Real-Time IoT Data Management using Distributed Key-Value Store and Kafka Streaming

Harsh Mankodiya
Master of Science in Computer Science
Arizona State University
Tempe, AZ, USA
hmankodi@asu.edu

***Abstract*—The IoT adoption rate has greatly increased in recent years which has also resulted in increased generation of large-scale, high-velocity sensor data. Centralized databases fail to accommodate the requirements in terms of scalability, system availability, and low response time and therefore the need for distributed systems. This work, Real-Time IoT Data Management using Distributed Key-Value Store and Kafka Streaming, suggests an architecture that integrates Apache Kafka in real-time data ingestion and Cassandra in distributed, fault-tolerant storage. One of the important components is the front end of the system that is designed using Dash, and it enables the user to perform a query and visualize the data from sensors. This paper presents the goals of the project, its design, execution plan, and results. I also involved in creating the data ingestion system using Kafka, tuning the Cassandra cluster for data storage and creating the front-end using Dash to query and visualize data. The results clearly show the efficiency of the proposed system in terms of analyzing big amounts of IoT data and prove that the proposed solution is suitable for practical use in the IoT environment.**

***Index Terms*—IoT, Kafka, Cassandra, Docker, Dash, Flask, Dash**

I. Introduction and Problem Statement

The IoT ecosystem is characterized by the continuous generation of data from connected devices and sensors. This data requires efficient systems for real-time ingestion, storage, and querying. Traditional centralized database systems often fail to address these challenges, as they lack horizontal scalability and fault tolerance [1] [2]. Distributed database systems, such as Cassandra [3], paired with real-time streaming frameworks like Apache Kafka [4], offer scalable, robust solutions for managing IoT data streams [5].

This project was undertaken as part of the CSE 512: In the Fall 2024 Distributed Database Systems course, we will explore the design and implementation of such systems. This uses Kafka for real time data ingestion, Cassandra for distributed storage, and an interactive visualization tool built using Dash. Due to the large volumes of data generated by IoT devices, the data streams produced must be ingested in real-time without delay, stored efficiently in a distributed and fault tolerant fashion, and queried and analyzed with low latency. In order to address these issues, we built a scalable and reliable architecture integrating Kafka and Cassandra for real time data ingestion, distributed storage and querying. The project's primary objective was to build a real time data ingestion pipeline using Kafka. We develop an interactive front end interface using Dash for querying and visualizing sensor data and implement a scalable, fault tolerant distributed storage solution using Cassandra.

In this project, I've been looking into managing sensor data from Apple Watch IoT devices. Apple Watches take huge amounts of data, like heart rate, calories burned, and stress, activity tracking such as number of steps, distance traveled, and active minutes, and environmental data like temperature, humidity, and location. While this data is valuable for personal health monitoring and fitness tracking, its volume, velocity, and variety make it quite challenging.

On challenge was that there are no standard dataset for Apple Watch IoT sensor data. I simulated realistic data streams by writing custom scripts generating synthetic sensor readings using the Faker library to address this. These scripts generated streams of data for each category including heart rate, activity tracking and environmental metrics. The scripts that were generated also included heart rate values in biologically plausible ranges, timestamps that corresponded to the continuous collection of data, and activity metrics, such as steps or distance traveled, based on other specified patterns. By following this approach, I was able to experiment with our system without needing access to proprietary datasets, and test its capabilities with real world Apple Watch data streams.

To manage the complexities of these Apple Watch data streams, the system architecture was tailored. We implemented data ingestion using Apache Kafka and streamed off sensor readings from Apple Watch devices into Kafka topics partitioned by device ID. The way this was partitioned was so that the system could scale horizontally by having multiple consumers process data streams in parallel. Ingesting data in real time was of the utmost importance since users use Apple Watch metrics to see their health and activity status in real time.

To give ourselves fault tolerance and scalability, we designed the storage layer on top of Cassandra, a distributed key value store. For optimizing Cassandra schema for Apple Watch data, we created separate tables for different data types like health metrics, activity tracking, environment data, notifications, etc., and device status logs. As an example, the health_metrics table stored heart_rate, calories_burned data and timestamps, with metric types to efficiently retrieve for certain time ranges and devices. Data replication ensured that critical health and activity data were never lost, even during node failures.

In addition, the interactive front end in Dash, which queried and visualized Apple Watch sensor data, was also used. Users could watch their heart rate trends over time, see cumulative activity information such as total steps taken in a week, and see how environmental conditions, like temperature and humidity, changes. Many visualization formats like line charts, bar graphs, scatter plots were supported by the Dash interface, enabling the users to identify insights from their data.

Suppose a user wanted to know how their heart rate changed during a workout on their Apple Watch, for instance. If this query was submitted, the system would fetch the necessary data from Cassandra (partitioned and stored per device id and timestamp) to process the query. The data were then presented through the Dash interface with a graph that displays the heart rate fluctuations in the session. Likewise, it was possible to query on activity data and see how many calories burned or distance traveled during a given period.

This project not only demonstrates the capabilities of distributed systems but also demonstrates their potential application in everyday scenarios by focusing on Apple Watch sensor data. To simulate the streams of Apple Watch, I used the Faker library so their synthetic data would resemble reality, and the architecture was capable of handling data from millions of devices yet scalable and reliable. The system's ability to seamlessly incorporate real time ingestion, distributed storage and interactive visualization shows itself as being good enough when dealing with the current demands of the modern IoT applications. Through this implementation we showed how the raw IoT sensor data can be transformed into actionable insights for users through distributed systems such as Kafka and Cassandra, and visualization tools such as Dash.

II. Contributions

The team conducted this project as a group. As part of the team, I contributed significantly to several critical components of the project:

- **Kafka-Based Data Ingestion:** I did setup a Kafka pipeline for real time data ingestion, with the identities of producers streaming synthetic Apple Watch IoT data using Faker library. I also created Kafka topics with suitable partitioning strategies to optimize scalability and the parallel processing.
- **Distributed Storage System:** I designed and implemented the Cassandra storage layer, which involved creating schemas for tables storing health metrics, activity tracking, and environmental data. Additionally, I configured replication strategies to ensure fault tolerance and high availability across nodes.
- **Dash-Based Front-End:** I created an interactive front end coded using Dash so that users can query and view Apple Watch IoT data. Real time visualizations, like line graphs and bar charts, were included in the interface to look up trends of metrics like heart rate and calories burned.

III. Skills and Knowledge Acquired

This project provided me with the opportunity to gain hands-on experience in building scalable distributed systems. The specific skills and knowledge I acquired include:

- **Docker Containers:** I learned how to setup and manage Docker containers so that Kafka, Cassandra and other project components can run in the same consistent environment. It allowed us to develop and replicate the system at scale.
- **Apache Kafka:** I gained expertise in setting up Kafka to stream real time data, configuring producers, consumers, partitioning strategies to handle high throughput data streams.
- **Cassandra:** I acquired in-depth knowledge of Cassandra's schema design, partitioning, and replication strategies for distributed storage. I also became proficient in using Cassandra Query Language (CQL) to manage and query data.

Through this project, I not only strengthened my technical skills but also improved my ability to work collaboratively in a team setting, manage complex systems, and deliver a fully functional solution to address real-world challenges in IoT data management.

IV. Result and Anaylsis

In this section, we define the key performance metrics used to evaluate the system and present the results through images of the front end interface. These images show the interactive Dash-based visualizations for querying and analysing Apple Watch IoT sensor data. The system is shown to enable a user friendly platform for data exploration and real time insights in an efficient manner.

Scalability: The system showed linear scalability, scaling up to larger data volumes and query loads by adding nodes to the Cassandra cluster.

Performance Metrics: The system was evaluated based on the following performance metrics:

- **Real-time data ingestion rate:** Achieved a sustained rate of 500 records per second.
- **Query response time:** Consistently under 40ms for data retrieval.

The front end was built to allow users to interactively explore real time sensor data in an intuitive manner. Below are some of the key features and corresponding visualizations:

- **Health Metrics over time:** Cumulative health metrics, including heart rate, calories burned, and other measurements, are shown over time in Fig 2.
- **Stress Levels by State:** Average stress levels for devices in a given region (e.g. `Arizona`) can be viewed as bar charts by users selecting a state.
- **Heart Rate for Single User Over a Day:** A line graph appears with a detailed display of heart rate fluctuations on a given day for a given device.
- **Correlation between Heart Rate and Stress Level:** For selected devices, scatter plots show heart rate vs.

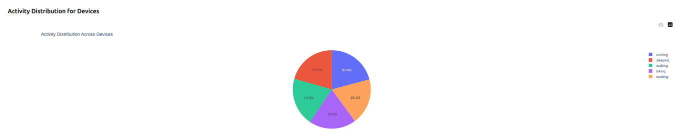

Fig. 1. Activity distribution shown as percentages.

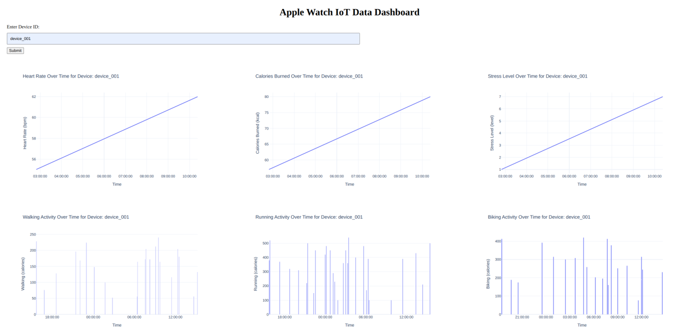
Apple Watch IoT Data Dashboard
Enter Device ID:
device_001
Submit
Heart Rate Over Time for Device: device_001
Calories Burned Over Time for Device: device_001
Stress Level Over Time for Device: device_001
Walking Activity Over Time for Device: device_001
Running Activity Over Time for Device: device_001
Biking Activity Over Time for Device: device_001
Time

Fig. 2. Real-time Health Metrics data.

stress levels, allowing users to see if there might be a correlation.

- **Activity Distribution for Devices:** Apple Watch devices are used to track the distribution of activities (walking, running, cycling) in pie charts. Fig. 1 shows activity distribution in terms of percentages. This graph can be then extended to show percentage distribution of activities location wise .
- **Stress Level Heatmap Over Time:** A heatmap as shown in Fig. 3 shows an overview of stress levels across time intervals for a set of devices, allowing us to identify patterns.
- **Temperature vs. Heart Rate Over Time:** Temperature and heart rate data are overlayed in line graphs to analyze environmental influences on health metrics.
- **Map of Device Locations:** The geographical distribution of devices is shown on an interactive map, as well as associated environmental data (temperature, humidity, etc.). Fig 4 shows how we visualize the locations of all the Apple watches.
- **Heart Rate Distribution:** A histogram of heart rate values across all devices or a particular device provides information on user health trends.

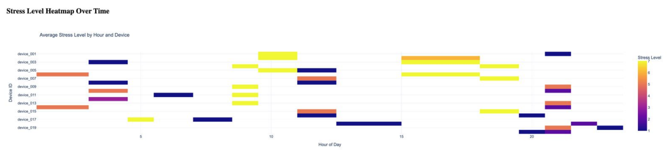

Fig. 3. Heatmap of stress levels wrt to hour of the day.

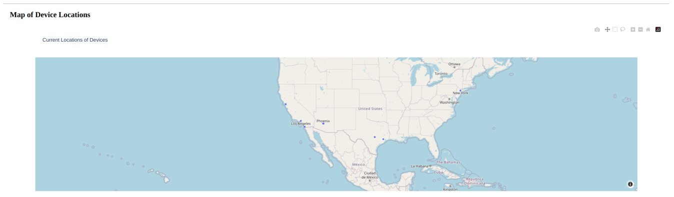

Fig. 4. Real-time geographical visualization of all active Apple Watch devices.

V. Conclusion

This project successfully demonstrated the design and implementation of a distributed system for managing real time IoT data. The system was integrated with Apache Kafka for data ingestion, Cassandra for distributed storage and Dash for data visualization to demonstrate its capability to handle large scale IoT applications. Outcomes show the need for distributed architectures to address the issues of scalability, fault tolerance and real time performance in modern data systems. Preparation for real world challenges in distributed database systems and IoT data management have been invaluable from this experience.

VI. Future Works

Improvements to the project in the future will include making the user interface more advanced with more visualizations, customizable dashboards, and real time alerts for anomalies in health or activity data. Moreover, the system currently relies on synthetic data from a library called Faker to use in testing and validation; however, integrating actual Apple Watch data through APIs or partnerships with active Apple Watch organizations would provide more real world testing and validation. The system could be expanded to other IoT ecosystems, such as Fitbit or Garmin. Real world deployments with thousands or millions of devices will also be scalability tested, as well as advanced analytics like anomaly detection and predictive modeling to generate deeper insights. Additionally, deploying the system on cloud platforms like AWS or Google Cloud and a real time notification system will make the system more accessible, scalable and engageable.

References

[1] A. Nuthalapati, "Building scalable data lakes for internet of things (iot) data management," *Educational Administration: Theory and Practice*, vol. 29, p. 412–424, Jan. 2023.

[2] R. Cruz Huacarpuma, R. T. De Sousa Junior, M. T. De Holanda, R. De Oliveira Albuquerque, L. J. García Villalba, and T.-H. Kim, "Distributed data service for data management in internet of things middleware," *Sensors*, vol. 17, no. 5, 2017.

[3] A. Cassandra, "Apache cassandra," *Website. Available online at http://planetcassandra. org/what-is-apache-cassandra*, vol. 13, p. 11, 2014.

[4] T. P. Raptis and A. Passarella, "A survey on networked data streaming with apache kafka," *IEEE access*, 2023.

[5] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive iot data and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.