Project Report for CSE 511: Data Processing at Scale, Project #2

Harsh Sanjaykumar Nagoriya
Arizona State University
Email: hnagoriy@asu.edu

Krushali Shah
Arizona State University
Email: kshah51@asu.edu

Savan Dixesh Doshi
Arizona State University
Email: sdoshi7@asu.edu

I. Introduction

Geospatial data, often known as spatial data, is a resource with a geographic component.In other terms, the information in this type of data set often comprises 2 or more dimensional attributes such as graph coordinates. Two spatial query mechanisms shall be implemented in this case namely

1) Hot zone analysis
2) hot cell/hotspot analysis.

The level of geographical data collected has increased dramatically in recent years. The most significant characteristics of a cab company are the client and driver locations. As a result, they must strive to query and analyze such massive volumes of spatial data in order to achieve high performance and low latency. The 'hot cell analysis' applies spatial statistics to Spatio-temporal Big Data in order to identify statistically significant hot spots using Apache Spark.

II. Development Approach

In our project meetings, we began with a requirement analysis of Phase 2 and several ways we may utilize to solve the given problem statement. We also addressed the next steps, determining what needed to be done to complete our skeleton code. We compared our output with the offered result example after completing the Hot Zone and Hot Cell Analysis. We reviewed peer code after it was implemented, improved it, and tested it.

III. Requirements

As stated above, to compile the scaffolding code, we needed to have following resources in our machines.

A. Machine requirements

We used the machines with following specifications in order to compile and test the code.

1) MacBook Air: 8-Core CPU, 7-Core GPU, 8GB Unified Memory, 256GB SSD Storage

2) Dell Inspiron 15 Gaming 5577: Intel Core i7-7700, NVIDIA GeForce GTX 1050 Ti, 8GB Unified Memory, 128GB SSD Storage

B. Operating System requirements

Since the dependencies were developed in specified version of software and platform Ubuntu, we needed to install Ubuntu in our systems.

C. Software requirements

We used the following software configurations in order to compile and test the code.

1) Java: Version 1.8.0_321
2) Scala: Version 2.11
3) OpenJDK: Version 8
4) Spark: Version 2.4.7
5) Hadoop: Version 2.7
6) Gedit Text Editor: Version 3.3
7) Visual Studio Code: Version 1.66

D. Importing the solution

The provided answer JAR files for both phases can be imported into any system with enough RAM to perform spatial queries and complete the assessment, preferably one with enough RAM to run the software versions specified above. To complete the project, follow the steps listed below.

1) Go to command prompt and the project directory. Run sbt assembly.
2) The above step would create a jar file in target.scala-2.11 folder.
3) Copy jar file and paste it in the project root folder.
4) Run the command spark-submit with input files added as arguments
5) The solution for both functions is stored in respective csv files in respective folders.

So as to keep all these things working, we set proper environment variables.

IV. Work Done

The Hot Zone analysis and the Range Join Query from Phase 1 had the same requirements. There would be a set of Rectangles R and a set of points S. The first step in implementing this query would be to discover all the (Point, Rectangle) pairs where the point is inside the rectangle, and then use a GROUP BY clause to aggregate the output for each rectangle to determine its hotness. We went over the offered template to understand the provided inputs, pre-processing, and desired output in order to fully understand the task for Hot Cell Analysis. The template included cells with two spatial and one temporal feature, such as latitude, longitude, and month day. To calculate

$$\sum_{j=1}^{n} W_{i,j}$$

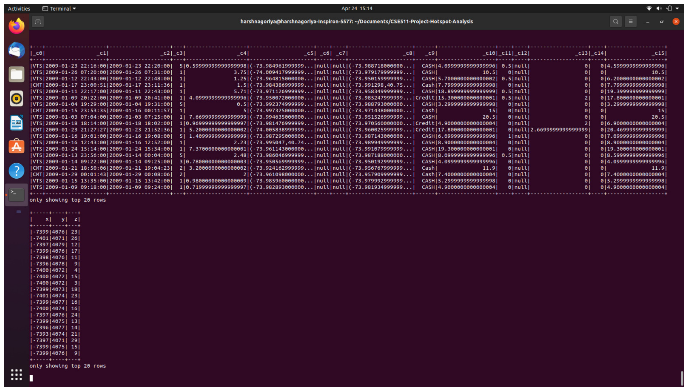
(which essentially is the number of neighbors for cell) we have used the geometric location of the cell to calculate its number of neighbors. There will be 27 count for number of neighbors if the dimension on boundary is zero. Similarly, the neighbor count for cells 1, 2, and 3 would be 18,12, and 8, respectively. Is then the

$$\sum_{j=1}^{n} W_{i,j}$$

count of trips in neighboring cells as the weight for each neighbor is equal. Now that we have all of the relevant tables, we can construct Getis-ord statistics and look at the numerator and denominator. The top 50 cells are then returned as a scala dataframe after sorting the result in descending order based on the G—score value.

V. Solution Verification

As a consequence, a Data-Frame with 50 cells and three locations for the cells with the greatest Getis-Ord statistics is constructed. We debugged the code, revealing intermediate data frames from the previous part's calculations as well as statistics like Mean and Standard Deviation. To make the JAR file, we used the **sbt clean assembly** command. To launch the JAR file that we used to check the findings, type **spark-submit**. As a result, a CSV file is created in the desired output directory. The input test data for both Hot Cell Analysis and Hot Zone Analysis was provided in CSV format. We compared our output to the actual CSV-formatted result file. **Fig 1** shows the obtained output after running the JAR file.


Activities
Terminal
Apr 24 15:14
harshnagorlya@harshnagorlya-Inspiron-5577: ~/Documents/CSE511-Project-Hotspot-Analysis

```
+---+-------------------+-------------------+---+-------------------+--------------------+----+----+--------------------+------+------------------+----+----+------------------+----+------------------+
|_c0|                _c1|                _c2|_c3|                _c4|                 _c5| _c6| _c7|                 _c8|   _c9|              _c10|_c11|_c12|              _c13|_c14|              _c15|
+---+-------------------+-------------------+---+-------------------+--------------------+----+----+--------------------+------+------------------+----+----+------------------+----+------------------+
|VTS|2009-01-23 22:16:00|2009-01-23 22:20:00|  5|0.59999999999999998|(-73.984961999999...|null|null|(-73.988718000000...|  CASH|4.0999999999999996| 0.5|null|                 0|   0|4.5999999999999996|
|VTS|2009-01-26 07:20:00|2009-01-26 07:31:00|  1|               3.75|(-74.009417999999...|null|null|(-73.979179999999...|  CASH|              10.5|   0|null|                 0|   0|              10.5|
|VTS|2009-01-12 22:43:00|2009-01-12 22:48:00|  1|               1.25|(-73.964815000000...|null|null|(-73.950159999999...|  CASH|5.7000000000000002| 0.5|null|                 0|   0|6.2000000000000002|
|CMT|2009-01-17 23:00:51|2009-01-17 23:11:36|  1|                1.5|(-73.984386999999...|null|null|(-73.991298,40.75...|  Cash|7.7999999999999998|   0|null|                 0|   0|7.7999999999999998|
|VTS|2009-01-11 22:17:00|2009-01-11 22:43:00|  1|               5.71|(-73.971126999999...|null|null|(-73.958349999999...|  CASH|18.899999999999999| 0.5|null|                 0|   0|19.399999999999999|
|VTS|2009-01-09 20:22:00|2009-01-09 20:41:00|  1| 4.0999999999999996|(-73.950072000000...|null|null|(-73.985247999999...|Credit|15.300000000000001| 0.5|null|                 2|   0|17.800000000000001|
|VTS|2009-01-04 19:29:00|2009-01-04 19:31:00|  5|                0.5|(-73.992374999999...|null|null|(-73.988793000000...|  CASH|3.2999999999999998|   0|null|                 0|   0|3.2999999999999998|
|CMT|2009-01-15 23:53:35|2009-01-16 00:11:57|  1|                  5|(-73.997325000000...|null|null|(-73.971438000000...|  Cash|                15|   0|null|                 0|   0|                15|
|VTS|2009-01-03 07:04:00|2009-01-03 07:25:00|  1| 7.6699999999999999|(-73.994635000000...|null|null|(-73.951526999999...|  CASH|              20.5|   0|null|                 0|   0|              20.5|
|VTS|2009-01-18 18:14:00|2009-01-18 18:02:00|  1|0.96999999999999997|(-73.981476999999...|null|null|(-73.970560000000...|Credit|4.9000000000000004|   0|null|                 2|   0|6.9000000000000004|
|CMT|2009-01-23 21:27:27|2009-01-23 21:52:36|  1| 5.2000000000000002|(-74.005838999999...|null|null|(-73.960025999999...|Credit|17.800000000000001|   0|null|2.6699999999999999|   0|20.469999999999999|
|VTS|2009-01-16 19:01:00|2009-01-16 19:08:00|  5| 1.4099999999999999|(-73.987295000000...|null|null|(-73.987143000000...|  CASH|6.0999999999999996|   1|null|                 0|   0|7.0999999999999996|
|VTS|2009-01-16 12:43:00|2009-01-16 12:52:00|  1|               2.23|(-73.995047,40.74...|null|null|(-73.989949999999...|  CASH|8.9000000000000004|   0|null|                 0|   0|8.9000000000000004|
|VTS|2009-01-24 15:14:00|2009-01-24 15:34:00|  1| 7.3700000000000001|(-73.961143000000...|null|null|(-73.991079999999...|  CASH|19.300000000000001|   0|null|                 0|   0|19.300000000000001|
|VTS|2009-01-13 23:56:00|2009-01-14 00:04:00|  5|               2.48|(-73.986046999999...|null|null|(-73.987188000000...|  CASH|8.0999999999999996| 0.5|null|                 0|   0|8.5999999999999996|
|VTS|2009-01-14 09:22:00|2009-01-14 09:25:00|  3|0.78000000000000003|(-73.950569999999...|null|null|(-73.950192999999...|  CASH|4.0999999999999996|   0|null|                 0|   0|4.0999999999999996|
|CMT|2009-01-21 18:50:06|2009-01-21 19:04:23|  2| 3.2000000000000002|(-73.924162999999...|null|null|(-73.956767999999...|  Cash|              11.9|   0|null|                 0|   0|              11.9|
|CMT|2009-01-29 00:01:43|2009-01-29 00:08:06|  2|                  2|(-73.961098000000...|null|null|(-73.957909999999...|  Cash|7.4000000000000004|   0|null|                 0|   0|7.4000000000000004|
|VTS|2009-01-15 13:35:00|2009-01-15 13:42:00|  1|0.98000000000000009|(-73.985960000000...|null|null|(-73.979992999999...|  CASH|5.2999999999999998|   0|null|                 0|   0|5.2999999999999998|
|VTS|2009-01-09 09:18:00|2009-01-09 09:24:00|  1|0.71999999999999997|(-73.982893000000...|null|null|(-73.981934999999...|  CASH|4.9000000000000004|   0|null|                 0|   0|4.9000000000000004|
+---+-------------------+-------------------+---+-------------------+--------------------+----+----+--------------------+------+------------------+----+----+------------------+----+------------------+
only showing top 20 rows

+-----+----+---+
|    x|   y|  z|
+-----+----+---+
|-7399|4076| 23|
|-7401|4071| 26|
|-7397|4079| 12|
|-7399|4076| 17|
|-7398|4076| 11|
|-7396|4078|  9|
|-7400|4072|  4|
|-7400|4072| 15|
|-7400|4072|  3|
|-7399|4073| 18|
|-7401|4074| 23|
|-7399|4077| 16|
|-7400|4074| 16|
|-7397|4076| 24|
|-7399|4075| 13|
|-7396|4077| 14|
|-7393|4074| 21|
|-7397|4071| 29|
|-7399|4075| 15|
|-7399|4076|  9|
+-----+----+---+
only showing top 20 rows
```

Fig. 1. Output

VI. Conclusion

To create our data, we used Apache Spark and Scala. The built solution was utilized to extract critical and decisive data from the provided dataset, which could later be used to make strategic decisions. We've also included a few custom spatial queries in the given solution. The solution provides the client with critical geographic data that can assist them make business strategy decisions and give better service to their customers.

VII. Contributions

We, all three students, have worked equally and all together with no conflicts of interest.

VIII. Acknowledgement

We, authors, would like to thank Dr. Zhichao Cao (Instructor), Akkamahadevi Hanni (Teaching Assistant), Rithvik Chokkam (Grad Service Assistant) Arizona State University for their constant encouragement towards the realization of this work.

References

[1] Spark SQL - Quick Guide, Available at: https://www.tutorialspoint.com/spark_sql/spark_sql_quick_guide.htm

[2] Downloads — Apache Spark, https://spark.apache.org/downloads.html

[3] Jia Yu, ”CSE512-Project-Hotspot-Analysis-Template”, https://github.com/jiayuasu/CSE512-Project-Hotspot-Analysis-Template

[4] Jia Yu, ”CSE512-Project-Phase2-Requirement”, https://github.com/jiayuasu/CSE512-Project-Phase2-Template

[5] Download & install apache-spark on MacOS (Big Sur, Monterey, Catalina, Mojave) via Homebrew / brew, Available at: https://www.youtube.com/watch?v=6c1uP_UbuBg