Data Processing at Scale Portfolio

Aman Peshin (1225476655)
apeshin@asu.edu

December 14, 2022

1 Introduction

The introduction of NoSQL databases has drastically changed how we transact and store data, these databases often provide better scalability and give better performance with nonstructural data. In this report, we would be focusing on two projects undertaken in this course, Finding Businesses based on query conditions and Hot-spot Analysis. Project 1 deals with finding the business around based on the city under consideration and the distance between the user and businesses.
Hot-spot analysis also has two sub-tasks that we implemented, namely hot zone analysis and hot cell analysis. In Hot zone analysis, we perform joins on point and rectangle databases and determine which rectangle is the hottest based on the number of points that are found inside it. Hot cell analysis deals with finding spatial hot spots based on spatial-temporal big data.

2 Description of Solution:

2.1 Project Phase 1:

In project 1, I have implemented two functions write_to_file and calcDistance. Write_to_file function serve the purpose of writing the found output to a text file based on the way we query and filter our businesses. The calcDistance function is a much more complicated function that finds the distance in miles based on the latitude and longitude of two locations.
We then implement the two driving functions FindBusinessBasedOnCity and FindBusinessBasedOnLocation.

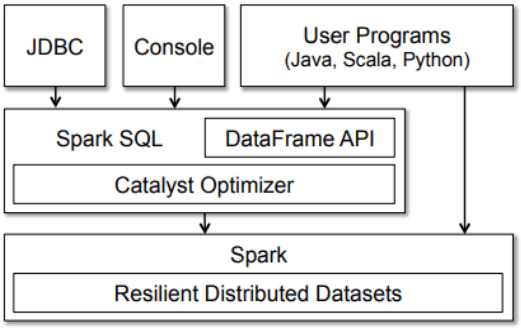

Figure 1: Spark SQL Block diagram
User programs are written in Scala (object-oriented and functional language)
Spark SQL is used to interact with the RDD datasets

2.1.1 Find business based on city:

The first mentioned function is straightforward where we query all the data present in our NoSQL database, we then iterate over the entire dataset and find all the key-value pairs of cities that match our considered city. We then store all the key-value pairs of business name, city, address, and state in a dictionary and write it to the output text file using the function write_to_file.

2.1.2 Find business based on location:

The second function FindBusinessBasedOnLocation is much more interesting because it makes use of spatial coordinate data in order to give us recommendations of the nearest businesses. This function takes the inputs categoriesToSearch, myLocation, maxDistance, saveLocation2, and collection. Here we take the queried data in the form of collection and find the distance between the business and myLocation iteratively, we then check whether the distance is less than the maxDis-

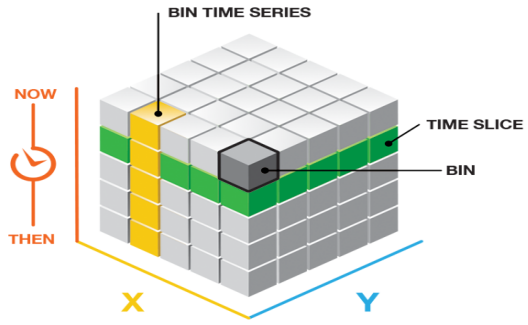

Figure 2: Space-Time Cube representation

tance specified. After this we check if the business meets all the categories mentioned in the categoriesToSearch parameter, if it even matches one of them like a buffet, Italian, or continental we then add this row of data to the result dictionary and export it to a txt file using write_to_file.

2.2 Project Phase 2:

In the HotSpot Analysis project, we also have two subtasks that we had to execute which were Hotzone and Hotcell Analysis. I had implemented a function ST_Contains that essentially splits the queryRectangle and pointStringbased on a delimiter and then checks for corner vertices, it then gives a Boolean output which tells us whether the point is present in the given rectangle or not.

2.2.1 Hot zone analysis:

In this function, we try and sort all the rectangles found based on the number of points inside the rectangle. A rectangle is called hotter than another when it has more points that are found using a range query on the rectangle and point dataset.
The function first loads point path data after which it cleans the data by removing the delimiters and trimming it. It then finds the hotness of each rectangle using ST_Contains function.

2.2.2 Hot cell analysis:

In this function, we process spatial data, and we calculate and output three values x, y, and z in this function, we sort these values based on the Z score, while the x, and y values are the pickup latitude and longitude. In order to calculate the Z

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

Figure 3: Getis-Ord formulas
x denotes the attribute value
n denotes the count of cells
w(i,j) denotes the spatial weight between i and j

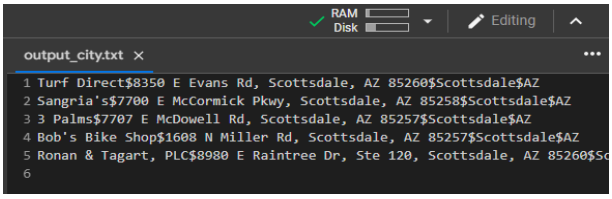

Figure 4: Business results based on city

score we make use of a space-time cube and spatial weights. A Space-time cube is used to represent the time-determined value of latitude and longitude of pickup locations spatial weights are calculated for a given edge where adjacent cells are given a value of 1 and others 0.

2.2.3 Getis-Ord statistics:

The G_i statistics returns a z value for each feature in the dataset, basically, it means that the larger the z score higher is the clustering of hot spots and similarly lower the z score means the lower the clustering of hot spots.

3 Results

In Project 1, we dump the filtered data into a text file for both our functions which find businesses based on location and city. The auto grader then compares the contents of the ideal results text file and our generated text file to determine if the tests have passed. The results for finding business based on the city are in the form of the restaurant name, city, state, and address delimited by a $. Some sample outputs of base test cases are given below.

In project 2, the results are generated in a different way. In this, we first build our Scala

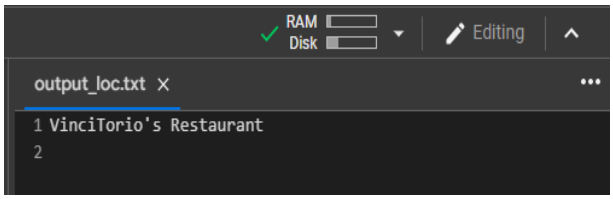

Figure 5: Business results based on location

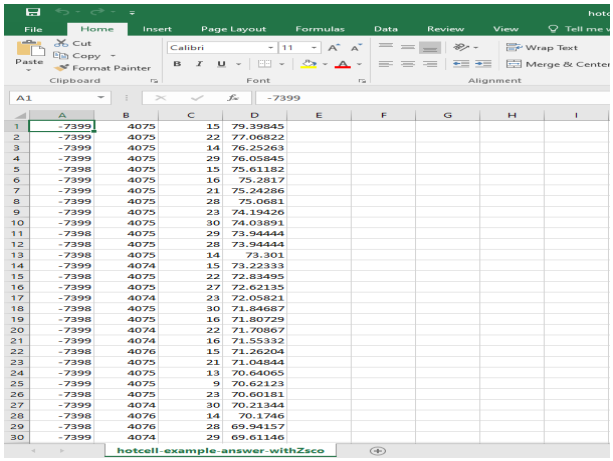

Figure 6: Hotcell analysis results

files and generate a .jar file from this. This .jar file is then used by us to run analysis on the sample database in CSV format which generated two CSV file outputs for hot cell and hot zone analysis. The sample outputs for the above two implementations are given above.

3.1 Real world application:

The two projects that we have implemented have great value in real-world applications, by implementing finding businesses based on location we can see how apps like google maps recommend places to visit, gas stations, and restaurants nearby. Our Project may be a simplistic implementation of this tool but nonetheless, it lets us work with spatial data and helps us explore how latitudes and longitudes are used to calculate distances between two locations. Hot cell and hot zone analysis allow us to find statistics about any type of data, in this project we used NYC taxi trips and analyzed which areas have the maximum pickup and drop. Stats such as these allow for user booking distribution which helps cab booking apps to have more cabs in densely populated areas to serve their customers better.

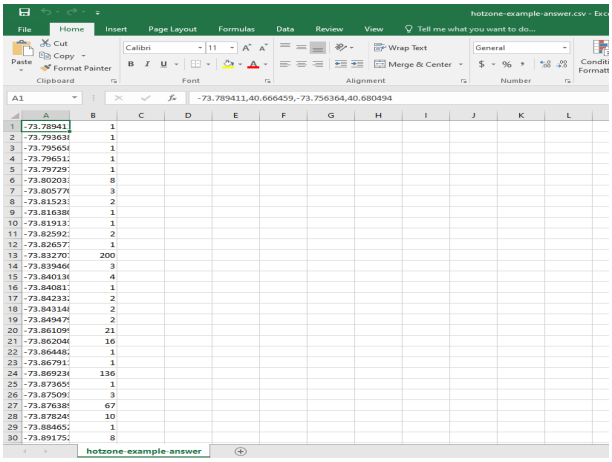

Figure 7: Hotzone analysis results

4 Lessons Learned

The tools and technologies that were used in implementing these projects gave us in-depth knowledge of how real-world applications are built and how they consume data to give relevant analytics. I got great hands-on experience with IDEs like Jupyter notebook and Google colab which are great to work with large datasets for visualizations. The databases we used were Non-relational and it gave us great insight into alternate storage structures that are used other than relational databases, NoSQL, and MongoDB helped me understand the different structures like key-value, document, and file structure of Non-relational DB.

Having not used any big data tools like Apache Spark, this project gave me the necessary exposure over the entire semester to work with this tool. In this project, we got the opportunity to learn new technologies and implement them for hot cell and hot zone analysis. We used Scala which is compiled into Java Byte Code and then executed by Java Virtual machine like in the case of our project setup. We have used a Java development kit to run Scala. SparkSQL was used to analyze the yellow taxi dataset, the main advantage while using this tool was the advantage it possesses in using Data Frames and can act as a distributed query engine.

References

[1] Getis–Ord Spatial Statistics to Identify Hot Spots by Using Incident Management Data

[2] The earth model-calculating field size and distance between points using GPS coordinates.