Distributed Data Management System

with Dynamic Load Balancing and Sharding

Abstract

This project aims to develop a distributed data management system that seamlessly integrates advanced load balancing techniques with efficient data sharding strategies. Our architecture addresses the challenges of maintaining high availability, data consistency, and scalability in modern distributed computing environments.

The architecture will consist of a customizable load balancer employing consistent hashing to distribute client requests evenly across multiple server instances, alongside a sharded database system that partitions data across several server containers to enable horizontal scaling. By integrating these components, our system will offer improved throughput, resource utilization, and fault tolerance.

Key features will include dynamic scaling capabilities, automatic failure detection and recovery, parallel read/write operations across different shards, and replica management for data redundancy. The system will be containerized using Docker to ensure easy deployment and scalability.

Through this project, we aim to demonstrate how the combination of load balancing and data sharding can create a resilient and high-performance distributed system capable of handling large-scale data processing requirements.

Weekly Work Plan

Week 1: System Design and Initial Implementation

- Design system architecture integrating load balancer and sharded database
- Implement basic server functionality for handling requests
- Set up Docker environment for containerization
- Develop consistent hashing mechanism for the load balancer

Week 2: Load Balancer and Core Functionality

- Complete load balancer implementation with request routing
- Implement server health monitoring and failure recovery
- Develop APIs for managing server instances (add/remove)
- Create test harness for load balancer performance evaluation

Week 3: Sharded Database Implementation

- Implement data sharding mechanism across server containers
- Develop shard replication for improved read performance
- Create APIs for data operations (read/write/update/delete)
- Implement mechanisms for maintaining data consistency across replicas

Week 4: Integration and Testing

- Integrate load balancer with sharded database system
- Perform comprehensive testing for scalability and fault tolerance
- Conduct performance analysis under various load conditions
- Prepare and finalize the final presentation.