

## CSC 555 / DSC 333

### Assignment 1 (due Friday, April 9<sup>th</sup>)

Name: Serena Yang

Suggested reading: *Mining of Massive Datasets*: Chapter 1, Chapter 2 (sections 2.1, 2.1 only).

*Hadoop: The Definitive Guide*: Appendix A (file also available on D2L).

Supplemental document **UsingAmazonAWS.doc**.

## Part 1

- a) Compute (you can use any tool to compute answers in this part – but if you do not know to perform this computation, please talk to me about your prerequisites):

$$2^{12} = 4096$$

$$(2^4)^4 = 65536$$

$$4^4 = 256$$

$$8^5 = 32768$$

$$837 \text{ MOD } 100 \text{ (MOD is the modulo operator, a.k.a. the remainder)} = 37$$

$$842 \text{ MOD } 20 = 2$$

$$22 \text{ MOD } 111 = 22$$

$$111 \text{ MOD } 22 = 1$$

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

$$V2 - V1 = (0, 1, -1)$$

$$V1 + V1 = (2, 2, 6)$$

$$|V1| \text{ (Euclidean vector length, not the number of dimensions)} = 3.32$$

$$|V2| = 3$$

$$M * V2 \text{ (matrix times vector, transpose it as necessary)} = 3 \times 1 \text{ matrix } (10, 9, 3)$$

$$M * M \text{ (or } M^2) = [(8, 4, 11), (6, 5, 9), (3, 1, 4)]$$

$$M^3 = [(31, 16, 43), (26, 16, 37), (11, 5, 15)]$$

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

$$HTHH = 0.6 * 0.4 * 0.6 * 0.6 = 0.0864$$

$$THTH = 0.4 * 0.6 * 0.4 * 0.6 = 0.0576$$

$$\text{Exactly 2 Heads out of a sequence of 3 coin flips.} = 0.6 * 0.6 * 0.4 * 3 = 0.432$$

$$\text{Exactly 1 Tail out of sequence of 3 coin flips.} = 0.4 * 0.6 * 0.6 * 3 = 0.432$$

- d) Consider a database schema consisting of two tables, Employee (ID, Name, Address), Project (PID, Name, Deadline), Assign(EID, PID, 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 projects that are not assigned to any employees (PID and Deadline of the project).

```
SELECT Project.PID, Project.Deadline
FROM Project
LEFT JOIN Assign on Assign.PID = Project.PID
WHERE Assign.EID IS NULL;
```

- ii. For each date, find how many assignments were made that day.

```
SELECT UNIQUE Date, Count(Date)
FROM Assign
GROUP BY Date;
```

- iii. Find all projects that have either 0, 1, or 2 employees assigned to them (note that the answer should include 0 employees to be correct).

```
SELECT Distinct Project.PID, Count(Assign.PID)
FROM Project LEFT JOIN Assign on Assign.PID = Project.PID
GROUP BY Project.PID
HAVING Count(Assign.PID) < 3;
```

- e) Mining of Massive Datasets, Exercise 1.3.3

Justify your answer (giving an example only would be worth partial credit)

When the hash function is  $h(x) = x \bmod 15$ , if the  $c$  is 2, only bucket 0, 2, 4, 6, 8, 10, 12, 14 will be assigned and bucket 1, 3, 5, 7, 9, 11, 13 will be empty. When the  $c$  equals an unusual number like a prime number, only a few buckets will be assigned. With the  $c$  has common factors with all, the possibility of hash-keys divided equally into buckets will be increase. Therefore, when  $c$  equals 1, a large random choice of hash-keys will be divided roughly equally into buckets.

- f) Hadoop Distributed Filesystem.

- i. What are the guarantees offered by a replication factor of 3 (3 copies of each block)?

Fault tolerance includes node failure recovery and availability.

- ii. What action does NameNode have to take when a machine in the Hadoop cluster fails/crashes?

When a machine in the Hadoop cluster fails/crashes, the NameNode will get an alert to recover the duplication of failed node into a new node.

- iii. What is the storage cost for a file of size X MBs, when the HDFS replication factor is set to 3?

3X MBs

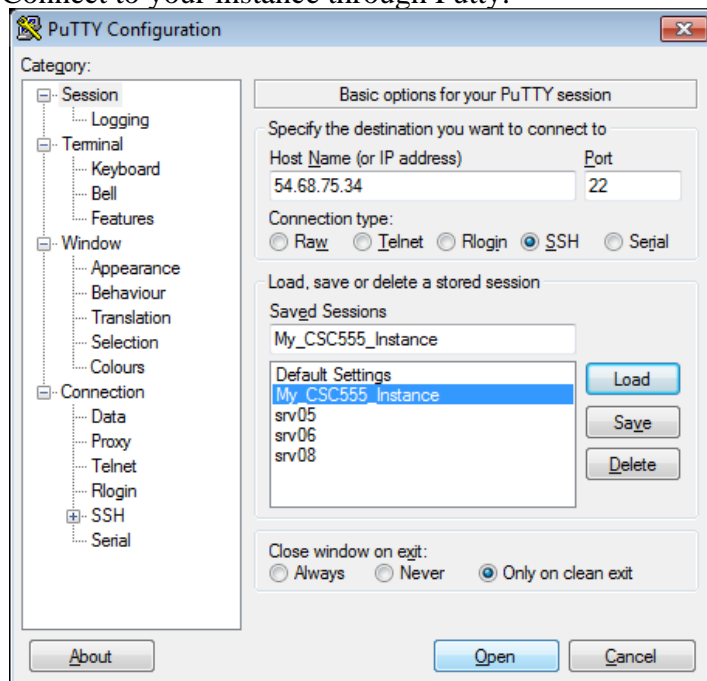
## Part 2: Linux Intro

This part of the assignment will serve as an introduction to Linux. Make sure you go through the steps below and submit screenshots where requested – submit the entire screenshot of a command terminal.

Use at least a t2.small instance or Hadoop may not run properly with insufficient memory. All Linux commands are in **Berlin Sans FB**. Do not type the “\$” symbol. The “\$” represents the prompt “[ec2-user@ip-xxx-xx-xx-xxx ~] \$ ” in your particular Linux instance.

**0. Login to your Amazon EC2 Instance** (NOTE: instructions on how to create a new instance and log in to it are provided in a separate file, UsingAmazonAWS.doc)

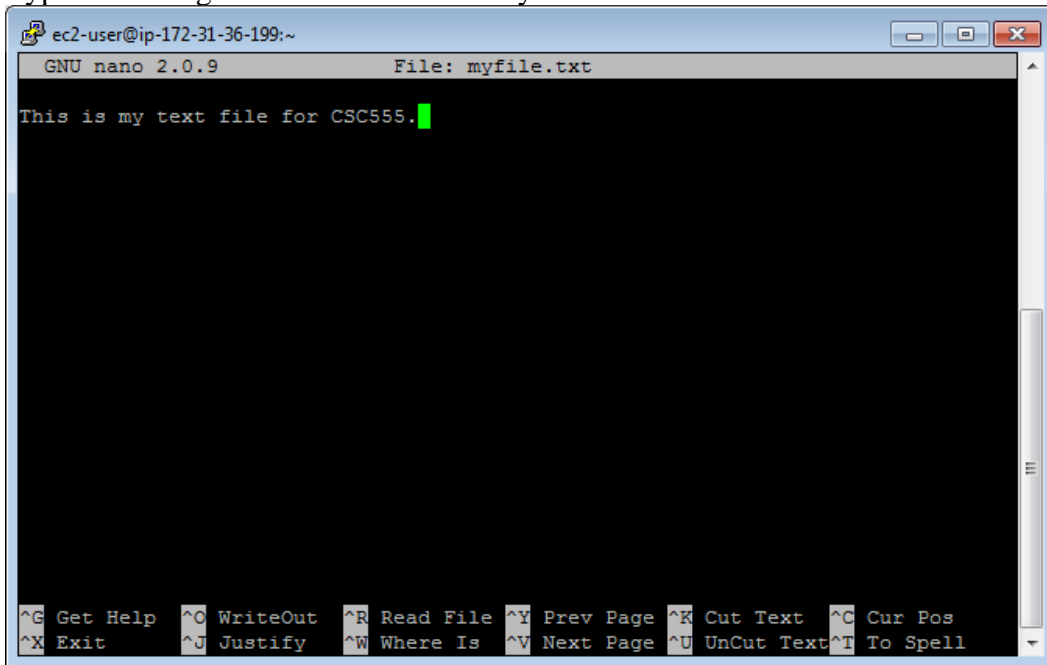
Connect to your instance through Putty.



Your instance should look like as the following image:



Type something into the file: "This is my text file for CSC555."



```
ec2-user@ip-172-31-36-199:~
GNU nano 2.0.9      File: myfile.txt
This is my text file for CSC555.
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Save changes: Ctrl-o and hit Enter.

Exit: Ctrl-x

Emacs Instructions (Option 2):

You will need to install emacs.

**\$ sudo yum install emacs**

Type "y" when asked if this is OK to install.

**\$ emacs myfile.txt**

Type something into the file: "This is my text file for CSC555."

Save changes: Ctrl-x, Ctrl-s

Exit: Ctrl-x Ctrl-z

Vim Instructions(Option 3):

- NOTE: When **vim** opens, you are in *command* mode. Any key you enter will be bound to a command instead of inserted into the file. To enter *insert* mode press the key "i". To save a file or exit you will need to hit Esc to get back into *command* mode.

**\$ vim myfile.txt**

Type "i" to enter *insert* mode

Type something into the file: "This is my text file for CSC555."

Save changes: hit Esc to enter *command* mode then type ":w"

Exit: (still in *command* mode) type ":x"

Confirm your file has been saved by listing the files in the working directory.

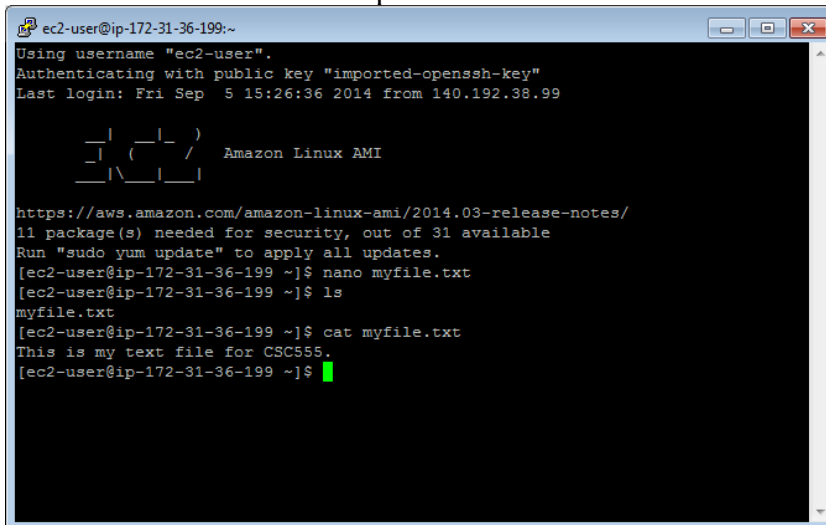
**\$ ls**

You should see your file.

Display the contents of the file on the screen.

```
$ cat myfile.txt
```

Your file contents should be printed to the terminal.



```
ec2-user@ip-172-31-36-199:~
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Fri Sep  5 15:26:36 2014 from 140.192.38.99

 _ _ | _ _ | _ _ |
 _ | ( _ | /      Amazon Linux AMI
 _ | \ _ | _ | _ |

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ nano myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-36-199 ~]$
```

- **Tip:** Linux will fill in partially typed commands if you hit Tab.

```
$ cat myfi
```

Hit Tab and “myfi” should be completed to “myfile.txt”. If there are multiple completion options, hit Tab twice and a list of all possible completions will be printed. This also applies to commands themselves, i.e. you can type in **ca** and see all possible commands that begin with ca.

## 2. Copy your file.

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.

**SUBMIT:** Take a screen shot of the contents of your copied file displayed on the terminal screen.

```
[[ec2-user@ip-172-31-2-159 ~]$ cp myfile.txt mycopy.txt
[[ec2-user@ip-172-31-2-159 ~]$ ls
grail mycopy.txt myfile.txt
```

## 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
```

#### 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.

```
[[ec2-user@ip-172-31-2-159 ~]$ cd CSC555
```

```
[[ec2-user@ip-172-31-2-159 CSC555]$ ls
```

```
mycopy.txt  myfile.txt
```

—

#### 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.

```
[[ec2-user@ip-172-31-2-159 ~]$ cd CSC555
[[ec2-user@ip-172-31-2-159 CSC555]$ mv myzipfile.zip /home/ec2-user/
[[ec2-user@ip-172-31-2-159 CSC555]$ cd ..
[[ec2-user@ip-172-31-2-159 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
  extracting: mycopy.txt
  extracting: myfile.txt
[[ec2-user@ip-172-31-2-159 ~]$ ls
CSC555  mycopy.txt  myfile.txt  myzipfile.zip
```

## 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 1<sup>st</sup> 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 2<sup>nd</sup> Column is the number of links to the file. The 3<sup>rd</sup> and 4<sup>th</sup> columns are the owner and the group of the file. The 5<sup>th</sup> column displays the size of the file in bytes. The 6<sup>th</sup> column is the date and time the file was last modified. The 7<sup>th</sup> 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.

```
[[ec2-user@ip-172-31-2-159 ~]$ ls -lh
total 88K
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9 2000 grail
-rw-rw-r-- 1 ec2-user ec2-user  33 Apr 11 23:18 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Apr 11 23:17 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 384 Apr 11 23:23 myzipfile.zip
```

## 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

```
[ec2-user@ip-172-31-2-159 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-2-159 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
```

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 "*" * 25
print "My Lucky Numbers".rjust(20)
print "*" * 25
```

```
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  
**\$ 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 a word count total for each word with the number of times that word occurs in the entire file. That is, if the file has the word “Hadoop” occurs in the file 5 times, your code should print “Hadoop 5”. It should output the count of all words occurring in a 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.

```
import re
```

```
import sys
```

```
b = {}
```

```
c = 1
```

```
for line in sys.stdin:
```

```
    word = re.findall("\w+", line)
```

```
    for i in word:
```

```
        if i not in b:
```

```
            b[i] = c
```

else:

c = b[i]

c += 1

b[i] = c

for key, value in b.items():

print(key, value)

```
[ec2-user@ip-172-31-2-159 ~]$ cat myfile.txt | python count.py
('for', 2)
('This', 2)
('text', 2)
('is', 2)
('file', 2)
('CSC555', 2)
('my', 2)
```

## 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

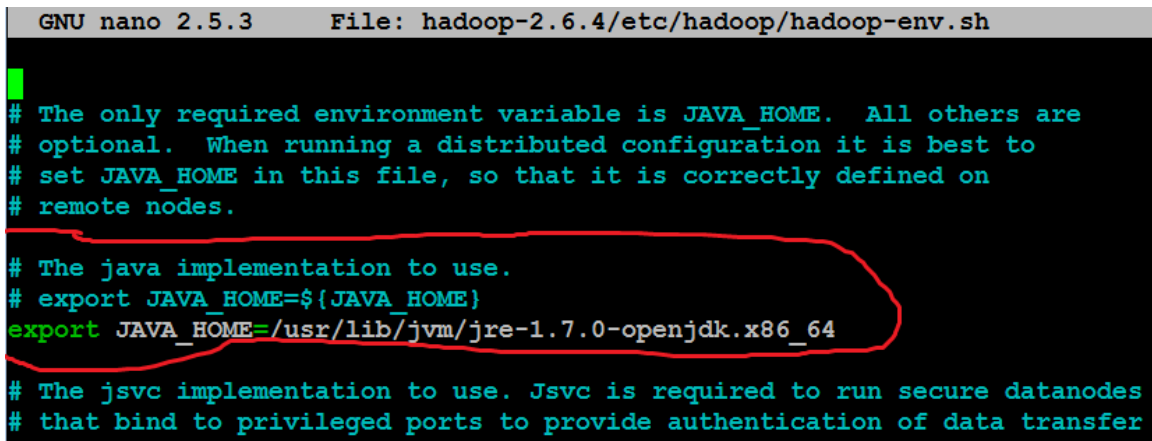
```
[ec2-user@ip-172-31-16-63 ~]$ readlink -f $(which java)
```

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:

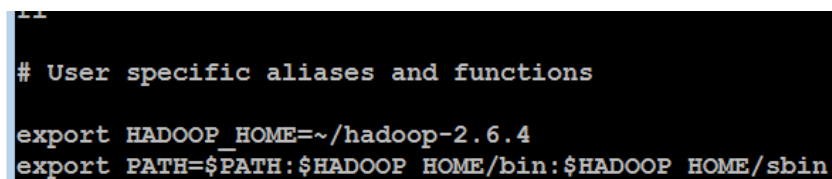
```
export HADOOP_HOME=~/.hadoop-2.6.4
```

```
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**

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**

```
[ec2-user@ip-172-31-2-159 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-2-159 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r-- 1 ec2-user supergroup 231149003 2021-04-20 01:20 /data/bioproject.xml
```

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 one command, 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

```
[ec2-user@ip-172-31-2-159 ~]$ time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar wordcount /data/bioproject.xml /data/wordcount1
21/04/20 01:20:52 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
21/04/20 01:20:53 INFO input.FileInputFormat: Total input paths to process : 1
21/04/20 01:20:53 INFO mapreduce.JobSubmitter: number of splits:2
21/04/20 01:20:53 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1618881539705_0001
21/04/20 01:20:53 INFO impl.YarnClientImpl: Submitted application application_1618881539705_0001
21/04/20 01:20:53 INFO mapreduce.Job: The url to track the job: http://ip-172-31-2-159.us-east-2.compute.internal:8088/proxy/application_1618881539705_0001/
21/04/20 01:21:04 INFO mapreduce.Job: Running job: job_1618881539705_0001
21/04/20 01:21:04 INFO mapreduce.Job: Job job_1618881539705_0001 running in uber mode : false
21/04/20 01:21:04 INFO mapreduce.Job: map 0% reduce 0%
21/04/20 01:21:19 INFO mapreduce.Job: map 23% reduce 0%
21/04/20 01:21:22 INFO mapreduce.Job: map 26% reduce 0%
21/04/20 01:21:28 INFO mapreduce.Job: map 28% reduce 0%
21/04/20 01:21:31 INFO mapreduce.Job: map 42% reduce 0%
21/04/20 01:21:34 INFO mapreduce.Job: map 47% reduce 0%
21/04/20 01:21:40 INFO mapreduce.Job: map 58% reduce 0%
21/04/20 01:21:43 INFO mapreduce.Job: map 60% reduce 0%
21/04/20 01:21:46 INFO mapreduce.Job: map 66% reduce 0%
21/04/20 01:21:47 INFO mapreduce.Job: map 77% reduce 0%
21/04/20 01:21:49 INFO mapreduce.Job: map 83% reduce 0%
21/04/20 01:21:58 INFO mapreduce.Job: map 100% reduce 0%
21/04/20 01:22:01 INFO mapreduce.Job: map 100% reduce 72%
21/04/20 01:22:02 INFO mapreduce.Job: map 100% reduce 100%
21/04/20 01:22:02 INFO mapreduce.Job: Job job_1618881539705_0001 completed successfully
21/04/20 01:22:02 INFO mapreduce.Job: Counters: 49

File System Counters
  FILE: Number of bytes read=59605201
  FILE: Number of bytes written=86827934
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=231153301
  HDFS: Number of bytes written=20056175
  HDFS: Number of read operations=9
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2

Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=94064
  Total time spent by all reduces in occupied slots (ms)=12456
  Total time spent by all map tasks (ms)=94064
  Total time spent by all reduce tasks (ms)=12456
  Total vcore-milliseconds taken by all map tasks=94064
  Total vcore-milliseconds taken by all reduce tasks=12456
  Total megabyte-milliseconds taken by all map tasks=96321536
  Total megabyte-milliseconds taken by all reduce tasks=12754944

Map-Reduce Framework
  Map input records=5284546
  Map output records=18562366
  Map output bytes=279356680
  Map output materialized bytes=26902454
  Input split bytes=202
  Combine input records=20053191
  Combine output records=2673165
  Reduce input groups=1040390
  Reduce shuffle bytes=26902454
  Reduce input records=1182340
  Reduce output records=1040390
  Spilled Records=3855505
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=1052
  CPU time spent (ms)=41480
  Physical memory (bytes) snapshot=567083008
  Virtual memory (bytes) snapshot=6320037888
  Total committed heap usage (bytes)=334323712

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    1m13.331s
user    0m3.985s
sys     0m0.203s
```

(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.

Congratulations, you just finished running wordcount using Hadoop.

Submit a single document containing your written answers. Be sure that this document contains your name and “CSC 555 Assignment 1” at the top.