Project 3 for CS585/DS503: Big Data Management Fall 2017

Working with Spark

Total Points: **100 points**

Given Out: **Wednesday, 11th Oct, 2017**

Due Date: **Monday, 6th Nov, 2017 (11:59PM)**

Submit the project via CANVAS.

Team Project **Project is to be done in teams of 3+ students each, called your Project3-team. Team members will be assigned by CS585 staff.**

Project Scope and Submission

In this project, you will write Scala/SparkSQL code for executing jobs over the Spark infrastructure. You can either work on a *Cluster Mode* (files are read/written over HDFS) or a *Local Mode* (files are read/written over the local file system). In either case, your code should be designed with a high degree of parallelization.

You will submit below as one **single zip file** via CANVAS, namely:

1. You will submit a file containing the **Scala/SparkSQL code**.
2. This should include ONE **report (.pdf or .doc)** containing all required documentation and explanations of your solutions.
3. Also, provide **clear install and execution instructions**, along with data sets used (a sample of the data files, if too large, or the data generator). If your install script and environment description are not clear and the TA cannot run your code, there may be some point deductions – or at least some delay in grading your project.
4. In this report, you again indicate the **relative contributions** of each of your team members to this project. All team members must discuss and agree to a joint statement that then is submitted as a part of the above project report – and that reflects indeed the division of labor as indicated in your report.
5. Lastly, **each of you will independently** submit your peer team comments to the CS585 staff via a survey (GOOGLE form). This is your personal assessment of your own contributions and effort as well as the contributions of your team members to this project. These comments will be treated confidentially.

<u>Spark Virtual Machine</u>

You can either build your own VM or work with the one provided to you.

1- You need to download the Virtual Machine from this link:
http://web.cs.wpi.edu/~meltabakh/VM/Spark.vdi

2- To run the machine, download the VirtualBox tool. It is free and can be obtained from:
https://www.virtualbox.org/wiki/Downloads

3- When open VirtualBox → Click the "New" icon to start setting up the machine.
OS Type: Linux
Version: Ubuntu (64 bits)
memory: Recommended to have between 4 to 8 GBs (The higher the better)
Hard Drive: Use exiting virtual hard drive <<and select the downloaded file>>

4- When the machine is setup and you start it, the OS username is "mqp", password is "mqp123", and root password is the same.

5- You should find a **<u>Readme</u>** file on the desktop with useful instructions on how to start.. Make sure to read this file.

<u>Problem 1 SparkSQL (Transaction Data Processing) [50 points]</u>

Write a program that creates two datasets (files): ***Customers*** and ***Transactions***. Each line in *Customers* file represents one customer, and each line in *Transactions* file represents one transaction. The attributes within each line are comma separated.

The ***Customers*** dataset should have the following attributes for each customer:

ID: unique sequential number (integer) from 1 to 50,000 (that is the file will have 50,000 line)
Name: random sequence of characters of length between 10 and 20 ***(do not include commas)***
Age: random number (integer) between 10 to 70
CountryCode: random number (integer) between 1 and 10
Salary: random number (float) between 100 and 10000

The ***Transactions*** dataset should have the following attributes for each transaction:

TransID: unique sequential number (integer) from 1 to 5,000,000 (the file has 5M transactions)
CustID: References one of the customer IDs, i.e., from 1 to 50,000 (on Avg. a customer has 100 trans.)
TransTotal: random number (float) between 10 and 1000
TransNumItems: random number (integer) between 1 and 10
TransDesc: random text of characters of length between 20 and 50 ***(do not include commas)***

Note: The column names will NOT be stored in the file. The values are comma separated. From the order of the columns; you will know what each column represents.

Load your data into your storage. Then create a Spark workflow using SparkSQL to do the following:

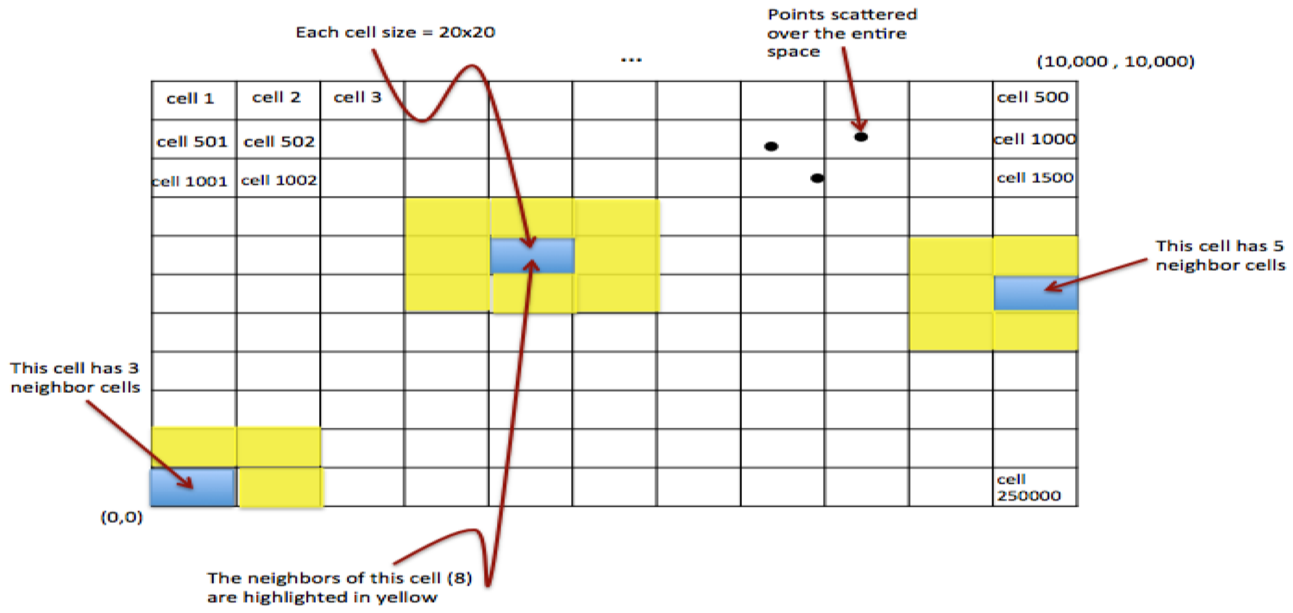
1) T1: Filter out (drop) the transactions from **T** whose total amount is less than $200
2) T2: Over T1, group the transactions by the Number of Items it has, and for each group calculate the sum of total amounts, the average of total amounts, the min and the max of the total amounts.
3) Report back T2 to the client side.

4) T3: Over T1, group the transactions by customer ID, and for each group report the customer ID, and the transactions' count.
5) T4: Filter out (drop) the transactions from **T** whose total amount is less than $600
6) T5: Over T4, group the transactions by customer ID, and for each group report the customer ID, and the transactions' count.
7) T6: Select the customer IDs whose T5.count * 3 < T3.count
8) Report back T6 to the client side

Problem 2: Spark-RDDs: Grid Cells of High Relative-Density Score. [50 points]

Overview:

Assume a two-dimensional space that extends from 1…10,000 in each dimension as shown in Figure 1. There are points scattered across the space. The space is divided into pre-defined grid-cells, each of size 20x20. That is, there is 500,000 grid cells. Each cell has a unique ID as indicated in the figure. Given an ID of a grid cell, you can calculate the row and the column it belongs to.



Each cell size = 20x20
Points scattered over the entire space
...
(10,000 , 10,000)
cell 1
cell 2
cell 3
cell 500
cell 501
cell 502
cell 1000
cell 1001
cell 1002
cell 1500
This cell has 5 neighbor cells
This cell has 3 neighbor cells
cell 250000
(0,0)
The neighbors of this cell (8) are highlighted in yellow

Neighbor Definition: For a given grid cell X, N(X) is the set of all neighbor cells of X, which are the cells with which X has a common edge or corner. The figure illustrates different examples of neighbors. Each non-boundary grid cell has 8 neighbors. A boundary cell will have a smaller number of neighbors. Since the grid cell size is fixed, the IDs of the neighbor cells of a given cell can be computed.

Example: N(Cell 1) = {Cell 2, Cell 501, Cell 502}
N(Cell 1002) = {Cell 501, Cell 502, Cell 503, Cell 1001, Cell 1003, Cell 1501, Cell 1502, Cell 1503}

Relative-Density Score: For a given grid cell X, I(X) is a decimal number that indicates the relative density of cell X compared to its neighbors. It is calculated as follows:

$$I(X) = X.\text{count} / \text{Average}(Y_1.\text{count}, Y_2.\text{count}, \ldots Y_n.\text{count})$$

where "X.count" means the count of points inside grid cell X, and
$\{Y_1, Y_2, \ldots, Y_n\}$ refers to the neighbor cells of cell X, that is $N(X) = \{Y_1, Y_2, \ldots, Y_n\}$.

P2.A: Create the Datasets [5 Points]

You can re-use your code from Project 2, if you like.

- Your task is to create one dataset *P* (set of 2D points). Assume the space extends from 1…10,000 in the both the X and the Y axis. Each line in the file should contain one point in the format (*a*, *b*), where *a* is the value in the X axis, and *b* denotes the value in the Y axis. Choose a random function to create the points.
- Scale the dataset to be at least 100MB.
- Upload and store the file into HDFS

P2.B: Report the TOP k grid cells w.r.t their Relative-Density Score [30 Points]

In this task, you will write Scala (or Java code or for extra credit try both -- it is your choice) to report the top k grid cells X that have the highest **I** (X) score with k=100. You are to return the grid cell IDs, and not the points inside the grid cell. In other words, the output itself will be small. Write the workflow that reports the top k cell IDs along with their relative-density score. *Your code should be fully distributed and scalable.*

P2.C: Report the TOP k grid cells w.r.t their Relative-Density Scores [15 Points]

Continue using the results from the problem P2.B above, namely, the list of reported top 100 grid cells. For each of the reported top 100 grid cells, report their IDs and the relative-density scores of its neighbor cells.

- The end –