Electrical and Computer Engineering, Purdue University Northwest
Big Data (ECE 595), Fall 2019
Hands-on Assignment 1, Due Date: 10/25/2019

Objective: Python and Hadoop Map reduce Programming

Note: For grading, submit a single PDF file for the report and code files should be uploaded separately in the submission.

Task 0 [5 points]: Follow the steps mentioned below to install Virtual Box and run the class virtual machine (VM).

a) Download Virtual Box from https://www.virtualbox.org/ and install it in your system.
b) Follow the video (uploaded in Blackboard) to create the virtual machine. The username is **ece595** and password is **bigdataece595**
c) Submit screenshot of running Ubuntu in your machine.

Task 1 [10 points]: Assume in an ATM machine dispenses bills for $100, $50, $20, $10, and $5 only. Write a python program **`atm.py`** to simulate an ATM machine in which the customer will enter the amount to be withdrawn as input and it will dispense the cash with a minimum number of bills if it can dispense the input amount with the available bills. Following are a few test cases for your program:

```
Enter the amount for withdrawal: 145
Please collect your bills as follows:
$100: 1
 $20: 2
  $5: 1

Enter the amount for withdrawal: 73
The amount cannot be withdrawn.
```

Attach your code screenshot and outputs of the program in the report. Also, upload your code file separately in the submission.

Task 2 [20 points]: Write a **`registrar.py`** program that will load the **`records.txt`** (available in Blackboard) file that has students' scores in different subjects. The program will perform the following tasks:

- Display the grades in different subjects for each student. You may consider this grading scale: A >= 90, 89 >= B >= 80, 79 >= C >= 70, 69 >= D >= 60, and 59 >= F
- Display the highest score and scorer in each subject
- Display the GPA of each student. The credit hours for each course is 3 and grade points for A, B, C, D, and F are 4, 3, 2, 1, and 0, respectively.

$$GPA = \frac{\sum Credit\_hours\_i \times Grade\_points\_i}{\sum Credit\_hours\_i}$$

Attach your code screenshot and outputs of the program in the report. Also, upload your code file separately in the submission.

Task 3 [10 points]: Hadoop pseudo-cluster set-up in Ubuntu virtual machine

1. Open the terminal
2. Go to **software** directory: **cd software**
3. Extract Hadoop files from the compressed file: **tar xzvf hadoop-2.9.1.tar.gz**
4. Move extracted Hadoop directory to /usr/local directory: **sudo mv hadoop-2.9.1 /usr/local/hadoop**
5. Change ownership for hadoop directory: **sudo chown –R ece595:ece595 /usr/local/hadoop**
   Note: **ece595** is a user here.
6. Find directory for Java: **readlink -f /usr/bin/java | sed "s:/bin/java::"**
   The output of the command will be the directory for Java.
7. Go to home directory: **cd**
8. Open .bashrc file: **gedit .bashrc**
   Add following lines:
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
   export HADOOP_HOME=/usr/local/hadoop
   export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
   Note: Value for JAVA_HOME is the output of the previous command in Step 6 for finding Java directory
   Close .bashrc file and execute command: **. .bashrc**
   or close the terminal and open it again**.**
   You can verify the changes with the command: **echo $JAVA_HOME**
9. Open /usr/local/hadoop/etc/hadoop/hadoop-env.sh:
   gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
   Replace **export JAVA_HOME=${JAVA_HOME}** with
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
   Also add this: **export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true**
   Close the file
10. Create a directory /app/hadoop/tmp: **sudo mkdir -p /app/hadoop/tmp**
    Change ownership: **sudo chown –R ece595:ece595 /app/hadoop/tmp**
11. Open /usr/local/hadoop/etc/hadoop/core-site.xml:
    gedit /usr/local/hadoop/etc/hadoop/core-site.xml
    Add these lines inside <configuration> </configuration> tags
    ```
    <property>
    <name>hadoop.tmp.dir</name>
    <value>/app/hadoop/tmp</value>
    </property>
    <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
    </property>
    ```
    Close the file

12. Open /usr/local/hadoop/etc/hadoop/hdfs-site.xml:
    gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
    Add these lines inside <configuration> </configuration> tags
    <property>
    <name>dfs.replication</name>
    <value>1</value>
    </property>
    Close the file.
13. Open /usr/local/hadoop/etc/hadoop/mapred-site.xml:
    gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml
    Add these lines inside <configuration> </configuration> tags
    <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    </property>
    Note: mapred-site.xml is not available. Rename mapred-site.xml.template file to mapred-site.xml file: **mv mapred-site.xml.template mapred-site.xml**
    Close the file.
14. Open /usr/local/hadoop/etc/hadoop/yarn-site.xml:
    gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
    Add these lines inside <configuration> </configuration> tags
    <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
    </property>
15. Perform following commands to set-up SSH public key authentication
    ssh-keygen -t rsa -P ""
    cat /home/ece595/.ssh/id_rsa.pub >> /home/ece595/.ssh/authorized_keys
    Note: **ece595** is a user here.
16. Format HDFS: **hadoop namenode -format**
17. Start service: **start-all.sh**
18. Execute command to check services are running: **jps**
19. Once your job is done, you can stop services: **stop-all.sh**
    Do not execute this command now, first do something in HDFS cluster as described in Task 4.

Attach the screenshot for the installation process in the report.

Task 4 [10 points]: HDFS commands

1. Assume you are at your home directory or execute: **cd**
2. List files in HDFS cluster: **hdfs dfs –ls /**
   You will see nothing in the output as you have not created anything yet.
3. Create a directory in HDFS cluster: **hdfs dfs -mkdir /user**
4. List files again in HDFS cluster: **hdfs dfs -ls**
   You will see the listing of /user directory in the output
5. Create **bigdata** directory inside /user directory:
   hdfs dfs -mkdir /user/bigdata

6. List that directory in HDFS cluster: **hdfs dfs -ls /user/**
7. Create a text file, say **demo.txt**, using gedit and write something in it and save it.
8. Upload the file in the cluster**:**
   hdfs dfs -copyFromLocal demo.txt /user/bigdata/
9. List the file in the cluster: **hdfs dfs -ls /user/bigdata/**
10. See the content of file: **hdfs dfs -cat /user/bigdata/demo.txt**

Attach the screenshot for the command outputs in the report.

Task 5 [45 points]: Map Reduce Programming

Note: Find the instructions for running the programs at the end.

A. [**10 points**] Write Mapper and Reducer programs that read "passage.txt" file from HDFS cluster and finds top 10 most frequent words and their frequencies. In the text file, many words may appear in different forms, e.g. The, the, you have to treat them as same word. In addition, some words may have double quote, single quote, or other non-alphabetic characters either in prefix or suffix, your program should be able to remove them and then consider the remaining characters as word.

B. [**20 points**] Write Mapper and Reducer programs that read "purchases.txt" (taken from Udacity's Hadoop course) file from HDFS cluster and finds the standard deviation of stores' sales in each city. To calculate the standard deviation refer Welford's online algorithm. You may find Python code snippet in the link. The required parts of the algorithm are as follows:

$$\overline{x_n} = \overline{x_{n-1}} + \frac{x_n - \overline{x_{n-1}}}{n}$$

$$M_{2,n} = M_{2,n-1} + (x_n - \overline{x_{n-1}})(x_n - \overline{x_n})$$

$$\sigma^2{}_n = \frac{M_{2,n}}{n}$$

Here, x_n denotes the sample mean of the first n samples and $\sigma^2{}_n$ their population variance. Once you calculate the popular variance for each city and then you can take the square root of it, which will give you the standard deviation.

C. **[15 points]** Write Mapper and Reducer programs that read a web server log file "http_log.txt" and finds the number of **unique files** request sent by each client. You may refer the link to understand the fields in the log file. The file request could be found inside the double quotes after GET in each line.

Attach your code screenshot and outputs of the program in the report. Also, upload your code files separately in the submission.

Instructions

You can create a folder where you can keep your Map Reduce code. Use naming format **mapper_*firstname_lastname_letter*.py** for Mapper code and **reducer_*firstname_lastname_letter*.py** for reducer code for each problem. For example, in my case, mapper_quamar_niyaz_A.py and reducer_quamar_niyaz_A.py will be the files' names for Problem A.

Once you finish mapper and reducer code, you can verify their correctness by using them for a small file similar to the problem data file. You can create it by examining the file content for the given problem or copy-pasting first 50-100 lines from the data file.

Mapper verification

```
cat problem_file_small.txt | your_mapper_ Python_file
```

Check whether you are getting the same output that you want. If you do not then you need to fix the mapper code and execute the command again. It is assumed that you are executing the command from the folder where you have put Map Reduce code and small data file.

Reducer code verification

```
cat problem_file_small.txt | mapper.py | sort | your_reducer_ Python_file
```

Check whether you are getting the same output that you want. If you do not then you need to fix the reducer code and execute the command again. It is assumed that you are executing the command from the folder where you have put Map Reduce code and small data file.

If your mapper and reducer work fine, then move the actual data file in the cluster and run Mapper and Reducer code on it. Since your code have been written in Python, you need to use Hadoop streaming. You can find the jar file for it inside **/usr/local/hadoop/share/hadoop/tools/lib/** folder. To avoid the long typing, you can copy the jar file in /usr/local/hadoop folder.

```
cp /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.1.jar /usr/local/hadoop/
```

Now execute the command similar to the example command given below:

```
hadoop jar /usr/local/hadoop/hadoop-streaming-2.9.1.jar  \
-mapper /path/to/your/MapReduceCode/folder/your_mapper_Python_file \
-reducer /path/to/your/MapReduceCode/folder/your_mapper_Python_file \
-input /path/to/folder/for/your/data/file/in/cluster/ -output
/path/to/your/output/folder/in/cluster
```

The output file will be created in the output folder that you have provided in the command. The output file will have a prefix part. You can see the content of the file using **hdfs dfs -cat** command or you can download the file in the local file system using **hdfs dfs -get** command.

Note: Do not use any folder name for the output folder if that folder already exists in HDFS cluster. Either remove the existing folder or use other folder name.