大数据技术基础综合实验
实验报告

毛子恒　　李康童
2019211397　　2019211408
北京邮电大学 计算机学院

日期：2022 年 6 月 25 日

Part I
基础部分

1　概述

1.1　实验目的

1. 掌握 Lambda 架构

1.2　实验步骤

1. 安装 kafka
2. 安装 Redis
3. 运行推荐系统

2　实验结果及分析

安装 kafka　安装并启动 kafka 的结果如图 1所示。

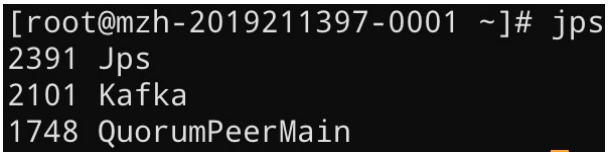

图 1: kafka 启动结果

安装 Redis　安装并启动 redis 的结果如图 2所示。

```
[root@mzh-2019211397-0001 ~]# ps -ef | grep redis
root      6735     1  0 10:48 ?        00:00:00 redis-server *:6379
root      6741  2497  0 10:49 pts/0    00:00:00 grep --color=auto redis
```

图 2: redis 启动结果

运行推荐系统 步骤如下:

1. 启动 HDFS
2. 启动 zookeeper
3. 启动 HBase
4. 配置 HBase Thrift 连接，以便 python 中的 happybase 库能够连接 Hbase (详见一些命令.txt)
5. 在 HBase 中创建对应的表
6. 启动 load_train_ratings_hbase.py（需要运行完）
7. 启动 redis
8. 启动 load_movie_redis.py（需要运行完）
9. 启动 Kafka 并创建 Kafka Topic
10. 启动 generatorRecord.py（这个程序会一直运行，不需要等待停止）
11. 启动 hbase2spark、kafkaStreaming、recommend
12. 启动 recommend_server.py
13. 启动 recommend_client.py

    推荐结果如图 3所示。

```
(py39) xqmmcqs@xqmmcqsdeMBP server-client % python3 ./recommend_client.py 127.0.
0.1 23456
1234
Recommend_List:[userid=1234]
[Movie-1     : Blade Runner (1982)]
[Movie-2     : Forrest Gump (1994)]
[Movie-3     : Fargo (1996)]
[Movie-4     : Pulp Fiction (1994)]
[Movie-5     : Usual Suspects, The (1995)]
[Movie-6     : Alien (1979)]
[Movie-7     : Some Like It Hot (1959)]
[Movie-8     : Donnie Darko (2001)]
[Movie-9     : Gone with the Wind (1939)]
[Movie-10    : Snow White and the Seven Dwarfs (1937)]
```

图 3: 推荐系统运行结果

Part II

选做部分

2.1 实验步骤

1. 使用 Hive/SparkSQL 实现简易后台分析统计
2. 将后台分析统计结果可视化

3 实验结果及分析

启动 Hive

```
hive --service hiveserver2 >/dev/null 2>&1 &
```

创建表 在 Hive 中创建外部表，连接到 HBase 的电影记录表。

```
create external table movie_rating_details(
    key string,
    movieId int,
    rating int,
    times int,
    userId int)
    stored by
    'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    WITH SERDEPROPERTIES ("hbase.columns.mapping" =
    ":key,details:movieId,details:rating,details:timestamp,details:userId")
    TBLPROPERTIES("hbase.table.name" = "movie_records");
```

创建表结果如图 4所示。

```
hive> create external table movie_rating_details(
    > key string,
    > movieId int,
    > rating int,
    > times int,
    > userId int)
    > stored by
    > 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping" =
    > ":key,details:movieId,details:rating,details:timestamp,details:userId")
    > TBLPROPERTIES("hbase.table.name" = "movie_records");
OK
Time taken: 2.17 seconds
```

图 4: Hive 建表结果

表结构如图 5、图 6所示。

```
hive> show tables;
OK
movie_rating_details
wordcount
Time taken: 1.135 seconds, Fetched: 2 row(s)
```

图 5: 表名

分析统计与可视化 编写如下 Python 代码，实现分析统计功能。

```
hive> select * from movie_rating_details limit 20;
OK
1000018415rating        3030    1       1000018415      448
1000085668rating        1097    1       1000085668      45
1000148915rating        1089    1       1000148915      452
1000197419rating        3868    1       1000197419      448
1000320523rating        3224    1       1000320523      410
1000337900rating        1101    1       1000337900      164
1000379642rating        4128    1       1000379642      452
1000406475rating        2797    1       1000406475      367
1000408820rating        4056    0       1000408820      453
1000454489rating        3471    1       1000454489      267
1000511431rating        3466    1       1000511431      367
1000533601rating        50      1       1000533601      42
1000539518rating        3082    1       1000539518      42
1000580416rating        3471    1       1000580416      367
1000587977rating        3695    0       1000587977      42
1000603658rating        1196    1       1000603658      494
1000613298rating        3114    1       1000613298      367
1000646730rating        922     1       1000646730      410
1000652096rating        2067    1       1000652096      238
1000657234rating        4447    1       1000657234      367
Time taken: 1.51 seconds, Fetched: 20 row(s)
```

图 6: 表内容

```python
import sys
import time
from impala.dbapi import connect
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def getArgs():
    argv = sys.argv[1:]
    return argv[0], argv[1], argv[2]

def load_file(filename):
    dataSet = pd.read_csv(filename)
    return dataSet

def hive_connect(host="mzh-2019211397-0001", port=10000):
    return connect(host=host, port=port, database='default')

if __name__ == '__main__':
    host, port, file_path = getArgs()
    conn = hive_connect(host, port)
    df = load_file(file_path)
    movieId2Category = dict()
    count_result = dict()
    for i in df.columns[4:]:
        count_result[i] = 0
    for row in df.iterrows():
```

```
        movieId2Category[row[1]['movieId']] = row[1][4:].to_list()
    plt.ion()
    ax = df.columns[4:].to_list()
    while True:
        count_result = [0] * 19
        time.sleep(1800)
        with conn.cursor() as cursor:
            cursor.execute('SELECT movieId, count(*) FROM movie_rating_details
            ↪  GROUP BY movieId')
            for row in cursor:
                for i, value in enumerate(movieId2Category[row[0]]):
                    count_result[i] += value * row[1]
        print(count_result)
        plt.clf()
        plt.bar(ax,count_result)
        plt.xticks(rotation=90)
        plt.pause(0.001)
```

代码从 Hive 中读取电影的记录，并且根据 movies.csv 中的电影类别信息统计不同类型的电影数目，最后采用 matplotlib 进行可视化，程序每隔半个小时更新图表。

采用以下命令运行：

```
python3 category_count.py mzh-2019211397-0001 10000 "../data/movies.csv"
```

运行结果如图 7所示。

Part III

提高部分

见A节。

4 实验总结

本次实验中我们基于 Lambda 架构搭建起简单的推荐系统，并且实现了简单的后台监控功能。在实验过程中我们对 Lambda 架构、HBase、Spark Streaming 组件的理解更加深刻。

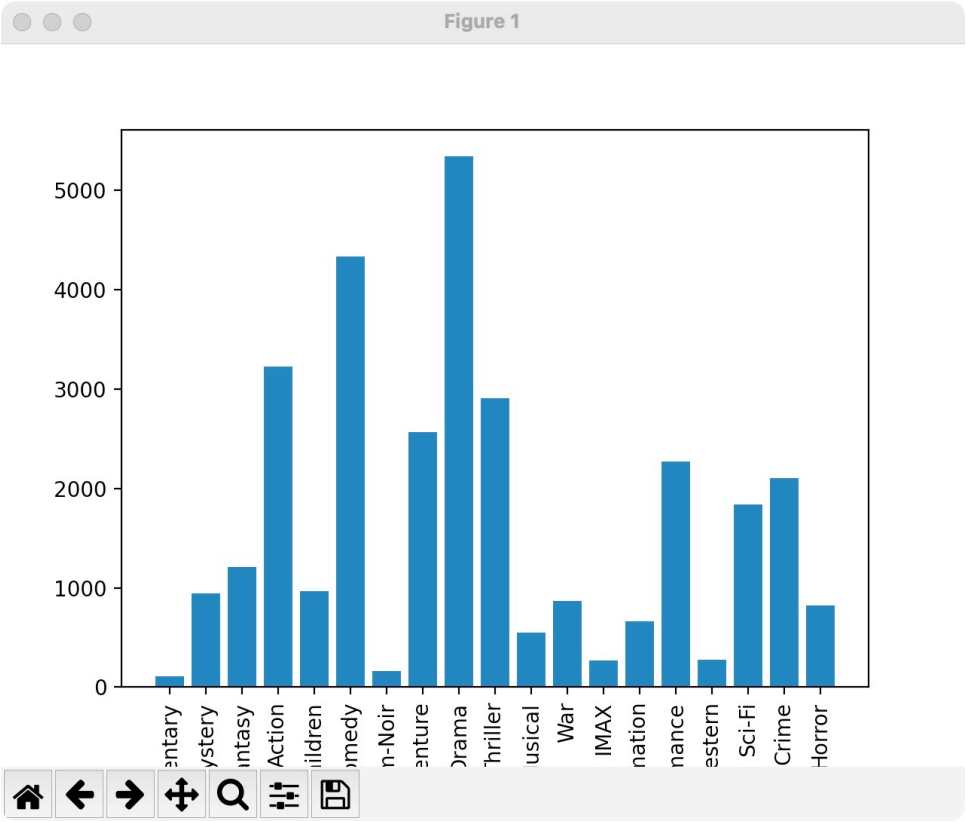

图 7: 电影类别统计

A 大数据框架调研