CHIN-TING KO FALL 2016

APACHE MAHOUT

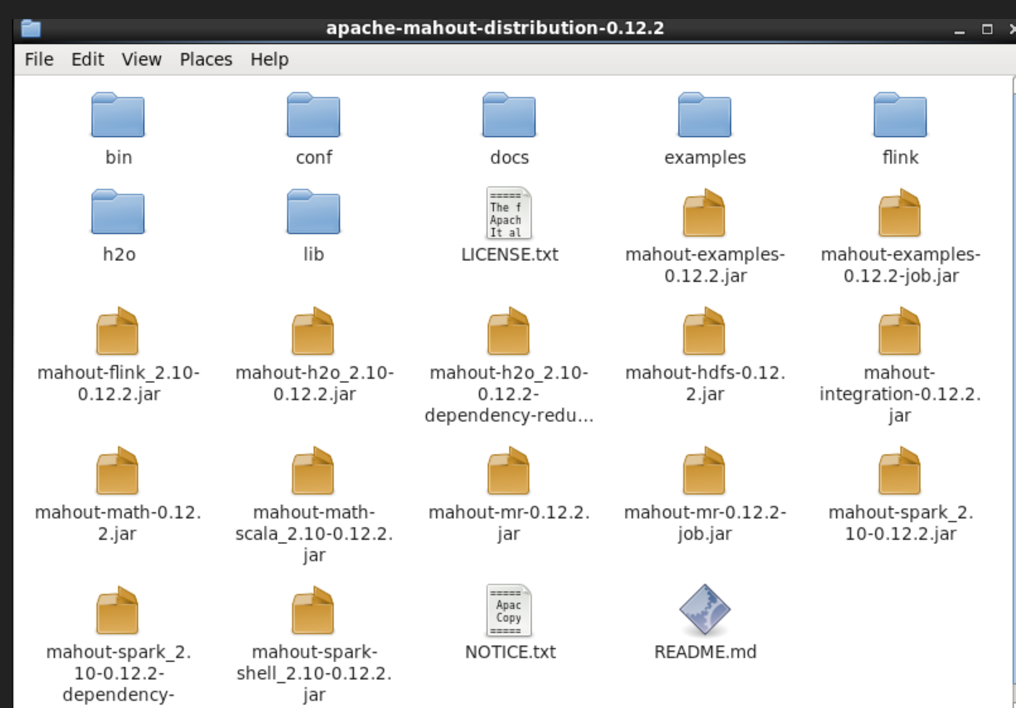
605.788 FINAL PROJECT

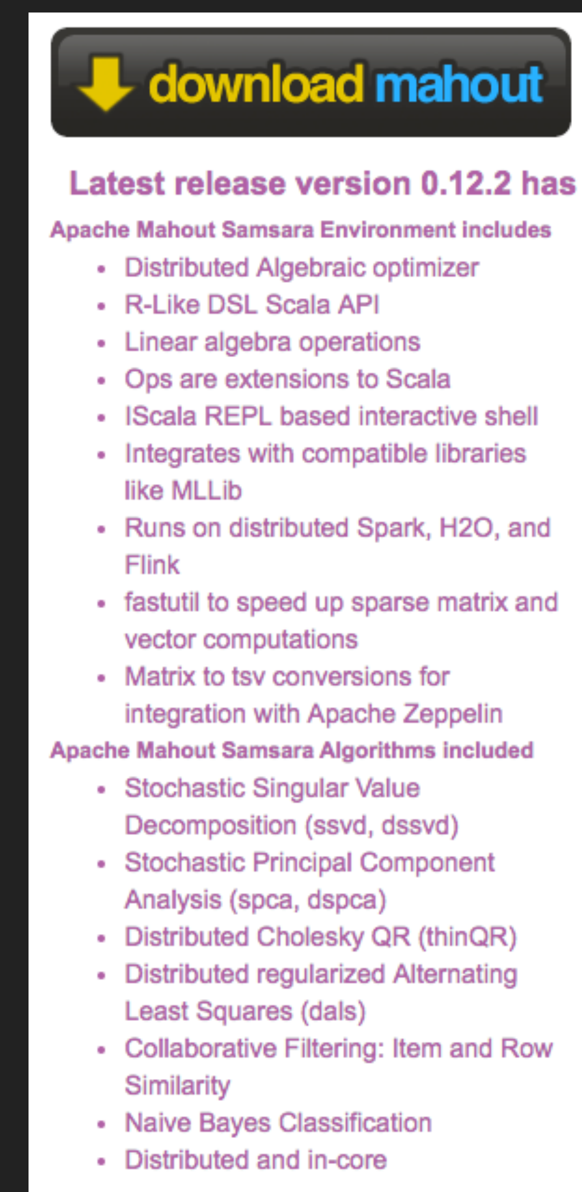
WHAT IS APACHE MAHOUT?

- Apache Mahout is a library of scalable machine learning algorithms on top of Hadoop and using MapReduce paradigm.
- Build an environment for quickly creating scalable performant machine learning applications
- Popular machine learning techniques such as
  - Recommendation
  - Classification
  - Clustering
- Once big data stored no the HDFS, Mahout makes it faster and easier to turn big data into big information

SETTING UP MAHOUT

- Install Java and IDE
- Install Hadoop
- Install Mahout (0.12.2)
- Edit .bash_profile to set up Mahout_HOME



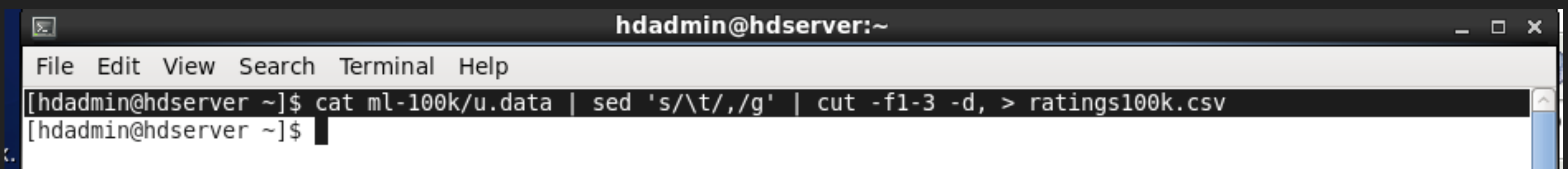


MOVIE RECOMMENDER USING MAHOUT

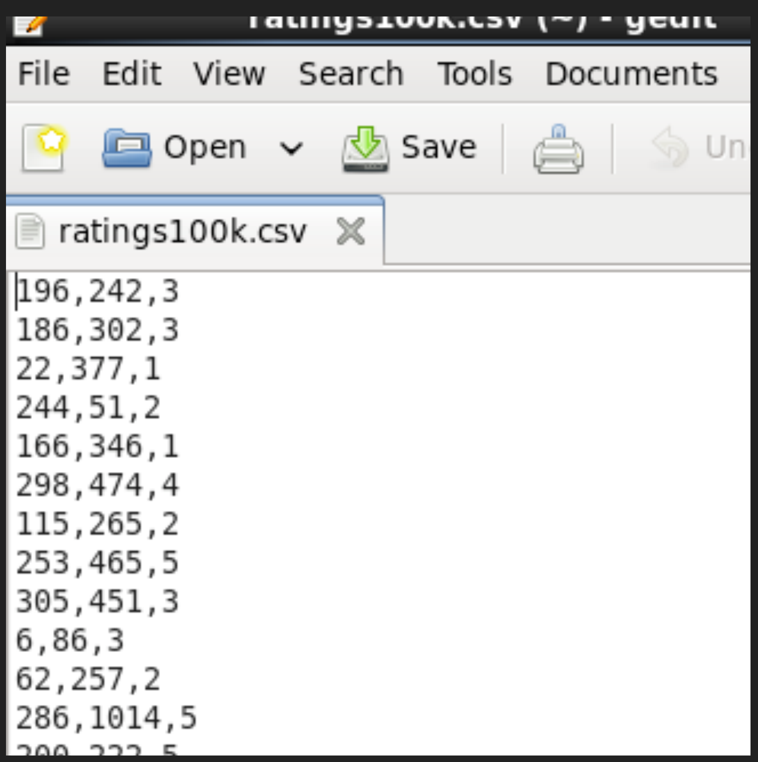
- Gather Input File (UserID, ItemID, Rating)
- Recommender Engine (Pick a similarity measure)
- Run Mahout Command
- Making Use of the Output File

INPUT DATASET

- http://grouplens.org/datasets/movielens/
- ml-100k.zip/u.data (UserID ItemID Rating Timestamp)



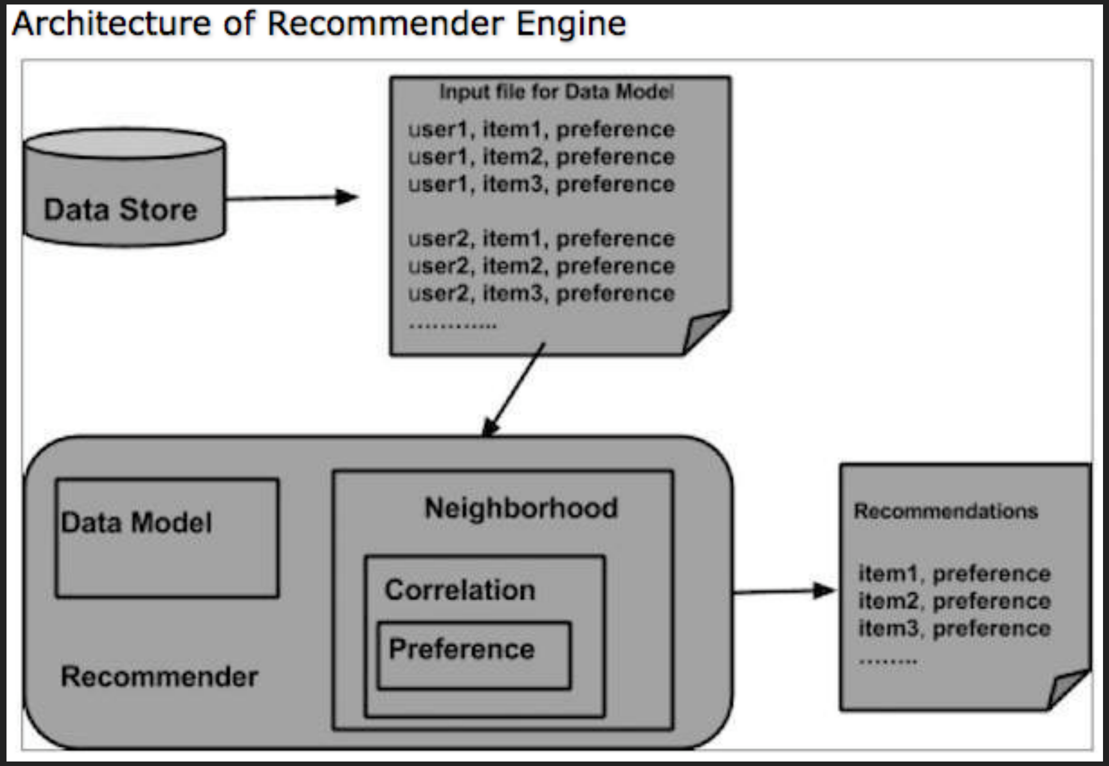
- ml-1m.zip/rating.dat (UserID::ItemID::Rating::Timestamp)
- UserID, ItemID, Rating



MAHOUT RECOMMENDER ENGINE

- Collaborative Filtering
  - Look for users who share same rating patterns
  - Use the ratings from those like-minded users found to calculate prediction

- Building a Recommender using Mahout
  - Create DataModel Object
  - Create UserSimilarity Object
  - Create Recommender Object
  - Recommend Items to User



RECOMMENDER ENGINE

- Copy File to HDFS:

```
hdadmin@hdserver:~
File  Edit  View  Search  Terminal  Help
-rw-r--r--   1 hdadmin supergroup     979173 2016-11-22 15:28 /ratings100k.csv
-rw-r--r--   1 hdadmin supergroup   11553456 2016-11-22 15:27 /ratings1m.csv
drwxrwx---   - hdadmin supergroup          0 2016-11-13 11:41 /tmp
drwxr-xr-x   - hdadmin supergroup          0 2016-11-06 11:12 /user
[hdadmin@hdserver ~]$ 
```

- Run Mahout Recommendation Job:

```
hdadmin@hdserver:~
File  Edit  View  Search  Terminal  Help
[hdadmin@hdserver ~]$ mahout recommenditembased --input /ratings100k.csv --output recommendations100k --numRecommendations 25 --s
imilarityClassname SIMILARITY_COSINE
Running on hadoop, using /usr/local/hadoop/hadoop-2.7.2/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /usr/local/hadoop/apache-mahout-distribution-0.12.2/mahout-examples-0.12.2-job.jar
16/11/22 15:49:54 INFO AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/ratings100
k.csv], --maxPrefsInItemSimilarity=[500], --maxPrefsPerUser=[10], --maxSimilaritiesPerItem=[100], --minPrefsPerUser=[1], --numRec
ommendations=[25], --output=[recommendations100k], --similarityClassname=[SIMILARITY_COSINE], --startPhase=[0], --tempDir=[temp]}
16/11/22 15:49:54 INFO AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/ratings100
k.csv], --minPrefsPerUser=[1], --output=[temp/preparePreferenceMatrix], --ratingShift=[0.0], --startPhase=[0], --tempDir=[temp]}
Java HotSpot(TM) Server VM warning: You have loaded library /usr/local/hadoop/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which mi
ght have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
16/11/22 15:49:55 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
16/11/22 15:49:55 INFO deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
16/11/22 15:49:55 INFO deprecation: mapred.compress.map.output is deprecated. Instead, use mapreduce.map.output.compress
16/11/22 15:49:55 INFO deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output.fileoutputformat.outputdir
16/11/22 15:49:55 INFO RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/11/22 15:49:58 INFO FileInputFormat: Total input paths to process : 1
16/11/22 15:49:58 INFO JobSubmitter: number of splits:1
16/11/22 15:49:59 INFO JobSubmitter: Submitting tokens for job: job_1479854934870_0002
16/11/22 15:49:59 INFO YarnClientImpl: Submitted application application_1479854934870_0002
16/11/22 15:49:59 INFO Job: The url to track the job: http://localhost:8088/proxy/application_1479854934870_0002/
16/11/22 15:49:59 INFO Job: Running job: job_1479854934870_0002
16/11/22 15:50:14 INFO Job: Job job_1479854934870_0002 running in uber mode : false
16/11/22 15:50:14 INFO Job:  map 0% reduce 0%
16/11/22 15:50:22 INFO Job:  map 100% reduce 0%
16/11/22 15:50:31 INFO Job:  map 100% reduce 100%
16/11/22 15:50:31 INFO Job: Job job_1479854934870_0002 completed successfully
16/11/22 15:50:31 INFO Job: Counters: 49
```

OUTPUT FILE

- Each line represents UserID with associated recommended ItemID and their scores.

```
hdadmin@hdserver:~
File  Edit  View  Search  Terminal  Help
1       [133:5.0,523:5.0,265:5.0,789:5.0,1578:5.0,655:5.0,528:5.0,660:5.0,526:5.0,530:5.0,260:5.0,663:5.0,533:5.0,531:5.0
2       [196:5.0,293:5.0,265:5.0,134:5.0,129:5.0,294:5.0,297:5.0,197:5.0,990:5.0,200:5.0,300:5.0,135:5.0,887:5.0,298:5.0,
3       [285:5.0,284:5.0,124:5.0,47:5.0,129:5.0,654:5.0,137:5.0,248:5.0,693:5.0,508:4.814447,276:4.7747316,150:4.7706466,
262:4.700115,475:4.6875877,243:4.6727667,23:4.669365,1143:4.6647367]
4       [1191:5.0,262:5.0,989:5.0,70:5.0,64:5.0,995:5.0,263:5.0,990:5.0,196:5.0,268:5.0,325:5.0,195:5.0,270:5.0,323:5.0,2
5       [196:5.0,131:5.0,265:5.0,200:5.0,199:5.0,129:5.0,792:5.0,67:5.0,529:5.0,134:5.0,727:5.0,201:5.0,133:5.0,135:5.0,1
6       [530:5.0,663:5.0,660:5.0,196:5.0,463:5.0,1126:5.0,396:5.0,659:5.0,529:5.0,200:5.0,129:5.0,461:5.0,268:5.0,70:5.0,
7       [392:5.0,270:5.0,395:5.0,400:5.0,531:5.0,1448:5.0,396:5.0,1188:5.0,789:5.0,268:5.0,137:5.0,1454:5.0,523:5.0,525:5
8       [200:5.0,298:5.0,197:5.0,134:5.0,199:5.0,300:5.0,67:5.0,232:5.0,526:5.0,231:5.0,559:5.0,135:5.0,31:5.0,133:5.0,99
9       [692:5.0,133:5.0,265:5.0,229:5.0,135:5.0,724:5.0,528:5.0,100:5.0,65:5.0,231:5.0,196:5.0,559:5.0,31:5.0,199:5.0,33
10      [724:5.0,663:5.0,265:5.0,727:5.0,461:5.0,1126:5.0,528:5.0,660:5.0,527:5.0,530:5.0,131:5.0,462:5.0,463:5.0,70:5.0,
cat: Unable to write to output stream.
[hdadmin@hdserver ~]$
```

SPARK – MLLIB

- MLlib is Apache Spark's scalable machine learning library.
- MLlib was built on top of Spark to take advantage of Spark's efficiency when running iterative Machine Learning algorithms.
- MLlib uses the alternating least squares (ALS) algorithm to learn these latent factors.
    - Alternating Least Squares (ALS) is an optimization technique to solve matrix factorization problems.
    - ALS works by iteratively solving a series of least squares regression problems. In each iteration, one of the user- or item-factor matrices is treated as fixed, while the other one is updated using the fixed factor and the rating data. Then, the factor matrix that was solved for is, in turn, treated as fixed, while the other one is updated. This process continues until the model has converged (or for a fixed number of iterations).

(https://www.packtpub.com/books/content/building-recommendation-engine-spark)

OUTPUT FILE (SPARK MLLIB)

- Each line represents UserID with associated recommended ItemID and their scores.

hdadmin@hdserver:~/MovieRecommendation

File Edit View Search Terminal Help

```
scala> val topRecsForUser1 = model.recommendProducts(1,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(1,618,6.774438867943637), Rating(1,3867,6.6489927917
52741), Rating(1,966,6.522484173964295), Rating(1,572,6.427094942354792), Rating(1,3149,6.347160717816235), Rating(1,3523,6.245779377862
082), Rating(1,2962,6.0278801095579695), Rating(1,2482,5.900282961200457), Rating(1,1685,5.892318867776706), Rating(1,3933,5.75546569128
9622))

scala> val topRecsForUser1 = model.recommendProducts(2,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(2,1872,6.832084684845792), Rating(2,864,6.7293577321
87241), Rating(2,572,6.13254322790024), Rating(2,1058,5.730026263746217), Rating(2,2503,5.370907329267695), Rating(2,1574,5.347824820420
602), Rating(2,1436,5.295925227379191), Rating(2,687,5.278860715744662), Rating(2,3670,5.2319661479476185), Rating(2,3245,5.149588571716
4285))

scala> val topRecsForUser1 = model.recommendProducts(3,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(3,811,11.1624144104375), Rating(3,751,11.05831602680
2977), Rating(3,197,10.829263737951194), Rating(3,2913,9.877369870311949), Rating(3,1549,9.303079940465825), Rating(3,974,9.234115368084
6), Rating(3,939,9.163269248141692), Rating(3,1529,9.006818203338263), Rating(3,1898,8.998920132163294), Rating(3,632,8.959103487192792)
)

scala> val topRecsForUser1 = model.recommendProducts(4,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(4,1000,10.47886024462249), Rating(4,2342,9.929486630
314619), Rating(4,3245,9.022109081705754), Rating(4,2238,8.827577460318551), Rating(4,831,8.811759641124747), Rating(4,3944,8.6624112673
7358), Rating(4,2192,8.617173358848197), Rating(4,2773,8.514338413500267), Rating(4,3951,8.16717151759865), Rating(4,3224,8.162175315542
498))

scala> val topRecsForUser1 = model.recommendProducts(5,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(5,503,5.788375141269848), Rating(5,1002,5.7112524913
53883), Rating(5,3645,5.695299660152859), Rating(5,3776,5.586769089511085), Rating(5,1144,5.414294176773753), Rating(5,1423,5.3554156874
888434), Rating(5,1664,5.330862239434492), Rating(5,702,5.282304519114816), Rating(5,557,5.235724554946875), Rating(5,1153,5.23539494641
8802))

scala> val topRecsForUser1 = model.recommendProducts(6,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(6,2933,7.906932108263031), Rating(6,1872,7.148926771
2952795), Rating(6,2129,7.053008226427211), Rating(6,2056,7.035079732165646), Rating(6,1471,6.632216980420657), Rating(6,638,6.476366184
663778), Rating(6,2913,6.351026973095998), Rating(6,687,6.296795083953017), Rating(6,2332,6.244670194409402), Rating(6,2503,6.2298639191
964815))

scala> val topRecsForUser1 = model.recommendProducts(7,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(7,2964,9.710648821264051), Rating(7,1471,7.819228601
9784625), Rating(7,2063,7.78771032750209), Rating(7,2129,7.746815693528694), Rating(7,958,7.717085278713771), Rating(7,618,7.57059447208
84055), Rating(7,3636,7.547774995633008), Rating(7,811,7.417765929041326), Rating(7,2933,7.3167072524981505), Rating(7,1232,7.2766931329
87369))

scala> val topRecsForUser1 = model.recommendProducts(8,10)
topRecsForUser1: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(8,1685,6.436333469169517), Rating(8,108,5.9160031231
82164), Rating(8,3803,5.6484589181330325), Rating(8,557,5.60014498876392), Rating(8,572,5.5432001728285965), Rating(8,2309,5.42043940559
1808), Rating(8,296,5.138825191285687), Rating(8,2964,5.133273427686358), Rating(8,966,5.115977196882945), Rating(8,925,5.11470551297196
4))
```

SUMMARY/ OBSERVATION

- Mahout was build to run on top of Hadoop. Hadoop writes to disk on every iteration, this makes Mahout run very slowly.
- Spark also has a machine learning library (MLlib) to address this issue.(processing using distributed memory)
- Mahout is moving away from MapReduce
    - 11 April 2015 - Apache Mahout 0.10.0 released
    - The Hadoop MapReduce versions of Mahout algorithms are still maintained but no new MapReduce contributions are accepted. From this release onwards contributions must be Mahout Samsara based or at least run on Spark.

SUMMARY/ OBSERVATION

- ml-100k: 5 minutes (Mahout)

```
hdadmin@hdserver:~
File  Edit  View  Search  Terminal  Help
16/11/22 16:03:21 INFO MahoutDriver: Program took 303190 ms (Minutes: 5.053166666666667)
[hdadmin@hdserver ~]$
```

- ml-1m: 8.5 minutes (Mahout)

```
hdadmin@hdserver:~
File  Edit  View  Search  Terminal  Help
        File Input Format Counters
                Bytes Read=8298284
        File Output Format Counters
                Bytes Written=1447337
16/11/22 16:13:59 INFO MahoutDriver: Program took 512026 ms (Minutes: 8.533766666666667)
[hdadmin@hdserver ~]$
```

- ml-1m : less than 1 min (Spark MLlib)
  - train the model: less than 30 second
  - query: 1~2 second

REFERENCE

http://mahout.apache.org/
https://www.manning.com/books/mahout-in-action
http://www.tutorialspoint.com/mahout/
https://en.wikipedia.org/wiki/Apache_Mahout
https://www.youtube.com/watch?v=bXtX6lPoBME
https://www.youtube.com/watch?v=iMAMYzfRiS4
http://leoyeh.me:8080/2014/12/20/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-Mahout-%E8%99%95%E7%90%86-1/
http://blog.csdn.net/huhui_cs/article/details/8596388
http://blog.cloudera.com/blog/2011/11/recommendation-with-apache-mahout-in-cdh3/
https://www.quora.com/When-should-one-use-Sparks-MLlib-and-not-Apache-Mahout
https://databricks-training.s3.amazonaws.com/movie-recommendation-with-mllib.html
http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html
https://www.packtpub.com/books/content/building-recommendation-engine-spark