Geo-Spatial Data Analysis using SparkSQL

Shashi Kiran Chilukuri
Arizona State University
Schiluk6@asu.edu

Yingkang Luo
Arizona State University
Yingkang.Luo@asu.edu

Santosh Dahal
Arizona State University
sdahal7@asu.edu

***Abstract*—With the evolution of technology, the volume of location-aware data is increasing at a staggering rate. Many companies want to understand the spatio-temporal aspect in this geo-spatial data to better serve their customers. It is because, this spatial context helps to identify the trends in a complex relationship between the people and geographic locations. In this "Geo-Spatial Analysis" project, we have developed and applied spatial analytics on New York city Yellow Cab data to identify fifty most significant hot spots as specified by the ACM SIGSPATIAL 2016 Cup using Spark SQL. This paper will first introduce to the project, provide the solution in detail, followed by the results, personal contribution, lessons learned and references.**

I. Introduction

As part of "Geo-Spatial Data analysis" project in *CSE 511: Data Processing at Scale* course for summer 2020, we need to develop and apply geo-spatial analytics on the New York city Yellow Cab data with the objective of identifying hot zones for pickups within the city limits. This given dataset is a collection of New York City Yellow Cab monthly taxi trip records spanning from 2009 to 2012 and it contains geographic data as well as real-time location data of the customers. It is given that source data may be clipped to an envelope encompassing the five New York City boroughs in order to remove some of the noisy error data. To implement this project we used the Spark SQL module of the Apache Spark cluster framework.

To develop the solution, project was divided into two phases. In each phase, we were given with spatio-temporal dataset and a code template that needs to be developed.

In the first phase of the project, we need to develop two Spark SQL functions and use them to perform four spatial queries namely, Range, Range join, Distance and Distance join queries. While one of the functions needs to check if the two points are within the given distance, another needs to check if the given query rectangle coordinates contains the point.

In the second phase of the project, we need to implement two major tasks namely, "hot zone analysis" and "hot cell analysis". The "hot zone analysis" uses the function defined earlier to check if the given query rectangle coordinates contains the point and then identifies the number of such points in each rectangle area to indicate the hotness of that rectangle. The "hot cell analysis" uses Getis-Ord statistics to identify statically significant hot spots.

At the end, our goal was to

- Develop Spark SQL functions to perform spatial queries for extracting meaningful information related to customer's location and the geographic data.
- Implement "hot zone analysis" and to identify hot zones (profitable zones).
- Implement "hot cell analysis" and to identify statistically significant hot spots.
- Identify 50 most significant hot spots that may help in making operational and strategic level decisions.

Resources used for the project:

TABLE I
RESOURCES

	Resources
Environment	Apache Spark, Scala
Integrated Development Environment (IDE)	Intellij / Eclipse
Query Language	SparkSQL
Dataset	Points Data S3 Bucket, Zone data: "src/resources/zone-hotzone"

Reasons for using Apache Spark:

Geo-spatial data contains a very large number of data points and analyzing this high-density data is a complex problem. It requires high computational and operational power to analyze and extract meaningful information. To solve this, one would require large scale distributed processing framework such as Spark. Spark is an Apache project, advertised as "lightning fast cluster computing". It provides a faster and more general data processing platform, and provides APIs in Scala, Java, and Python, with support for other languages. Also, it introduces the concept of Resilient Distributed Dataset (RDD) which is an immutable fault-tolerant, distributed collection of objects that can be operated on in parallel. An RDD can contain any type of object and is created by loading an external dataset or distributing a collection from the driver program. Because of these reasons, we are using Apache Spark with Spark SQL and Scala programming.

II. Solution

A. Phase One: Spatial Queries

In this phase, we developed two user defined functions namely, "ST_Contains" and "ST_Within". Later, these two functions were used in executing the four spatial queries for analyzing the data.

Function “ST_Contains” takes two parameters, a point and a rectangle as inputs and returns a Boolean to indicate whether the point is in the rectangle or not. Here are the details of this function:

- Function: “ST_Contains”
- Input: Point string, Rectangle string (Boolean: true or false)
- Output: Boolean (true or false)

Steps to implement “ST_Contains” function:

1) A Point string contains X and Y co-ordinates. Need to the Point string and store X and Y values separately.
2) A Rectangle string contains two points (X1, Y1) (X2, Y2) on either end of the diagonal of a rectangle. Need to parse these two points to get X1, Y1, X2, Y2 respectively.
3) Now, check if the given point’s X coordinate falls between Rectangle’s X1, X2 coordinates and also if the given points Y coordinate falls between Rectangle’s Y1, Y2 coordinates then return TRUE. This indicates, rectangle fully contains the point.
4) Otherwise FALSE. This indicates, rectangle doesn’t contain the point.

Function “ST_Within” takes three parameters, two points P1 and P2 and, a distance value. This function returns an output as “Boolean (true or false)” to indicate whether the distance between the two points is less than the distance value given. Here are the details of this function:

- Function: “ST_Within”
- Input: Point string1, Point string2, distance
- Output: Boolean (true or false)

Steps to implement “ST_Contains” function:

1) A Point string P1 contains X1 and Y1 co-ordinates. Need to parse the Point string and store X1 and Y1 values separately.
2) A Point string P2 contains X2 and Y2 co-ordinates. Need to parse the Point string and store X2 and Y2 values separately.
3) Calculate the Euclidean distance between the two points P1 and P2. Here is the formula used to compute the Euclidean distance:

$$\mathrm{d} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

4) If the obtained value is less than or equal to given distance value, then it is TRUE. This indicates that the points P1 and P2 are within given distance.
5) Otherwise, FALSE. This indicates that the points P1 and P2 are not within given distance.

To execute four spatial queries namely, Range query, Range join query, Distance query and Distance join query, above defined function are used. Here are the details of four spatial queries.

- *Range query*: It uses ST_Contains function in the query. When given with a query rectangle R and a set of points P, this query finds all the points within R.
- *Range join query*: It uses ST_Contains function in the query. When given with a set of Rectangles R and a set of Points S, this query will find all (Point, Rectangle) pairs such that the point is within the rectangle.
- *Distance query*: It uses ST_Within function in the query. When given with a point location P and distance D in km, this query will find all the points that lie within a distance D from P.
- *Distance join query*: It uses ST_Within function in this query. When given with a set of Points S1 and a set of Points S2 and a distance D in km, this query will find all (s1, s2) pairs such that s1 is within a distance D from s2 (i.e., s1 belongs to S1 and s2 belongs to S2).

B. Phase Two: Geo-Spatial Analysis

In this phase, performed spatial hot spot analysis. In particular, completed two different hot spot analysis tasks namely, “hot zone analysis” and “hot cell analysis”.For this task, New York City Yellow Cab monthly taxi trip records spanning from 2009 to 2012 is used. Also, in order to remove noise in the data, the source data filtered to encompass only the five New York City boroughs. Here is the pickup location (blue dots) map[1]:

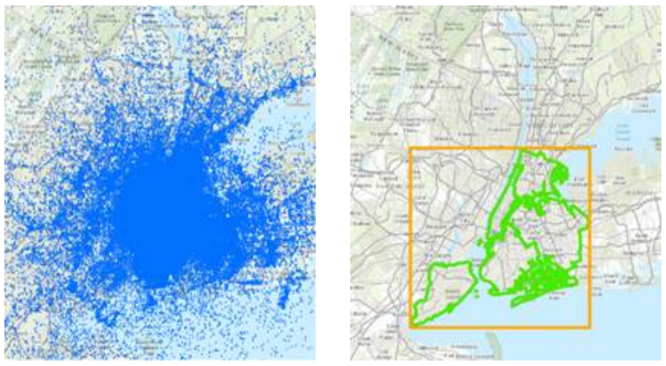

Fig. 1. Yellow cab pickup locations in New York[2]

Hot zone analysis: In this task, need to identity the hot (profitable) zones. For that, performed a range join operation on a rectangle dataset and a point dataset. For each rectangle, the number of points located within the rectangle will be obtained. The more the number of points in a rectangle, the hotter it will be. So, this function will return the hotness of all the rectangles. Here are the details of the function:

- Function: runHotZoneAnalysis.
- Input: Spark Session, point Path, rectangle Path.

- Output: Data frame with rectangle and points count in ascending order.

Steps involved to perform Hot zone analysis:

1) Retrieve the points data from the point path.
2) Retrieve the rectangles data from the rectangle path.
3) Perform the join operation on above datasets by using the "ST_Contains" function.
4) Perform the grouping operation on the rectangle and count the points on above dataset.
5) Display the resulting rectangles and their hotness in ascending order.

Hot cell analysis: In this task, need to identify the statically significant hot spots. To achieve this, applied Getis-ord statistic to spatio-temporal data. Here are the details of the function:

- Function: runHotcellAnalysis.
- Input: Spark Session, point Path.
- Output: Data frame of cells and their hotness.

Steps involved to perform this task:

1) Retrieve the points data (x, y and z co-ordinates of space-time cube) from the point path. Here x represents latitude of pickup location, y represents longitude of pickup location and z represents pickup day.
2) Get the number of points which are inside the cube.
3) Get the number of pick up location points.
4) Get the value of X-Bar and S. These are used for calculating the Getis-Ord Gi*
5) Calculate Getis-Ord Gi* statistic for each a cell in above data fame to return a list of the fifty most significant hot spot cells in time and space.

Space-Time Cube: When we aggregate Time and Space together they form the Space-Time Cube. The x, y and z coordinates that we got from above forms this space-time cube. Here x represents latitude of pickup location, y represents longitude of pickup location and z represents pickup day.

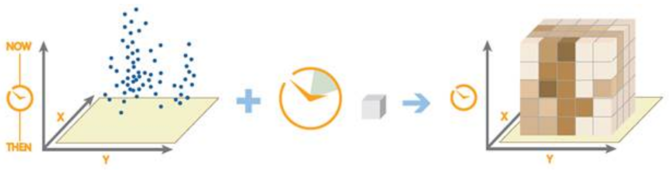

Fig. 2. Space-Time Cube[2]

The neighborhood for each cell in the space-time cube is established by the neighbors in a grid based on subdividing latitude and longitude uniformly. This spatial neighborhood is created for the preceding, current, and following time periods (i.e., each cell has 26 neighbors). For simplicity of computation, the weight of each neighbor cell is presumed to be equal[2].

Getis-Ord Statistic Calculation: Here is the Getis-Ord statistic formula for finding the list of significant hot spot cells.

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

Fig. 3. Getis-Ord statistic formula[2]

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:

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

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

III. Results

Based on the requirements provided, we had successfully developed spatial analytics to analyze geo-spatial data and identified fifty most significant hot spots. In the phase one, we developed two user defined functions "ST_Contains" and "ST_Within". While the former function checks if the rectangle coordinates fully contains the given point, the later function checks if the distance between two points is less than given distance. In turn, these two functions were used to perform spatial queries namely, Range query, Range join query, Distance query and Distance join query. In the phase two of the project, completed two different hot spot analysis tasks namely, "hot zone analysis" and "hot spot analysis". In the "hot zone analysis", hotness of all the rectangles are calculated and in the "hot spot analysis", applied spatial statistics to spatio-temporal big data in order to identify fifty most statistically significant spatial hot spots. Finally, developed code components are packaged in a jar by using the spark submit application for submission.

IV. Contributions

In this team project, I was involved in all the phases, right from understanding the project requirements to preparing the report for project submission. Here is the list of activities I performed as part of this project:

- Understand the project requirements and its tasks[3].
- Plan project with other team members based on project milestones.
- Project communication including scheduling zoom calls, Chat sessions etc.
- Create and maintain project to-do list.

- Create and maintain private Git code repository for version control.
- Suggest code logic for project project implementation.
- Develop spatial analytics functions and queries using Spark SQL and Scala.
- Review/debug the code developed by other team members.
- Prepare the system documentation report.
- Project code and document submission.

V. Lessons Learned

This is a team project. I worked with Yingkang Luo and Santosh Dahal to successfully implement this project. Here are some of the lessons learned from this it:

- Development of Geo-Spatial Queries using Spark SQL and Scala.
- Working with large scale data using Apache Spark clusters.
- Use of Getis-Ord statistic to identify statistically significant clusters.
- Holistic idea to implement real-world Geo-Spatial analytics projects.

References

[1] Dataset: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/
[2] Getis-Ord Statistic: http://sigspatial2016.sigspatial.org/giscup2016/problem
[3] Project requirements: https://www.coursera.org/learn/cse578/supplement/B4Lrb/course-project-introduction