Assignment details

Overall Contribution: The individual assignment has 100 points in total and it counts towards 40% of the overall mark.

Assignment format: The assignment submitted should be a **single PDF document**, which should include all of your answers, with all the relevant workings, programming code and charts.

Exercises: You should complete all exercises. Below is the marking scheme for each exercise.

Exercise	1	2	3	4	5	6	7
Points	25	15	10	18	12	15	5

A. **Datasets:** On the below assignment we will make use of 3 out of the 5 MovieLens datasets – provided as part of the course material - that are found in the `movies` subfolder inside the `Datasets` folder. These datasets are `u.data`, `u.item`, `u.user`. If you have successfully set up your Hadoop cluster and followed the instructions in class, these datasets should be available on your hdfs under `./input/movies` directory, and you can also view them through the web UI on `localhost:50070`, on tab `Utilities/Browse the file system`. If the datasets are not in your HDFS, you need to import them from your local drive.

B. **Hadoop daemons:** For all exercises you will need to start all four daemons of your Hadoop cluster. Hadoop should be run in the background, whether you are working on any exercises in Hadoop Streaming, Hive or Pig. Make sure to also start the job history server, when starting up Hadoop so that you can refer to a job after it has been completed – you can do this by `mapred historyserver`.

C. **Hive:** Remember that you first need to start the Derby network server in a command prompt window, and then start Hive in another command prompt window. D. **Pig:** You just need to start pig in a command prompt window.

1 Hadoop Streaming with Python

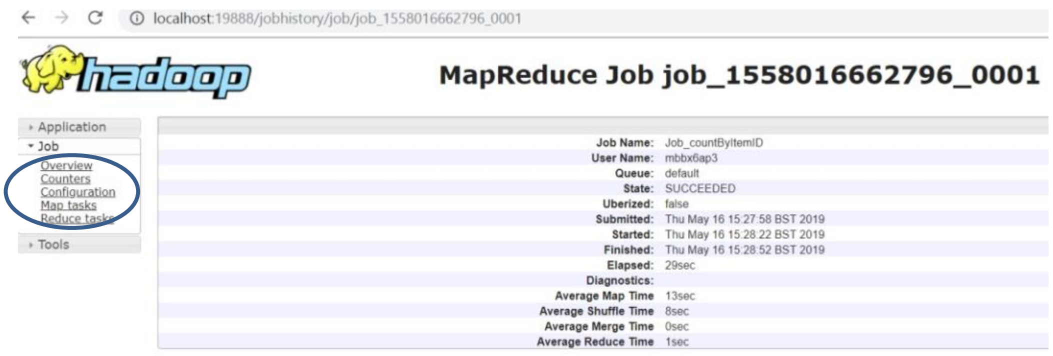
For all questions in this section, provide

- the python scripts you have written
- the Hadoop streaming command you used to run the job(s) on the Hadoop cluster
- The job/application ID that was automatically assigned to each mapreduce job that you have run for each exercise

Exercise 1 – Counting popularity of movies

a) Write a python mapper and reducer script to count the number of users that have rated each movie (which is an estimate of a movie's popularity). Output should be in the form `(itemID \t number of users rated itemID)`. Define the job name as "`job_countByItemID`" and run the streaming job.

b) Navigate to the web UI for the history server (`localhost:19888`). Select the job with name "`job_countByItemID`" and then use the link on the left (as shown below) to navigate to the *Counters*.



localhost:19888/jobhistory/job/job_1558016662796_0001

MapReduce Job job_1558016662796_0001

Job Name:	Job_countByItemID
User Name:	mbbx6ap3
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu May 16 15:27:58 BST 2019
Started:	Thu May 16 15:28:22 BST 2019
Finished:	Thu May 16 15:28:52 BST 2019
Elapsed:	29sec
Diagnostics:	
Average Map Time	13sec
Average Shuffle Time	8sec
Average Merge Time	0sec
Average Reduce Time	1sec

How many *Map input records, Map output records, Reduce input records* and *Reduce output records* do you see?

c) Save the result in a local directory and provide an Excel bar chart of only the first 20 movies (`itemIDs`) you see in the list, along with their frequencies (note that the output does NOT necessarily need to be sorted by popularity).

Exercise 2 – Counting popularity of movies (added combiner)

a) Consider about adding a combiner step on the previous streaming job.
   - What is the purpose of adding a combiner?
   - Is it possible to construct a combiner function for the above problem? Justify your answer.
   - What is the corresponding script for the combiner for this problem?

b) Run the job again with having added a combiner and with the name of this job specified as "`job_countByItemID_withCombiner`".

c) Inspect the *Counters* for this job and determine whether the combiner has been able to reduce the data transfer of key-value pairs between the mapper and the reducer phase. Provide necessary explanation.

Exercise 3 – Histogram of movie counts

a) Based on the output of Exercise 1, write a python mapper and reducer script to "count the counts". In other words, you need to find how many movies have been rated once, how many twice, three times and so on? Output should be in the form of `(frequency, number of movies with that frequency)`.

b) Export the resulting pairs to a local directory and draw the respective scatterplot in Excel How many movies have been rated only once?
    How many ratings did the most popular movie have?

2 Exercises using Hive

Use Hive to answer the following questions. For all questions provide the hive scripts you have written.

<u>Note for creating Hive **managed** tables:</u> For the creation of the below tables you may use

- either the relevant datasets that are on your local drive `C:/B1415/input/movies`
- or the datasets that reside on your hdfs, in hdfs directory `./input/movies`
  Remember that in this case, the relevant dataset will be **moved** from the hdfs directory `user/yourName/input/movies` to the **hdfs hive** directory in `user/hive/warehouse`.

Exercise 4 – Popularity of movies and average ratings

Write a hive query (or a set of hive queries) to:

a) create a managed table in Hive named `uData` to store **all** columns of the `u.data` dataset. The column names for `uData` should be `userID, itemID, rating, unixtimestamp` and they should all be specified as integers. Load the respective `u.data` into this and then display the first 20 records of the generated table.

b) create a managed table in Hive named `uItem` to store the **first two** columns of the `u.item` dataset. The column names for `uItem` should be `itemID, itemTitle,` and they should be specified as integer and string respectively. Load the respective `u.item` into this and then display the first 20 records of the generated table.

c) join the `uData` table together with the `uItem` table in order to assign to all movie ids, their respective movie title. The resulting joined table should have the following columns: `userID, itemID, rating, unixtimestamp, itemTitle.` The joined table should be named `uData_jn_Item.` Display the first 20 records of this table.

d) count the number of users that have rated each movie (which is an estimate of a movie's popularity), and sort them in popularity descending order.
   - How many movies have been rated at least 300 times
   - Export the table into the local PC and draw a bar chart of the respective movie titles and their frequencies in Excel.

e) find the most-rated movie that has an average rating greater than or equal to 4.5?

Exercise 5 – Breakdown of movies by gender

Write a hive query (or a set of hive queries) to:

a) create a managed table in Hive named `uUser` to store **all** columns of the `u.user` dataset. The column names for `uUser` should be `userID, age, gender, job, zipCode`, where userID, age, zipCode are all integers, while gender and job are strings. Load the respective u.user into this and then display the first 20 records of this table.

b) to join the uData_jn_Item table together with the uUser table in order to assign to all users, their respective gender. The new table should have the following columns: userID, itemID, rating, unixtimestamp, itemTitle, gender. Display the first 20 records of this table.

c) display the breakdown of number of records by men (M) and women (F), and also the average rating provided for each gender.
   - Are most ratings provided by men or women?
   - Does their average rating provided by men and women differ substantially?

d) find the most-rated movie for men that has an average rating of 5, and which for women?

3 Exercises in Pig

Use Pig to do the following tasks. For all exercises provide the pig statements you have written.

Exercise 6 – Find the top 20 movies by average rating score

Write a pig statement sentence to:

a) load u.data and specify a schema of four columns named userID, itemID, rating, unix_timestamp all specified as integers
b) to group the relation in (a) by the itemID field
c) to calculate the average rating score for each movie based on the relation in (b)
d) to sort the previous relation (c) in descending order
e) to keep only the top 20 movie ids from the relation in (d)
f) execute data flow and instruct pig to store the relation in (e) on HDFS. Provide a list of the top 20 movies ids and their average rating score. **Exercise 7 – Find the worst movies based on average rating score**

Write a pig statement sentence to:

a) to keep only movies with average rating score not greater than 1.5.
b) execute data flow and store the result on HDFS. How many movies are there?

END OF INDIVIDUAL ASSIGNMENT