**Project 3**

**Map/Reduce programming exercise and analysis**

(Course: Database Models & Implementation Techniques, CSE:5331-001)

*Submitted By,*

Group: 23

Group members:

Krishna Sankaramani Ramamoorthie

(UTA ID: 1001980344)

Archana Prakash Nikam

(UTA ID: 1001912520)

*Under the guidance of,*

Prof. Sharma Chakravarthy

## TA: Billah, Umme Hafsa

T**able of Contents**

1. Introduction

2. Overall status of the project

3. Encountered Difficulties

4. File Description

5. Division of Labor

6. Handling of logical errors

7. Analysis of main problem

8. Analysis of bonus problem

9. References

**1. Introduction**

One of the advantages of cloud computing is its ability to deal with very large data sets and still have a reasonable response time. Typically, the map/reduce paradigm is used for these types of problems in contrast to the RDBMS approach for storing, managing, and manipulating this data. An immediate analysis of a large data set does not require designing a schema and loading the data set into an RDBMS.

We have used Hadoop Map/Reduce software framework for writing applications that process vast amounts of data in parallel on large clusters. In this project, we have used the Map/Reduce program to list the names of people, who have directed and acted in the same IMDb title/category of any genre and output title, director, actor, genre, and year.

2. Overall status of the project:

In this project, we have implemented 3 mappers 1 reducer, and 3 mappers 2 reducers. The mapper takes input as a key, and value pair and produces a set of intermediate key-value pairs. Then the mapper groups together all the intermediate values associated with the intermediate key pairs.

The results from the mapper are sent to the reducer by shuffling the intermediate key-value lists. Then the reducer takes the key, value lists sets, and produces the result of the desired job.

Below are the classes implemented,

Class ActorsMapper()

Class CrewMapper()

Class TitlesMapper()

Class ImdbReducer()

Class MainRunner()

3. File description

Our goal was to successfully execute the Map/Reduce program to list the names of people, who have directed and acted in the same IMDb title/category of any genre and output title, director, actor, genre, and year. These class operations are implemented as below. An explanation of each class is given below.

* **Class ActorsMapper(Key, Value, Context):**

In this class first we preprocess the dataset by splitting the semicolons, using split() and trim(). Then we prefix actor data with “actorData:” to identify the actor data in the reducer class. Finally, we export the actor data from the mapper with titleId as key and actor data as value. This key, value pair will then be sent to the reducer.

* **Class CrewMapper(Key, Value, Context):**

In this class first we preprocess the dataset by splitting the semicolons, using split() and trim(). Then we prefix director data with “directorId:” to identify the director data in the reducer class. Finally, we export the director data from the mapper with titleId as key and directorId as value. This key, value pair will then be sent to the reducer.

* **Class TitleMapper(Key, Value, Context):**

For our team the given parameter is tvEpisode. We first preprocess the dataset by splitting the semicolons, using split() and trim(). Then we populate titleType only if it's from one of the title types which tvEpisode that we are searching for. If the year has \N in it we replace it with 0 and we will remove all occurrences of \N with or without commas for genres. We will export the title data only if it matches title types tvEpisode and between the years 1997 to 2007 (inclusive). Titles data will be exported from the mapper with titleId as key and title data as value.

* **Class ImdbReducer(titleId, mappedData, Context):**

First we loop over all the entries received for the key of titleId from 3 mappers. Then we will check if the data received is from TitleMapper or ActorMapper or CrewMapper based on the prefix values which we have assigned before. Since there can be multiple directors for the title, we split director Id on comma delimiter and put all director Ids in a list.. So, for every directorId received, we will check if that Id is also present in the actors list for that title. If it's present then the director is also acting in that movie title. We need to write data to output files only if the actor and director are the same who is the title. We will print separate rows for each file if there is more than one director is found acting in the title.

* **Class MainRunner(args):**

In MainRunner class we will initialize the reducer value as 1 and 2. We will call 3 mapper classes by passing the input arguments to those classes and the reducer class.

* **Bonus**

for the bonus part we have just increased the number of mappers for each input by adjusting the splitsize in job configuration. and increased the reducers as well. Other aspects of the code will be the same as the main code. Please refer below Result Analysis section for more details.

4. Encountered difficulty:

One of the laborious tasks in our project was figuring out datasets and understanding the information in them. Running the file in expanse was quite challenging as we need to check each time we run the program. It was challenging and confusing because there were several executions. It took some time for us to begin getting the expected output that we desired.

5. Division of labor:

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Name** | **Task worked on** |
| 1 | Sankaramani Ramamoorthie, Krishna | 1. Implemented three mapper class for main problem  2. Worked on configuration of bonus problem  3. Worked on testing and documentation part  4. Worked on sql queries |
| 2 | Archana Prakash Nikam | 1. Implemented reducer class and MainRunner as Main class.  2. Worked on splitsize calculation for multiple mappers  3. Worked on debugging and testing on expanse.  4. Worked on bonus problem |

6. Handling of logical errors:

We encountered few logical errors while running program in the expanse like input path with the build file, local & dist.run files. We have handled those errors by putting correct input file paths.

In addition, for bonus problem we face some errors with split size for multiple mappers. We overcome this error with help of help ppt. We followed the given configuration.

7. Analysis for main problem

* **Result verification :**

- **Asssigned params – year range : 1997-2007 and titletype: tvEpisode**

- We verified the SQL query results with Hadoop MapReduce program output for the given parameters and we got the exact same count. **Count : 8858**

Text

Description automatically generated with low confidence

**SQL query for result :**

SELECT TB.primarytitle ||','|| TB.genres ||','|| NB.primaryname ||','|| TB.startyear

FROM imdb00.title\_basics TB, imdb00.name\_basics NB, imdb00.title\_crew TC, imdb00.title\_principals TP

WHERE TB.TCONST = TC.TCONST

AND TB.TCONST = TP.TCONST

AND TC.TCONST = TP.TCONST

AND INSTR(TC.Directors,TP.nconst) > 0

AND TP.nconst = NB.nconst

AND TB.startyear <> '\N'

AND TC.Directors <> '\N'

AND TP.nconst <> '\N'

AND 1997 <= CAST(TB.startyear AS NUMERIC) AND CAST(TB.startyear AS NUMERIC) <= 2007

AND TB.titletype = 'tvEpisode'

AND (lower(TP.category) = lower('actor') OR lower(TP.category) = lower('actress'));

**SQL query for count :**

SELECT count(1)

FROM imdb00.title\_basics TB, imdb00.name\_basics NB, imdb00.title\_crew TC, imdb00.title\_principals TP

WHERE TB.TCONST = TC.TCONST

AND TB.TCONST = TP.TCONST

AND TC.TCONST = TP.TCONST

AND INSTR(TC.Directors,TP.nconst) > 0

AND TP.nconst = NB.nconst

AND TB.startyear <> '\N'

AND TC.Directors <> '\N'

AND TP.nconst <> '\N'

AND 1997 <= CAST(TB.startyear AS NUMERIC) AND CAST(TB.startyear AS NUMERIC) <= 2007

AND TB.titletype = 'tvEpisode'

AND (lower(TP.category) = lower('actor') OR lower(TP.category) = lower('actress'));

* **Comparison with response time**

We have implemented a 3 mapper with 2 reducer and 3mapper with 1 reducer, where we have observed that,

For 3M1R, launched maptask are 3 , total time taken by map task is 100.8 seconds where for 3M2R, total time taken by 3 map task is 77.248 seconds.

CPU execution time(3M1R) : 1464 seconds

CPU execution time(3M2R) : 1277 seconds

So it is observed that 3M2R map tasks are more faster than 3M1R

Also with 1 reducers task, total time spent is 66.947 seconds.

With 2 reducer task, total time spent is 100.987 seconds

So it is observed that 2 reducer task more faster than 1 reducer task with 3 mapper.

We have created a analysis spreadsheet to compare on the timing including CPU.

Attached with the report.

8. Analysis for Bonus problem

In Job Configuration Minimum split size is used for dividing the input file into blocks of mentioned size. This split size is given in bytes. We can change this split size to change the number of mapper taks being used for each input.

We decided to use 2 Different configurations for increasing number of mapper tasks

1. We decided to use split size as 400\*1024\*1024 bytes (400 MB). This would essentially process our 3 input file as follows
   1. title.basics.tsv – Size is **742 MB** – So with **400 MB** splitsize it will use **2 mapper tasks.**
   2. imdb00-title-actors.csv– Size is **666 MB** – So with **400 MB** splitsize it will use **2 mappers tasks.**
   3. title.crew.tsv– Size is **283 MB** – So with **400 MB** splitsize it will use **1 mapper tasks.**

**Total 5 mappers** tasks for 3 input files.

1. We used split size as 250\*1024\*1024 bytes (250 MB). This would essentially process our 3 input file as follows
   1. title.basics.tsv – Size is **742 MB** – So with **250 MB** splitsize it will use **3 mapper tasks.**
   2. imdb00-title-actors.csv– Size is **666 MB** – So with **250 MB** splitsize it will use **3 mapper tasks.**
   3. title.crew.tsv– Size is **283 MB** – So with **250 MB** splitsize it will use **2 mapper tasks.**

**Total 8 mapper tasks** for 3 input files.

We also increased number of reducers for these above configurations and following are our observations,

* With 5 mapper task & 1 Reducer task the total time taken in seconds for execution is 1163 seconds.
* With 5 mapper task & 2 Reducer task the total time taken in seconds for execution is 1014 seconds.

Hence, with increased number of reducer with 3 mapper task is more faster than a 1 reducer.

* Also we tried 2 reducer configuration for 8 mapper tasks where we observed that total time taken in seconds for execution is 1135 seconds which is little faster than 5 mapper with 1 reducer task.

We have also taken other observations with all mappers & reducers which is given in the separate analysis report attached with bonus problem folder.

9. References

1. project-3-description.pdf

2. proj3-help.pdf

3. Lecture slides

4. Book : Ramakrishnan - Database Management Systems 3rd Edition.pdf