Implementation details

Flash Memory Controller

(The fourth group)

Members :

312831005 Chih Chuan, Kao

312831009 Hsin, Lin

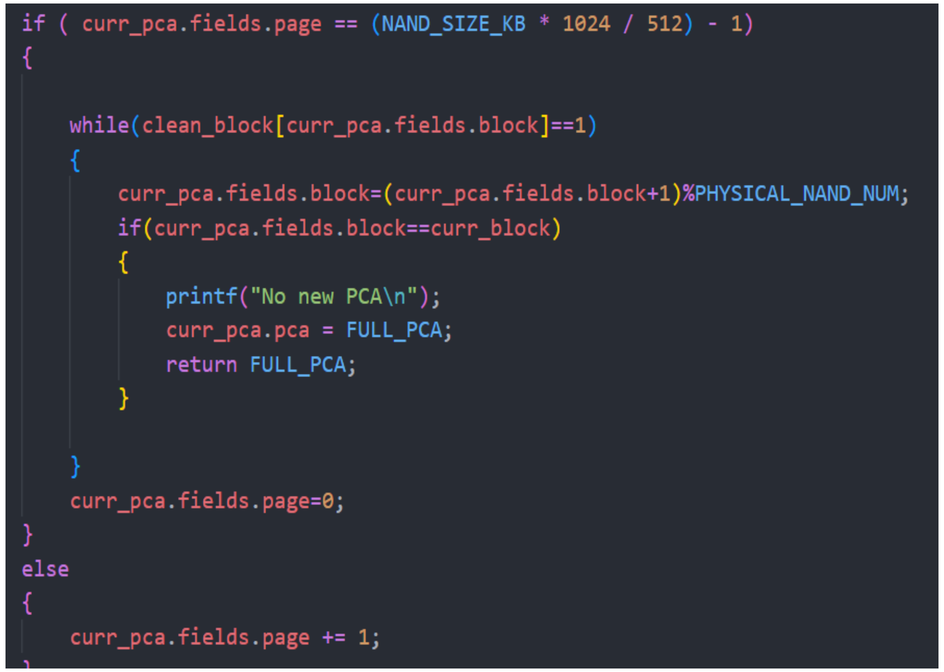
312832006 Yu-Wen, Teng

Outline

- Get_next_pca()
- ssd_do_write()
- Garbage Collection
- Optimization

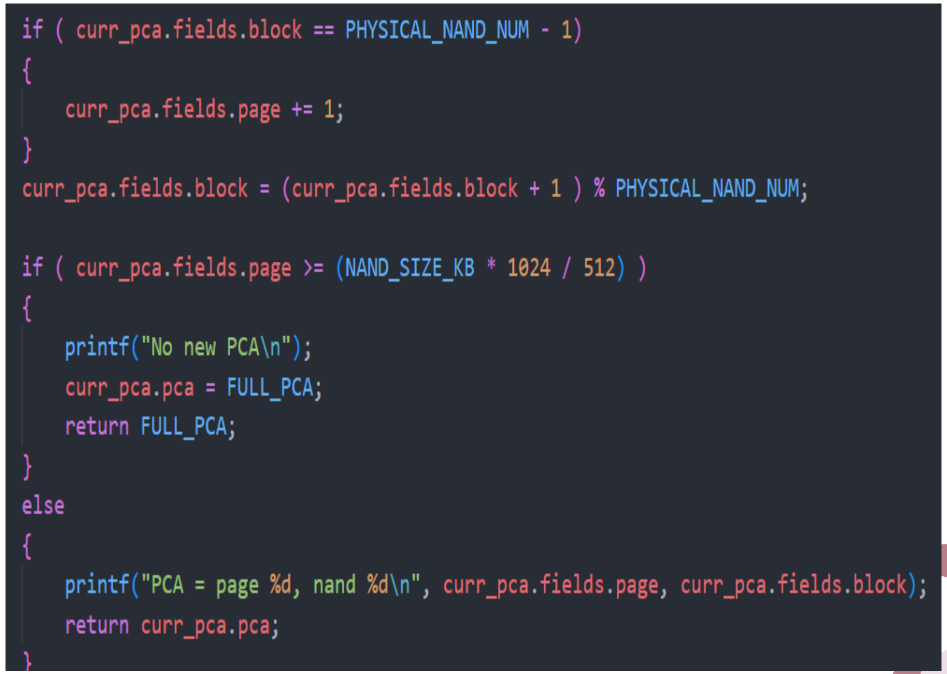
Get_next_pca()

sequence A to sequence B



```c
if ( curr_pca.fields.page == (NAND_SIZE_KB * 1024 / 512) - 1)
{

    while(clean_block[curr_pca.fields.block]==1)
    {
        curr_pca.fields.block=(curr_pca.fields.block+1)%PHYSICAL_NAND_NUM;
        if(curr_pca.fields.block==curr_block)
        {
            printf("No new PCA\n");
            curr_pca.pca = FULL_PCA;
            return FULL_PCA;
        }

    }
    curr_pca.fields.page=0;
}
else
{
    curr_pca.fields.page += 1;
```

sequence A

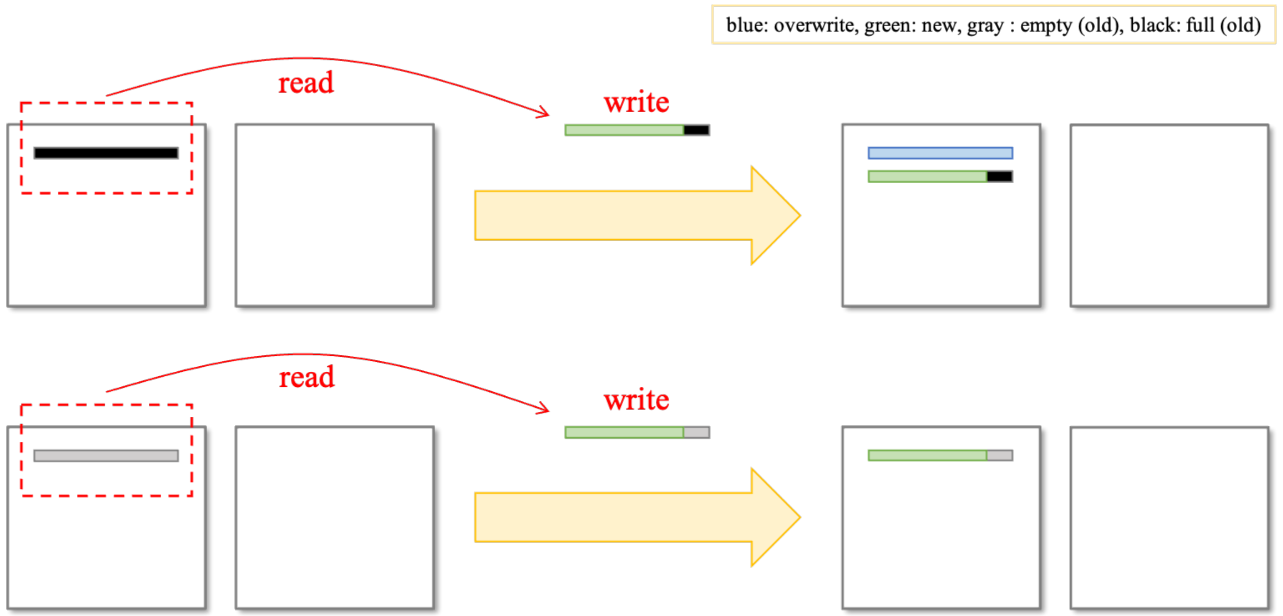


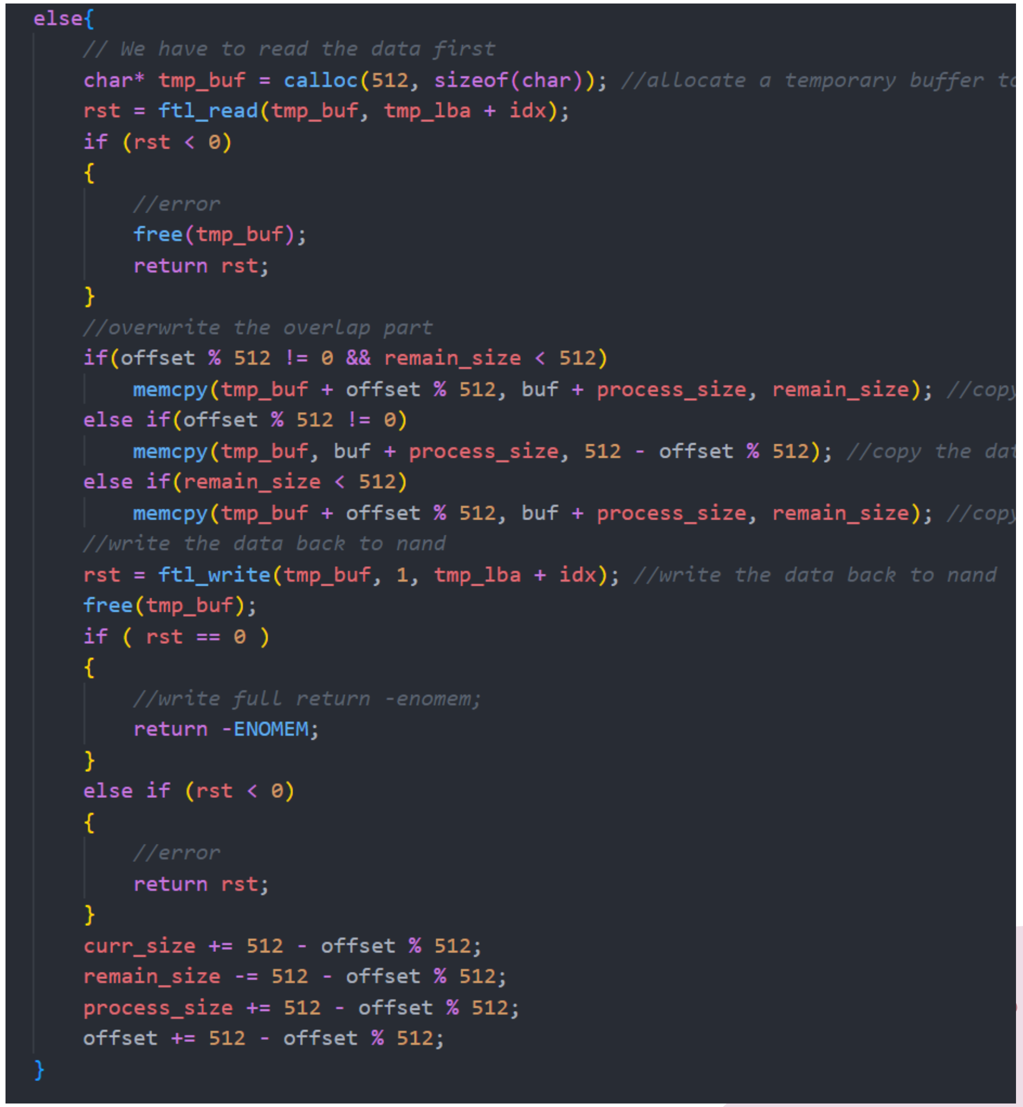
```c
if ( curr_pca.fields.block == PHYSICAL_NAND_NUM - 1)
{
    curr_pca.fields.page += 1;
}
curr_pca.fields.block = (curr_pca.fields.block + 1 ) % PHYSICAL_NAND_NUM;

if ( curr_pca.fields.page >= (NAND_SIZE_KB * 1024 / 512) )
{
    printf("No new PCA\n");
    curr_pca.pca = FULL_PCA;
    return FULL_PCA;
}
else
{
    printf("PCA = page %d, nand %d\n", curr_pca.fields.page, curr_pca.fields.block);
    return curr_pca.pca;
```

sequence B

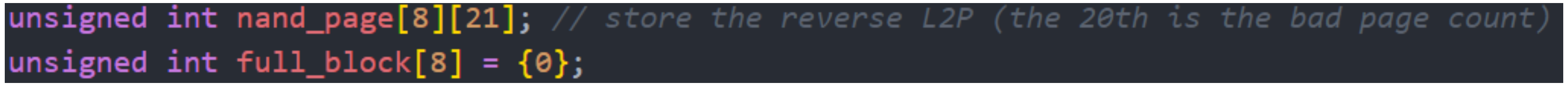
ssd_do_write()

not align


blue: overwrite, green: new, gray : empty (old), black: full (old)
read
write
read
write



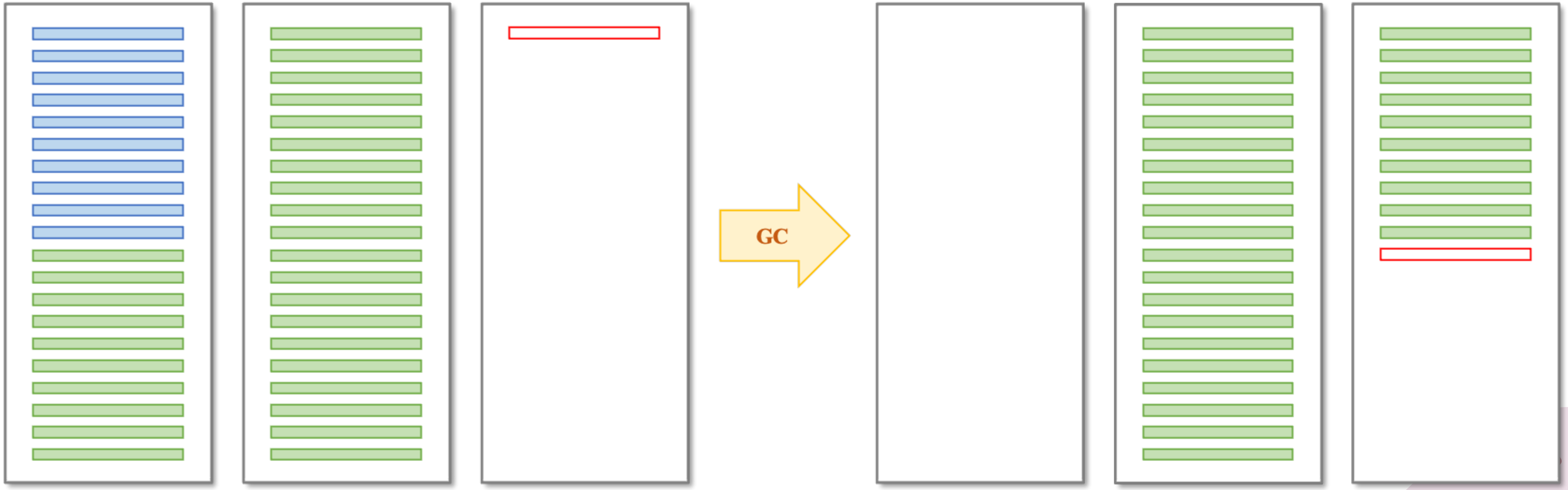
```
else{
    // We have to read the data first
    char* tmp_buf = calloc(512, sizeof(char)); //allocate a temporary buffer t
    rst = ftl_read(tmp_buf, tmp_lba + idx);
    if (rst < 0)
    {
        //error
        free(tmp_buf);
        return rst;
    }
    //overwrite the overlap part
    if(offset % 512 != 0 && remain_size < 512)
        memcpy(tmp_buf + offset % 512, buf + process_size, remain_size); //copy
    else if(offset % 512 != 0)
        memcpy(tmp_buf, buf + process_size, 512 - offset % 512); //copy the dat
    else if(remain_size < 512)
        memcpy(tmp_buf + offset % 512, buf + process_size, remain_size); //copy
    //write the data back to nand
    rst = ftl_write(tmp_buf, 1, tmp_lba + idx); //write the data back to nand
    free(tmp_buf);
    if ( rst == 0 )
    {
        //write full return -enomem;
        return -ENOMEM;
    }
    else if (rst < 0)
    {
        //error
        return rst;
    }
    curr_size += 512 - offset % 512;
    remain_size -= 512 - offset % 512;
    process_size += 512 - offset % 512;
    offset += 512 - offset % 512;
}
```



```
unsigned int nand_page[8][21]; // store the reverse L2P (the 20th is the bad page count)
unsigned int full_block[8] = {0};
```

Garbage Collection

blue: overwrite, green: new, red : next pca



Optimization (1/3)

- **Perform garbage collection on the NAND with the most garbage**: When one nand is left empty, we will find the most overwritten nand and do gc, to minimize the number of times it is moved.

Optimization (2/3)

- **Increase logical size**: Because garbage collection is only triggered when there is only one NAND left, the logical size can be expanded to two less than the physical size.

Optimization (3/3)

- **Read-Write Balance**: Because our get_next_pca() function operates in a loop, it ensures that each NAND undergoes read and write operations. However, we do not consider situations where certain NANDs have not been overwritten.

Thank you for listening!