CMPSC 311 Technical Report

The JBOD library is a software library that provides an interface for clients to communicate with a JBOD (Just a Bunch Of Disks) server. The library includes functions for connecting to the server, sending requests for JBOD operations, and receiving responses from the server.

The library uses a packet-based protocol to communicate with the server. A packet is a structured message that contains information about the JBOD operation to be performed or the response to a request. The library defines a packet structure that includes fields for the length of the packet, the operation code, a return code indicating the success or failure of the operation, and a block of data.

The library includes functions for creating and sending packets, as well as functions for receiving and processing responses from the server. The library also includes functions for connecting to and disconnecting from the server.

Overall, the JBOD library provides a convenient and efficient way for clients to interact with a JBOD server, allowing them to perform a variety of storage operations on a collection of disks as if they were a single disk.

Mdadm

The JBOD system consists of a set of disks with a fixed size of 256 blocks, where each block is 4 bytes. The system has four operations: MOUNT, UNMOUNT, READ_BLOCK, and WRITE_BLOCK. The MOUNT operation is used to initialize the disks, while the UNMOUNT operation is used to safely shut down the disks. The READ_BLOCK operation reads a block of data from a specific disk and block number, while the WRITE_BLOCK operation writes a block of data to a specific disk and block number.

The **mdadm_mount** function mounts the JBOD system by sending a MOUNT operation to the disks. If the system is already mounted, the function returns an error.

The **mdadm_unmount** function unmounts the JBOD system by sending an UNMOUNT operation to the disks. If the system is not mounted, the function returns an error.

The **mdadm_read** function reads data from the JBOD system by using the READ_BLOCK operation. The function calculates the disk number, block number, and block offset values based on the given address. It then sends SEEK_TO_DISK and SEEK_TO_BLOCK operations to the disks to position the read buffer to the correct block. The function reads the block of data into a temporary buffer using the READ_BLOCK operation, copies the data to the output buffer, and repeats the process until all requested data has been read.

The **mdadm_write** function writes data to the JBOD system by using the WRITE_BLOCK operation. The function performs similar calculations and operations to the mdadm_read function to position the write head to the correct block. The function writes the data from the input buffer to the block of data using the WRITE_BLOCK operation, and repeats the process until all data has been written.

Cache

The cache is implemented as a clock algorithm. When a cache entry is accessed, the clock is incremented, and the access time is stored in the access_time field of the corresponding cache_entry_t structure. When a new block needs to be cached and there are no empty cache entries, the clock algorithm is used to find the least recently used (LRU) entry, which is evicted to make room for the new block.

c**ache_create(int num_entries):** Creates a cache with a specified number of entries, and initializes the cache with zeroed-out entries. Returns 1 on success, -1 on failure.

cache_destroy(): Frees the memory used by the cache and resets all metadata. Returns 1 on success, -1 on failure.

cache_lookup(int disk_num, int block_num, uint8_t *buf): Looks up a disk block in the cache. If the block is found, its contents are copied into the provided buffer, and the access time of the cache entry is updated. Returns 1 on success, -1 on failure.

cache_insert: Inserts a new disk block into the cache. If the block is already in the cache, returns -1. If the cache is full, the LRU block is evicted to make room. Returns 1 on success, -1 on failure.

Net

This is an implementation of a client-server system for executing JBOD operations. The client sends a request packet to the server, the server processes the request, and then sends a response packet back to the client. The following is a description of each function in the code:

nread: This function attempts to read len bytes from the file descriptor fd into the buffer buf. It returns true if it successfully reads all len bytes, and false if it encounters an error.

nwrite: This function attempts to write len bytes from the buffer buf to the file descriptor fd. It returns true if it successfully writes all len bytes, and false if it encounters an error.

recv_packet: This function attempts to receive a packet from the file descriptor fd. It reads the packet length from the packet buffer, extracts the operation code and return code from the buffer, and stores them in the variables pointed to by op and ret, respectively. If block is not NULL, it also extracts the data block from the packet buffer and stores it in block. It returns true if it successfully receives the packet, and false if it encounters an error.

send_packet: This function creates a packet with the given operation code and data block using the packetcreation() function. It then sends the packet to the file descriptor sd using the nwrite() function, and returns true if it successfully sends the packet, and false if it encounters an error.

jbod_connect: This function attempts to connect to a server at the IP address ip and port number port. It converts the IP address to binary form using inet_pton(), creates a socket using socket(), and connects to the server using connect(). If it successfully connects, it sets the global variable cli_sd to the socket descriptor and returns true. If it encounters an error, it closes the socket and sets cli_sd to -1, and returns false.

void jbod_disconnect: This function closes the socket.

int jbod_client_operation: This function sends a reads packets and writes responses.