Recommendation System

--Using Spark Platform

Group Member:
Group 5
Xie, Yiding
Xing, Yang
Zhou, Zhibo

Outline

Data

Eco-system

Alternating Least Squares

Performance

Data| PART1

Data Source:

https://movielens.org/


```
movie                    58098 obs. of 3 variables
  movieId: int 1 2 3 4 5 6 7 8 9 10 ...
  title : chr "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to E…
  genres : chr "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy"…
rating                   27753444 obs. of 4 variables
  userId : int 1 1 1 1 1 1 1 1 1 1 ...
  movieId : int 307 481 1091 1257 1449 1590 1591 2134 2478 2840 ...
  rating : num 3.5 3.5 1.5 4.5 4.5 2.5 1.5 4.5 4 3 ...
  timestamp: int 1256677221 1256677456 1256677471 1256677460 1256677264 1256677236 125667…
```

Eco-System| PART2

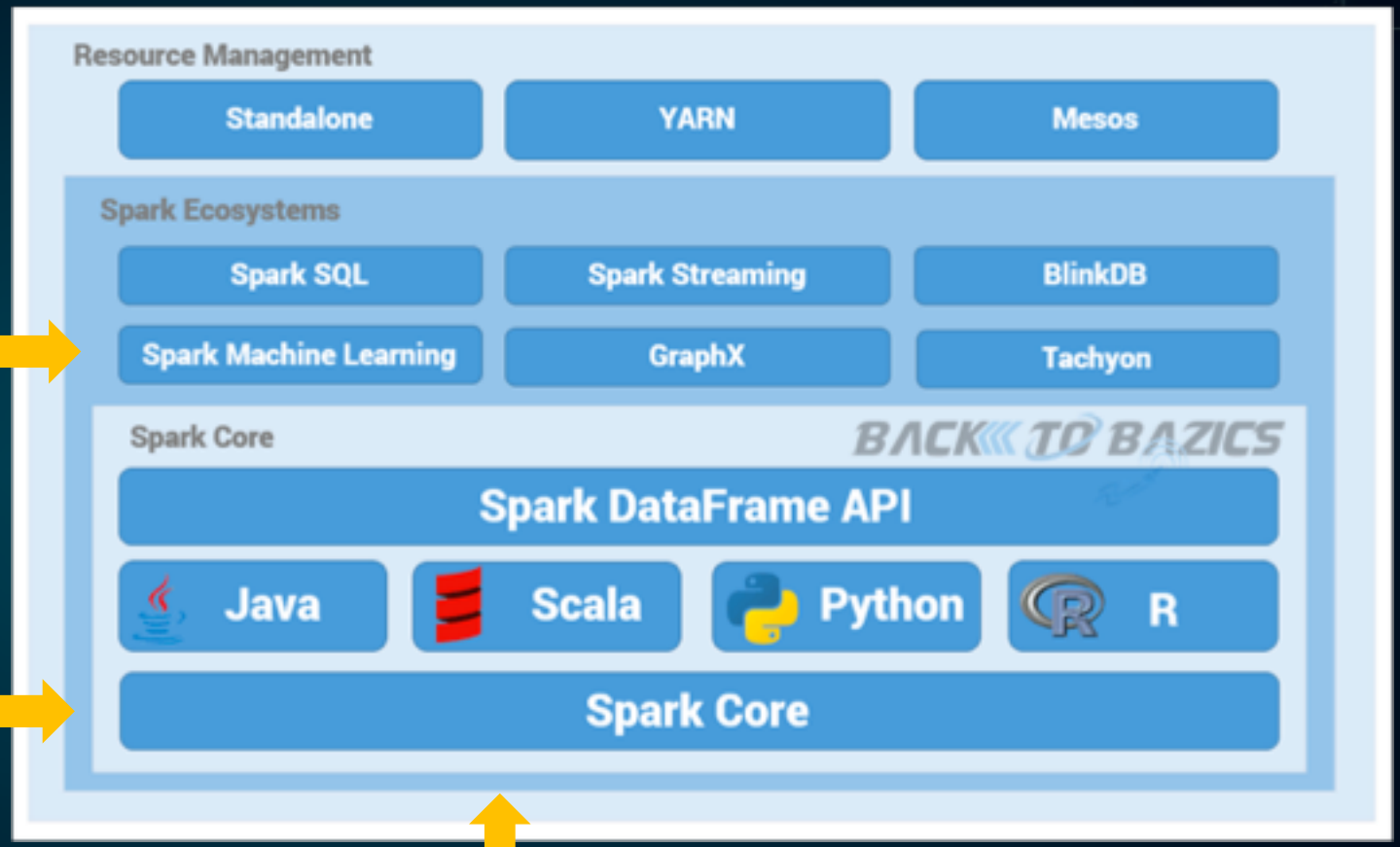
Spark Ecosystem

Machine Learning: MLlib

Machine learning has quickly emerged as a critical piece in mining Big Data for actionable insights. Built on top of Spark, MLlib is a scalable machine learning library that delivers both high-quality algorithms (e.g., multiple iterations to increase accuracy) and blazing speed (up to 100x faster than MapReduce). The library is usable in Java, Scala, and Python as part of Spark applications, so that you can include it in complete workflows.

General Execution: Spark Core

Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built on top of. It provides in-memory computing capabilities to deliver speed, a generalized execution model to support a wide variety of applications, and Java, Scala, and Python APIs for ease of development.

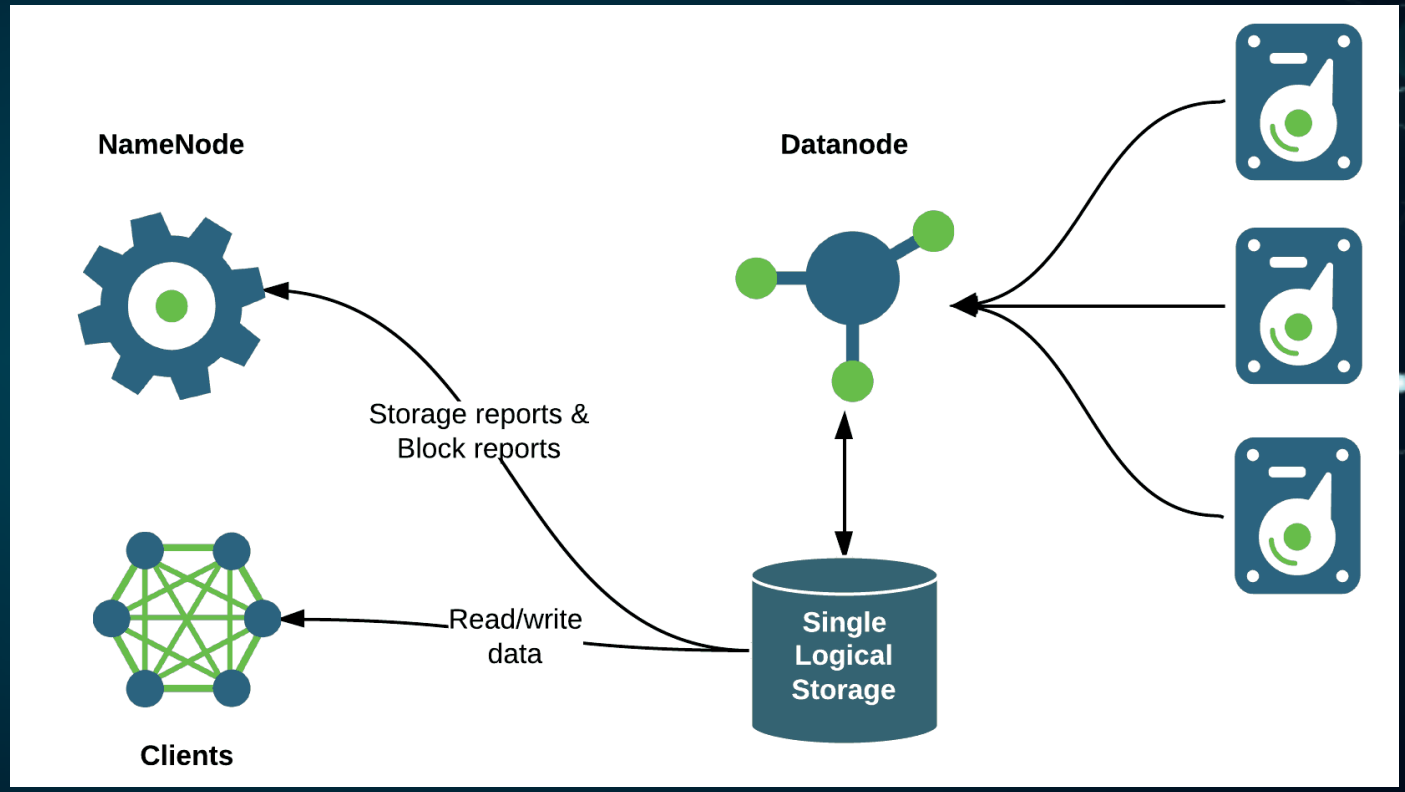


APACHE SPARK

- Apache Spark is a framework for real time data analytics in a distributed computing environment.
- The Spark is written in Scala and was originally developed at the University of California, Berkeley.
- It executes in-memory computations to increase speed of data processing over Map-Reduce.

Hadoop Distributed File System

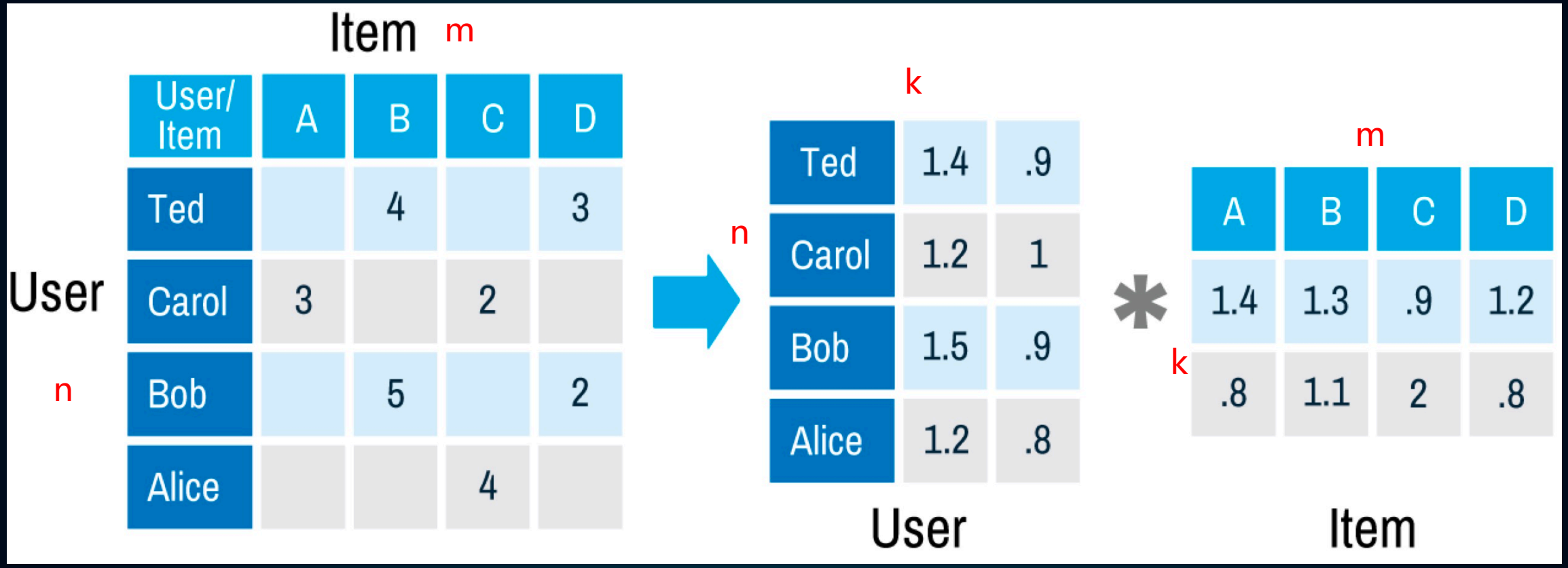
1. The **NameNode** is the main node, it doesn't store the actual data.
2. **DataNodes** is where data stored, hence it requires more storage resources.
3. Clients always communicate to the **NameNode** while writing the data.



```
[hadoop@ip-172-31-94-195 datasets]$ hadoop fs -ls /user
Found 8 items
drwxrwxrwx   - hadoop hadoop          0 2018-12-04 02:30 /user/hadoop
drwxr-xr-x   - mapred mapred          0 2018-12-04 02:30 /user/history
drwxrwxrwx   - hdfs   hadoop          0 2018-12-04 02:30 /user/hive
drwxrwxrwx   - hue    hue             0 2018-12-04 02:30 /user/hue
drwxrwxrwx   - livy   livy            0 2018-12-04 03:10 /user/livy
drwxrwxrwx   - oozie  oozie           0 2018-12-04 02:30 /user/oozie
drwxrwxrwx   - root   hadoop          0 2018-12-04 02:30 /user/root
drwxrwxrwx   - spark  spark           0 2018-12-04 02:30 /user/spark
[hadoop@ip-172-31-94-195 datasets]$ hadoop fs -ls /user/livy
Found 1 items
drwxr-xr-x   - livy livy          0 2018-12-04 03:10 /user/livy/.sparkStaging
[hadoop@ip-172-31-94-195 datasets]$ ^C
[hadoop@ip-172-31-94-195 datasets]$ hadoop fs -put ~/home/hadoop/datasets /user/
livy
put: `/home/hadoop/home/hadoop/datasets': No such file or directory
[hadoop@ip-172-31-94-195 datasets]$ cd
[hadoop@ip-172-31-94-195 ~]$ hadoop fs -put /home/hadoop/datasets /user/livy
[hadoop@ip-172-31-94-195 ~]$ hadoop fs -ls /user/livy
Found 2 items
drwxr-xr-x   - livy   livy          0 2018-12-04 03:10 /user/livy/.sparkStaging
drwxr-xr-x   - hadoop livy          0 2018-12-04 03:46 /user/livy/datasets
[hadoop@ip-172-31-94-195 ~]$ hadoop fs -ls /user/livy/datasets
Found 2 items
drwxr-xr-x   - hadoop livy          0 2018-12-04 03:46 /user/livy/datasets/ml-la
test
-rw-r--r--   1 hadoop livy  277113433 2018-12-04 03:46 /user/livy/datasets/ml-la
test.zip
[hadoop@ip-172-31-94-195 ~]$
Broadcast message from root@ip-172-31-94-195
        (unknown) at 4:32 ...
```

Alternating Least Squares | PART 3

Main Idea of ALS



Number of user (n): 27,753,444
Number of item (m): 58,098

General Idea

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix}, Y = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_m \\ | & & | \end{bmatrix}$$

If there are n users and m items, we are given an n × m matrix R in which the (u, i) th entry is r_{ui} – the rating for item i by user

$$r_{ui} \approx x_u^\intercal y_i$$

$$\min_{X,Y} \sum_{r_{ui}\, observed} (r_{ui} - x_u^\intercal y_i)^2 + \lambda(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

Algorithm

Initialize X, Y

repeat

for $u = 1 \dots n$ **do**

$$x_u = \left(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i$$

end for

for $i = 1 \dots m$ **do**

$$y_i = \left(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u$$

end for

until convergence

Implementation
(From Pyspark.MlLib)

```
model = ALS.train(training_RDD, best_rank, seed=seed, iterations=iterations,
                  lambda_=regularization_parameter)
predictions = model.predictAll(test_for_predict_RDD).map(lambda r: ((r[0], r[1]), r[2]))
rates_and_preds = test_RDD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())

print 'For testing data the RMSE is %s' % (error)
```

```
For testing data the RMSE is 0.818791604071
```

```
new_user_ID = 0

# The format of each line is (userID, movieID, rating)
new_user_ratings = [
     (0,283,2), # New Jersey Drive (1995)
     (0,1,3), # Toy Story (1995)
     (0,16,3), # Casino (1995)
     (0,25,4), # Leaving Las Vegas (1995)
     (0,32,4), # Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
     (0,335,1), # Flintstones, The (1994)
     (0,379,1), # Timecop (1994)
     (0,296,3), # Pulp Fiction (1994)
     (0,858,5) , # Godfather, The (1972)
     (0,50,4) # Usual Suspects, The (1995)
    ]
new_user_ratings_RDD = sc.parallelize(new_user_ratings)
print 'New user ratings: %s' % new_user_ratings_RDD.take(10)
```

```
from time import time

t0 = time()
new_ratings_model = ALS.train(complete_data_with_new_ratings_RDD, best_rank, seed=seed,
                              iterations=iterations, lambda_=regularization_parameter)
tt = time() - t0

print "New model trained in %s seconds" % round(tt,3)
```

```
New model trained in 97.435 seconds
```

Recommendation

```
TOP recommended movies (with more than 25 reviews):
(u'Slaying the Badger', 3.966705157439656, 25)
(u'Magic & Bird: A Courtship of Rivals (2010)', 3.9580754817432635, 27)
(u'Mikra Anglia (2013)', 3.9410029556849295, 27)
(u'"Godfather', 3.923830960619915, 60904)
(u"Long Night's Journey Into Day (2000)", 3.886173660237716, 35)
(u'"Godfather: Part II', 3.8550242120448273, 38875)
(u'Cosmos', 3.744092866081248, 157)
(u'"Two Escobars', 3.743587541374063, 78)
(u'Pulp Fiction (1994)', 3.731949117108328, 92406)
(u'Small Potatoes - Who Killed the USFL? (2009)', 3.7110161774156656, 26)
(u'Voices from the List (2004)', 3.701612067494449, 1800)
(u'The Godfather Trilogy: 1972-1990 (1992)', 3.696807783857102, 421)
(u'Roots (2016)', 3.6779815597313883, 28)
(u'Frozen Planet (2011)', 3.667588669754866, 402)
(u"Schindler's List (1993)", 3.66587088193783, 71516)
(u'"Shawshank Redemption', 3.6654771227714313, 97999)
(u'Heimat - A Chronicle of Germany (Heimat - Eine deutsche Chronik) (1984)', 3.657693967084001, 35)
(u'"Woman In Berlin', 3.651183111151667, 39)
(u'Death on the Staircase (Soup\xe7ons) (2004)', 3.646502406350472, 130)
(u'"Civil War', 3.6396583906771833, 431)
(u'History of the Eagles (2013)', 3.6374157520057833, 32)
(u'"Usual Suspects', 3.6353825929712174, 62180)
(u'Band of Brothers (2001)', 3.633756961432, 984)
(u'"Not on Your Life (Verdugo', 3.6331826250591366, 28)
(u"One Flew Over the Cuckoo's Nest (1975)", 3.619225585797985, 42181)
```

Performance| PART4

Performance

The Netflix Prize:

- Given the training data
  - 100 million ratings
- Rating Scale: 1-5

- Netflix's system
  - RMSE: 0.9514

Our model:

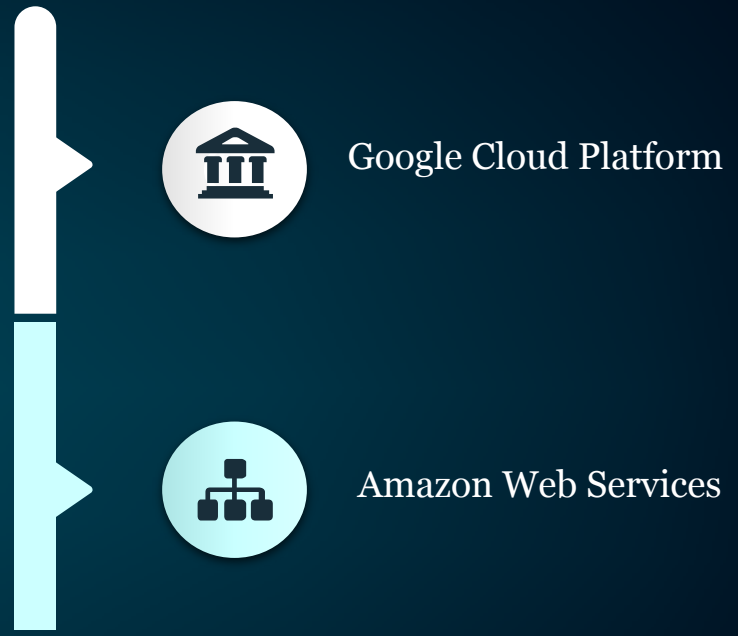
- Given the training data
  - Number of user: 27,753,444
  - Number of item: 58,098
- Rating Scale: 1-5

- RMSE: 0.8188

Journey

– Google Cloud Platform(GCP)
– Single Machine
– 16 Core, 96G RAM
– ALS algorithm, out of memory

– Amazon Web Services(AWS) EMR
– One master and three slaves, each machine has 16 core, 96G RAM
– Total Running Time : 97s



Conclusion

1. Taking advantage of cloud computing
2. Model has the capability of adding new user/rating and re-train in a very short period of time
3. Going forward, with more computational power, we can further reduce the computing time and hopefully close to real time

Reference

- http://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf
- https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/
- https://www.edureka.co/blog/hadoop-ecosystem

THANK YOU