Term: Fall 2024 **Subject:** Computer Science & Engineering (CSE) **Number:** 512

Course Title: Distributed **D**atabase **S**ystems (CSE 512)

GROUP PROJECT PROPOSAL

Project Title:

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

Team Name and Members:

Team Name: DistributedEngineers

Team Members:

- Aayush Dipenkumar Kansara (ASU ID: 1231954440)
- Devdutt Dhavalkumar Thakkar (ASU ID: 1229365087)
- Harsh Nitinbhai Mankodiya (ASU ID: 1228756128)
- Saurin Anilkumar Prajapati (ASU ID: 1234209355)

1. Introduction

1.1 Background

With the rise of the Internet of Things (IoT), the need for scalable and efficient data management systems has become critical. IoT sensors generate massive amounts of data that need to be processed, stored, and queried in real time for

insights. Distributed database systems play a key role in managing the scalability, fault tolerance, and availability of such systems. Traditional centralized databases often fail to meet these demands, while distributed key-value stores (e.g., Cassandra, HBase) offer an effective solution for handling large-scale sensor data with horizontal scaling.

This project aims to design and implement a real-time IoT data management system using distributed key-value store technologies and real-time streaming via Apache Kafka.

1.2 Problem Statement

The growing volume and velocity of IoT sensor data present a challenge for real-time data ingestion, storage, and retrieval. Existing centralized systems struggle with scaling, and real-time processing is often difficult to achieve in traditional databases. This project addresses the need for a scalable, fault-tolerant system capable of ingesting high-velocity IoT data streams and providing real-time access to sensor readings. The system will ensure that sensor data can be partitioned, replicated, and queried efficiently.

1.3 Objectives

- Design and implement a distributed key-value store for IoT sensor data management.
- Stream sensor data in real-time using Apache Kafka.
- Evaluate system performance under different loads and conditions.
- Explore techniques for data partitioning, replication, fault tolerance, and consistency.
- Provide real-time querying capability for IoT sensor data.

2. Project Description

2.1 System Design

The system will use Apache Kafka for real-time data streaming and Cassandra (a distributed key-value store) for storing and managing IoT sensor data. Kafka will act as the data ingestion layer, accepting sensor readings from various devices, while Cassandra will store the data in a fault-tolerant, scalable manner. We will implement partitioning based on sensor IDs and timestamps, ensuring efficient querying and fault tolerance through data replication across multiple nodes.

Key Components:

- Data Ingestion Layer: Apache Kafka will stream sensor data to multiple topics.
- Storage Layer: Cassandra will store the data in a distributed fashion, ensuring partitioning and replication.
- API Layer: A Flask-based REST API will provide query access to the stored data.

2.2 Implementation Plan

- Programming Languages: Python for Kafka producer/consumer, Flask for the API.
- Databases: Cassandra for distributed key-value store.
- Tools & Frameworks:
    - Kafka for real-time data streaming.

    - Cassandra for storage.
    - Docker for environment setup and containerized services.

- Libraries:
    - Kafka-Python (for Kafka producer/consumer).
    - Cassandra-Driver (for connecting Python to Cassandra).

2.3 Data Strategy

We will generate synthetic IoT sensor data (e.g., temperature, humidity, motion, etc.). The data will include sensor ID, timestamp, sensor type, and sensor readings. Data privacy concerns are minimal as we use synthetic data, but the architecture will support secure storage and streaming in real-world applications, including potential encryption of sensitive sensor data.

3. Methodology

3.1 Technique 1: Data Partitioning

We will implement data partitioning based on sensor ID and timestamp to allow efficient distribution of data across the Cassandra cluster. Each node in the cluster will store a subset of the sensor data, providing scalability.

3.2 Technique 2: Data Replication

Cassandra's built-in replication strategy will ensure data is replicated across multiple nodes to provide high availability and fault tolerance. We will explore different replication strategies (SimpleStrategy vs. NetworkTopologyStrategy).

3.3 Technique 3: Real-Time Streaming with Apache Kafka

Kafka will be used to ingest IoT data in real time. Kafka's partitioning and fault tolerance mechanisms will ensure the system can handle high-throughput data ingestion from multiple IoT devices.

3.4 Technique 4: Consistency Management

We will explore different consistency levels provided by Cassandra (e.g., ONE, QUORUM, ALL) to manage trade-offs between data consistency, availability, and latency.

3.5 Technique 5: Query Optimization

We will optimize queries using time-based indexes and partitioning strategies to ensure low-latency access to sensor data based on sensor ID and time range.

4. Evaluation Plan

4.1 Metrics for Evaluation

We will evaluate the system based on:

- Performance: Query response time and throughput under different workloads.
- Scalability: Ability to handle increasing data volume and queries.
- Fault Tolerance: System behavior when nodes fail or when there is network partitioning.
- Latency: Real-time processing delays in Kafka and data ingestion into Cassandra.

4.2 Expected Outcomes

- A fully functional IoT data management system capable of real-time data ingestion and querying.
- A scalable, fault-tolerant distributed database solution using Cassandra.
- Demonstrated ability to handle high-volume IoT data streams via Kafka.
- Insights into trade-offs between consistency, availability, and performance in distributed systems.

5. Timeline

Include a timeline with key milestones and deadlines. You can use a table format or a Gantt chart to illustrate your schedule.

Milestone	Task	Start Date	End Date	Team Member
Project Planning and Research	Drafting problem descriptions and data strategies	10/01/2024	10/04/2024	Aayush, Harsh
	System design outline and methodology	10/04/2024	10/08/2024	Saurin, Devdutt
	Proposal review and final edits	10/08/2024	10/10/2024	All Members
System Design And Infrastructure Setup	Detailed system architecture and diagrams	10/14/2024	10/18/2024	All Members
	Installing Cassandra, Setting up data models	10/18/2024	10/20/2024	Aayush, Harsh
	Installing Kafka, Configuring Kafka cluster	10/18/2024	10/20/2024	Saurin, Devdutt
Data Generation and Streaming	Creating data generation scripts, Configuring data streaming in Kafka	10/22/2024	10/26/2024	Aayush, Saurin
	Integrating Cassandra with Kafka for real-time streaming	10/26/2024	10/30/2024	Harsh, Devdutt

API Development (Query Layer)	Building query layer API, Writing unit tests for the API	11/01/2024	11/14/2024	All Members
Testing and Performance Evaluation	Conducting load testing, Measuring response times	11/14/2024	11/22/2024	All Members
	Identifying bottlenecks, Optimizing performance	11/14/2024	11/22/2024	All Members
Documentation and Final Report	Writing project report, Creating video presentation	11/22/2024	11/30/2024	All Members

6. Conclusion

This project explores the design and implementation of a scalable and fault-tolerant distributed key-value store for IoT sensor data management. By leveraging Apache Kafka for real-time data streaming and Cassandra for distributed storage, we aim to build a system capable of handling large volumes of IoT data in real-time. The project's outcomes will provide valuable insights into the challenges of managing distributed systems and the trade-offs between consistency, availability, and scalability in distributed databases.

References

- Nuthalapati, A., 2023. Building Scalable Data Lakes For Internet Of Things (IoT) Data Management. Educational Administration: Theory and Practice, 29(1), pp.412-424.
- Cruz Huacarpuma, R., de Sousa Junior, R.T., De Holanda, M.T., de Oliveira Albuquerque, R., García Villalba, L.J. and Kim, T.H., 2017. Distributed data service for data management in internet of things middleware. *Sensors*, *17*(5), p.977.
- OpenAI. (2023). ChatGPT (Oct 2024 version) [Large language model]: https://chat.openai.com/chat