Real Time Visualization and Analysis of Taxi Routes

Harshit Hajela (haha4350), Nikhil Jain (nija5462)

Project Overview and Goals

The objectives of our project were to develop a service that can:-

1. Track the routes, location and paths of thousands of taxis in a busy city in real time.
2. Compute additional metrics like speed, total distance travelled and halt time for all vehicles over time
3. Provide an interactive visualization for the vehicle trajectories and other mentioned metrics via a web application

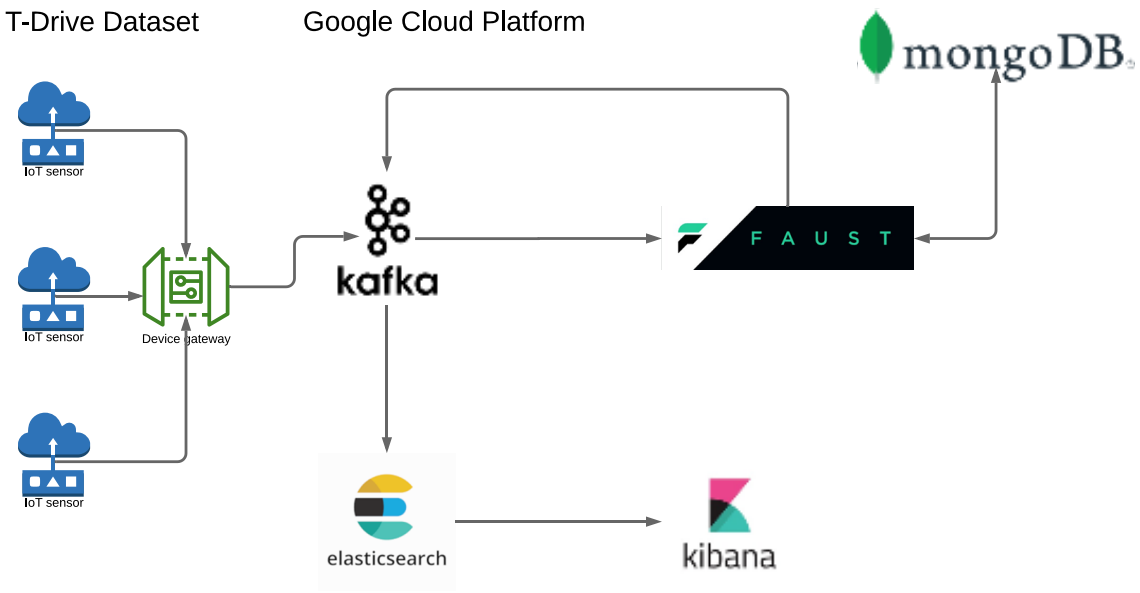
This was accomplished by aggregating and analysing the simulated streamed data from sensors on each unique vehicle which conveyed its location at short intervals of time.

While the intended use case for this service was to process actual sensor data streamed from vehicles in real time, for the purposes of implementing this project we simulated this by streaming from an existing dataset. The dataset in question is the Microsoft Research T-Drive trajectory data sample that contains one-week trajectories of 10,357 taxis in the city of Beijing.

To better simulate the intended real-life use case and investigate scalability concerns, we grew the size of the data using scripts to duplicate taxi location information by timeshifting existing taxi location data to create new virtual taxis.

The data was enriched by computing the discussed additional metrics and finally input to an interactive web-based dashboard that allows users to view and visualize this information for all the vehicles.

Components Used



Kafka Pub-Sub

Purpose:-

Here we are utilizing Kafka for two purposes:-

1. Collecting and buffering messages from the IoT sensors
2. As a pub sub mechanism to feed into two topics.

The first of these topics is consumed by a Faust Streams consumer (discussed in detail below) which computes the additional metrics discussed above and feeds the enriched data back into Kafka.

The second of these topics is consumed by Elasticsearch which takes as its input the enriched data feed that was computed earlier by Faust.

Advantages:-

Kafka is very low latency and offers high throughput in relation to its resource footprint which makes it ideal for high volume sensor data like ours which requires quick processing times. Additionally, it offers good persistence and fault tolerance capabilities which are pertinent to our requirements of computing metrics like speed, halt time and distance that depends on previously received data. Its wide acceptance and support is another advantage.

Disadvantages:-

Compared to simpler options like RabbitMQ, Kafka has a higher learning curve and requires a lot more configuration especially for non-trivial use cases. When things do go wrong, looking at the logs to analyse the cause of the malfunctions is not straightforward due to proprietary formats and multiple log locations.

Interaction with other components:-

In our architecture Kafka is the bridge and the message communication mechanism between all the other components. It is used to read messages from the IOT sensors and buffer them, to feed and get data from Faust stream and to provide this enriched data to another Faust stream to be fed into Elasticsearch.

Faust Streams

Purpose:-

We are utilizing Faust Streams for two purposes:-

1. As our primary stream processing system for computing additional metrics on the incoming sensor data
2. To act as a bridge between Kafka and components like MongoDB and Elasticsearch

Advantages:-

For our requirement of analysing and computing metrics on the sensor data we required stream processing operations. We had a lot of choice of frameworks to use here based on different paradigms and with different strengths and weaknesses but decided to go with Faust streams for which the biggest reason was it being written in Python which integrated well with our chosen clients for other components and our coding experience. Its simplicity and ease of setup was another major factor.

Disadvantages:-

Faust uses different terminology for a lot of the same stream processing concepts compared to other frameworks which can be slightly confusing initially.

Interaction with other components:-

Faust streams here acts as our bridge layer that relays messages from Kafka and Elasticsearch. Also, it is connected to MongoDB and uses it as a persistence store for its metric computation purpose.

MongoDB

Purpose:- We are using MongoDB here to serve as a NoSQL persistence store to be utilized by Faust stream processing for the computation of additional metrics like halt time, average speed etc. which are dependent on previously read sensor values.

Advantages:-

MongoDB has good performance characteristics, is very well documented and used widely with robust and easy to use Python clients.

Disadvantages:-

After performing more research we found we could have obtained even better performance with stores like RocksDB.

Interaction with other components:-

MongoDB is connected to Faust which utilizes it as a persistence store for computation.

Elasticsearch and Kibana

Purpose:- We are using Elasticsearch to serve as the data store for the enriched sensor data which Kibana can query and search to create interactive visualizations of the computed metrics and location data for end users.

Advantages:-

We're utilizing Elasticsearch for its high speed, scalability, distributed nature and fast and powerful searching capabilities. Kibana was then the logical choice for our visualization tool due to its strong integration with Elasticsearch and its highly interactive and powerful visualization capabilities which are great for gleaning information from data and identifying trends.

Disadvantages:-

Elasticsearch and Kibana are quite tightly coupled together and do not play so well with other visualization tools. Some of the visualization features in Kibana also leave a lot to be desired in terms of customizability.

Interaction with other components:-

Elasticsearch is connected to Faust streams and utilizes it to get the enriched location information messages from Kafka.

Capabilities and Limitations

1. Our project was capable of meeting our intended goal of visualizing the trajectories of many thousands of taxis in real time by aggregating sensor data from different sources and make our system scale with the size of the input
2. We were also able to compute additional metrics on the data and visualize them
3. Additionally, through the use of powerful visualization tools we were able to make these visualizations highly interactive and expressive
4. However, we did realize that such a service is highly dependent on the quality of data it uses as its input (which in real life would be the sensors) as incorrect data can often lead to erroneous values for metrics, thus reducing the efficacy of the tool.
5. Our chosen language, Python might have led to a slight overhead and reduced performance compared to a compiled language like Java