HTTP-based KV server with in-mem Cache

This project implements a simple HTTP key-value server with a custom in-memory cache, a PostgreSQL database, and a multithreaded closed-loop load generator for performance experiments. The server supports create/read/delete operations and evaluates throughput and latency under configurable load level and workload.

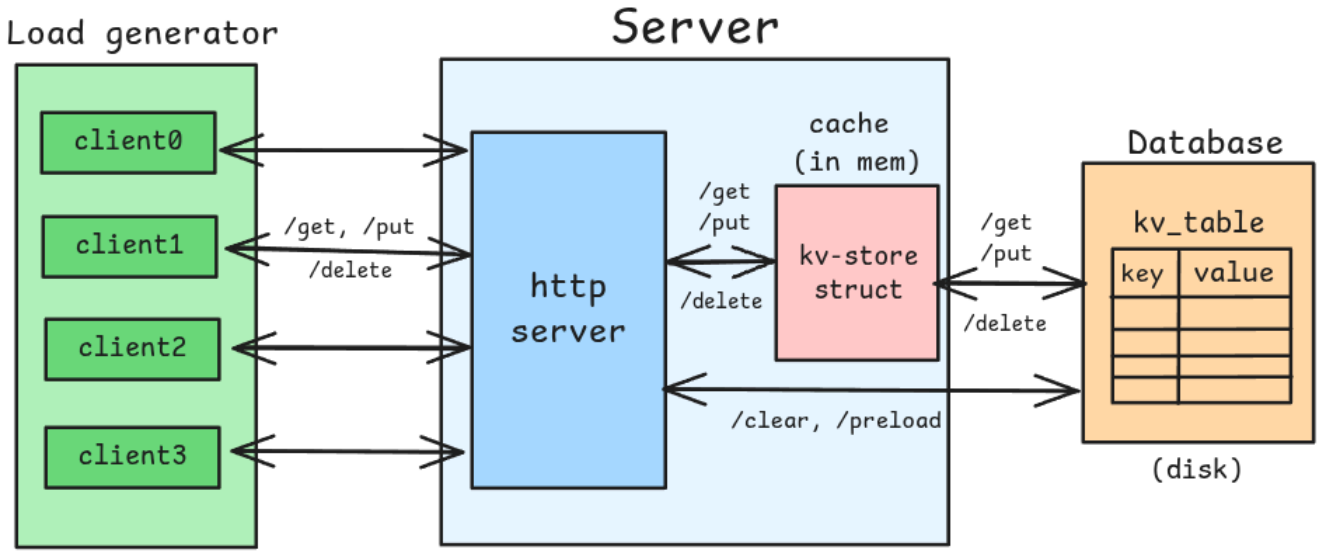
Github Repo: https://github.com/s1ddh3sh/CS744-project

System Overview :

The system consists of four main components:

- Server: HTTP server (httplib) with /create, /get, /delete, /preload, /clear, /cache-stats endpoints. Coordinates cache and DB access.
- In-memory Cache:Thread-safe, bounded in-memory key-value store (kv-store.c / kv.store.h) with FCFS eviction policy.
- Database Layer : DB connectivity layer (db.c / db.h) using libpq and a PostgreSQL database.
- Load Generator: multi-threaded client program (loadgen.cpp) that generates deterministic workloads and measures throughput and response time.

Architecture :



Implementation Notes:

- HTTP Server is a multi-threaded server with fixed 32 threads.
- Load generator can be executed with configurable load level (#threads) and workload types (get_all, get_popular, put-all, mixed).
- For GET workloads, DB is cleared and pre-populated with 10k keys to avoid DB-misses (using /preload and /clear).
- Custom cache (kv-store) is used rather than map data structure.
- Cache and database functions are properly guarded with pthread mutex for thread-safety.
- Please read README.md for more details regarding the execution with taskset.

Workloads :

- put_all: 50% PUT + 50% DELETE requests.
- get_all : repeated GET requests for 10k keys.
- get_popular : repeated GET requests for popular 1k keys.
- mixed : 70% GET + 20% PUT + 10% DELETE requests.

Metrics used for performance evaluation :

- Average throughput: successful requests per second (measured at loadgen).
- Average response time: avg response latency measured at the client
- Server Cache hit rate : for GET specific workloads (measured at the server) and accessed by the loadgen via /cache-stats.
- CPU core utilization : For the server-affined core using “mpstat” (future work)
- Disk utilization : For the postgres DB accesses using “iostat” (future work)