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

Made available on:	**10/28/2018**
Submit and Demo by:	**12/4/2018 (11:55 PM) No extension for this project**
Submit to:	**Blackboard (1 zipped folder containing all the files/sub-folders) https://elearn.uta.edu/**
Weight:	**15% of total**
Total Points:	**100**

Bonus Part (opportunity for making up to improve your grade)

Made available on:	**10/28/2018**
Submit and Demo by:	**12/4/2018 (11:55 PM) No extension for this project**
Submit by:	**Blackboard (1 zipped folder containing all the files/sub-folders) https://elearn.uta.edu/**
Weight:	**15% of total**
Total Points:	**50**

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. **Installation: There are two options**
   A. **RECOMMENDED - Hadoop Single Node Cluster Setup (Hadoop 2.9.1):** We advise you to use a Linux installation such as Ubuntu 16.04 on your system in order to breeze through the steps of Hadoop cluster set up. You can use a virtual machine for this purpose. The steps to install a *single node cluster in the pseudo distributed mode* are given here - https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html . Some important points that must be considered while setting up the cluster are:
      - Preferred Mirror site for download: http://mirrors.ibiblio.org/apache/hadoop/common/hadoop-2.9.1/
      - After completing the prerequisites and running the command ($ bin/hadoop), jump to https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation, and

follow the steps for Configuration, Setup passphraseless ssh, Execution (Only steps 1 – 4) and YARN on single node (Steps 1-3)

- The cluster will start up by running the commands,
  $ sbin/start-dfs.sh
  $ sbin/start-yarn.sh
  which you must have done while following the above steps.
- To check if the Namenode, Datanode, Secondary Namenode, ResourceManager, NodeManager are running as separate processes, run the command
  $ jps
- Once the cluster is up, run the most basic code, WordCount, that counts the number of occurrences of each word in the input files
  - Download the WordCount.java code from blackboard into the updated Hadoop folder
  - Make a directory (example, inputFiles) on the HDFS (Hadoop Distributed File System) to store the input files
    $ **bin/hdfs dfs -mkdir** /inputFiles
  - Copy a file from the local file system to the HDFS using the command
    $ **bin/hdfs dfs -put** <path_of_file_on_local_system> /inputFiles
  - To execute the code, follow the steps given here, https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v1.0
- To stop the cluster, run the commands
  $ sbin/stop-dfs.sh
  $ sbin/stop-yarn.sh
- Some useful shell commands for the HDFS can be found here, https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

B. **Cloud Services (self-exploration required):** The second option is to use Amazon Elastic Map/Reduce. Amazon EMR (Elastic Map/Reduce) is a web service provided by Amazon that uses Hadoop and distributes large datasets and processes them into multiple EC2 instances. You have to sign up for AWS. Please make sure you read carefully your AWS agreements/contracts/free use. You may have sufficient free services to complete the projects, but you should monitor and understand your use. Don't leave things running when not necessary. Note that if you exceed your monthly quota, you will get charged. Also, a credit/debit card is necessary for signing up for this service. You may use other cloud environments like IBM cloud, Google cloud or Microsoft Azure. **For these, we will not be able to provide assistance.** There should be online help available for different cloud providers regarding how to set up a cluster and run the first basic map/reduce code.

Note that if you use Amazon/ECS, you can use multiple mappers and reducers and will be able to better understand and appreciate distributed aspect of map/reduce and how the response time changes (should decrease) with more resources. You may not be able to do this on your installation of Hadoop on your laptop or desktop.

2. IMDB Dataset

IMDB is a dataset containing 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.

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]** Write a Map/Reduce program to count the number of movies belonging to each genre for every movie that was released during or after the year 2000 (Dataset to be used: IMDB_TITLES.txt). Example output

2000,Comedy,123
2000,Drama,89
2001,Thriller,56

Using the generated result, analyze and plot (or analyze) 5 interesting inferences. *For example, you can plot a histogram of each genre for this period. You can come up with other types of analysis, such as the genre that is declining (meaning less movies are produced) during this period, or genre that are going strong (better than the average) during this period. Trends for different genres across years?* You can use spreadsheet to post process the results obtained from the Map/Reduce program to perform the analysis. You can plot histograms, line graphs, scatter plots etc., in order to justify your inferences. Think out of the box!

ii. **[PROJECT 3 BONUS PROBLEM – 50 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*. Write a Map/Reduce program to list the names of people who have directed and acted in the same IMDb title (of any type), at least 100 times. The two datasets that need to be used for this are – IMDB_DIRECTORS.txt and IMDB_ACTORS.txt. As you will see, it will be easy to change the number to get different analysis outputs.

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 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.

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 Results:**
Explain all the results and related inferences (with graphs if needed) that were drawn.

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. **Logical errors and how you handled them**:
List at least 3 logical errors you encountered during the implementation of the project. Pick those that challenged you. This will provide us some insights into how we can improve the description and forewarn students for future assignments.

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 you're **a) source code, b) outputs from the M/R code, c) raw analysis results (spreadsheets etc.) as well as the d) report**
- All of the above files should be placed in a single zipped folder named as - '**proj3_team_<*TEAM_NO*>**'. **Only one zipped folder should be uploaded using blackboard.**
- 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**
- **Project 3 and the bonus part need to be submitted separately on bb! Use the name 'bonus_proj3_team_<TEAM_NO>' for the bonus part submission.**

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 Code **20**
b. Correctness of the Reduce Code **20**
c. Correctness of the Output **20**
d. Report **40**
   i. Analyzing the results obtained - *30*
      - **5 Analysis Results with graphs, plots etc., wherever necessary**
   ii. Overall Completeness of report – *10*
      - **Status, File Descriptions, Division of Labor, Logical Errors**

The **BONUS PROBLEM** will be graded **out of 50** using the following scheme:

e. Correctness of the Map Code **10**
f. Correctness of the Reduce Code **10**
g. Correctness of the Output **15**
h. Report **15**
   i. Analyzing the results obtained - *10*
      - **2 Analysis Results with graphs, plots etc., wherever necessary**
   ii. Overall Completeness of report – *5*
      - **Status, File Descriptions, Division of Labor, Logical Errors**

8. Project 3 Demonstrations: Mandatory Team-wise demos will be scheduled for **December 4/5, 2018**. A sign-up sheet for the demo Schedule will be provided. If you finish early, you are welcome to demonstrate early by sending the TA an email.