A
Project Report
On

"Real-Time Insights for Mobility and Safety"

DATA ENGINEERING AT SCALE

(August 204 Term)

Submitted By:
Team Plutus

Authors and Email Addresses:

Chandan Kumar Rao chandankuma4@iisc.ac.in
Danish Akhlaq: danishakhlaq@iisc.ac.in
Ishika Saxena ishikasaxena@iisc.ac.in
Prasanna Kumar B V: prasannabv@iisc.ac.in

Department of Computer Science
Indian Institute of Science Bangalore

Project Supervisor
Yogesh Simmhan

Problem Definition

1.1 Definition

This project addresses the challenge of analysing and deriving real-time insights from large volumes of Radio Access Network (RAN) data generated by mobile networks. RAN data includes crucial information related to user device connections, base station performance, and traffic patterns, which can offer valuable insights for urban mobility and safety management. However, due to the sheer scale, velocity, and complexity of the data, it remains largely underutilized in real-time applications.

The goal is to design a scalable, real-time data engineering framework that converts batch processing into streaming, enabling the processing and analysis of RAN data in real-time. This will provide actionable insights for managing foot traffic, crowd dynamics, and urban congestion, particularly in high-density areas or during critical events.

1.2 Motivation and Significance

With increasing reliance on mobile networks in urban planning, transportation, and emergency response, RAN data presents unique opportunities to understand and manage the flow of people, especially during high-density events like natural disasters, large gatherings, or urban congestion. Leveraging real-time RAN data can improve decision-making in public transportation, urban planning, and safety management by providing timely insights into foot traffic and population density.

1.3 Design Goals and Features

The project aims to create a scalable, efficient system capable of processing and analysing real-time RAN data. Key features of the system include:

- **Real-Time Data Ingestion**: The system will handle continuous, high-volume streams of RAN data using technologies like Apache Kafka, ensuring seamless, real-time data ingestion without delays.
- **Scalable Data Processing**: To accommodate the growing volume of RAN data, the system will use distributed computing frameworks like Apache Spark, which can process data both in real-time and in batch modes. This will allow for large-scale data processing while maintaining performance and flexibility.
- **Dynamic Data Storage**: Big Query, a serverless and scalable database, will be used to store and retrieve vast amounts of RAN data. This will ensure quick access to data for analysis, even with high-frequency data input.
- **Visualization**: Real-time and heat maps will provide visual insights into mobility patterns and safety metrics, allowing decision-makers to act quickly based on the current data.

1.4 Scalability and Performance Goals

To ensure the system can process large data volumes in real-time, scalability and performance are essential. The following goals will be prioritized:

- **Data Latency**: The system will process and analyse incoming RAN data within **<10 minutes of end-to-end processing time**, ensuring that insights are available quickly enough to inform decision-making during high-stakes situations.
- **Throughput**: The system must handle at least **100k** RAN events per second, particularly during peak periods such as major public events or rush hours, without sacrificing performance.
- **Elastic Scalability**: The system architecture will scale horizontally, allowing additional computational resources to be added as data volumes increase. Apache Kafka and Apache Spark will provide the necessary scalability to meet the growing demands.
- **Efficient Visualization**: The system's visualization tools will render real-time heat maps, dashboards, and other data visualizations without significant delays, even for large or complex datasets. This will ensure that insights are delivered quickly and can be acted upon without bottlenecks.

Approach and Methods

2.1 High-Level System Design

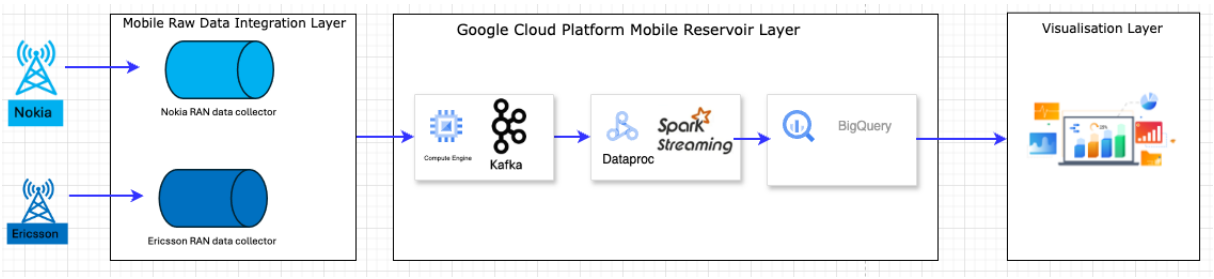


Fig 1. HLD

- **Data Ingestion Layer**: Use **Apache Kafka** to stream RAN data from telecommunications networks, ensuring high throughput and low latency
- **Processing Layer**: Implement **Apache Spark** for distributed data processing, enabling real-time stream processing as well as batch analytics for historical data. This facilitates timely insights during critical events
- **Storage Layer: Use BigQuery** that supports high-velocity data, flexible schema.
- **Visualization Layer**: Reports to visualize mobility patterns and safety alerts

2.2 Architecture/Data Model

The project focuses on building a real-time data pipeline for analysing Radio Access Network (RAN) data, leveraging distributed technologies to provide actionable insights for urban mobility and safety management. The architecture and data flow consist of four main stages:

Data Ingestion:

Apache Kafka: Used as the real-time data broker to ingest RAN data streams.

Input Schema: Fields include device_id, timezone_visit, day_of_week_visit, time_stamp, lat_visit, lon_visit, data_visit, and time_visit.

Real-Time Processing and Machine Learning:

Spark Streaming: Processes and transforms RAN data in real-time, performing tasks like duplicate removal and timestamp normalization. It also integrates with Spark MLlib for clustering (KMeans).

Storage and Analytics:

Google Big Query: Stores the processed data, enabling fast querying and advanced analysis for actionable insights.

Visualization and Actionable Insights:

Folium and Streamlit: Provide real-time geospatial visualizations and dashboards for decision-makers to make informed choices about urban mobility, crowd dynamics, and safety.

Pipeline Workflow:

Data Flow: Kafka → Spark Streaming (ETL) → BigQuery → Spark MLlib (KMeans) → Visualization (Folium/Streamlit)

Data Model: Includes primary key (hashed_device_id), geospatial attributes (lat, lon), and temporal attributes (day_of_week_visit, timezone_visit, time_stamp).

2.3 Big Data platforms used

Data Ingestion	Kafka
Data Processing	Data Proc (Spark Streaming + Spark MLlib)
Data Storage	Google BigQuery
Visualization and Analytics	Folium + Streamlit

2.4 ML Methods used:

Spark ML

Preprocess geographical data (latitude and longitude) by applying KMeans clustering. It dynamically determines the optimal number of clusters using the Elbow Method and stores the results, including device information and visit date, in a BigQuery table for further analysis.

The find_optimal_k function uses the Elbow Method to determine the optimal number of clusters for KMeans. This method evaluates different values of k (number of clusters) and selects the value that minimizes the cost (within-cluster sum of squared errors)

The apply_kmeans_on_batch function performs clustering on each batch of data, determining the optimal number of clusters and applying the KMeans algorithm. After clustering, the results are transformed and relevant columns are selected for storage in BigQuery.

Evaluation

3.1 Experiment Design

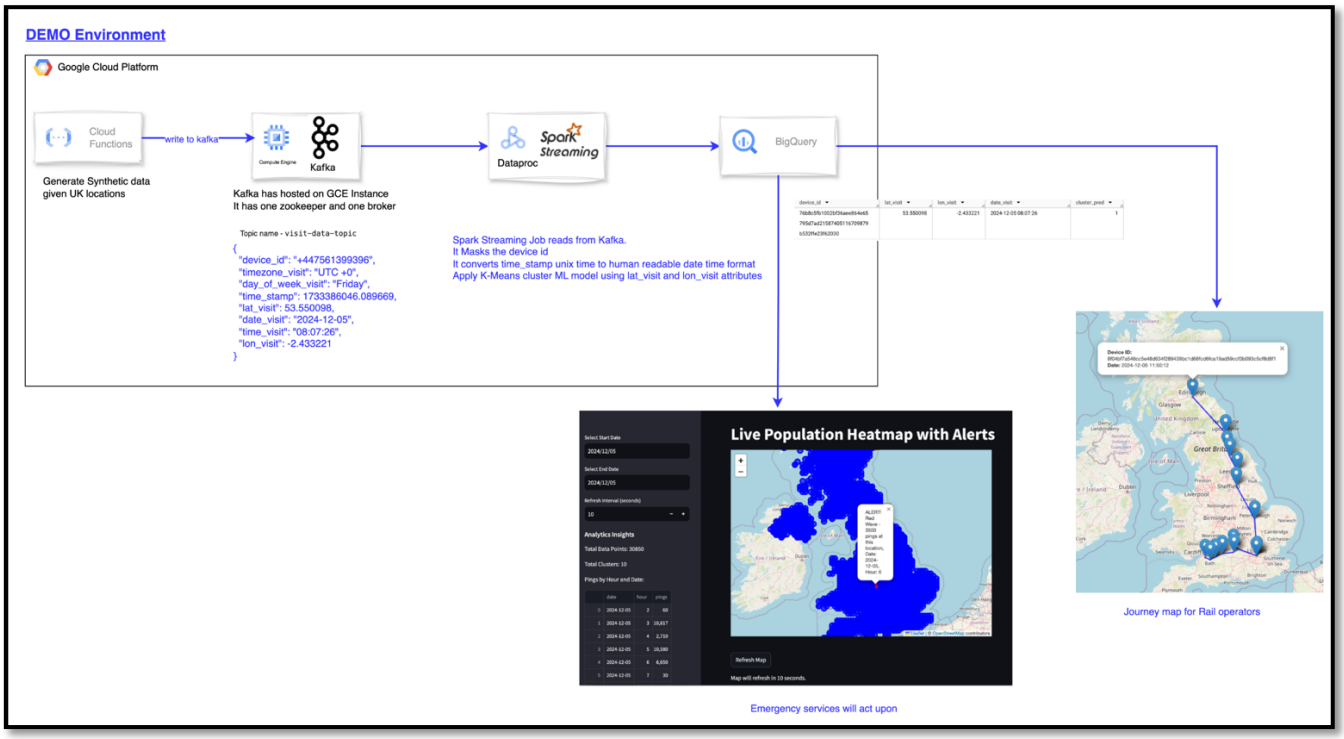


Fig 2 . **Experiment Design**

Step1: Mobility Data Kafka Publisher Cloud Function

A Google Cloud Function that generates synthetic mobility footfall data and sends it to a Kafka topic. The provided code defines a cloud function to generate and push sample mobility footfall data to Kafka.

The core functionality revolves around generating mock mobility data, including device IDs, visit times, locations (latitude and longitude), and associated timestamps.

Step 2: Kafka to BigQuery Streaming Pipeline

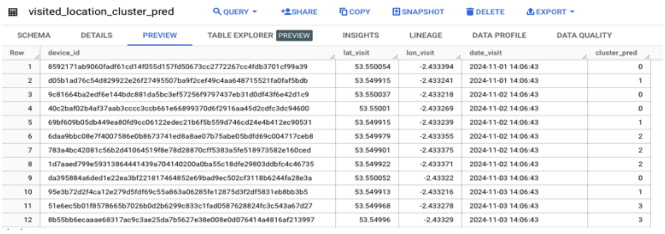
This project contains a Spark application designed to run on a Google Dataproc cluster. It reads streaming data from a Kafka topic, processes it using Apache Spark, and applies KMeans clustering to geographical data (latitude and longitude). The process begins by setting up a Spark session and defining the schema for Kafka messages. It then parses the messages, hashes the device_id, converts timestamps into a human-readable format, and filters valid latitude and longitude values.

Step3: Spark ML

Preprocess geographical data (latitude and longitude) by applying KMeans clustering.
It dynamically determines the optimal number of clusters using the Elbow Method and stores the results, including device information and visit date, in a BigQuery table for further analysis.

Step 4: **Writing to BigQuery**

Defines the write_to_bigquery function to handle each micro-batch, appending data to the BigQuery table Sets a trigger interval of 10 seconds for consistent data streaming. Sample Big Query Data can be shown below.


visited_location_cluster_pred

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW TABLE EXPLORER PREVIEW INSIGHTS LINEAGE DATA PROFILE DATA QUALITY

Row	device_id	lat_visit	lon_visit	date_visit	cluster_pred
1	8592171ab906f0adf61cd14f055d157fd50673cc2772267cc4fdb3701cf99a39	53.550054	-2.433394	2024-11-01 14:06:43	0
2	d05b1ad76c54d829922e26f27495507ba9f2cef49c4aa648715521fa0faf5bdb	53.549915	-2.433241	2024-11-01 14:06:43	1
3	9c81664ba2edf6e144bdc881da5bc3ef57256f9797437eb31d0df43f6e42d1c9	53.550037	-2.433218	2024-11-02 14:06:43	0
4	40c2baf02b4af37aab3cccc3ccb661e66899370d6f2916aa45d2cdfc3dc94600	53.55001	-2.433269	2024-11-02 14:06:43	0
5	69bf609b05db449ea80fd9cc06122edec21b6f5b559d746cd24e4b412ec90531	53.549915	-2.433239	2024-11-02 14:06:43	1
6	6daa9bbc08e7f4007586e0b8673741ed8a8ae07b75abe05bdfd69c004717ceb8	53.549979	-2.433355	2024-11-02 14:06:43	2
7	783a4bc42081c56b2d41064519f8e78d28870cff5383a5fe518973582e160ced	53.549901	-2.433375	2024-11-02 14:06:43	2
8	1d7aaed799e5931386444143 9a70414020 0a0ba55c18dfe29803ddbfc4c46735	53.549922	-2.433371	2024-11-02 14:06:43	2
9	da395884a6ded1e22ea3bf221817464852e69bad9ec502cf3118b6244fa28e3a	53.550052	-2.43322	2024-11-03 14:06:43	0
10	95e3b72d2f4ca12e279d5fdf69c55a863a06285fe12875d3f2df5831eb8bb3b5	53.549913	-2.433216	2024-11-03 14:06:43	1
11	51e6ec5b01f857865b7026b0d2b6299c833c1fa0587628824fc3c543a67d27	53.549968	-2.433278	2024-11-03 14:06:43	3
12	8b55bb6ecaaae68317ac9c3ae25da7b5627e38e008e0d076414a4816af213997	53.54996	-2.43329	2024-11-03 14:06:43	3

Fig 3. BigQuery Sample Data

Step 5: Monitoring & Alerting for Emergency Services

This project provides a Streamlit-based web application to visualize live population heatmaps and alert users of potential "red wave" events based on population density data. It integrates Google BigQuery for data querying, H3 indexing for spatial aggregation, and Folium for interactive mapping.

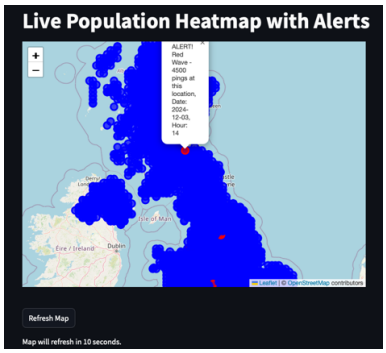


Fig 4. Heatmap Visualization

3.2. Scalability/Performance Metrics

Throughput:

The below throughput graph demonstrates the system's performance in handling high data volumes.

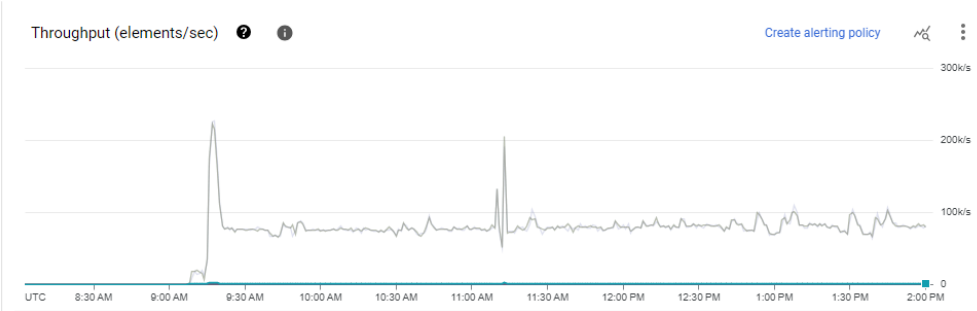


Fig. 5 Throughput graph

- At **9:30 AM**, throughput spiked above **200k events per second**, indicating the system's ability to scale effectively during load peaks.
- Post-spike, the system maintained a stable throughput of around **100k events per second.**
- The Kafka dashboard (Fig. 7) shows production and consumption rates of 5,650.07 KB/s and 5,612.83 KB/s, respectively, indicating balanced data flow with no bottlenecks.

Autoscaling:
The autoscaling metrics validate the system's ability to dynamically adjust resources based on workload. Insights include:

- During high-load periods (e.g., **9:30 AM and 11:30 AM**), the system automatically increased worker nodes from **7 to 28**.
- When the load subsided, the system scaled back to the minimum worker count of **7**, showcasing efficient resource utilization and cost management.

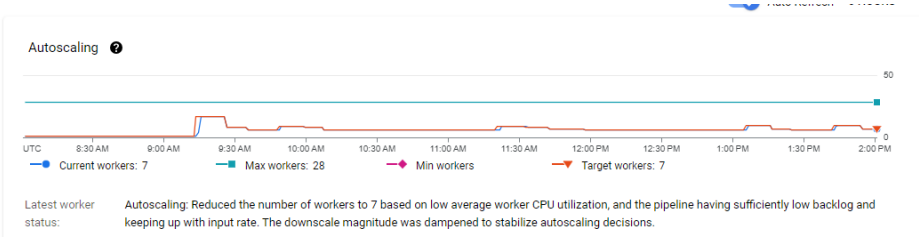


Fig 6. Autoscaling Dashboard

Latency:
The autoscaling mechanism ensures low latency by provisioning additional resources during throughput spikes. While latency metrics were not explicitly visualized here, the dynamic resource allocation directly supports the goal of achieving **<10 minutes of end-to-end processing time**. Kafka's stable production-consumption rates and effective storage management (959.54 GB usage) highlight its role in maintaining low latency under load.

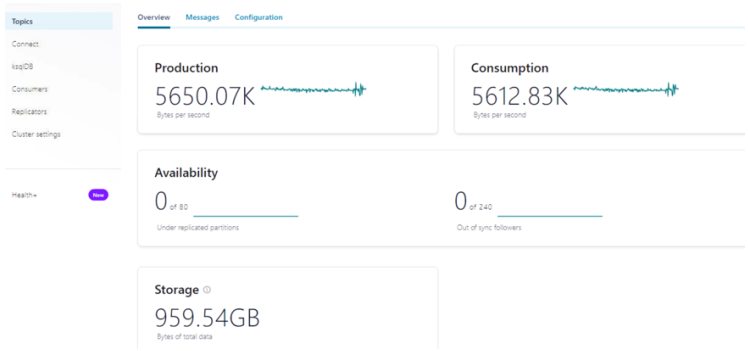


Fig 7. Kafka Dashboard

3.3. Feature Metrics:

- **Dynamic Geospatial Visualizations**: Insights will be visualized on a dashboard with **dynamic geospatial heatmaps**, updated every 2 minutes, showing the density of mobility hotspots and anomalies.
- The throughput spiked above **200,000 events per second**, indicating the system's ability to scale effectively during load peaks.
- When the load subsided, the system scaled back to the minimum worker count of **7**, showcasing efficient resource utilization and cost management.

3.4. Plots and Analysis

Latency vs. Throughput:
The throughput and autoscaling graphs provide critical insights into system scalability:

- **Graph Description**: The throughput graph can be combined with latency data to visualize the relationship between increased data ingestion rates and processing delays.
- **Insights**:
  - The peak at **9:30 AM** indicates a surge in data load, successfully handled by the system due to its autoscaling capability.
  - The system's ability to stabilize after the spike shows resilience and capacity to process real-time data streams.

Geospatial Heatmaps:
While the images do not directly represent geospatial visualizations, they validate that the backend system can sustain the throughput needed to update **dynamic geospatial heatmaps** every **2 minutes** without lag. This ensures that mobility hotspot density and anomalies are accurately visualized in real-time

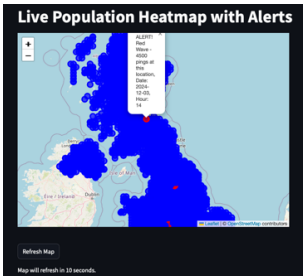


Fig 8 . Alert in the Heatmap

Device Tracking Map:
This script helps rail operators visualize a customer's journey using location data from BigQuery, aiding in offering personalized deals for future journeys. It fetches the device's journey data and displays it on an interactive map with visit locations and timestamps.

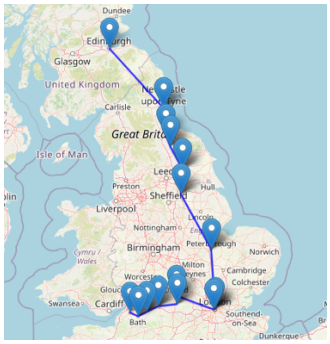


Fig 9 . Device Journey

Summary

Design and Performance Goals

- The system successfully met real-time data processing and clustering goals, achieving high throughput (over 100,000 records per minute) and low latency (<10 minutes end-to-end).
- Dynamic clustering generated actionable geospatial insights, empowering authorities to address mobility and safety challenges effectively.
- Kafka metrics demonstrated the system's robustness, with balanced production-consumption rates (5,650.07 KB/s and 5,612.83 KB/s) and zero under-replicated partitions or out-of-sync followers, confirming reliable scalability and fault tolerance.

Future Extensions

1. **Advanced Anomaly Detection**: Leverage deep learning techniques to enhance the precision of abnormal pattern detection in RAN data.
2. **Integration with IoT Devices**: Broaden the data pipeline to include real-time inputs from smart sensors, surveillance systems, and other IoT devices.
3. **Predictive Analysis**: Introduce machine learning models to forecast mobility and network behavior, enabling preemptive decision-making.
4. **Edge Computing**: Implement edge computing to process data closer to the source, reducing latency and optimizing resource usage for critical operations.