CSE 512 Distributed and Parallel Database Systems
Course Project Phase-3 Report

Group Name: Evil Geniuses

Member	ASU ID	E-Mail ID
Dhananjayan Santhanakrishnan	**1211181423**	**dsantha1@asu.edu**
Dilip Mallya Kasargod	**1211222360**	**dkasargo@asu.edu**
Sruthi Pusunuru	**1210888780**	**spusunur@asu.edu**
Vishwak Thatikonda	**1211246735**	**vthatik1@asu.edu**

Machines Used

1. Master: 12G + i7 4 Cores
2. Slave 1: 8G + i5 4 Cores
3. Slave 2: 8G + i3 4 Cores

Total Memory Available: 28G

Work done prior to Phase 3:
Phase 1:

1. Demonstrated bi-directional password-less SSH communication between Master and 2 slaves on a Hadoop cluster.
2. Used Spark to read data from HDFS.
3. Used zcta510 as query window.

Phase 2:
Task A

1) Implemented a java function to do a spatial join query using simple Cartesian product algorithm, the rectangle is compared with all the point data sets using regular GeoSpark spatial range query.
2) The jar file is compiled and is attached with the submission.
3) Scala code for importing, initializing and execution of the function have been compiled into a Readme file.

Task B

1) Comparison of execution time / average memory / average CPU utilization of entire cluster
   A. Compare Task 2a & 2b from Phase 1
   B. Compare Task 3a & 3b from Phase 1
   C. Compare Task 4a & 4b from Phase 1
   D. Compare Task 4a & 4c from Phase 1

2) Comparison of execution time / average memory / average CPU utilization of Phase 1 Task 4a, 4b, 4c & Phase2 Task A

We implemented the following tasks in phase 3:

1) Read the Jan 2015 NYC Taxi Trip Dataset into Hadoop cluster.
2) Generate key value pairs for the data present in the following range:
   40.5 <= latitude <= 40.9 and -74.2 <= longitude <= -73.7
3) Constructed each cell of size 0.01 * 0.01 in terms of latitude and longitude degrees.
4) Used time step size as 1 Day.

Problem:
We are provided a dataset consisting of over 1 billion records which contains information regarding the New York City Yellow Cab Taxi trips between the time frame of Jan 2009 and Jan 2015. This dataset contains key information regarding each taxi trip like pickup, drop off time, date, coordinates, trip distance, fare amount etc.

We are asked to construct an effective algorithm to find out the 50 most frequent drop off locations by passenger count in time and space by using Getis-Ord statistics.

Algorithm Implemented:

We wrote a *SpatialStatistics* class to implement the algorithm
Below are the functions implemented in the class

1) GetValues:
   I. Reads tuple values from the Hadoop cluster.
   II. Checks if the coordinates are within the required boundaries.
   III. Creates a key with Latitude, Longitude, Day and value is the number of times the key appears in the dataset.
   Key: Latitude, Longitude, Day
   Value: Count
   IV. Returns the key value pair

2) GetNeighbor:
   I. Gets neighbors of the given point and returns the list. As the keys are already in 4 digit interger values, we iterate through 3 loops each with -1 to 1 resembling the longitude, latitude and day while taking into consideration the corner cases that we may encounter. A list with all the neighbor is returned while the number of neighbor is the number of elements in the list.

3) SpatialStatistics:
   I. Calculates the Total count, average, standard deviation all of which are required to calculate the Z score of the given point.

II. Based on the average and standard deviation, we take every data point and assign a score to each data point. We calculate this to compute density or gScore (metric for density).

4) *SortByComparator:*

I. This function is used to sort in descending order based on gscore.

5) *FileWriter:*

I. This function is used to write the values into the file that has been passed as an argument

II. The format of the output is as follows:
Cell_x, Cell_y, time_step, zscore

Calculation of Z Score:
We need to calculate the Getis-Ord G_i^* statistic to calculate the gScore. We use the following formula:

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j} x_j - \bar{X} \sum_{j=1}^{n} w_{i,j}}{S \sqrt{\frac{\left[n \sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2\right]}{n-1}}}$$

where x_j is the attribute value for cell *j*, $w_{i,j}$ is the spatial weight between cell *i* and *j*, *n* is equal to the total number of cells, and:

$$\bar{X} = \frac{\sum_{j=1}^{n} x_j}{n}$$

$$S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\bar{X})^2}$$

The G_i^* statistic is a z-score, so no further calculations are required.

Mapping Phase:
We map the number of occurrences to a key which comprises of the Latitude, Longitude and Day.
Key: Latitude, Longitude, Day
Value: Number of occurrences
The Latitude and longitude values are in 4 digit int numbers and the negative values are converted to positive values. Therefore, 73.99 becomes 7399 for convenience in the next steps.

Filter Phase:
The values that do not belong to the designated zone have been removed. During the creation of the key value pair, any values that do not belong in the region have been given a value of 0 and key as “default”. The user of the filter function removes all the tuples with the key default to remove the unnecessary values.

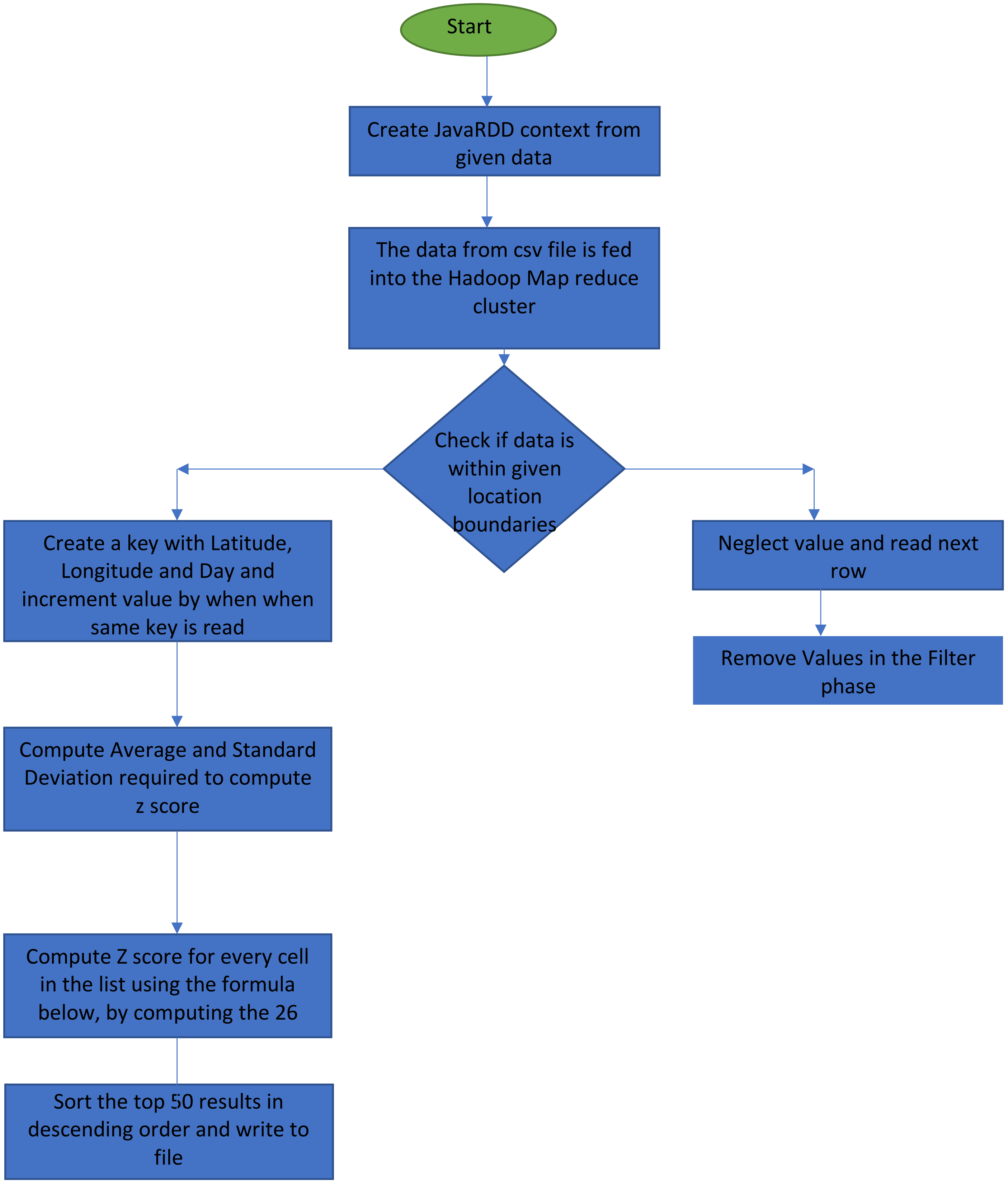
Reduce Phase:
This is implemented to reduce the total number of occurrences based on the key. So all the tuples which have the same key will become one tuple with the sum of all the values. So you basically get a count of the number of points in a particular 0.0.1 * 0.01 box.

Conclusion:
Our algorithm creates a Java RDD context out of the given data. The functions are parallelized using Map Reduce concept in the Hadoop cluster. This drastically reduces the run time of the program to under a minute.

We were successfully able to perform the operations asked on the dataset provided and been able to output the top 50 hotspots using our algorithm on the Hadoop cluster.

Values Acquired:
Mean – 183.3
Standard Deviation – 1507.56
Number of key-value pairs after reduce() – 15517
N – 40 * 55 * 31

Flowchart for Algorithm implemented in Phase 3


Start
Create JavaRDD context from given data
The data from csv file is fed into the Hadoop Map reduce cluster
Check if data is within given location boundaries
Create a key with Latitude, Longitude and Day and increment value by when when same key is read
Neglect value and read next row
Remove Values in the Filter phase
Compute Average and Standard Deviation required to compute z score
Compute Z score for every cell in the list using the formula below, by computing the 26
Sort the top 50 results in descending order and write to file