**CSC 555 Mining Big Data**

Assignment 4

**Due Tuesday, March 5th**

1. Consider a Hadoop job that will result in 78 blocks of output to HDFS.

Suppose that writing an output block to HDFS takes 1 minute. The HDFS replication factor is set to 3 (for simplicity, we charge reducers for the cost of writing replicated blocks).

* 1. 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).
  2. 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)
  3. 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)
  4. 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)
  5. 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)

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

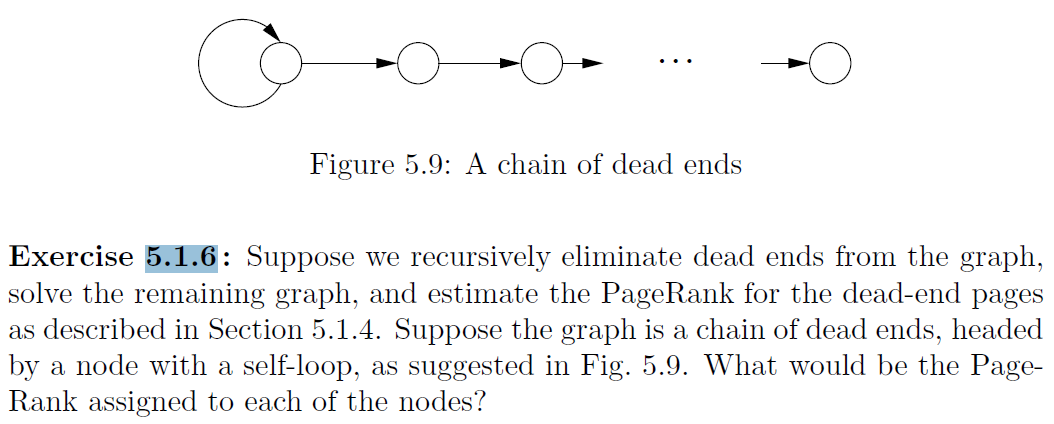
1. 1. Consider the following graph

Compute the page rank for the nodes in this graph. If you are multiplying matrices manually, you may stop after 5 steps. If you use a tool (e.g., Matlab, website, etc.) for matrix multiplication, you should get your answer to converge.

* 1. Now consider a dead-end node Q and P:

What is the page rank of Q?

* 1. Exercise 5.1.6 from Mining of Massive Datasets



1. Run another custom MapReduce job, implementing a solution for the following query:

For Employee(EID, EFirst, ELast, Phone) and Customer(CID, CFirst, CLast, Address), find everyone with the same name using MapReduce:

SELECT EFirst, ELast, Phone, Address

FROM Employee, Customer

WHERE EFirst = CFirst AND ELast = CLast;

You can use this input data:

http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/employee.txt

http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/customer.txt

Note that you can print all EID/Address values together (you do not have to print separate lines for each joined row that SQL query would output). Be sure to submit your python code, the command line and the screenshot of successful execution of your code.

1. In this section you will practice using HBase and setup Mahout and run the curve-clustering example.
   1. Note that HBase runs on top of HDFS, bypassing MapReduce (so only NameNode and DataNode need to be running). You can use your 4-node cluster or the 1-node cluster to run HBase, but be sure to specify which one you used.

For this exercise I elected to use my 1-node instance.

cd

(Download HBase)

wget http://rasinsrv07.cstcis.cti.depaul.edu/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'**

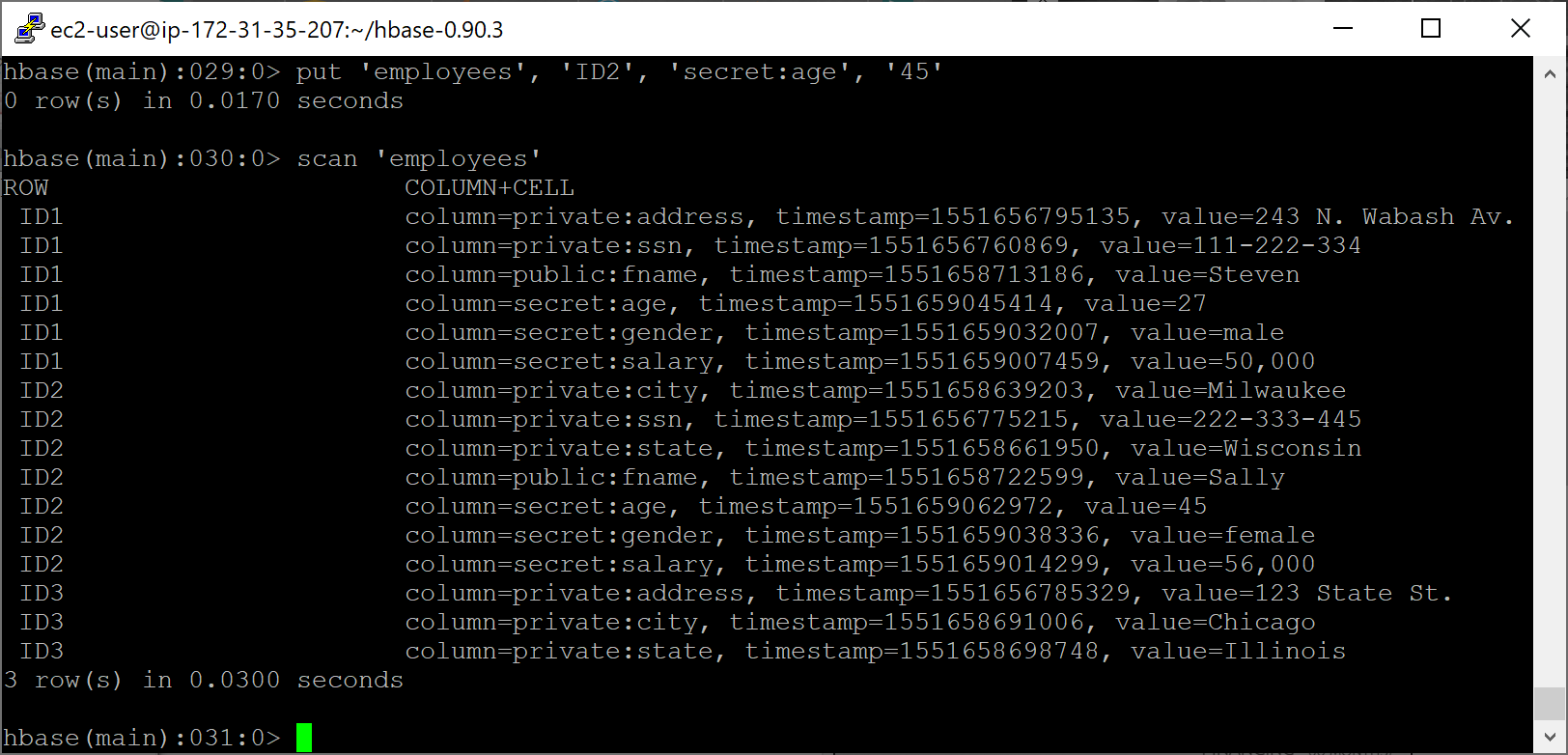
**put 'employees', 'ID2', 'private:ssn', '222-333-445'**

**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 2 new columns to the private family, 1 new column to the public family and create a brand new family with at least 3 columns. For each of these you should introduce at least 2 values -- so a total of (2+1+3) \* 2 = 12 values inserted. Verify that the table has been filled in properly with scan command and submit a screenshot.



Commands Used:

put 'employees', 'ID2', 'private:city', 'Milwaukee'

put 'employees', 'ID2', 'private:state', 'Wisconsin'

put 'employees', 'ID3', 'private:city', 'Chicago'

put 'employees', 'ID3', 'private:state', 'Illinois'

put 'employees', 'ID1', 'public:fname', 'Steven'

put 'employees', 'ID2', 'public:fname', 'Sally'

disable 'employees'

alter 'employees', {NAME => 'secret'}

enable 'employees'

put 'employees', 'ID1', 'secret:salary', '50,000'

put 'employees', 'ID2', 'secret:salary', '56,000'

put 'employees', 'ID1', 'secret:gender', 'male'

put 'employees', 'ID2', 'secret:gender', 'female'

put 'employees', 'ID1', 'secret:age', '27'

put 'employees', 'ID2', 'secret:age', '45'

* 1. Download and setup Mahout:

cd

(download mahout zip package)

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-mahout-distribution-0.11.2.zip

(Unzip the file)

unzip apache-mahout-distribution-0.11.2.zip

set the environment variables (as always, you can put these commands in ~/.bashrc to automatically set these variables every time you open a new connection, source ~/.bashrc to refresh)

export MAHOUT\_HOME=/home/ec2-user/apache-mahout-distribution-0.11.2

export PATH=/home/ec2-user/apache-mahout-distribution-0.11.2/bin:$PATH

be absolutely sure you set Hadoop home variable (if you haven’t):

Download and prepare synthetic data – it represents a list of 2D curves, represented as a 50-point vector.

Download the synthetic data example:

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/synthetic_control.data>

(make a testdata directory in HDFS, the example KMeans algorithm assumes the data lives there by default.

hadoop fs -mkdir -p testdatals

(copy the synthetic data over to the testdata directory on HDFS side. You can inspect the contents of the file by running nano synthetic\_control.data – as you can see this is a list of 600 vectors, with individual values separated by a space)

hadoop fs -put synthetic\_control.data testdata/

Please be sure to report the runtime of any command that includes “time”

time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job

real 5m16.566s

user 0m14.818s

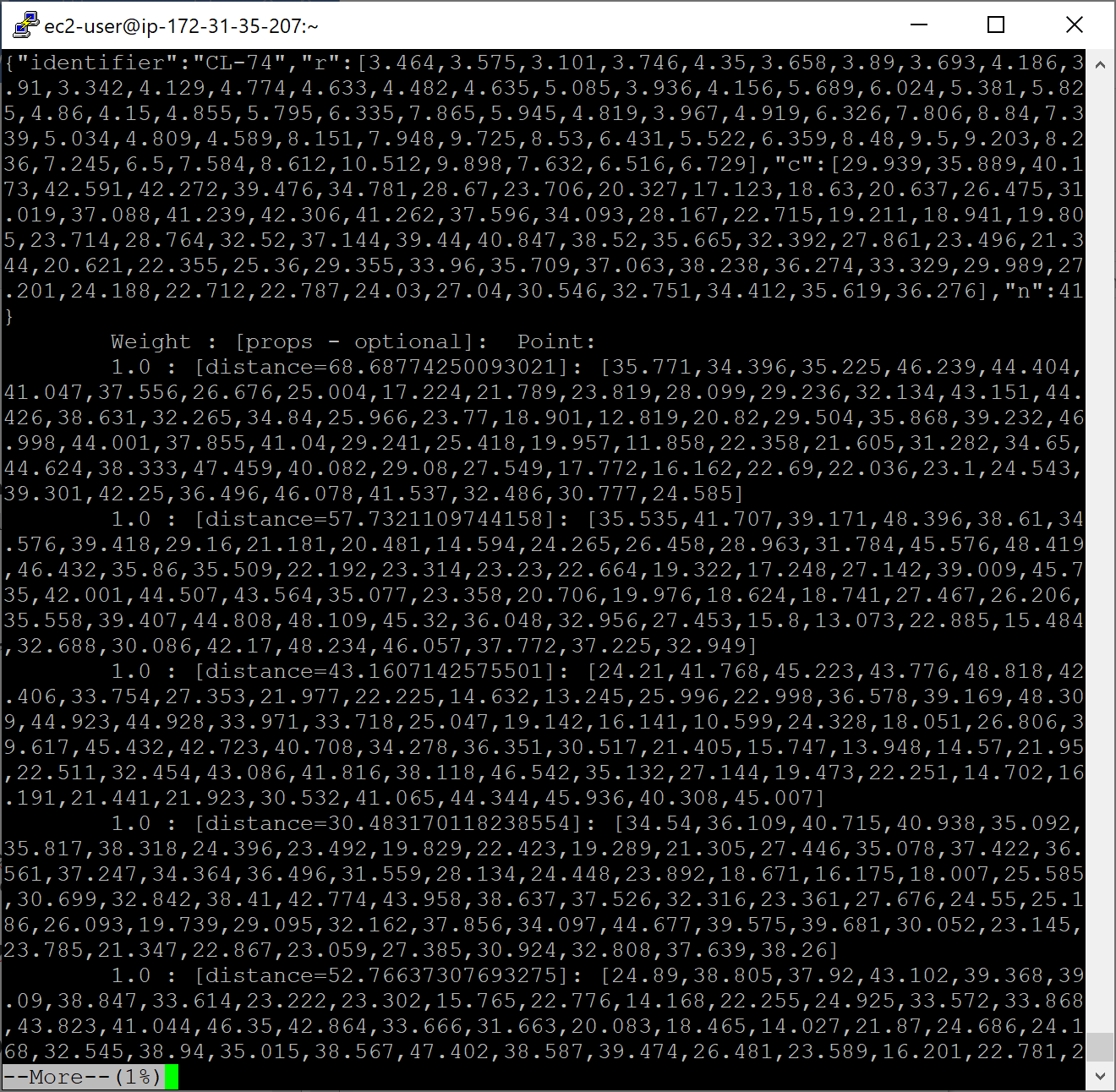
sys 0m3.608s

(clusterdump is a built-in Mahout command that will produce the result of KMeans. Output file is written to clusters-10-final because that is where the output is written after 10 iterations. The center points are placed in a separate file, called clusteredPoints)

mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --output clusteranalyze.txt

The file clusteranalyze.txt contains the results of the Kmeans run after 10 iterations.

Submit the screenshot of the first page from clusteranalyze.txt (e.g., from more clusteranalyze.txt)



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