作業系統總整與實作 Final Project

0816034 蔡家倫

0. code

Github Link

1. FUSE 目的及原理

FUSE 是 filesystem in userspace 的簡稱，可以讓 userspace programs 建立虛擬檔案系統，不用重新 compile kernel。本次 lab 就是使用 FUSE 模擬 ssd 。FUSE 原理為用 libfuse library 接管 read request from kernel，處理完後再 send responses back。在 ssd_fuse.c 中的 fuse_operations 就是我們要定義如何接管這些 read/write 方法。

2. 修改的地方

我修改的部分有 TODO 的四個部分 (ssd_do_write, ftl_write, ssd_do_read, ftl_read) 以及 garbage collection

3. 實作的方法

詳細方法打在 code 註解裡

ssd_do_write

```
static int ssd_do_write(const char* buf, size_t size, off_t offset)
{
    int tmp_lba, tmp_lba_range, process_size;
    int idx, curr_size, remain_size, rst;
    char* tmp_buf;

    host_write_size += size;
    if (ssd_expand(offset + size) != 0)
    {
        return -ENOMEM;
    }

    tmp_lba = offset / 512;
    tmp_lba_range = (offset + size - 1) / 512 - (tmp_lba) + 1;

    process_size = 0; // 這次 for 要處理的 size
    remain_size = size; // 剩下的 size
    curr_size = 0; // 目前已處理的 size

    tmp_buf = (char *)malloc(512);
    for (idx = 0; idx < tmp_lba_range; idx++)
    {
        // TODO
        // 如果沒有 free block 以及要移的 block 跟剩下的 page 相等時做 garbage
```

```
collection

        if (free_block_number == 0) {
            int del = -1;
            unsigned int pages = 11;
            for (int i = 0; i < 13; i++) {
                if (i == curr_pca.fields.nand) continue;
                if (valid_count[i] < pages) {
                    del = i;
                    pages = valid_count[i];
                }
            }
            if (del == -1 || pages == FREE_BLOCK) {
                break;
            }
            if (9-curr_pca.fields.lba == pages) {
                printf("-----gc\n");
                gc();
            }

        }


        // read
        ftl_read(tmp_buf, tmp_lba+idx);

        // modify
        if (idx == 0) {
            process_size = 512 - (offset % 512);
            if (size < process_size) process_size = size; // size < 512
            memcpy((char *)tmp_buf+(offset % 512), (char *)buf+curr_size,
process_size);
        }
        else if (idx == tmp_lba_range-1) {
            process_size = remain_size;
            memcpy((char *)tmp_buf, (char *)buf+curr_size, process_size);
        }
        else {
            process_size = 512;
            memcpy((char *)tmp_buf, (char *)buf+curr_size, process_size);
        }
        curr_size += process_size;
        remain_size -= process_size;
        //printf("[process] %d [curr] %d [remain] %d\n", process_size, curr_size,
remain_size);

        // write
        ftl_write(tmp_buf, 0, tmp_lba+idx);
    }
    free(tmp_buf);
    return size;
}
```

ftl_write

```
static int ftl_write(const char* buf, size_t lba_range, size_t lba)
{
    // TODO
    // 如果不是 INVALID_PCA 表示此 logic page 已被寫過，
    // 舊的 physical 要清除 P2L 紀錄
    if (L2P[lba] != INVALID_PCA) {
        valid_count[L2P[lba]>>16]--;
        P2L[(L2P[lba]>>16)*10+(L2P[lba]&0xffff)] = INVALID_LBA;
    }
    // 更新 L2P, P2L 表
    PCA_RULE temp;
    temp.pca = get_next_pca();
    L2P[lba] = temp.pca;
    P2L[temp.fields.nand*10 + temp.fields.lba] = lba;
    // 呼叫 nand_write
    nand_write(buf, L2P[lba]);
}
```

ssd_do_read

```
static int ssd_do_read(char* buf, size_t size, off_t offset)
{
    int tmp_lba, tmp_lba_range, rst ;
    char* tmp_buf;

    //off limit
    if ((offset ) >= logic_size)
    {
        return 0;
    }
    if ( size > logic_size - offset)
    {
        //is valid data section
        size = logic_size - offset;
    }

    tmp_lba = offset / 512;
    tmp_lba_range = (offset + size - 1) / 512 - (tmp_lba) + 1;
    tmp_buf = (char *)calloc(tmp_lba_range * 512, sizeof(char));

    for (int i = 0; i < tmp_lba_range; i++) {
        // TODO
        // 呼叫 ftl_read 依序寫進 tmp_buf
        ftl_read(tmp_buf+(512*i), tmp_lba+i);
    }

    memcpy(buf, tmp_buf + (offset % 512), size);
```

```
    free(tmp_buf);
    return size;
}
```

ftl_read

```
static int ftl_read( char* buf, size_t lba)
{
    // TODO
    // 呼叫 nand_read
    nand_read(buf, L2P[lba]);
}
```

garbage collection

```
static void gc() {
    for (int i = 0; i < 13; i++) {
        printf("%x ", valid_count[i]);
    }
    printf("\n");

    // free 1 block
    while(free_block_number < 1) {

        // 找移的 page 最少的 nand
        int del = -1;
        unsigned int pages = 11;
        for (int i = 0; i < 13; i++) {
            if (i == curr_pca.fields.nand) continue;
            if (valid_count[i] < pages) {
                del = i;
                pages = valid_count[i];
            }
        }
        if (del == -1 || pages == FREE_BLOCK) {
            break;
        }
        printf("[GC] %d ,%d pages\n", del, 10-pages);
        PCA_RULE temp;
        char *buf = (char *)malloc(512);
        for (int i = 0; i < 10; i++) {

            // 不是 INVALID_LBA 的 page 要移動
            if (P2L[del*10+i] != INVALID_LBA) {
                ftl_read(buf, P2L[del*10+i]);
                ftl_write(buf, 0, P2L[del*10+i]);
                P2L[del*10+i] = INVALID_LBA;
            }
```

```
            }
            free(buf);

            // 都移動完了就 erase 該 nand
            nand_erase(del);
        }

        for (int i = 0; i < 13; i++) {
            printf("%x ", valid_count[i]);
        }
        printf("\n");
        fflush(stdout);
        return;
    }
```

4.結果分析

附上 test.sh 以及各 test 的成績:

- test1: 1.000000
- test2: 1.000000
- test3: 1.000000
- test4: 1.659645
- test5: 1.000032
- mytest: 1.334437
- test7: 1.150820

```
#!/bin/bash

SSD_FILE="/tmp/ssd/ssd_file"
GOLDEN="/tmp/ssd_file_golden"
TEMP="/tmp/temp"
touch ${GOLDEN}
truncate -s 0 ${SSD_FILE}
truncate -s 0 ${GOLDEN}

rand(){
    min=$1
    max=$(($2-$min))
    num=$(cat /dev/urandom | head -n 10 | cksum | awk -F ' ' '{print $1}')
    echo $(($num%$max))
}

case "$1" in
    "test1")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        ;;
    "test2")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
```

```
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 11264 > ${TEMP}
        for i in $(seq 0 9)
        do
            dd if=${TEMP} iflag=skip_bytes skip=$(($i*1024)) of=${GOLDEN}
oflag=seek_bytes seek=$(($i*5120)) bs=1024 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} iflag=skip_bytes skip=$(($i*1024)) of=${SSD_FILE}
oflag=seek_bytes seek=$(($i*5120)) bs=1024 count=1 conv=notrunc 2> /dev/null
        done
        dd if=${TEMP} iflag=skip_bytes skip=10240 of=${GOLDEN} oflag=seek_bytes
seek=0 bs=1024 count=1 conv=notrunc 2> /dev/null
        dd if=${TEMP} iflag=skip_bytes skip=10240 of=${SSD_FILE} oflag=seek_bytes
seek=0 bs=1024 count=1 conv=notrunc 2> /dev/null
        ;;
    "test3")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 11264 > ${TEMP}
        for i in $(seq 0 49)
        do
            dd if=${TEMP} iflag=skip_bytes skip=$(($i*512)) of=${GOLDEN}
oflag=seek_bytes seek=$(($i*1024)) bs=1024 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} iflag=skip_bytes skip=$(($i*512)) of=${SSD_FILE}
oflag=seek_bytes seek=$(($i*1024)) bs=1024 count=1 conv=notrunc 2> /dev/null
        done
        dd if=${TEMP} iflag=skip_bytes skip=10240 of=${GOLDEN} oflag=seek_bytes
seek=0 bs=1024 count=1 conv=notrunc 2> /dev/null
        dd if=${TEMP} iflag=skip_bytes skip=10240 of=${SSD_FILE} oflag=seek_bytes
seek=0 bs=1024 count=1 conv=notrunc 2> /dev/null
        ;;
    "test4")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 > ${TEMP}
        for i in $(seq 0 400)
        do
            skip_b=$(shuf -i 10240-20480 -n 1)
            seek_b=$(shuf -i 10240-20480 -n 1)
            #echo $skip_b $seek_b
            dd if=${TEMP} iflag=skip_bytes skip=${skip_b} of=${GOLDEN}
oflag=seek_bytes seek=${seek_b} bs=1024 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} iflag=skip_bytes skip=${skip_b} of=${SSD_FILE}
oflag=seek_bytes seek=${seek_b} bs=1024 count=1 conv=notrunc 2> /dev/null
            # if [ ! -z "$(diff ${GOLDEN} ${SSD_FILE})" ]; then
            #     echo -1
            #     exit 1
            # fi
        done
        ;;
    "test5")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 11264 > ${TEMP}
        dd if=${TEMP} iflag=skip_bytes skip=2 of=${GOLDEN} oflag=seek_bytes seek=0
```

```
bs=30720 count=1 conv=notrunc 2> /dev/null
        dd if=${TEMP} iflag=skip_bytes skip=2 of=${SSD_FILE} oflag=seek_bytes
seek=0 bs=30720 count=1 conv=notrunc 2> /dev/null
        ;;
    "mytest")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 > ${TEMP}
        skip_b=$(shuf -i 0-50176 -n 1)
        seek_b=$(shuf -i 0-50176 -n 1)
        for i in $(seq 0 100)
        do
            dd if=${TEMP} iflag=skip_bytes skip=${skip_b} of=${GOLDEN}
oflag=seek_bytes seek=${seek_b} bs=1024 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} iflag=skip_bytes skip=${skip_b} of=${SSD_FILE}
oflag=seek_bytes seek=${seek_b} bs=1024 count=1 conv=notrunc 2> /dev/null
        done
        ;;
    "test7")
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 51200 | tee
${SSD_FILE} > ${GOLDEN} 2> /dev/null
        cat /dev/urandom | tr -dc '[:alpha:][:digit:]' | head -c 11264 > ${TEMP}
        for i in $(seq 0 1000)
        do
            dd if=${TEMP} skip=1024 of=${GOLDEN} iflag=skip_bytes oflag=seek_bytes
seek=6789 bs=5000 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} skip=1024 of=${SSD_FILE} iflag=skip_bytes
oflag=seek_bytes seek=6789 bs=5000 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} skip=2024 of=${GOLDEN} iflag=skip_bytes oflag=seek_bytes
seek=123 bs=777 count=1 conv=notrunc 2> /dev/null
            dd if=${TEMP} skip=2024 of=${SSD_FILE} iflag=skip_bytes
oflag=seek_bytes seek=123 bs=777 count=1 conv=notrunc 2> /dev/null
        done
        ;;
    *)
        printf "Usage: sh test.sh test_pattern\n"
        printf "\n"
        printf "test_pattern\n"
        printf "test1: Sequential write whole SSD size(51200bytes)\n"
        printf "       test basic SSD read & write\n"
        printf "test2:\n"
        printf "       1: Sequential write whole SSD size(51200bytes)\n"
        printf "       2: Override 0, 1, 10, 11, 20, 21, 30, 31, 40, 41, 50, 51,
60, 61, 70, 71, 80, 81, 90, 91 page \n"
        printf "       2: Override 0, 1 page \n"
        printf "       test GC's result\n"
        return
        ;;
esac

# check
diff ${GOLDEN} ${SSD_FILE}
if [ $? -eq 0 ]
then
```

```
    echo "success!"
else
    echo "fail!"
fi

echo "WA:"
./ssd_fuse_dut /tmp/ssd/ssd_file W
rm -rf ${TEMP} ${GOLDEN}
```