Practical Assignment
VLBA II: System Architecture

Task 5

What is the average revenue of each movie genre? Give a list of all genresand the average revenue per movie of each of those. Additionally present to each genre the movie with the biggest revenue.

Name: Niha Mohanty
Magdeburg, 5. May 2019
Student Number: 221269

Otto-von-Guericke-Universität Magdeburg
Magdeburg Research and Competence Cluster
Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme
Very Large Business Applications Lab

Contents – Task 5

General Overview

Requirement:

Perform the given tasks using Hadoop Framework.

Input :

A CSV file of data is given. - A huge date set. In this case Movie Data Set is given to perform the task.

Output:

The input is processed in the Framework to obtain the desired output in the output folder.

Process:

In the Hadoop Framework the task of obtaining the desired out is performed by the following steps:

1. Here first the configuration is set by fetching the arguments from the command line and use them further processing to perform the given tasks. In my program the arguments are:

    I. **Parent Directory**: It represents the path of the parent folder where the input file reside. Its a mandatory field. It is represented in the command line as
    -parentDir=<PATH_OF_PARENT_DIR>

    II. **Input Folder:** It represents the name of the folder where the input file resides. It is also a mandatory field. It is represented in the command line as
    -inFolder=<FOLDER_NAME>

    III. **Output Folder:** It represents the folder name where the output of the performed task will be stored. It is a optional field. The default value is ***Solution.*** It is represented in the command line as
    -out*Folder=<FOLDER_NAME>*

    IV. ***TERMS:*** It represents certain terms (can range from ***1 … N***) that need to be found along with its occurrence per Movie from the **Overview** field. The terms to be provided in this field are to be separated using **"|".** It is also an optional field but if provided will help to perform the 3rd Part of the task.It is represented in the command line as
    -term=<term1|term2….termN>

    V. ***IMDB ID:*** It represents the unique id of a Movie given the data set and fetch the runtime of the movie and use it for further processing of the task. It is a optional field. If this value is not provided the task won't performed. It is represented in the command line as
    -imdbId=<IMDB_ID>

    VI. ***RELEASE DATE:*** It represents a date in the format **dd.MM.yyyy.** It is also an optional field but if **IMDB_ID** is provided then this becomes a mandatory field. These both the values are interdependent as the task demands. It is represented in

the command line as
-releaseDate=<dd.MM.yyyy>

N.B.: *All these above parameters are need to be space separated in the command line for the execution of the task.*

2. In the Hadoop framework the input file is first fetched and the values are the collected to further process them in **MapReduce.** An input to each phase is **<*key-value*>** pairs and we need to specify two functions: ***map function*** and **reduce function**. The whole process goes through four phases of execution i.e., ***splitting, mapping, shuffling, and reducing***.

    I. ***Splitting:*** The input data first split in row-wise and then are collect in **<*key-value*>** pairs which stored in context and provided for further processing.

    II. ***Mapping:*** The split data is then provided to the Mapper which in turn face the required values and stored them in **<*key-value*>** pairs to and sent to the **Reducer** for further processing.

    III. ***Shuffling:*** The **<*key-value*>** pairs passed on from the mapping phase is then shuffled and grouped according to their keys. This is handled by the Hadoop framework and the data is then passed on to the next phase.

    IV. ***Reducing:*** This is the final phase where the real operations for the given tasks are performed. Here the values from the **Shuffling** phase are aggregated to return a single output value.

3. After the **MapReduce** process is completed the output is then stored in the Hadoop file system which will give us our desired result.

Part 1

Problem Statement:

What is the average revenue of each movie genre? Give a list of all genres and the average revenue per movie of each of those. Additionally present to each genre the movie with the biggest revenue.

Solution:

The given Input file is processed under the Hadoop Framework where the value from the Input data file is fetched into the mapper class. It is then split using "\t" and the desired fields [id, genres, imdb_id, original_title, overview, release_date, cinema_revenue, runtime, title] are retrieved and passed on to the Reducer Class.

The above problem statement make us to calculate the average revenue of each movie genre and the corresponding movie with highest revenue. To achieve this all the logical statements are performed in the Reducer Class and the output is then stored in the Hadoop file system.

Part 2

Problem Statement:

Write down a second program that takes the IMDB_id and release_date as parameters. Identify each movie that has similar runtime to the provided one +-10 minutes. The procedure shall start with movies from the given release_date. Each entry of the list shall include: id, runtime, original title and release date.

Solution:

The parameters that are passed from the command line IMDB_ID and RELEASED_DATE are used for this task. First the IMDB_ID is used to find the RUNTIME of the respective movie. This RUNTIME is used along with RELEASED_DATE to further find the movies that fall under +/- 10 minutes of the RUNTIME. The logical statements regarding this task are performed in the Reducer Class and the output is then stored again in the Hadoop file system.

Part 3

Problem Statement:

Write down a third program thatperforms a count ofthe occurrence for each of three input terms withinthefield“overview”. Such as for “me, agent, mafia”. Those terms need to be provided as parameters. Each occurrence of the defined term(s)should be counted and the respected movie id tracked. The program should work with one, two and three terms. If no parametersare provided, instead a warning should be displayed.The final list should provide the movie id as well as the number of eachmatched word(count).

Solution:

The parameter ***terms*** that are passed in the command line are used here. According to the problem statement the *terms* that are required are fetched from the above said parameter to find their total count per movie from the **Overview** field. The logical statements regarding this task are performed in the Reducer Class and the output is then stored again in the Hadoop file system.

Approach

I have combined all the three task into MapReduce process where their corresponding logical statements are performed one after the other in the Reduce function and consequently their results are written back to the content for the Hardoop Framework to store them as output.

In my program since all the task are clubbed together, there will be a single JAR file to execute all the task. These task are only distinguished from each other through the command line parameters. The parameters that are used for these task are as follows:

I. **TASK 1:**

Parent Directory, Input Folder, Output Folder

II. TASK 2:

Parent Directory, Input Folder, Output Folder, IMDB_ID, Release Date

III. TASK 3:

Parent Directory, Input Folder, Output Folder, Terms

How to Run the Task

With the provided JAR file of the given Tasks and the parameters we can run the following command in the terminal to get the desired output.

COMMANDS:

The following are the commands to run the tasks individually or together:

I. **TASK 1:**

```
hadoop jar <PATH_TO_THE_JAR_FILE>/<JAR_NAME>.jar
-parentDir=<PATH_TO_THE_PARENT_FOLDER _OF_THE_INPUT_FILE>
-inFolder=<INPUT_FOLDER_NAME> -outFolder=<OUTPUT_FOLDER_NAME>
```

Example:

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput -outFolder=/MovieOutput
```

or

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput
```

II. **TASK 2:**

```
hadoop jar <PATH_TO_THE_JAR_FILE>/<JAR_NAME>.jar
-parentDir=<PATH_TO_THE_PARENT_FOLDER _OF_THE_INPUT_FILE>
-inFolder=<INPUT_FOLDER_NAME> -outFolder=<OUTPUT_FOLDER_NAME>
-releaseDate=<DATE_TO_COMPARE_WITH_RELEASE_DATE>
```

Example:

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput -outFolder=/MovieOutput -imdbId=tt1530535 -releaseDate=22.12.2010
```

or

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput -imdbId=tt1530535 -releaseDate=22.12.2010
```

III. **TASK 3:**

```
hadoop jar <PATH_TO_THE_JAR_FILE>/<JAR_NAME>.jar
-parentDir=<PATH_TO_THE_PARENT_FOLDER _OF_THE_INPUT_FILE>
-inFolder=<INPUT_FOLDER_NAME> -outFolder=<OUTPUT_FOLDER_NAME>
-terms=<TERM_1|TERM_2|TERM_3>
```

Example:

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput -outFolder=/MovieOutput -terms=kkk\|hhh
```

or

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=/user/tompi
-inFolder=/MovieInput -terms=kkk\|hhh
```

III. **TASK 1 & TASK 2 & TASK 3: <All the 3 task together>**

```
hadoop jar <PATH_TO_THE_JAR_FILE>/<JAR_NAME>.jar
-parentDir=<PATH_TO_THE_PARENT_FOLDER _OF_THE_INPUT_FILE>
-inFolder=<INPUT_FOLDER_NAME> -outFolder=<OUTPUT_FOLDER_NAME>
-terms=<TERM_1|TERM_2|TERM_3> -imdbId=<IMDB_ID>
-releaseDate=<DATE_TO_COMPARE_WITH_RELEASE_DATE>
```

Example:

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=user/tompi
-inFolder=/MovieInput -outFolder=/MovieOutput -terms=primary\|argue\|host
-imdbId=tt1530535 -releaseDate=22.12.2010
```

or

```
hadoop jar /home/nihatompi/Desktop/MoviesManager.jar -parentDir=user/tompi
-inFolder=/MovieInput -terms=primary\|argue\|host -imdbId=tt1530535
-releaseDate=22.12.2010
```