Redis Clone

Nikhil Verma

1 Introduction

This report outlines the development of a Redis clone, a high-performance, in-memory key-value store. The clone was developed using TCP for communication, with protocol parsing and non-blocking I/O implemented using polling and timers to handle concurrency. The system is optimized for performance, leveraging multi-threading and efficient data structures.

2 Key Features

2.1 TCP Server and Protocol Parsing

- Developed a TCP server to handle client connections and communication.
- Implemented protocol parsing to interpret and process Redis commands sent by clients.

2.2 Concurrency Management

- Utilized non-blocking I/O with polling mechanisms to manage multiple client connections efficiently.
- Employed timers to handle timeouts and other asynchronous events.
- Optimized the system by incorporating multi-threading, particularly for efficient destructor offloading using semaphores.

2.3 Data Structures and Command Implementation

- Implemented key Redis commands, using efficient data structures such as hashtables, AVL trees, linked lists, and heaps.
- Ensured that the data structures are optimized for speed and memory usage, aligning with Redis's high-performance requirements.

3 Performance Optimization

- Multi-threading was employed to offload destructors, reducing the load on the main thread and improving overall system performance.
- Semaphores were used to manage thread synchronization, ensuring safe and efficient multi-threaded execution.

4 Technology Stack

- **Programming Language**: C/C++ for system-level programming.
- **Concurrency Mechanisms**: Polling, non-blocking I/O, timers, multi-threading, and semaphores.
- **Data Structures**: Hashtables, AVL trees, linked lists, and heaps.

5 Conclusion

The developed Redis clone effectively replicates the core functionalities of Redis, achieving high performance through careful implementation of concurrency mechanisms and data structures. The project demonstrates the potential of custom-built in-memory key-value stores in scenarios demanding low latency and high throughput.