Virtual Disk System with Caching and TCP

JaeMin Birdsall

August 1 2024

1 Overview

This project implements a simulated storage system using a cache for block-level data retrieval. The system interacts with a virtual disk array (JBOD - Just a Bunch Of Disks) to perform read and write operations efficiently. The project includes several modules that handle storage management, caching, and networking, making it suitable for testing storage algorithms in low-level memory management environments.

2 Modules

2.1 MDADM Storage Controller

The MDADM controller, implemented in `mdadm.c`, coordinates higher-level operations for interacting with the JBOD system. It manages mounting, unmounting, and both read and write operations. The key functions are:

- `mdadm_mount()`: Mounts the JBOD system and prepares it for read/write operations.
- `mdadm_unmount()`: Unmounts the JBOD system, preventing further access to the disks.
- `mdadm_read()`: Reads data from the JBOD starting at a specific address for a given length.
- `mdadm_write()`: Writes data to the JBOD at a specified address for a given length, ensuring write permissions are granted.

2.2 Cache Management

The cache system, implemented in `cache.c`, provides temporary storage for frequently accessed blocks. It uses the Least Frequently Used (LFU) algorithm for cache management. The main cache functions are:

- `cache_create()`: Initializes the cache with a specified number of entries.
- `cache_lookup()`: Retrieves a block from the cache if it exists.
- `cache_insert()`: Inserts a block into the cache, evicting the least frequently accessed entry if necessary.
- `cache_update()`: Updates an existing cache entry with new data.
- `cache_destroy()`: Frees the memory allocated to the cache when it is no longer needed.

2.3 Networking: Sending and Receiving Packets over IP/TCP

The `net.c` module is responsible for communication between the client and server using IP/TCP sockets. It allows remote execution of JBOD operations by sending and receiving packets over the network. The primary functions in this module are:

- `jbod_connect()`: Establishes a connection to the JBOD server using a specified IP address and port via TCP.

- `send_packet()`: Constructs and sends a request packet over the socket, including the JBOD opcode and optional block data for write operations.
- `recv_packet()`: Receives a response packet from the server, parsing the opcode, status, and optional block data from the server.
- `jbod_client_operation()`: Sends a JBOD operation request to the server and processes the response, handling the communication between client and server.
- `jbod_disconnect()`: Closes the connection to the server and frees any associated resources.

3 Usage

To run the system, first compile all the source files using a standard C compiler. Use the provided `tester.c` program to simulate workloads and test the system's performance. The cache can be enabled using the `-s` flag with a specified size, and workloads can be defined in external files.

```
$ ./tester -w workload.txt -s 64
```

The server must be running to handle JBOD operations via networking.

4 Conclusion

This project provides a comprehensive solution for managing a block-level storage system with multiple components working together to ensure efficient operation. The MDADM controller allows for seamless mounting, unmounting, reading, and writing to a simulated disk array. The caching layer optimizes performance by reducing the number of disk accesses, using a Least Frequently Used (LFU) policy to store frequently accessed blocks in memory. The networking module enables remote interaction with the storage system, facilitating the sending and receiving of packets over IP/TCP, allowing for distributed storage operations in a client-server architecture.

Together, these components create a versatile and scalable system for experimenting with storage management techniques, caching algorithms, and network-based communication in storage systems. This project serves as a foundational summary of low level memory caching and networking.