Storage Engine with DPDK Integration

Chinmay Bhusari - 21CS30015
Ronit Nanwani - 21CS30043
Apoorv Kumar - 21CS10008
Sanchit Yewale - 21CS10056

1 Overview

This document presents the design of a high-performance key-value store server that implements the RESP-2 protocol, used by Redis. The server is capable of handling multiple TCP client connections concurrently using asynchronous I/O, with an optimized version utilizing DPDK (Data Plane Development Kit) to enhance network performance. This design aims to support the primary Redis commands — `SET`, `GET`, and `DEL` — while ensuring efficient packet processing for high-throughput workloads.

2 System Design

The overall system consists of two core components:

- **Server**: Responsible for accepting incoming TCP connections, processing commands asynchronously, and managing client requests efficiently.
- **Client**: Interacts with the server using the RESP-2 protocol to issue commands and receive responses.

2.1 Server Architecture

The server design is split into two versions: a basic version that uses traditional socket I/O, and a more advanced version that integrates DPDK for enhanced packet processing.

2.1.1 Managing Client Connections

The server employs asynchronous I/O techniques to manage multiple client connections. Using `select()` or a similar mechanism, the server monitors several connections without blocking any individual connection.

- The server listens for incoming connections on TCP port `6379`.
- When a new connection is detected, it is added to the monitored file descriptors.
- Each connection is processed asynchronously, ensuring non-blocking operations while reading commands, performing necessary actions, and responding to clients.

In the DPDK-enhanced version, the server bypasses the traditional kernel network stack by directly accessing the NIC (Network Interface Controller) in user space, reducing packet processing latency and enhancing performance.

2.1.2 Command Processing

The server processes commands based on the RESP-2 protocol. The main supported commands include:

- `SET key "value"`: Stores a key-value pair in the in-memory store.
- `GET key`: Retrieves the value associated with the key.
- `DEL key`: Removes a key from the store.

Each of these commands is handled by a specific function, such as `handle_set()`, `handle_get()`, and `handle_del()`, and the data is managed using an in-memory data structure like a hash table or dictionary.

2.1.3 Handling Errors

In accordance with the RESP-2 protocol, the server returns an error message for invalid or unsupported commands in the format:

```
-ERR <error message>
```

2.1.4 DPDK Performance Optimization

The DPDK-enabled server version offloads network packet processing directly to user space, bypassing the operating system's kernel stack. This reduces network processing overhead and improves packet I/O throughput. Key features include:

- Direct access to the NIC via DPDK's `rte_eth_dev` API.
- Increased packet processing efficiency due to the elimination of kernel-level networking overhead.
- Better utilization of CPU resources thanks to DPDK's optimized data plane.

2.1.5 Non-blocking I/O and Multithreading

To ensure high performance and scalability, the server uses an event-driven, non-blocking I/O model. This allows the server to process multiple client connections simultaneously without blocking any one connection.

2.2 Client Interaction

The client interacts with the server through the RESP-2 protocol, designed for efficient communication. The key client features include:

- **Command Input**: Users can input commands such as `SET`, `GET`, and `DEL`, adhering to the RESP-2 format.
- **Non-blocking I/O**: The client uses non-blocking sockets to interact with the server asynchronously, providing fast responses and supporting multiple concurrent requests.
- **Response Parsing**: The client parses and handles responses from the server, whether retrieving data for a `GET` request or displaying error messages for invalid commands.

The client is designed for interactive use in a REPL (Read-Eval-Print Loop) environment, allowing users to enter commands and immediately see results.

3 Conclusion

This document presents a detailed design for a RESP-2 compliant key-value store server supporting basic Redis operations (`SET`, `GET`, `DEL`). The server is optimized to handle multiple concurrent connections using non-blocking I/O, and an advanced version integrates DPDK to significantly improve network packet processing performance. By bypassing the kernel's network stack, DPDK enhances throughput and reduces packet processing latency. Through benchmarks, the performance benefits of the DPDK-enabled version will be demonstrated.