

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP LỚN
LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU LỚN
ĐỀ TÀI: HỆ THỐNG THU THẬP, LƯU TRỮ, XỬ LÝ, PHÂN
TÍCH DỮ LIỆU PHIM IMDB VÀ XÂY DỰNG MÔ HÌNH GỢI Ý

Nhóm sinh viên thực hiện: Squad Game

Lại Ngọc Thăng Long	20183581
Nguyễn Đình Dũng	20183506
Nguyễn Thành Long	20183586
Nguyễn Khương Duy	20183513

Giảng viên hướng dẫn: **TS. Đào Thành Chung**

Hà Nội, tháng 12 năm 2021

Mục lục

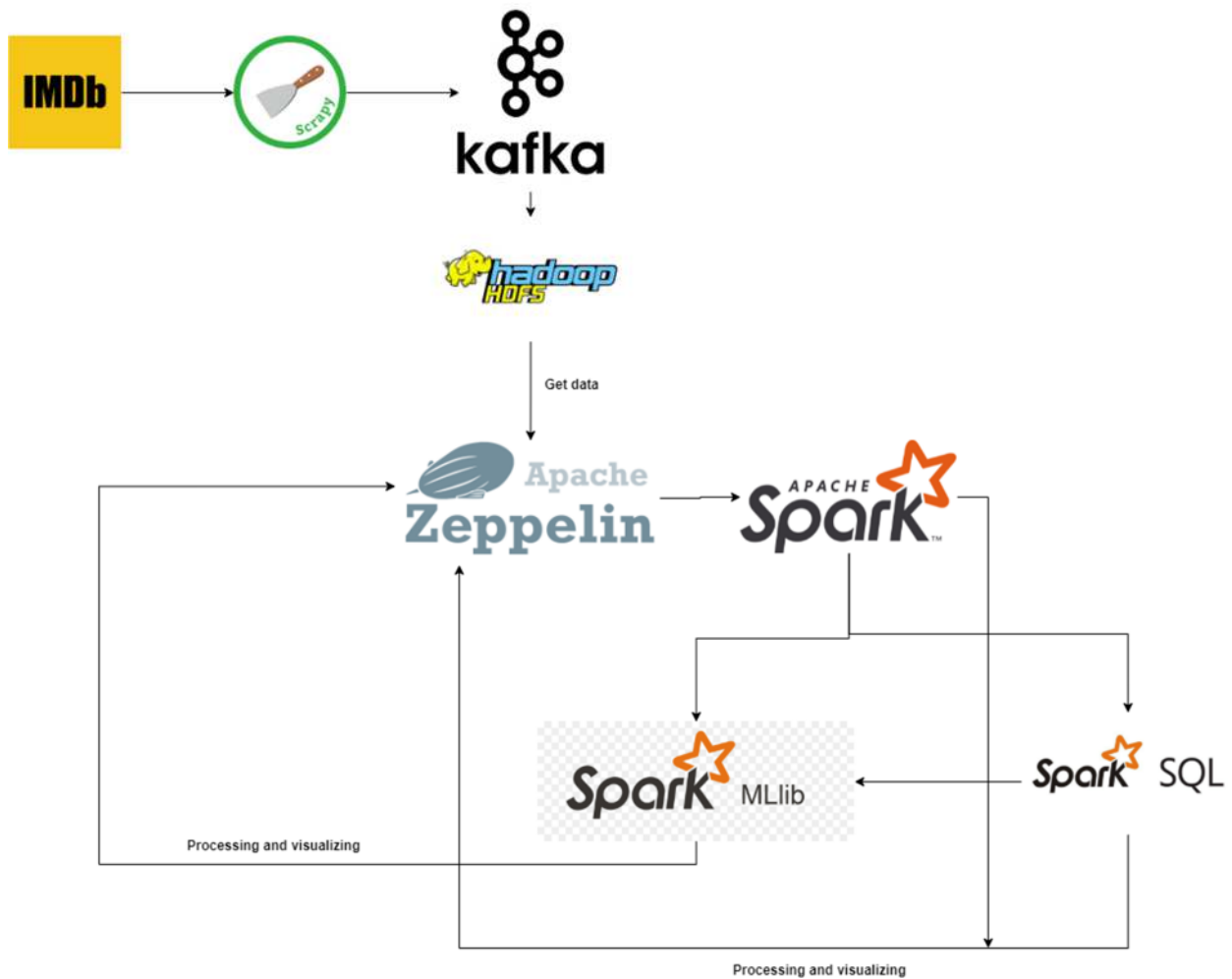
1. ĐẶT VẤN ĐỀ	3
2. MÔ HÌNH HỆ THỐNG.....	4
3. QUÁ TRÌNH THỰC HIỆN.....	4
3.1. Setup hệ thống.....	4
3.1.1. Cài cụm hadoop và HDFS	5
3.1.2. Cài đặt cụm Kafka.....	9
3.1.3. Cài đặt cụm Spark	9
3.1.4. Cài đặt Zeppelin	10
3.2. Thu thập dữ liệu.....	11
3.2.1. Crawl dữ liệu	11
3.2.2. Mô tả dữ liệu.....	13
3.3. Phân tích khám phá dữ liệu (EDA)	14
3.3.1. Đọc dữ liệu từ HDFS.....	14
3.3.2. Sử dụng mô thức Map-Reduce trong Spark để xử lý dữ liệu và visualize bằng Matplotlib để EDA dữ liệu:.....	15
3.4. Xây dựng hệ thống gợi ý phim cho người dùng.....	24
3.4.1. Loading và parsing dữ liệu	25
3.4.2. Phân tích và xử lý dữ liệu	26
3.4.3. Chia tập dataset thành tập train và tập test	28
3.4.4. Xây dựng model ALS và training	29
3.4.5. Save model vào HDFS.....	30
3.4.6. Đưa ra dự đoán trên tập test	31
3.4.7. Đánh giá mô hình trên tập test bằng điểm RMSE.....	31
3.4.8. Đưa ra gợi ý cho người dùng mới	32
3.5. Kết quả chạy Job trên Spark Cluster	33
4. KẾT LUẬN.....	33

1. ĐẶT VẤN ĐỀ

Trong thời đại kỷ nguyên số như hiện tại, Internet mỗi ngày sản sinh ra lượng dữ liệu vô cùng lớn và phong phú. Lượng dữ liệu này rất hữu ích nếu chúng ta có thể thu thập và xử lý chúng và hướng tới sử dụng cho nhiều mục đích khác nhau như nắm bắt mẫu và xu hướng xã hội từ dữ liệu, hay dự đoán hoặc gợi ý để cải thiện trải nghiệm người dùng trên các nền tảng thông tin số và mạng xã hội,... Cùng với sự phát triển của kinh tế - xã hội, nhu cầu giải trí của con người cũng tăng cao, đặc biệt là ở bộ môn nghệ thuật thứ bảy – Điện ảnh.

Trong những năm trở lại đây, ta có thể thấy Điện ảnh phát triển mạnh mẽ như thế nào với số lượng rạp chiếu phim tăng và hàng loạt nền tảng chiếu phim trực tuyến như Netflix, HBO Max,...Cùng với đó, khán giả cũng sẵn đón các thông tin về các bộ phim cũng như có nhu cầu đánh giá các bộ phim mà họ đã xem. Từ đó, dữ liệu về các bộ phim trở thành nguồn dữ liệu hữu ích cho các hệ thống gợi ý phim thương mại. Nhận thấy tầm quan trọng và sự thú vị trong dữ liệu về các bộ phim, chúng em tiến hành thu thập, xử lý, phân tích và xây dựng một hệ gợi ý phim cơ bản dựa trên đặc điểm của các bộ phim. Chúng em sử dụng dữ liệu được thu thập từ nền tảng thông tin điện ảnh trực tuyến IMDB.com và từ một số nguồn khác

2. MÔ HÌNH HỆ THỐNG



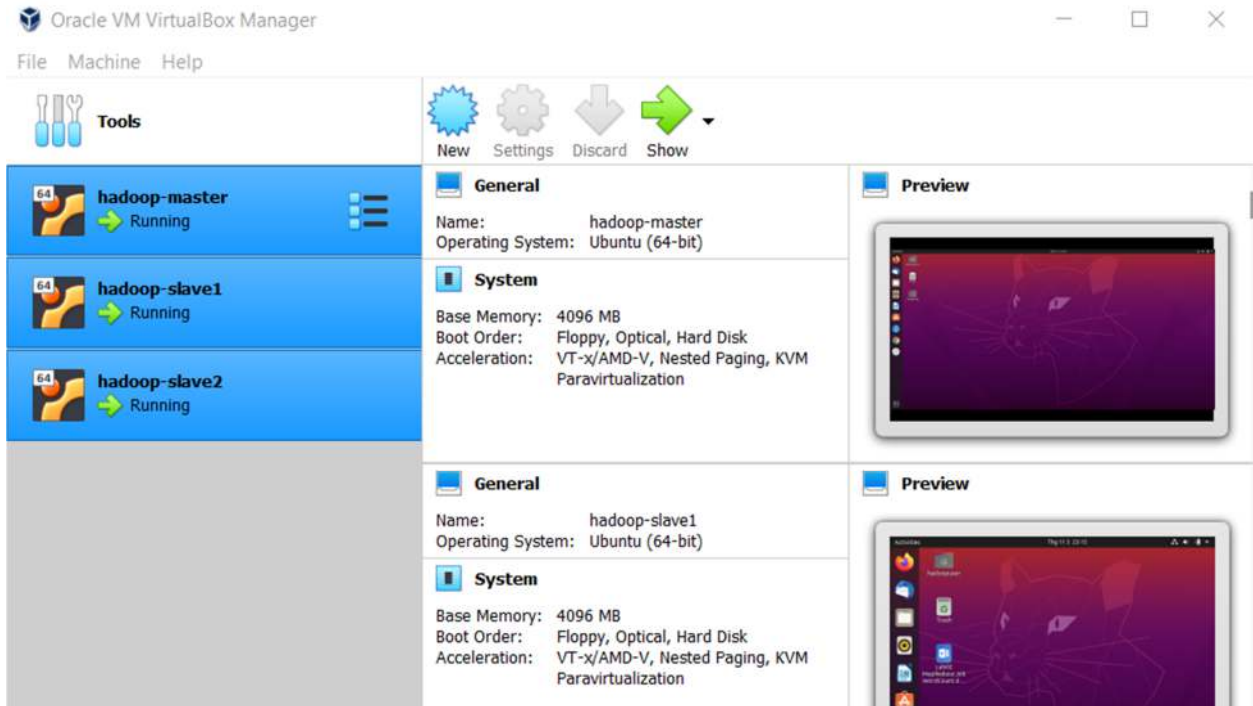
Các bước thực hiện:

- Thu thập dữ liệu chủ yếu từ trang imdb.com bằng Scrapy và từ một số nguồn khác
- Làm sạch tập dữ liệu
- Dữ liệu thu thập được lưu trữ vào HDFS thông qua Kafka
- Trên Zeppelin Notebook truy xuất dữ liệu từ HDFS, xử lý và phân tích dữ liệu sử dụng Spark để EDA các thông tin quan trọng của dữ liệu
- Sử dụng Spark SQL, Spark MLlib để xử lý và dùng các thuật toán học máy để đưa ra gợi ý phim cho người dùng

3. QUÁ TRÌNH THỰC HIỆN

3.1. Setup hệ thống

3 máy Master, Slave1, Slave2



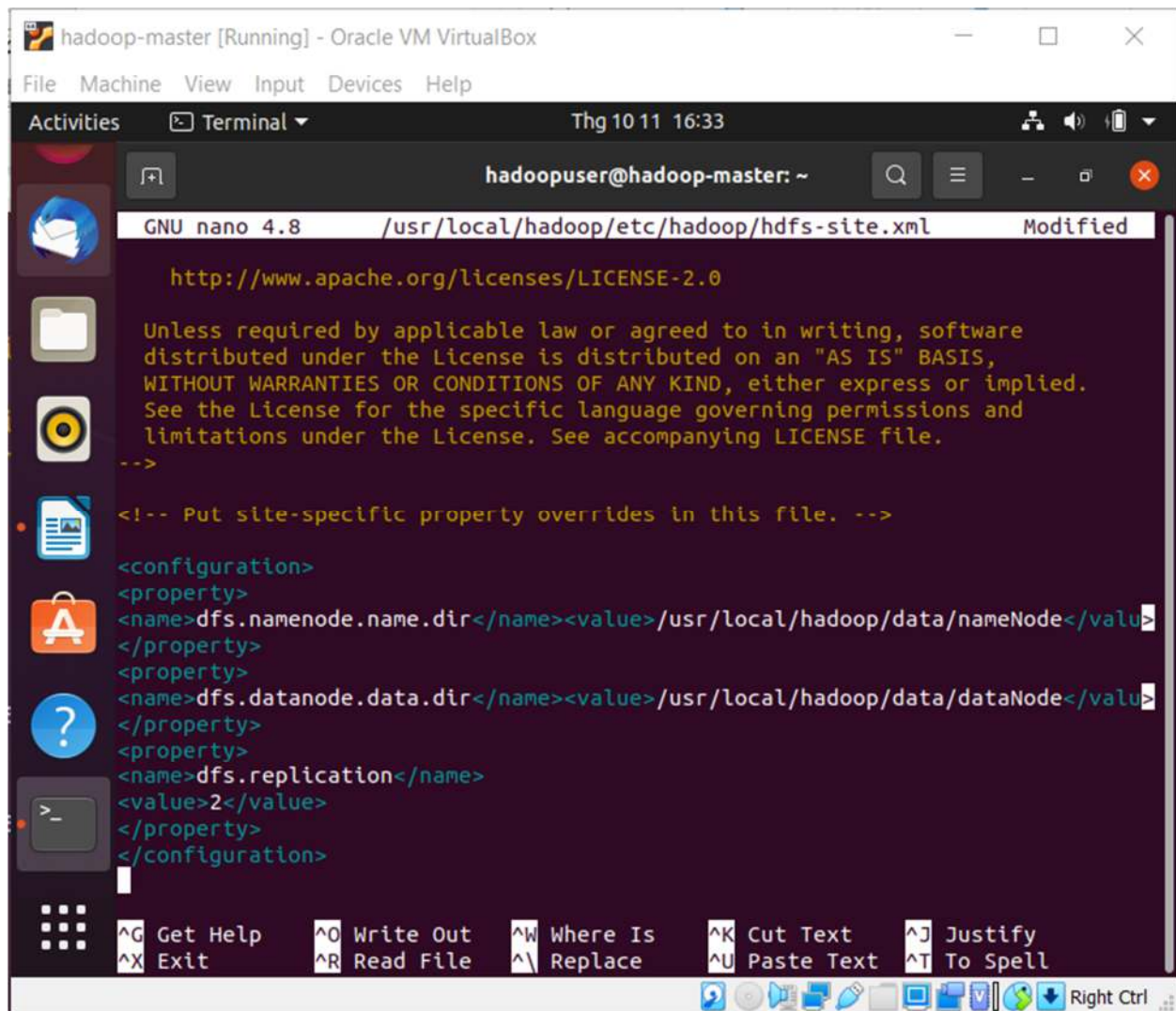
Địa chỉ IP của 3 máy:

```
hadoopuser@hadoop-master: ~
GNU nano 4.8 /etc/hosts
127.0.0.1    localhost
127.0.1.1    hadoop-VirtualBox

192.168.56.101 hadoop-master
192.168.56.102 hadoop-slave1
192.168.56.103 hadoop-slave2
# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

3.1.1. Cài cụm hadoop và HDFS

Name Node là máy hadoop-master có địa chỉ IP là: 192.168.56.101, 2 Data Node là 2 máy hadoop-slave1 và hadoop-slave2 với địa chỉ IP lần lượt là: 192.168.56.102 và 192.168.56.103. Đặt block-size = 128MB và replication = 2:



The screenshot shows a VirtualBox window titled "hadoop-master [Running] - Oracle VM VirtualBox". Inside, a terminal window is open with the prompt "hadoopuser@hadoop-master: ~". The terminal is running the GNU nano 4.8 text editor, editing the file "/usr/local/hadoop/etc/hadoop/hdfs-site.xml". The file content includes the Apache License 2.0 text and XML configuration for HDFS. The configuration section is as follows:

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

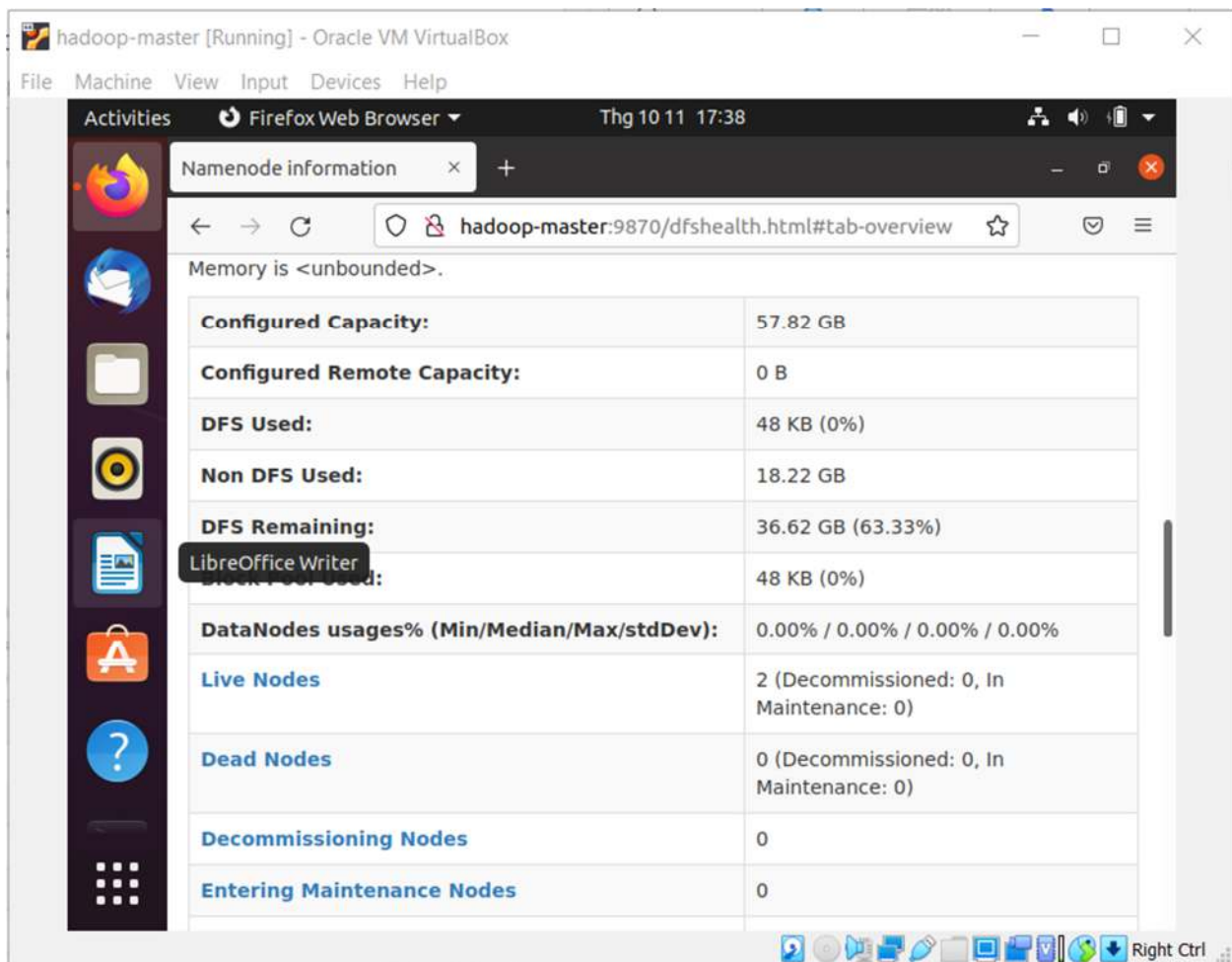
<configuration>
<property>
<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

The bottom of the terminal window shows a list of keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^\ Replace, ^U Paste Text, ^T To Spell. The system tray at the bottom right shows the date and time "Thg 10 11 16:33" and a "Right Ctrl" button.

Sử dụng lệnh sau để start:

```
$/usr/local/hadoop/sbin/start-dfs.sh
```

Thông số cluster:



hadoop-master [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Firefox Web Browser Thg 10 11 17:39

Namenode information x +

hadoop-master:9870/dfshealth.html#tab-datanode

In operation

Show 25 entries

Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Ve
✓ hadoop-slave1:9866 (192.168.56.102:9866)	http://hadoop-slave1:9864	2s	4m	28.91 GB	0	24 KB (0%)	
✓ hadoop-slave2:9866 (192.168.56.103:9866)	http://hadoop-slave2:9864	2s	4m	28.91 GB	0	24 KB (0%)	

Showing 1 to 2 of 2 entries

Previous 1 Next

Right Ctrl

Hadoop HDFS: cổng 9870

Overview 'hadoop-master:9000' (active)

Started:	Sat Nov 20 14:49:56 +0700 2021
Version:	3.2.1, r03cbbb467e22ea829b38084b7001d07e0b3842
Compiled:	Tue Sep 10 22:56:00 +0700 2019 by rohithsharmaks from branch-3.2.1
Cluster ID:	CID-e6a7a06c-93a9-4b47-b774-3eee9f616abb
Block Pool ID:	BP-972345355-192.168.56.101-1633948422136

Summary

Security is off.
 Safemode is off.
 25 files and directories, 24 blocks (24 replicated blocks, 0 erasure coded block groups) = 49 total filesystem object(s).
 Heap Memory used 42.05 MB of 61.86 MB Heap Memory. Max Heap Memory is 951.25 MB.
 Non Heap Memory used 60.23 MB of 61.73 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	57.82 GB
Configured Remote Capacity:	0 B
DFS Used:	4.35 GB (7.52%)
Non DFS Used:	24.62 GB
DFS Remaining:	25.87 GB (44.75%)
Block Pool Used:	4.35 GB (7.52%)
DataNodes usages% (Min/Median/Max/stdDev):	7.52% / 7.52% / 7.52% / 0.00%
Live Nodes	2 (Decommissioned: 0, In Maintenance: 0)

3.1.2. Cài đặt cụm Kafka

Kafka: 2.8.1

Ref: <https://blog.clairvoyantsoft.com/kafka-series-3-creating-3-node-kafka-cluster-on-virtual-box-87d5edc85594>

Mặc định kafka giao tiếp ở cổng 9092, zookeeper giao tiếp ở cổng 2181

Tạo topic mới bằng kafka

Topic "squadgame" tại 192.168.56.101:9092

```
zookeeper.connect=zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka
```

```
server.1=zookeeper1:2888:3888
server.2=zookeeper2:2888:3888
server.3=zookeeper3:2888:3888
```

Xem danh sách các topic tại 192.168.56.101:2181

3.1.3. Cài đặt cụm Spark

Máy Master là máy hadoop-master với địa chỉ IP: 192.168.56.101, 2 máy Worker là hadoop-slave1 và hadoop-slave2 với địa chỉ IP lần lượt là: 192.168.56.102 và 192.168.56.103.

Sử dụng lệnh sau để start:

```
$ cd /usr/local/spark
$ ./sbin/start-all.sh
```

Spark: cổng 8080

Spark Master at spark://192.168.56.101:7077

URL: spark://192.168.56.101:7077
 Alive Workers: 3
 Cores in use: 3 Total, 0 Used
 Memory in use: 8.5 GB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory
worker-20211120145453-192.168.56.102-39365	192.168.56.102:39365	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)
worker-20211120145453-192.168.56.103-40701	192.168.56.103:40701	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)
worker-20211120145457-192.168.56.101-45639	192.168.56.101:45639	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

3.1.4. Cài đặt Zeppelin

Config lại port của Zeppelin thành 8888 để tránh trùng với port của Spark

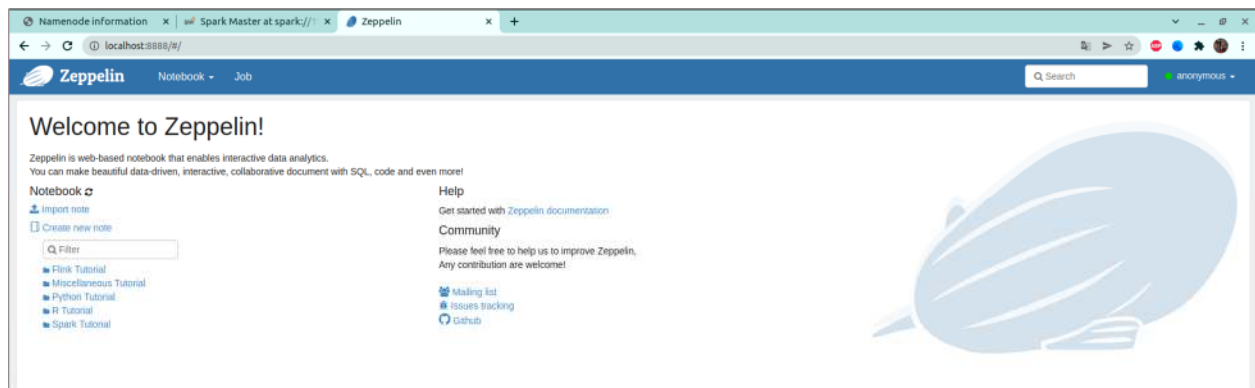
```

hadoopuser@hadoop-master: ~/zeppelin-0.10.0-bin-all
GNU nano 4.8 conf/zeppelin-env.sh Modified
# export JAVA_HOME=
# export USE_HADOOP=
# export SPARK_MASTER=
# export ZEPPELIN_ADDR
# export ZEPPELIN_PORT
# export ZEPPELIN_LOCAL_IP
# export ZEPPELIN_JAVA_OPTS
# export ZEPPELIN_MEM
# export ZEPPELIN_INTTP_MEM
# export ZEPPELIN_INTTP_JAVA_OPTS
# export ZEPPELIN_SSL_PORT
# export ZEPPELIN_JMX_ENABLE
# export ZEPPELIN_JMX_PORT
export ZEPPELIN_PORT=8888
# export ZEPPELIN_LOG_DIR
# Whether include hadoop jars i
# Spark master url. eg. spark:/>
# Bind address (default 127.0.0.>
# port number to listen (defaul>
# Zeppelin's thrift server ip a>
# Additional jvm options. for e>
# Zeppelin jvm mem options Defa>
# zeppelin interpreter process >
# zeppelin interpreter process >
# ssl port (used when ssl envir>
# Enable JMX feature by definin>
# Port number which JMX uses. I>
# Where log files are stored. >
  
```

Start Zeppelin:

Vào thư mục cài Zeppelin gõ lệnh: `./bin/zeppeliin-daemon.sh start`

Mở UI: localhost:8888



Vào interpreter config spark.master từ local[*] thành địa chỉ master cluster để chạy trên Spark cluster:

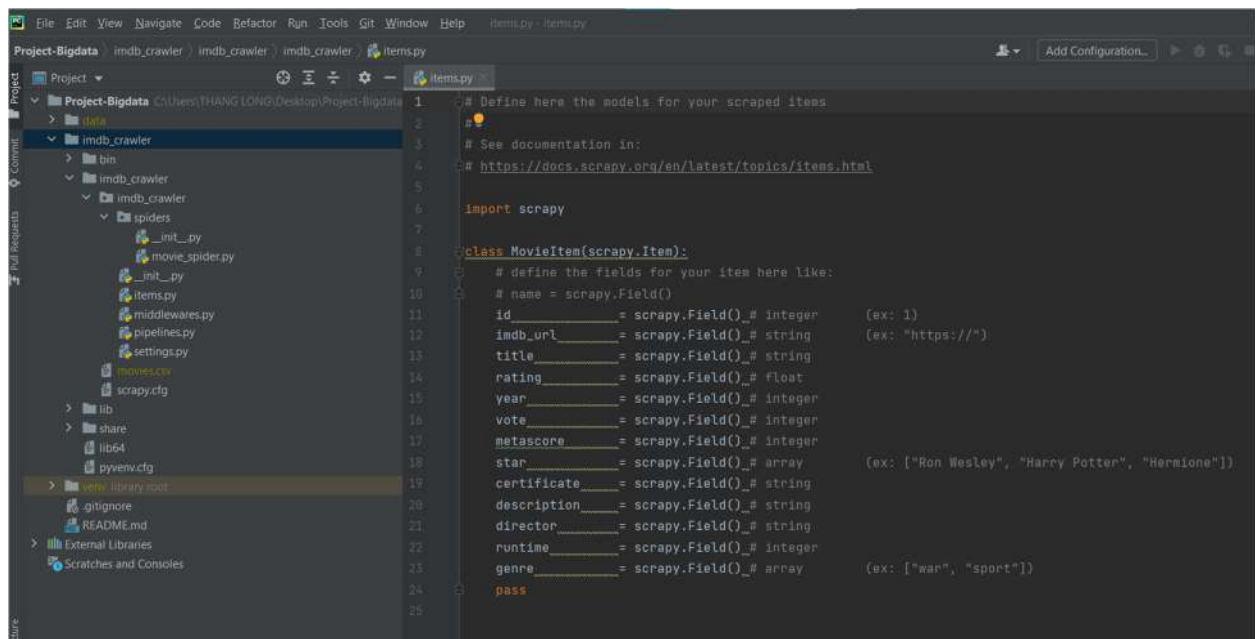
spark.master spark://192.168.56.101:7077

3.2. Thu thập dữ liệu

3.2.1. Crawl dữ liệu

Crawl dữ liệu trên trang imdb.com sử dụng Scrapy

Mã nguồn: cấu trúc 1 item:



Cụ thể:

- **id**: Định danh của bộ phim trên IMDb
- **imdb_url**: Địa chỉ url của bộ phim trên imdb
- **title**: Tên bộ phim
- **rating**: Điểm đánh giá trung bình bộ phim nhận được trên IMDb

- *year*: Năm phát hành của bộ phim
- *vote*: Số lượng vote cho bộ phim
- *metascore*: Điểm đánh giá từ những nguồn khác
- *star*: Danh sách các ngôi sao tham gia vào bộ phim
- *certificate*: Chứng chỉ của bộ phim (vd: C13)
- *description*: Mô tả nội dung phim
- *director*: Đạo diễn của bộ phim
- *runtime*: Thời lượng của bộ phim
- *genre*: Danh sách thể loại của bộ phim

Để tiến hành crawl dữ liệu cần chạy mã nguồn trên bằng lệnh: `scrapy crawl movie -o movies.csv -t csv`

Kết quả crawl:

```
Terminal - Project-Bigdata
Terminal: Local x +
2021-11-06 01:19:32 [scrapy.core.engine] INFO: Closing spider (finished)
2021-11-06 01:19:32 [scrapy.extensions.feedexport] INFO: Stored csv feed (226295 items) in: movies.csv
2021-11-06 01:19:32 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 572698,
 'downloader/request_count': 911,
 'downloader/request_method_count/GET': 911,
 'downloader/response_bytes': 1157288587,
 'downloader/response_count': 911,
 'downloader/response_status_count/200': 908,
 'downloader/response_status_count/301': 1,
 'downloader/response_status_count/500': 2,
 'dupefilter/filtered': 41,
 'elapsed_time_seconds': 12214.197993,
 'feedexport/success_count/FileFeedStorage': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2021, 11, 5, 18, 19, 32, 665969),
 'item_scraped_count': 226295,
 'log_count/DEBUG': 227207,
 'log_count/INFO': 214,
 'request_depth_max': 904,
 'response_received_count': 908,
 'retry/count': 2,
 'retry/reason_count/500 Internal Server Error': 2,
 'robotstxt/request_count': 1,
 'robotstxt/response_count': 1,
 'robotstxt/response_status_count/200': 1,
 'scheduler/dequeued': 910,
 'scheduler/dequeued/memory': 910,
 'scheduler/enqueued': 910,
 'scheduler/enqueued/memory': 910,
 'start_time': datetime.datetime(2021, 11, 5, 14, 55, 58, 467976)}
2021-11-06 01:19:32 [scrapy.core.engine] INFO: Spider closed (finished)

(venv) C:\Users\THANG LONG\Desktop\Project-Bigdata\imdb_crawler\imdb_crawler>
```

Mã nguồn crawl dữ liệu imdb: <https://github.com/thanglong2000pro/Project-Bigdata>

3.2.2. Mô tả dữ liệu

Tổng dữ liệu gần 2GB lưu trên HDFS:

Browse Directory

Show 25 entries
 Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	395.63 MB	Nov 20 16:05	2	128 MB	genome-scores.csv	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	17.68 KB	Nov 20 16:05	2	128 MB	genome-tags.csv	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	1.21 MB	Nov 20 16:05	2	128 MB	links.csv	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	2.73 MB	Nov 20 16:05	2	128 MB	movies.csv	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	724.03 MB	Nov 20 16:05	2	128 MB	ratings.csv	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	37.9 MB	Nov 20 16:05	2	128 MB	tags.csv	

Showing 1 to 6 of 6 entries

Previous
 1
 Next

Hadoop, 2019.

Browse Directory

Show 25 entries
 Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoopuser	supergroup	679.06 MB	Nov 20 16:06	2	128 MB	movies.csv	

Showing 1 to 1 of 1 entries

Previous
 1
 Next

Hadoop, 2019.

3.3. Phân tích khám phá dữ liệu (EDA)

3.3.1. Đọc dữ liệu từ HDFS

- Check config file hadoop hdfs: **core-site.xml** để lấy chính xác đường dẫn tuyệt đối của HDFS, từ đó đọc được file từ HDFS vào Spark (ở đây là hdfs://hadoop-master:9000/...):

```
sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
hadoopuser@hadoop-master:~$ sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml
[sudo] password for hadoopuser:
```

3.3.2. Sử dụng mô thức Map-Reduce trong Spark để xử lý dữ liệu và visualize bằng Matplotlib để EDA dữ liệu:

3.3.2.1. Top đạo diễn sản xuất nhiều phim nhất

```
%pyspark
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel('WARN')

spark = SparkSession(sc)

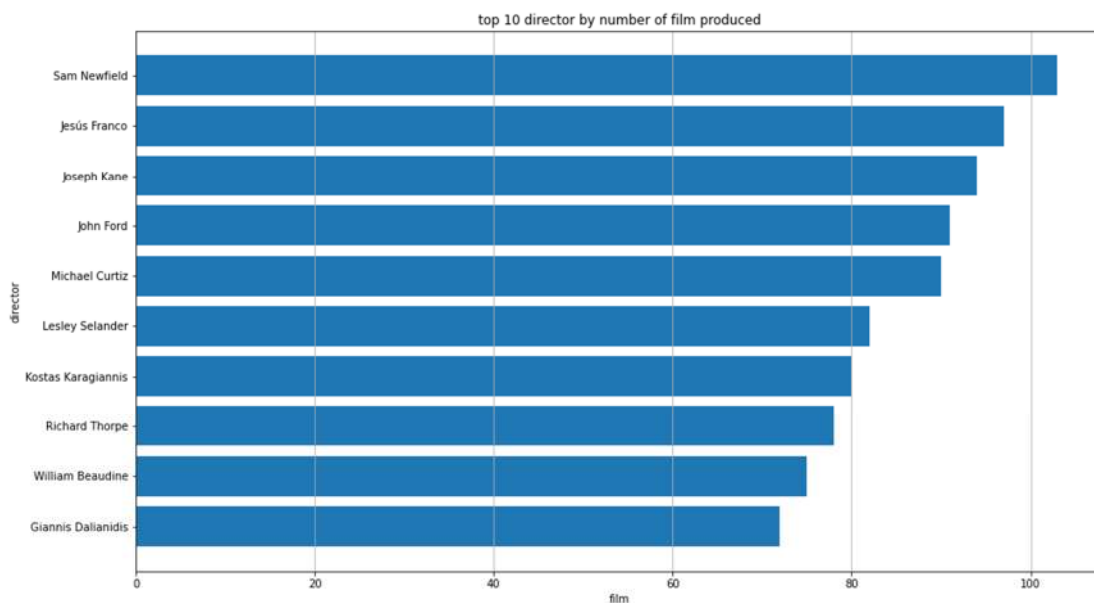
movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

directorRDD = movieDF.select("director").filter(movieDF.director != '').rdd
directorRDD = directorRDD.flatMap(lambda x: x)
directorPairRDD = directorRDD.map(lambda x: (x, 1))
directorPairRDD = directorPairRDD.reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=True)

directorKeys = directorPairRDD.keys().collect()
directorValues = directorPairRDD.values().collect()

figure(figsize=(16, 9))
plt.grid(axis="x")
plt.xlabel("film")
plt.ylabel("director")
plt.title("top 11 director by number of film produced")

plt.barh(directorKeys[-11:], directorValues[-11:])
plt.show()
#plt.savefig("/datashared/figure1.png")
```



Kết luận: Sam Newfield là vị đạo diễn nhiều phim nhất với hơn 100 phim được ghi nhận

3.3.2.2. Tỷ lệ các chứng chỉ phim

```
%pyspark
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

certRDD = movieDF.select("certificate").filter(movieDF.certificate != '').rdd
certPairRDD = certRDD.flatMap(lambda x: x).map(lambda x: (x, 1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)

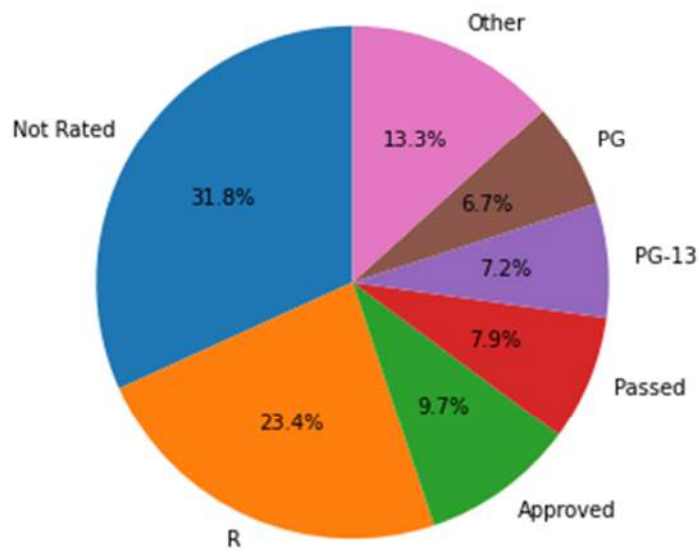
allCertKeys = certPairRDD.keys().collect()
allCertValues = certPairRDD.values().collect()

otherKey = "Other"
otherValue = 0
for i in allCertValues[6:]:
    otherValue += i

certKeys = allCertKeys[:6]
certKeys.append(otherKey)

certValues = allCertValues[:6]
certValues.append(otherValue)

plt.pie(certValues, labels = certKeys, startangle=90, autopct='%1.1f%%')
plt.show()
# plt.savefig("/datashared/figure2.png")
```



Kết luận:

- Các phim chưa có chứng nhận (Not Rated) chiếm tỷ trọng lớn (31.8%)

- Các phim có nhãn R (Restricted) cũng chiếm một tỷ trọng lớn nhất (23.4%)
- Các phim có nhãn PG, PG-13, Passed, Approved chiếm tỷ trọng khá tương đồng nhau
- Còn lại là các phim có nhãn khác

3.3.2.3. Phân bố số lượng phim theo điểm đánh giá rating

```
%pyspark
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel('WARN')

spark = SparkSession(sc)

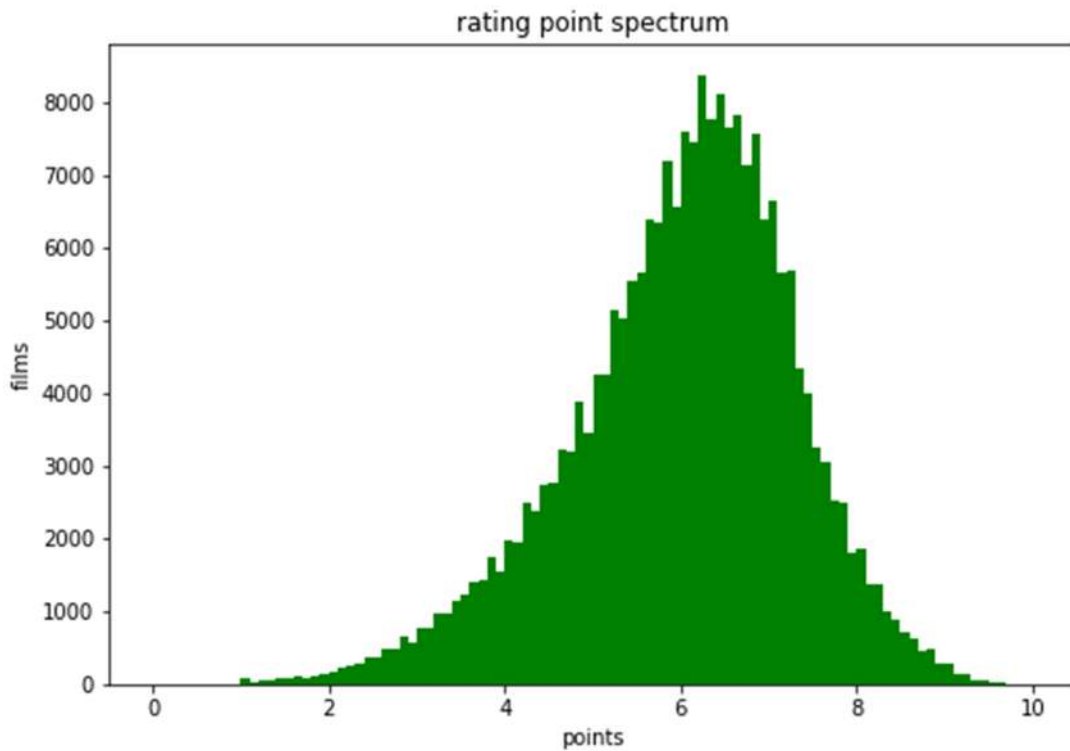
movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

ratingRDD = movieDF.select("rating").filter(movieDF.rating != '').rdd

def isfloat(value):
    try:
        float(value)
        return True
    except ValueError:
        return False

ratings = ratingRDD.flatMap(lambda x: x).filter(lambda x: isfloat(x)).collect()
for i in range(len(ratings)):
    ratings[i] = float(ratings[i])

bins = [x/10 for x in range(101)]
plt.xlabel("points")
plt.ylabel("films")
plt.title("rating point spectrum")
plt.hist(ratings, bins, color='g')
plt.show()
# plt.savefig("/datashared/figure3.png")
```



Kết luận: Số lượng phim tập trung đồng nhất ở mức 6 điểm với tới hơn 8000 phim. Đa số các phim có số điểm từ 5.5 – 6.5 điểm. Không có phim nào đạt điểm 10. Có rất ít phim đạt điểm > 9 (chưa đến 500 phim)

3.3.2.4. Phân bố số lượng phim theo điểm đánh giá metascore

```
%pyspark
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

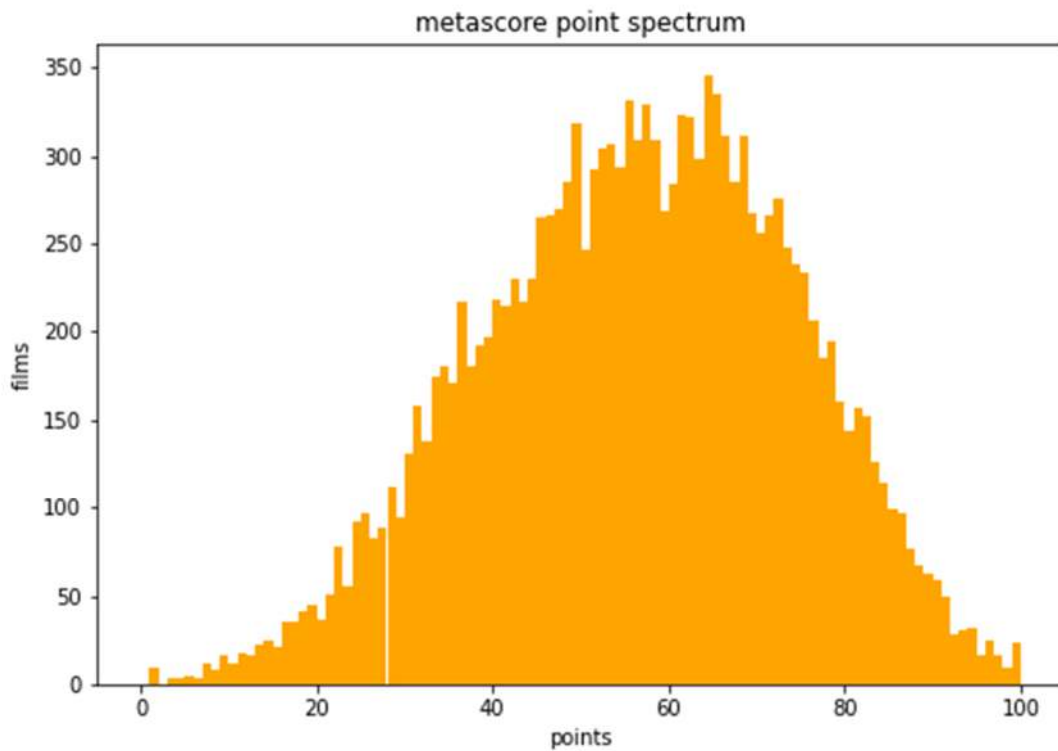
movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

ratingRDD = movieDF.select("metascore").filter(movieDF.metascore != '').rdd

def isfloat(value):
    try:
        float(value)
        return True
    except ValueError:
        return False

ratings = ratingRDD.flatMap(lambda x: x).filter(lambda x: isfloat(x)).collect()
for i in range(len(ratings)):
    ratings[i] = float(ratings[i])

bins = [x for x in range(101)]
plt.xlabel("points")
plt.ylabel("films")
plt.title("metascore point spectrum")
plt.hist(ratings, bins, color="orange")
plt.show()
```



Kết luận: Số lượng phim đạt điểm 70 là nhiều nhất với hơn 300 phim, có chưa đến 50 phim đạt 100 điểm metascore. Đa số phim có số điểm phân bố từ 40 cho đến 80 điểm.

3.3.2.5. Mối tương quan giữa điểm IMDB và điểm Metacritic

```
%pyspark
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

mPairRDD = movieDF.select("rating", "metascore").filter((movieDF.rating != '') & (movieDF.metascore != '')).rdd

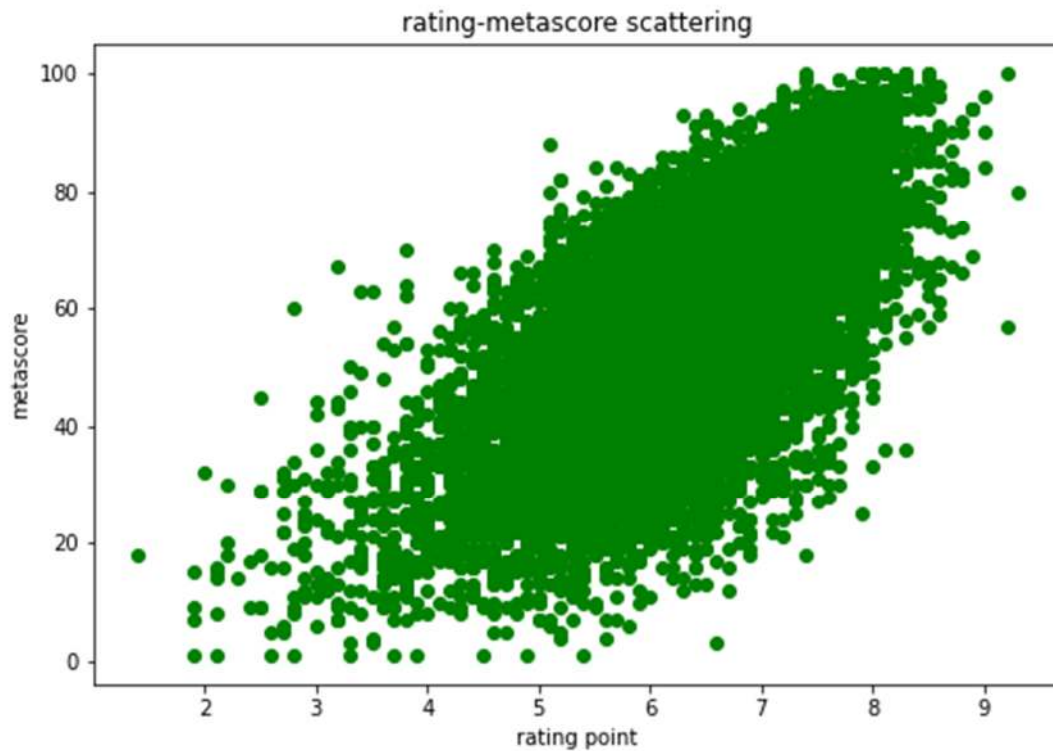
mRatings = mPairRDD.keys().collect()
mMetascores = mPairRDD.values().collect()

for i in range(len(mRatings)):
    mRatings[i] = float(mRatings[i])

for i in range(len(mMetascores)):
    mMetascores[i] = int(mMetascores[i])

plt.xlabel("rating point")
plt.ylabel("metascore")
plt.title("rating-metascore scattering")

plt.scatter(mRatings, mMetascores, color='orange')
plt.show()
#plt.savefig("/datashared/figure5.png")
```



Kết luận: Có sự khác nhau về điểm đánh giá giữa hai hệ thống IMBb (rating point) và Metacritic (metascore) là khá lớn: có rất nhiều phim được đánh giá tốt ở IMBb nhưng lại bị đánh giá kém ở Metacritic và ngược lại.

3.3.2.6. Số lượng phim theo năm ra mắt

```
%pyspark
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

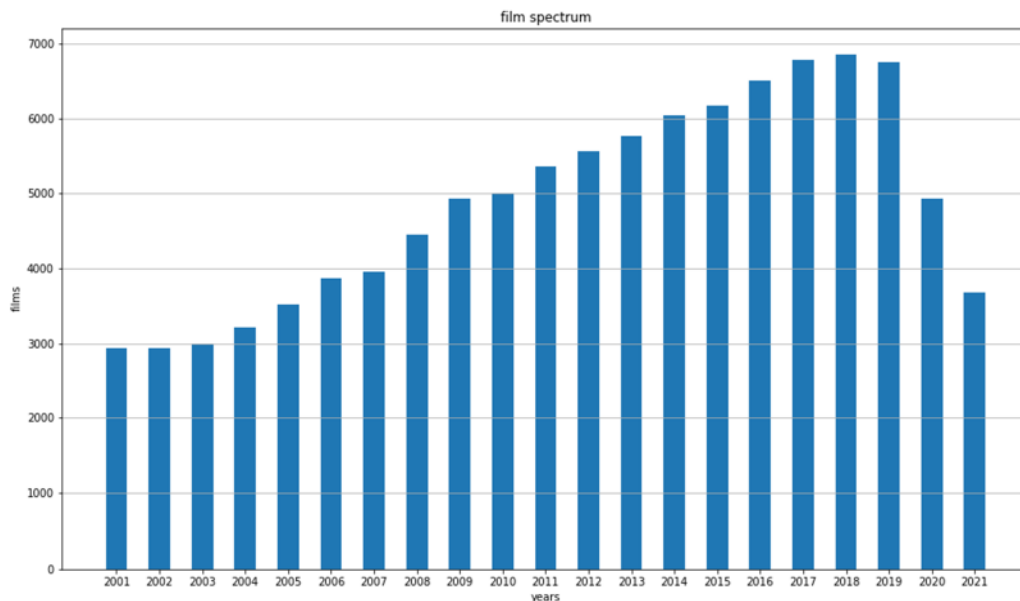
movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

yearRDD = movieDF.select("year").filter(movieDF.year != '').rdd
yearPairRDD = yearRDD.flatMap(lambda x: x).filter(lambda x: x.isnumeric()).map(lambda x: (x, 1))
yearPairRDD = yearPairRDD.reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[0], ascending=True)

yearKeys = yearPairRDD.keys().collect()
yearValues = yearPairRDD.values().collect()

figure(figsize=(16, 9))
plt.xlabel("years")
plt.ylabel("films")
plt.title("film spectrum")
plt.grid(axis="y")

plt.bar(yearKeys[-21:], yearValues[-21:], width=0.5)
plt.show()
# plt.savefig("/datashared/figure4.png")
```



Kết luận:

- Số lượng phim được xuất bản tăng đều từ đầu thế kỷ 21: từ mức xấp xỉ 3000 phim ở năm 2001, sau 17 năm đã lên tới gần 7000 phim ở năm 2018
- Dịch covid 19 bùng phát cuối năm 2019 đã ảnh hưởng trầm trọng tới ngành công nghiệp điện ảnh, số lượng phim sản xuất và ra mắt giảm sút đáng kể ở các năm sau đó

3.3.2.7. Thống kê thời lượng các bộ phim

```
%pyspark
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

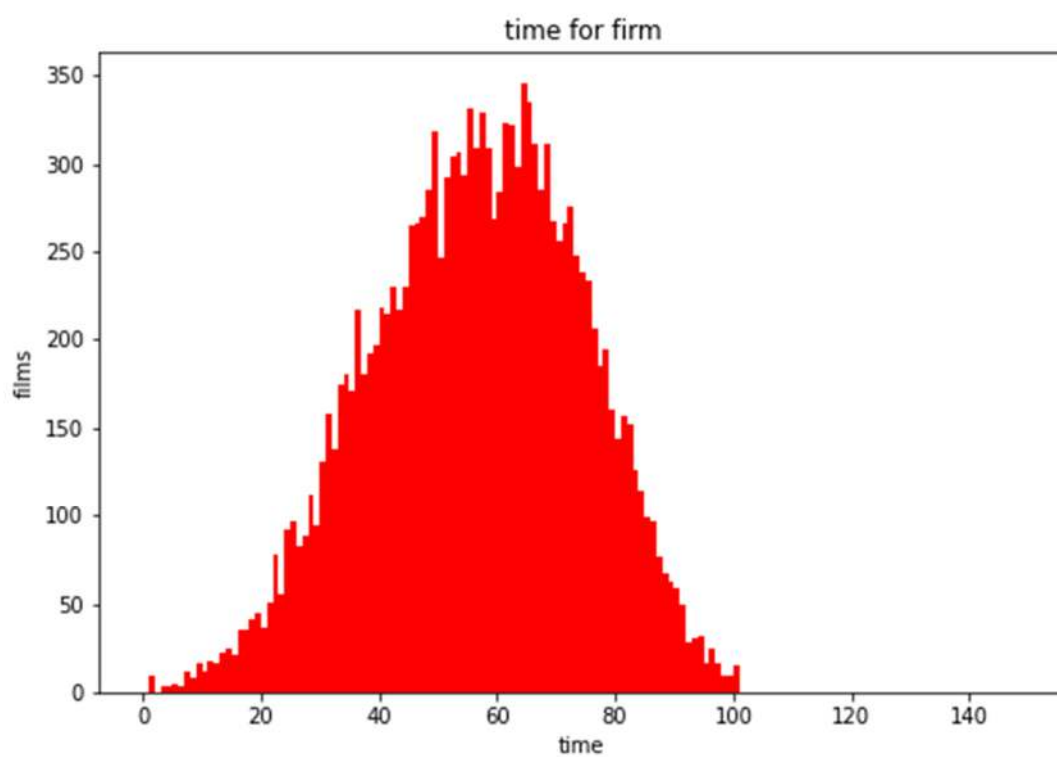
movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

runtimeRDD = movieDF.select("runtime").filter(movieDF.runtime != '').rdd

def isfloat(value):
    try:
        float(value)
        return True
    except ValueError:
        return False

runtime = runtimeRDD.flatMap(lambda x: x).filter(lambda x: isfloat(x)).collect()
for i in range(len(runtime)):
    runtime[i] = float(runtime[i])

bins = [x for x in range(150)]
plt.xlabel("time")
plt.ylabel("films")
plt.title("time for firm")
plt.hist(runtime, bins, color="red")
plt.show()
```



Kết luận: Các phim chủ yếu có thời lượng từ 50 phút cho đến 90 phút

3.3.2.8. Top thể loại phim được yêu thích nhất

```

%pyspark
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import numpy as np

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('Squad_Game')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

movieDF = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset2/movies.csv")

genreRDD = movieDF.select("genre").filter(movieDF.genre != '').rdd
totalFilm = genreRDD.count()
genrePairRDD = genreRDD.flatMap(lambda x: x).flatMap(lambda x: x.split(",")).map(lambda x: (x, 1))
genrePairRDD = genrePairRDD.reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)

allGenreKeys = genrePairRDD.keys().collect()
allGenreValues = genrePairRDD.values().collect()

otherKey = "Other"
otherValue = 0
for i in allGenreValues[8:]:
    otherValue += i

genreValues = allGenreValues[:8]
genreValues.append(otherValue)

genreKeys = allGenreKeys[:8]
genreKeys.append(otherKey)
for i in range(len(genreKeys)):
    genreKeys[i] = str(genreValues[i]) + str(" films ") + genreKeys[i]

fig, ax = plt.subplots(figsize=(16, 9), subplot_kw=dict(aspect="equal"))

wedges, texts = ax.pie(genreValues, wedgeprops=dict(width=0.5), startangle=90)

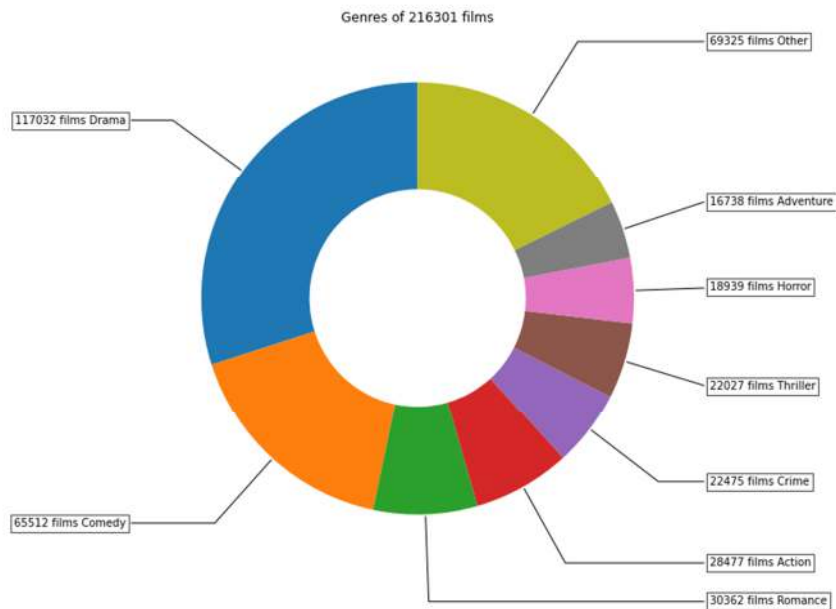
bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
kw = dict(arrowprops=dict(arrowstyle="-"),
        bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(genreKeys[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y), horizontalalignment=horizontalalignment, **kw)

ax.set_title("Genres of " + str(totalFilm) + " films")

plt.show()
# plt.savefig("/datashared/figure6.png")

```



Kết luận:

- Dòng phim thể loại Drama là dòng phim có số lượng phim nhiều nhất (hơn 100K phim), tiếp sau đó là dòng phim Comedy (hài hước) với hơn 60K phim
- Dòng phim Romance (lãng mạn) và Action (hành động) cũng có số lượng khá ấn tượng (~ 30K phim) và cũng thu hút nhiều sự quan tâm của người xem

3.4. Xây dựng hệ thống gợi ý phim cho người dùng

Xây dựng hệ thống gợi ý phim dựa trên điểm rating. Lọc cộng tác (Collaborative filtering) thường được sử dụng cho các hệ thống khuyến nghị. Các kỹ thuật này nhằm mục đích điền vào các mục còn thiếu của ma trận liên kết mục người dùng, trong bài toán này là ma trận xếp hạng phim người dùng. MLlib hiện hỗ trợ lọc cộng tác dựa trên mô hình, trong đó người dùng và sản phẩm được mô tả bằng một tập hợp nhỏ các yếu tố tiềm ẩn có thể được sử dụng để dự đoán các mục nhập bị thiếu. Ở đây nhóm em triển khai thuật toán bình phương tối thiểu xen kẽ (ALS) để tìm hiểu các yếu tố tiềm ẩn này.

3.4.1. Loading và parsing dữ liệu

Loading and parsing datasets

```
%pyspark
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkConf

conf = SparkConf()
conf.setMaster('spark://192.168.56.101:7077')
conf.setAppName('recommend')

sc = SparkContext.getOrCreate(conf=conf)
#sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")

spark = SparkSession(sc)

data = spark.read.format("csv").option("header", "true").option("encoding", "UTF-8").load("hdfs://hadoop-master:9000/dataset/dataset1/ratings.csv")
data.printSchema()

root
|-- userId: string (nullable = true)
|-- movieId: string (nullable = true)
|-- rating: string (nullable = true)
|-- timestamp: string (nullable = true)
```

3.4.2. Phân tích và xử lý dữ liệu

```
%sql
CREATE TABLE ratings (
  userId int,
  movieId int,
  rating double,
  ts int
)
USING CSV
OPTIONS (path "hdfs://hadoop-master:9000/dataset/dataset1/ratings.csv", header="true")
```

Took 2 sec. Last updated by anonymous at November 22 2021, 11:19:02 AM.

```
%pyspark
data = spark.table("ratings")
data.show()
```

```
+-----+-----+-----+-----+
|userId|movieId|rating|      ts|
+-----+-----+-----+-----+
|      1|      307|    3.5|1256677221|
|      1|      481|    3.5|1256677456|
|      1|     1091|    1.5|1256677471|
|      1|     1257|    4.5|1256677460|
|      1|     1449|    4.5|1256677264|
|      1|     1590|    2.5|1256677236|
|      1|     1591|    1.5|1256677475|
|      1|     2134|    4.5|1256677464|
|      1|     2478|    4.0|1256677239|
|      1|     2840|    3.0|1256677500|
|      1|     2986|    2.5|1256677496|
|      1|     3020|    4.0|1256677260|
|      1|     3424|    4.5|1256677444|
|      1|     3698|    3.5|1256677243|
|      1|     3826|    2.0|1256677210|
```

```
%pyspark
#Count null value
from pyspark.sql.functions import col,sum
data.select(*(sum(col(c).isNull().cast("int")).alias(c) for c in data.columns)).show()
```

```
+-----+-----+-----+-----+
|userId|movieId|rating|timestamp|
+-----+-----+-----+-----+
|      0|      0|      0|      0|
+-----+-----+-----+-----+
```

Took 1 min 7 sec. Last updated by anonymous at November 22 2021, 10:55:02 AM.

```
%pyspark
#Count Null value
from pyspark.sql.functions import lit, col

rows = data.count()
summary = data.describe().filter(col("summary") == "count")
summary.select*((lit(rows)-col(c)).alias(c) for c in data.columns)).show()
```

```
+-----+-----+-----+-----+
|userId|movieId|rating|timestamp|
+-----+-----+-----+-----+
|  0.0|  0.0|  0.0|  0.0|
+-----+-----+-----+-----+
```

Took 2 min 56 sec. Last updated by anonymous at November 22 2021, 10:58:27 AM.

```
%pyspark
print('No. of row: %d' % data.count())
data.show(5)
```

No. of row: 27753444

```
+-----+-----+-----+-----+
|userId|movieId|rating|timestamp|
+-----+-----+-----+-----+
|      1|      307|      3.5|1256677221|
|      1|      481|      3.5|1256677456|
|      1|     1091|      1.5|1256677471|
|      1|     1257|      4.5|1256677460|
|      1|     1449|      4.5|1256677264|
+-----+-----+-----+-----+
```

only showing top 5 rows

```
%pyspark
# count, mean, std, min & max
data.describe().show()
```

summary	userId	movieId	rating	timestamp
count	27753444	27753444	27753444	27753444
mean	141942.01557064414	18487.99983414671	3.5304452124932677	1.1931218549319255E9
stddev	81707.40009148984	35102.6252474677	1.0663527502319696	2.1604822852233613E8
min	1	1	0.5	1000000065
max	99999	99999	5.0	999999978

3.4.3. Chia tập dataset thành tập train và tập test

```
%pyspark
#Split dataset to train and test
train_data, test_data = data.randomSplit([0.8, 0.2])
```

3.4.4. Xây dựng model ALS và training

Alternating Least Squares (ALS)

```
%pyspark
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
```

Took 0 sec. Last updated by anonymous at November 22 2021, 11:20:43 AM.

```
%pyspark
# Build the recommendation model using ALS on the training data
als = ALS(maxIter=10, regParam=0.1, rank=8, nonnegative=True, coldStartStrategy="drop", \
         userCol='userId', itemCol='movieId', ratingCol='rating')
model = als.fit(train_data)
```

Took 13 min 50 sec. Last updated by anonymous at November 22 2021, 11:34:38 AM.

```
%pyspark
print('Factorized user matrix with rank = %d' % model.rank)
model.userFactors.show(5)

print('-'*50)

print('Factorized item matrix with rank = %d' % model.rank)
model.itemFactors.show(5)
```

Factorized user matrix with rank = 8

```
+-----+
| id|          features|
+-----+
| 10|[0.06614851, 1.02...|
| 20|[0.0, 1.469784, 0...|
| 30|[0.4757676, 0.544...|
| 40|[1.3379295, 0.604...|
| 50|[0.48787823, 1.37...|
```

only showing top 5 rows

Factorized item matrix with rank = 8

```
+-----+
| id|          features|
+-----+
| 10|[0.06614851, 1.02...|
```

```
%pyspark
print('Recommended top users (e.g. 1 top user) for all items with the corresponding predicted ratings:')
model.recommendForAllItems(1).show(5)

print('-'*50)

print('Recommended top items (e.g. 1 top item) for all users with the corresponding predicted ratings:')
model.recommendForAllUsers(1).show(5)
```

Recommended top users (e.g. 1 top user) for all items with the corresponding predicted ratings:

```
+-----+-----+
|movieId| recommendations|
+-----+-----+
| 148| [[197646, 4.665048]]|
| 463| [[29446, 5.262753]]|
| 471| [[197646, 5.436685]]|
| 496| [[141256, 5.210578]]|
| 833| [[268794, 5.78613...]|
```

only showing top 5 rows

Recommended top items (e.g. 1 top item) for all users with the corresponding predicted ratings:

```
+-----+-----+
|userId| recommendations|
+-----+-----+
| 148| [[197646, 4.665048]]|
```

3.4.5. Save model vào HDFS

Save model

```
%pyspark
model.save('hdfs://hadoop-master:9000/model')
```

3.4.6. Đưa ra dự đoán trên tập test

Make predictions on test_data

```
%pyspark
#Let see how the model perform
predictions = model.transform(test_data)
predictions.show()
```

```
+-----+-----+-----+-----+-----+
|userId|movieId|rating|      ts|prediction|
+-----+-----+-----+-----+
|146376|  148|   5.0|836529818|  3.515584|
|264081|  148|   3.0|1220164291| 2.9387372|
| 60382|  148|   4.0| 830077276|  3.406487|
|211963|  148|   3.0| 903702432| 2.3034613|
|196553|  148|   2.5|1498443856|  2.902321|
|209436|  148|   2.0| 844627287| 2.0477366|
| 36198|  148|   5.0| 842004154| 2.9668329|
|111686|  148|   4.0| 877608120| 2.6435108|
| 82425|  148|   3.0| 860111242|  3.117376|
|233502|  148|   4.0| 836930469|   2.7279|
| 76830|  148|   2.0| 842275770| 2.7176235|
|220323|  148|   3.0|1276969740| 2.5964088|
| 47895|  148|   3.0| 840699559| 2.8798094|
|117357|  148|   5.0| 940583818| 3.0361912|
|1100247| 148|   2.0| 826605672| 2.6014761|
```

Took 3 min 17 sec. Last updated by anonymous at November 22 2021, 12:42:25 PM.

```
%pyspark
predictions.printSchema()
```

```
root
|-- userId: integer (nullable = true)
|-- movieId: integer (nullable = true)
|-- rating: double (nullable = true)
|-- ts: integer (nullable = true)
|-- prediction: float (nullable = false)
```

3.4.7. Đánh giá mô hình trên tập test bằng điểm RMSE

Evaluate the predictions

```
%pyspark
# check the root mean squared error
evaluator = RegressionEvaluator(metricName='rmse', predictionCol='prediction', labelCol='rating')
rmse = evaluator.evaluate(predictions)
print('Root mean squared error of the test_data: %.4f' % rmse)
```

Root mean squared error of the test_data: 0.8246

3.4.8. Đưa ra gợi ý cho người dùng mới

Ví dụ: Gợi ý phim cho User có ID: 11

Recommend movie for user (eg: user id 11)

```
%pyspark
# see historical rating of the user
user_history = train_data.filter(train_data['userId']==11)
user_history.show()
```

```
+-----+-----+-----+
|userId|movieId|rating|      ts|
+-----+-----+-----+
|    11|     48|    3.0|1112135389|
|    11|    527|    4.0|1112135526|
|    11|   1193|    4.5|1112135548|
|    11|   1282|    4.5|1112135458|
|    11|   1639|    4.0|1112135379|
|    11|   1722|    3.5|1112135463|
|    11|   1777|    3.5|1112135447|
|    11|   1876|    3.0|1112135455|
|    11|   2054|    2.5|1112135369|
|    11|   2302|    3.5|1112135386|
|    11|   2791|    3.5|1112135339|
|    11|   3623|    3.5|1112135425|
|    11|   3977|    2.5|1112135435|
|    11|   5952|    4.5|1112135382|
|    11|   7261|    5.0|1112135520|
```

Took 1 min 58 sec. Last updated by anonymous at November 22 2021, 12:50:28 PM. (outdated)

```
%pyspark
# a list of movies we are thinking to offer
user_suggest = test_data.filter(train_data['userId']==11).select(['movieId', 'userId'])
user_suggest.show()
```

```
+-----+-----+
|movieId|userId|
+-----+-----+
|    158|     11|
|   2502|     11|
|   3052|     11|
|   4886|     11|
+-----+-----+
```



```
%pyspark
# offer movies with a high predicted rating
user_offer = model.transform(user_suggest)
user_offer.orderBy('prediction', ascending=False).show()
```

```
+-----+-----+-----+
|movieId|userId|prediction|
+-----+-----+-----+
| 2502 | 11 | 4.1108456 |
| 4886 | 11 | 3.9533806 |
| 3052 | 11 | 3.7803314 |
| 158 | 11 | 2.6779819 |
+-----+-----+-----+
```

3.5. Kết quả chạy Job trên Spark Cluster

Spark Master at spark://192.168.56.101:7077

URL: spark://192.168.56.101:7077
 Alive Workers: 3
 Cores in use: 3 Total, 0 Used
 Memory in use: 8.5 GB Total, 0.0 B Used
 Applications: 0 Running, 2 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory
worker-20211121182832-192.168.56.102-46645	192.168.56.102:46645	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)
worker-20211121182832-192.168.56.103-45493	192.168.56.103:45493	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)
worker-20211121182837-192.168.56.101-33997	192.168.56.101:33997	ALIVE	1 (0 Used)	2.8 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20211121204719-0001	Zeppelin	3	1024.0 MB	2021/11/21 20:47:19	hadoopuser	FINISHED	7.7 min
app-20211121193042-0000	Zeppelin	3	1024.0 MB	2021/11/21 19:30:42	hadoopuser	FINISHED	37 min

4. KẾT LUẬN

Qua bài tập lớn lần này, chúng em đã tích lũy được nhiều kỹ năng trong các tác vụ thu thập dữ liệu, trực quan hóa dữ liệu, xử lý dữ liệu, trích xuất các đặc trưng từ dữ liệu, đặc biệt là thao tác trên dữ liệu lớn qua hệ phân tán và thực hiện xử lý thông qua các mô hình lập trình, ứng dụng thường dùng cho xử lý dữ liệu lớn. Bên cạnh đó, chúng em cũng sử dụng các dữ liệu sau khi xử lý để ứng dụng đơn giản cho các mô hình dự đoán điểm đánh giá phim và mô hình gợi ý phim cho người dùng.

Qua quá trình thực hiện, các khó khăn là không thể tránh khỏi. Việc thao tác với dữ liệu lớn như thế nào để có thể thấy được các giá trị tiềm ẩn của dữ liệu là một thách thức lớn. Đặc biệt là việc xây dựng một luồng hoàn chỉnh cho xử lý dữ liệu lớn từ đầu đến cuối (end to end). Nhưng đây cũng là mục đích chính của môn học “Lưu trữ và xử lý dữ liệu lớn”. Chúng em xin chân thành cảm ơn TS. Đào Thành Chung đã tạo điều kiện cho chúng em được làm việc, học tập về lĩnh vực rất quan trọng trong cuộc sống.

Trong tương lai, chúng em sẽ cải thiện các kỹ năng để có thể thu thập được dữ liệu tương tác giữa người dùng với các bộ phim để có thể đánh giá được tốt hơn thái độ và xu hướng, sở thích của người dùng. Từ đó, việc gợi ý phim cũng như dự đoán điểm đánh giá của các bộ phim cũng sẽ trở nên hiệu quả hơn và có thể xây dựng hệ thống gợi ý phim dựa trên luồng dữ liệu streaming nhận được theo thời gian thực.