COEN 242 Big Data
Programming Assignment

Naimisha Tummu (W1411263) & Chitra Elangovan (W1284522)

Dataset location

Small dataset

/HADOOP/hdfs/user/bigdata11/dataset/movies/movies.csv

/HADOOP/hdfs/user/bigdata11/dataset/reviews/reviews.csv

Large dataset

/HADOOP/hdfs/user/bigdata11/dataset_large/movies/movies_large.csv

/HADOOP/hdfs/user/bigdata11/dataset_large/reviews/reviews_large.csv

Hive

Database : imdb_bigdata11

Table:

Small dataset - movies , reviews

Large dataset - movies_large , reviews_large

50M dataset - movies_50m, reviews_50m

300M dataset - movies_300m, reviews_300m

Create Table Scripts

Movies table

For inserting data into the movies table from the csv file we made use of **CSV SerDe** to delimit the fields correctly. SerDe takes care of commas inside the string and reads the file as expected. Since CSVSerDe creates only string fields, we loaded a temporary table first using CSVSerDe and inserted the data into the final table from the temporary table casting the columns as required.

1) Temp table(my_table) for loading the values of movies CSV file

```
CREATE TABLE my_table(movieid int, title string,genres string)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE
tblproperties("skip.header.line.count"="1");
```

2) Loading data to temp table

load data local inpath "/HADOOP/hdfs/user/bigdata11/dataset/movies/movies.csv" overwrite into table imdb_bigdata11.my_table;

3) Creating movies table

```
CREATE TABLE movies (movieid INT, title STRING,genres STRING);
```

4) Inserting data to movies table from the temp table

```
insert into movies
select
CAST(movieid as INT),
title,
genres
from my_table;
```

5) Dropping the temp table after creating movies table

```
drop table my_table;
```

Reviews table

1) Creating reviews table

```
CREATE TABLE reviews (userid INT,movieid INT, rating DOUBLE,timestamp
BIGINT) row format delimited fields terminated BY ','
tblproperties("skip.header.line.count"="1");
```

2) Loading data to reviews table

```
load data local inpath
"/HADOOP/hdfs/user/bigdata11/dataset/reviews/reviews.csv" overwrite into
table imdb_bigdata11.reviews_large;
```

Similarly, we created the tables for large, 50m and 300m dataset. After table creation, we ran the hive query files for the different datasets.

Hive Query Files :

Dataset	**Query 1**	**Query 2**
Small	query1.hql	query2.hql
50M	query1_50m.hql	query2_50m.hql
300M	query1_300m.hql	query2_300m.hql
Large	query1_large.hql	query2_large.hql

Mapreduce

For Mapreduce program, we have used Mapside join to join the content of Movies and Reviews files based on movieid. Mapside join is effective when one of the two files can fit entirely in the memory. We used DistributedCache method to add the smaller file (movies.csv) to the cache of the node where the mapper is being executed.

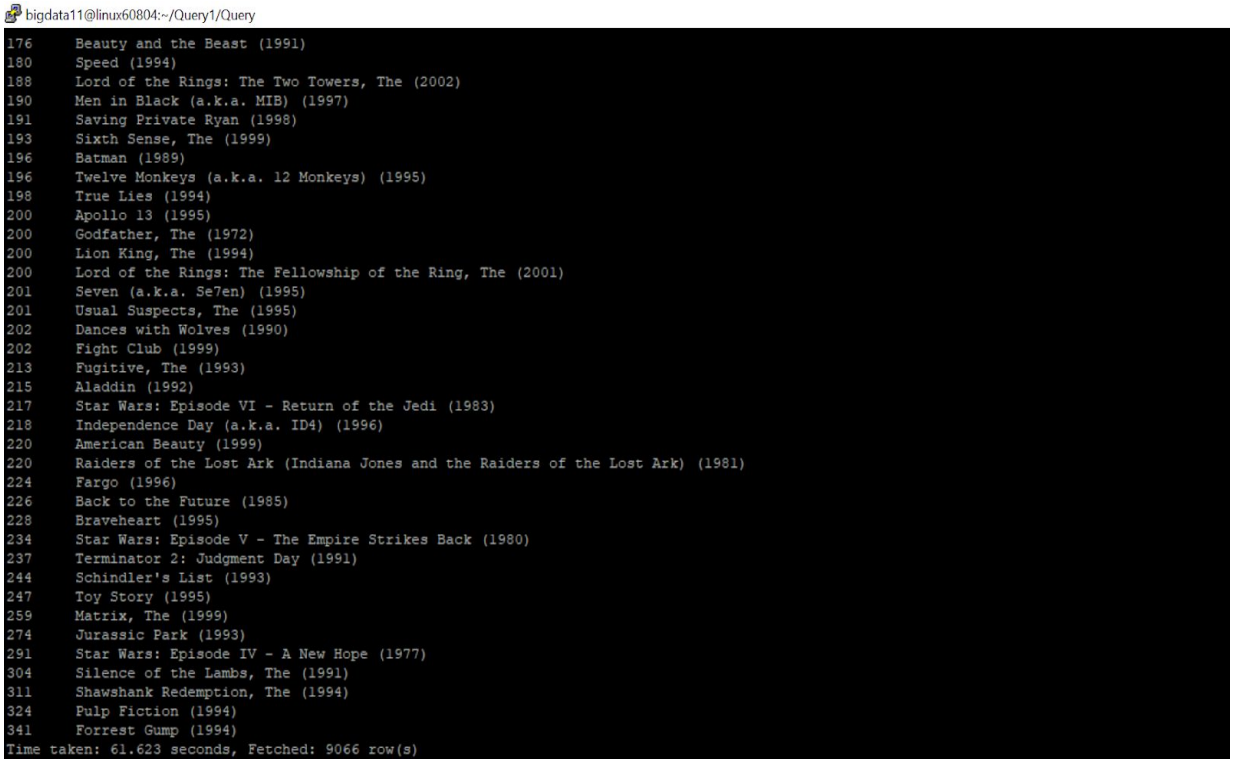
Logic for handling commas in the movie title - Each line in the file is read line by line skipping the header alone. The lines were then split based on commas and stored in an String array. The 0th element of the array denotes the movieId and all other elements of the array except the last ones are concatenated to form the movie title. This logic is handled by code within the Mapreduce program.

Commands & Screenshot of Output

<u>Part 1</u>

Hive

1) Small dataset
   hive -f query1.hql > output_query1.txt



2) Large dataset

hive -f query1_large.hql > output_query1_large.txt

bigdata11@linux60804:~/Query1/Query

```
45303   Lion King, The (1994)
45413   Gladiator (2000)
45544   Speed (1994)
49643   Sixth Sense, The (1999)
50168   True Lies (1994)
50375   Aladdin (1992)
50809   Saving Private Ryan (1998)
51338   Dances with Wolves (1990)
51837   Lord of the Rings: The Return of the King, The (2003)
51882   Lord of the Rings: The Two Towers, The (2002)
52474   Fargo (1996)
52658   Seven (a.k.a. Se7en) (1995)
53398   Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
53717   Batman (1989)
54783   Back to the Future (1985)
56820   Fugitive, The (1993)
56827   Lord of the Rings: The Fellowship of the Ring, The (2001)
57070   Godfather, The (1972)
57232   Independence Day (a.k.a. ID4) (1996)
57416   Apollo 13 (1995)
57879   American Beauty (1999)
59271   Usual Suspects, The (1995)
59693   Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
60024   Fight Club (1999)
61672   Star Wars: Episode V - The Empire Strikes Back (1980)
61836   Terminator 2: Judgment Day (1991)
62714   Star Wars: Episode VI - Return of the Jedi (1983)
66008   Toy Story (1995)
66512   Braveheart (1995)
67662   Schindler's List (1993)
74355   Jurassic Park (1993)
77045   Star Wars: Episode IV - A New Hope (1977)
77960   Matrix, The (1999)
84078   Silence of the Lambs, The (1991)
87901   Pulp Fiction (1994)
91082   Shawshank Redemption, The (1994)
91921   Forrest Gump (1994)
Time taken: 66.632 seconds, Fetched: 45115 row(s)
```

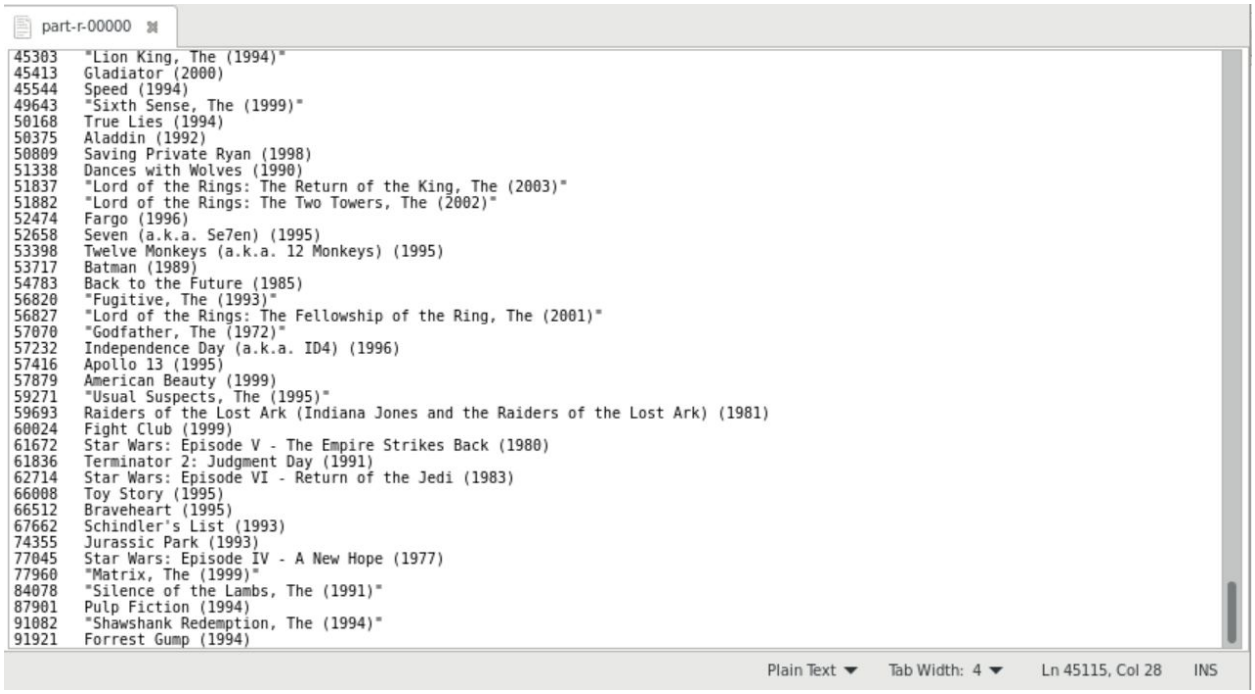
MapReduce

1) Small dataset

hadoop jar MovieRank.jar MovieRank 1 ./dataset/movies/movies.csv ./dataset/reviews/reviews.csv ./output_small1

2) Large dataset

hadoop jar MovieRank.jar MovieRank 1 ./dataset_large/movies/movies_large.csv ./dataset_large/reviews/reviews_large.csv ./output_large1



Part 2

Hive

1) Small dataset

hive -f query2.hql > output_query2.txt

bigdata11@linux60804:~/Query2/Query

```
Blood Simple (1984)     4.291666666666667       24
Smoke (1995)    4.291666666666667       24
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)   4.294871794871795       39
City of God (Cidade de Deus) (2002)     4.297101449275362       69
Treasure of the Sierra Madre, The (1948)        4.3     30
Schindler's List (1993) 4.30327868852459        244
12 Angry Men (1957)     4.304054054054054       74
Conversation, The (1974)        4.304347826086956       23
Rear Window (1954)      4.315217391304348       92
Central Station (Central do Brasil) (1998)      4.318181818181818       11
Happiness (1998)        4.326086956521739       23
Chinatown (1974)        4.3355263157894735      76
Raging Bull (1980)      4.35    50
Philadelphia Story, The (1940)  4.351351351351352       37
Modern Times (1936)     4.359375        32
Rush (2013)     4.363636363636363       11
Lifeboat (1944) 4.363636363636363       11
Paths of Glory (1957)   4.366666666666666       15
Usual Suspects, The (1995)      4.370646766169155       201
It Happened One Night (1934)    4.38    25
Godfather: Part II, The (1974)  4.385185185185185       135
Band of Brothers (2001) 4.386363636363637       22
Maltese Falcon, The (1941)      4.387096774193548       62
Roger & Me (1989)       4.392857142857143       42
Shall We Dance (1937)   4.409090909090909       11
Mister Roberts (1955)   4.411764705882353       17
African Queen, The (1951)       4.42    50
Ran (1985)      4.423076923076923       26
All About Eve (1950)    4.434210526315789       38
When We Were Kings (1996)       4.4375  16
On the Waterfront (1954)        4.448275862068965       29
Gladiator (1992)        4.454545454545454       11
Tom Jones (1963)        4.458333333333333       12
Shawshank Redemption, The (1994)        4.487138263665595       311
Godfather, The (1972)   4.4875  200
Inherit the Wind (1960) 4.541666666666667       12
Best Years of Our Lives, The (1946)     4.636363636363637       11
Time taken: 61.935 seconds, Fetched: 287 row(s)
```

2) Large dataset

hive -f query2_large.hql > output_query2_large.txt

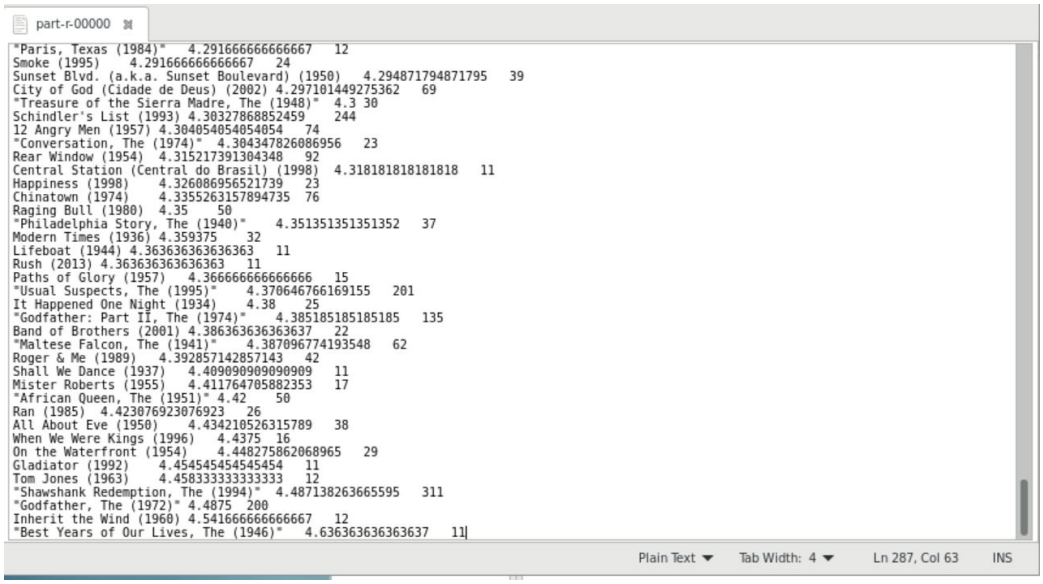
bigdata11@linux60804:~/Query2/Query

```
Goodfellas (1990)       4.17828875746609        33987
Yojimbo (1961)  4.180264741275572       4155
Guten Tag, Ramón (2013) 4.181818181818182       11
Dark Knight, The (2008) 4.182070707070707       39600
Still Bill (2009)       4.1875  16
City of God (Cidade de Deus) (2002)     4.187872863087181       19947
O Auto da Compadecida (Dog's Will, A) (2000)    4.191558441558442       154
Lives of Others, The (Das leben der Anderen) (2006)     4.199038891372374       8948
A Silent Voice (2016)   4.2     20
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)   4.200819672131147       7930
Spirited Away (Sen to Chihiro no kamikakushi) (2001)    4.202589307120594       20855
Double Indemnity (1944) 4.202603887997147       5607
Paths of Glory (1957)   4.20264575040974        4271
North by Northwest (1959)       4.205228001893441       19013
Third Man, The (1949)   4.209418968212611       7676
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)     4.213030410183875       28280
Casablanca (1942)       4.2143927037912325      30043
The Blue Planet (2001)  4.217948717948718       273
Over the Garden Wall (2013)     4.219745222929936       157
Whiplash (2013) 4.226775956284153       183
One Flew Over the Cuckoo's Nest (1975)  4.22913497743311        40103
Fight Club (1999)       4.2307160469145675      60024
12 Angry Men (1957)     4.231208570075758       16896
Rear Window (1954)      4.232552144363722       21335
Seven Samurai (Shichinin no samurai) (1954)     4.255073602972702       13994
Godfather: Part II, The (1974)  4.263475012950189       36679
Human (2015)    4.264705882352941       34
Schindler's List (1993) 4.266530696698294       67662
Human Planet (2011)     4.271573604060913       197
Usual Suspects, The (1995)      4.300188962561792       59271
O Pátio das Cantigas (1942)     4.3076923076923075      13
Welfare (1975)  4.318181818181818       11
Godfather, The (1972)   4.339810758717364       57070
Planet Earth II (2016)  4.352631578947369       95
Band of Brothers (2001) 4.394366197183099       284
Shawshank Redemption, The (1994)        4.429014514393623       91082
Planet Earth (2006)     4.478779840848806       754
Time taken: 77.745 seconds, Fetched: 381 row(s)
```

Mapreduce

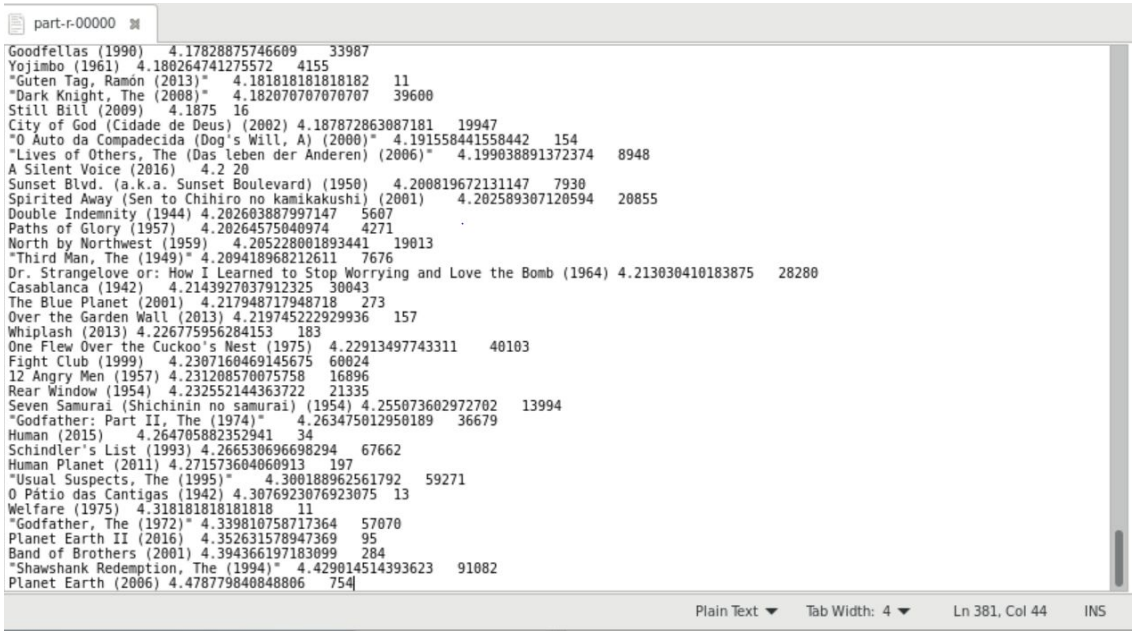
1) Small dataset

hadoop jar MovieRating.jar MovieRating 1 ./dataset/movies/movies.csv ./dataset/reviews/reviews.csv ./output_small2



2) Large dataset

hadoop jar MovieRating.jar MovieRating 1 ./dataset_large/movies/movies_large.csv ./dataset_large/reviews/reviews_large.csv ./output_large2



Part 3

Hive vs MapReduce

For both Hive and Mapreduce, 2 jobs are launched. However, with the increase in the size of dataset, hive is taking more time compared to Mapreduce because of the internal conversion from hive query to Mapreduce jobs and higher level of abstraction.

Analysis

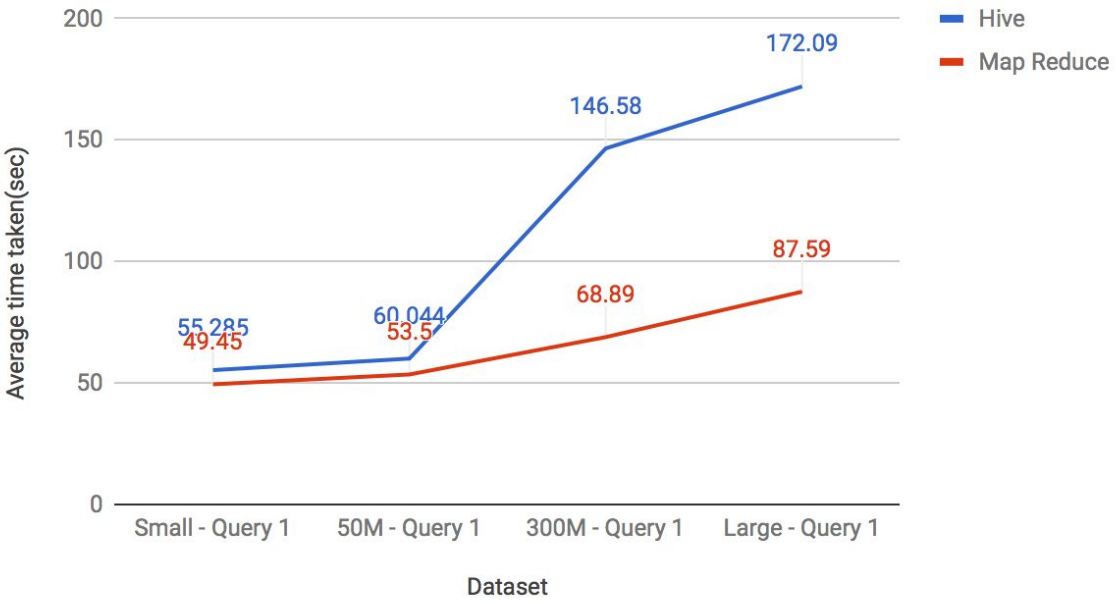
The time mentioned for hive is the total time taken that is displayed in the output of the hive query. However, for mapreduce, making note of the total time(start time and end time difference) through java code is not very helpful as most of the time taken is for connecting to the resource manager(varying from 2 secs to 2 minutes and is mostly random).
As a fix, we ignored the time taken to connect to the system. Instead, we recorded the clock time displayed in the output after getting connected to resource manager. We added 10 secs to the execution time as the connecting time to resource manager.

Below times are the average of 5 runs for both hive and mapreduce.

	Hive		MapReduce	
Dataset	**Number of Jobs**	**Time taken (sec)**	**Number of Jobs**	**Time Taken (sec)**
Small - Query 1	2	55.285	2	49.45
50M - Query 1	2	60.044	2	53.5
300M - Query 1	2	146.58	2	68.89
Large - Query 1	2	172.09	2	87.59
Small - Query 2	2	46.078	2	45.42
50M - Query2	2	66.599	2	58.65
300 - Query 2	2	67.174	2	71.5
Large - Query 2	2	91.591	2	77.95

Hive took less lines of code and involved less development effort compared to the Mapreduce. So from our analysis, we notice that mapreduce is efficient with respect to performance when compared to Hive. In our case, there is no significant difference between mapreduce and hive for small dataset and for part 2 because, hive is converted into highly optimised mapreduce queries while execution which is difficult for our mapreduce code to compete with. But as the size of the data increases, mapreduce gives the best performance.

Hive vs Map Reduce - Part 1



Hive vs Map Reduce - Part 2

Part 4

We did the performance analysis for part 1(movie ranking) by manipulating the number of reducers and mappers in the mapreduce code.

Analysis

The difference between the start and the end time programmed in the java code may not be an ideal way for calculating the execution time of mapreduce as the connectivity to resource manager is not stable and is taking random times.

As an alternative, we came up with 2 approaches -

1. We are calculating the clock time displayed in the output from the point it is connected to the resource manager(and adding 10 secs as the connection time).
2. We can use the Job counters -

Total vcore-milliseconds taken by all map tasks(same as Total time spent by all map tasks (ms))
Total vcore-milliseconds taken by all reduce tasks(same as Total time spent by all reduce tasks (ms))

Vcore stands for virtual core in CPU of a computer. VCORE-MILLIS-MAPS measures the cpu resources used by all the mappers. It is the aggregated number of vcores that each mapper had been allocated times the number of seconds that mapper had run. The number of vcores allocated to a mapper is 1 by default. Similarly, VCORE-MILLIS-REDUCES gives the sum of time taken by all the reducers together.

With the increase in the number of mappers and reducers, the value of the counters VCORE-MILLIS-MAPS and VCORE-MILLIS-REDUCES also increases but the overall execution time of the job decreases as all the mapper and reducer instances run in parallel.

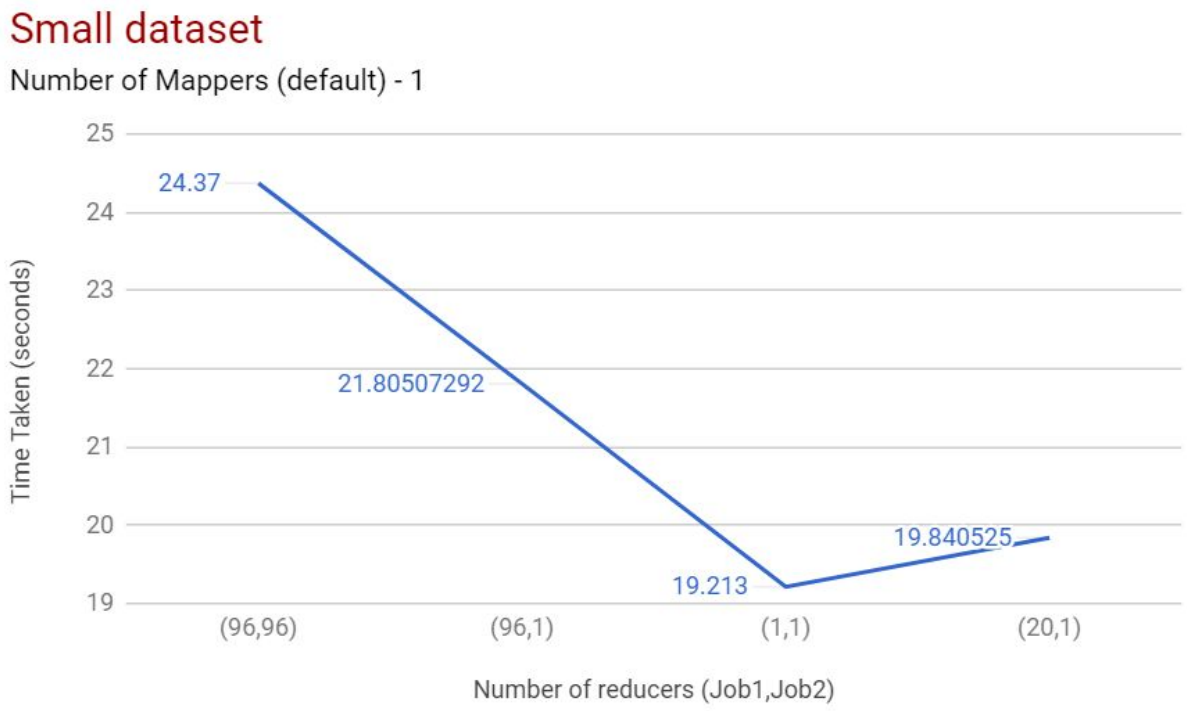
In our case, for calculating the time taken by mapreduce code for movie ranking, we divided the VCORE-MILLIS-MAPS time and VCORE-MILLIS-REDUCES time with the number of mappers and reducers used respectively for that job.

Using these counters gives us the exact time taken for the map and reduce operations for both job1 and job2 without taking the setup time into consideration. Since we want to analyse the time taken with different combinations of mappers and reducers, taking just the map and reduce time for comparing will be the best solution.

Modifying the number of reducers

For modifying the number of reducers for job1 and job2, we are sending the values through command line and setting the reducers in the code using the below method-

job.setNumReduceTasks()

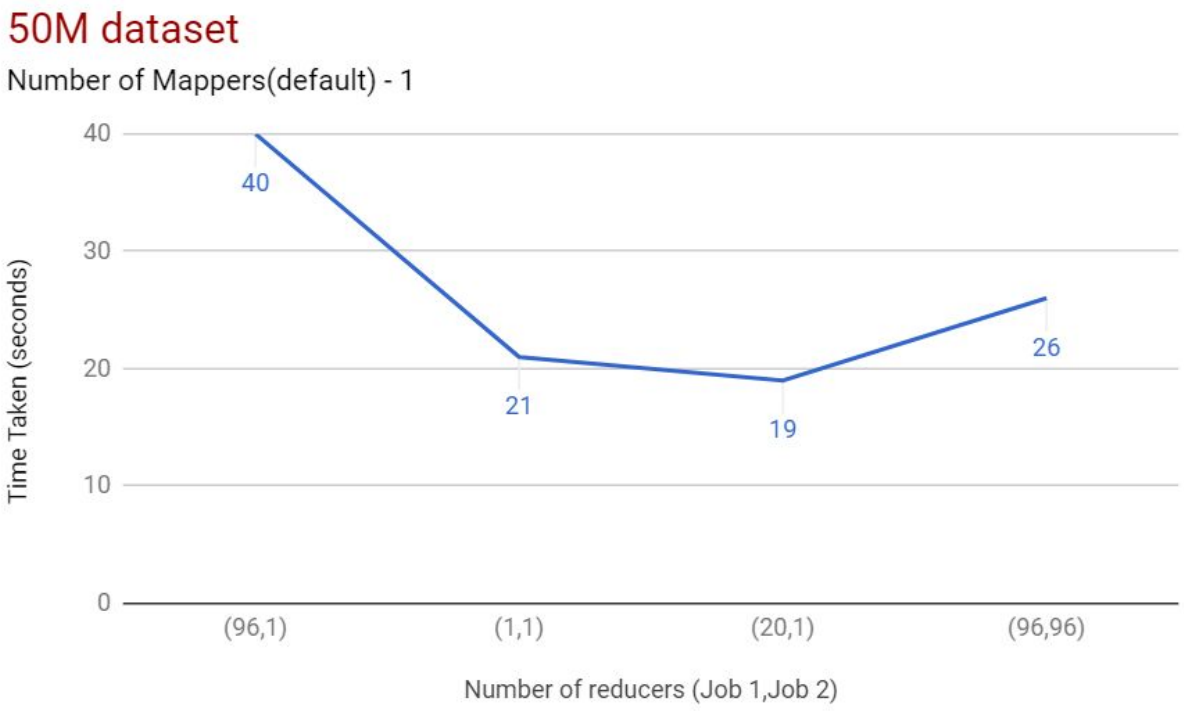


The size of the smaller dataset(reviews,csv) is 2.4MB. The default number of input splits(mappers) for the small dataset is 1.

The default number of reducers for job1 and job2(96,96) took maximum time. With the decrease in the number of reducers for job1 and job2, the time taken is also decreasing. Again, increasing the number of reducers for job1 is increasing the execution time. Therefor**e, the minimum time is obtained when the reducers for job1 and job2 are set to (1,1).**

50M dataset

The behaviour of 50M dataset is similar to the small dataset. The time taken started decreasing with the decrease in number of reducers and the best performance is obtained at the value of 20,1 reducers for job1 and job2 respectively.



Modifying the number of mappers and reducers

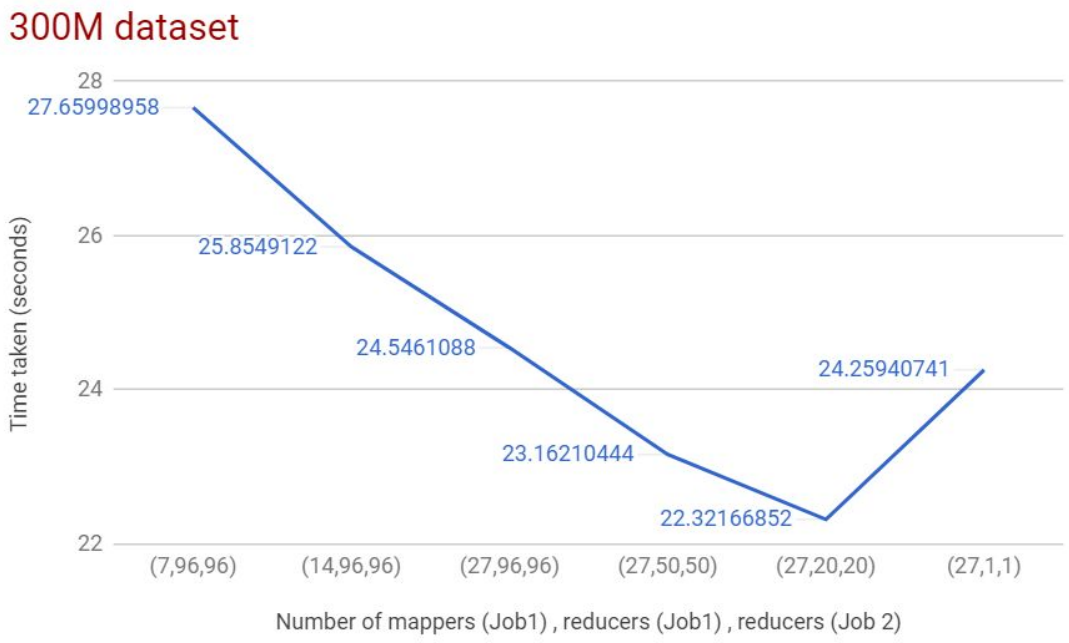
For 300M and large datasets, the number of input splits is greater than 1.
We tried to modify the split size and reducer count(both job1 and job2) for these datasets to arrive at the best possible combination of mappers and reducers. The number of mappers for the second job is determined by the system itself.
For modifying the number of mappers, we are changing the input split size in the code using
FileInputFormat.setMaxInputSplitSize(job,size)
FileInputFormat.setMinInputSplitSize(job,size)

300M Dataset

For 300M dataset, the default number of mappers and reducers for job1 and job2 are (3,96) and (96,96) respectively.We have tried the split size of 128(default), 100, 50, 25, and 12.5MB. Smaller split size results in more number of mappers.

With the increase in the number of mappers and same number of reducers, the time taken started decreasing. The performance improvement is almost saturated when the split size is **12.5MB(27 mappers).** Decreasing the split size further is not improving the performance. We then started decreasing the number of reducers from 96 to 1. The performance further

improved when reducers are decreased from 96 and 20. The below graph shows that the best performance is obtained for the combination o**f 27,20 mappers and reducers**.



Large Dataset

For large dataset, the default number of mappers and reducers for job1 and job2 are (6,96) and (96,96) respectively.

We have tried the split size of 128MB(default), 50, 25, and 12.5MB. Smaller split size results in more number of mappers.

With the increase in the number of mappers and same number of reducers, the time taken started decreasing. The performance improvement is almost saturated when the split size is 25MB(29 mappers) and there is no significant decrease in time by decreasing the split size further. So, we started decreasing the number of reducers from 96. The time taken is almost constant until the reducer count is decreased to 20 and then, it started increasing.

So, the best performance is obtained for the range of combinations from (29,96) to (57,20) mappers and reducers.