ALEX TEBOUL - CSC 555 Mining Big Data

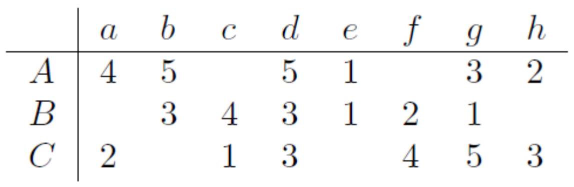
Assignment 5

Due Saturday, 3/14

Suggested Reading: Hadoop: The Definitive Guide Ch19; Mining of Massive Datasets: Ch9

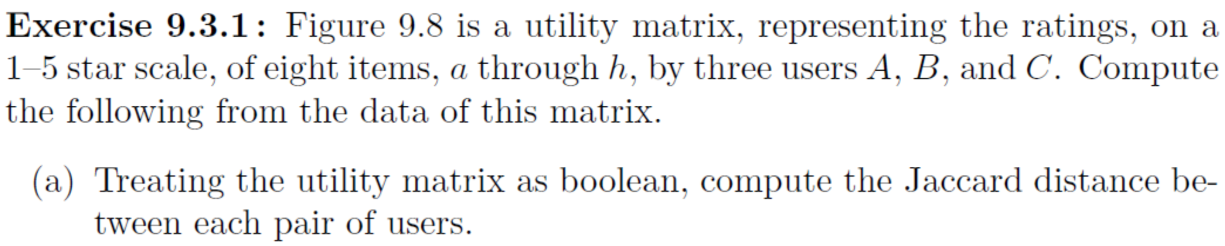
1)

a) Solve 9.3.1-a (normalize the ratings based on a threshold), 9.3.1-e



	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

Figure 9.8: A utility matrix for exercises



Exercise 9.3.1 : Figure 9.8 is a utility matrix, representing the ratings, on a 1–5 star scale, of eight items, a through h, by three users A, B, and C. Compute the following from the data of this matrix.

(a) Treating the utility matrix as boolean, compute the Jaccard distance between each pair of users.

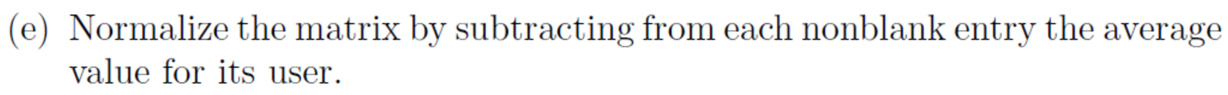
Jaccard Distances between user pairs:

➔ A & B: 1 – 2/5 = 3/5 ~ 0.6

➔ B & C: 1 – 1/6 = 5/6 ~ 0.833

➔ A & C: 1 – 2/6 = 2/3 ~ 0.666

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
A	1	1		1			1	
B		1	1	1				
C				1		1	1	1



(e) Normalize the matrix by subtracting from each nonblank entry the average value for its user.

➔ Get Average

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**Average**
A	4	5		5	1		3	2	3.333
B		3	4	3	1	2	1		2.333
C	2		1	3		4	5	3	3

→The matrix normalized by subtraction

	a	b	c	d	e	f	g	h
A	0.67	1.67		1.67	-2.33		-0.33	-0.67
B		0.67	1.67	0.67	-1.33	-0.33	-1.33	
C	-1		-2	0		1	2	0

b) Describe a strategy that is used to make a utility matrix less sparse

The mmds textbook Chapter 9 explains that you can cluster items and/or users in order to deal with a sparse utility matrix. This is important because it can be hard to detect similarity among either items or users when there is little information about item-user pairs in a sparse utility matrix.

2)

a) How does Spark ensure that data is not lost when a failure occurs?

➔ With Spark, data should be read from HDFS to memory for processing, so it's stored in HDFS as well ideally. Replication of the data should be sufficient to account for failures.

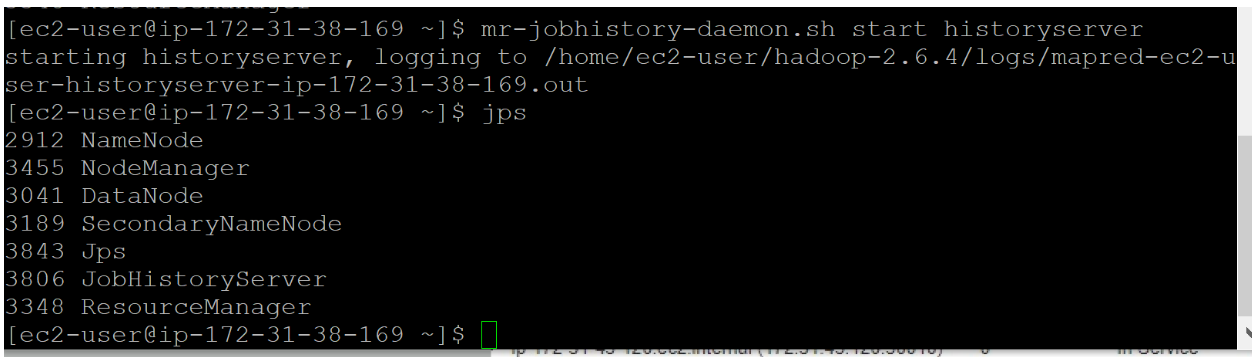
b) From a resource manage perspective, which Hadoop nodes should be chosen to run Spark tasks?

➔ Seems like the Spark tasks should be run locally with the nodes that have the appropriate data – wherever the data is then run the task there.

3)

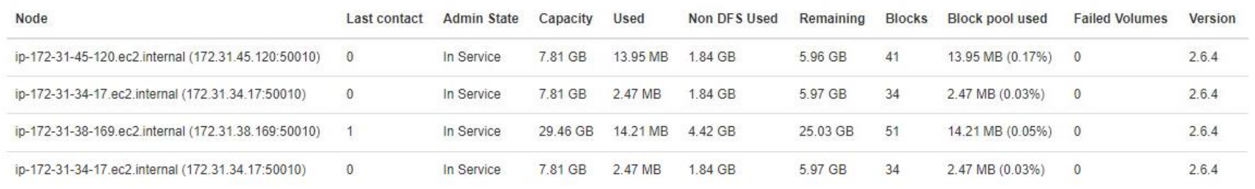
a) Add one more node to your existing cluster (e.g., go from 3 to 4 nodes) following the instructions from the previous assignment and examples in class. You can do that by creating a new AWS instance, setting up ssh access (public-private key) to that instance and copying Hadoop to that new instance as you have with two other workers before. Keep in mind that you do not need to configure anything again except for editing the slaves file to reference the new worker node private IP. Everything else should be taken care of by your already existing cluster setup.
Submit a screenshot of the new cluster view.

➔ Show Running



```
[ec2-user@ip-172-31-38-169 ~]$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/ec2-user/hadoop-2.6.4/logs/mapred-ec2-u
ser-historyserver-ip-172-31-38-169.out
[ec2-user@ip-172-31-38-169 ~]$ jps
2912 NameNode
3455 NodeManager
3041 DataNode
3189 SecondaryNameNode
3843 Jps
3806 JobHistoryServer
3348 ResourceManager
[ec2-user@ip-172-31-38-169 ~]$
```

➔ Show 4 node Cluster at http://54.242.68.30:50070/dfshealth.html#tab-datanode



Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-45-120.ec2.internal (172.31.45.120:50010)	0	In Service	7.81 GB	13.95 MB	1.84 GB	5.96 GB	41	13.95 MB (0.17%)	0	2.6.4
ip-172-31-34-17.ec2.internal (172.31.34.17:50010)	0	In Service	7.81 GB	2.47 MB	1.84 GB	5.97 GB	34	2.47 MB (0.03%)	0	2.6.4
ip-172-31-38-169.ec2.internal (172.31.38.169:50010)	1	In Service	29.46 GB	14.21 MB	4.42 GB	25.03 GB	51	14.21 MB (0.05%)	0	2.6.4
ip-172-31-34-17.ec2.internal (172.31.34.17:50010)	0	In Service	7.81 GB	2.47 MB	1.84 GB	5.97 GB	34	2.47 MB (0.03%)	0	2.6.4

b) Pick one of the hadoop streaming tasks (from this or previous homework) and run it as-is on the new cluster. Record the time it took (you can time a command by prepending it with time, e.g., **time hadoop jar**…). You do not need to write any new code, just time one of your existing examples.

➔ Chose to use the streaming task from Phase1 of the project assignment

Implement query 0.3 using Hadoop streaming with python. You don't need to implement other queries.
--Q0.3 SELECT lo_quantity, SUM(lo_revenue) FROM lineorder WHERE lo_discount BETWEEN 3 AND 5 GROUP BY lo_quantity;

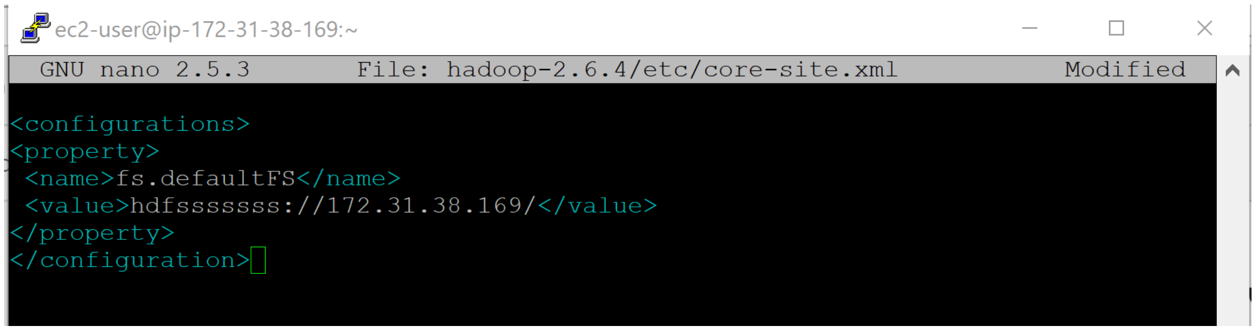
hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input /user/ec2user/streaming -output /data/streaming03 -mapper mapper03.py -reducer reducer03.py -file mapper03.py -file reducer03.py

➔ real 0m 37.401s
➔ user 0m 3.104s
➔ sys 0m 0.220s

c) Repeat the previous task (3-b), but shut down one of the nodes (from Amazon console, imitating a failure) **while the task** is running. Record the time it took. How does it compare to the previous execution?

➔ real 1m 24.664s
➔ user 0m 4.055s
➔ sys 0m 0.364s
➔ TIME Increase on the task with failure

d) Finally, modify one of the configuration files in Hadoop to introduce a typo (you can do that on the cluster or on the single-node setup) so it produces an error when you start dfs or yarn. Take a screenshot of the modified config file and the corresponding error message that you received.



```
  GNU nano 2.5.3          File: hadoop-2.6.4/etc/core-site.xml          Modified

<configurations>
<property>
 <name>fs.defaultFS</name>
 <value>hdfsssssss://172.31.38.169/</value>
</property>
</configuration>
```

- ➔ Exception in thread "main" java.lang.IllegalArgumentException: Invalid URI for NameNode address (check fs.defaultFS): hdfsssssss://172.31.38.169/ is not of scheme 'hdfs'.
- ➔ ^Main error code

4) Run a recommender on the MoveLens dataset.

(Create a directory for movie lens dataset)

```
mkdir Movielens
cd Movielens
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/ml-1m.zip
```

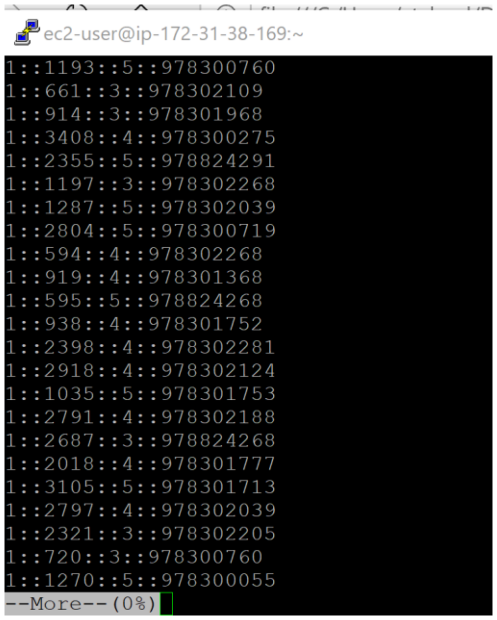
(Unzip the dataset, this one happens to be compressed with Zip rather than GZip)

```
unzip ml-1m.zip
cd ..
```

Take a look at the data file:

```
more Movielens/ml-1m/ratings.dat
```

(you can press q or Ctrl-C to exit, **more** command shows the first few lines worth of text. Each line contains user ID, movie ID, user rating and the timestamp of the rating, as already discussed in class)



```
ec2-user@ip-172-31-38-169:~
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
1::938::4::978301752
1::2398::4::978302281
1::2918::4::978302124
1::1035::5::978301753
1::2791::4::978302188
1::2687::3::978824268
1::2018::4::978301777
1::3105::5::978301713
1::2797::4::978302039
1::2321::3::978302205
1::720::3::978300760
1::1270::5::978300055
--More--(0%)
```

The next step is to use aa Linux command to convert :: separated file into a comma-separated file. First part (cat) will simply output the file. Second part substitutes , for :: and third part of the command extracts just 3 attributes relevant to us (no timestamp)

cat Movielens/ml-1m/ratings.dat | sed -e s/::/,/g| cut -d, -f1,2,3 > Movielens/ml-1m/ratings.csv

(NOTE: if you wanted to extract all 4 columns from the original data set, you could run the same command with "1,2,3,4" instead of "1,2,3").

Create a movielens directory and copy the articles over to HDFS into that directory:

$HADOOP_HOME/bin/hadoop fs -mkdir movielens
$HADOOP_HOME/bin/hadoop fs -put Movielens/ml-1m/ratings.csv movielens

Split the data set into the 90% training set and 10% evaluation set. In this case we are using Hadoop to perform the split. Naturally, you can change the percentages here to any other value instead of 0.9/0.1. bin/mahout will only work from the $MAHOUT_HOME directory, or you can change it as others.

bin/mahout splitDataset –input movielens/ratings.csv –output ml_dataset –trainingPercentage 0.9 –probePercentage 0.1 –tempDir dataset/tmp

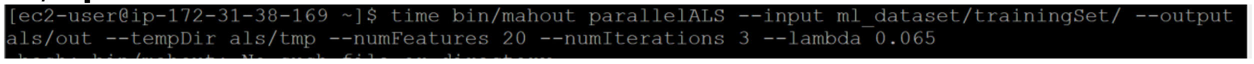
<u>Verify and report</u> the file sizes of the input ratings.csv file and the two sampled files (the two files are in the /user/ec2-user/ml_dataset/trainingSet/ and /user/ec2-user/ml_dataset/probeSet directories on HDFS side). Do the sampled file sizes add up to the original input file size?

- ➔ Yes, they add up to the original input file size of roughly 11.55MB
- ➔ 1,156,339 bytes + 10,397,117 bytes = 11,553,456 bytes

```
-rw-r--r--   2 ec2-user supergroup   11553456 2020-03-14 05:42 /user/ec2-
user/movielens/ratings.csv
-rw-r--r--   2 ec2-user supergroup   10397117 2020-03-14 05:45 /user/ec2-
user/ml_dataset/trainingSet/part-m-00000
-rw-r--r--   2 ec2-user supergroup    1156339 2020-03-14 05:45 /user/ec2-
user/ml_dataset/probeSet/part-m-00000
```

Factorize the rating matrix based on the training set. As always, this is a single line command, be sure to run it as such. The --numfeatures value configures the set of "hidden" variables or the dimension size to use in matrix factorization. --numIterations sets how many passes to perform; we expect a better match with more iterations

```
time bin/mahout parallelALS –input ml_dataset/trainingSet/ –output als/out –tempDir
als/tmp –numFeatures 20 –numIterations 3 –lambda 0.065
```



```
[ec2-user@ip-172-31-38-169 ~]$ time bin/mahout parallelALS --input ml_dataset/trainingSet/ --output
als/out --tempDir als/tmp --numFeatures 20 --numIterations 3 --lambda 0.065
```

- ➔ real 4m 01.023s
- ➔ user 0m 13.226s
- ➔ sys 0m 3.467s

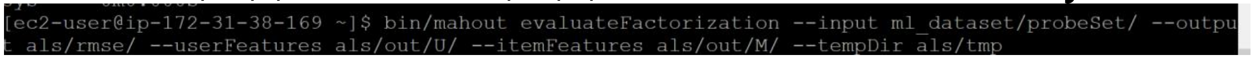
Measure the prediction against the training set:

```
bin/mahout evaluateFactorization –input ml_dataset/probeSet/ –output als/rmse/ –
userFeatures als/out/U/ –itemFeatures als/out/M/ –tempDir als/tmp
```

<u>**What is the resulting RMSE value?**</u> (rmse.txt file in /user/ec2-user/als/rmse/ on HDFS)

Finally, let's generate some predictions:

```
bin/mahout recommendfactorized –input als/out/userRatings/ –output recommendations/ –
userFeatures als/out/U/ –itemFeatures als/out/M/ –numRecommendations 6 –maxRating 5
```



```
[ec2-user@ip-172-31-38-169 ~]$ bin/mahout evaluateFactorization --input ml_dataset/probeSet/ --outpu
t als/rmse/ --userFeatures als/out/U/ --itemFeatures als/out/M/ --tempDir als/tmp
```

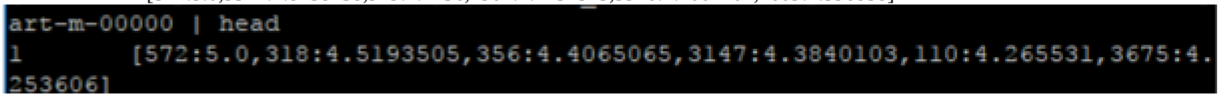
- ➔ RMSE ~ 0.88

Look at recommendations/part-m-00000 and report the first 10 rows by running the following command. These are top-6 recommendations (note that --numRecommendation setting in the previous command) for each user. Each recommendation consists of movieID and the estimated rating that the user might give to that movie.

Ex.

```
➔ [572:5.0,2197:4.6908126,3314:4.687907,2156:4.4995503,356:4.382499,3222:4.3752384]
➔ [572:4.8178015,2197:4.649721,527:4.455281,1721:4.318738,2762:4.2395806,2324:4.1847873]
➔ ***[572:4.753747,2913:4.745884,318:4.6085696,2762:4.569885,110:4.56089,3443:4.523036]
➔ ****[3245:5.0,3092:5.0,923:5.0,1423:5.0,2905:5.0,1212:5.0]
➔ *****[1423:4.7975807,3245:4.595153,668:4.564381,1002:4.4648323,3570:4.428102,3645:4.3494987]
```

- ➔ [572:5.0,3314:5.0,2197:4.848169,3675:4.432844,3916:4.426566,3161:4.3900433]
- ➔ [1036:4.686574,1240:4.6430492,2494:4.614736,1198:4.608802,3508:4.5733657,3552:4.5691805]
- ➔ [858:4.798158,2905:4.676086,318:4.6493897,1198:4.646571,50:4.6085334,3038:4.587847]
- ➔ [50:4.3938804,858:4.342568,296:4.325224,2329:4.26149,2905:4.2136726,110:4.213341]
- ➔ [572:5.0,3314:4.8458138,919:4.4136,1907:4.4132323,3916:4.4002104,2609:4.396858]



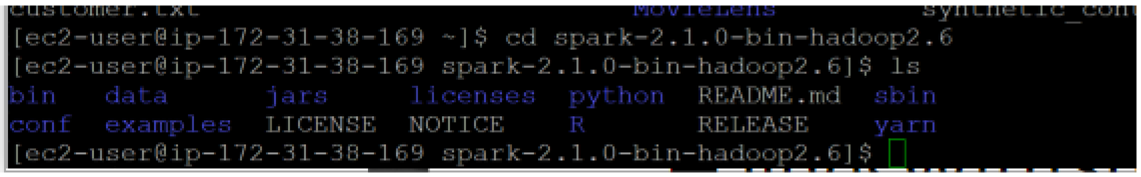
```
art-m-00000 | head
1       [572:5.0,318:4.5193505,356:4.4065065,3147:4.3840103,110:4.265531,3675:4.
253606]
```

$HADOOP_HOME/bin/hadoop fs -cat recommendations/part-m-00000 | head

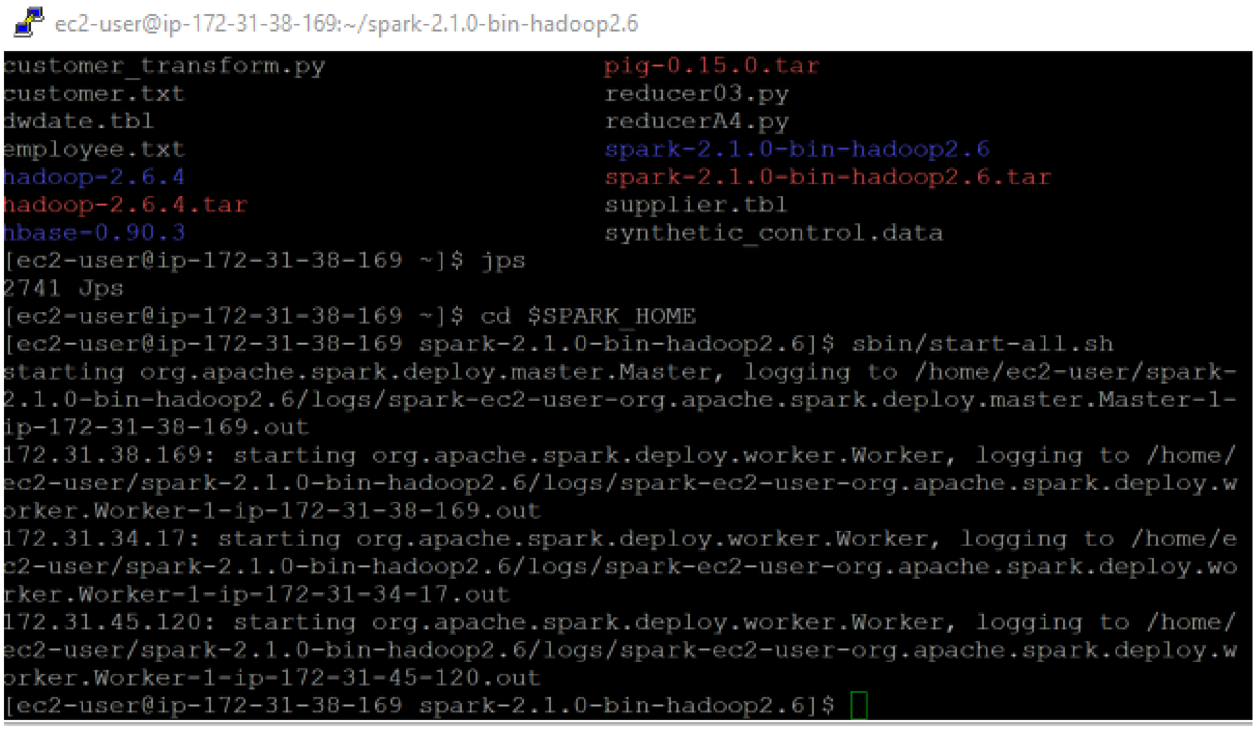
What is the top movie recommendation (movie ID) for users 3, 4 and 5?

- ➔ I believe this would be 572, 3245, and 1423 respectively for 3, 4, and 5 from what can be seen above, but it seems like they are pretty close to others in the top 6 recs.

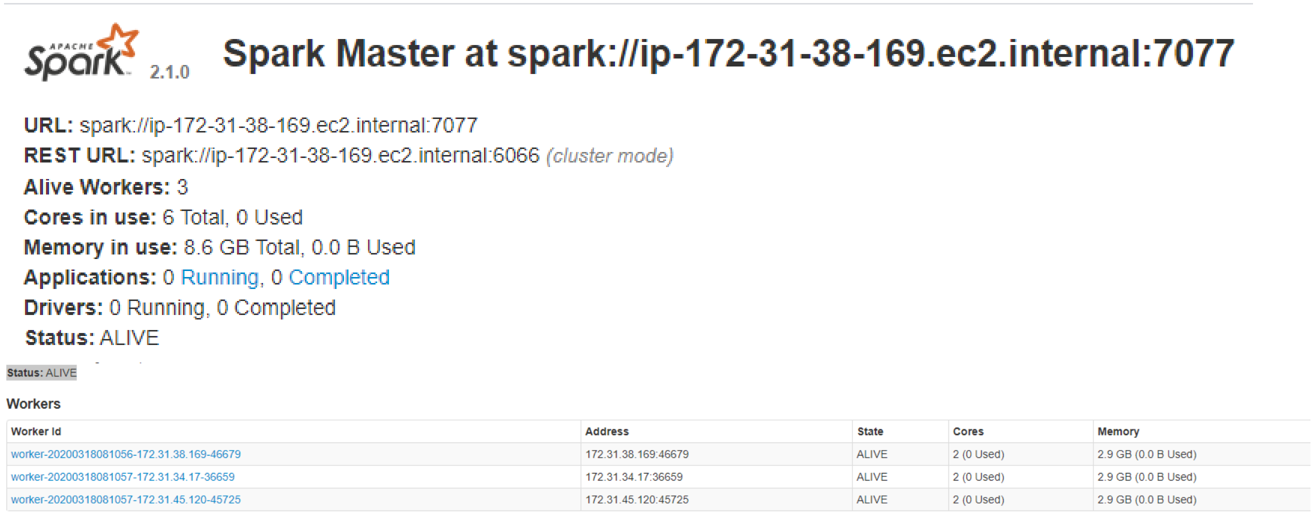
5) Set up stand-alone minimum 3-node Spark cluster (instructions available at http://spark.apache.org/docs/latest/spark-standalone.html). Note that you can use your existing cluster, you just need to configure hadoop-env.sh and add slaves file to the conf directory in the spark folder. Cluster view status can be seen at the browser page at port 8080 (you would need to modify your firewall settings to open port 8080 exactly as you have with port 50700). You can find a spark binary of the right version here: http://dbgroup.cdm.depaul.edu/Courses/CSC555/spark-2.1.0-bin-hadoop2.6.tar



```
customer.txt                          MovieLens          synthetic_con
[ec2-user@ip-172-31-38-169 ~]$ cd spark-2.1.0-bin-hadoop2.6
[ec2-user@ip-172-31-38-169 spark-2.1.0-bin-hadoop2.6]$ ls
bin   data      jars     licenses  python   README.md  sbin
conf  examples  LICENSE  NOTICE    R        RELEASE    yarn
[ec2-user@ip-172-31-38-169 spark-2.1.0-bin-hadoop2.6]$
```


ec2-user@ip-172-31-38-169:~/spark-2.1.0-bin-hadoop2.6

```
customer_transform.py                       pig-0.15.0.tar
customer.txt                                reducer03.py
dwdate.tbl                                  reducerA4.py
employee.txt                                spark-2.1.0-bin-hadoop2.6
hadoop-2.6.4                                spark-2.1.0-bin-hadoop2.6.tar
hadoop-2.6.4.tar                            supplier.tbl
hbase-0.90.3                                synthetic_control.data
[ec2-user@ip-172-31-38-169 ~]$ jps
2741 Jps
[ec2-user@ip-172-31-38-169 ~]$ cd $SPARK_HOME
[ec2-user@ip-172-31-38-169 spark-2.1.0-bin-hadoop2.6]$ sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/ec2-user/spark-
2.1.0-bin-hadoop2.6/logs/spark-ec2-user-org.apache.spark.deploy.master.Master-1-
ip-172-31-38-169.out
172.31.38.169: starting org.apache.spark.deploy.worker.Worker, logging to /home/
ec2-user/spark-2.1.0-bin-hadoop2.6/logs/spark-ec2-user-org.apache.spark.deploy.w
orker.Worker-1-ip-172-31-38-169.out
172.31.34.17: starting org.apache.spark.deploy.worker.Worker, logging to /home/e
c2-user/spark-2.1.0-bin-hadoop2.6/logs/spark-ec2-user-org.apache.spark.deploy.wo
rker.Worker-1-ip-172-31-34-17.out
172.31.45.120: starting org.apache.spark.deploy.worker.Worker, logging to /home/
ec2-user/spark-2.1.0-bin-hadoop2.6/logs/spark-ec2-user-org.apache.spark.deploy.w
orker.Worker-1-ip-172-31-45-120.out
[ec2-user@ip-172-31-38-169 spark-2.1.0-bin-hadoop2.6]$
```

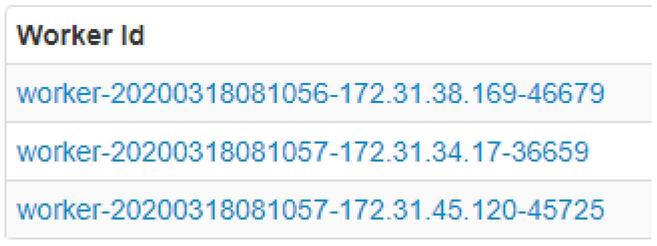

2.1.0 Spark Master at spark://ip-172-31-38-169.ec2.internal:7077

URL: spark://ip-172-31-38-169.ec2.internal:7077
REST URL: spark://ip-172-31-38-169.ec2.internal:6066 *(cluster mode)*
Alive Workers: 3
Cores in use: 6 Total, 0 Used
Memory in use: 8.6 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20200318081056-172.31.38.169-46679	172.31.38.169:46679	ALIVE	2 (0 Used)	2.9 GB (0.0 B Used)
worker-20200318081057-172.31.34.17-36659	172.31.34.17:36659	ALIVE	2 (0 Used)	2.9 GB (0.0 B Used)
worker-20200318081057-172.31.45.120-45725	172.31.45.120:45725	ALIVE	2 (0 Used)	2.9 GB (0.0 B Used)



Worker Id
worker-20200318081056-172.31.38.169-46679
worker-20200318081057-172.31.34.17-36659
worker-20200318081057-172.31.45.120-45725

**Enlarged:
172.31.38.169 & 172.31.45.120 & 172.31.34.17

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 5" at the top.