GROUP 18 - Data Processing at Scale

CSE 511

SYSTEM DOCUMENTATION REPORT

Members:
Prathyusha Kodali
Gayathri Alloju
Gouthami Kanchukatla
Alexandra Szilagy
Anubhab Guha

Table of Contents

CSE 511
Systems Documentation Report

1. System requirements

The motivation behind this project is to develop data processing methods for spatial databases. These functions are developed using SparkSQL. The language being used is Scala, which is a child language of Java. The primary IDE that is being used to develop and debug the software is Intellij.

2. Project background

A major peer-to-peer taxi cab firm has a large database that contains geographic data as well as real-time location data of their customers. They are looking for a team to develop and run multiple spatial queries on their database. A spatial query is a special query that is supported by both geodatabases and spatial databases. The goal of the project is to extract data from the company's database and perform analytical tasks which can be strategically used by the cab firm.

Phase 1

The goal of the first phase is to create two user-defined functions: ST_Within and ST_Contains. These functions are necessary to run four spatial queries through four different functions.

- **ST_Within**
  ST_Within takes three parameters, a point p1, another point p2 and a double value distance. It returns a boolean value indicating whether the distance between point p1 and point p2 is less than the distance value provided or not.

- **ST_Contains**
  ST_Contains is a user-defined function. It takes in the parameters of: a list of four values that make the points of a rectangle R and a list of two values that make a point P. Here, the rectangle R represents a geographical boundary in a city and the set of points P represents all the customers who are requesting the client's taxi cab service within the location bounded by R. The required functionality for this function is to return a boolean variable that determines if the point is contained completely within the rectangle.

- **Range query**
  The Range Query is a function called runRangeQuery that is calling the user-defined function ST_Contains and is passing the values of a query rectangle R and a set of points P as the parameters. The purpose of runRangeQuery is to find out all the points of set P that lie within the rectangle R.

- **Range Join Query**
  Similar to Range Query, a function called runRangeJoinQuery is used to execute the query. The function runRangeQuery is calling the

function ST_Contains and is passing the values of a set of rectangles R and a set of points P. The function's required functionality is to determine which rectangles contain which points.

- **Distance query**
  Given a fixed point location P and distance D (in kilometers), we need to use the 'ST_Within' function to find all points that lie within a distance D from P.

- **Distance join query**
  Given two sets of points P1 and P2, and a distance D (in kilometers), we need to find all (p1, p2) pairs such that p1 is within a distance D from p2 (i.e., p1 belongs to P1 and p2 belongs to P2) with the use of 'ST_Within' function.

Phase 2

The main goal of the second phase is to determine the amount of "hotness" that an area in space has. This "hotness" can be determined by how many data points, which represents customers requesting a taxi cab service, is in that area of space, represented by a rectangle, within a certain amount of time. So, the higher the "hotness" within an area, the higher is the profit for the client.

- **Hot zone analysis**
  The hot zone analysis function takes a data set of rectangles and a data set of points as parameters. Its required functionality is to calculate the hotness of each rectangle by determining how many points are contained in said rectangle. For example, a rectangle with 20 points is hotter than a rectangle with 3 points.

- **Hot cell analysis**
  The hot cell analysis function takes in the parameter of a data set of points. This function requires the use of spatial Gi* statistics that returns a z-score for each feature in the dataset to calculate the hotness of three dimensional cubes of space where the z axis is a range of time. For statistically significant positive z-scores, the larger the z-score implies more intense the clustering of high values (hot spot). For example, a rectangle may be hotter at 5:00-5:30 pm than it is at 1:00-1:30 pm.

3. Development process

To provide structure through the development process, the team divided the work for every phase before the first phase even began. This made it so that there was no confusion or uncertainty later in the development process.

Phase 1

- **ST_Contains**

Function: ST_Contains
Input: String point, String rectangle
Output: Boolean
Steps:
(a) "point" string contains x and y coordinate of the point separated by ",". Therefore, it is split into coordinates and assigned to variable x and y respectively.

(b) “rectangle“ string contains two points which are diagonal to each other. Therefore, the rectangle string is split into 2 points which in turn have 2 coordinates and have been assigned to x1, y1, x2, y2 respectively.
(c) If the value of x falls between x1 and x2 and the value of y falls between y1 and y2, that means the point lies within the rectangle, and hence, True is returned, or else False is returned if the point lies outside the rectangle.
Contributors: Anubhab Guha, Gouthami Kanchukatla, Alexandra Szilagy

- **ST_Within**

Function: ST_Within
Input: String p1, String p2, Double distance
Output: Boolean
Steps:
(a) “p1“ string contains x and y coordinate of the point separated by “,“. Therefore it is split into coordinates and assigned to variable x1 and y1 respectively.
(b) “p2“ string contains x and y coordinate of another point separated by .ĂIJ,.Ăİ. Therefore it is split into coordinates and assigned to variable x2 and y2 respectively.
(c) Find Euclidean distance between p1 and p2 using the below formula and assign the result to variable d.

$$d = sqrt((x1 - x2)2 + (y1 - y2)2)$$

(d) If the value of d is less than or equal to the given distance value, then that means that the point p1 is within the given distance from point p2 , then true is returned else false is returned point p1 is not within the given distance from point p2.
Contributors: Prathyusha Kodali

- **Range query**

Function: runRangeQuery
Input: SparkSession spark, String arg1, String arg2
Output: Long count
Steps:
(a) Load the file passed as input in arg1 into the pointDf dataframe and create a temporary view “point” with each row containing a point in pointDf.
(b) Execute a spark SQL query that selects all the rows from the point view where the function ST_Contains returns true for a point and the given rectangle.
(c) ST_Contains will return a boolean for each row in the point view. It will return true for points that are contained within the rectangle.
(d) The output of the function is the count of how many rows the spark SQL query returns.
Contributors: Anubhab Guha, Gouthami Kanchukatla, Alexandra Szilagy

- **Range join query**

Function: runRangeJoinQuery
Input: SparkSession spark, String arg1, String arg2
Output: Long count
Steps:
(a) Load the file passed as input in arg1 into the pointDf dataframe and create a temporary view “point” with each row containing a point in pointDf.
(b) Repeat step a, but pass arg2 into the rectangleDf dataframe and create a temporary view “rectangle” with each row containing a rectangle in rectangleDf.
(c) Execute a spark SQL query that selects all the rows from the rectangle view and all the rows from the point view where the function

ST_Contains returns true for a point and a rectangle.
(d) ST_Contains will return a boolean for each possible pair of points and rectangles. It will compare each rectangle with each point and return true for each point that is contained in the rectangle and return false for each point that is not contained in the rectangle.
(e) The output of the function is the count of how many rows the spark SQL query returns.
Contributors: Anubhab Guha, Gouthami Kanchukatla, Alexandra Szilagy

- **Distance query**

Distance query uses the ST_Within function implemented to find all the points that lie within and distance D (in kilometers) from a fixed point location P.
Function: runDistanceQuery
Input: SparkSession spark, String arg1, String arg2, Double arg3
Output: Long Count
Steps:
(a) Load the file passed as input in arg1 into pointDf dataframe and create a temporary view '**point'** with each row containing a point in pointDf.
(b). Execute a spark SQL query to select all the rows from the **'point'** view for which boolean value 'true' is returned by user defined function ST_Within.
(c) ST_Within function takes the point p0 stored in the first column of each row in 'point' view, a fixed point P passed in arg2 and distance D passed in arg3 as inputs and returns true when the fixed point P is at a distance of D from the point p0. If the point p0 is not within distance D from P then ST_Within returns false.
(d) The number of rows (Count) returned by the spark SQLquery is the output of the runDistanceQuery function.
Contributors: Gayathri Alloju

- **Distance join query**

Distance join query uses the ST_Within function implemented to find all the pairs of points (p1, p2) such that p1 is within a distance D from p2 , given two sets of points P1 and P2, and a distance D (in kilometers).
Function: runDistanceJoinQuery
Input: SparkSession spark, String arg1, String arg2, Double arg3
Output: Long Count
Steps:
(a) Load the file passed as input in arg1 into pointDf dataframe and create a temporary view '**point1'** with each row containing a point in pointDf.
(b) Load the file passed as input in arg2 into pointDf2 dataframe and create a temporary view '**point2'** with each row containing a point in pointDf2.
(c) Execute a spark SQL query that selects all the rows resulting from **join** operation on **'point1'** and ''**point2'** on a condition that boolean value returned by the user defined function ST_Within is true.
(d) ST_Within function takes a point p1 of 'point1' view, point p2 of 'point2' view and distance D passed in arg3 as the three inputs and returns true when p1 is at a distance of D from point p2. If the point p1 is not within distance D from p2 then ST_Within returns false.
(e) The number of rows (Count) returned by the SQLquery is the output of the runDistanceJoinQuery function.

Contributors: Gayathri Alloju, Prathyusha Kodali

Phase 2

The distribution of phase two was difficult because there are only two methods that have to be distributed between five team members. This challenge was levitated by assigning three team members to the more difficult method.

- **Hot zone analysis**

In Hot Zone Analysis, we have used a rectangle dataset and a point dataset which contains the points that lie inside this rectangle. We determine how hot the rectangle is based on the number of points that lie inside the rectangle, the more points a rectangle contains, the hotter it will be.

Input: SparkSession spark, String pointPath, String rectanglePath
Output: Sorted dataframe representing rectangle and the number
of points in it (hotness).
Steps:
(1) Get all the points from pointPath.
(2) Get all the rectangles from rectanglePath.
(3) Run a spark join query on the above two sets with join condition obtained from ST_Contains method to get the join result dataframe.
(4) Run a spark group by query on the join result data frame grouping on rectangle and count(points) as aggregation function
sorted by ascending order of rectangle to obtain sorted dataframe of rectangles and their hotness

Contributors: Gayathri Alloju, Prathyusha Kodali

- **Hot cell analysis**

The ‘hot cell analysis’ applies spatial statistics to spatio-temporal Big Data in order to identify statistically significant hot spots using Apache Spark.

Name: runHotcellAnalysis
Input: SparkSession spark, String pointPath
Output: sorted dataframe representing cells and their hotness
Steps:
(1) Get all the points from pointPath.
(2) Run spark sql query on all the points which will output dataframe with x, y and z as columns. x gets the latitude from the point, y gets the longitude from the point and z gets the pickup day. Result is a cell dataframe.
(3) Find Getis - Ord Statistic value for each of the cell in the dataframe and return the top 50 cells in the descending order of the Getis - Ord Statistic value (hotness).
Contributors: Anubhab Guha, Gouthami Kanchukatla, Alexandra Szilagy