Sector-based Ftl Simulator Report

2016311821 한승하

이번 Sector-based ftl 에선 지난번 page mapping ftl simulator코드의 많은 부분을 그대로 사용하였습니다. 따라서 지난 코드에서 sector base read, write를 지원하게 하는 수정부분 위주로 서술하였습니다.

<Last Lab>

```
void lftl_read(u32 lpn, u32 *read_buffer)
{
        int bank = lpn%N_BANKS;
        u32 addr = l2p[lpn].ppn;
        int blk = (addr - bank*BLKS_PER_BANK*PAGES_PER_BLK)/PAGES_PER_BLK;
        int page = (addr - bank*BLKS_PER_BANK*PAGES_PER_BLK - blk*PAGES_PER_BLK);
        u32 spare;
        nand_read(bank,blk,page,read_buffer,&spare);
        return;
}

void lftl_write(u32 lpn, u32 *write_buffer)
{
        int bank = lpn%N_BANKS;
        if(freeblocks[bank] == 1) garbage_collection(bank);
        if(l2p[lpn].v != 'n')
        {
                u32 addr;
                addr = l2p[lpn].ppn;
                ppn[addr].v = 'i';
                ppn[addr].p_age = now();
        }
        if((ptr[bank]+1)%(BLKS_PER_BANK) == 0) freeblocks[bank]--;
        int blk = ptr[bank]/PAGES_PER_BLK;
        int page = ptr[bank]%PAGES_PER_BLK;
        l2p[lpn].ppn = bank*BLKS_PER_BANK*PAGES_PER_BLK + blk*PAGES_PER_BLK + page;
        l2p[lpn].v = 'w';
        nand_write(bank,blk,page,write_buffer,lpn);
        ppn[bank*BLKS_PER_BANK*PAGES_PER_BLK+ptr[bank]].v = 'v';
        if(ptr[bank] == BLKS_PER_BANK*PAGES_PER_BLK-1)
        {
                ptr[bank] = 0;
                while(ppn[bank*BLKS_PER_BANK*PAGES_PER_BLK + ptr[bank]].v != 'f') ptr[bank]++;
        }
        else ptr[bank]++;
        return;
}
```

이번 과제에서 ftl_read와 ftl_write함수가 따로 정의되어 있기에 lftl_read와 lftl_write으로 지난번 과제에서의 함수들을 사용했습니다. Lftl_write에 경우 지난과제에서 잘못 구현했던 age에 관한 부분을 page가 invalid 되는 타이밍에 age가 now로 기록될 수 있도록 수정해 주었습니다.

<Ftl Open>

Ftl_open 함수는 이전 과제와 동일합니다.

<Ftl Read>

```
void ftl_read(u32 lba, u32 num_sectors, u32 *read_buffer)
{
        //printf("read %d!\n",num_sectors);
        u32 *buffer = (u32 *)calloc(SECTORS_PER_PAGE, sizeof(u32));
        int i;
        u32 read_buffer_ptr = 0;
        u32 num_sectors_to_read;
        u32 lpn = lba / SECTORS_PER_PAGE;
        u32 sect_offset = lba % SECTORS_PER_PAGE;
        u32 sectors_remain = num_sectors;
        while(sectors_remain != 0)
        {
                if(sect_offset + sectors_remain < SECTORS_PER_PAGE) num_sectors_to_read = sectors_remain;
                else num_sectors_to_read = SECTORS_PER_PAGE - sect_offset;
                lftl_read(lpn,buffer);
                for(i=0;i<num_sectors_to_read;i++)
                {
                        read_buffer[read_buffer_ptr + i] = buffer[sect_offset + i];
                }
                read_buffer_ptr += num_sectors_to_read;
                sect_offset = 0;
                sectors_remain -=num_sectors_to_read;
                lpn++;
        }
        return;
}
```

Ftl_Read와 Ftl_write함수를 Sector_based로 동작하게 하는 코드는 상당부분 Open_ssd firmware를 참고하여 구현했습니다. Lba/sectors_per_page를 통해 지난시간 만들었던 lpn table을 그대로 사용할 수 있도록 하였고, while문을 돌며, 읽고자 하는 범위가 page보다 작을 경우 남은 sector전부를, page범위를 벗어날 경우 page범위로 분할하여 읽을 수 있게 해 주었습니다. read범위가 page범위를 넘어갈 경우 1page의 단위 (Sector_per_page)에서 offset을 빼주어 한 page단위의 read로 반복할 수 있게 해 주었습니다. 이는 sectors_remain이 0이 될 때 까지 즉 더 읽어와야 하는 sector가 없을 때 까지 반복합니다.

Read_buffer_ptr은 매 cycle마다 읽어온 sector만큼 증가하며 이를 통해 Return해야 하는 Read_bufffer에 값이 쓰여야 하는 위치를 나타낼 수 있게 하였고, page_read에 대해서는 지난 Lab 과제에서 사용하였던 lftl_read함수를 사용하였습니다.

Buffer를 추가로 선언해 page단위로 값을 읽어와 read_buffer에 전달할 수 있게 해주었습니다.

<Ftl Write>

```
void ftl_write(u32 lba, u32 num_sectors, u32 *write_buffer)
{
        //printf("write %d!\n",num_sectors);
        u32 num_sectors_to_write;
        u32 *buffer = (u32 *)calloc(SECTORS_PER_PAGE, sizeof(u32));
        int i;
        u32 write_buffer_ptr = 0;
        u32 lpn = lba / SECTORS_PER_PAGE;
        u32 sect_offset = lba % SECTORS_PER_PAGE;
        u32 remain_sectors = num_sectors;
        while(remain_sectors != 0)
        {
                for(i=0;i<SECTORS_PER_PAGE;i++) buffer[i] = 0;
                if(sect_offset + remain_sectors >= SECTORS_PER_PAGE) num_sectors_to_write = SECTORS_PER_PAGE - sect_offset;
                else num_sectors_to_write = remain_sectors;
                if(sect_offset != 0 && l2p[lpn].v != 'n')
                {
                        lftl_read(lpn,buffer);
                }
                for(i=0;i<num_sectors_to_write;i++)
                {
                        buffer[sect_offset + i] = write_buffer[write_buffer_ptr + i];
                }
                lftl_write(lpn,buffer);
                s.ftl_write += SECTORS_PER_PAGE;
                write_buffer_ptr += num_sectors_to_write;
                sect_offset = 0;
                remain_sectors -= num_sectors_to_write;
                lpn++;
        }
        return;
}
```

Ftl_write함수입니다. Ftl_read와 마찬가지로 lpn, sect_offset, remain_sectors, buffer를 통해 page단위로 분할하여 write할 수 있게 해주었습니다.

전체적인 구조는 ftl_read와 동일하나, write의 경우 page에 이미 쓰여진 sectors를 보존해야 하기에, sector_offset이 0이 아니며 (sector_offset이 0이면 page의 맨 첫 sector부터 작성해야 하므로 읽어올 필요가 없음) l2p[lpn].v 가 n 이 아닌 경우 (l2p array는 지난 lab과제에서 만들어 놓은 logical address to physical address의 mapping table이며, v값은 visited으로 lpn에 대한 접근이 최초인지 아닌지를 판단합니다. Lpn에 대해 접근한 적이 없으면 v값이 n으로 초기화 되어 있으며, 접근한적이 있는 lpn에 대해서는 v값을 w로 바꾸어 주었습니다.) page를 읽어와, sector_offset (write하기 시작하는 sector) 이전 값은 그대로 보존하여 page단위로 write 할 수 있게 해주었습니다. Write의 경우에도 지난 과제에서 사용했던 page_based ftl_write함수를 사용해 주었습니다.

page단위 분할에 대해서는 ftl_read와 같은 방식을 사용하였고, write_buffer_ptr을 통해 매 cycle마다 기록해야 하는 값을 buffer로 옮겨 기록할 수 있게 해 주었습니다.

<Garbage Collection>

```
for(i=0;i<BLKS_PER_BANK;i++)
{
        count = 0;
        nfree = 0;
        age = 0;
        for(j=0;j<PAGES_PER_BLK;j++)
        {
                if((ppn[s_point+i*PAGES_PER_BLK+j].v == 'v') && (ppn[s_point+i*PAGES_PER_BLK+j].p_age > age))
                age = ppn[s_point+i*PAGES_PER_BLK+j].p_age;
                if(ppn[s_point+i*PAGES_PER_BLK+j].v == 'i') count++;
                if(ppn[s_point+i*PAGES_PER_BLK+j].v == 'f') nfree++;
        }
        if(count == PAGES_PER_BLK)
        {
                victim = i;
                st = count * ltime;
        }
        else    cost_b = (count*(ltime-age))/(PAGES_PER_BLK - count);
        if(cost_b>st)
        {
                victim = i;
                st = cost_b;
        }
        if(nfree == BLKS_PER_BANK){
                freeblk = i;
        }
}
```

<이전과제 ftl.c GC Cost_benefit의 일부>

```
for(i=0;i<BLKS_PER_BANK;i++)
{
        count = 0;
        nfree = 0;
        age = 0;
        for(j=0;j<PAGES_PER_BLK;j++)
        {
                if((ppn[s_point+i*PAGES_PER_BLK+j].v == 'v') && (ppn[s_point+i*PAGES_PER_BLK+j].p_age > age))
                age = ppn[s_point+i*PAGES_PER_BLK+j].p_age;
                if(ppn[s_point+i*PAGES_PER_BLK+j].v == 'i') count++;
                if(ppn[s_point+i*PAGES_PER_BLK+j].v == 'f') nfree++;
        }
        if(count == PAGES_PER_BLK)
        {
                victim = i;
                st = count * now();
        }
        else cost_b = (count*(now()-age))/(PAGES_PER_BLK - count);
        if(cost_b>st)
        {
                victim = i;
                st = cost_b;
        }
        if(nfree == BLKS_PER_BANK){
                freeblk = i;
        }
}
```

<수정본>

Garbage Collection은 지난 과제와 상당히 유사하나, cost_benefit일 경우에 대해 지난과제에 잘못 구현했었던 write시 age갱신을 삭제하고, logical time변수로 사용하던 ltime을 삭제하고, now()함수로 대체해 주었습니다.

<Result Analize>

- Write Amplification Factor (WAF)

$$= \frac{Bytes\ written\ to\ Flash}{Bytes\ written\ from\ Host} = \frac{Bytes\ written\ from\ Host + Bytes\ written\ during\ GC}{Bytes\ writen\ from\ Host}$$

Host가 요청한 write request에 대한 GC로 인한 실제 일어나는 write의 비유율을 나타내는 WAF 값은 GC의 성능을 알려주는 지표가 됩니다. 하지만 현재 구현한 Sector_based ftl은 write가 page_aligned 되어있지 않으므로, 실제 요청한 sector 보다 더 많은 Sector를 기록해야 하며 이에 따라 page_aligned ftl보다 WAF의 값은 더욱 커질 수밖에 없습니다.

```
                printf("[Run %d] host %ld, ftl %ld, valid page copy %ld, GC# %d, WAF %.2f\n", \
                        (int)iter/N_LPNS, s.host_write, s.ftl_write, s.gc_write / SECTORS_PER_PAGE, s.gc, (double)(s.ftl_write+s.gc_write)/
(double)s.host_write);
```

본 과제에서 WAF값은 (s.ftl_write + s.gc_write) / s.host_write 으로 정의되어 있습니다.

<Greedy – Random>

```
han@han-VirtualBox:~/다운로드/2016311821$ ./ftl_sim
Bank: 2
Blocks / Bank: 32 blocks
Pages / Block: 32 pages
OP ratio: 7%
Physical Blocks: 64
User Blocks: 56
OP Blocks: 8
PPNs: 2048
LPNs: 1792
Workload: Random
FTL: Greedy policy
Max Sectors: 32

[Run 1] host 29505, ftl 42040, valid page copy 4441, GC# 241, WAF 2.63
[Run 2] host 58945, ftl 83952, valid page copy 21312, GC# 932, WAF 4.32
[Run 3] host 88378, ftl 126024, valid page copy 41078, GC# 1714, WAF 5.14
[Run 4] host 118058, ftl 168224, valid page copy 60633, GC# 2490, WAF 5.53
[Run 5] host 147609, ftl 210072, valid page copy 80393, GC# 3271, WAF 5.78
[Run 6] host 176615, ftl 251544, valid page copy 99780, GC# 4039, WAF 5.94
[Run 7] host 206005, ftl 293704, valid page copy 119539, GC# 4821, WAF 6.07
[Run 8] host 235649, ftl 335920, valid page copy 139381, GC# 5606, WAF 6.16
[Run 9] host 265513, ftl 378256, valid page copy 159337, GC# 6395, WAF 6.23
[Run 10] host 295240, ftl 420568, valid page copy 179100, GC# 7178, WAF 6.28
```

```
[Run 66] host 1950379, ftl 2776920, valid page copy 1281522, GC# 50833, WAF 6.68
[Run 67] host 1980499, ftl 2819456, valid page copy 1301227, GC# 51615, WAF 6.68
[Run 68] host 2010274, ftl 2861912, valid page copy 1320917, GC# 52396, WAF 6.68
[Run 69] host 2039896, ftl 2904200, valid page copy 1340716, GC# 53180, WAF 6.68
[Run 70] host 2069241, ftl 2946160, valid page copy 1360428, GC# 53960, WAF 6.68
[Run 71] host 2097919, ftl 2987264, valid page copy 1379542, GC# 54718, WAF 6.68
[Run 72] host 2127255, ftl 3029008, valid page copy 1399416, GC# 55502, WAF 6.69
[Run 73] host 2157069, ftl 3071424, valid page copy 1419072, GC# 56282, WAF 6.69
[Run 74] host 2186740, ftl 3113488, valid page copy 1438773, GC# 57062, WAF 6.69
[Run 75] host 2216049, ftl 3155400, valid page copy 1458398, GC# 57839, WAF 6.69
[Run 76] host 2245465, ftl 3197128, valid page copy 1477698, GC# 58605, WAF 6.69
[Run 77] host 2274918, ftl 3239184, valid page copy 1497402, GC# 59385, WAF 6.69
[Run 78] host 2304098, ftl 3280640, valid page copy 1516762, GC# 60152, WAF 6.69
[Run 79] host 2334208, ftl 3323568, valid page copy 1536744, GC# 60944, WAF 6.69
[Run 80] host 2362733, ftl 3364488, valid page copy 1555785, GC# 61699, WAF 6.69
[Run 81] host 2391778, ftl 3406056, valid page copy 1575452, GC# 62476, WAF 6.69
[Run 82] host 2421387, ftl 3448128, valid page copy 1594930, GC# 63249, WAF 6.69
[Run 83] host 2451009, ftl 3490504, valid page copy 1614497, GC# 64026, WAF 6.69
[Run 84] host 2480936, ftl 3533160, valid page copy 1634735, GC# 64825, WAF 6.70
[Run 85] host 2511005, ftl 3575648, valid page copy 1654451, GC# 65607, WAF 6.70
[Run 86] host 2541065, ftl 3618264, valid page copy 1674147, GC# 66389, WAF 6.69
[Run 87] host 2570491, ftl 3660312, valid page copy 1693879, GC# 67170, WAF 6.70
[Run 88] host 2600009, ftl 3702416, valid page copy 1713767, GC# 67956, WAF 6.70
[Run 89] host 2628770, ftl 3743736, valid page copy 1732953, GC# 68717, WAF 6.70
[Run 90] host 2658322, ftl 3785896, valid page copy 1753024, GC# 69509, WAF 6.70
[Run 91] host 2687874, ftl 3827832, valid page copy 1772718, GC# 70288, WAF 6.70
[Run 92] host 2717383, ftl 3869848, valid page copy 1792711, GC# 71077, WAF 6.70
[Run 93] host 2746829, ftl 3911880, valid page copy 1812416, GC# 71857, WAF 6.70
[Run 94] host 2777146, ftl 3954680, valid page copy 1832250, GC# 72644, WAF 6.70
[Run 95] host 2806583, ftl 3996776, valid page copy 1851887, GC# 73422, WAF 6.70
[Run 96] host 2836397, ftl 4039136, valid page copy 1871516, GC# 74201, WAF 6.70
[Run 97] host 2865417, ftl 4080864, valid page copy 1891196, GC# 74979, WAF 6.70
[Run 98] host 2894179, ftl 4122264, valid page copy 1910568, GC# 75746, WAF 6.71
[Run 99] host 2923674, ftl 4164376, valid page copy 1929944, GC# 76516, WAF 6.71
[Run 100] host 2953017, ftl 4206352, valid page copy 1949653, GC# 77296, WAF 6.71

Results ------
Host write sectors: 2953017
FTL write sectors: 4206352
GC write sectors: 15597224
Number of GCs: 77296
Valid pages per GC: 25.22 pages
WAF: 6.71
```

Greedy Policy의 경우, Random한 write에 대해 WAF값이 6.71을 기록하였습니다. 이는 Sector write 1회 요청당 6.71회의 실제 sector write가 일어났다는 것을 나타냅니다.

<Greedy-Hot-Cold>

```
han@han-VirtualBox:~/다운로드/2016311821$ ./ftl_sim
Bank: 2
Blocks / Bank: 32 blocks
Pages / Block: 32 pages
OP ratio: 7%
Physical Blocks: 64
User Blocks: 56
OP Blocks: 8
PPNs: 2048
LPNs: 1792
Workload: Hot 96 / Cold 4
FTL: Greedy policy
Max Sectors: 32

[Run 1] host 29695, ftl 42384, valid page copy 127, GC# 108, WAF 1.46
[Run 2] host 59446, ftl 84864, valid page copy 849, GC# 297, WAF 1.54
[Run 3] host 89222, ftl 127160, valid page copy 2135, GC# 502, WAF 1.62
[Run 4] host 118726, ftl 169256, valid page copy 4104, GC# 728, WAF 1.70
[Run 5] host 148722, ftl 211728, valid page copy 6859, GC# 980, WAF 1.79
[Run 6] host 178811, ftl 254216, valid page copy 10510, GC# 1260, WAF 1.89
[Run 7] host 208126, ftl 295904, valid page copy 14994, GC# 1563, WAF 2.00
[Run 8] host 238032, ftl 338280, valid page copy 20445, GC# 1899, WAF 2.11
[Run 9] host 268037, ftl 380776, valid page copy 26952, GC# 2268, WAF 2.23
[Run 10] host 297668, ftl 423104, valid page copy 34233, GC# 2661, WAF 2.34
[Run 11] host 328040, ftl 466024, valid page copy 42339, GC# 3082, WAF 2.45
[Run 12] host 357495, ftl 508120, valid page copy 51032, GC# 3518, WAF 2.56
```

```
[Run 66] host 1949960, ftl 2777040, valid page copy 1428960, GC# 55441, WAF 7.29
[Run 67] host 1979572, ftl 2819320, valid page copy 1460091, GC# 56579, WAF 7.32
[Run 68] host 2009080, ftl 2861464, valid page copy 1489515, GC# 57663, WAF 7.36
[Run 69] host 2038772, ftl 2903648, valid page copy 1519629, GC# 58769, WAF 7.39
[Run 70] host 2068424, ftl 2945504, valid page copy 1549348, GC# 59861, WAF 7.42
[Run 71] host 2097781, ftl 2987504, valid page copy 1579135, GC# 60956, WAF 7.45
[Run 72] host 2126660, ftl 3028728, valid page copy 1608860, GC# 62046, WAF 7.48
[Run 73] host 2155972, ftl 3070456, valid page copy 1639418, GC# 63164, WAF 7.51
[Run 74] host 2185160, ftl 3112224, valid page copy 1669368, GC# 64263, WAF 7.54
[Run 75] host 2214279, ftl 3153848, valid page copy 1699558, GC# 65369, WAF 7.56
[Run 76] host 2243226, ftl 3195336, valid page copy 1729542, GC# 66468, WAF 7.59
[Run 77] host 2273218, ftl 3238048, valid page copy 1760807, GC# 67612, WAF 7.62
[Run 78] host 2303429, ftl 3280920, valid page copy 1791446, GC# 68737, WAF 7.65
[Run 79] host 2332585, ftl 3322752, valid page copy 1821323, GC# 69834, WAF 7.67
[Run 80] host 2362097, ftl 3364904, valid page copy 1850995, GC# 70926, WAF 7.69
[Run 81] host 2391095, ftl 3406536, valid page copy 1880003, GC# 71995, WAF 7.71
[Run 82] host 2421058, ftl 3449056, valid page copy 1910398, GC# 73111, WAF 7.74
[Run 83] host 2450829, ftl 3491288, valid page copy 1942176, GC# 74269, WAF 7.76
[Run 84] host 2480073, ftl 3533024, valid page copy 1971713, GC# 75355, WAF 7.78
[Run 85] host 2508904, ftl 3574584, valid page copy 2001938, GC# 76462, WAF 7.81
[Run 86] host 2538121, ftl 3616384, valid page copy 2031981, GC# 77564, WAF 7.83
[Run 87] host 2567161, ftl 3657824, valid page copy 2061135, GC# 78637, WAF 7.85
[Run 88] host 2596269, ftl 3699408, valid page copy 2090466, GC# 79716, WAF 7.87
[Run 89] host 2625906, ftl 3741632, valid page copy 2120164, GC# 80809, WAF 7.88
[Run 90] host 2655578, ftl 3783696, valid page copy 2149756, GC# 81898, WAF 7.90
[Run 91] host 2685011, ftl 3825704, valid page copy 2179415, GC# 82989, WAF 7.92
[Run 92] host 2714491, ftl 3867568, valid page copy 2209188, GC# 84083, WAF 7.94
[Run 93] host 2743841, ftl 3909672, valid page copy 2238964, GC# 85178, WAF 7.95
[Run 94] host 2774021, ftl 3952632, valid page copy 2269182, GC# 86290, WAF 7.97
[Run 95] host 2803384, ftl 3994512, valid page copy 2299368, GC# 87397, WAF 7.99
[Run 96] host 2832794, ftl 4036392, valid page copy 2328889, GC# 88483, WAF 8.00
[Run 97] host 2862090, ftl 4078328, valid page copy 2358333, GC# 89567, WAF 8.02
[Run 98] host 2891681, ftl 4120488, valid page copy 2388422, GC# 90672, WAF 8.03
[Run 99] host 2921229, ftl 4162456, valid page copy 2419081, GC# 91794, WAF 8.05
[Run 100] host 2950614, ftl 4204192, valid page copy 2448968, GC# 92891, WAF 8.06

Results ------
Host write sectors: 2950614
FTL write sectors: 4204192
GC write sectors: 19591744
Number of GCs: 92891
Valid pages per GC: 26.36 pages
WAF: 8.06
```

같은 Greedy Policy에서 Hot-cold으로 실험해본 결과, WAF값은 8.06을 기록하여 실제 host에서 요청한 Sector대비 8배의 더 많은 Sector에 대한 Write가 일어났다는 것을 알 수 있습니다. Hot-cold와 Random모두 page에 기록되어 있는 sector를 읽어 온 후 기록을 해야 하기에 ftl_wirte시 write횟수는 비슷할 것이므로 (실제로 ftl write sector 값은 두 결과 모두 유사합니다) WAF값의 차이는 Hot-cold write로 인한 GC의 증가라고 볼 수 있습니다. 실제로 GC는 random 대비 약 15000 회 더 많이 일어났습니다.

<Cost_benifit – Random>

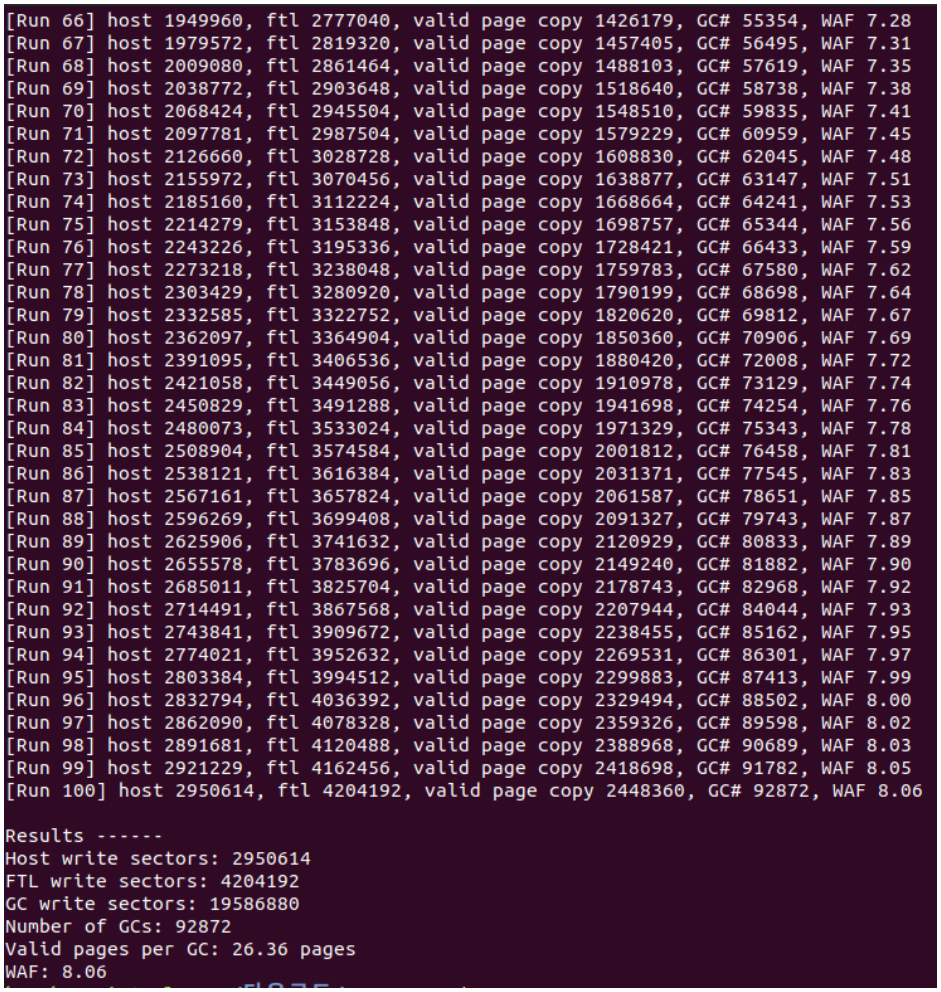
```
han@han-VirtualBox:~/다운로드/2016311821$ ./ftl_sim
Bank: 2
Blocks / Bank: 32 blocks
Pages / Block: 32 pages
OP ratio: 7%
Physical Blocks: 64
User Blocks: 56
OP Blocks: 8
PPNs: 2048
LPNs: 1792
Workload: Random
FTL: Cost-Benefit policy
Max Sectors: 32

[Run 1] host 29505, ftl 42040, valid page copy 4441, GC# 241, WAF 2.63
[Run 2] host 58945, ftl 83952, valid page copy 21312, GC# 932, WAF 4.32
[Run 3] host 88378, ftl 126024, valid page copy 41078, GC# 1714, WAF 5.14
[Run 4] host 118058, ftl 168224, valid page copy 60633, GC# 2490, WAF 5.53
[Run 5] host 147609, ftl 210072, valid page copy 80393, GC# 3271, WAF 5.78
[Run 6] host 176615, ftl 251544, valid page copy 99780, GC# 4039, WAF 5.94
[Run 7] host 206005, ftl 293704, valid page copy 119539, GC# 4821, WAF 6.07
[Run 8] host 235649, ftl 335920, valid page copy 139381, GC# 5606, WAF 6.16
[Run 9] host 265513, ftl 378256, valid page copy 159337, GC# 6395, WAF 6.23
[Run 10] host 295240, ftl 420568, valid page copy 179100, GC# 7178, WAF 6.28
```

```
[Run 66] host 1950379, ftl 2776920, valid page copy 1281522, GC# 50833, WAF 6.68
[Run 67] host 1980499, ftl 2819456, valid page copy 1301227, GC# 51615, WAF 6.68
[Run 68] host 2010274, ftl 2861912, valid page copy 1320917, GC# 52396, WAF 6.68
[Run 69] host 2039896, ftl 2904200, valid page copy 1340716, GC# 53180, WAF 6.68
[Run 70] host 2069241, ftl 2946160, valid page copy 1360428, GC# 53960, WAF 6.68
[Run 71] host 2097919, ftl 2987264, valid page copy 1379542, GC# 54718, WAF 6.68
[Run 72] host 2127255, ftl 3029008, valid page copy 1399416, GC# 55502, WAF 6.69
[Run 73] host 2157069, ftl 3071424, valid page copy 1419072, GC# 56282, WAF 6.69
[Run 74] host 2186740, ftl 3113488, valid page copy 1438773, GC# 57062, WAF 6.69
[Run 75] host 2216049, ftl 3155400, valid page copy 1458398, GC# 57839, WAF 6.69
[Run 76] host 2245465, ftl 3197128, valid page copy 1477698, GC# 58605, WAF 6.69
[Run 77] host 2274918, ftl 3239184, valid page copy 1497402, GC# 59385, WAF 6.69
[Run 78] host 2304098, ftl 3280640, valid page copy 1516762, GC# 60152, WAF 6.69
[Run 79] host 2334208, ftl 3323568, valid page copy 1536744, GC# 60944, WAF 6.69
[Run 80] host 2362733, ftl 3364488, valid page copy 1555785, GC# 61699, WAF 6.69
[Run 81] host 2391778, ftl 3406056, valid page copy 1575452, GC# 62476, WAF 6.69
[Run 82] host 2421387, ftl 3448128, valid page copy 1594930, GC# 63249, WAF 6.69
[Run 83] host 2451009, ftl 3490504, valid page copy 1614497, GC# 64026, WAF 6.69
[Run 84] host 2480936, ftl 3533160, valid page copy 1634735, GC# 64825, WAF 6.70
[Run 85] host 2511005, ftl 3575648, valid page copy 1654451, GC# 65607, WAF 6.70
[Run 86] host 2541065, ftl 3618264, valid page copy 1674147, GC# 66389, WAF 6.69
[Run 87] host 2570491, ftl 3660312, valid page copy 1693879, GC# 67170, WAF 6.70
[Run 88] host 2600009, ftl 3702416, valid page copy 1713767, GC# 67956, WAF 6.70
[Run 89] host 2628770, ftl 3743736, valid page copy 1732953, GC# 68717, WAF 6.70
[Run 90] host 2658322, ftl 3785896, valid page copy 1753024, GC# 69509, WAF 6.70
[Run 91] host 2687874, ftl 3827832, valid page copy 1772718, GC# 70288, WAF 6.70
[Run 92] host 2717383, ftl 3869848, valid page copy 1792711, GC# 71077, WAF 6.70
[Run 93] host 2746829, ftl 3911880, valid page copy 1812416, GC# 71857, WAF 6.70
[Run 94] host 2777146, ftl 3954680, valid page copy 1832250, GC# 72644, WAF 6.70
[Run 95] host 2806583, ftl 3996776, valid page copy 1851887, GC# 73422, WAF 6.70
[Run 96] host 2836397, ftl 4039136, valid page copy 1871516, GC# 74201, WAF 6.70
[Run 97] host 2865417, ftl 4080864, valid page copy 1891196, GC# 74979, WAF 6.70
[Run 98] host 2894179, ftl 4122264, valid page copy 1910568, GC# 75746, WAF 6.71
[Run 99] host 2923674, ftl 4164376, valid page copy 1929944, GC# 76516, WAF 6.71
[Run 100] host 2953017, ftl 4206352, valid page copy 1949653, GC# 77296, WAF 6.71

Results ------
Host write sectors: 2953017
FTL write sectors: 4206352
GC write sectors: 15597224
Number of GCs: 77296
Valid pages per GC: 25.22 pages
WAF: 6.71
```

Cost_benefit Random에 경우에는 Greedy Random과 유사한 수치를 보였습니다. 실제 ftl write sectors는 420만번으로 유사한 write를 기록한 것으로 보아 sector_based ftl에서 cost_benefit과 greedy policy의 효율이 비슷한 것으로 판단할 수 있습니다.

<Cost_Benefit – Hot-Cold>

```
han@han-VirtualBox:~/다운로드/2016311821$ ./ftl_sim
Bank: 2
Blocks / Bank: 32 blocks
Pages / Block: 32 pages
OP ratio: 7%
Physical Blocks: 64
User Blocks: 56
OP Blocks: 8
PPNs: 2048
LPNs: 1792
Workload: Hot 96 / Cold 4
FTL: Cost-Benefit policy
Max Sectors: 32

[Run 1] host 29695, ftl 42384, valid page copy 127, GC# 108, WAF 1.46
[Run 2] host 59446, ftl 84864, valid page copy 848, GC# 297, WAF 1.54
[Run 3] host 89222, ftl 127160, valid page copy 2145, GC# 503, WAF 1.62
[Run 4] host 118726, ftl 169256, valid page copy 4100, GC# 728, WAF 1.70
[Run 5] host 148722, ftl 211728, valid page copy 6858, GC# 980, WAF 1.79
[Run 6] host 178811, ftl 254216, valid page copy 10568, GC# 1262, WAF 1.89
[Run 7] host 208126, ftl 295904, valid page copy 14957, GC# 1562, WAF 2.00
[Run 8] host 238032, ftl 338280, valid page copy 20325, GC# 1895, WAF 2.10
[Run 9] host 268037, ftl 380776, valid page copy 26797, GC# 2263, WAF 2.22
[Run 10] host 297668, ftl 423104, valid page copy 34142, GC# 2658, WAF 2.34
```



Cost_benefit Hot-Cold또한 Greedy때와 비슷한 증가율을 보이며, Hot-Cold로 인한 GC횟수의 증가 때문임을 알 수 있습니다. 두 Policy에서 두 Case모두 비슷한 WAF값을 보이며 두 Policy가 유사한 성능을 지니고 있음을 알 수 있습니다.