CSC 555: Mining Big Data

Assignment 1 (due Sunday, April 14th)

Lavinia Wang #1473704

Part 1

a) Compute (you can use any tool you wish – but if you do not know to perform this computation, please talk to me about prerequisites):

$2^{10} = 1024$
$4^5 = 1024$
$8^5 = 32768$
837 MOD 100 (MOD is the modulo operator, a.k.a. the remainder) = 37
842 MOD 20 = 2
16 MOD 37 = 16
37 MOD 16 = 5

b) Given vectors V1 = (1, 2, 3) and V2 = (2, 1, 2) and a 3x3 matrix M = [(2, 1, 3), (1, 2, 2), (1, 0, 2)], compute:

V2 – V1 = [1, -1, -1]
V1 + V1 = [2, 4, 6]
|V1| = sqrt($1^2 + 2^2 + 3^2$) = 3.7416573867739413
|V2| = sqrt($2^2 + 1^2 + 2^2$) = 3.0

$$M * V2 = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 11 \\ 8 \\ 6 \end{bmatrix}$$

$$M * M \text{ (or } M^2) = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 8 & 4 & 14 \\ 6 & 5 & 11 \\ 4 & 1 & 7 \end{bmatrix}$$

$$M^3 = \begin{bmatrix} 8 & 4 & 14 \\ 6 & 5 & 11 \\ 4 & 1 & 7 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 34 & 16 & 60 \\ 28 & 16 & 50 \\ 16 & 6 & 28 \end{bmatrix}$$

c) Suppose we are flipping a coin with Head (H) and Tail (T) sides. The coin is <u>not</u> balanced with 0.6 probability of H coming up (and 0.4 of T). Compute the probabilities of getting:

P(HTHT) = $0.6 * 0.4 * 0.6 * 0.4 = 0.0576$
P(THTT) = $0.4 * 0.6 * 0.4 * 0.4 = 0.0384$
Exactly 1 Head out of a sequence of 4 coin flips. $P = \binom{4}{1} \times (0.6 * 0.4^3) = 0.1536$
Exactly 1 Tail out of sequence of 4 coin flips. = $P = \binom{4}{1} \times (0.4 * 0.6^3) = 0.3456$

d) Consider a database schema consisting of two tables, Employee (<u>ID</u>, Name, Address), Project (<u>PID</u>, Name, Deadline), Assign(<u>EID</u>, <u>PID</u>, Date). Assign.EID is a foreign key referencing employee's ID and Assign.PID is a foreign key referencing the project.

Write SQL queries for:

i. Find unassigned projects (PID and Name of the project).

```
SELECT PID, Name
FROM Project p
WHERE NOT EXISTS (
        SELECT 1
        FROM Assign a
        WHERE a.PID = p.PID)
```

ii. Find employees that are assigned to more than 3 projects.

```
SELECT e.ID, e.Name, e.Address, COUNT(a.PID) AS ProjectCount
FROM Employee e
INNER JOIN Assign a ON a.EID = e.ID
GROUP BY e.ID, e.Name, e.Address
HAVING COUNT(a.PID) >= 3
```

iii. Find all projects that have fewer than 4 employees assigned to them (note that this should include 3, 2, 1 and 0 in order to be correct)

```
SELECT p.PID, p.Name, p.Deadline, COUNT(a.PID) AS EmployeeCount
FROM Project p
LEFT JOIN Assign a ON a.PID = p.PID
GROUP BY p.PID, p.Name, p.Deadline
HAVING COUNT(a.PID) <= 3
```

e) Mining of Massive Datasets, Exercise 1.3.3
Suppose hash-keys are drawn from the population of all non-negative integers that are multiples of some constant c, and hash function h(x) is x mod 15. For what values of c will h be a suitable hash function, i.e., a large random choice of hash-keys will be divided roughly equally into buckets?

<u>Justify your answer</u> – simply naming a constant will not provide credit.

Consider the hash function, h(x) = x mod 15. Since h(x) can give only values from 0 to 15, we can have only 15 buckets numbered from 0, 1…14. For c, all nonnegative integers will work except for 0, 3, 5, and their multiples. Meaning in this case, c cannot be a factor or multiple of 15, nor can it have a shared factor with 15. Examples of unacceptable values for c are 0, 3, 5, 6, 9, 10, 12, 15, and multiples of these numbers.

Acceptable values include 1, 2, 4, 7, and any other number not divisible by 3 or 5.

f) Describe how you would implement a MapReduce job consisting of Map and Reduce description. You <u>do not</u> have to write code or even pseudo-code. Just describe, in your own words, what the Map and Reduce tasks are going to do. Map task reads the input file and produces (key, value) pairs. Reduce task takes a list of (key, value) pairs for each key and combines all values for each key.
Please remember that Map operates on individual blocks and Reduce on individual keys with a set of values. Thus, for Mapper you need to state what your code does given a

block of data (i.e., for each block, not for the whole file) and for Reduce you need to state what your reducer does for each key (without being able to see other keys).
For a data file that contains the following columns: (ID, First, Last, Grade)

i. For each first name, find the GPA (grade point average) of each student, i.e., **SELECT First, AVG(Grade) FROM Student GROUP BY First**.

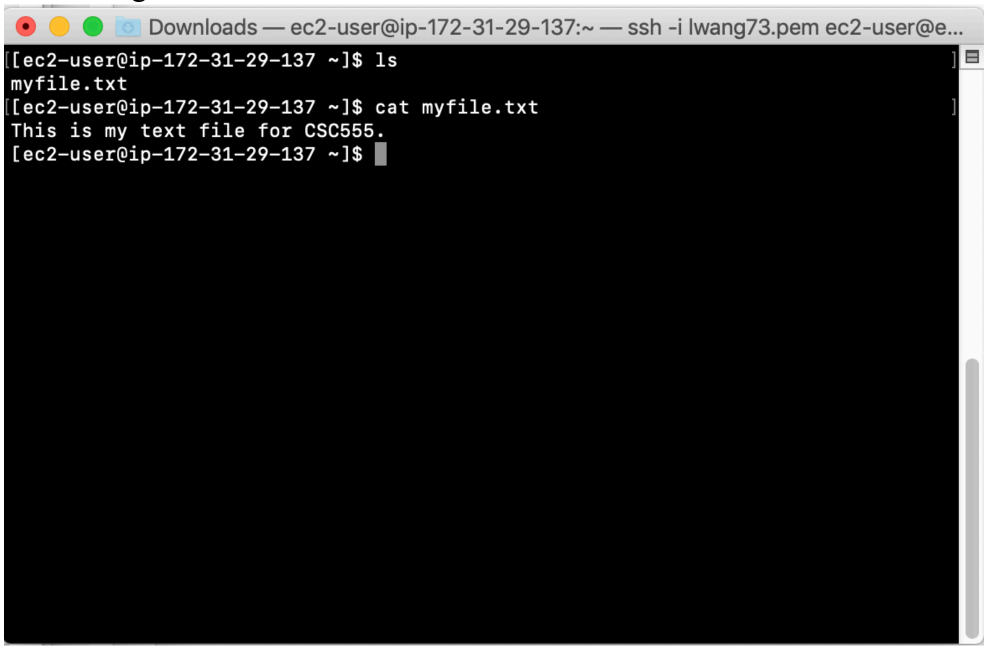
   For each block, the mapper will use the First column to determine the key and pair the associated Grade value with the key. The reducer(s) will calculate the average grade values for each of the keys, i.e. first name.

ii. For each full student name, find the best grade, i.e., **SELECT First, Last, MAX(Grade) FROM Student GROUP BY First, Last**.

   For each block of data, the mapper(s) will use both the First and Last columns as a combined key and pair the combination with the associated Grade value for that pair. The reducer(s) will evaluate the Grade values for each of the keys and determine the maximum grade for each key, i.e. first and last name.

Part 2: Linux Intro

1. **Create a text file.**
   Instructions for 3 different text editors are provided below. You only need to choose one editor that you prefer. **nano** is a more basic text editor, and is much easier to start. **vim** and **emacs** are more advanced and rely on keyboard shortcuts quite a bit and thus have a steeper learning curve.

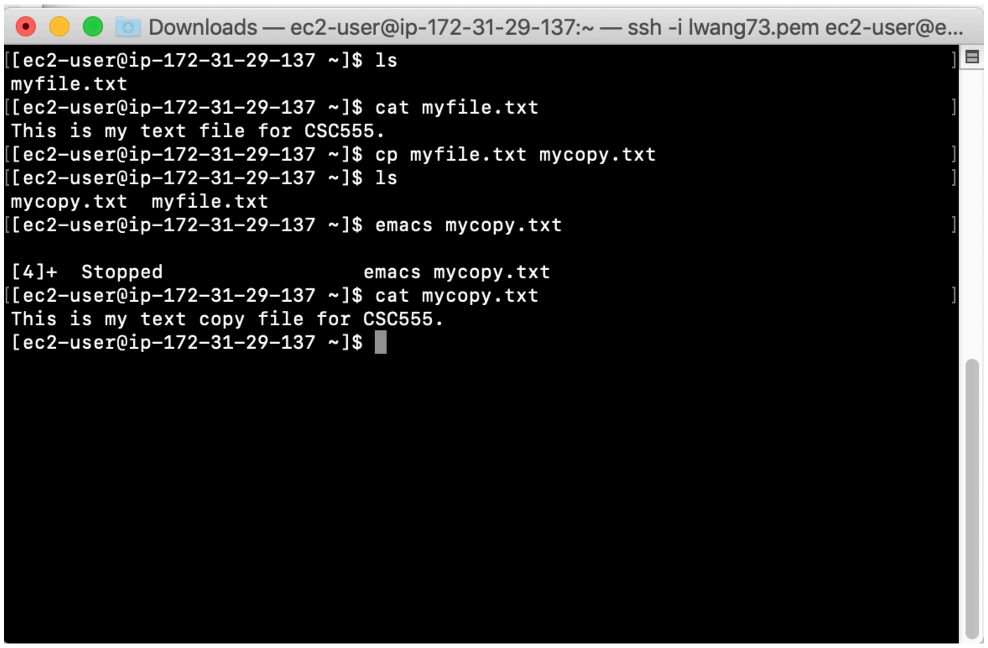

Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
[ec2-user@ip-172-31-29-137 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-29-137 ~]$
```

2. Copy your file.

Lavinia Wang #1473704

Make a copy.
$ cp myfile.txt mycopy.txt
Confirm this file has been created by listing the files in the working directory.
Edit this file so it contains different text than the original file using the text editor instructions, and confirm your changes by displaying the contents of the file on the screen.


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
[ec2-user@ip-172-31-29-137 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-29-137 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-29-137 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ emacs mycopy.txt

[4]+  Stopped                 emacs mycopy.txt
[ec2-user@ip-172-31-29-137 ~]$ cat mycopy.txt
This is my text copy file for CSC555.
[ec2-user@ip-172-31-29-137 ~]$ 
```

3. Delete a file

Make a copy to delete.
$ cp myfile.txt filetodelete.txt
$ ls

Remove the file.
$ rm filetodelete.txt
$ ls

4. Create a directory to put your files.

Make a directory.
$mkdir CSC555

Change the current directory to your new directory.
$cd CSC555

Print your current working directory
$pwd

Lavinia Wang #1473704

5. Move your files to your new directory.

Return to your home directory.
$cd
OR
$cd ..
OR
$ cd /home/ec2-user/

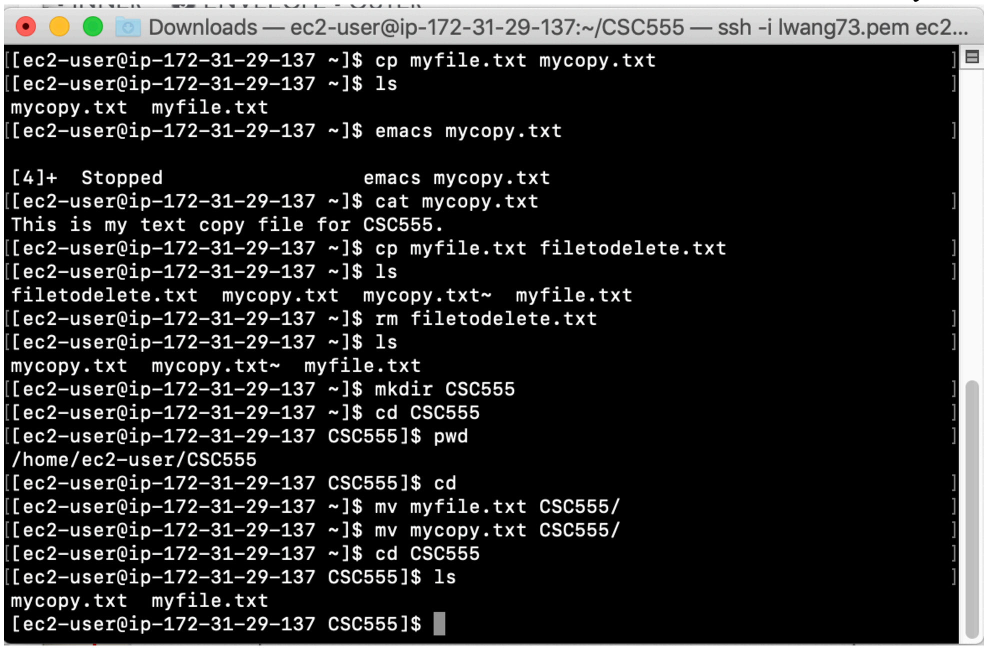
• NOTE: **cd** will always take you to your home directory. **cd ..** will move you up one directory level (to the parent). Your home directory is "/home/[user name]", /home/ec2-user in our case

Move your files to your new directory.

$ mv myfile.txt CSC555/
$ mv mycopy.txt CSC555/

Change the current directory to CSC555 and list the files in this directory.

SUBMIT: Take a screen shot of the files listed in the CSC555 directory.


Downloads — ec2-user@ip-172-31-29-137:~/CSC555 — ssh -i lwang73.pem ec2...

```
[ec2-user@ip-172-31-29-137 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-29-137 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ emacs mycopy.txt

[4]+  Stopped                 emacs mycopy.txt
[ec2-user@ip-172-31-29-137 ~]$ cat mycopy.txt
This is my text copy file for CSC555.
[ec2-user@ip-172-31-29-137 ~]$ cp myfile.txt filetodelete.txt
[ec2-user@ip-172-31-29-137 ~]$ ls
filetodelete.txt  mycopy.txt  mycopy.txt~  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-29-137 ~]$ ls
mycopy.txt  mycopy.txt~  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ mkdir CSC555
[ec2-user@ip-172-31-29-137 ~]$ cd CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ cd
[ec2-user@ip-172-31-29-137 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-29-137 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-29-137 ~]$ cd CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-29-137 CSC555]$ 
```

6. Zip and Unzip your files.

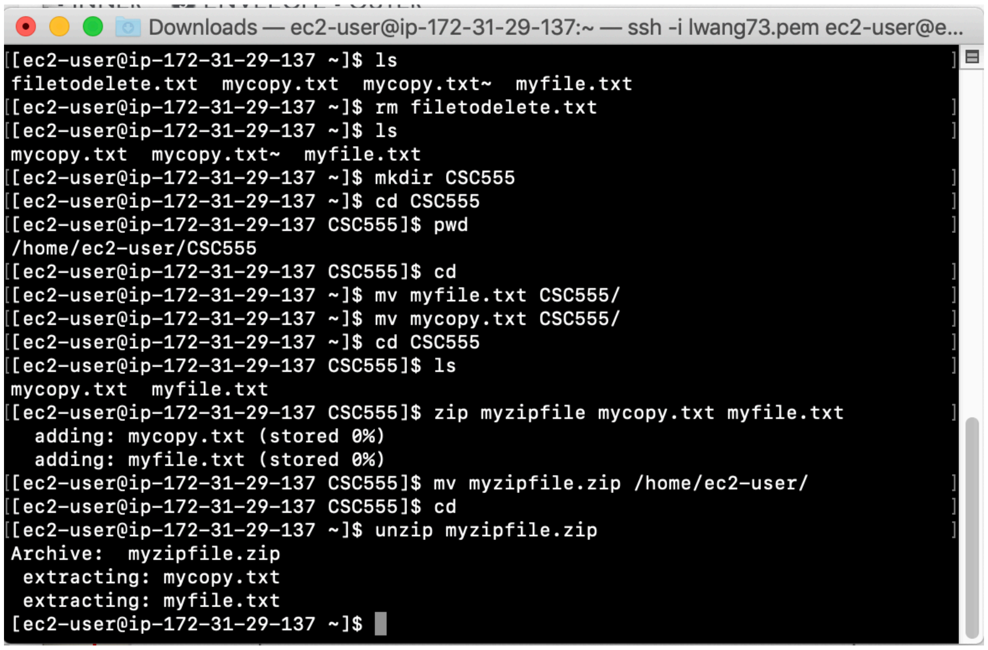
Zip the files.
$ zip myzipfile mycopy.txt myfile.txt
OR
$ zip myzipfile *

• NOTE: * is the wildcard symbol that matches everything in current directory. If there should be any additional files in the current directory, they will also be placed into the zip archive. Wildcard can also be used to match files selectively. For example **zip myzipfile my*** will zip-up all files beginning with "my" in the current directory.

Move your zip file to your home directory.
$ mv myzipfile.zip /home/ec2-user/

Return to your home directory.
Extract the files.
$ unzip myzipfile.zip

SUBMIT: Take a screen shot of the screen after this command.


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
[ec2-user@ip-172-31-29-137 ~]$ ls
filetodelete.txt  mycopy.txt  mycopy.txt~  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-29-137 ~]$ ls
mycopy.txt  mycopy.txt~  myfile.txt
[ec2-user@ip-172-31-29-137 ~]$ mkdir CSC555
[ec2-user@ip-172-31-29-137 ~]$ cd CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ cd
[ec2-user@ip-172-31-29-137 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-29-137 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-29-137 ~]$ cd CSC555
[ec2-user@ip-172-31-29-137 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-29-137 CSC555]$ zip myzipfile mycopy.txt myfile.txt
  adding: mycopy.txt (stored 0%)
  adding: myfile.txt (stored 0%)
[ec2-user@ip-172-31-29-137 CSC555]$ mv myzipfile.zip /home/ec2-user/
[ec2-user@ip-172-31-29-137 CSC555]$ cd
[ec2-user@ip-172-31-29-137 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
 extracting: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-29-137 ~]$
```

7. Remove your CSC555 directory.

• **Warning**: Executing "rm -rf" has the potential to delete ALL files in a given directory, including sub-directories ("r" stands for recursive). You should use this command very carefully.
Delete your CSC555 directory.
$ rm -rf CSC555/

8. Download a file from the web.

Download the script for Monty Python and the Holy Grail.
$ wget http://www.textfiles.com/media/SCRIPTS/grail

The file should be saved as “grail” by default.

9. ls formats

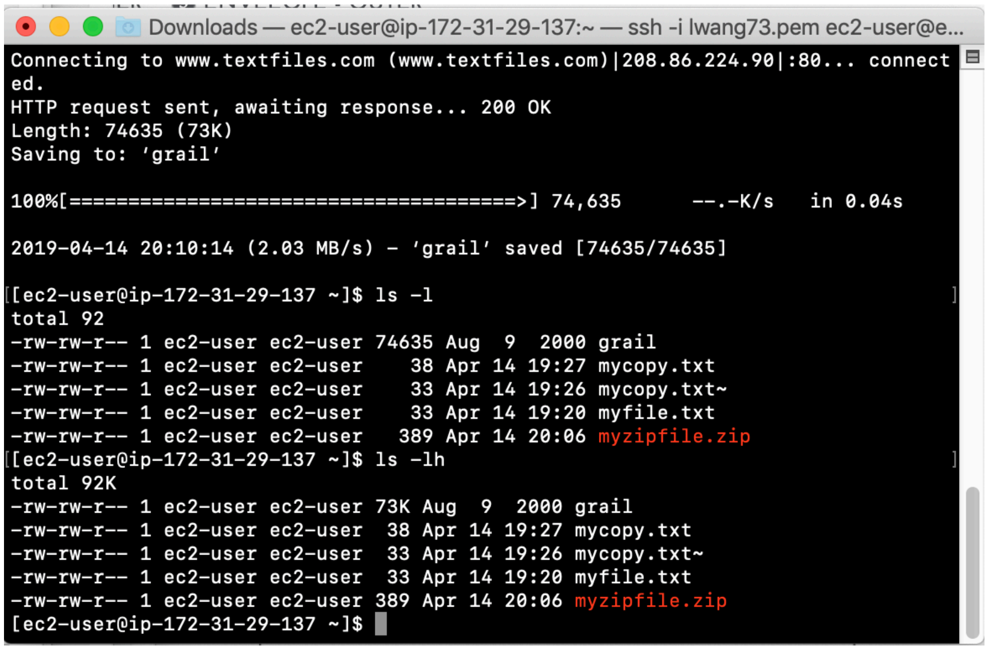
List all contents of the current directory in long list format.
• **Note**: the option following “ls” is the character “l”; not “one”.
$ ls -l

The 1st column gives information regarding file permissions (which we will discuss in more detail later). For now, note that the first character of the 10 total will be “-“ for normal files and “d” for directories. The 2nd Column is the number of links to the file. The 3rd and 4th columns are the owner and the group of the file. The 5th column displays the size of the file in bytes. The 6th column is the date and time the file was last modified. The 7th column is the file or directory name.

List all contents of the current directory in long list and human readable formats. “-h” will put large files in more readable units than bytes.
$ ls -lh

SUBMIT: The size of the grail file.

74,635 bytes or 73K


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
Connecting to www.textfiles.com (www.textfiles.com)|208.86.224.90|:80... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 74635 (73K)
Saving to: ‘grail’

100%[=====================================>] 74,635      --.-K/s   in 0.04s

2019-04-14 20:10:14 (2.03 MB/s) - ‘grail’ saved [74635/74635]

[ec2-user@ip-172-31-29-137 ~]$ ls -l
total 92
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user    38 Apr 14 19:27 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user    33 Apr 14 19:26 mycopy.txt~
-rw-rw-r-- 1 ec2-user ec2-user    33 Apr 14 19:20 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   389 Apr 14 20:06 myzipfile.zip
[ec2-user@ip-172-31-29-137 ~]$ ls -lh
total 92K
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user  38 Apr 14 19:27 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Apr 14 19:26 mycopy.txt~
-rw-rw-r-- 1 ec2-user ec2-user  33 Apr 14 19:20 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 389 Apr 14 20:06 myzipfile.zip
[ec2-user@ip-172-31-29-137 ~]$
```

10. More on viewing files.

If you issue “cat grail”, the contents of grail will be printed. However, this file is too large to fit on the screen.

Show the grail file one page at a time.

```
$ more grail
```

Hit the spacebar to go to the next page. Type "b" to go page up, hit "space" key to go page down. Type "q" to quit.

OR

```
$ less grail
```

Less has more options than *more* ("*less* is more and *more* is less"). You can now use the keyboard Arrows and Page Up/Down to scroll. You can type "h" for help, which will display additional options.

View the line numbers in the grail file. The *cat* command has the -n option, which prints line numbers, but you may also want to use *more* to view the file one page at a time. A solution is to *pipe* the output from *cat* to *more*. A *pipe* redirects the output from one program to another program for further processing, and it is represented with "|".

```
$ cat -n grail | more
```

Redirect the standard output (stdout) of a command.

```
$ cat myfile.txt > redirect1.txt
$ ls -lh > redirect2.txt
```

Append the stdout to a file.

```
$ cat mycopy.txt >> myfile.txt
```

mycopy.txt will be appended to myfile.txt.

• **Note**: "cat mycopy.txt > myfile.txt" will overwrite myfile.txt with the contents output by "cat mycopy.txt". Thus using >> is crucial if you want to preserve the existing file contents.

11. Change access permissions to objects with the *change mode* command.

The following represent roles:
u – user, g – group, o – others, a - all

The following represent permissions:
r – read, w – write, x – execute

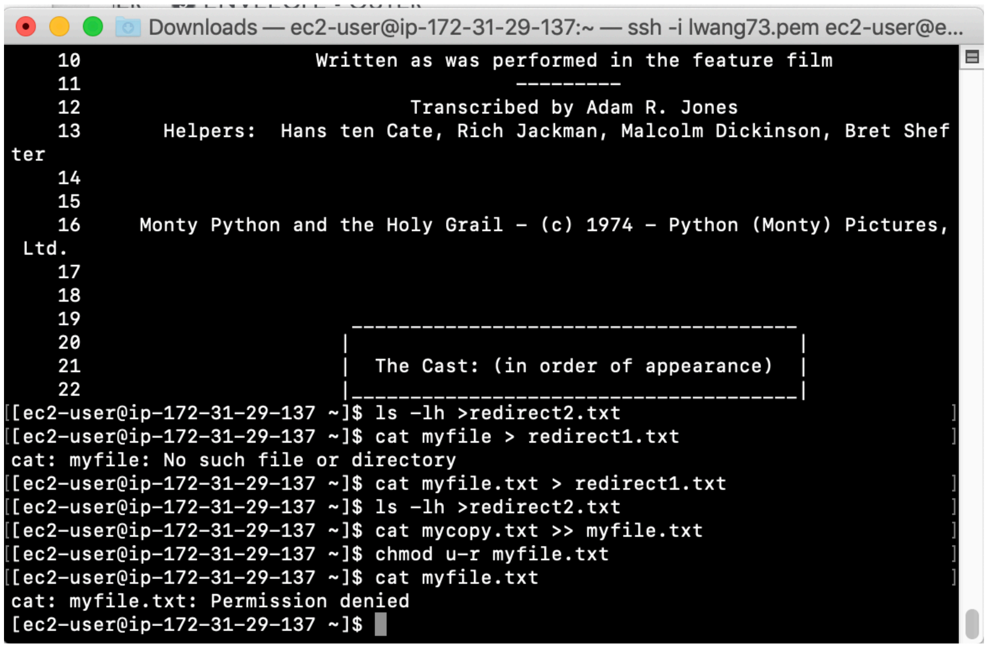
Remove the read permission for your user on a file.

```
$ chmod u-r myfile.txt
```

Try to read this file. You should receive a "permission denied" message because you are the user who owns the file.

SUBMIT: The screenshot of the permission denied error

Lavinia Wang #1473704


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
    10                    Written as was performed in the feature film
    11                                      ---------
    12                            Transcribed by Adam R. Jones
    13       Helpers:  Hans ten Cate, Rich Jackman, Malcolm Dickinson, Bret Shef
ter
    14
    15
    16     Monty Python and the Holy Grail - (c) 1974 - Python (Monty) Pictures,
 Ltd.
    17
    18
    19                             ______________________________________
    20                            |                                      |
    21                            |  The Cast: (in order of appearance)  |
    22                            |______________________________________|
[[ec2-user@ip-172-31-29-137 ~]$ ls -lh >redirect2.txt
[[ec2-user@ip-172-31-29-137 ~]$ cat myfile > redirect1.txt
cat: myfile: No such file or directory
[[ec2-user@ip-172-31-29-137 ~]$ cat myfile.txt > redirect1.txt
[[ec2-user@ip-172-31-29-137 ~]$ ls -lh >redirect2.txt
[[ec2-user@ip-172-31-29-137 ~]$ cat mycopy.txt >> myfile.txt
[[ec2-user@ip-172-31-29-137 ~]$ chmod u-r myfile.txt
[[ec2-user@ip-172-31-29-137 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
[ec2-user@ip-172-31-29-137 ~]$
```

Give your user read permission on a file. Use the same file you removed the read permission from.

$ chmod u+r myfile.txt

You should now be able to read this file again.

12. Python examples

Install Python if it is not available on your machine.

$ sudo yum install python

Create a Python file. These instructions will use Emacs as a text editor, but you can still chose the text editor you want.

$ emacs lucky.py

(Write a simple Python program)

```
print "*"*22
print "My Lucky Numbers".rjust(20)
print "*"*22

for i in range(10):
   lucky_nbr = (i + 1)*2
   print "My lucky number is %s!" % lucky_nbr
```

Run your Python program.

$ python lucky.py

Redirect your output to a file

Lavinia Wang #1473704

```
$ python lucky.py > lucky.txt
```

Pipe the stdout from lucky.py to another Python program that will replace “is” with “was”.

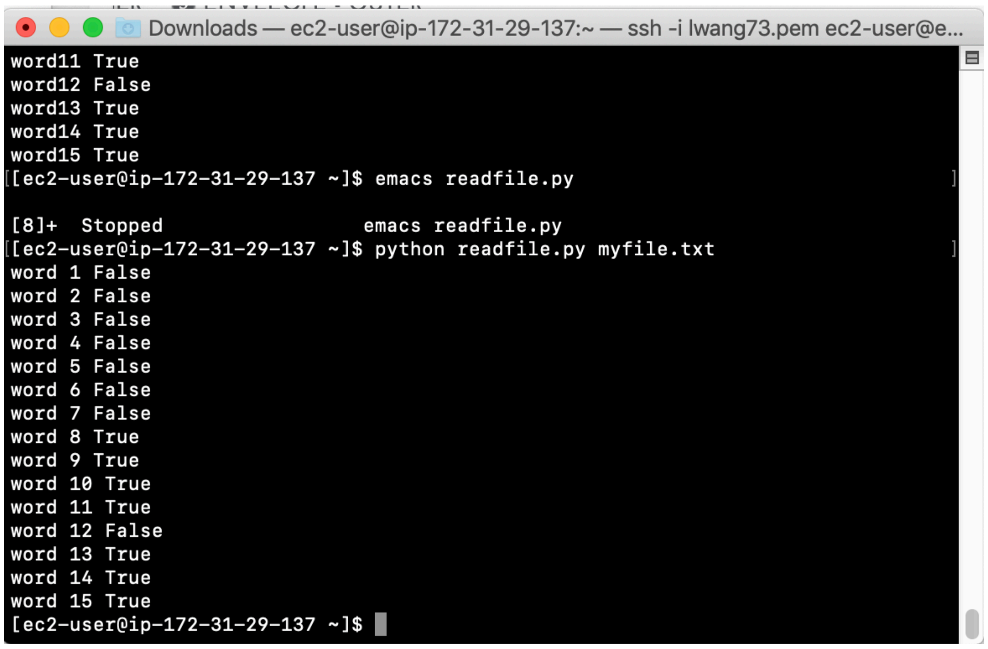
```
$ emacs was.py
```

```
import sys

for line in sys.stdin:
   print line.replace("is", "was")
```

```
$ python lucky.py | python was.py
```

Write python code to read a text file (you can use myfile.txt) and output word count for each word with the number of times that word occurs in the entire file.

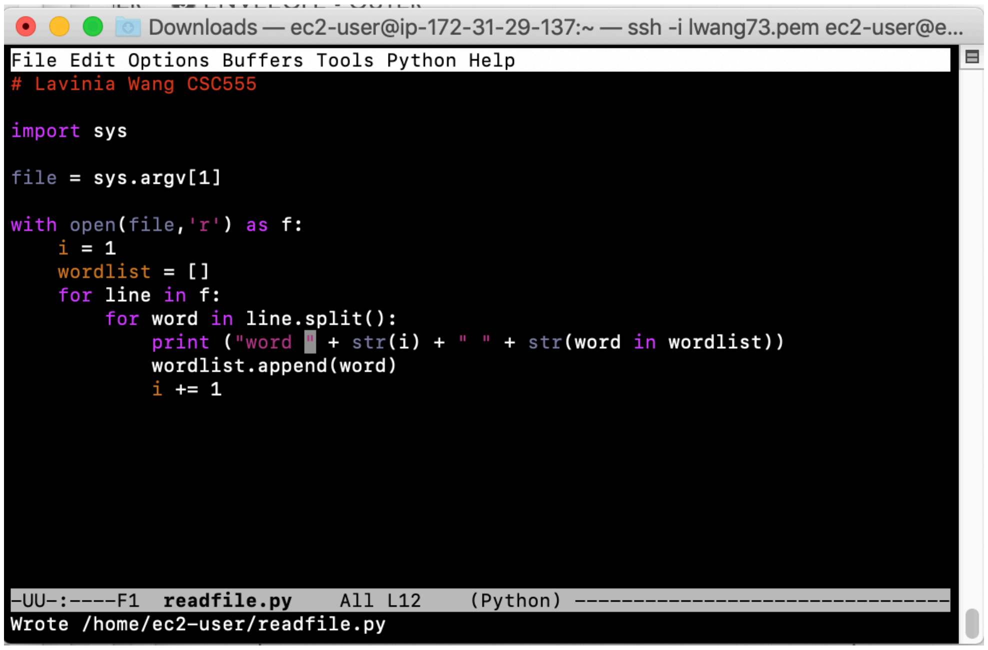
SUBMIT: The screen output from running your python code and a copy of your python code. Homework submissions without code will receive no credit.



Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
word11 True
word12 False
word13 True
word14 True
word15 True
[ec2-user@ip-172-31-29-137 ~]$ emacs readfile.py

[8]+  Stopped                 emacs readfile.py
[ec2-user@ip-172-31-29-137 ~]$ python readfile.py myfile.txt
word 1 False
word 2 False
word 3 False
word 4 False
word 5 False
word 6 False
word 7 False
word 8 True
word 9 True
word 10 True
word 11 True
word 12 False
word 13 True
word 14 True
word 15 True
[ec2-user@ip-172-31-29-137 ~]$ 
```

Lavinia Wang #1473704


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...
File Edit Options Buffers Tools Python Help

```
# Lavinia Wang CSC555

import sys

file = sys.argv[1]

with open(file,'r') as f:
    i = 1
    wordlist = []
    for line in f:
        for word in line.split():
            print ("word " + str(i) + " " + str(word in wordlist))
            wordlist.append(word)
            i += 1

-UU-:----F1  readfile.py    All L12    (Python) ------------------------------
Wrote /home/ec2-user/readfile.py
```

```
# Lavinia Wang CSC555

import sys

file = sys.argv[1]

with open(file,'r') as f:
   i = 1
   wordlist = []
   for line in f:
      for word in line.split():
        print("word " + str(i) + " " +  str(word in wordlist))
        wordlist.append(word)
        i+=1
```

Part 3: Lab

For this part of the assignment, you will run wordcount on a single-node Hadoop instance. I am going to provide detailed instructions to help you get Hadoop running. The instructions are following Hadoop: The Definitive Guide instructions presented in Appendix A: Installing Apache Hadoop.

You can download 2.6.4 from here. You can copy-paste these commands (right-click in PuTTy to paste, but please watch out for error messages and run commands one by one)

Install ant to list java processes
sudo yum install ant

(wget command stands for "web get" and lets you download files to your instance from a URL link)
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hadoop-2.6.4.tar.gz

(unpack the archive)
tar xzf hadoop-2.6.4.tar.gz

Modify the conf/hadoop-env.sh to add to it the JAVA_HOME configuration
You can open it by running (using nano or your favorite editor instead of nano).
nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh
Note that the # comments out the line, so you would comment out the original JAVA_HOME line replacing it by the new one as below.

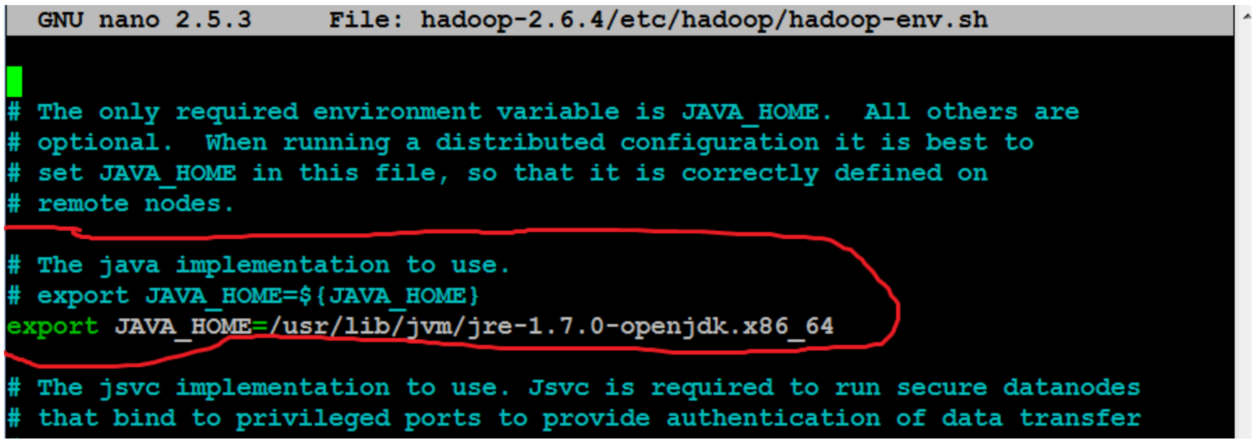
NOTE: you would need to determine the correct Java configuration line by executing the following (underlined) command

```
[ec2-user@ip-172-31-16-63 ~]$ readlink -f $(which java)
```

which will output:

```
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/bin/java
```

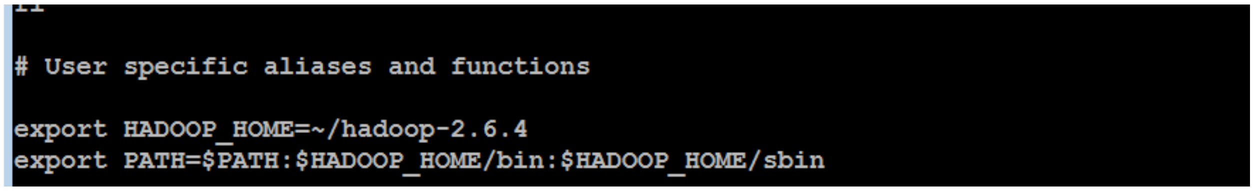
In my case, Java home is at (remove the bin/java from the output above):
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/



```
  GNU nano 2.5.3      File: hadoop-2.6.4/etc/hadoop/hadoop-env.sh


# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
# export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
```

modify the .bashrc file to add these two lines:
export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

.bashrc file contains environment settings to be configured automatically on each login.
You can open the .bashrc file by running
nano ~/.bashrc



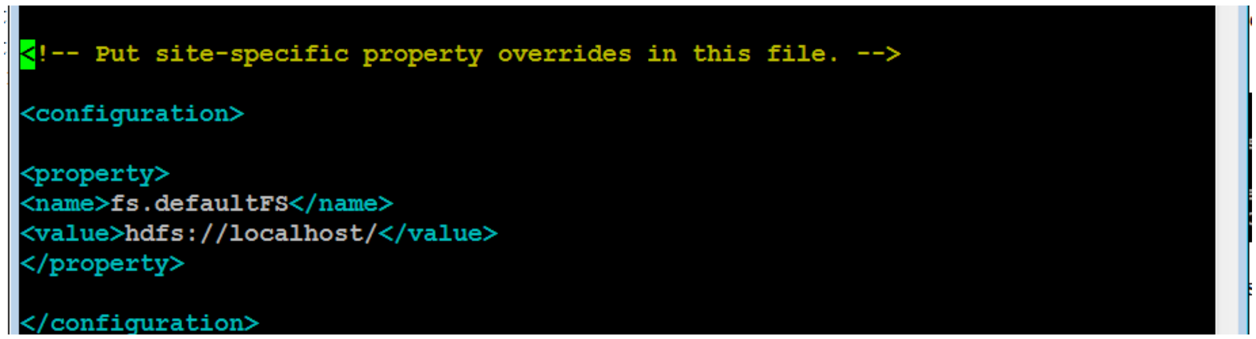
```
# User specific aliases and functions

export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

To immediately refresh the settings (that will be automatic on next login), run
source ~/.bashrc

Next, follow the instructions for Pseudodistributed Mode for all 4 files.

(to edit the first config file)
nano hadoop-2.6.4/etc/hadoop/core-site.xml

Make sure you paste the settings between the <configuration> and </configuration> tags, like in the screenshot below. NOTE: The screenshot below is only one of the 4 files, all files are different. The contents of each file are described in the **Appendix A** in the Hadoop book, the relevant appendix is also included with the homework assignment. I am also including a .txt file (HadoopConfigurationText) so that it is easier to copy-paste.



```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost/</value>
</property>

</configuration>
```

nano hadoop-2.6.4/etc/hadoop/hdfs-site.xml
(mapred-site.xml file is not there, run the following single line command to create it by copying from template. Then you can edit it as other files.)
cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml
nano hadoop-2.6.4/etc/hadoop/mapred-site.xml
nano hadoop-2.6.4/etc/hadoop/yarn-site.xml

To enable passwordless ssh access (we will discuss SSH and public/private keys in class), run these commands:
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

test by running (and confirming yes to a one-time warning)
ssh localhost
exit

Lavinia Wang #1473704

Format HDFS (i.e., first time initialize)

```
hdfs namenode -format
```

Start HDFS, Hadoop and history server (answer a 1-time yes if you asked about host authenticity)

```
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
```

Verify if everything is running:

```
jps
```

(NameNode and DataNode are responsible for HDFS management; NodeManager and ResourceManager are serving the function similar to JobTracker and TaskTracker. We will discuss function of all of those on Thursday.)

Create a destination directory
hadoop fs -mkdir /data

Download a large text file using

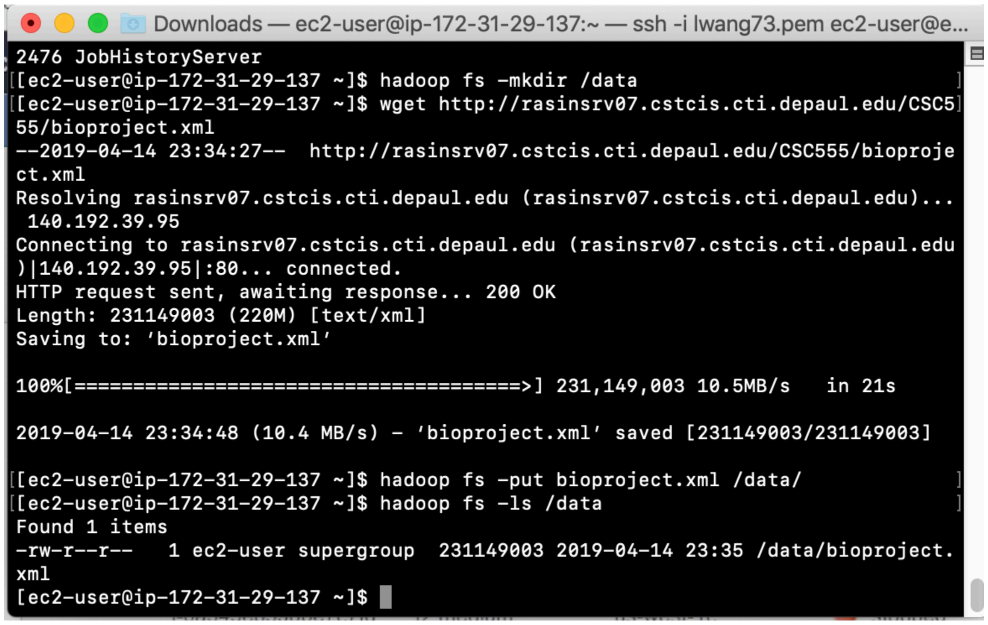
```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproject.xml
```

Copy the file to HDFS for processing

```
hadoop fs -put bioproject.xml /data/
```

(you can optimally verify that the file was uploaded to HDFS by **hadoop fs -ls /data**)
Submit a screenshot of this command

Lavinia Wang #1473704

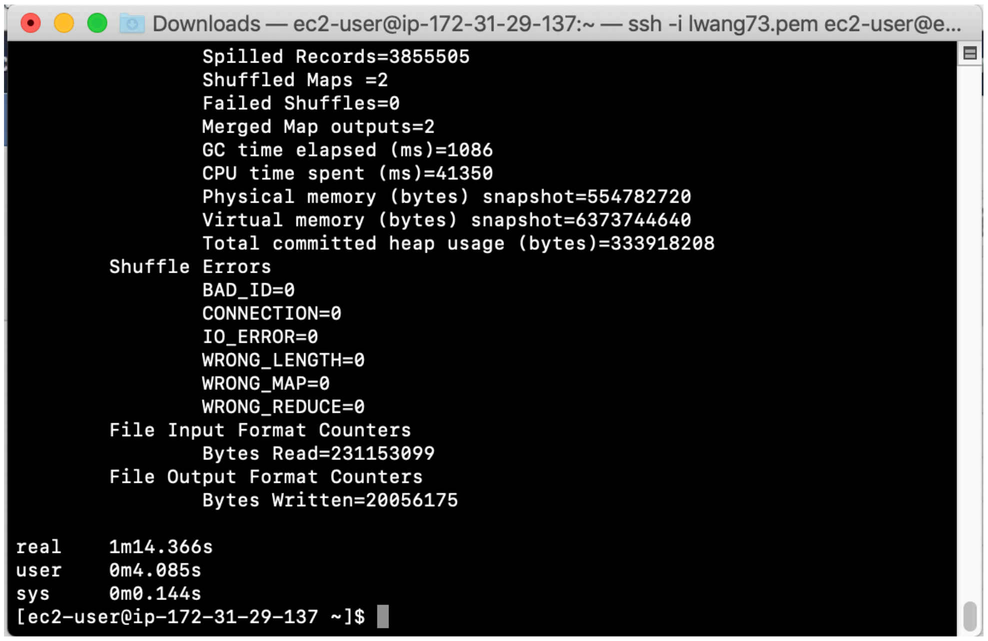

Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
2476 JobHistoryServer
[ec2-user@ip-172-31-29-137 ~]$ hadoop fs -mkdir /data
[ec2-user@ip-172-31-29-137 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC5
55/bioproject.xml
--2019-04-14 23:34:27--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproje
ct.xml
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)...
 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu
)|140.192.39.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231149003 (220M) [text/xml]
Saving to: 'bioproject.xml'

100%[=====================================>] 231,149,003 10.5MB/s   in 21s

2019-04-14 23:34:48 (10.4 MB/s) - 'bioproject.xml' saved [231149003/231149003]

[ec2-user@ip-172-31-29-137 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-29-137 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   1 ec2-user supergroup  231149003 2019-04-14 23:35 /data/bioproject.
xml
[ec2-user@ip-172-31-29-137 ~]$
```

Run word count on the downloaded text file, using the time command to determine the total runtime of the MapReduce job.   You can use the following (single-line!) command. This invokes the wordcount example built into the example jar file, supplying /data/bioproject.xml as the input and /data/wordcount1 as the output directory. Please remember this is <u>one command</u>, if you do not paste it as a single line, it will not work.

time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar wordcount /data/bioproject.xml /data/wordcount1

Report the time that the job took to execute as screenshot


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

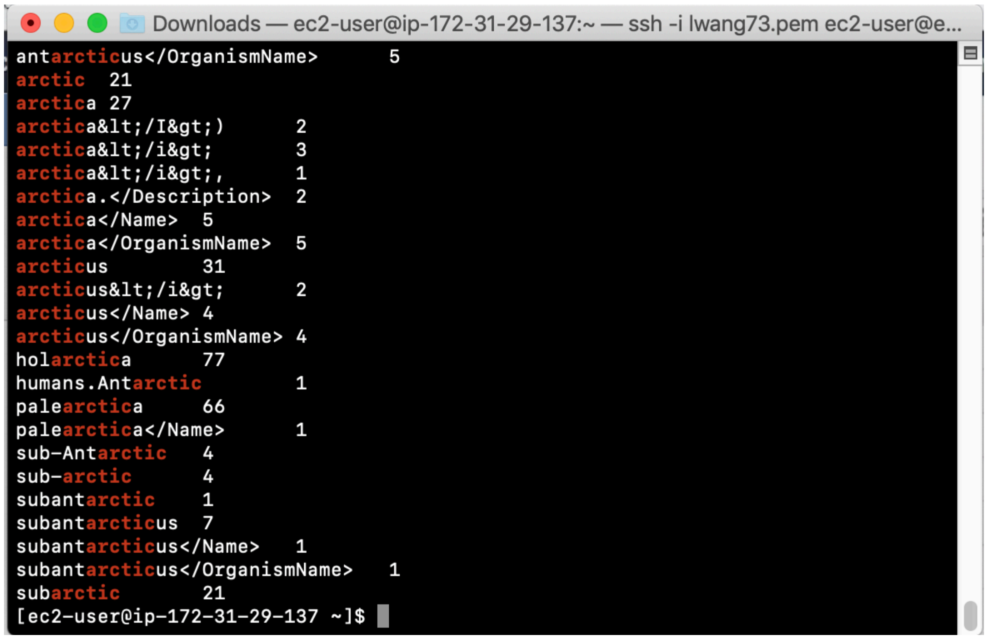
```
                Spilled Records=3855505
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=1086
                CPU time spent (ms)=41350
                Physical memory (bytes) snapshot=554782720
                Virtual memory (bytes) snapshot=6373744640
                Total committed heap usage (bytes)=333918208
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=231153099
        File Output Format Counters
                Bytes Written=20056175

real    1m14.366s
user    0m4.085s
sys     0m0.144s
[ec2-user@ip-172-31-29-137 ~]$
```

(this reports the size of a particular file or directory in HDFS. The output file will be named part-r-00000)
hadoop fs -du /data/wordcount1/

(Just like in Linux, the cat HDFS command will dump the output of the entire file and grep command will filter the output to all lines that matches this particular word). To determine the count of occurrences of "arctic", run the following command:

hadoop fs -cat /data/wordcount1/part-r-00000 | grep arctic

It outputs the entire content of part-r-00000 file and then uses pipe | operator to filter it through grep (filter) command. If you remove the pipe and grep, you will get the entire word count content dumped to screen, similar to cat command.


Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
antarcticus</OrganismName>      5
arctic  21
arctica 27
arctica&lt;/I&gt;)       2
arctica&lt;/i&gt;        3
arctica&lt;/i&gt;,       1
arctica.</Description>  2
arctica</Name>  5
arctica</OrganismName>  5
arcticus        31
arcticus&lt;/i&gt;       2
arcticus</Name> 4
arcticus</OrganismName> 4
holarctica      77
humans.Antarctic        1
palearctica     66
palearctica</Name>      1
sub-Antarctic   4
sub-arctic      4
subantarctic    1
subantarcticus  7
subantarcticus</Name>   1
subantarcticus</OrganismName>   1
subarctic       21
[ec2-user@ip-172-31-29-137 ~]$ 
```

Congratulations, you just finished running wordcount using Hadoop.

<u>Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 1" at the top.</u>