DBMS Models and implementation
Instructor: Sharma Chakravarthy
Project 3: Data Analysis using Map/Reduce

Made available on:	**11/14/2020**
Submit and Demo by:	**12/7/2021 (11:59 PM) No extension for this project (last day of classes)**
Submit to:	**Canvas (1 zipped folder containing all the files/sub-folders)**
	https://elearn.uta.edu/
Weight:	**15% of total**
Total Points:	**100**

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. Hadoop is a widely used open source map/reduce platform. Hadoop Map/Reduce is a software framework for writing applications, which process vast amounts of data in parallel on large clusters. In this project, you will use the IMDB (International Movies) dataset and develop programs to get interesting insights into the dataset using Hadoop map/reduce paradigm. Please use the following links for a better understanding of Hadoop and Map/Reduce (https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html)

1. XSEDE Expanse M/R system

You will be using the XSEDE Comet system for your project. Your login has been added for usage. Instructions have been given for using Comet. This is a facility supported by NSF for educational usage. Please make sure you stay within the quota for usage which is approximately 500 SU's per team.

You can install Hadoop on your laptop/desktop for developing and testing the code before you run it on Comet. This is for your convenience. Please look up and install Hadoop if you plan to do that.

2. IMDB Dataset

IMDB is a data contains information about movies (international) and TV episodes from their beginnings. The information includes movie titles, directors, actors, genre, and year produced/started. Some rating information is also included. The same is true for TV episodes and includes number of seasons in terms of start and end years as well as episodes in each season. It is quite large and should not require additional information to understand the data set. The data of this database that you will use is given below. Here is the download link. It has 3 files

IMDb_Data.zip

i. IMDB_TITLES.txt

This dataset contains the information about the various **IMDb titles** (movies, tv episodes, documentaries etc.), produced across the world. Each field in this dataset is separated by a semi-colon. A random sample from this file has been shown below

tt0000091;short;The House of the Devil;1896;Horror,Short
tt0468569;movie;The Dark Knight;2008;Action,Crime,Thriller
tt0088610;tvSeries;Small Wonder;1985;Comedy,Family,Sci-Fi

The description of the various fields has been given below

Field Number	Field Name	Field Description
1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*
2	TITLE TYPE	Every IMDB title in the file is categorized as one of the following TITLETYPEs • *tvSpecial* • *tvMovie* • *tvShort* • *short* • *tvEpisode* • *videogame* • *movie* • *tvSeries* • *tvMiniSeries* • *video*
3	TITLE NAME	The name of the IMDB Title, example: *The Dark Knight*
4	YEAR	The year of release, example: *2008*
5	GENRE LIST	Multiple genres can be attached to a particular title, and they are separated by **commas,** example: *Action,Crime,Thriller*

ii. IMDB_ACTORS.txt

This dataset contains the information about the **actors from each IMDB title**. Each field in this dataset is separated by a semi-colon. A random sample from this file has been shown below

tt1410063;nm0000288;Christian Bale
tt1429751;nm0004266;Anne Hathaway
tt1872194;nm0000375;Robert Downey Jr.

The description of the various fields has been given below

Field Number	Field Name	Field Description
1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*

2	ACTOR ID	The 9-digit unique actor identifier, example: *nm0000288*
3	ACTOR NAME	The name of the actor, example: *Christian Bale*

iii. IMDB_DIRECTORS.txt

This dataset contains the information about the **directors for each IMDB title**. Each field in this dataset is separated by a semi-colon. A random sample from this file has been shown below

tt3724976;nm5387279
tt2181625;nm1608926
tt6642042;nm8844724

The description of the various fields has been given below

Field Number	Field Name	Field Description
1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*
2	DIRECTOR ID	The 9-digit unique director identifier, example: *nm5387279*

3. Project 3 Problem Specification: You need to compute the following for the given data using the map/reduce paradigm. Try to compare and understand how you would do it using RDBMS if these files were stored as relations. This may help you understand when map/reduce is meaningful and when to use a RDBMS.

i. **[PROJECT 3 – 100 points]** There are many instances when a person directs a movie, TV series etc., in which he also acts. For example, *Ben Affleck directed and starred in the 2012 movie Argo*. You need to do this for 3 title types (movie should be one of them) and 3 genres chosen by you. There are about 20+ genres altogether. **Write a Map/Reduce program to list the names of people, who have directed and acted in the same IMDb title (of one of the three genres chosen) along with title, and year.**

Hint: This problem corresponds to a typical SQL query which has joins, group by and having clauses. The purpose of this bonus problem is to understand how some of the relational computations can be performed using the map/reduce paradigm. You can also write an SQL for this and run it on the Omega Oracle IMDb database that has been setup.

The output from the 1st map/reducer task may look like a list of the following

title_type, title name, person name (both acted/directd), year, genre type (optional)
…
…

You need to design and develop a map program (including a combiner if needed) and a reduce program to solve the above problems. The most important aspects of this design will be to

identify the <key, value> pairs to be output by the mapper and computations in the reducer to produce the desired final output.

First get it working on 1M/1R. Then use: 2M/2R, and 4M/4R for the 1st map/reduce task for each input to analyze the performance. For the 2nd map/reduce task, you can use one mapper and 1 reducer. Comet uses shards of size 128MB, but can be reduced to match the number of mappers chosen.

4. **Project Report:** Please include (at least) the following sections in a **REPORT.{txt, pdf, doc}** file that you will turn in with your code:

   i. **Overall Status**
   Give a *brief* overview of how you implemented the major components. If you were unable to finish any portion of the project, please give details about what is completed and your understanding of what is not. (This information is useful when determining partial credit.)

   ii. **Analysis:**
   Explain all the results and related inferences (with graphs if needed) clearly. Especially, the how performance (total time taken) changes (improves?) when the number of mappers are increased. Feel free to experiment with than what is required for this project.

   iii. **File Descriptions**
   List the files you have created and *briefly* explain their major functions and/or data structures.

   iv. **Division of Labor**
   Describe how you divided the work, i.e. which group member did what. Please also include how much time each of you spent on this project. (This has no impact on your grade whatsoever; we will only use this as feedback in planning future projects -- so be honest!)

   v. **M/R configuration details for multiple inputs and other details**:
   What libraries and packages you have used for dealing with multiple inputs.

5. **What to submit:**

   - After you are satisfied that your code does exactly what the project requires, you may turn it in for grading. Please submit your project report with your project.
   - You will turn in one zipped file containing **a) source code, b) outputs from the M/R code for each input using 1M/1R, 2M/2R and 4M/4R, c) logs generated, and d) raw analysis results (spreadsheets etc.) as well as the d) report. This is for each input. This is not required for 2ns map/reduce pair.**
   - All of the above files should be placed in a single zipped folder named as - 'Fall_2021_**proj3_team_<*TEAM_NO*>**'. **Only one zipped folder should be uploaded using canvas.**
   - You can submit your zip file at most 3 times. The latest one (based on timestamp) will be used for grading. So, be careful in what you turn in and when!
   - **Only one person per group should turn in the zip file!**
   - **Late Submissions not allowed**

6. Coding style:

Be sure to observe the following standard Java naming conventions and style. These will be used across all projects for this course; hence it is necessary that you understand and follow them correctly. You can look this up on the web. Remember the following:

i. Class names begin with an upper-case letter, as do any subsequent words in the class name.
ii. Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter.
iii. Class, instance and local variables begin with a lower-case letter, and any subsequent words in the name of that variable begin with an upper-case letter.
iv. No hardwiring of constants. Constants should be declared using all upper case identifiers with _ as separators.
v. All user prompts (if any) must be clear and understandable
vi. Give meaningful names for classes, methods, and variables even if they seem to be long. The point is that the names should be easy to understand for a new person looking at your code
vii. Your program is properly indented to make it understandable. Proper matching of if … then … else and other control structures is important and should be easily understandable
viii. Do not put multiple statements in a single line

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation for generating meaningful Javadoc.

7. Grading scheme:

The **PROBLEM 1** will be graded **out of 100** using the following scheme:

a. Correctness of the Map/combiner Code **15**
b. Correctness of the Reduce Code **15**
c. Correctness of the Output **20**
d. Report **25**
   i. Analyzing the results obtained - *20*
      - **Analysis Results with graphs, plots etc., wherever necessary**
   ii. Overall Completeness of report – *5*
      - **Status, File Descriptions, Division of Labor, config info**
   iii. Answering questions during demo **25**

8. Project 3 Demonstrations: Mandatory Team-wise demos will be scheduled for after the last day of classes. A sign-up sheet for the demo Schedule will be provided. If you finish early, you are welcome to demonstrate early by sending an email to the TA.