

Azure Spark-Based Taxi Route Analysis

Marina Samprovalaki || f3322310

Konstantinos Kivotos || f3322304

Anastasios Angelidis || f3322301

Introduction

This technical report details the implementation and analysis of a scalable and fault-tolerant system for processing extensive taxi route data in New York City. Leveraging Microsoft Azure's cloud platform, the system utilizes Apache Spark and various Azure services to address complex queries on taxi route data. The report outlines the methodology, tools, and outcomes of this data processing system.

Technicalities

The implementation for each of the questions heavily relies on [PySpark](#), an interface for Apache Spark in Python. The code execution is facilitated using [Databricks](#), with the exception of Question 4, where a Function App was employed. Additionally, [Matplotlib](#) was utilized to create some of the graphs, while [Power BI](#) played a pivotal role in addressing Question 6.

The data handling approach involved saving datasets as blobs in a storage account. [Blob storage](#), a cloud-based storage service, was chosen due to its capability to efficiently store and manage large volumes of unstructured data.

[Apache Spark](#), a multi-language engine for data engineering, data science, and machine learning, was preferred for its scalability. The system can be scaled horizontally by utilizing more machines or clusters, enhancing the computational resources for larger datasets. Additionally, serverless functions employed in the project exhibit similar scalability. Azure Function app was chosen for its flexibility, ability to execute code without managing the underlying infrastructure, and automatic scaling of resources based on increased load.

Regarding scalability, the blob size can be increased to store more data in a single blob beyond the default 4 MiB. Apache Spark's horizontal scaling and the scalability of serverless functions provide flexibility in handling larger datasets efficiently.

In terms of fault tolerance, Azure's Blob Storage ensures high availability by replicating blobs across multiple machines. Apache Spark achieves fault tolerance through [Resilient Distributed Datasets \(RDDs\)](#), using an immutable graph of operations. In case of failure, operations can be retried by following the links to parent nodes in the graph. Serverless functions benefit from the fault tolerance mechanisms provided by the Azure cloud platform. Functions are designed to be retried in case of failure, ensuring adaptability to potential errors. Also, executing functions in parallel reduces the risk of delays due to failure in a single function.

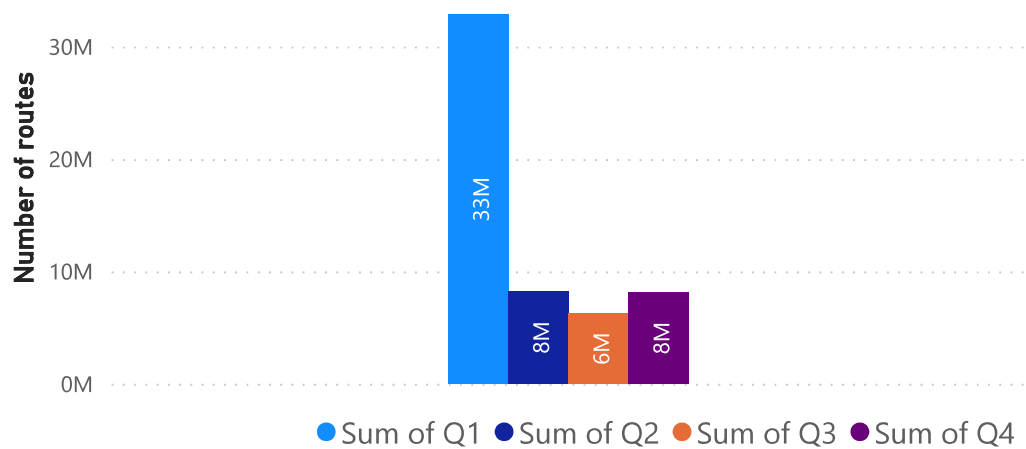
Data Preparation

The analysis initiated by connecting to Azure Blob Storage, where taxi route datasets were stored. After successfully mounting the Blob Storage, four distinct DataFrames (df1, df2, df3, and df4) were created, each based on specific quantiles rather than months. These individual DataFrames were strategically merged into a unified dataset (df) to provide a comprehensive perspective of the taxi route data based on quantile divisions.

1st Question - Geographical Quadrant Analysis

To address the first question, taxi routes were systematically categorized based on their pickup locations within defined geographical quadrants around a central point in the city. PySpark's DataFrame operations were employed to establish quadrant-specific conditions, creating new columns indicating the quadrant of each taxi pickup. Ride counts for each quadrant were aggregated, providing insights into taxi activity in different geographical regions. The processed DataFrame was persisted in the default database for lasting accessibility and query efficiency.

Number of routes per Quadrant



2nd Question - Route Metrics Calculation

For the second question, we identified routes that exceeded one kilometer in distance, lasted more than 10 minutes, and involved more than two passengers. By applying fundamental principles of physics, and considering the determined mean price of \$2.5 per kilometer for taxi drives, as well as assuming an average speed of 20 km/h, we computed the distance covered by each taxi route using the formula

$$u = \frac{x}{t}$$

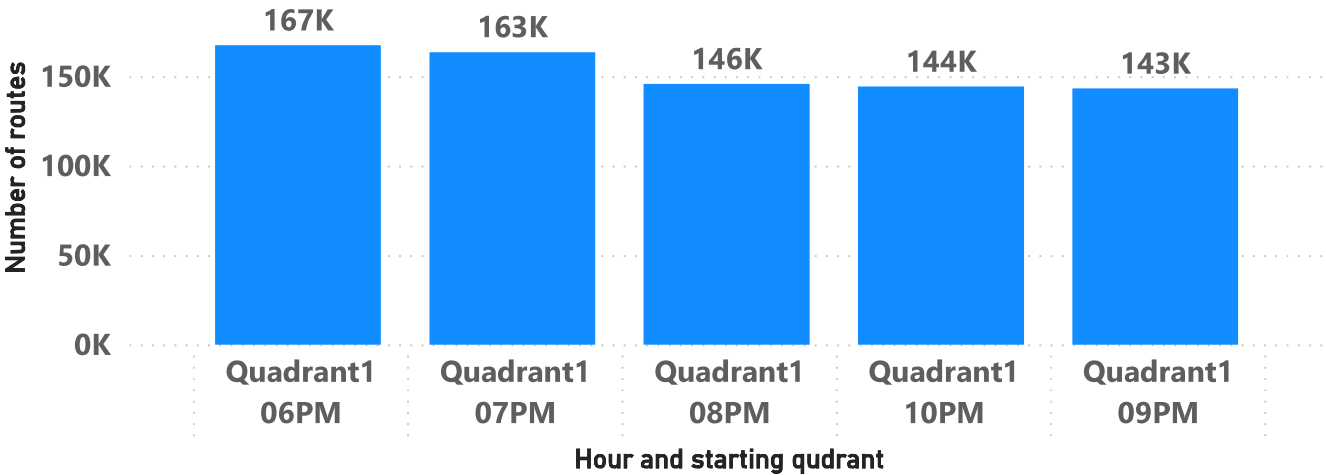
where u represents speed, x represents distance, and t represents time.

key	distance	fare_amount	passenger_count	time_minutes
30/6/2015 11:59:25 μμ	5,80	14,50	5	17,40
30/6/2015 11:59:14 μμ	8,62	21,54	6	25,86
30/6/2015 11:57:44 μμ	23,02	57,54	4	69,06
30/6/2015 11:57:21 μμ	7,20	18,00	5	21,60
30/6/2015 11:57:09 μμ	8,20	20,50	5	24,60
30/6/2015 11:56:31 μμ	5,40	13,50	5	16,20
30/6/2015 11:56:19 μμ	4,80	12,00	6	14,40
30/6/2015 11:56:12 μμ	13,42	33,54	5	40,26
30/6/2015 11:54:20 μμ	9,20	23,00	5	27,60
30/6/2015 11:53:56 μμ	3,80	9,50	3	11,40
30/6/2015 11:52:55 μμ	3,60	9,00	4	10,80
30/6/2015 11:52:23 μμ	3,80	9,50	5	11,40
30/6/2015 11:51:18 μμ	4,20	10,50	3	12,60
30/6/2015 11:51:17 μμ	5,20	13,00	4	15,60
30/6/2015 11:50:56 μμ	4,80	12,00	4	14,40

3rd Question - Temporal and Spatial Characteristics

The third question explored the temporal and spatial characteristics of filtered routes. Frequency analysis for each hour of the day identified the top five hours for route commencement. Additionally, spatial analysis determined the most popular quadrants from which these routes originated. The results were presented in tabular format, enhancing understanding of both temporal and spatial dynamics.

5 Most popular hours



4th Question - Streaming Analysis with Function App

The fourth question involved streaming analysis with a Function App. A blob trigger was employed to continuously check Blob Storage for new data. Upon detecting new blobs, the Function App reads the blob and calculates the routes that have started from each quadrant and then saves the results in a new blob, demonstrating the real-time analytical capabilities of the system.

5th Question - Location-Based Route Analysis

The fifth question assessed taxi routes within a 5 km radius of the location from Question 1. Routes that commenced within this radius and incurred a fare exceeding \$10 were identified. Precise PySpark DataFrame operations were implemented, and the results were grouped by the hour of pickup to discern patterns over time.

Total routes for central spot

