Large-Scale Geospatial Data Analysis using SparkSQL

Mahathi Srinivasan
Computer Science
Arizona State University
msrini22@asu.edu

Sai Sruthi Mallineni
Computer Science
Arizona State University
smallin1@asu.edu

Lakshmi Sindhuja Karuparti
Computer Science
Arizona State University
lkarupar@asu.edu

Abstract: In today's businesses, where production, resource tracking supply chain management is central to their profit, a lot of strategic decisions for the same are gained from valuable insights obtained from the analysis of Spatio-temporal or geographical data. Analysis of such data is an ordeal due to a plethora of unique, unusual patterns found in the data. Thus, many techniques are used to study a region's topological, spatial, or geographical characteristics. In this work, Spatial statistics is implemented on the New York Taxi dataset, a big spatiotemporal data, to identify the fifty most significant hot zones using the Getis ord Gi* statistic. The data is analyzed using the Spark SQL module of the Apache Spark Cluster framework, and the entire setup is run on an AWS EC2 instance.

Keywords: Apache Spark, Apache Hadoop, Amazon Web Services, EC2, Cloudwatch, SparkSQL, Getis-ord

I. INTRODUCTION

Analyzing geospatial data is an uphill task due to the nature of the data; it consists of a large number of data points, hence mandates a lot of computation and processing power to gain a better understanding of the data on a high level and thus, obtain valuable insights from it. One of the challenges in taking up such a task is to achieve low latency and high throughput from querying and analyzing a humongous amount of spatial data. This project report describes the methodology used to analyze the New York Taxi data from the years 2009 to 2012 to identify hot zones for pickups in different parts of the city. As part of this work, the methodology uses three EC2 instances of AWS (Amazon Web Services) along with the Apache Spark and Hadoop frameworks. The GeoSpark Application programming interface performs spatial data analysis on Apache Spark, and the data is queried using the SparkSQL module of Apache Spark.

The main aim lies within finding the set of points within a given set of rectangles, hot zones, and hot cells based on the number of customer pickups at a location within a given large dataset. A statistic 'Getis - Ord Gi*', is used to calculate the hot zones based on the number of customer pickups at a location. Preprocessing and conversion to spatial time cube considering latitude and longitude by day of a particular pickup within the taxi trip data set are done. Each cell in this space-time cube contains a range of locations per day, and the spatial proximity of each cell is determined using 'Getis - Ord Gi*' statistic by weighting the values to other cells.

II. EXPERIMENTAL SETUP

A. Infrastructure

Amazon Web Services EC2 Instances has inbuilt disk imaging capabilities and a variety of snapshots. It monitors Virtual Machine instances by incorporating adaptive control and is cost-effective in terms of meeting all requirements to conduct a geospatial analysis. Hence, it was chosen as the cloud infrastructure for this project. With EC2, we were able to achieve high and efficient network computing performance for the spatial analysis of the New York City Taxi dataset.

The system architecture comprises a master node("master") and two slave/worker nodes ("slave-1" and "slave-2"). All the nodes are present in the same local network.

SSH is used to login to remote servers for the execution of commands and programs through Command-Line tools (CLI). The above EC2 instances communicate with themselves as well as the other instances through password-less SSH. The instances are accessed on the web interface using port-forwarding and the addition of specific instance rules.

B. Frameworks

For geospatial data analysis, Apache Hadoop and Apache Spark are installed and integrated.

Apache Hadoop: Apache Hadoop[2] is a set of software utilities that facilitate using a network of master-slaves to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using Hadoop Distributed File System (HDFS) and the MapReduce programming model. HDFS incorporates replication of data and hence supports data integrity

Hadoop is installed on all three instances. All the necessary configurations are made to make the master node identify the worker nodes. The status of the Hadoop cluster can be verified by entering "master_ public_ip:50070" in the browser.

Apache Spark: Apache Spark[3] is an open-source distributed general-purpose and in-memory cluster-computing framework. It provides an interface performing operations on entire clusters of Resilient Distributed Dataset (RDD) with implicit data parallelism and fault tolerance.

The spark server is installed on all the instances and started on the master node. The status of the Spark master can be verified by entering "master_public_ip:8080". The three nodes present indicate the three instances that were set-up.

C. Considerations

EC2 instance, Hadoop and Spark setup:

EC2 instances were set up during this phase, and Hadoop and spark were respectively installed. As these instances were present on AWS, SSH was used to access these them.

Password-less SSH:

In this phase, the worker nodes were configured to have the ability to SSH into each other without using passwords to do so. The following steps were performed to achieve this:

1. Generate the public and private keys for each worker machine using the "ssh-keygen" command.

2. Copy all the public keys of the worker nodes onto the master nodes authorized_keys file.

3. Copy the public keys of each machine onto itself in the authorized_keys file. This will give the ability for every machine to communicate bidirectionally with every other machine as well.

4. Test on every machine if you can ssh without using the password by typing the command "ssh ubuntu@target_machine".

Hadoop Cluster:

1. Make the following changes in the file as part of the initial setup:

File: hadoop-env.sh
Property:JAVA_HOME
Value: /usr/lib/jvm/java_folder_location
File: core-site.xml
Property: fs.default.name
Value: hdfs://master:54310

2. Add the following lines in the Hadoop-Folder/etc/Hadoop/slaves file on the master machine:

- master
- slave-1
- slave-2

3. Use the command "hadoop namenode-format" to format the HDFS system.

4. Start the process on the master machine using the command "sbin/start-all.sh".

5. Test if the installation was successful and the process is active by opening "localhost:50070" in a web browser.

Spark Cluster:

Spark cluster is set up on three machines using the steps mentioned below:

1. In file "SparkFolder/conf/spark-env.sh", add the line "SPARK_MASTER_IP=master".

2. Add the following line in Spark-Folder/conf/slaves.template only in Master Node. Node, then rename slaves.template to /slaves to make it come into effect.

- master
- slave-1
- slave-2

3. Start the spark process by executing the command "sbin/start-master.sh".

4. Open "localhost:8080" on a web browser to check the status of the spark master.

D. Geospatial Data Analysis

We used SparkSQL to run four spatial queries: Range Query, Range Join Query, Distance Query, and Distance Join Query. These queries use two user-defined functions ST_Contains(returns if the given point is within the rectangle or not) and ST_Within(returns if the distance between point p1 and point p2 is less than the given distance)to complete the task.

Range Query: This query takes a rectangle R and a set of points P as input and returns the points that lie inside rectangle R using ST_Contains.

Range Join Query: This query takes a set of rectangles and a set of points and returns all rectangle that contains the given point pairs using ST_Contains.

Distance Query: This query takes a point P and distance D as input and returns all points that lie within distance D from point P using ST_Within.

Distance Join Query: This query takes two sets of points P_set1 and P_set2, and a distance D as input and returns all pairs of points p1, p2, which satisfies the condition.

E. New York Taxi Cab Data Geospatial Analysis

Spatial hot spot analysis (Hot Zone analysis and Hot Cell analysis) is performed on the given dataset that contains the pickup points within Newyork City.

Hot Zone Analysis: This task takes a rectangle dataset and a point dataset. It runs Range Join query that groups rectangles such that count of all the points within each rectangle can be obtained that makes the analysis of the hotness of rectangle.

Hot Cell Analysis: This task takes a list of cells, and their hotness based on Getis - Ord statistic is returned. A cell is a space-time cube with the x-axis representing the latitude of the pickup location, y-axis representing the longitude of the pickup location, and z-axis representing the date of the pickup.

III. RESULTS

This aims to measure the scalability of the system using three EC2 instances with one node as master and other as slaves. We measure metrics such as CPU utilization and the network traffic transferred.

Each data point in the below graphs measures the performance metrics at the time mentioned on the x-axis.

We have used the default monitoring system "amazon cloudwatch" that is available in aws to measure performance. Cloudwatch gives us various metrics like disk reads, disk writes, status checks, etc. Out of these, we found maximum variation in CPU Utilization and Network-Traffic, which are explained below.

CPU Utilization: It measures the percentage of allocated EC2 compute resources that are currently being used by the

instance. It calculates the processing power required to run an application upon a selected instance.

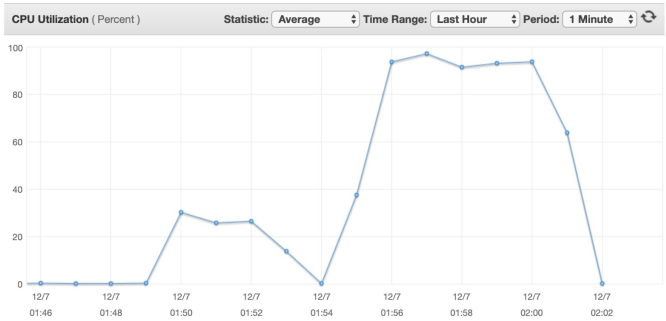

Fig. 1. CPU Utilisation of Master with 2 slaves

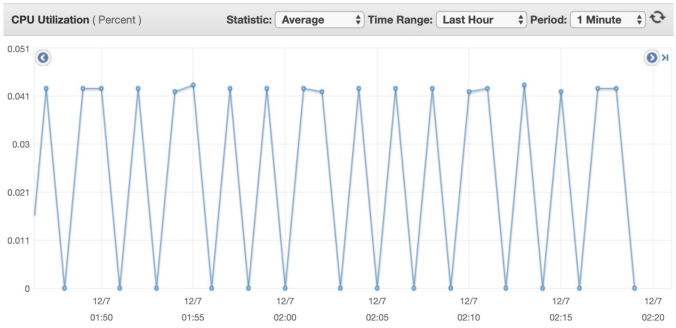

Fig. 2. CPU Utilisation of Slave 1 with 2 slaves

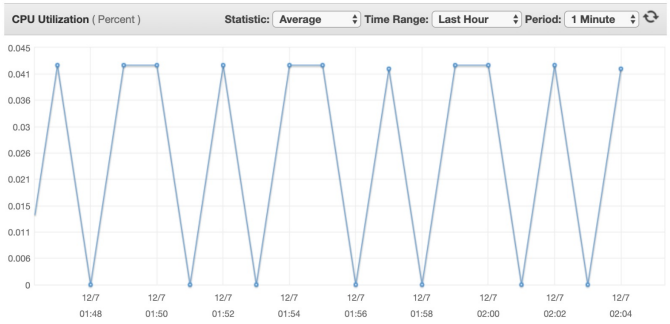

Fig. 3. CPU Utilisation of Slave 2 with 2 slaves

Network-Traffic: It is the number of bytes received on all network interfaces by the instance. It identifies the volume of incoming network traffic (the number of bytes) to a single instance.

From graphs, we find that the master node's CPU Utilization and Network-Traffic values are scaled at a higher rate than slave nodes. This is because the master node would be active all through the execution of the program since it constantly monitors the slave nodes, gets the intermediate results from them, and gives out the output.

Also, we find that the trend in CPU Utilization and Network-Traffic is the same between two slave nodes, which shows that the work is equally distributed between them, the master node ensures that the same amount of work is given to its slaves.

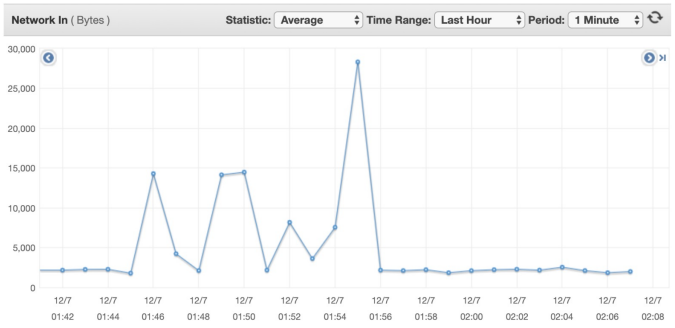

Fig. 4. Network Utilisation Master with 2 slaves

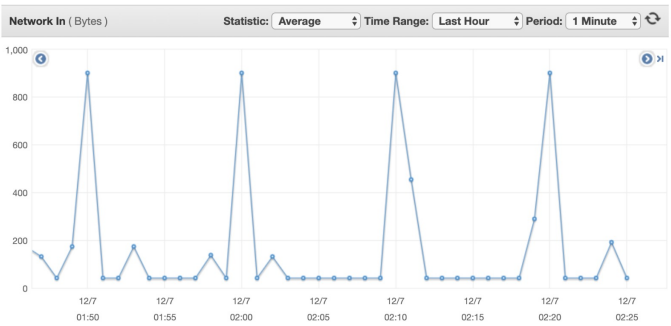

Fig. 5. Network Utilisation Slave 1 with 2 slaves

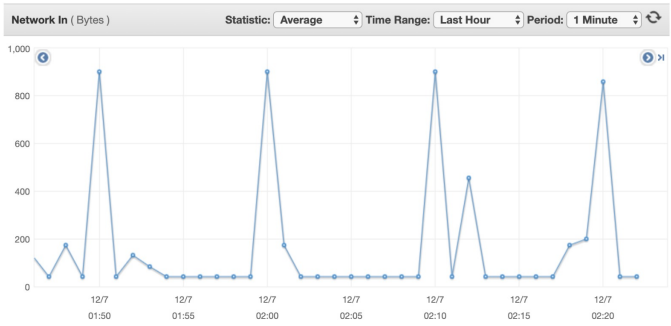

Fig. 6. Network Utilisation Slave 2 with 2 slaves

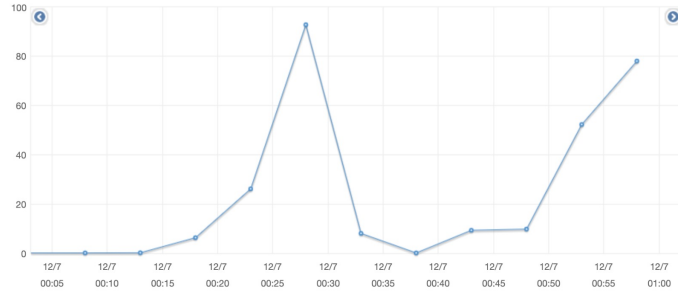

Fig. 7. CPU Utilisation Master with no slaves

We have considered three different variations by changing the number of nodes in the cluster. The variants are:

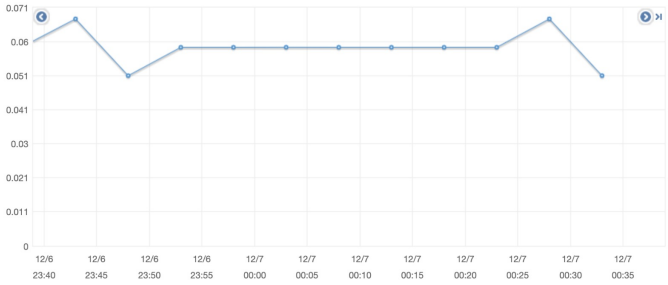

Fig. 8. CPU Utilisation Master with 1 Slave

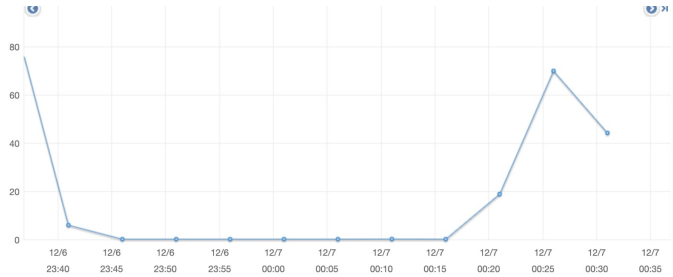

Fig. 9. CPU Utilisation Slave with 1 Slave

1. one master, two slaves
2. one master, one slave
3. one master, zero slaves

The difference that we observed is that the number of peaks reduced as we decreased the number of nodes. The reason could be because of the decreased number of interactions between the nodes as the number of nodes decreased.

IV. REFERENCES

[1]http://sigspatial2016.sigspatial.org/giscup2016/problem
[2]https://en.wikipedia.org/wiki/Apache_Hadoop
[3]https://en.wikipedia.org/wiki/Apache_Spark