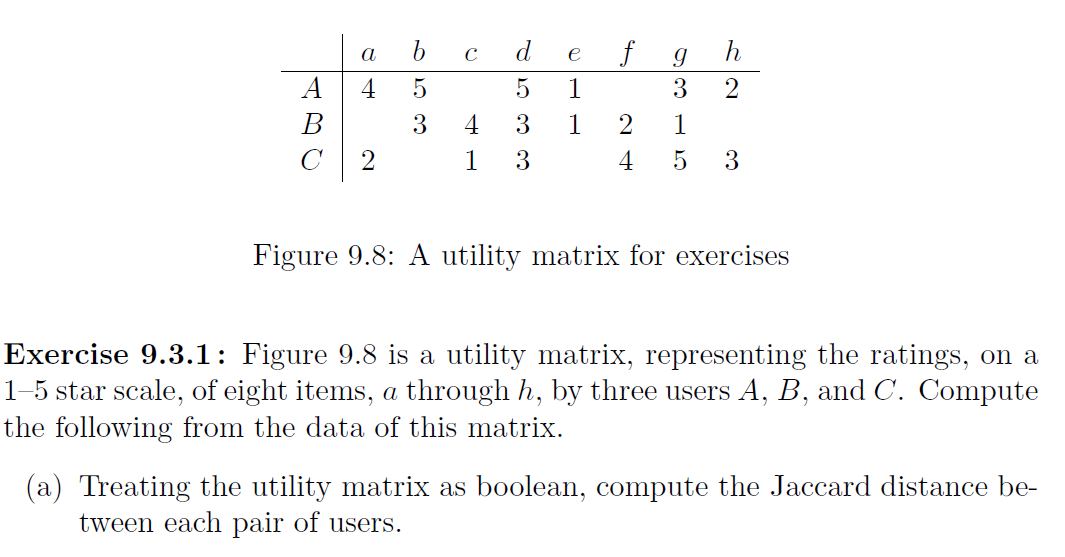
**Due Sunday, 6/9**

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

* 1. 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 | 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)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

* 1. 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.

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

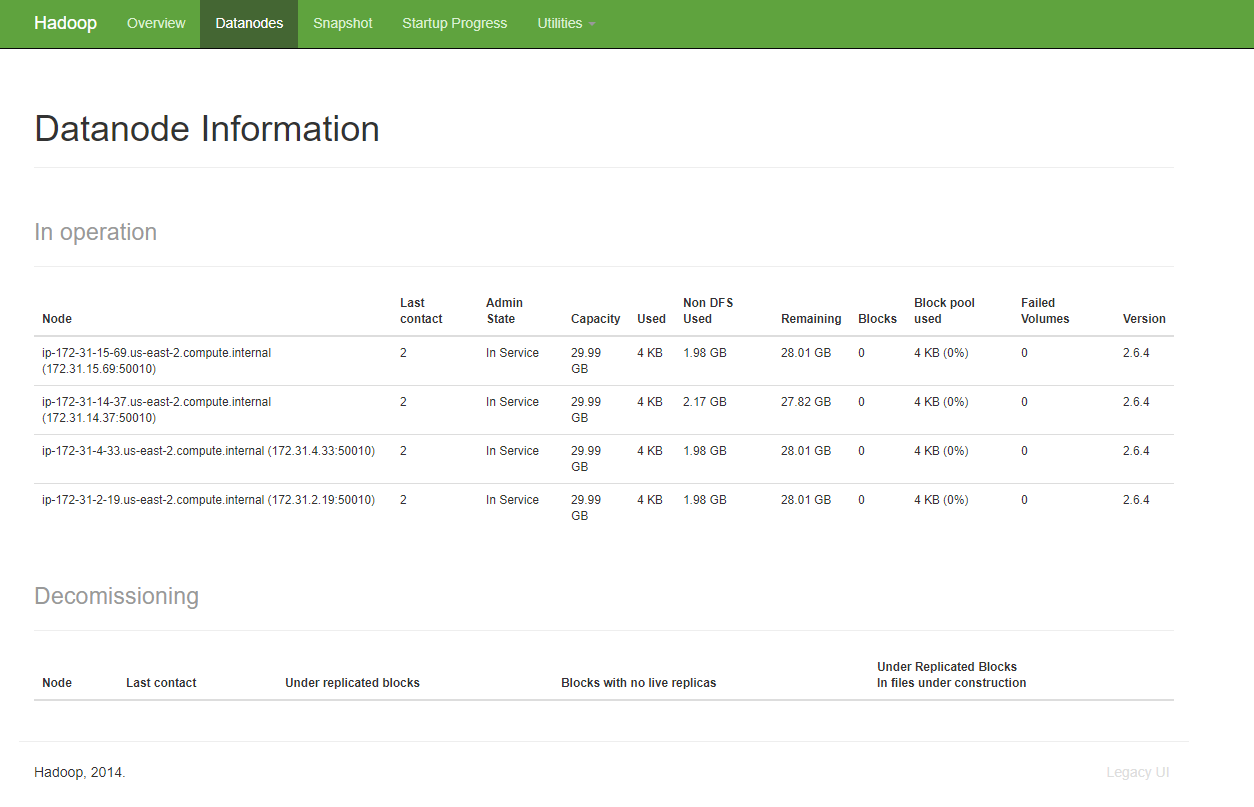
Spark typically reads data from HDFS into memory for processing but is otherwise stored in HDFS, thus preventing data loss during failure, provided sufficient replication is defined for HDFS.

* 1. 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.

* 1. 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.



* 1. 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

* 1. 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.

* 1. 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.



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.java:192)

0.0.0.0: at org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode.main(SecondaryNameNode.java:671)

1. 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

**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

1. 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>



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