CISC 432*/CMPE 432*

Assignment #2 – Spark, Hadoop MapReduce and HDFS

Due Monday Nov 5, 2018 – 11:59pm

The assignment objective is to get you started with Big Data platforms such as Apache Spark and Apache Hadoop. It is highly recommended that you use the Hortonworks Sandbox Distribution to implement the requirements of this assignment. You may use the Hortonworks Sandbox on Microsoft Azure service. In addition, you can use the VirtualBox software to create a virtual node on your local machine, and then install the Hortonworks Sandbox Image on that virtual machine. You are required to implement the assignment using Apache Spark and Apache Hadoop. An important outcome of this assignment is your experience with the Hadoop Eco-System and writing an Apache Spark application.

Rules:

- Teams of maximum 2 people **[Very Recommended].**
- One submission per team.
- Your submission **MUST** include:
  - A report as stated in the requirements section below. The report must have a cover letter that states team members' names and IDs.
  - All code developed (python or java). Add the code inside a folder with the requirement number as the folder name. Add a readme.txt file inside each folder that states the steps to run the code.
  - The randomly generated ratings netIDs.dat from Dataset#1. The one you used in the requirements and in Assignment#1.
  - The results for each script per system
- Follow the folder structure inside the SubmissionTemplate folder. Create a folder for your submission and compress it into one archive (.zip). Upload your archive to OnQ. Name the archive with your student number (actual number). For example:
  - studentNumber1_studentNumber2.zip (group of 2 students), eg:1432412_3412313.zip
  - studentNumber1.zip (1 student)

Assignment archive should be uploaded on OnQ before the due date. Late assignments are subject to a 10% per day late penalty, with weekends counted up to maximum 5 days.

In this is assignment you will perform few map / reduce tasks using Spark and Hadoop MapReduce. Using your generated NetId (60K) file from Assignment#1 dataset#1, you are required to compute a query as a data processing job. The query is defined below:

- **Find the number of reviews and average rating for each movie. Sort the movies in an ascending order using the average rating of each movie.**

[*] Please note that the number of reviews also means how many times a movie is rated.

The sorted output of the results should be in the following format:

- MovieId, Average Rating, Number of Reviews
- MovieId, Average Rating, Number of Reviews
- MovieId, Average Rating, Number of Reviews
- MovieId, Average Rating, Number of Reviews
- … down to 20 movies

You are free to change the order of columns of this format. For example, the following formats are also allowed:

- MovieId, Number of Reviews , Average Rating, … Or
- Number of Reviews, Average Rating, MovieId ..

* Please note that the movies are sorted in an ascending order using the Average Rating. In your implementation, you are required to calculate the average rating of each movie using Big Data processing systems required in this assignment. You are not allowed to create a side function to do that for you in the code.

Requirements:

Write a 3-6 pages report (12 pt. font, double-spaced) on your experience of using the Apache Spark and Apache Hadoop systems. The report should include the following:

1. Discuss the differences between MapReduce and MapReduce2.0. Also, discuss the differences between SparkRDD and SparkSQL.
2. Add your data to Hadoop HDFS. All the scripts in this assignment should read data directly from Hadoop HDFS. Discuss the steps you did to add your data to Hadoop HDFS.
3. Implement the required data processing using Hadoop MapReduce. Describe each line and step in your implementation (including the comments in the code). Save the output in the appropriate location in your submission folder.
4. Implement the required processing using Spark RDDs. Describe each line and step in your implementation (including the comments in the code). Save the output in the appropriate location in your submission folder.
5. Implement the required processing using Spark SQL. Describe each line and step in your implementation (including the comments in the code). Save the output in the appropriate location in your submission folder.
6. Apply a performance comparison using the execution time for the scripts and systems execution. Explain the reasons behind the differences between the systems performance. The performance comparison should be in the following table format:

	Hadoop MapReduce	**Spark RDD**	**Spark SQL**
Run#1: Execution Time	…	…	…

[*] Save the output of each script in a separate file under the folder **scripts_output** within the submission folder.

[*] You can use Python, Java, R, or Scala.

Bonus Challenge [15% Bonus ~ additional three marks]:

- **Find the movies that have been rated by at least by two users and have at least 1 rating with value larger than 3. Sort the movies in a descending order using the average rating of each movie.**

- B1. Using the same data, implement the query above once using Python Spark-RDD and a second time using Scala-Spark-SQL. Describe each line and step in your implementation (including the comments in the code). All the scripts in this assignment should read data directly from Hadoop HDFS. Save the **script and output** of each run in the Bonus folder within your submission template.
- B2. Apply a performance comparison using the execution time for the scripts and systems execution. Explain the reasons behind the performance difference between the two scripts.
- B3. You will only earn the bonus marks if you completed all the requirements in (B1 and B2). **No partial credits for incomplete requirements**. Write your bonus in a separate report and add it to the Bonus folder.

	Python - Spark RDD	Scala Spark- SQL
Run#1: Execution Time	…	…

-

Resources:
https://media.readthedocs.org/pdf/mrjob/latest/mrjob.pdf
https://spark.apache.org/docs/latest/rdd-programming-guide.html
https://spark.apache.org/docs/latest/sql-programming-guide.html