Quick Start of Using High Performance Computing (HPC) in Texas Tech University

Sunthud Pornprasertmanit

June 19, 2015

High Performance Computing Center (HPCC) in TTU provides a large amount of computing resources. The number of computing nodes is higher than 10,000. That is, people can run up to 10,000 jobs at the same time. Imagine that one simulation takes one year using one computer can be run in only a day using 400 nodes on HPC. The goal of this class is to encourage students to make more use of the HPC by showing how simple it is to access and use. This paper cannot cover all details about using the HPC. For further information, please refer to computing documentation in HPCC webpage (`http://www.depts.ttu.edu/hpcc/`).

1 How to Log In

Login require the use of a secure shell program (ssh). Details are platform specific.

1.1 Windows

There are at least two ways to log in the HPC by using **Putty** or **Xterm**. Unlike other universities, TTU does not require users to login VPN if using off campus.

1.1.1 PuTTY

Putty is an secure shell (ssh) that allows users to connect to HPC and send commands with a keyboard. Download here: `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

1. Open the PuTTY program from Start menu or the file in a folder
2. Put `hrothgar.hpcc.ttu.edu` in the Host Name (or IP address)
3. Put 22 in Port
4. Click on Open. . .
5. Type Username and Password in the pop-up window. If you cannot access by your TTU ID, go to `http://www.depts.ttu.edu/hpcc/accounts/accountrequest.php` to sign up.

1.1.2 Xterm

Xterm is an X server program for Windows that allows users to interact with the HPC using more than just a keyboard. For information about setting up an X server in Windows and using Xterm see Paul Johnson's YouTube video here: `https://www.youtube.com/watch?v=JOFvvPIvMKO`. You may download Xterm from `http://sourceforge.net/projects/xming/`. Xterm is very useful for any people who use Emacs to interact with R in Linux. I personally do not use it so I will simply use PuTTY for now.

1.2 Mac OS

Mac OS have a built-in application for connecting to HPC via ssh (what Putty does).

1. Go to the Terminal program in your Mac (Applications folder => Utilities folder => Terminal).
2. Type in "ssh username@hrothgar.hpcc.ttu.edu"
3. You will be prompted for your password. Once this is entered, you're in!

1.3 Linux

The steps for logging in with Linux is similar to a Mac. Make sure you have an ssh program installed!

1. Go to a terminal.
2. Type in "ssh username@hrothgar.hpcc.ttu.edu"
3. You will be prompted for your password. Once this is entered, you're in!

1.4 Exercise

Log in to the HPC. After that, type the following line:

```
echo "I am glad I live long enough to learn how to use HPC"
```

2 Basic Linux commands

After you have logged in the HPC, you are in the Linux OS. You need to know basic commands in the Linux system. Here is a list of some of the most useful commands:

`ls -la`	List all files in the current directory
`pwd`	Show the path of the current directory
`cd x`	Move into a directory x
`cd ..`	Move into the directory higher than the current directory
`cp x y`	Copy a file x to a file y
`rm x`	Remove a file x
`rm -rf d`	Remove a directory d without prompts
`mkdir d`	Make a directory d
`mv x y`	Move/rename a file x to a file y. The file x will not exist after running this command.
`cat x`	Show content of a file x
`exit`	Log out of the HPC

2.1 Tips

- Tab will automatically complete file or function names (assuming the names are unique)
- Use `man X` to view the help file for a command X

2.2 Exercise

List the files in the home directory. Create a new folder named `test` from the home directory. Copy a file to the built directory. Go into the built directory. Rename the copied file to `messitup.omg`. Delete the renamed file. Go back to the home directory. Delete the `test` folder.

3 Put Files in HPC

- **FileZilla, WinSCP, and other programs.** This also helps you to put files into any network drive. For the HPC drive, host is hrothgar.hpcc.ttu.edu and port is 22. If you put username and password correctly, you should be able to access the network drive. Note: FileZilla is multiplatform and works with Windows, Mac and Linux but it lacks some of the features of WinSCP (only available on Windows).
- **Barebone SCP.** In Macs or Linux, users may use the terminal program to copy files from their hard drive to the HPC directly:

```
scp -r hdFolder hpc.quant.ku.edu:
```

Alternatively. users may copy files from HPC to their hard drive.

```
scp -r hpc.quant.ku.edu:hpcFolder .
```

3.1 Run a program interactively

Let's run R in HPC. If we simply type R, R will be not recognized. To activate R, we need to activate R shortcut in the workspace:

```
> soft add +R
> which R
+ lustre/work/apps/R/bin/R
```

The `which` command is used to show where R is run from the shortcut. From here, you can simply type `R` in the console. R will be activated. You can quit R by typing `q()` or use Ctrl+C to escape from this application.

3.2 Exercise

1. Download files to your local hard drive by browsing:
   `http://www.myweb.ttu.edu/spornpra/RSIM/rsimexamples.tgz`
2. Upload the downloaded file to the temp folder you created in HPC.
3. Type `tar xvzf rsimexamples.tgz` to extract files. Note that you may tar all files by typing `tar cvzf rsimexamples2.tgz *.batch *.R`
4. On HPC type: `wget https://dl.dropboxusercontent.com/u/15234014/rsimexamples.tgz`

4 Shell in the Nutshell

The following is a submission script to execute a set of commands saved in a file. A complete discussion of submission options is beyond this paper; instead I will discuss the details related to sending commands to HPC. Here is an example script to run an Mplus job.

```
#!/bin/csh
#$ -S /bin/bash
#$ -cwd
#$ -V
#$ -q serial
#$ -P hrothgar
#$ -N test1

/lustre/work/apps/R/bin/R -vanilla -no-save < testR1.R >
```

- `-N` is the name of the job.
- `-q` is the type of clusters in HPCC that you wish to use. HPCC allows us to use `serial` and `normal` for nonparallel and parallel processing, respectively. We are using only single node in this job so `serial` is used.

- `/lustre/work/apps/R/bin/R` is the command to call R. `-vanilla` is to ask R to not initiate anything that is required for R in interactive model. `-no-save` is to ask R to not save workspace. The file in the pointy bracket is the target R file to be run. If users wish to get the printed screen of running the target R file, users can put a file name after the pointy bracket.

Note that shell is case-sensitive. `Test1.R` is different from `Test1.r`

4.1 Code for Running Programs

4.1.1 R

```
/lustre/work/apps/R/bin/R -no-save -vanilla -f "filename.R"
```

4.1.2 Matlab

```
/lustre/work/apps/MATLAB/R2013b/bin/matlab -nodisplay -r "filename,
exit"
```

Note that HPCC requires users who run Matlab in HPC to use `#$ -q serial` in the submission job.

4.2 Exercise

Examine the contents inside the following files: `testR1.batch`, `testR1.R`, `matrixmultiply.batch`, and `matrixmultiply.m`.

Create an R script based on the exercise from the second class (i.e., the simulation on logistic regression). Make sure that you save your results to a file at the end of the script. `dput`, `write.table`, or `write.csv` could be used here. Then, create a submission file for the new R script.

5 Use HPC

`qstat`	Check your job status. You might put `-Q` or `-q` to see the shorter report.
`qsub x`	Submit a job, following by the name of a shell file x
`qdel n`	Delete a job, following by your job id (n)
`qselect`	Select the job names from a specified criterion, such as `-u` for user name or `-s` for the state of the job

After a job is finished, you will see four additional files. <job_name>.o<job_id> shows the printout of running your job in a program. <job_name>.e<job_id> shows the errors from running your job. <job_name>.po<job_id> and <job_name>.pe<job_id> are the parallel environment startup output and error files. Usually, we will expect only '`.e`' and '`.o`' files.

5.1 Tips

- To view the current status of each type of job in HPCC: `http://hygd.hpcc.ttu.edu/Hqueues.txt`
- If you have multiple jobs that the file names are different in only a running number. You may use `-t` command in a submission script.

```
#!/bin/csh
#$ -S /bin/bash
#$ -t 1-5:1
#$ -cwd
#$ -V
#$ -q serial
#$ -P hrothgar
#$ -N test1

/lustre/work/apps/R/bin/R -vanilla -no-save < testR$SGE_TASK_ID.R
>
```

  Notice that there are two changes from the previous script. First, `-t` is added. The running number is 1 to 5 increased by 1. Second, `testR1.R` is changed to `testR$SGE_TASK_ID.R`. `$SGE_TASK_ID` represents the running number. It will be changed to 1, 2, 3, 4, and 5, respectively. You can save this script in a file and submit this file using `qsub` command. See `repeated.batch` in the example.

- If your script is in subfolder, such as `sim1/testR1.R`, ..., `sim5/testR5.R`. The last line of the previous script can be changed as follows:

```
/lustre/work/apps/R/bin/R -vanilla -no-save <
./sim$SGE_TASK_ID/testR$SGE_TASK_ID.R >
```

- I usually edit my file in Windows and transfer files to Linux. Most Linux folks hate this but it is convenient for me. If you want to edit your files in Linux, you may use editors like `nano`, `vi` or `Emacs`. Sometimes, I use these for a quick edit.
- Sometimes, the file is not able to execute. This is a feature of Linux that controls users who can read, write, or execute files. To make your file executable, you need to change the status of the file by

```
chmod -u +x repeated.batch
```

- If we want to delete all of your jobs from the HPC, you may use the following command

```
qselect -u "username" | xargs qdel
```

or

```
qdel $(qselect -u "username")
```

5.2 Exercise

Type `qstat` to check the current status of the HPC. If you do not have any running jobs, nothing should be shown. Then, submit all `batch`-extension files from the test folder. Check the current status of the HPC again. Examine the '`.e`' and '`.o`' files resulted from the submission. Then, change the name of `testR1.R` to `testR1.r`. Submit the job from the `testR1.batch` file again. Check the resulting '`e`' and '`o`' files.

Next, submit your job from Section 4.2. Make sure that your job is run successfully and the appropriate output is saved.

6 File Handling with R

- Sometimes, you would like to create multiple R scripts. These scripts are very similar except some design conditions or replication numbers. The `gsub` function is very useful in this task. For example, we would like to run the simulation example from the first class.

```
> # Here is the R script template used to create data and save the results
> # The 'subSampleSize', 'subVariance', and 'subNum' are terms
> # that will be substituted later.
> script <- '
+ set.seed(1234)
+ nrep <- 1000
+ n1 <- subSampleSize
+ n2 <- 120 - n1
+ v1 <- 1
+ v2 <- subVariance
+ result <- rep(NA, nrep)
+ for(i in 1:nrep) {
+   y1 <- rnorm(n1, 0, sqrt(v1))
+   y2 <- rnorm(n2, 0, sqrt(v2))
+   y <- c(y1, y2)
+   g <- c(rep(1, n1), rep(2, n2))
+   dat <- data.frame(g = g, y = y)
+   out <- t.test(y ~ g, data = dat, var.equal = TRUE)
+   result[i] <- out[["p.value"]]
+ }
+ write.csv(result, file="outputsubNum.csv")
+ '
> # Create combination of the sample-size condition and the variance
> # condition. The 'expand.grid' function is used to create all
> # possible combination of all conditions.
> n1 <- c(20, 40, 60, 80, 100)
> v2 <- c(1, 3, 10)
> condition <- expand.grid(n1, v2)
> # The R script template will be substituted by the condition values
> # from each combination. Then, the substituted template will be
> # written in sim*.R
> for(i in 1:nrow(condition)) {
+   temp <- gsub("subSampleSize", condition[i, 1], script)
+   temp <- gsub("subVariance", condition[i, 2], temp)
+   temp <- gsub("subNum", i, temp)
+   write(temp, paste("sim", i, ".R", sep=""))
+ }
```

See `hpcclass1.R` and `hpcclass1.batch` for the example files.

- If all files are saved in the same folder, the amount of files is intractable. We need to create folders for each job. We can use R to create multiple folders:

```
> # Create folders called n100 and n200 by R
> n <- c(100, 200)
> for(i in 1:length(n)) {
+   temp1 <- paste("n", n[i], sep = "")
+   dir.create(temp1)
+ }
```

- If we want to delete files from R, use the `file.remove` function
- To check whether a file is in the current directory, use the `file.exists` function
- To read all file names in the current directory, use `list.files()` function

6.1 Exercise

Create two more files implementing the same simulation on logistic regression from Section 5.2 but with different X distributions. The X distributions of two files are (a) t distribution with *df* of 3 and (b) chi-square distribution with *df* of 4. Make sure that the means and standard deviations from the new distributions are derived from the population mean and standard deviation of 0 and 1, respectively.

7 Using HPC jobs using parallel processing

HPC allows us to ask for multiple processors in each job. For example, we may ask 12 processors to run 1,000 replications. Each job will use 1 processor as a head node and the other 11 processors as slave nodes. Each replication will be distributed to 11 processors and 11 processors will run simultaneously. Running different jobs with parallel processing is the most efficient way to use HPC. As an example, let's make a simulation that uses parallel processing.

```
> library(parallel)
> set.seed(123321)
> nrep <- 1000
> FUN <- function(temp) {
+   n <- 200
+   b <- 0.5
+   x <- rnorm(n, 0, 1)
+   e <- rnorm(n, 0, sqrt(1 - b^2))
+   y <- b*x + e
+   dat <- data.frame(x = x, y = y)
+   out <- lm(y ~ x, data = dat)
+   summary(out)[["coefficients"]][2, c(1, 2, 4)]
+ }
> result <- mclapply(1:nrep, FUN, mc.cores = 11)
> result <- do.call(rbind, result)
> save(result, file=paste("result.rda", sep=""))
```

This script uses the `mclapply` function to run the parallel processing where different replications were distributed to different nodes. The result of the function is a list so the `do.call` function is used to change list to matrix. Let's save this script in a file called `parallel.R`. Then, the submission file named `parallel.batch` to send this job to HPC requesting 12 nodes.

```
#!/bin/csh
#$ -S /bin/bash
#$ -cwd
#$ -V
#$ -q normal
#$ -pe fill 12
#$ -P hrothgar
#$ -N test1

/lustre/work/apps/R/bin/R -vanilla -no-save < parallel.R >
```

The change in this code is the `-pe` option. This option indicates the number of requested nodes, which is 12 here. The queue is changed to normal to run parallel processing.

The problem of running paralelling processing across replications is that the replications are not purely independent, which could lead to erroneous results. To ensure the independence is to set the seed number in each replication. The best practice is to set seed numbers based on the L'Ecuyer method for each replication.

```
> set.seed(123321)
> nrep <- 1000
> library(parallel)
> RNGkind("L'Ecuyer-CMRG") # Use L'Ecuyer method
> seed.l <- list() # Create list of seed number to provide for each rep
> s <- .Random.seed # Extract the current seed number
> for (i in 1:nrep) {
+   seed.l[[i]] <- s # Save seed number to the list
+   s <- nextRNGStream(s) # Build independent seed number
+ }
> FUN <- function(repseed) {
+   RNGkind("L'Ecuyer-CMRG")
+
+   # Assign the seed for each rep inside each node
+   assign(".Random.seed", repseed, envir = .GlobalEnv)
+   n <- 200
+   b <- 0.5
+   x <- rnorm(n, 0, 1)
+   e <- rnorm(n, 0, sqrt(1 - b^2))
+   y <- b*x + e
+   dat <- data.frame(x = x, y = y)
+   out <- lm(y ~ x, data = dat)
+   summary(out)[["coefficients"]][2, c(1, 2, 4)]
+ }
> # Use mclapply to distribute seed number for each rep
> result <- mclapply(seed.l, FUN, mc.cores = 11)
> result <- do.call(rbind, result)
> save(result, file=paste("result2.rda", sep=""))
```

This method ensures the independence of each replication. You can also replicate the simulation results as well. See `parallel2.R` and `parallel2.batch` for the simulation with parallel method using L'Ecuyer seed.

7.1 Exercises

Modify three files from Section 6.1 to run with parallel processing. Submit your R file with the tip for submitting repeated jobs.

8 Future topics

- Check `http://www.depts.ttu.edu/hpcc/userguides/index.php` for HPCC users' guides.
- Check `http://crmda.ku.edu/computing` for the wiki page of the KU HPC.
- Check `http://winstat.quant.ku.edu/svn/hpcexample/trunk` for plenty of example scripts.