IMDB Analysis
A Proof of Concept using Big Data and Hadoop

By Sejal Vaidya

Under the guidance of Prof. Venkat Krishnan

Jan 6, 2014

I. Introduction

This project, as a part of Hadoop training, is to demonstrate the ability and usability of the Hadoop framework for analysing data of large volumes and across different varieties, at high velocities (Big Data).

The Internet Movie Database (abbreviated IMDb) is an online database of information related to films, television programs, and video games, including cast, production crew, fictional characters, biographies, plot summaries, trivia and reviews. As of January 18, 2015, IMDb had 3,156,631 titles (includes episodes) and 6,329,543 personalities in its database.
Hadoop, here, has been utilized to perform analysis on the millions of rows within IMDB's datasets on movies to:

1. Analyse the data corpus to find the top 100 movies as ranked by users in IMDB
2. Summarize data further on the basis of ratings, users, genres etc.
3. Load the summarized results to RDBMS

II. Solution Requirements

The solution to gather required metrics will be developed with the use of Hadoop API on a system with Linux/Unix operating system. The solution will demonstrate key features of Hadoop API, such as,

1. Map Reduce for data transformations and parsing of the data
2.
    a. Hive for sorting and finding records with top values.
    b. Pig for filtering and creating subsets of data

3. Sqoop for exporting the processed records back to MySQL

III. Architecture

Operating System	Standalone ubuntu-server-12.04 running in VMWare on Windows 7, 64bit
Number of Processors	2
Physical Memory	8GB.
RAM Allocated for Ubuntu	3.5GB
IDE	Eclipse
Java	1.6
Hadoop Release	Hadoop-1.0.3
Other Tools	Putty, WinSCP

IV. Map Reduce

Objective

Analyse the data corpus to find the top 100 movies as ranked by users in IMDB

Source

Dataset URL: ftp://ftp.fu-berlin.de/pub/misc/movies/database/movies.list.gz

No. of rows: 149011196 Dataset size: 142 MB (after extraction)

Data Structure: Each line in a file represents a single movie record containing following fields - movie title, year, imdb rank, rating distribution, total user votes. Headers, footers and any summary information of the dataset needs to be removed. Dataset contains Tabs/Multiple Spaces as delimiter.

Environment and Approach
Hadoop, Java, Design Patterns, Regex

Solution
Jar: imdb.jar, Package: IMDBLab

Execution
1. Copy the source dataset to HDFS: /input/movies.list
2. Copy imdb.jar file to "programs" directory in HDFS
3. Execute the following, from the "programs" directory:
   ```
   hadoop jar imdb.jar imdb.top100.Top100 /input/movies.list /output/res1
   ```

Output
```
hadoop fs -cat /output/res1/part-r-00000
```

V. Hive

Objective
Summarize data on the basis of ratings, users, genres etc.:

1. Using Hive script, find top 10 average rated "Action" movies with descending order of rating
2. Using Hive script, List all the movies with its genre where the movie genre is Action or Drama and the average movie rating is in between 4.4 - 4.7 and only the male users rate the movie
3. Using Hive script and movies dataset:
   1. create one **table dynamically partitioned** by genre and picking data from 'movies' table
   2. list all the partitions created
   3. create a Hive query that selects all columns from the table for the virtual column genre of value= '(no genres listed)'
4. Create three tables that have three columns each (MovieID, MovieName, Genre). Each table will represent a year. The three genres are `SciFi`, `Fantasy` and `Thriller`. Using **Hive multi-table insert**, insert values from the table you created in (3) to these three tables (each table should have names of movies e.g., `movies_SciFi` etc. for the specified year)

Source
Dataset URL: http://grouplens.org/datasets/movielens -> movies.dat, ratings.dat, users.dat

No. of rows: 500835 (movies), 235105473 (ratings), 110208 (users)

Dataset size: 490KB (movies), 233 MB (ratings), 114KB (users), after extraction

Data Structure:

movies_new:	MovieID~Title~Genres
ratings_new:	UserID~MovieID~Rating~Timestamp
users_new:	UserID~Gender~Age~Occupation~Zip-code

Dataset contains tilde (~) as delimiter.

Environment and Approach
Hive, HDFS

Solution
The following scripts provide the solutions to the above problem statements:

1. hw3q5.sql
2. hw3q6.sql
3. hw3q7.sql
4. hw3q8.sql

Execution

1. Copy the source dataset to HDFS: ~/lab/data/movies.dat, ~/lab/data/ratings.dat, ~/lab/data/users.dat
2. Create database 'imdb' in hive prompt
3. Copy hive scripts (.sql files mentioned in solutions) to "programs" directory in HDFS
4. Execute the scripts, at the hadoop prompt:

```
$ hive -f /home/notroot/lab/programs/hw3q5.sql
$ hive -f /home/notroot/lab/programs/hw3q6.sql
$ hive -f /home/notroot/lab/programs/hw3q7.sql
$ hive -f /home/notroot/lab/programs/hw3q8.sql
```

Output

To see the partitions (created dynamically in hw3q7.sql):

```
$ hadoop fs -ls /user/hive/warehouse/imdb.db/movies_partition
```

To see the output, execute the following at the hadoop prompt:

```
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/5/000000_0
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/6/000000_0
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/7/000000_0
```

For the output of problem 4 (hw3q8.sql), go to the hive prompt and execute the following:

```
hive> use imdb;
hive> select * from movies_SciFi;
hive> select * from movies_Fantasy;
hive> select * from movies_Thriller;
```

VI. Pig

Objective

Summarize data on the basis of ratings, users, genres etc.:

1. Using Pig Latin script, list the unique userid of female users whose age between 20 - 30 and who have rated the highest rated Action and War movies. Hint: Consider average rating to calculate the highest rated movies. While finding the Action and War movies, you should count all users not only the female users

2. Using Pig Latin script, Apply **cogroup** command on MovieID for the datasets ratings_new and movies_new

3. Repeat Question 2 (implement join) with cogroup commands

Source

Dataset URL: http://grouplens.org/datasets/movielens -> movies.dat, ratings.dat, users.dat

No. of rows: 500835 (movies), 235105473 (ratings), 110208 (users)

Dataset size: 490KB (movies), 233 MB (ratings), 114KB (users), after extraction

Data Structure:

movies_new:	MovieID~Title~Genres
ratings_new:	UserID~MovieID~Rating~Timestamp
users_new:	UserID~Gender~Age~Occupation~Zip-code

Dataset contains tilde (~) as delimiter

Environment and Approach

Pig, HDFS

Solution

The following scripts provide the solutions to the above problem statements:

1. hw3q1.pig
2. hw3q2.pig
3. hw3q3.pig

Execution

1. Copy the source dataset to HDFS: /input/movies.dat, /input/ratings.dat, /input/users.dat
2. Create database 'imdb' in hive prompt
3. Copy pig scripts (.pig files mentioned in solutions) to "programs" directory in HDFS
4. Execute the scripts, at the hadoop prompt:

```
$ pig -x mapreduce /home/notroot/lab/programs/hw3q1.pig
$ pig -x mapreduce /home/notroot/lab/programs/hw3q2.pig
$ pig -x mapreduce /home/notroot/lab/programs/hw3q3.pig
```

Output

Execute the following at the hadoop prompt:

```
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/1/part-r-00000
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/2/part-r-00000
$ hadoop fs -cat /user/hive/warehouse/imdb.db/out/3/part-r-00000
```

VII. Sqoop

Objective

Export the processed records from Hive 3.ii to a table in MySQL

Source

Data Structure: MovieID~Title~Genres
Dataset contains tilde (~) as delimiter

Environment and Approach

MySQL, HDFS, Sqoop

Execution

1. In MySQL, create database imdb, and table movies_nogenre:

```
mysql> create database imdb;
mysql> use imdb;
mysql> create table movies_nogenre (
        movieid varchar(8) not null,
        title varchar(35) not null,
        genre varchar(15) not null,
        primary key (movieid));
```

2. Execute the following, , at the hadoop prompt, to export the records to the MySQL table:

```
$ sqoop export --connect jdbc:mysql://localhost/imdb --table movies_nogenre --username root --password
<password> --export-dir /user/hive/warehouse/imdb.db/out/7/000000_0
```

Output

For the output, go to the MySQL prompt and execute the following:

```
mysql> use imdb;
mysql> select * from movies_nogenre;
```