CSC 555 Mining Big Data

Assignment 5

Lavinia Wang 1473704

Due Sunday, 6/9

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.

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

A & B: Distance = 3/5 (0.6)

A & C: Distance = 6/7 (0.86)

B & C: Distance = 5/6 (0.83)

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

	a	b	c	d	e	f	g	h
A	5/6	1 5/6		1 5/6	-2 1/6		- 1/6	-2 1/6
B		2/3	1 2/3	2/3	-1 1/3	- 1/3	-1 1/3	
C	-1		-2	0		1	2	0

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

One strategy is to cluster items and/or users based upon a distance measure. For example, items can be clustered into a smaller subset and the values assigned would be the average ratings for each user for the columns in each cluster.

Lavinia Wang 1473704

2)

a) Where does Spark typically read the data from (and how does it ensure that data is not lost when a failure occurs)?

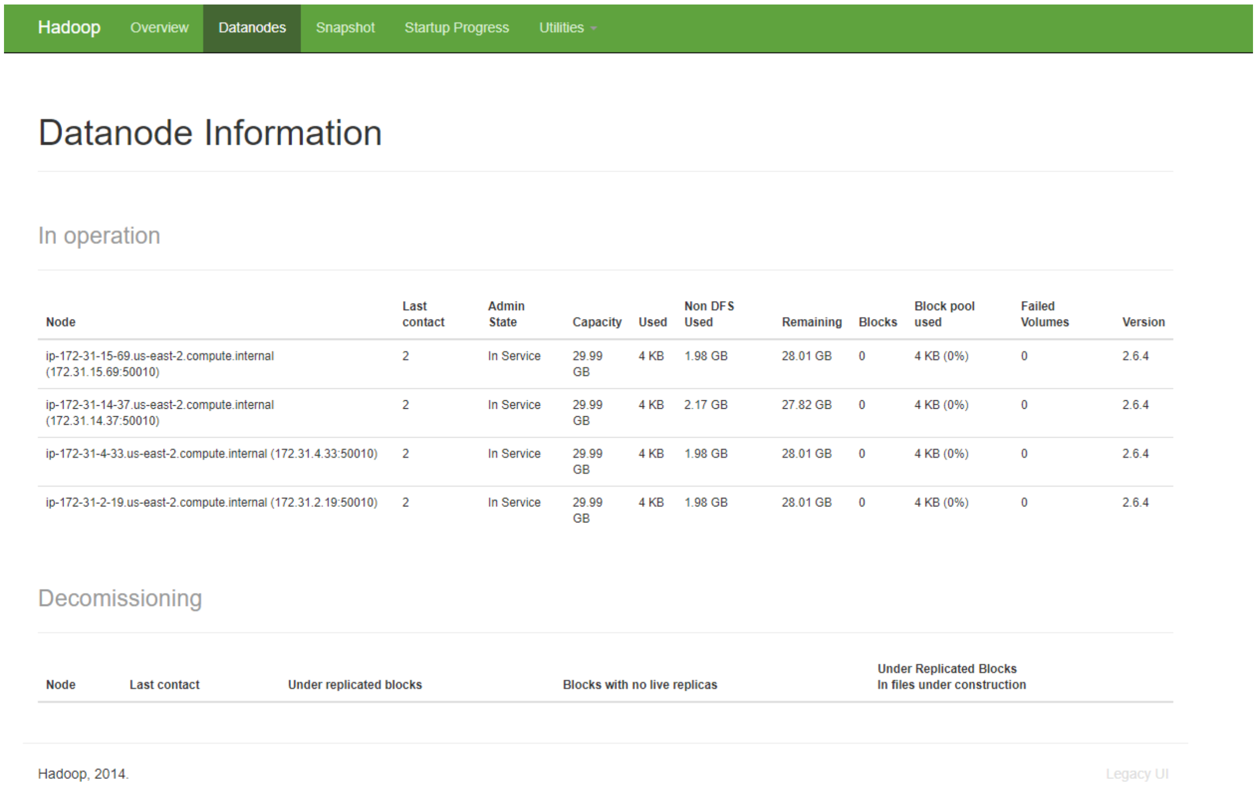
Spark typically reads data from HDFS into memory for processing but is otherwise stored in HDFS, thus preventing data loss during failure, provided sufficient replication is defined for HDFS.

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

The NameNode is the Hadoop master, which allocate resources across applications.

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.



Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-15-69.us-east-2.compute.internal (172.31.15.69:50010)	2	In Service	29.99 GB	4 KB	1.98 GB	28.01 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-14-37.us-east-2.compute.internal (172.31.14.37:50010)	2	In Service	29.99 GB	4 KB	2.17 GB	27.82 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-4-33.us-east-2.compute.internal (172.31.4.33:50010)	2	In Service	29.99 GB	4 KB	1.98 GB	28.01 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-2-19.us-east-2.compute.internal (172.31.2.19:50010)	2	In Service	29.99 GB	4 KB	1.98 GB	28.01 GB	0	4 KB (0%)	0	2.6.4

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

Hadoop, 2014.

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.

From Project Phase 1:

```
time hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/ssbm -output
/data/output1 -mapper myMapper.0.3.py -reducer myReducer.0.3.py -file
myReducer.0.3.py -file myMapper.0.3.py

19/06/09 11:27:34 INFO streaming.StreamJob: Output directory: /data/output2

real    0m40.282s
user    0m3.801s
sys     0m0.225s
```

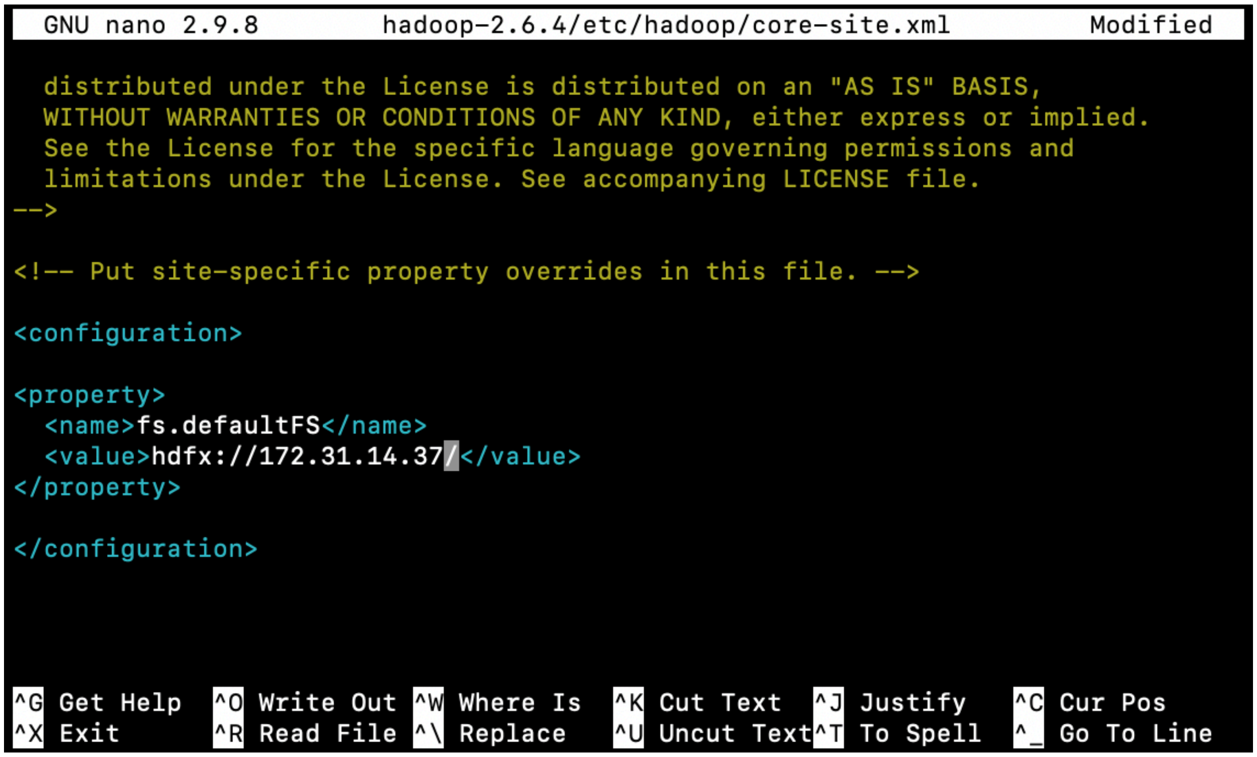
c) Repeat the previous task (3-b), but shut down one of the nodes (from Amazon console, imitating a failure) **while the task** is running. Record the time it took. How does it compare to the previous execution?

```
time hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/ssbm -output
/data/output3 -mapper myMapper.0.3.py -reducer myReducer.0.23.py -file
myReducer.0.3.py -file myMapper.0.3.py

19/06/09 11:31:50 INFO streaming.StreamJob: Output directory: /data/output3

real    1m31.321s
user    0m3.995s
sys     0m0.343s
```

The execution time was slower. The loss of a node results in longer time due to fewer mappers and reducers to divide the work as well as recovery time to restart any mapper tasks on the lost node.

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.

Lavinia Wang 1473704



```
  GNU nano 2.9.8        hadoop-2.6.4/etc/hadoop/core-site.xml        Modified

  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
  <name>fs.defaultFS</name>
  <value>hdfx://172.31.14.37/</value>
</property>

</configuration>



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Main error message below in **bold**.

```
[ec2-user@ip-172-31-14-37 ~]$ start-dfs.sh
Incorrect configuration: namenode address dfs.namenode.servicerpc-address or
dfs.namenode.rpc-address is not configured.
Starting namenodes on []
172.31.15.69: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-namenode-ip-172-31-15-69.us-east-2.compute.internal.out
172.31.4.33: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-namenode-ip-172-31-4-33.us-east-2.compute.internal.out
172.31.6.149: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-namenode-ip-172-31-6-149.us-east-2.compute.internal.out
172.31.14.37: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-namenode-ip-172-31-14-37.us-east-2.compute.internal.out
172.31.2.19: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-namenode-ip-172-31-2-19.us-east-2.compute.internal.out
172.31.14.37: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-datanode-ip-172-31-14-37.us-east-2.compute.internal.out
172.31.6.149: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-datanode-ip-172-31-6-149.us-east-2.compute.internal.out
172.31.15.69: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-datanode-ip-172-31-15-69.us-east-2.compute.internal.out
172.31.2.19: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-datanode-ip-172-31-2-19.us-east-2.compute.internal.out
172.31.4.33: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-
ec2-user-datanode-ip-172-31-4-33.us-east-2.compute.internal.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ec2-user/hadoop-
2.6.4/logs/hadoop-ec2-user-secondarynamenode-ip-172-31-14-37.us-east-
2.compute.internal.out
```

```
0.0.0.0: Exception in thread "main" java.lang.IllegalArgumentException: Invalid URI
for NameNode address (check fs.defaultFS): hdfx://172.31.14.37/ is not of scheme
'hdfs'.
0.0.0.0:        at
org.apache.hadoop.hdfs.server.namenode.NameNode.getAddress(NameNode.java:431)
0.0.0.0:        at
org.apache.hadoop.hdfs.server.namenode.NameNode.getAddress(NameNode.java:415)
0.0.0.0:        at
org.apache.hadoop.hdfs.server.namenode.NameNode.getServiceAddress(NameNode.java:408)
0.0.0.0:        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.initialize(SecondaryNameNode
.java:229)
0.0.0.0:        at
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.<init>(SecondaryNameNode.jav
a:192)
       0.0.0.0:        at
       org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.main(SecondaryNameNod
       e.java:671)
```

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)

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

Verify and report 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?

-rw-r--r--   2 ec2-user supergroup   **11553456** 2019-06-09 13:18 /user/ec2-user/movielens/ratings.csv

-rw-r--r--   2 ec2-user supergroup   **10397117** 2019-06-09 13:54 /user/ec2-user/ml_dataset/trainingSet/part-m-00000

-rw-r--r--   2 ec2-user supergroup   **1156339** 2019-06-09 13:54 /user/ec2-user/ml_dataset/probeSet/part-m-00000

10397117 + 1156339 = 11553456

Yes, they do.

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

```
real    3m50.248s
user    0m12.721s
sys     0m3.329s
```

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

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

[ec2-user@ip-172-31-29-137 apache-mahout-distribution-0.11.2]$ hadoop fs -cat /user/ec2-user/als/rmse/rmse.txt

Lavinia Wang 1473704

0.8834598335250934

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

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.

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

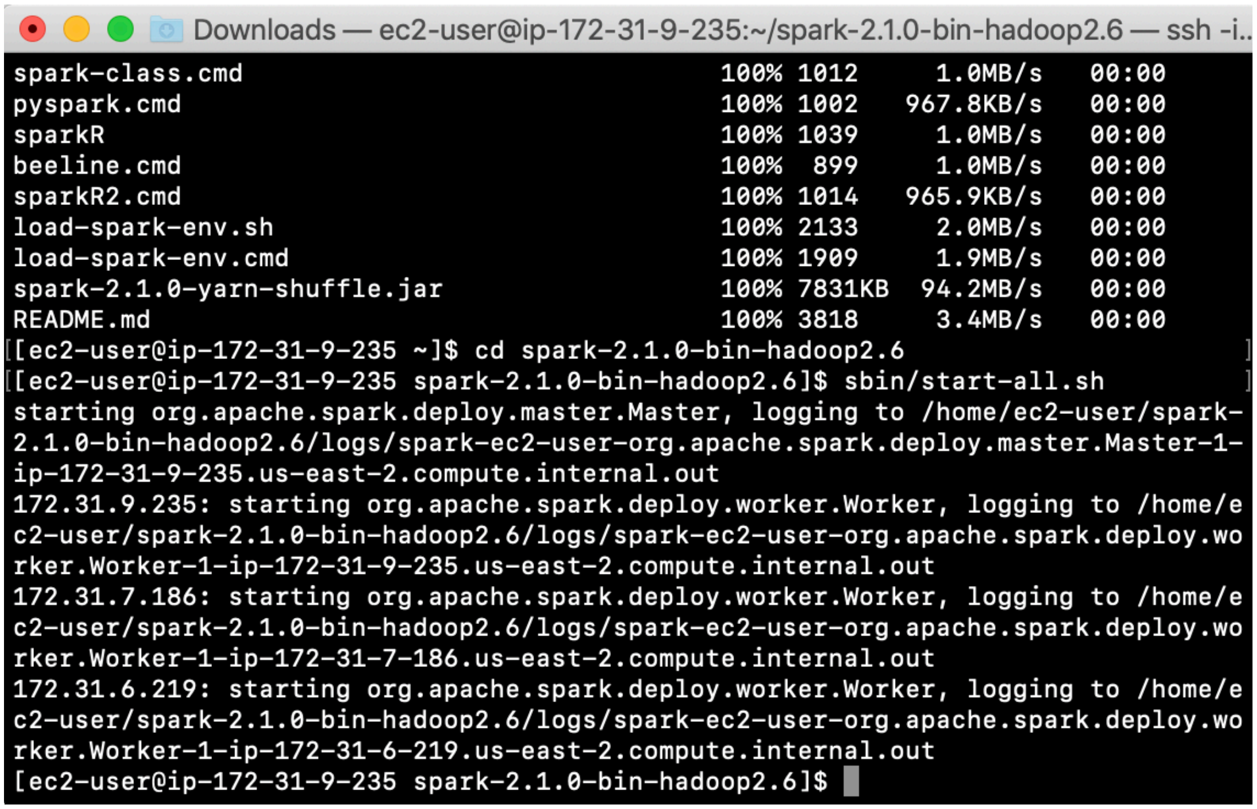
```
[ec2-user@ip-172-31-29-137 apache-mahout-distribution-0.11.2]$ $HADOOP_HOME/bin/hadoop fs -cat
recommendations/part-m-00000 | head
1       [572:5.0,2197:4.6908126,3314:4.687907,2156:4.4995503,356:4.382499,3222:4.3752384]
2       [572:4.8178015,2197:4.649721,527:4.455281,1721:4.318738,2762:4.2395806,2324:4.1847873]
3       [572:4.753747,2913:4.745884,318:4.6085696,2762:4.569885,110:4.56089,3443:4.523036]
4       [3245:5.0,3092:5.0,923:5.0,1423:5.0,2905:5.0,1212:5.0]
5
[1423:4.7975807,3245:4.595153,668:4.564381,1002:4.4648323,3570:4.428102,3645:4.3494987]
6       [572:5.0,3314:5.0,2197:4.848169,3675:4.432844,3916:4.426566,3161:4.3900433]
7
[1036:4.686574,1240:4.6430492,2494:4.614736,1198:4.608802,3508:4.5733657,3552:4.5691805]
8       [858:4.798158,2905:4.676086,318:4.6493897,1198:4.646571,50:4.6085334,3038:4.587847]
9       [50:4.3938804,858:4.342568,296:4.325224,2329:4.26149,2905:4.2136726,110:4.213341]
10      [572:5.0,3314:4.8458138,919:4.4136,1907:4.4132323,3916:4.4002104,2609:4.396858]
```

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

User 3 = 572
User 4 = 3245
User 5 = 1423

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 spark folder. Browser page is at port 8080. 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

Lavinia Wang 1473704


Downloads — ec2-user@ip-172-31-9-235:~/spark-2.1.0-bin-hadoop2.6 — ssh -i..

```
spark-class.cmd                                     100% 1012     1.0MB/s   00:00
pyspark.cmd                                         100% 1002   967.8KB/s   00:00
sparkR                                              100% 1039     1.0MB/s   00:00
beeline.cmd                                         100%  899     1.0MB/s   00:00
sparkR2.cmd                                         100% 1014   965.9KB/s   00:00
load-spark-env.sh                                   100% 2133     2.0MB/s   00:00
load-spark-env.cmd                                  100% 1909     1.9MB/s   00:00
spark-2.1.0-yarn-shuffle.jar                        100% 7831KB  94.2MB/s   00:00
README.md                                           100% 3818     3.4MB/s   00:00
[ec2-user@ip-172-31-9-235 ~]$ cd spark-2.1.0-bin-hadoop2.6
[ec2-user@ip-172-31-9-235 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-172-31-9-235.us-east-2.compute.internal.out
172.31.9.235: 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-172-31-9-235.us-east-2.compute.internal.out
172.31.7.186: 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-172-31-7-186.us-east-2.compute.internal.out
172.31.6.219: 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-172-31-6-219.us-east-2.compute.internal.out
[ec2-user@ip-172-31-9-235 spark-2.1.0-bin-hadoop2.6]$
```

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