R_LIBS_USER=/ifs/home/msph/LeeLab/zc2326/R_LIB:/ifs/scratch/msph/software/R
/library311:/ifs/scratch/msph/software/R/library:$R_LIBS_USER

${R} --vanilla < ${SCRIPT}
```

Submitting Jobs – Parallel computing

- Parallel computation via multiple tasks under a single job
  - Use task id to specify simulation setting
  - Always remember to include code in R script for saving results.

- Example 1 (Simulation):

3 parameters: R, m, n

Here are the possible values of these parameter:

R = (80, 100)

m = (1, 2, 3)

n = (20, 25, 30)

You have a fancy Simulation function: fun(R, m, n)

Want to run simulation for all possible combinations of the parameters.

example1.sh

```
#!/bin/bash
#$ -cwd -l mem=3g,time=00:10:00 -S /bin/bash -N JOBa1 -j y -t 1-18

currind=$SGE_TASK_ID

R=/nfs/apps/R/3.6.0/bin/R
export R_LIBS_USER=/ifs/home/msph/biostat/zc2326/R_LIB:/ifs/scratch/msph/software/R/library360:/
ifs/scratch/msph/software/R/library:$R_LIBS_USER

${R} --vanilla --args $currind < example1.R
```

example1.R

```r
args<-commandArgs(TRUE)
currind <-as.integer(args[1])
print(currind)

R.candi <- c(rep(80, 9), rep(100 ,9))
m.candi <- rep(c(rep(1, 3), rep(2, 3),rep(3, 3) ) , 2)
n.candi <- rep(c(20, 25, 30), 6)

para <- cbind( R.candi,  m.candi,n.candi)

fun <- function(R, m, n){
  Q = R + m + n
  c(R, m, n, Q)
}

for (ii in 1:length(R.candi)){
  if(ii==currind) {
    t <- fun( para[ii,1], para[ii,2], para[ii,3])
  }
}


write.csv(t, file = paste0("res", currind, ".csv"))
```

```
qsub example1.sh example1.R
```

- Example 2 (Bootstrap):

- In order to conduct 10,000 bootstraps, you can split the job into 10 separate jobs, each with 1,000 bootstraps.
- Each job returns a csv file with the results. Then combine all results and do the analysis.

```
# Bootstrap

args<-commandArgs(TRUE)
currind <-as.integer(args[1])
print(currind)


B <- 1000

for (ii in 1:10){
  if(ii==currind) {
    for (i in 1:B){
      p.res <- BS_EM_ABO(260, 270, 420, 70, 10000)
    }
  }
}

write.csv(p.res, file = paste0("thirdproblem", currind, ".csv"))
```

Check cluster usage

qacct -o zc2326 -b 202203010000 -e 202203161600

- The result will show mem and cpu usage in second

Calculate: Cpu(s)*0.02/3600 + mem(s)*0.0067/3600 <$2000