Computing on Clark and More Parallel Computing

J.D. Peiffer

2/7/2021

Section 1: Connecting to Clark with mobaXterm

1. Visit https://missouri.app.box.com/v/rcss-mobaxterm-new and download both files into your downloads folder.
2. Run the .exe file, accept all defaults for installation.
3. Open mobaXterm and click "Start Local Terminal" This will open a command interface for you to connect to Clark with.
4. Generate a ssh key:
   i) In your terminal type `ssh-keygen`.
   ii) Accept all defaults by pushing enter. Do not enter a password.
   iii) You only need to do this once!
5. Connect to Clark
   i) In your terminal, type `ssh <username>@clark.rnet.missouri.edu`
   Replace `<username>` with your pawprint.

```
11/02/2021 13:47.59 /home/mobaxterm ssh jdp6n8@clark.rnet.missouri.edu
Enter passphrase for key '/home/mobaxterm/.ssh/id_rsa':
```

   ii) Do not include brackets, for example `ssh jdp6n8@clark.rnet.missouri.edu`
   iii) When prompted for your password, enter your university password. **NOTE THAT WHAT YOU TYPE DOES NOT SHOW UP IN THE TERMINAL** You should then see something like this:

```
* ********************************************************************
* Welcome to clark-r630-login-node907.cobalt.lan, CentOS, 7.9.2009
* Archive of news is available in /etc/motd-archive
* University of Missouri Division of IT Research Computing Support Services
*
* Use of this system is governed by the rules and regulations of the
* University of Missouri and the University of Missouri System and also
* by requirements of various granting agencies.
*
* Users must be familiar with and abide by the UM System acceptable use
* policy (CRR 110.005) and the UM System Data Classification System (DCL).
* http://umsystem.edu/ums/rules/collected_rules/facilities/ch110
* http://umsystem.edu/ums/is/infosec/classification

* Helpful Links
* News: http://doit.missouri.edu/research/research-computing-news
* Support: rcss-support@missouri.edu
* Announcement list: https://po.missouri.edu/cgi-bin/wa?A0=rcss-announce-l
* Training: http://docs.rnet.missouri.edu/training
* Documentation: http://docs.rnet.missouri.edu/

* ********************************************************************
[jdp6n8@clark-r630-login-node907 ~]$
```

Section 2: Linux commands on Clark

Watch **this video** if you are not familiar with bash/Linux commands. These will be useful in mobaXterm and Clark.

Question 2.1 In your own words, provide a description of what each of the following bash terminal commands do (google is your friend):

- `pwd`
- `ls`
- `cd`
- `touch`
- `rm`
- `cat`
- `cp`
- `mv`
- `mkdir`

Question 2.2 In your home directory, create a folder named `hw4` and in it create a file name `test.txt`. Screenshot the output of `ls` in this directory (Hint: windows users hold `windows button + shift + s`)

When you log into Clark, you are logging into part of a computer cluster called a *node*. Every time, you start out in the *login node*. Here, you can perform basic tasks like making folders and files, but more data intensive tasks, like running R or NCL, must be done on a compute node.

To request a node for this task, type this into your terminal: `srun -n 1 --mem 8G --pty /bin/bash`

Notice the node name at your command line changes!

To load the rstudio software, type `module load rstudio/1.1.456`
Next, type `rstudio`
Congratulations! You are running R on Clark!

Data Transfer

1. Exit your rstudio session by clicking the "X" button. Exit your compute node by typing `exit` and exit Clark by typing `exit`.
2. You are now in your local computer's mobaXterm folder. We need to navigate to the hw4 data files on our computer and transfer them to Clark.
3. Type `cd /drives/c/Users`

4. Type `ls` to list available options and then `cd` followed by your username.

5. Type `cd Downloads` or navigate to the folder with your hw4 data.
   The general format of the data transfer command is `rsync -avhP DATA_FILE DESTINATION` where DATA_FILE is the name of your file and DESTINATION will be the address of Clark.

6. Type `rsync -avhP allprcp.Rdata <pawprint>@clark.rnet.missouri.edu:/home/<pawprint>/hw4/` where is your pawprint.

7. For code: `rsync -avhP hw4_code.R <pawprint>@clark.rnet.missouri.edu:/home/<pawprint>/hw4/`

Section 3: Load and Characterize Data

Make sure your working directory contains the file "allprcp.Rdata".

Load the variable into your workspace with `load("allprcp.Rdata")`

Question 3.1 What are the dimensions of this dataset? What is represented on the x and y dimension? (Hint: similar to hw3).
Question 3.2 What is the beginning and end date of this time series?
Question 3.3 What is the column range of the data? (Names begin with US....) Use this information to make a new data frame called rain.

```
data_start_column = #Enter your answer here
data_stop_column = #Enter your answer here

rain = allprcp[, data_start_column:data_stop_column]
```

Question 3.4 How many missing data entries are there in the entirety of this dataset? (Hint: `sum()` and `is.na()`)
Question 3.5 What percentage of the total data entries is NA?

That's a lot of missing data! Let us try and identify a suitable date to being our analysis. If we start at 1980 instead of 1970, there may be less missing data! Remember what we did in hw3.

Section 4: Quantify NAs in Serial

This section will use the same code as hw3, but since the Clark cluster has many cores, it allows us to highly parallelize this task!
First, let us run the code to quantify NAs with only one processor! This may take a **while**, it took me around 8 minutes on Clark.

```
addNA = function(data_with_some_nas, starting_row) {
  nas = is.na(data_with_some_nas[starting_row:dim(data_with_some_nas)[1], ])
  return(apply(nas, 2, sum))
}

NAs_in_rain = rain * 0
system.time({
  for (i in 1:dim(rain)[1]) {
    NAs_in_rain[i, ] = addNA(rain, i)
  }
})
```

Question 4.1 How long did this take to take to run on Clark? (Do not try on your personal computer).

Section 5: Quantify NAs in Parallel

Exit your rstudio session and type `exit` in your command prompt. This will exit your compute node. To run parallel jobs, we need to request more cores!

Type `srun -n 8 --mem 8G --pty /bin/bash`. Load rstudio and your data as done previously.
Now we will install the parallel package and parallelize our code.

```
install.packages("doParallel")
library(doParallel)

registerDoParallel()
getDoParWorkers()
```

Question 5.1 What is the output of getDoParWorkers()? What does this represent?

Now, run this task using two cores!

```
registerDoParallel(cores=2)
system.time({
  NAs_in_rain_par = foreach(i = 1:dim(allprcp)[1],
                            .export = c("addNA", "rain"),
                            .combine = "rbind") %dopar% {
    addNA(rain, i)
  }
})
```

Question 5.2 What is the elapsed time of this code? Is it faster than our earlier serial code? Why is this?
Question 5.3 Run the same code with 4 cores, 6 cores, and 8 cores. Record and plot the output with cores on the x axis and time on the y. Use any (R) plotting method you like.
Question 5.4 Describe your graph.

Section 6: Plotting Data

We are finally ready to view our data and choose at what date to start our analysis.

```
library(ggplot2)
sumgt = function(x) {
  #simply adds up number of stations with less than lessThan missing days
  #this is what we will consider to be "too many" missing days
  lessThan = 500
  return(sum(x < lessThan))
}
lthan = 500

missings = apply(NAs_in_rain_par8, 1, sumgt)
missings = data.frame(missings, date = as.Date(allprcp$date))   #plots things
ggplot(data = missings, aes(x = date, y = missings, group = 1)) + geom_line(col =
                                                                              "blue", size = 2) + scale_
  labs(
    x = "Date of cutoff",
```

```
    y = paste('Sites with less than', lthan, '\nmissing days'),
    title = "Cutoff date vs Missing Days"
  ) +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14, face = "bold"))
```

Question 6.1 Where would you begin your analysis?