CSC 555 Mining Big Data

Assignment 4

Lavinia Wang #1473704

Due Thursday, May 30th

1) Consider a Hadoop job that will result in 80 blocks of output to HDFS.
Suppose that reading a block takes 1 minute and writing an output block to HDFS takes 1 minute. The HDFS replication factor is set to 2 (therefore, it would cost reducer 2 block writes for each output block)

   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).

   32 minutes
   Calculation: 80/5 = 16 iterations
   16 iterations x 1 minute = 16 minutes
   16 minutes x 2 replication factor = 32 minutes

   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)

   8 minutes

   Calculation: 80/10 = 8 iterations
   8 iterations x 1 minute = 8 minutes
   8 minutes x 1 replication factor = 8 minutes

   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 2)

   16 minutes

   Calculation: 80/10 = 8 iterations
   8 iterations x 1 minute = 8 minutes
   8 minutes x 2 replication factor = 16 minutes

   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)

   1 minute

   Calculation: 80/100 = . 8, ceiling = 1 iteration
   1 iteration x 1 minute = 1 minute
   1 minute x 1 replication factor = 1 minute

e) Suppose that replication factor was changed to 3: how long will it take for the reducer to write the job output on a 5-node Hadoop cluster? (same question as a) but with replication of 3).

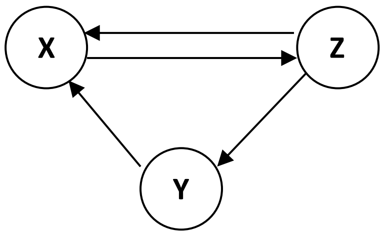
48 minutes

Calculation: 80/100 = 16 iteration
16 iteration x 1 minute = 16 minutes
16 minute x 3 replication factor = 48 minutes

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

2)

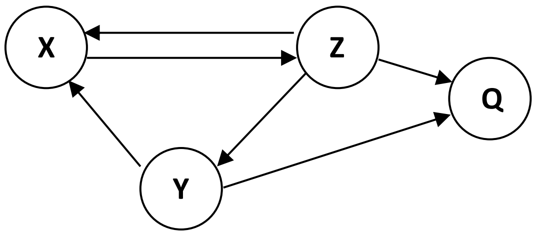
a) Consider the following graph



Compute the page rank for the nodes in this graph. If you are multiplying matrices manually, you may stop after 6 steps. If you use a tool (e.g., Matlab, website, etc.) for matrix multiplication, you should get your answer to converge.

		X	Y	Z		V		Rank
	X	0	1	1/2		2/5		2/5
Step 20	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		2/5		2/5

b) Now consider a dead-end node Q:



What is the page rank of Q?

		X	Y	Z		V		Rank
	X	0	1	1/2		2/5		2/5
Step 20	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		2/5		2/5

Q has two predecessors, Y and Z. Y has two successors, so contribute ½ of its page rank to Q. Z has three successors, so contribute 1/3 of its page rank to Q. Q's rank is calculated as:

1/5 * ½ + 2/5 * 1/3 = 7/30

c) Exercise 5.1.6 from Mining of Massive Datasets

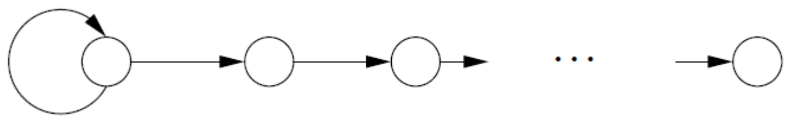


Figure 5.9: A chain of dead ends

Exercise 5.1.6: Suppose we recursively eliminate dead ends from the graph, solve the remaining graph, and estimate the PageRank for the dead-end pages as described in Section 5.1.4. Suppose the graph is a chain of dead ends, headed by a node with a self-loop, as suggested in Fig. 5.9. What would be the PageRank assigned to each of the nodes?

3) Given the input data [(1pm, $5), (2pm, $15), (3pm, $15), (4pm, $20), (5pm, $10), (6pm, $20), (7pm, $30), (8pm, $24), (9pm, $23), (10pm, $30), (11pm, $30), (12pm, $40)]. We will discuss Storm on Wednesday.

a) What will the Hive query "compute average price" return? (yes, this is as obvious as it seems, asked for comparison with part-b)

$21.83

b) What will a Storm query "compute average price per each 4 hour window" return? (tumbling, i.e., non-overlapping window of tuples, as many as you can fit). For example, the first window would 1pm-4pm, including 4 values.

1pm, 2om, 3pm, 4pm = $13.75
5pm, 6pm, 7pm, 8pm = $21
9pm, 10pm, 11pm, 12 pm = $30.75

c) What will a Storm query "compute average price per each 4 hour window" return? (sliding, i.e. overlapping window of tuples, moving the window forward 3 hours each time)

1pm, 2pm, 3pm, 4pm = $13.75
4pm, 5pm, 6pm, 7pm = $20
7pm, 8pm, 9pm, 10pm = $26.75
But 10pm, 11pm, 12pm has only two hours in the window, so Storm will not output anything.

Note, when Storm does not have a full window (e.g., only 2 results out of 4), you cannot output anything until the window fills.

4) In this section you will run another custom MapReduce job. You can either use Hadoop streaming or a Java implementation if you prefer.

   Implement and run MapReduce to compute: Count number of distinct odd integers in the input file – e.g., {1,2,3,1,5,2,3, 4, 4, 1} => should give you a count of 3 because 1, 3, and 5 are present at least once).
   You can generate the input file using this python code linked below (it will create a data file NumbersAndStrings.txt, run with **python numStrings.py**), it will take about a minute or so. Note that the text file contains numbers and non-numeric data. Your mapper will have to skip over elements of data that are not numbers.

   http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/numStrings.py

   Don't forget to submit the code you used, the output and screenshot of Hadoop streaming running (the final page).

myMapper.py

```
#!/usr/bin/python

import sys

# Create a list to store number only
number = []

for line in sys.stdin:
    line = line.strip()
    vals = line.split(" ")

    for val in vals:
      if type(val) == int:
            number.append(val)
            print (number)
```

myReducer.py

```
#!/usr/bin/python
import sys

unique_list = []

# The input comes from standard input (line by line)
for line in sys.stdin:
    line = line.strip()
    # parse the line and split it by '\t'
    ln = line.split('\t')

    for num in ln:
      if num %2 == 1:
          if num not in unique_list:
              unique_list.append(num)
```

```
print len(unique_list)
```

command

```
[ec2-user@ip-172-31-92-137~]$ hadoop jar hadoop-streaming-2.6.4.jar -
input /user/ec2-user/ -output /data/output -mapper myMapper.py -reducer
myReducer.py -file myReducer.py -file myMapper.py
```

5) In this section you will practice using HBase and setup Mahout and run the curve-clustering example that we discuss in class.

a) Note that HBase runs on top of HDFS, bypassing MapReduce (so only NameNode and DataNode need to be running). You can use your 3-node cluster or the 1-node cluster to run HBase, but specify which one you used.

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

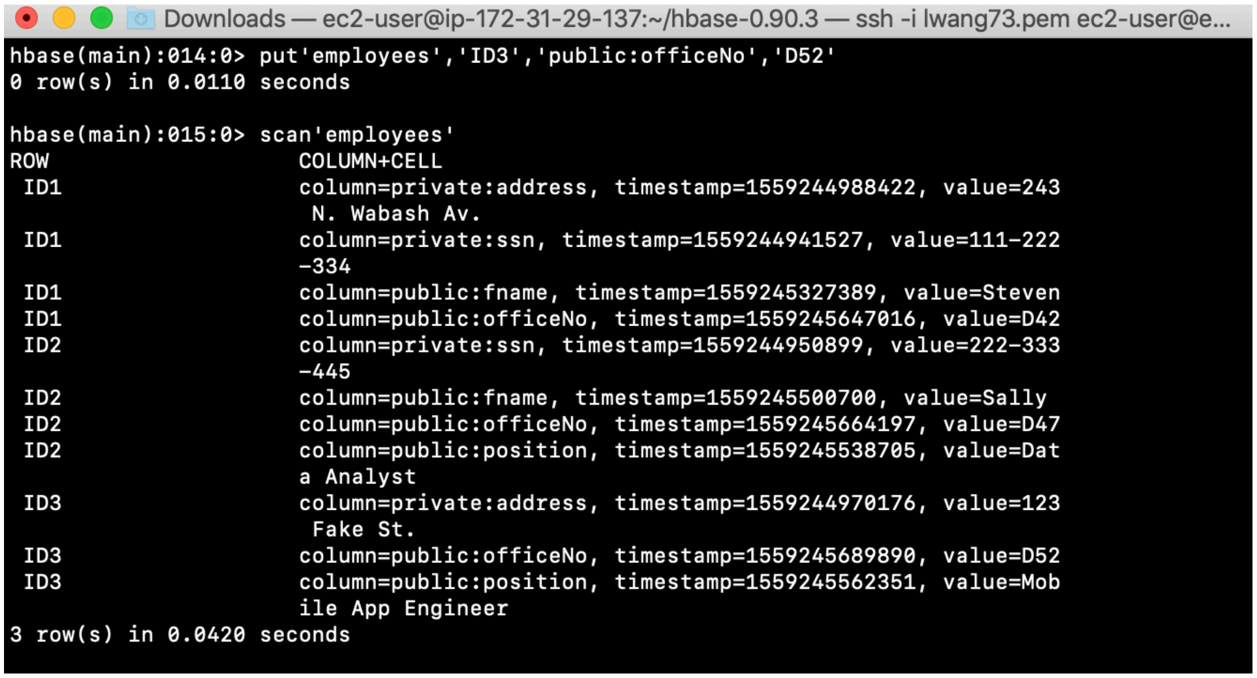
(Open the HBase shell – at this point jps should show HMaster)
bin/hbase shell

(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 Fake St.'
put 'employees', 'ID1', 'private:address', '243 N. Wabash Av.'

scan 'employees'

Now that we have filled in a few values, add at least 2 columns with at least 5 new values total to the “public” column family (e.g., position, officeNumber). Verify that the table has been filled in properly with scan command and submit a screenshot.

Lavinia Wang #1473704



Downloads — ec2-user@ip-172-31-29-137:~/hbase-0.90.3 — ssh -i lwang73.pem ec2-user@e...

```
hbase(main):014:0> put'employees','ID3','public:officeNo','D52'
0 row(s) in 0.0110 seconds

hbase(main):015:0> scan'employees'
ROW                   COLUMN+CELL
 ID1                  column=private:address, timestamp=1559244988422, value=243
                       N. Wabash Av.
 ID1                  column=private:ssn, timestamp=1559244941527, value=111-222
                      -334
 ID1                  column=public:fname, timestamp=1559245327389, value=Steven
 ID1                  column=public:officeNo, timestamp=1559245647016, value=D42
 ID2                  column=private:ssn, timestamp=1559244950899, value=222-333
                      -445
 ID2                  column=public:fname, timestamp=1559245500700, value=Sally
 ID2                  column=public:officeNo, timestamp=1559245664197, value=D47
 ID2                  column=public:position, timestamp=1559245538705, value=Dat
                      a Analyst
 ID3                  column=private:address, timestamp=1559244970176, value=123
                       Fake St.
 ID3                  column=public:officeNo, timestamp=1559245689890, value=D52
 ID3                  column=public:position, timestamp=1559245562351, value=Mob
                      ile App Engineer
3 row(s) in 0.0420 seconds
```

Command used:

```
put'employees', 'ID1','public:fname','Steven'
put'employees', 'ID2','public:fname','Sally'

put'employees', 'ID2','public:position','Data Analyst'
put'employees', 'ID3','public:position','Mobile App Enginee'

put'employees','ID1','public:officeNo','D42'
put'employees','ID2','public:officeNo','D47'
put'employees','ID3','public:officeNo','D52'
```

b) 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):

Lavinia Wang #1473704

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.

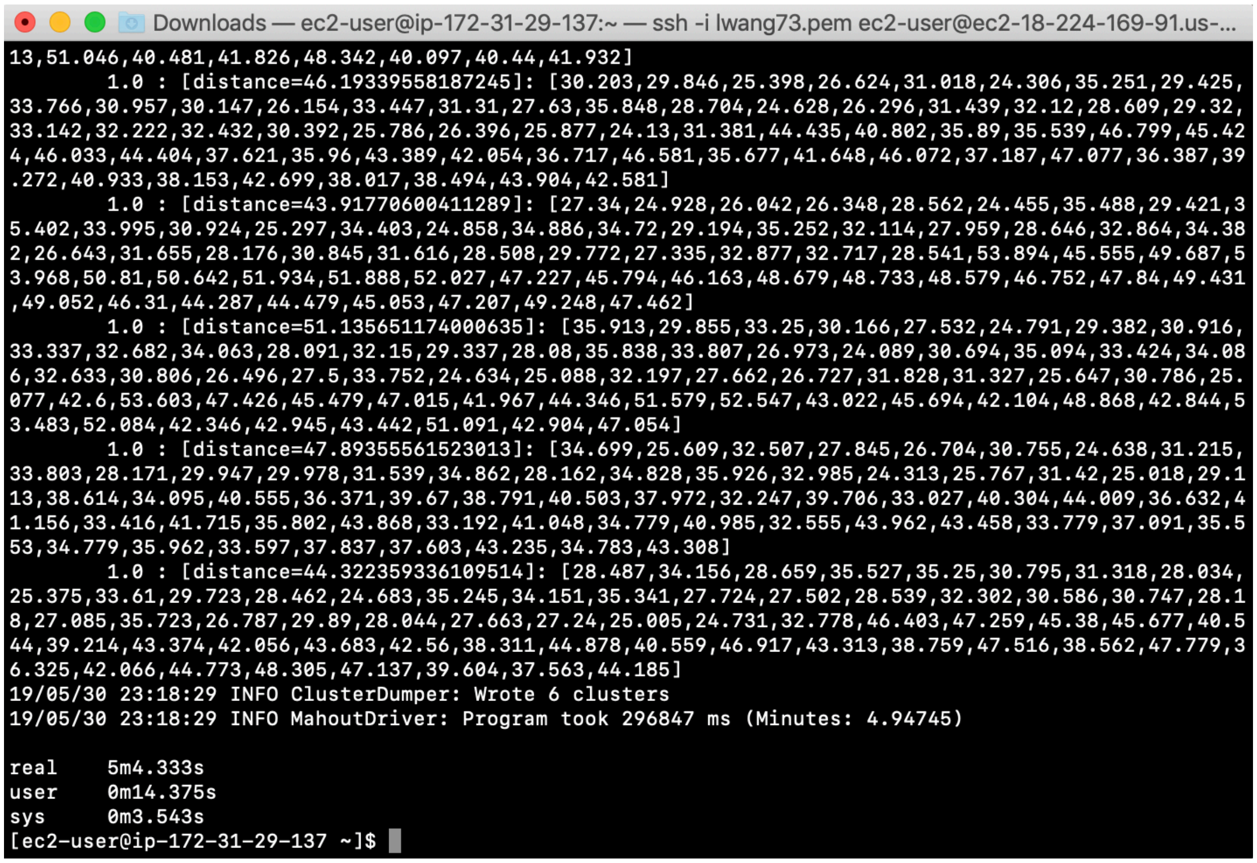
hadoop fs -mkdir -p testdata

(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



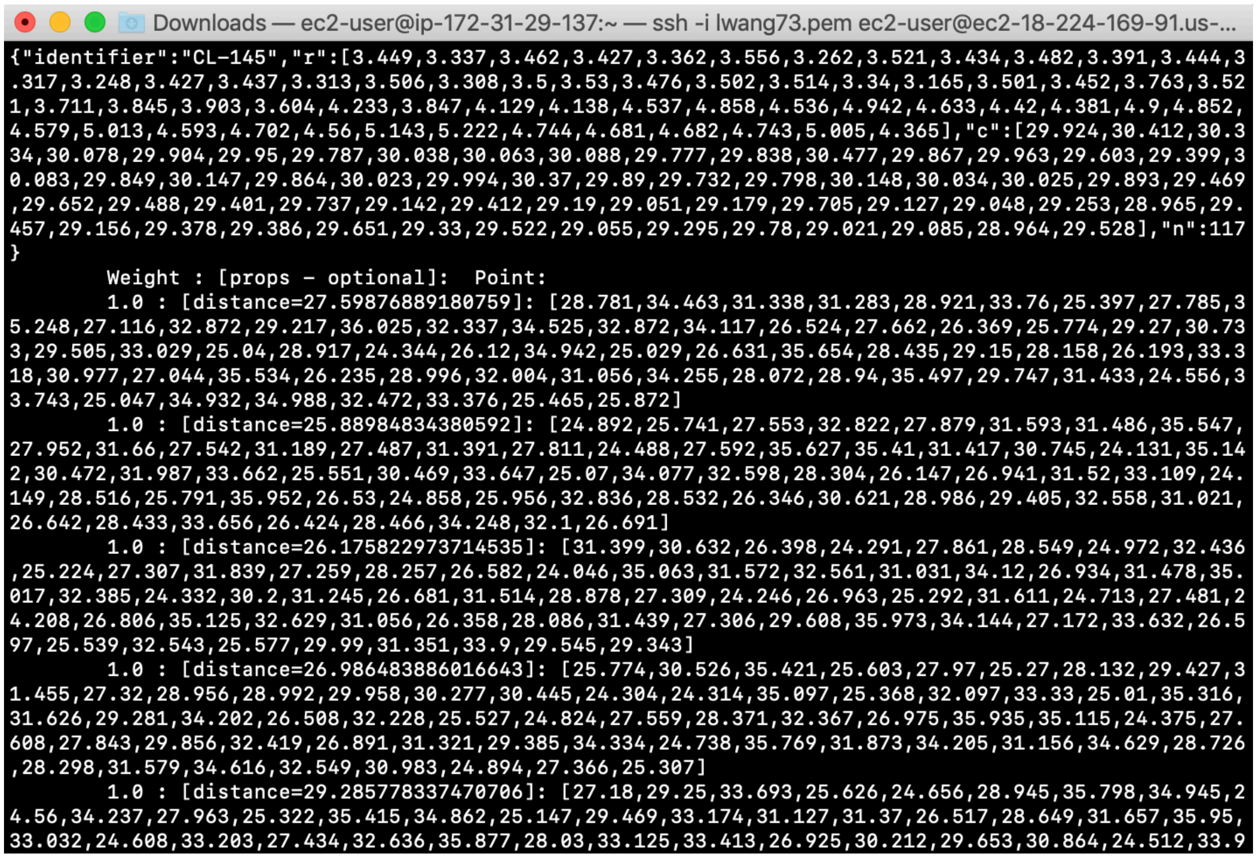
```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-224-169-91.us-...
13,51.046,40.481,41.826,48.342,40.097,40.44,41.932]
        1.0 : [distance=46.19339558187245]: [30.203,29.846,25.398,26.624,31.018,24.306,35.251,29.425,
33.766,30.957,30.147,26.154,33.447,31.31,27.63,35.848,28.704,24.628,26.296,31.439,32.12,28.609,29.32,
33.142,32.222,32.432,30.392,25.786,26.396,25.877,24.13,31.381,44.435,40.802,35.89,35.539,46.799,45.42
4,46.033,44.404,37.621,35.96,43.389,42.054,36.717,46.581,35.677,41.648,46.072,37.187,47.077,36.387,39
.272,40.933,38.153,42.699,38.017,38.494,43.904,42.581]
        1.0 : [distance=43.91770600411289]: [27.34,24.928,26.042,26.348,28.562,24.455,35.488,29.421,3
5.402,33.995,30.924,25.297,34.403,24.858,34.886,34.72,29.194,35.252,32.114,27.959,28.646,32.864,34.38
2,26.643,31.655,28.176,30.845,31.616,28.508,29.772,27.335,32.877,32.717,28.541,53.894,45.555,49.687,5
3.968,50.81,50.642,51.934,51.888,52.027,47.227,45.794,46.163,48.679,48.733,48.579,46.752,47.84,49.431
,49.052,46.31,44.287,44.479,45.053,47.207,49.248,47.462]
        1.0 : [distance=51.135651174000635]: [35.913,29.855,33.25,30.166,27.532,24.791,29.382,30.916,
33.337,32.682,34.063,28.091,32.15,29.337,28.08,35.838,33.807,26.973,24.089,30.694,35.094,33.424,34.08
6,32.633,30.806,26.496,27.5,33.752,24.634,25.088,32.197,27.662,26.727,31.828,31.327,25.647,30.786,25.
077,42.6,53.603,47.426,45.479,47.015,41.967,44.346,51.579,52.547,43.022,45.694,42.104,48.868,42.844,5
3.483,52.084,42.346,42.945,43.442,51.091,42.904,47.054]
        1.0 : [distance=47.89355561523013]: [34.699,25.609,32.507,27.845,26.704,30.755,24.638,31.215,
33.803,28.171,29.947,29.978,31.539,34.862,28.162,34.828,35.926,32.985,24.313,25.767,31.42,25.018,29.1
13,38.614,34.095,40.555,36.371,39.67,38.791,40.503,37.972,32.247,39.706,33.027,40.304,44.009,36.632,4
1.156,33.416,41.715,35.802,43.868,33.192,41.048,34.779,40.985,32.555,43.962,43.458,33.779,37.091,35.5
53,34.779,35.962,33.597,37.837,37.603,43.235,34.783,43.308]
        1.0 : [distance=44.322359336109514]: [28.487,34.156,28.659,35.527,35.25,30.795,31.318,28.034,
25.375,33.61,29.723,28.462,24.683,35.245,34.151,35.341,27.724,27.502,28.539,32.302,30.586,30.747,28.1
8,27.085,35.723,26.787,29.89,28.044,27.663,27.24,25.005,24.731,32.778,46.403,47.259,45.38,45.677,40.5
44,39.214,43.374,42.056,43.683,42.56,38.311,44.878,40.559,46.917,43.313,38.759,47.516,38.562,47.779,3
6.325,42.066,44.773,48.305,47.137,39.604,37.563,44.185]
19/05/30 23:18:29 INFO ClusterDumper: Wrote 6 clusters
19/05/30 23:18:29 INFO MahoutDriver: Program took 296847 ms (Minutes: 4.94745)

real    5m4.333s
user    0m14.375s
sys     0m3.543s
[ec2-user@ip-172-31-29-137 ~]$
```

(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.

Lavinia Wang #1473704



```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-224-169-91.us-...
{"identifier":"CL-145","r":[3.449,3.337,3.462,3.427,3.362,3.556,3.262,3.521,3.434,3.482,3.391,3.444,3
.317,3.248,3.427,3.437,3.313,3.506,3.308,3.5,3.53,3.476,3.502,3.514,3.34,3.165,3.501,3.452,3.763,3.52
1,3.711,3.845,3.903,3.604,4.233,3.847,4.129,4.138,4.537,4.858,4.536,4.942,4.633,4.42,4.381,4.9,4.852,
4.579,5.013,4.593,4.702,4.56,5.143,5.222,4.744,4.681,4.682,4.743,5.005,4.365],"c":[29.924,30.412,30.3
34,30.078,29.904,29.95,29.787,30.038,30.063,30.088,29.777,29.838,30.477,29.867,29.963,29.603,29.399,3
0.083,29.849,30.147,29.864,30.023,29.994,30.37,29.89,29.732,29.798,30.148,30.034,30.025,29.893,29.469
,29.652,29.488,29.401,29.737,29.142,29.412,29.19,29.051,29.179,29.705,29.127,29.048,29.253,28.965,29.
457,29.156,29.378,29.386,29.651,29.33,29.522,29.055,29.295,29.78,29.021,29.085,28.964,29.528],"n":117
}
        Weight : [props - optional]:  Point:
        1.0 : [distance=27.59876889180759]: [28.781,34.463,31.338,31.283,28.921,33.76,25.397,27.785,3
5.248,27.116,32.872,29.217,36.025,32.337,34.525,32.872,34.117,26.524,27.662,26.369,25.774,29.27,30.73
3,29.505,33.029,25.04,28.917,24.344,26.12,34.942,25.029,26.631,35.654,28.435,29.15,28.158,26.193,33.3
18,30.977,27.044,35.534,26.235,28.996,32.004,31.056,34.255,28.072,28.94,35.497,29.747,31.433,24.556,3
3.743,25.047,34.932,34.988,32.472,33.376,25.465,25.872]
        1.0 : [distance=25.88984834380592]: [24.892,25.741,27.553,32.822,27.879,31.593,31.486,35.547,
27.952,31.66,27.542,31.189,27.487,31.391,27.811,24.488,27.592,35.627,35.41,31.417,30.745,24.131,35.14
2,30.472,31.987,33.662,25.551,30.469,33.647,25.07,34.077,32.598,28.304,26.147,26.941,31.52,33.109,24.
149,28.516,25.791,35.952,26.53,24.858,25.956,32.836,28.532,26.346,30.621,28.986,29.405,32.558,31.021,
26.642,28.433,33.656,26.424,28.466,34.248,32.1,26.691]
        1.0 : [distance=26.175822973714535]: [31.399,30.632,26.398,24.291,27.861,28.549,24.972,32.436
,25.224,27.307,31.839,27.259,28.257,26.582,24.046,35.063,31.572,32.561,31.031,34.12,26.934,31.478,35.
017,32.385,24.332,30.2,31.245,26.681,31.514,28.878,27.309,24.246,26.963,25.292,31.611,24.713,27.481,2
4.208,26.806,35.125,32.629,31.056,26.358,28.086,31.439,27.306,29.608,35.973,34.144,27.172,33.632,26.5
97,25.539,32.543,25.577,29.99,31.351,33.9,29.545,29.343]
        1.0 : [distance=26.986483886016643]: [25.774,30.526,35.421,25.603,27.97,25.27,28.132,29.427,3
1.455,27.32,28.956,28.992,29.958,30.277,30.445,24.304,24.314,35.097,25.368,32.097,33.33,25.01,35.316,
31.626,29.281,34.202,26.508,32.228,25.527,24.824,27.559,28.371,32.367,26.975,35.935,35.115,24.375,27.
608,27.843,29.856,32.419,26.891,31.321,29.385,34.334,24.738,35.769,31.873,34.205,31.156,34.629,28.726
,28.298,31.579,34.616,32.549,30.983,24.894,27.366,25.307]
        1.0 : [distance=29.285778337470706]: [27.18,29.25,33.693,25.626,24.656,28.945,35.798,34.945,2
4.56,34.237,27.963,25.322,35.415,34.862,25.147,29.469,33.174,31.127,31.37,26.517,28.649,31.657,35.95,
33.032,24.608,33.203,27.434,32.636,35.877,28.03,33.125,33.413,26.925,30.212,29.653,30.864,24.512,33.9
```

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

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

2.a Full Calculation

		X	Y	Z		V		Rank
Step 1	X	0	1	1/2		1/3		1/2
	Y	0	0	1/2	x	1/3	=	1/6
	Z	1	0	0		1/3		1/3

		X	Y	Z		V		Rank
Step 2	X	0	1	1/2		1/2		1/3
	Y	0	0	1/2	x	1/6	=	1/6
	Z	1	0	0		1/3		1/2

		X	Y	Z		V		Rank
Step 3	X	0	1	1/2	x	1/3	=	5/12

Lavinia Wang #1473704

	Y	0	0	1/2		1/6		1/4
	Z	1	0	0		1/2		1/3

		X	Y	Z		V		Rank
	X	0	1	1/2		5/12		5/12
Step 4	Y	0	0	1/2	x	1/4	=	1/6
	Z	1	0	0		1/3		5/12

		X	Y	Z		V		Rank
	X	0	1	1/2		5/12		3/8
Step 5	Y	0	0	1/2	x	1/6	=	5/24
	Z	1	0	0		5/12		5/12

		X	Y	Z		V		Rank
	X	0	1	1/2		3/8		5/12
Step 6	Y	0	0	1/2	x	5/24	=	5/24
	Z	1	0	0		5/12		3/8

		X	Y	Z		V		Rank
	X	0	1	1/2		5/12		19/48
Step 7	Y	0	0	1/2	x	5/24	=	3/16
	Z	1	0	0		3/8		5/12

		X	Y	Z		V		Rank
	X	0	1	1/2		19/48		19/48
Step 8	Y	0	0	1/2	x	3/16	=	5/24
	Z	1	0	0		5/12		19/48

		X	Y	Z		V		Rank
	X	0	1	1/2		19/48		13/32
Step 9	Y	0	0	1/2	x	5/24	=	19/96
	Z	1	0	0		19/48		19/48

Lavinia Wang #1473704

		X	Y	Z		V		Rank
	X	0	1	1/2		13/32		19/48
Step 10	Y	0	0	1/2	x	19/96	=	19/96
	Z	1	0	0		19/48		13/32

		X	Y	Z		V		Rank
	X	0	1	1/2		19/48		77/192
Step 11	Y	0	0	1/2	x	19/96	=	13/64
	Z	1	0	0		13/32		19/48

		X	Y	Z		V		Rank
	X	0	1	1/2		77/192		77/192
Step 12	Y	0	0	1/2	x	13/64	=	19/96
	Z	1	0	0		19/48		77/192

		X	Y	Z		V		Rank
	X	0	1	1/2		77/192		51/128
Step 13	Y	0	0	1/2	x	19/96	=	77/384
	Z	1	0	0		77/192		77/192

		X	Y	Z		V		Rank
	X	0	1	1/2		51/128		77/192
Step 14	Y	0	0	1/2	x	77/384	=	77/384
	Z	1	0	0		77/192		51/128

		X	Y	Z		V		Rank
	X	0	1	1/2		77/192		307/768
Step 15	Y	0	0	1/2	x	77/384	=	51/256
	Z	1	0	0		51/128		77/192

		X	Y	Z		V		Rank
	X	0	1	1/2		307/768		307/768
Step 16	Y	0	0	1/2	x	51/256	=	77/384
	Z	1	0	0		77/192		307/768

		X	Y	Z		V		Rank
	X	0	1	1/2		307/768		205/512
Step 17	Y	0	0	1/2	x	77/384	=	1/5
	Z	1	0	0		307/768		307/768

		X	Y	Z		V		Rank
	X	0	1	1/2		205/512		2/5
Step 18	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		307/768		205/512

Lavinia Wang #1473704

		X	Y	Z		V		Rank
	X	0	1	1/2		2/5		2/5
Step 19	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		205/512		2/5

		X	Y	Z		V		Rank
	X	0	1	1/2		2/5		2/5
Step 20	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		2/5		2/5

		X	Y	Z		V		Rank
	X	0	1	1/2		2/5		2/5
Step 21	Y	0	0	1/2	x	1/5	=	1/5
	Z	1	0	0		2/5		2/5