Multi-Tier HTTP Key-Value Store

C++ with Dual LRU Caching & MySQL

Baldau Dhanoriya, (24M0839)
IIT Bombay CS 744 Project

Overview

This project implements a high-performance, multi-threaded HTTP-based key-value store in C++ with **dual three-tier architecture(cpu bound and io bound)**: HTTP server (cpp-httplib with 8 worker threads) → Dual LRU caches (KV + Hash) → MySQL dual tables (kv_pairs + hash_store).

Key Features:

- **Dual Cache Design:** Separate KV cache (1000) and Hash cache (500) prevent interference
- **Smart Hash Caching:**speedup for repeated hash computations via cache + DB persistence
- **Thread-Safe:** Connection pooling (10), mutex-protected operations
- **Collision-Safe:** Composite unique keys for hash storage

1 Quick Start

Install Dependencies:

```
sudo apt install -y build-essential cmake mysql-server libmysqlclient-dev
```

Setup Database:

```
./setup_mysql.sh  # Creates both kv_pairs and hash_store tables
```

Build and Run:

```
cmake -S . -B build && cmake --build build
./build/kv-server
```

2 Architecture

Three Tiers (Dual Design):

1. **HTTP Layer:** cpp-httplib with 8 worker threads
2. **Cache Layer:** KV Cache (1000) + Hash Cache (500), independent LRU
3. **Database Layer:** MySQL with kv_pairs + hash_store tables

Technology Stack:

- C++, cpp-httplib, MySQL 8.0+, CMake
- Custom LRU cache, Connection pooling (10 connections)

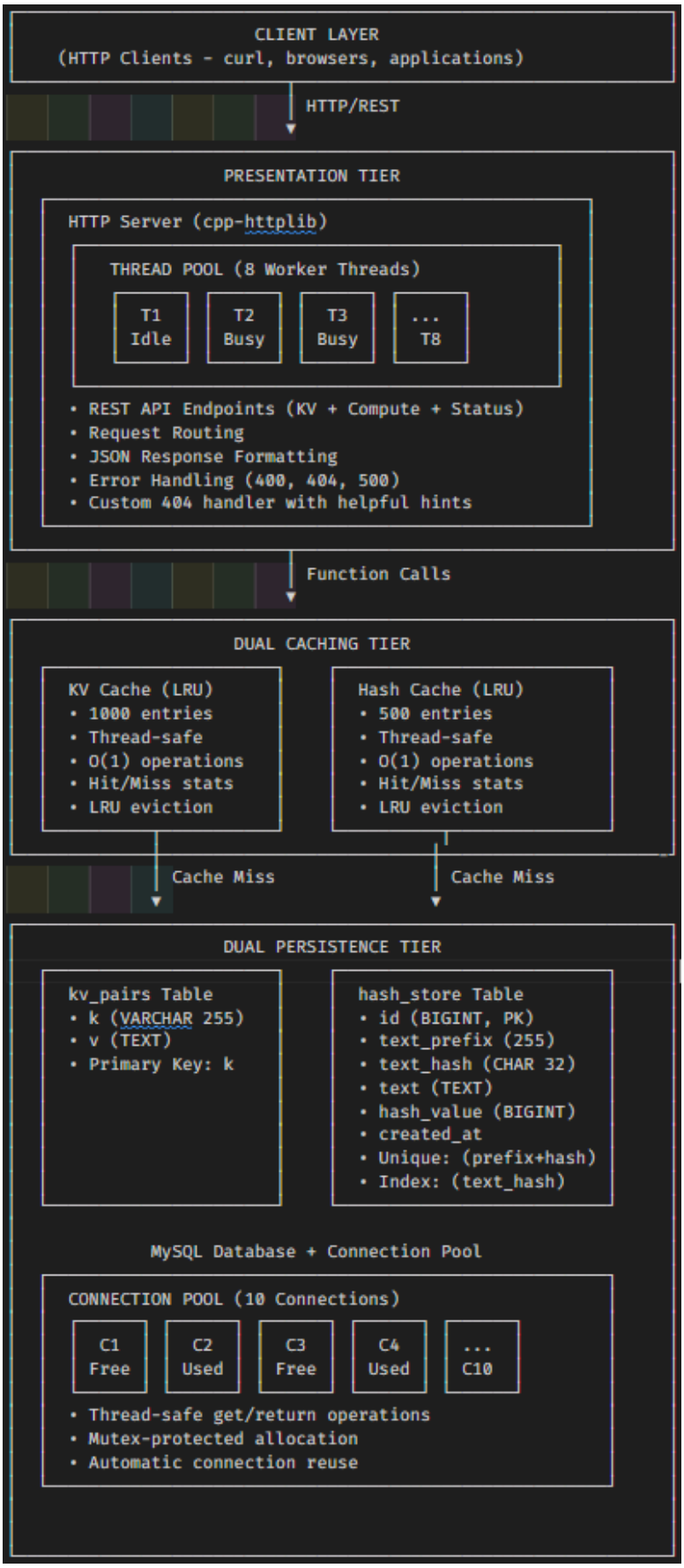

Figure 1: Dual Three-Tier Multi-Threaded System Architecture

3 API Reference

3.1 Key-Value Operations

POST /kv/create - Store key-value pair

```
curl -X POST "http://localhost:8080/kv/create?key=name&value=baldau" -d
    ""
# Response:
# {"success":true, "message":"Key created", "key":"name",
#  "value":"baldau", "overwritten":false}

# Overwrite existing key:
curl -X POST "http://localhost:8080/kv/create?key=name&value=varun" -d
    ""
# {"success":true, "message":"Key overwritten", "key":"name",
#  "value":"varun", "overwritten":true, "old_value":"baldau"}
```

GET /kv/read - Retrieve value

```
curl "http://localhost:8080/kv/read?key=name"
# First read (cache miss):
# {"success":true, "key":"name", "value":"baldau", "source":"database"}

# Second read (cache hit):
# {"success":true, "key":"name", "value":"baldau", "source":"cache"}
```

DELETE /kv/delete - Remove key

```
curl -X DELETE "http://localhost:8080/kv/delete?key=name"
# Response: {"success":true, "message":"Key deleted"}
```

3.2 Compute Operations

GET /compute/prime - Generate prime numbers

```
curl "http://localhost:8080/compute/prime?count=20"
# Response: {"success":true, "count":20,
#  "primes":[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71]}
```

GET /compute/hash - Compute hash (with caching)

```
curl "http://localhost:8080/compute/hash?text=baldau dhanoriya"
# First request (computed):
# {"success":true, "text":"baldau dhanoriya",
#  "hash":37485789347, "source":"computed"}

# Second request (cache hit : faster):
# {"success":true, "text":"baldau dhanoriya",
#  "hash":37485789347, "source":"cache"}

# Third request after cache eviction (database hit):
# {"success":true, "text":"baldau dhanoriya",
#  "hash":37485789347, "source":"database"}
```

3.3 Monitoring

GET /status - Dual cache statistics

```
curl "http://localhost:8080/status"
```

```
{
  "success": true,
  "data": {
    "server": "running",
    "kv_cache_size": 342,
    "kv_cache_hits": 875,
    "kv_cache_misses": 125,
    "kv_cache_hit_rate": 87.50,
    "hash_cache_size": 128,
    "hash_cache_hits": 456,
    "hash_cache_misses": 24,
    "hash_cache_hit_rate": 95.00,
    "database": "connected"
  }
}
```

4 Configuration

Configure via `include/config.h`:

```
namespace Config {
    const string HOST = "0.0.0.0";
    const int PORT = 8080;
    const int THREADS = 8;

    const string DB_HOST = "localhost";
    const string DB_USER = "root";
    const string DB_PASS = "";  // Set your password
    const string DB_NAME = "kvstore_db";
    const int DB_POOL = 10;

    const int CACHE_SIZE = 1000;         // KV cache
    const int HASH_CACHE_SIZE = 500;     // Hash cache
}
```

Rebuild after changes: `cmake -build build`

IIT Bombay CS 744 Project

`https://github.com/baldaudhanoriya/multi_threaded-HTTP-server.git`