Technical Report

I have created a created a library that allows multiple disks to be treated as a single linear device. Furthermore, I have created wrappers around the already existing JBOD operations to allow users to read and write variable amounts of data to and from specific addresses. This solves the issue of only using JBOD operation, which only permits read and writes of 256 starting from the beginning of every block (which are each 256 bytes). Essentially, JBOD operations can only read and write entire blocks. My library allows read and writes from any address from 0 to 1MB up and any size up to 1024 bytes.

This library would be useful for a file system in an operating system. Once the disks are mounted, the operation would view them as a single storage device. The data each block holds, which can represent a file, can be stored/modifed (mdadm_write) and accessed (mdadm_read). Files could be moved by reading data contents from the linear device, clearing it, and then writing it to a different address in the linear device.

The library also supports client side operations over a network. This can allow you to scale the JBOD system over a network and switch to one in case one fails to avoid down time. However, the library also supports local operations. Simply replace jbod_client_operation in mdadm.c with jbod_operation. They have the same parameters so you don't need to worry about those.

This library also includes fully associative least recently used caching to improve performance by reducing latency and operation costs.

To proficiently use the library, there are 8 calls you should know how to use. They are mdadm_mount, mdadm_unmount, mdadm_read, mdadm_write, cache_create, cache_destroy, jbod_connect, and jbod_disconnect().

Note that the library uses the libssl-dev API. Make sure this is installed to use the library.

int mdadm_mount(void);

mdadm_mount mounts the system and returns 1 on success and -1 on failure. Make sure to call this function and make sure it succeeded before attempting to read and write data.

int mdadm_unmount();

mdadm_unmount unmounts the system and returns 1 on success and -1 on failure.

int mdadm_read(uint32_t addr, uint32_t len, uint8_t *buf);

mdadm_read reads len bytes from the linear device starting at address addr, and populates buf with this data. There are a few restrictions to be aware of.

1. The system must be mounted

2. addr cannot exceed 1MB.

3. A read request in which len exceeds the amount of available bytes starting at addr will be rejected. In other words, the maximum amount possible of bytes will not be read.

4. At most 1024 bytes can be read. In other words, a len exceeding 1024 will not be read.

The amount of bytes read (len) will be returned on success and -1 will be returned on failure.

int mdadm_write(uint32_t addr, uint32_t len, const uint8_t *buf);

mdadm_read writes len bytes to the linear device from the buf starting at address addr. There are a few restrictions to be aware of.

1. The system must be mounted

2. addr cannot exceed 1MB.

3. A write request in which len exceeds the amount of available bytes starting at addr will be rejected. In other words, the maximum amount possible of bytes will not be read.

4. At most 1024 bytes can be read. In other words, a len exceeding 1024 will not be read.

The amount of bytes read (len) will be returned on success and -1 will be returned on failure.

int cache_create(void)

This function attempts to enable the cache.

This function takes in no parameters and returns 1 on success, -1 on failure.

int cache_destroy(void)

This function attempts to disable the cache. It takes no parameters and returns 1 on success, -1 on failure

bool jbod_connect(const char *ip, uint16_t port)

Attempts to establish connection to server. Parameters are a an IPv4 address passed as a string and a port integer

Returns true on success false on failure

void jbod-disconnect(void)

Closes connection to server and resets file descriptor of socket.