Phase 1: System Design Report

Synchronous KV Store with LRU Cache and PostgreSQL Replication

Department of Computer Engineering
Project Phase 1 Submission

Author: Monil Manish Desai

November 7, 2025

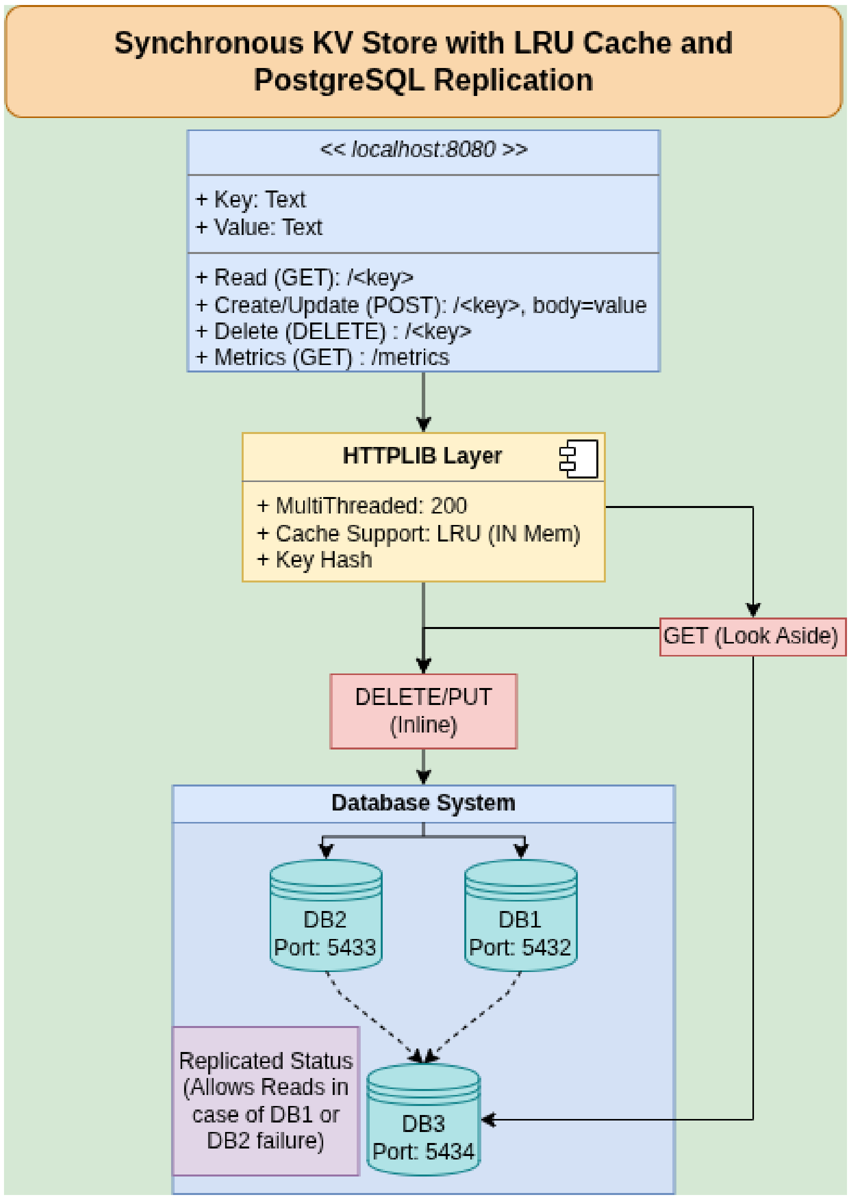
CS744 DECS - Project: HTTP-based Key-Value Server

This project implements a **distributed HTTP-based Key-Value (KV) store** with **PostgreSQL logical replication** for reliability and fault tolerance. The system uses multiple PostgreSQL instances managed through a lightweight, multi-threaded HTTP server.

GitHub Repo: `https://github.com/monil2003/cs744-project.git`

Overview

The following figure illustrates the architecture of the system.



Synchronous KV Store with LRU Cache and PostgreSQL Replication
<< localhost:8080 >>
+ Key: Text
+ Value: Text
+ Read (GET): /<key>
+ Create/Update (POST): /<key>, body=value
+ Delete (DELETE) : /<key>
+ Metrics (GET) : /metrics
HTTPLIB Layer
+ MultiThreaded: 200
+ Cache Support: LRU (IN Mem)
+ Key Hash
GET (Look Aside)
DELETE/PUT (Inline)
Database System
DB2 Port: 5433
DB1 Port: 5432
Replicated Status (Allows Reads in case of DB1 or DB2 failure)
DB3 Port: 5434

Figure 1: System Architecture of Synchronous KV Store with LRU Cache and Multi-Database Backend

HTTP Server Implementation (httplib, localhost:8080)

The server is implemented using the **cpp-httplib** library. It listens on port **8080** and supports the following RESTful endpoints:

- `GET /<key>` – Retrieve a value for a given key.
- `POST /<key>` – Insert or update a key-value pair.
- `DELETE /<key>` – Remove a key-value pair.
- `GET /metrics` – Return cache and database performance metrics.

Each HTTP request follows one of two main execution paths:

1. **In-memory access:** When the requested key is found in the LRU cache, providing faster responses.
2. **Disk access:** When the key is not in the cache and must be retrieved or updated in the PostgreSQL database.

PostgreSQL Databases

- **db1 (port 5432)** and **db2 (port 5433)** act as primary databases.
- **db3 (port 5434)** acts as a replica subscribing to both primaries for high availability.

Replication

Logical replication ensures data consistency across nodes, with **db3** automatically synchronizing updates from both primaries.

Modes of Operation

- **Replicated Mode:** db1, db2, and db3 (high availability)
- **Direct Mode:** db1 and db2 only (for performance testing)

Features

- Multi-threaded HTTP request handling
- LRU caching for frequent lookups

  – Manual PostgreSQL replication setup

  – Fault-tolerant data access via db3 fallback

  – Docker-based isolated deployment

Running the System

1. Start all PostgreSQL containers (db1, db2, db3).
2. Navigate to the `Databases` folder and start Docker:

   ```
   cd Databases
   docker compose up -d
   ```

3. Manually configure replication as per the documentation.
4. Navigate to the server folder:

   ```
   cd ../server
   ```

5. Launch the HTTP server:

   ```
   ./server_app
   ```

6. (Optional) Recompile using CMake:

   ```
   cd build
   cmake --build .
   ```

7. Refer to the `.txt` files in each folder for example commands and code snippets.