Practice Tasks using Hadoop (Gradable)

Task 0: Installation (Single Node)

For installation, any of the following options can be availed:

- **Option 1; Install Apache Hadoop directly:** Download and install Apache Hadoop dirctly on your own machine. Visit their website and follow the instructions (varies from distribution to distribution).
- **Option 2; Cloudera VM on your own machine:** Download the Cloudera VM from their official website and configure it on a virtual machine (VM) yourself. Just run it and give username cloudera and password cloudera. You would require at least 8 GB RAM to run it smoothly.
- **Option 3; Create an instance on Google Cloud Platform (GPC):** Create an instance on Google Cloud platform:
    - Visit http://cloud.google.com and create a Compute Engine Instance with the your username. In Boot Disk type, specify Ubuntu LTS latest version. In Firewall, allow http traffic.
    - Add your ssh key (public key of your local machine) to SSH Keys when clicking on your created instance.

```
ssh-keygen
```

    - Login via ssh to the machine.

```
Ssh <ip address of instance>
```

    - Perform some routine installations by running the following:

```
sudo apt update
sudo apt install openjdk-8-jdk
```

    - Create a non-root user for Hadoop with any password you like (e.g. test123), and then login to it using su.

```
sudo adduser hdoop

su - hdoop
```

    - To prevent frequent login, again use ssh-keygen (for the hdoop user) and add it to authorized keys as follows:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod 0600 ~/.ssh/authorized_keys
```

    - Now, download the latest hadoop version from their website and extract it.

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz

tar xzf hadoop-3.2.3.tar.gz
```

    - Set your environment variables by editing the file ~/.bashrc using nano or vim, and add the contents to the end of the file.

```
nano ~/.bashrc

export HADOOP_HOME=/home/hdoop/hadoop-3.2.3
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

- Specify the Java home location by running the following few lines. The location should end with openjdk-amd64 only.

```
readlink -f /usr/bin/javac
```

- Specify the JAVA_HOME variable in the following:

```
nano ~/hadoop-3.2.3/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- Perform some basic hadoop configurations using the following:

```
vim ~/hadoop-3.2.3/etc/hadoop/core-site.xml

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/omar/hadoop-3.2.3/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

- With the above, make sure you create the tmpdata directory in the right location you specified.
- Then, perform some basic HDFS settings using:

```
vim ~/hadoop-3.2.3/etc/hadoop/hdfs-site.xml

<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/omar/hadoop-3.2.3/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/omar/hadoop-3.2.3/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>
```

- Then, select the resource manager for your Map Reduce jobs as Yarn using:

```
vim ~/hadoop-3.2.3/etc/hadoop/mapred-site.xml

<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

```
</configuration>
```

- Then, give the following for resource manager Yarn settings:

```
vim ~/hadoop-3.2.3/etc/hadoop/yarn-site.xml

<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>34.125.156.43</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value>${yarn.nodemanager.hostname}:5349</value>
</property>
<property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>${yarn.nodemanager.hostname}:5350</value>
</property>

<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>${yarn.nodemanager.hostname}:5351</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>${yarn.nodemanager.hostname}:5352</value>
</property>
<property>
    <name>yarn.resourcemanager.address</name>
    <value>${yarn.nodemanager.hostname}:5353</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DI
STCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
</configuration>
```

- With the settings done, we can now format our HDFS file system using:

```
hdfs namenode -format
```

- Now, we start our all services, which can be checked by running Java Processes (jps) in the end:

```
~/hadoop-3.2.3/sbin/start-dfs.sh

~/hadoop-3.2.3/sbin/start-yarn.sh

jps
```

- Create firewall rule for your system IP address and port 9870 (for Hadoop) and 5349 (for Yarn Resource Manager) in Firewall settings of Google Cloud Platform.

Task 1: Creating Your Directory Space

The first thing is to create your own directory space on HDFS. Issue the following from your terminal:

```
hdfs dfs -mkdir /usr/$(whoami)
```

Verify that it exists by the following:

```
hdfs dfs -ls /usr

drwxr-xr-x   - omar omar          0 2020-05-16 09:49 /usr/omar
```

Note that the initial column contains drwxr-xr-x. This can be broken into multiple sections:

1. d, meaning that this is a directory
2. rwx, meaning that the user “omar” has read/write/execute permissions
3. r-x, meaning that all users in the hadoop group have read/execute permissions
4. r-x, meaning that everybody else on the system have read/execute permissions

You may set your folder to restricted mode to avoid other users tampering with your data:

```
hdfs dfs -chmod -R 700 /usr/omar
```

Task 2: Understanding the System

Using the following commands, address the questions:

```
hdfs dfsadmin -printTopology

hdfs dfsadmin -report

hdfs fsck /

hadoop fsck / -files -blocks -locations
```

Questions

1. How many datanodes are part of the hadoop topology?
2. What are the IP addresses of these datanodes?
3. What is the configured and present capacity of the HDFS?
4. What is the default file replication count?

Task 3: Getting Sample Data

Visit the link http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp using the following command line browser:

```
elinks http://transtats.bts.gov/OT_Delay/OT_DelayCause1.asp
```

Browser Usage

- You can use page-up and page-down key to move screen by screen on the page (also using your mouse)

- You can press the ESC button to show the menu bar. The menubar can be scrolled through the numeric keypads.
- You can use your numeric keypad to scroll through the different hyper-links

Data Selection

Choose the following options:

- Select a Carrier: All
- Select an Airport: All
- Period from: January 2020
- Period To: December 2019

Data Download

Scroll to Submit and press enter. Once the page is refreshed, scroll down to “Download Raw Data” and press enter. Follow the on-screen instructions to Save the file. Once done, exit by pressing ESC → File → exit.

Verify that the data has been downloaded on your system by running the command:

```
ls -lhtr

-rw-r—r-- 1 omar omar 6.0M May 16 2020 9757924_airline.zip
```

The zip file should be the last line you see. Next step, extract the zip file using the command:

```
unzip 9757924_airline.zip

-rw-r—r-- 1 omar omar  27M May 16 2020 9757924_airline.csv

-rw-r—r-- 1 omar omar 6.0M May 16 2020 9757924_airline.zip
```

Rename the CSV file to something simpler like airline_data.csv:

```
mv 9757924_airline.csv airline_data.csv
```

Format of Data

The header of the CSV contains the signature:

"year"," month","carrier","carrier_name","airport","airport_name","arr_flights","arr_del15","carrier_ct"," weather_ct","nas_ct","security_ct","late_aircraft_ct","arr_cancelled","arr_diverted"," arr_delay"," carrier_delay","weather_delay","nas_delay","security_delay","late_aircraft_delay"

What do these symbols mean? Fill the table from https://www.transtats.bts.gov/Fields.asp?Table_ID=236 (The names may not be exact, + understanding the data is part of any problem):

	Name	Description	Sample Output (First Record Only)
1	Year	Year	
2	Month	Month	
3	Carrier	Carrier Abbreviation	
4	Carrier_Name	Carrier Name	
5	Airport		
6	Airport_Name		
7	Arr_Flights		

8	Arr_Del15
9	Carrier_CT
10	Weather_CT
11	NAS_CT
12	Security_CT
13	Late_Aircraft_CT
14	Arr_Cancelled
15	Arr_Diverted
16	Arr_Delay
17	Carrier_Delay
18	Weather_Delay
19	NAS_Delay
20	Security_Delay
21	Late_Aircraft_Delay

Move Data to HDFS

Copy over your data using:

```
hdfs dfs -put airline_data.csv /usr/omar/airline_data1.csv
```

Verify that it exists by:

```
hdfs dfs -ls /usr/omar

Found 1 items

-rw-r--r-- 3 omar omar ........... 10:16 /usr/omar/airline_data.csv
```

Answer the following questions:

	Question	Answer
1	What is the default block size (in Mb) of the airline_data.csv file?	
2	Is there any missing replicas for the file airline_data.csv?	
3	What command will you use to change this block size to 6 Mb (remember to convert into bytes)	
4	How many blocks are used by airline_data.csv after changing block size in Question 2?	
5	How many missing replicas are there for file airline_data.csv after block change?	
6	Why are there missing replicas?	

Note: You can use the following to get data for some of the above answers:

```
hdfs fsck /usr/omar/airline_data.csv
```

Task 4: Setting up First Map Reduce Job

In this task, we will count the total flights per year. For this, use the following sample Mapper and Reducer codes (Python based)

Mapper Code (mapper.py)

```
#!/usr/bin/python
```

```
import sys

for line in sys.stdin:

        data = line.strip().split(",")
        key = data[0]
        value = 1
        print ("{0}\t{1}".format(key, value) )
```

Reducer Code (reducer.py)

```
#!/usr/bin/python

import sys
total = 0
oldkey = None

for line in sys.stdin:

        data = line.strip().split("\t")
        thiskey = data[0]
        value = data[1]

        if thiskey != oldkey and oldkey != None:

                print ("{0}\t{1}".format(oldkey, total))
                oldkey = thiskey
                total = 0

        oldkey = thiskey
        total += float(value)

if oldkey != None:

        print ("{0}\t{1}".format(oldkey, total))
```

Give both the mapper.py and Reducer.py executable permissions:

```
chmod u+x mapper.py

chmod u+x reducer.py
```

Testing Locally

Test the mapper and reducer on your local directory first, without map reduce:

```
cat airline_data.csv | ./mapper.py | sort | ./reducer.py

2010    17575.0
2011    15585.0
2012    14387.0
2013    16089.0
2014    13980.0
2015    13528.0
2016    12217.0
2017    12518.0
2018    20231.0
2019    20946.0
"year"  1.0
```

Testing on Hadoop

Test the mapper and reducer using hadoop:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar -file ./mapper.py -file ./reducer.py -mapper mapper.py -reducer reducer.py -input /usr/omar/airline_data.csv -output /usr/omar/query1_output
```

You will see plenty of output generated on the screen. Give answers to the following:

	Question	Answer
1	What was the <key,value> pair used in this query?	
2	How many mapper threads were used?	
3	How many reducer threads were used?	
4	What was the time spent by all mapper threads?	
5	What was the time spent by all reducer threads?	
6	What is the file name in which your output is located?	

Variation 1

For this task, you need to calculate execution time (mapper + reducer) by two variations:

1) play with block size of airline_data.csv using the **"-D dfs.blocksize=<>"** argument.

2) Play with thread variation using the **"-D mapred.reduce.tasks=<>"**, or the **"-jobconf mapred.reduce.tasks=<>"** argument.

# of Reducer Tasks	airline_data.csv block size variation (Mb)				
	2 Mb	4 Mb	8 Mb	16 Mb	Default
2					
4					
8					
16					

Highlight the best time with green and worst time with orange background.

	Question	Answer
1	How many output files are produced for 16 reducer threads.	
2	Why are some output files having 0 byte size?	

Variation 2

For this task, you need to calculate execution time (mapper + reducer) by two variations:

1) play with block size of airline_data.csv using the **"-D dfs.blocksize=<>"** argument.

2) Play with thread variation using the **"-D mapred.map.tasks=<>"**, or the **"-jobconf mapred.map.tasks=<>"** argument.

# of Map Tasks	airline_data.csv block size variation (Mb)				
	2 Mb	4 Mb	8 Mb	16 Mb	Default
2					
4					

8					
16					

Highlight the best time with green and worst time with orange background.

Variation 3

From the Variation 1 or Variation 2, choose the airline_data.csv block size which is giving best performance. And then, for this task, you need to calculate execution time (mapper + reducer) by two variations:

1) Play with thread variation using the **"-D mapred.reduce.tasks=<>"**, or the **"-jobconf mapred.reduce.tasks=<>"** argument.

2) Play with thread variation using the **"-D mapred.map.tasks=<>"**, or the **"-jobconf mapred.map.tasks=<>"** argument.

# of Map Tasks	**# of Reduce Tasks**			
	2	**4**	**8**	**16**
2				
4				
8				
16				

Highlight the best time with green and worst time with orange background.

Task 5: Designing Additional Queries

In this task, you have to design additional mapper/reducer threads for the following cases. In each case, it will be a good idea to put the corresponding mapper/reducer code in folders /usr/omar/task5a, /usr/omar/task5b, /usr/omar/task5c, and so on to avoid any confusion.

Task 5a: Present the Total Flights per Year as a percentage

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump:	

Task 5b: Which is the busiest month of airline traffic of all years?

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump:	

Task 5c: Which Airline Carrier has flown the most flights over the 10 year period?

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump (just the maximum):	

Task 5d: Which Airport has been the most busiest over the 10 year period?

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump (just the maximum):	

Task 5e: Which Airport has the Largest Flights to Cancellation Ratio?

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump (just one output):	

Task 5f: Find the Total Amount of Delay Minutes Grouped by Airline

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump (just top 10):	

Task 5g: Find the Airport with most Cancelled Flights in 2016.

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump (one answer):	

Task 5h: Find the average delay time for an airport that is the most busiest of all other airports.

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump	

Task 5j: What is the Probability that a Flight will be Cancelled due to Bad Weather at the Most Busiest Airport of all other Airports?

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump	

Task 5k: Design Any Query of your Choice (Output must be limited)

<Key, Value> Pair used:	
Folder Containing Code:	
Sample Output Dump	