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

Course Title: Distributed Database Systems (CSE 512)

GROUP PROJECT PROPOSAL

Project Title: Geo-Distributed Logistics Management System

Team Name: Shard Squad

Team Members:

Bimal Gajera (1229502575)
Dhairya Jadav (1229479695)
Dev Patel (1229417087)
Jaynil Vaidya (1228984473)

Submission Date: 10/12/2024

1. Introduction

1.1. Background

Logistics and supply chain management systems tackle substantial hurdles, including real-time shipment monitoring, inventory management, and system resiliency across several sites. Traditional centralized systems need more scalability and fault tolerance, resulting in inefficiencies.

Distributed database systems provide a solution by partitioning and replicating data across multiple nodes to ensure real-time data availability, scalability, and fault tolerance.

1.2. Problem Statement

As operations expand, traditional logistics systems have performance constraints, a lack of fault tolerance, and inconsistency in data management. These challenges come from the systems' centralized design, which struggles with scalability and cannot recover efficiently after failures.

Our project aims to address this gap by developing a distributed system for real-time logistics and supply chain management that includes geo-distributed databases, fault tolerance, and improved shipment tracking and fleet management performance.

1.3. Objectives

- Design a scalable and fault-tolerant distributed logistics and supply chain management database system.
- Implement data partitioning and replication strategies to ensure high availability and data consistency.
- Enable efficient query processing across distributed nodes for inventory management and shipment tracking.
- Evaluate the system's performance in terms of scalability, fault tolerance, and query response time.

2. Project Description

2.1. System Design

The proposed system will consist of distributed nodes representing various logistics hubs or warehouses, each managing data relevant to its specific location. The system will address key challenges through the following design components:

- **Data Partitioning:** We will use geographic partitioning to divide data based on regions, ensuring each node manages data relevant to its specific location.

- **Replication:** Critical data, such as shipment statuses and inventory levels, will be replicated across nodes to ensure fault tolerance.
- **Query Processing:** A distributed query engine will process inventory checks, shipment tracking, and fleet management by querying multiple nodes.

2.2. Implementation Plan

- **Programming Languages:** Python will be used for backend logic.
- **Databases:** The project will utilize PostgreSQL and MongoDB to manage distributed data. PostgreSQL will handle structured data with support for partitioning, while MongoDB will manage unstructured data with built-in geo-distributed capabilities to enhance scalability and fault tolerance.
- **Frameworks:** Apache Kafka will simulate real-time data streaming across the nodes.
- **Tools:** Docker containers will ensure scalability and portability across different environments.

2.3. Data Strategy

- **Data Selection**

  We will use a combination of real-world datasets and synthetic data:

  - **Open-Source Data:** We will reference publicly available datasets such as the USAID Global Health Supply Chain Program dataset and Mendeley's logistics dataset. These datasets provide relevant logistics information, such as shipment and supply chain performance data.
  - **Synthetic Data:** We will generate synthetic logistics data using AI tools like ChatGPT to simulate additional scenarios. This will allow us to create datasets representing logistics operations.

- **Data Privacy and Compliance**
    - **Data Encryption:** Sensitive data, such as customer details, will be encrypted at rest and in transit to ensure data protection.
    - **Compliance:** The system will comply with privacy regulations, and any personal information in synthetic data will be anonymized to protect privacy during testing.

3. Methodology

3.1. Data Partitioning

We will implement geographic partitioning to distribute data across regions, ensuring that each node stores and manages data related to its specific geographical area. For example, warehouses or logistics hubs in different locations will manage inventory and shipment data specific to their region. This approach reduces latency for region-specific queries and balances the system's load across nodes.

3.2. Distributed Query Processing

The system will feature a distributed query engine to handle queries across multiple nodes. Based on geographic partitioning, inventory checks, and shipment tracking will be executed by distributing the query to the relevant nodes. The query engine will aggregate results from multiple nodes, optimizing for speed and performance.

3.3. Horizontal Scalability

Using Docker containers, the system will be designed for horizontal scaling. This allows nodes to be added or removed, ensuring the system can handle growing operational demands while maintaining consistent performance.

4. Evaluation Plan

4.1. Metric for Evaluation

- **Response Time**: The time the system takes to process and return query results, especially for common operations like inventory checks and shipment tracking.
- **Throughput**: The number of queries the system can handle per second under different load conditions ensures that the system scales well and the number of requests increases.
- **Scalability**: The ability to maintain performance as new nodes are added. This will be measured by observing system behavior under increased load and data volumes as additional nodes are introduced.
- **Fault Tolerance**: The system can handle node failures without service disruption. We will simulate node failures and measure the system's recovery time and data consistency during the failure and recovery periods.

4.2. Expected Outcomes

- **High Performance**: The system should provide low-latency query responses, even under high load, due to the distributed query engine and geographic partitioning.
- **Resilience**: The fault-tolerant architecture should ensure no data is lost and that operations continue seamlessly during node failures. Replication strategies will ensure data availability across multiple nodes.
- **Efficient Resource Management**: Using Docker for horizontal scaling, the system should handle dynamic changes in demand without performance degradation.

5. Timeline

Milestone	Task	Start Date	End Date	Team Members Responsible
Project Proposal	Drafting problem description and data strategies	10/01/2024	10/05/2024	Bimal Gajera, Dhairya Jadav
	System design outline and methodology	10/06/2024	10/10/2024	Dev Patel, Jaynil Vaidya
	Proposal review and final edits	10/11/2024	10/13/2024	All members
System Design	Detailed system architecture and diagrams	10/14/2024	10/20/2024	Bimal Gajera, Dhairya Jadav
	Data partitioning and replication strategies	10/21/2024	10/25/2024	Dev Patel, Jaynil Vaidya
	Design review and feedback	10/26/2024	11/03/2024	All members
Implementation Phase	Backend development (Python)	11/04/2024	11/10/2024	Bimal Gajera, Dhairya Jadav
	Database implementation (PostgreSQL, MongoDB)	11/04/2024	11/10/2024	Dev Patel, Jaynil Vaidya
	Distributed query engine	11/11/2024	11/15/2024	Dev Patel, Jaynil Vaidya
	Docker containerization for scalability	11/16/2024	11/20/2024	Bimal Gajera, Dhairya Jadav
Testing and Evaluation	Performance analysis (response time, scaling)	11/21/2024	11/23/2024	Bimal Gajera, Dhairya Jadav
	Load testing and fault tolerance evaluation	11/24/2024	11/26/2024	Dev Patel, Jaynil Vaidya
Final Report and Submission	Writing and compiling the project report	11/27/2024	11/29/2024	Bimal Gajera, Dhairya Jadav
	Creating video presentation	12/30/2024	12/01/2024	Dev Patel, Jaynil Vaidya

6. Conclusion

This project addresses major logistics and supply chain management issues by developing a scalable, fault-tolerant, and geo-distributed database system. We will achieve high availability and performance across numerous logistical hubs by implementing data partitioning, replication techniques, and distributed query processing. The system can meet rising operational needs by adopting real-time data streaming and containerization for scalability while remaining resilient. The project will be evaluated using important metrics such as response time, fault tolerance, and scalability to ensure the system is reliable and suitable for practical implementation.

References

USAID GHSC-PSM Health Commodity Delivery Dataset

https://data.usaid.gov/Health/USAID-GHSC-PSM-Health-Commodity-Delivery-Dataset/tikn-bfy4

DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS

https://data.mendeley.com/datasets/8gx2fvg2k6/3

ChatGPT

https://chatgpt.com/

PostgreSQL

https://www.postgresql.org/docs/

MongoDB

https://www.mongodb.com/docs/