Ronaldlee Ejalu
CSC 555 Mining Big Data
Assignment 4

1) Consider a Hadoop job that will result in 89 blocks of output to HDFS.
Suppose that writing one output block to HDFS takes 1 minute. The HDFS replication factor is set to 3 unless otherwise noted (the cost of writing output should include the replication blocks, although the copies are written by HDFS rather than by reducer. In fact, reducer doesn't write any of the blocks, it just produces the computation result).

   a) How long will it take for the reducer to write the job output on a 5-node Hadoop cluster? (ignoring the cost of Map processing, but counting replication cost in the output writing).

      Writing one output block takes 1 min
      Also, reading one output block takes 1 min.
      HDFS replication factor is set to 3.
      There are 89 blocks of output to be read and written.

      So, a 5 node Hadoop cluster:
      89/5 = 17.8
      It takes 1 min to read a block and 1 min to write a block therefore:
      17.8 min * 3 = 53.4 mins.

   b) How long will it take for reducer(s) to write the job output to 10 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 1)

      Replication factor = 1
      10 nodes to process 89 blocks
      So 89/10 = 8.9mins
      Since the replication factor is 1 so 8.9min * 1 = 8.9mins

   c) How long will it take for reducer(s) to write the job output to 10 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 3)

      10 Hadoop worker nodes
      Node 89/10 = 8.9 mins
      Since Replication factor = 3
      8.9 mins * 3 = 26.7 mins

d) How long will it take for reducer(s) to write the job output to 100 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 1)
For 100 Hadoop worker nodes with a replication factor of 1 will take 1 min.

e) How long will it take for reducer(s) to write the job output to 100 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 3)

Node: 89/100 mins
Replication factor of 3 is equal: (89/100) * 3 = 2.67mins

You can ignore the network transfer costs as well as the possibility of node failure.

2) Repeat the exercise from Lecture 3 (RSA examples)

a) Select two (small) primes and generate a public-private key pair.

- p = 5 , q = 11
- Compute their system modulus n = p * q = 5 * 11 = 55
- Compute Ø(n) = (p – 1)(q – 1) = (5-1)(11-1) = 40
- Select e such it is co-prime with Ø(n) = 40 , N = 55 and it must be less than Ø(n), 40, where gcd(e, 40) = 1 so let e = 3.
- We determine d: de = 1 mod 40 and d < 40
  We got pick a number such that 3d (mod 40) = 1
  3, 9, 12, 15, 18, 21, 24, 27, 30…….
  So d = 27
  Where 3* 27 mod 40 = 1
  81 mod 40 =1
- Published public key KU = {3, 55}
- Private key = {27, 55}

b) Compute a sample ciphertext using your public key
Given message M = 12 (12 < 55)
Encryption KU = {3, 55} such that
C = 12^3 mod 55 = 23

c) Decrypt your ciphertext from 2-b using the private key
Decryption KR = {27, 55}
M = 23^27 mod 55
⇨ [(23^4 mod 55) * (23^4 mod 55) * (23^2 mod 55) * (23^4 mod 55) * (23^4 mod 55) * (23^2 mod 55)] mod 55 = 12

d) Why can't the encrypted message sent through this mechanism be larger than the value of n?

All the computation takes the remainder of 55, a single message can't exceed n, 55, because if you send 56, it will be truncated to 1, which usually makes the computation expensive, and this is used to exchange a regular password.

If it is larger than the value of n, it will be split.

3) Given the following keys: 1, 4, 5, 8, 11, 12, 14, 15, 17, 25, 26, 28, 50, 51, 59, 87, 89, 93, 98, design the following:

   a) A distribution of these keys across 3 reducers using the default key partitioner (% 3)

   The keys are distributed across 3 reducers based on the default key partitioner (%3) where the remainder is 0, 1, 2 distributed amongst three reducers R0, R1 and R3 respectively as below:
   R0 = {12, 15, 51, 87, 93}
   R1 = {1, 4, 25, 28}
   R2 = {5, 8, 11, 14, 17, 26, 50, 59, 89, 98}

   b) Design a custom sorting partitioner instead of the default one and describe the resulting output across the same 3 reducers

```
If key <= 25: /*all keys less than or equal to 25 will be assigned*/
        Reducer_0
Elfif key > 25 and key <= 50:
        Reducer_1
else:
        Reducer_2
```

   Reducer_0, Reducer_1, Reducer_2 will be the output of the custom Partitioner and the keys will be distributed to the reducers in the given ranges above.
   Reducer_0 = {1, 4, 5, 8, 11, 12, 14, 15, 17, 25}
   Reducer_1 = {26, 28, 50}
   Reducer_2 = {51, 59, 87, 89, 93, 98}

   c) What is the downside (i.e., extra overhead) of employing a custom partitioner?
   You need to run sampling in order to create a custom partitioner; this involves sampling the data and finding a distribution so that you can come up with a decision. The process ends up being so expensive and time consuming, especially when you end up with a wrong distribution; you will have to do it again until you get it right.

4) Implement the following query using Hadoop streaming and python with the lineorder table.
   http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql
   http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/lineorder.tbl

```
SELECT lo_shipmode, STDDEV(lo_tax)
FROM lineorder
WHERE lo_quantity BETWEEN 17 AND 24
```

GROUP BY lo_shipmode;

STDDEV is standard deviation.

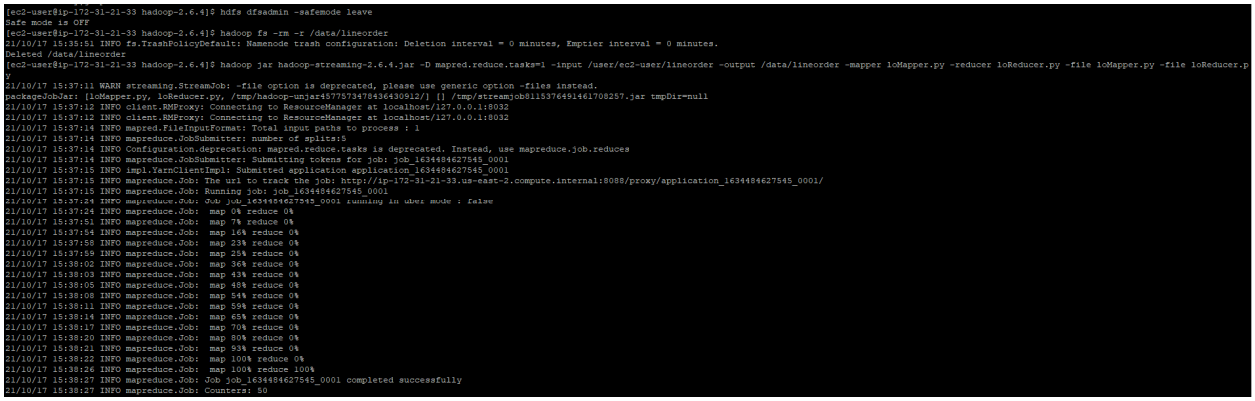
loMapper.py

```
#!/usr/bin/python
import sys
for line in sys.stdin:
        line  = line.strip()
        vals = line.split('|')
        if 17 <= int(vals[8]) <= 24:
                valsT = vals[16].strip()
                valsT = valsT.replace(' ','_')
                print('%s\t%d' %(valsT, int(vals[14])))
```

loReducer.py

```
#!/usr/bin/python
import sys
import numpy as np


wordL = []                                              # declare an empty list o
f words
curr_id = None                                          # current Id I am trackin
g.
id = ''                                                 # id derived from the val
ues of the string.
for item in sys.stdin:# Loop through the list of strings
    cleansedLine =item.strip()               # remove any white spaces

    # remember to put the right delimeter.
    splittedLinesL = cleansedLine.split('\t')         # split the string to creat
e a list of words, in hadoof, it has to be '\t' delimeted
    # print(splittedLinesL)
    id = splittedLinesL[0]                             #  pick up the key
    # print(id, type(splittedLinesL[1]))

    if curr_id == id:                                 # if i see the same key, add
the value to list
        bagOftax.append(float(splittedLinesL[1]))
    else:
        if curr_id: # compute the standard deviation, once the single current key
 is completed
            # convert the list into a numpy array
            arr = np.array(bagOftax)

            # compute the standard deviation
            computedStd = np.std(arr, dtype=np.float64)
            print('%s\t%.2f' %(curr_id, computedStd))
        curr_id = id
        bagOftax = []       # reset the list before adding  the value of the next
 key to the list
        bagOftax.append(float(splittedLinesL[1]))

# output the last key
# and compute the standard deviation of all the key's values in the list.
if curr_id == id:
    arr = np.array(bagOftax)
    computedStd = np.std(arr, dtype=np.float64)
    print('%s\t%.2f' %(curr_id, computedStd))
    # print('%s: %s' %(curr_id, bagOftax))
```

Don’t forget to submit your python code and the command lines you used to execute Hadoop streaming. I also recommend submitting the screenshot of execution to simplify the grader’s job.



```
        File System Counters
                FILE: Number of bytes read=8916674
                FILE: Number of bytes written=18493437
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=594329915
                HDFS: Number of bytes written=72
                HDFS: Number of read operations=18
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=6
                Launched reduce tasks=1
                Data-local map tasks=6
                Total time spent by all maps in occupied slots (ms)=267929
                Total time spent by all reduces in occupied slots (ms)=20161
                Total time spent by all map tasks (ms)=267929
                Total time spent by all reduce tasks (ms)=20161
                Total vcore-milliseconds taken by all map tasks=267929
                Total vcore-milliseconds taken by all reduce tasks=20161
                Total megabyte-milliseconds taken by all map tasks=274359296
                Total megabyte-milliseconds taken by all reduce tasks=20644864
        Map-Reduce Framework
                Map input records=6001215
                Map output records=960369
                Map output bytes=6995930
                Map output materialized bytes=8916698
                Input split bytes=530
                Combine input records=0
                Combine output records=0
                Reduce input groups=7
                Reduce shuffle bytes=8916698
                Reduce input records=960369
                Reduce output records=7
                Spilled Records=1920738
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=1789
                CPU time spent (ms)=26510
                Physical memory (bytes) snapshot=1066700800
                Virtual memory (bytes) snapshot=12631420928
                Total committed heap usage (bytes)=719736832
```

```
        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=594329385
        File Output Format Counters
                Bytes Written=72
21/10/17 15:38:27 INFO streaming.StreamJob: Output directory: /data/lineorder
```

```
[ec2-user@ip-172-31-21-33 hadoop-2.6.4]$ hadoop fs -ls /data/lineorder
Found 2 items
-rw-r--r--   3 ec2-user supergroup          0 2021-10-17 15:38 /data/lineorder/_SUCCESS
-rw-r--r--   3 ec2-user supergroup         72 2021-10-17 15:38 /data/lineorder/part-00000
[ec2-user@ip-172-31-21-33 hadoop-2.6.4]$ hadoop fs -cat /data/lineorder/part-0000
cat: `/data/lineorder/part-0000': No such file or directory
[ec2-user@ip-172-31-21-33 hadoop-2.6.4]$ hadoop fs -cat /data/lineorder/part-00000
AIR     2.58
FOB     2.58
MAIL    2.58
RAIL    2.58
REG_AIR 2.58
SHIP    2.58
TRUCK   2.58
```

5) In this section you will practice using HBase. Note that HBase runs on top of HDFS, bypassing the MapReduce engine.

cd
(Download HBase)
wget http://dbgroup.cdm.depaul.edu/Courses/CSC555/hbase-0.90.3.tar.gz
gunzip hbase-0.90.3.tar.gz
tar xvf hbase-0.90.3.tar
cd hbase-0.90.3

(Start HBase service, there is a corresponding stop service and this assumes Hadoop home is set)
bin/start-hbase.sh

(Open the HBase shell – at this point jps should show HMaster)
bin/hbase shell

(Create an employee table and two column families – private and public. Please watch the quotes, if ' turns into ‘, the commands will not work)
create 'employees', {NAME=> 'private'}, {NAME=> 'public'}
put 'employees', 'ID1', 'private:ssn', '111-222-334'
put 'employees', 'ID2', 'private:ssn', '222-338-446'
put 'employees', 'ID3', 'private:address', '123 State St.'
put 'employees', 'ID1', 'private:address', '243 N. Wabash Av.'
scan 'employees'

Now that we have filled in a couple of values, add 3 new columns to the private family, 1 new column to the public family and create a brand new family with at least 2 columns. For each of these you should introduce at least 2 values -- so a total of (3+1+2) * 2 = 12 values inserted. Verify that the table has been filled in properly with scan command and submit a screenshot.
put 'employees', 'ID1', 'private:firstname', 'Ronaldlee'
put 'employees', 'ID1', 'private:lastname', 'Ejalu'
put 'employees', 'ID1', 'private:age', '35'

put 'employees', 'ID2', 'private:firstname', 'Stevens'
put 'employees', 'ID2', 'private:lastname', 'Smith'
put 'employees', 'ID2', 'private:age', '46'

put 'employees', 'ID1', 'public:residentialstatus', 'Yes'
put 'employees', 'ID2', 'public:residentialstatus', 'Yes'

disable 'employees'
alter 'employees', 'department details'
enable 'employees'

put 'employees', 'ID1', 'department details:deptname', 'Data Services'
put 'employees', 'ID1', 'department details:deptno', 'HF001'

put 'employees', 'ID2', 'department details:deptname ', 'HR Services'
put 'employees', 'ID2', 'department details:deptno', 'HF100'

```
0 row(s) in 0.0100 seconds

hbase(main):009:0> put 'employees', 'ID2', 'public:residentialstatus', 'Yes'
0 row(s) in 0.0130 seconds

hbase(main):010:0> scan 'employees'
ROW                                                   COLUMN+CELL
 ID1                                                  column=private:address, timestamp=1634488875308, value=243 N. Wabash Av.
 ID1                                                  column=private:age, timestamp=1634493868979, value=35
 ID1                                                  column=private:firstname, timestamp=1634493776784, value=Ronaldlee
 ID1                                                  column=private:lastname, timestamp=1634493849374, value=Ejalu
 ID1                                                  column=private:ssn, timestamp=1634488547950, value=111-222-334
 ID1                                                  column=public:residentialstatus, timestamp=1634493973027, value=Yes
 ID2                                                  column=private:age, timestamp=1634493942153, value=46
 ID2                                                  column=private:firstname, timestamp=1634493917274, value=Stevens
 ID2                                                  column=private:lastname, timestamp=1634493929152, value=Smith
 ID2                                                  column=private:ssn, timestamp=1634488712672, value=222-338-446
 ID2                                                  column=public:residentialstatus, timestamp=1634493988544, value=Yes
 ID3                                                  column=private:address, timestamp=1634488802627, value=123  State St.
3 row(s) in 0.0480 seconds

hbase(main):011:0> disable 'employees'
0 row(s) in 2.1740 seconds

hbase(main):012:0> alter 'employees', 'department details'
0 row(s) in 0.0500 seconds

hbase(main):013:0> enable 'employees'
0 row(s) in 2.0330 seconds

hbase(main):014:0> put 'employees', 'ID1', 'department details:deptname', 'Data Services'
0 row(s) in 0.0120 seconds

hbase(main):015:0> put 'employees', 'ID1', 'department details:deptno', 'HF001'
0 row(s) in 0.0090 seconds

hbase(main):016:0> put 'employees', 'ID2', 'department details:deptname ', 'HR Services'
0 row(s) in 0.0110 seconds

hbase(main):017:0> put 'employees', 'ID2', 'department details:deptno', 'HF100'
0 row(s) in 0.0230 seconds

hbase(main):018:0> scan 'employees'
ROW                                                   COLUMN+CELL
 ID1                                                  column=department details:deptname, timestamp=1634494092388, value=Data Services
 ID1                                                  column=department details:deptno, timestamp=1634494106819, value=HF001
 ID1                                                  column=private:address, timestamp=1634488875308, value=243 N. Wabash Av.
 ID1                                                  column=private:age, timestamp=1634493868979, value=35
 ID1                                                  column=private:firstname, timestamp=1634493776784, value=Ronaldlee
 ID1                                                  column=private:lastname, timestamp=1634493849374, value=Ejalu
 ID1                                                  column=private:ssn, timestamp=1634488547950, value=111-222-334
 ID1                                                  column=public:residentialstatus, timestamp=1634493973027, value=Yes
 ID2                                                  column=department details:deptname , timestamp=1634494119088, value=HR Services
 ID2                                                  column=department details:deptno, timestamp=1634494134338, value=HF100
 ID2                                                  column=private:age, timestamp=1634493942153, value=46
 ID2                                                  column=private:firstname, timestamp=1634493917274, value=Stevens
 ID2                                                  column=private:lastname, timestamp=1634493929152, value=Smith
 ID2                                                  column=private:ssn, timestamp=1634488712672, value=222-338-446
 ID2                                                  column=public:residentialstatus, timestamp=1634493988544, value=Yes
 ID3                                                  column=private:address, timestamp=1634488802627, value=123  State St.
3 row(s) in 0.0390 seconds

hbase(main):019:0>
```

NOTE: In order to add a new column family to an HBase table called test, you would need to run the following commands

disable 'test'
alter 'test', 'myNewFavoriteColumnFamily'
enable 'test'

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 4" at the top.