2022/06/10 ~ 2022/06/17

PM stall 관련 실험

Flush instruction 없는 PM DAX write 실험

▼ **Cache line 크기 고려한 buffer 문자열 설정**

- Cache line은 64 bytes 크기이므로 이를 고려하여 64 bytes보다 큰 주기로 문자열이 반복되게 하거나 아니면 아예 랜덤으로 긴 문자열을 생성시켜 주어야 한다.
- 워크로드 시간 측정 이전에 한 번만 진행해주면 되며, 큰 오버헤드가 없는 것으로 생각되므로 stdlib 헤더파일에 정의된 rand, srand 함수와 time 헤더파일에 정의된 time 함수를 이용해 다음과 같이 난수 생성을 통한 랜덤한 긴 문자열을 생성시켜 준다.

```
int rand_num = 0;
srand((unsigned int)time(NULL));

for(i = 0; i < buf_size_in_byte; i++){
  rand_num = rand()%64;
  buf[i] = '0' + rand_num;
}
```

▼ **파일 생성 코드**

```
//Generate 1GB size file for mmap_write_seq workload(The block size can be change, and it will match with the block size which is
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include <errno.h>

int main(){
  int fd;
  int i;
  char *map;
  int GB_to_KB = 1048576; //1 GB = 1048576 KB
  int KB_to_Byte = 1024; //1 KB = 1024 Bytes

  int buf_size_in_byte = 4096; //4KB
  char buf[4096] = {0, }; //4KB


  int iter_num = (GB_to_KB*KB_to_Byte)/buf_size_in_byte;

  for(i = 0; i < buf_size_in_byte; i++){
        buf[i] = 'w';
  }
  if((fd = open("/pmem/file1", O_RDWR|O_CREAT,0644)) < 0){
        printf("failed to open file\n");
        return 0;
   }
  printf("Starting 1G file generating6\n");
  struct timeval startTime, endTime;
  double diffTime;
  gettimeofday(&startTime, NULL);
  for(i = 0; i < iter_num; i++){
        write(fd, buf, 4096);
  }
  gettimeofday(&endTime, NULL);
  if(endTime.tv_usec < startTime.tv_usec){
        endTime.tv_usec += 1000000;
        endTime.tv_sec -= 1;
  }
  printf("The end of file generating operation6 : %ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.tv_us
  close(fd);
  return 0;
}
```

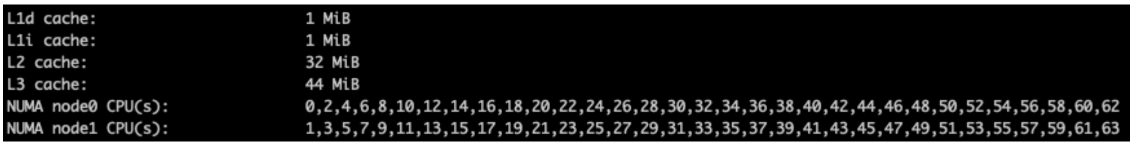
▼ **write 워크로드 코드**

총 20GB를 write하며, memcpy를 통해 buffer 크기만큼 sequential하게 1GB를 write하는 과정을 20번 반복한다

```
//Workload for writing to PM as DAX FS mode in sequencial blocks(the amount of sequential write size is restricted due to the limi
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <stdint.h>
#include <immintrin.h>

int main(){
  int fd;
  int i,j,k;
  char *map;
  int n_GB = 20;
  int GB_to_KB = 1048576; //1 GB = 1048576 KB
  int KB_to_Byte = 1024; //1 KB = 1024 Bytes
  int flush_iter = 4096/64;

  int buf_size_in_byte = 4096; //4KB:4096Bytes -> write in bytes
  char buf[4096] = {0, }; //4KB

  int iter_num = (GB_to_KB*KB_to_Byte)/buf_size_in_byte;
  int rand_num = 0;
  srand((unsigned int)time(NULL));

  for(i = 0; i < buf_size_in_byte; i++){
    rand_num = rand()%64;
    buf[i] = '0' + rand_num;
  }
  if((fd = open("/pmem/file1", O_RDWR|O_CREAT,0644)) < 0){
    printf("failed to open file\n");
    return 0;
  }

  //map = mmap(NULL, buf_size_in_byte, PROT_READ | PROT_WRITE, MAP_SHARED_VALIDATE|MAP_SYNC, fd, 0);
  map = mmap(NULL, GB_to_KB*KB_to_Byte, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0); //1GB mapping

  //printf("Starting mmap %dGB writing operation1\n", n_GB);
  struct timeval startTime, endTime, syncTime_before, syncTime_after, unmapTime_before, unmapTime_after;
  double diffTime, syncTime, unmapTime;
  gettimeofday(&startTime, NULL);

  for(i=0; i<n_GB; i++){
    for(j=0; j<iter_num; j++){ //1GB sequential write
      char *map_j = map + (buf_size_in_byte*j);
      memcpy(map_j, buf, buf_size_in_byte);
    }
  }

  gettimeofday(&endTime, NULL);
  if(endTime.tv_usec < startTime.tv_usec){
      endTime.tv_usec += 1000000;
      endTime.tv_sec -= 1;
  }
  //printf("The end of mmap write operation1 : %ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.tv_usec)
  printf("%ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.tv_usec);
  gettimeofday(&syncTime_before, NULL);
  msync(map, buf_size_in_byte, MS_SYNC);
  munmap(map, buf_size_in_byte);
  close(fd);
  return 0;
}
```

▼ 실험 환경

- Cache size & NUMA information



```
L1d cache:                       1 MiB
L1i cache:                       1 MiB
L2 cache:                        32 MiB
L3 cache:                        44 MiB
NUMA node0 CPU(s):               0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62
NUMA node1 CPU(s):               1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63
```

- PM0로 DAX write 진행
- jbd_2는 노드0의 8번 코어에 배치 (저널링 과정에서의 remote access 방지)
- 코어의 governer는 전부 performance mode로 설정

▼ 실험 과정

- PM0를 ext4로 포맷 후 DAX로 mount
- PM0에 1GB 크기의 파일 생성 (혹시 모를 요인을 배제하기 위해 파일 생성 프로세스는 노드0의 2번 코어에 배치)

  `taskset -c 2 ./(파일 생성 실행파일)`
- jbd_2의 PID를 알아내어 노드0의 8번 코어에 배치

  `ps -aux | grep jbd2 | grep pmem | grep -o '[0-9]*' | head -1`

  `taskset -cp 8 (jbd_2 PID)`
- cache drop

  `echo 3 > /proc/sys/vm/drop_caches`
- 20GB write 워크로드 실행 및 perf 프로파일링

  `perf stat -B -e cache-references,cache-misses,cycles,instructions,branches,faults,migrations,L1-dcache-load-misses,L1-dcache-loads,L1-dcache-stores,L1-icache-load-misses,LLC-loads,LLC-load-misses,LLC-stores,LLC-store-misses,LLC-prefetches,cycle_activity.stalls_l1d_miss,cycle_activity.stalls_l2_miss,cycle_activity.stalls_l3_miss,cycle_activity.stalls_tota`

  ` taskset -c 2 ./(워크로드 실행파일)`
- cf) 편의를 위해 단계 1~4까지를 하나의 bash script로 구성

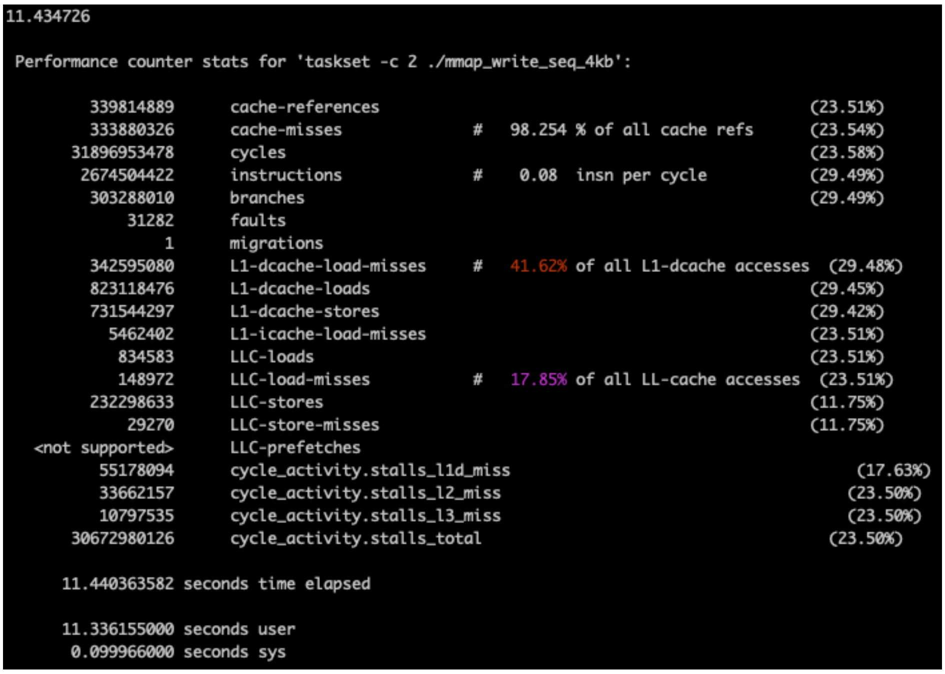
```
#!/bin/bash

rm /pmem/file1;
umount /pmem;
yes | mkfs.ext4 /dev/pmem0;
mount -o dax /dev/pmem0 /pmem;
taskset -c 2 ./mmap_generate_seq1;
JBDPID=$(ps -aux | grep jbd2 | grep pmem | grep -o '[0-9]*' | head -1)
BINDJBD='taskset -cp 8'
$BINDJBD $JBDPID
echo 3 > /proc/sys/vm/drop_caches;
```

- cf) L1d cache 크기가 1MiB이므로 buffer 크기(한 번에 memcpy로 내리는 크기)는 1MB 보다 작은 크기/1MB/1MB보다 큰 크기, 이렇게 3가지 종류로 설정함
  - 서버 사양인 L1d cache의 크기인 1MB보다 작은 4KB의 random 문자열을 반복적으로 sequential하게 write
  - 서버 사양인 L1d cache의 크기인 1MB보다 작은 256KB의 random 문자열을 반복적으로 sequential하게 write
  - 서버 사양인 L1d cache의 크기인 1MB와 동일한 1MB의 random 문자열을 반복적으로 sequential하게 write
  - 서버 사양인 L1d cache의 크기인 1MB보다 큰 2MB의 random 문자열을 반복적으로 sequential하게 write

▼ 실험 결과

- 4KB 단위 1GB sequential write

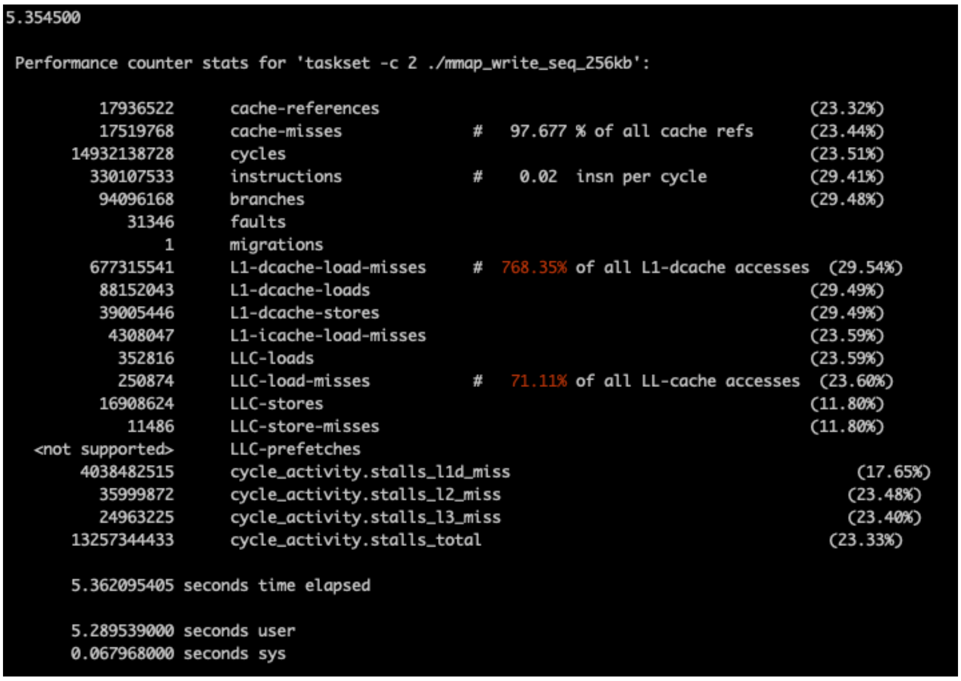
  ▼ perf 결과



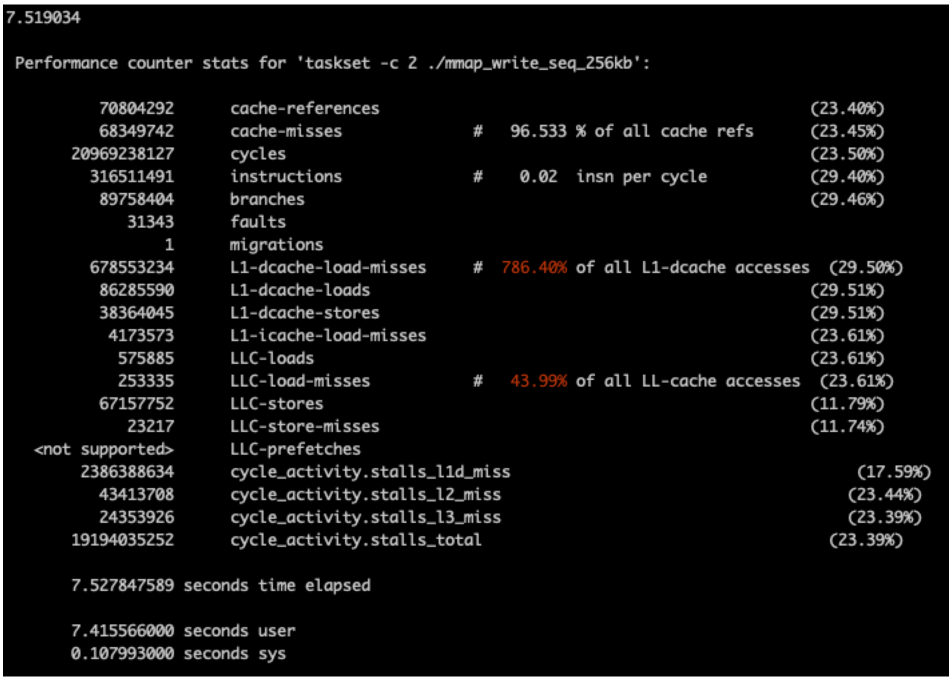
```
11.434726

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_4kb':

        339814889      cache-references                                              (23.51%)
        333880326      cache-misses              #   98.254 % of all cache refs      (23.54%)
      31896953478      cycles                                                        (23.58%)
       2674504422      instructions              #    0.08  insn per cycle           (29.49%)
        303288010      branches                                                      (29.49%)
            31282      faults
                1      migrations
        342595080      L1-dcache-load-misses     #   41.62% of all L1-dcache accesses  (29.48%)
        823118476      L1-dcache-loads                                               (29.45%)
        731544297      L1-dcache-stores                                              (29.42%)
          5462402      L1-icache-load-misses                                         (23.51%)
           834583      LLC-loads                                                     (23.51%)
           148972      LLC-load-misses           #   17.85% of all LL-cache accesses  (23.51%)
        232298633      LLC-stores                                                    (11.75%)
            29270      LLC-store-misses                                              (11.75%)
  <not supported>      LLC-prefetches
         55178094      cycle_activity.stalls_l1d_miss                                     (17.63%)
         33662157      cycle_activity.stalls_l2_miss                                     (23.50%)
         10797535      cycle_activity.stalls_l3_miss                                     (23.50%)
      30672980126      cycle_activity.stalls_total                                     (23.50%)

     11.440363582 seconds time elapsed

     11.336155000 seconds user
      0.099966000 seconds sys
```

- 256KB 단위 1GB sequential write

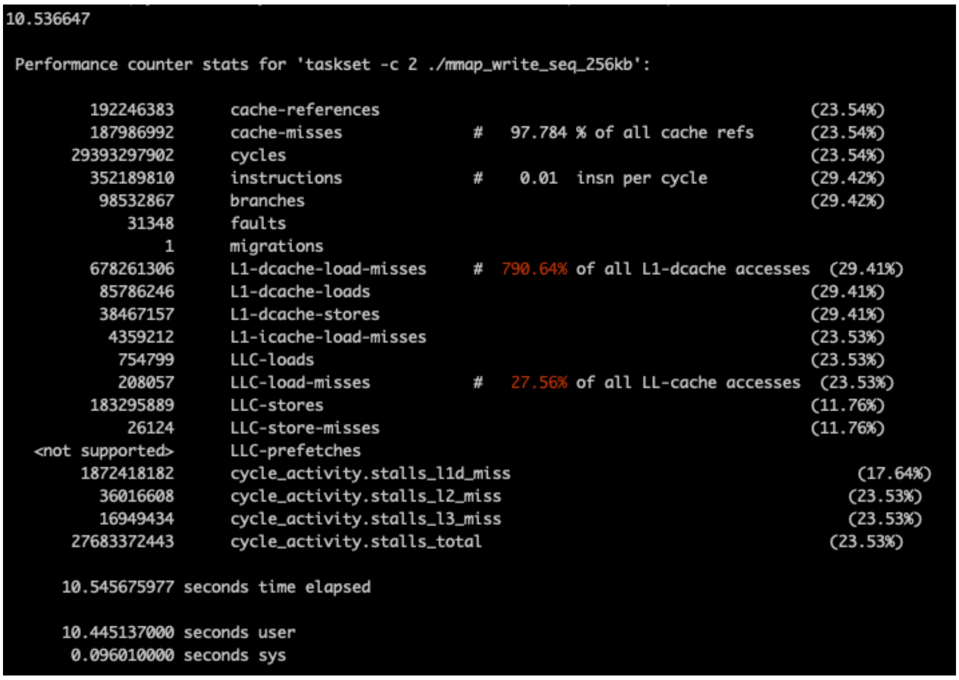
  ▼ perf 결과



```
5.354500

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_256kb':

         17936522      cache-references                                              (23.32%)
         17519768      cache-misses              #   97.677 % of all cache refs      (23.44%)
      14932138728      cycles                                                        (23.51%)
        330107533      instructions              #    0.02  insn per cycle           (29.41%)
         94096168      branches                                                      (29.48%)
            31346      faults
                1      migrations
        677315541      L1-dcache-load-misses     #  768.35% of all L1-dcache accesses  (29.54%)
         88152043      L1-dcache-loads                                               (29.49%)
         39005446      L1-dcache-stores                                              (29.49%)
          4308047      L1-icache-load-misses                                         (23.59%)
           352816      LLC-loads                                                     (23.59%)
           250874      LLC-load-misses           #   71.11% of all LL-cache accesses  (23.60%)
         16908624      LLC-stores                                                    (11.80%)
            11486      LLC-store-misses                                              (11.80%)
  <not supported>      LLC-prefetches
       4038482515      cycle_activity.stalls_l1d_miss                                     (17.65%)
         35999872      cycle_activity.stalls_l2_miss                                     (23.48%)
         24963225      cycle_activity.stalls_l3_miss                                     (23.40%)
      13257344433      cycle_activity.stalls_total                                     (23.33%)

      5.362095405 seconds time elapsed

      5.289539000 seconds user
      0.067968000 seconds sys
```



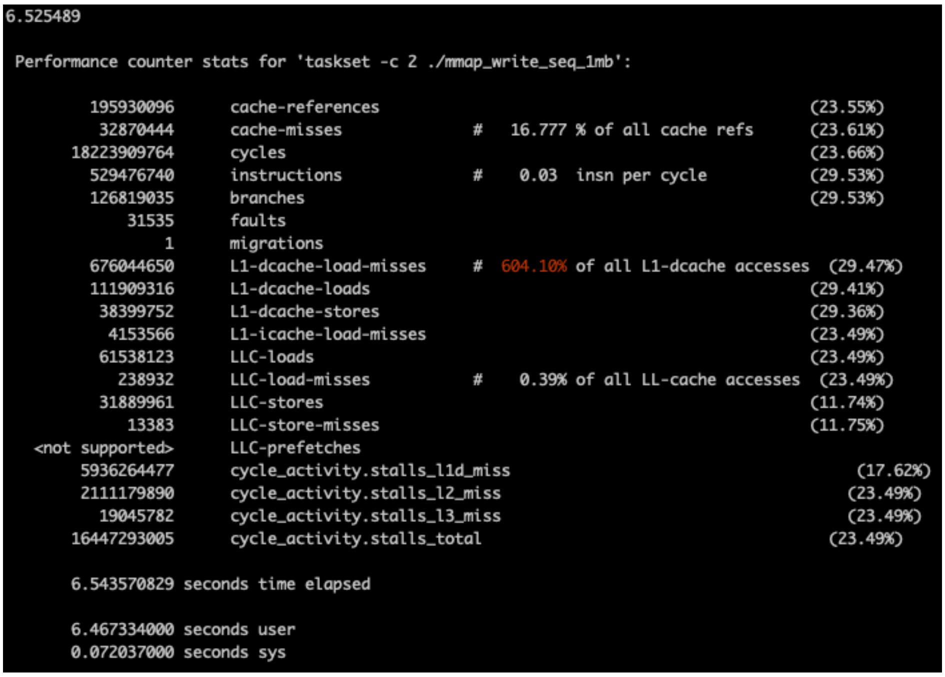
```
7.519034

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_256kb':

         70804292      cache-references                                              (23.40%)
         68349742      cache-misses              #   96.533 % of all cache refs      (23.45%)
      20969238127      cycles                                                        (23.50%)
        316511491      instructions              #    0.02  insn per cycle           (29.40%)
         89758404      branches                                                      (29.46%)
            31343      faults
                1      migrations
        678553234      L1-dcache-load-misses     #  786.40% of all L1-dcache accesses  (29.50%)
         86285590      L1-dcache-loads                                               (29.51%)
         38364045      L1-dcache-stores                                              (29.51%)
          4173573      L1-icache-load-misses                                         (23.61%)
           575885      LLC-loads                                                     (23.61%)
           253335      LLC-load-misses           #   43.99% of all LL-cache accesses  (23.61%)
         67157752      LLC-stores                                                    (11.79%)
            23217      LLC-store-misses                                              (11.74%)
   <not supported>      LLC-prefetches
       2386388634      cycle_activity.stalls_l1d_miss                                     (17.59%)
         43413708      cycle_activity.stalls_l2_miss                                    (23.44%)
         24353926      cycle_activity.stalls_l3_miss                                    (23.39%)
      19194035252      cycle_activity.stalls_total                                    (23.39%)

      7.527847589 seconds time elapsed

      7.415566000 seconds user
      0.107993000 seconds sys
```



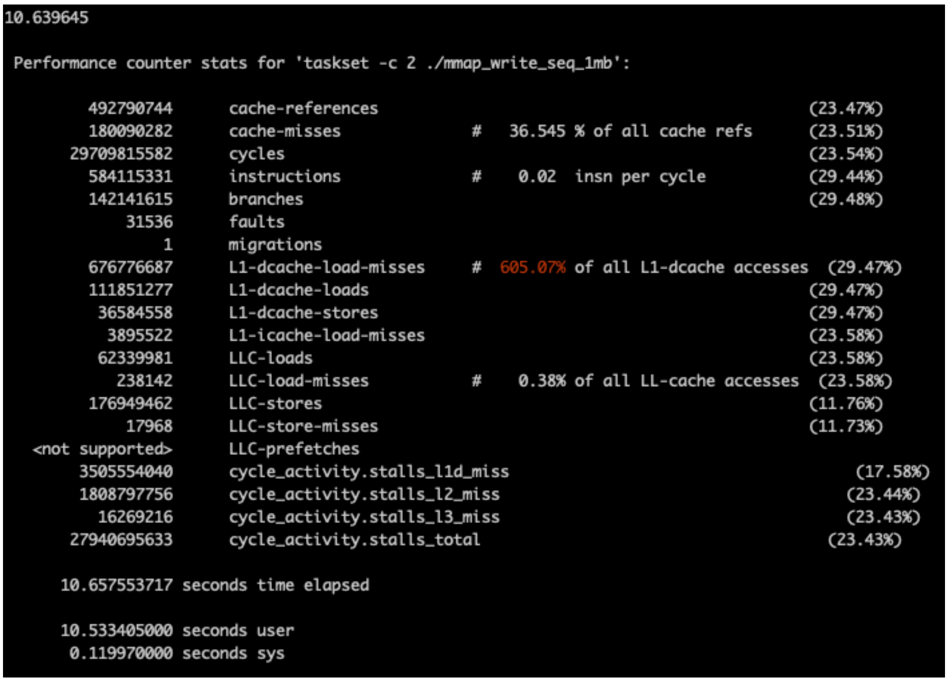
```
10.536647

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_256kb':

        192246383      cache-references                                              (23.54%)
        187986992      cache-misses              #   97.784 % of all cache refs      (23.54%)
      29393297902      cycles                                                        (23.54%)
        352189810      instructions              #    0.01  insn per cycle           (29.42%)
         98532867      branches                                                      (29.42%)
            31348      faults
                1      migrations
        678261306      L1-dcache-load-misses     #  790.64% of all L1-dcache accesses  (29.41%)
         85786246      L1-dcache-loads                                               (29.41%)
         38467157      L1-dcache-stores                                              (29.41%)
          4359212      L1-icache-load-misses                                         (23.53%)
           754799      LLC-loads                                                     (23.53%)
           208057      LLC-load-misses           #   27.56% of all LL-cache accesses  (23.53%)
        183295889      LLC-stores                                                    (11.76%)
            26124      LLC-store-misses                                              (11.76%)
   <not supported>      LLC-prefetches
       1872418182      cycle_activity.stalls_l1d_miss                                     (17.64%)
         36016608      cycle_activity.stalls_l2_miss                                    (23.53%)
         16949434      cycle_activity.stalls_l3_miss                                    (23.53%)
      27683372443      cycle_activity.stalls_total                                    (23.53%)

     10.545675977 seconds time elapsed

     10.445137000 seconds user
      0.096010000 seconds sys
```

- 1MB 단위 1GB sequential write

  ▼ perf 결과



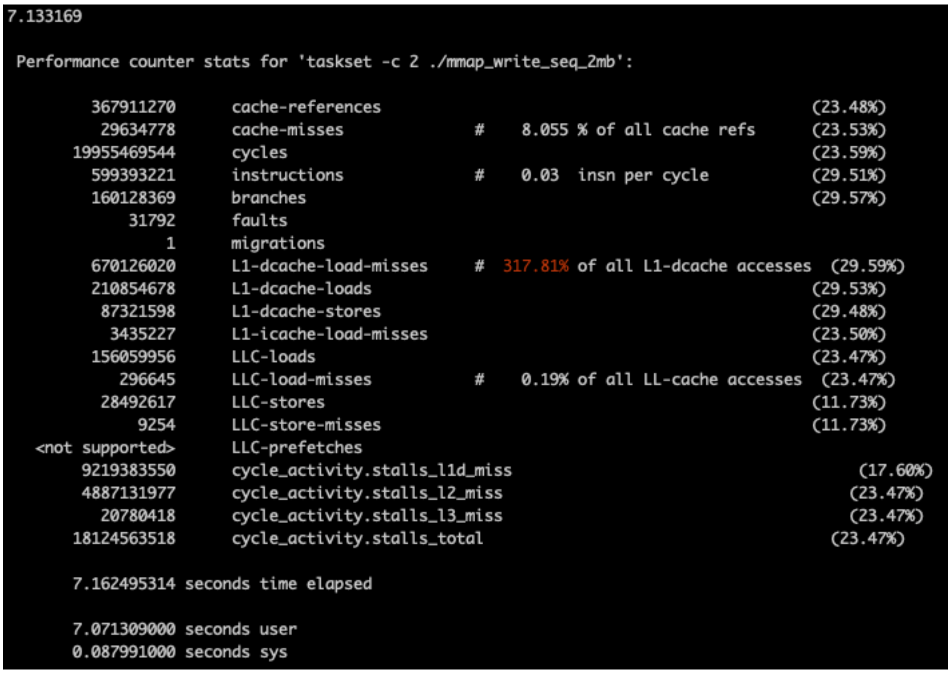
```
6.525489

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_1mb':

        195930096      cache-references                                              (23.55%)
         32870444      cache-misses              #   16.777 % of all cache refs      (23.61%)
      18223909764      cycles                                                        (23.66%)
        529476740      instructions              #    0.03  insn per cycle           (29.53%)
        126819035      branches                                                      (29.53%)
            31535      faults
                1      migrations
        676044650      L1-dcache-load-misses     #  604.10% of all L1-dcache accesses  (29.47%)
        111909316      L1-dcache-loads                                               (29.41%)
         38399752      L1-dcache-stores                                              (29.36%)
          4153566      L1-icache-load-misses                                         (23.49%)
         61538123      LLC-loads                                                     (23.49%)
           238932      LLC-load-misses           #    0.39% of all LL-cache accesses  (23.49%)
         31889961      LLC-stores                                                    (11.74%)
            13383      LLC-store-misses                                              (11.75%)
   <not supported>      LLC-prefetches
       5936264477      cycle_activity.stalls_l1d_miss                                     (17.62%)
       2111179890      cycle_activity.stalls_l2_miss                                     (23.49%)
         19045782      cycle_activity.stalls_l3_miss                                     (23.49%)
      16447293005      cycle_activity.stalls_total                                     (23.49%)

      6.543570829 seconds time elapsed

      6.467334000 seconds user
      0.072037000 seconds sys
```



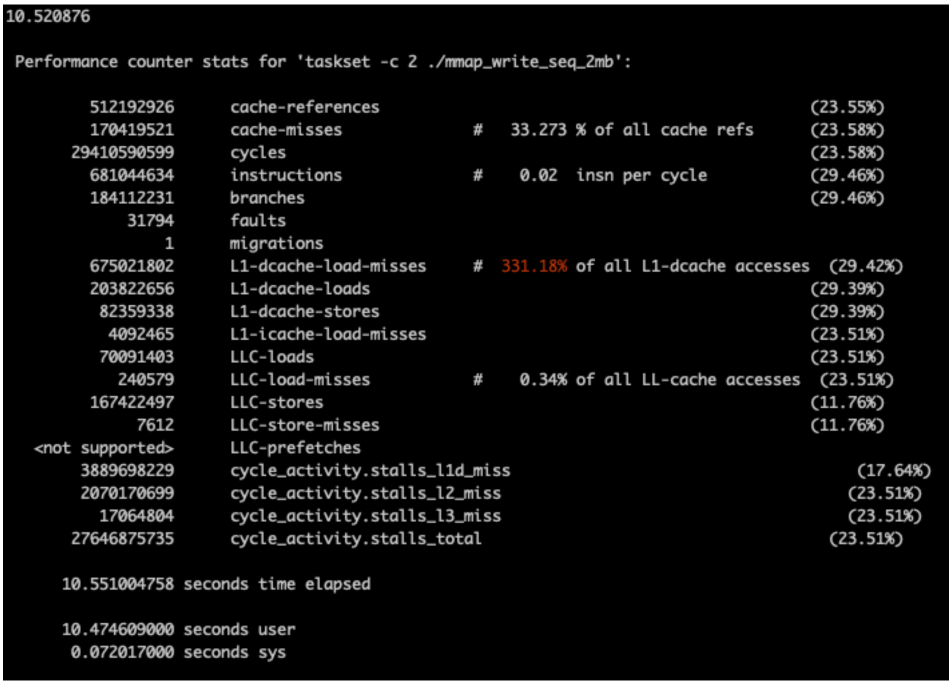
```
10.639645

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_1mb':

        492790744      cache-references                                              (23.47%)
        180090282      cache-misses              #   36.545 % of all cache refs      (23.51%)
      29709815582      cycles                                                        (23.54%)
        584115331      instructions              #    0.02  insn per cycle           (29.44%)
        142141615      branches                                                      (29.48%)
            31536      faults
                1      migrations
        676776687      L1-dcache-load-misses     #  605.07% of all L1-dcache accesses  (29.47%)
        111851277      L1-dcache-loads                                               (29.47%)
         36584558      L1-dcache-stores                                              (29.47%)
          3895522      L1-icache-load-misses                                         (23.58%)
         62339981      LLC-loads                                                     (23.58%)
           238142      LLC-load-misses           #    0.38% of all LL-cache accesses  (23.58%)
        176949462      LLC-stores                                                    (11.76%)
            17968      LLC-store-misses                                              (11.73%)
   <not supported>      LLC-prefetches
       3505554040      cycle_activity.stalls_l1d_miss                                     (17.58%)
       1808797756      cycle_activity.stalls_l2_miss                                     (23.44%)
         16269216      cycle_activity.stalls_l3_miss                                     (23.43%)
      27940695633      cycle_activity.stalls_total                                     (23.43%)

     10.657553717 seconds time elapsed

     10.533405000 seconds user
      0.119970000 seconds sys
```

- 2MB 단위 1GB sequential write

  ▼ perf 결과



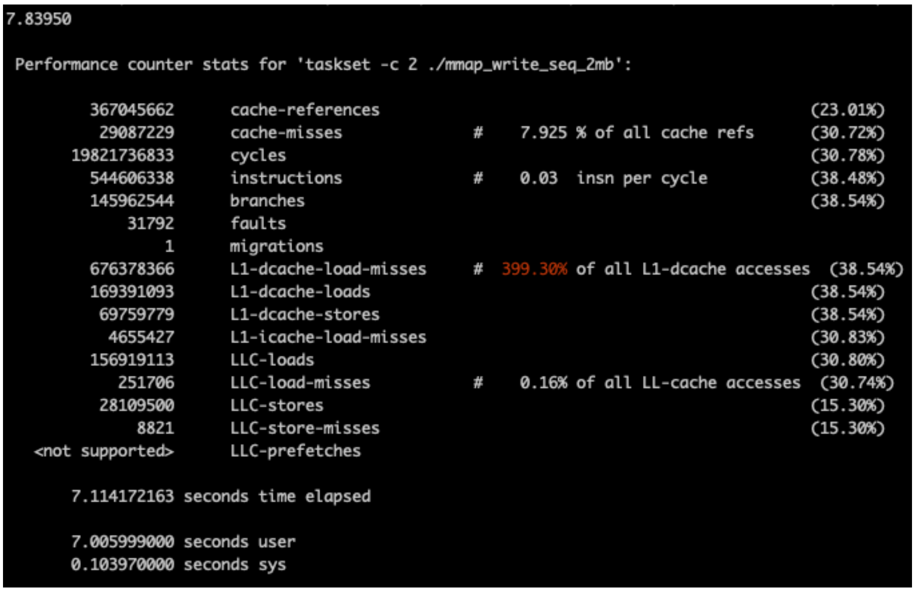
```
7.133169

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_2mb':

        367911270      cache-references                                              (23.48%)
         29634778      cache-misses              #    8.055 % of all cache refs      (23.53%)
      19955469544      cycles                                                        (23.59%)
        599393221      instructions              #    0.03  insn per cycle           (29.51%)
        160128369      branches                                                      (29.57%)
            31792      faults
                1      migrations
        670126020      L1-dcache-load-misses     #  317.81% of all L1-dcache accesses  (29.59%)
        210854678      L1-dcache-loads                                               (29.53%)
         87321598      L1-dcache-stores                                              (29.48%)
          3435227      L1-icache-load-misses                                         (23.50%)
        156059956      LLC-loads                                                     (23.47%)
           296645      LLC-load-misses           #    0.19% of all LL-cache accesses  (23.47%)
         28492617      LLC-stores                                                    (11.73%)
             9254      LLC-store-misses                                              (11.73%)
   <not supported>      LLC-prefetches
       9219383550      cycle_activity.stalls_l1d_miss                                     (17.60%)
       4887131977      cycle_activity.stalls_l2_miss                                     (23.47%)
         20780418      cycle_activity.stalls_l3_miss                                     (23.47%)
      18124563518      cycle_activity.stalls_total                                      (23.47%)

      7.162495314 seconds time elapsed

      7.071309000 seconds user
      0.087991000 seconds sys
```



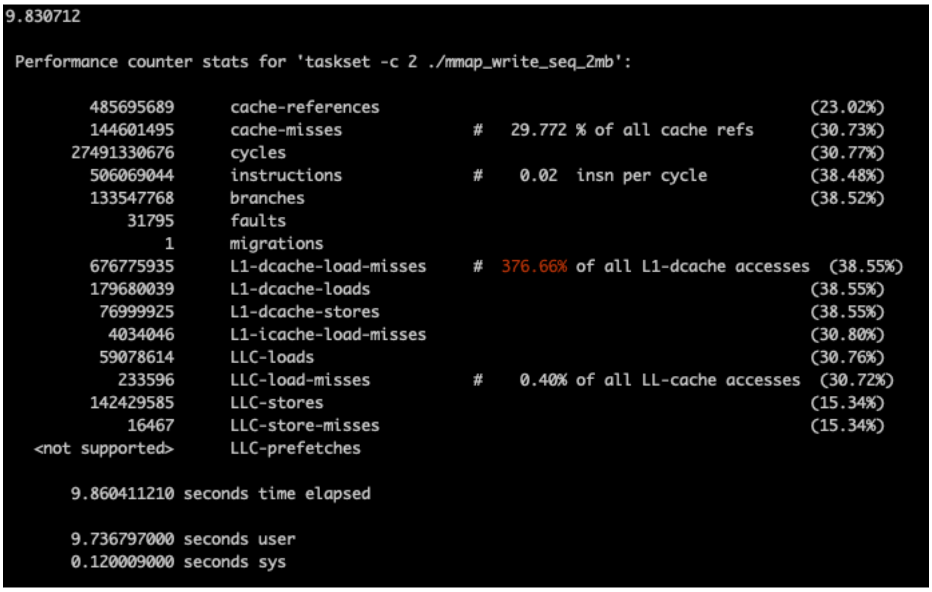
```
10.520876

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_2mb':

        512192926      cache-references                                              (23.55%)
        170419521      cache-misses              #   33.273 % of all cache refs      (23.58%)
      29410590599      cycles                                                        (23.58%)
        681044634      instructions              #    0.02  insn per cycle           (29.46%)
        184112231      branches                                                      (29.46%)
            31794      faults
                1      migrations
        675021802      L1-dcache-load-misses     #  331.18% of all L1-dcache accesses  (29.42%)
        203822656      L1-dcache-loads                                               (29.39%)
         82359338      L1-dcache-stores                                              (29.39%)
          4092465      L1-icache-load-misses                                         (23.51%)
         70091403      LLC-loads                                                     (23.51%)
           240579      LLC-load-misses           #    0.34% of all LL-cache accesses  (23.51%)
        167422497      LLC-stores                                                    (11.76%)
             7612      LLC-store-misses                                              (11.76%)
   <not supported>      LLC-prefetches
       3889698229      cycle_activity.stalls_l1d_miss                                     (17.64%)
       2070170699      cycle_activity.stalls_l2_miss                                     (23.51%)
         17064804      cycle_activity.stalls_l3_miss                                     (23.51%)
      27646875735      cycle_activity.stalls_total                                      (23.51%)

     10.551004758 seconds time elapsed

     10.474609000 seconds user
      0.072017000 seconds sys
```

▼ 문제점

- 동일한 조건으로 실험을 진행했을 때, 결과값이 다른 경우가 존재함 (jbd_2를 동일 노드 특정 코어에 고정시켰을 때도 결과값이 다른 경우가 발생함을 확인)
- 단, 4KB 랜덤 문자열 버퍼로 구성되어 4KB씩 write를 진행하는 워크로드의 경우 4회 실험 전부 동일 결과를 보여줌
- 아래의 두 perf 프로파일링 결과는 동일 조건에서 실행한 2MB 랜덤 문자열 버퍼로 구성된 1GB sequential PM DAX write without flush instruction 워크로드 실험 결과임



```
7.83950

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_2mb':

         367045662      cache-references                                              (23.01%)
          29087229      cache-misses              #    7.925 % of all cache refs      (30.72%)
       19821736833      cycles                                                        (30.78%)
         544606338      instructions              #    0.03  insn per cycle           (38.48%)
         145962544      branches                                                      (38.54%)
             31792      faults
                 1      migrations
         676378366      L1-dcache-load-misses     #  399.30% of all L1-dcache accesses  (38.54%)
         169391093      L1-dcache-loads                                               (38.54%)
          69759779      L1-dcache-stores                                              (38.54%)
           4655427      L1-icache-load-misses                                         (30.83%)
         156919113      LLC-loads                                                     (30.80%)
            251706      LLC-load-misses           #    0.16% of all LL-cache accesses  (30.74%)
          28109500      LLC-stores                                                    (15.30%)
              8821      LLC-store-misses                                              (15.30%)
   <not supported>      LLC-prefetches

       7.114172163 seconds time elapsed

       7.005999000 seconds user
       0.103970000 seconds sys
```



```
9.830712

 Performance counter stats for 'taskset -c 2 ./mmap_write_seq_2mb':

         485695689      cache-references                                              (23.02%)
         144601495      cache-misses              #   29.772 % of all cache refs      (30.73%)
       27491330676      cycles                                                        (30.77%)
         506069044      instructions              #    0.02  insn per cycle           (38.48%)
         133547768      branches                                                      (38.52%)
             31795      faults
                 1      migrations
         676775935      L1-dcache-load-misses     #  376.66% of all L1-dcache accesses  (38.55%)
         179680039      L1-dcache-loads                                               (38.55%)
          76999925      L1-dcache-stores                                              (38.55%)
           4034046      L1-icache-load-misses                                         (30.80%)
          59078614      LLC-loads                                                     (30.76%)
            233596      LLC-load-misses           #    0.40% of all LL-cache accesses  (30.72%)
         142429585      LLC-stores                                                    (15.34%)
             16467      LLC-store-misses                                              (15.34%)
   <not supported>      LLC-prefetches

       9.860411210 seconds time elapsed

       9.736797000 seconds user
       0.120009000 seconds sys
```

PM DAX read 실험(Read이므로 flush instruction은 기본적으로 없음)

▼ Buffer 문자열 설정

- 실험 구성이 4KB, 256KB, 1MB, 2MB 단위로 1GB를 sequential하게 읽어오기 때문에 적어도 2MB보다 큰 랜덤 문자열로 buffer를 구성해주는 것이 좋다.
- 따라서 위의 PM DAX write 실험의 random 문자열 생성 코드를 이용하여 4MB 크기의 buffer에 랜덤 문자열을 생성해주었다.

▼ 파일 생성 코드

- PM DAX write 실험과 다르게 cache line으로의 흡수를 막기 위해 4MB 단위의 랜덤 문자열로 write를 진행하여 1GB 크기의 파일을 생성하도록 하였다.

```
//Generate 1GB size file for mmap_write_seq workload(The block size can be change, and it will match with the block size which
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <errno.h>
```

```
int main(){
  int fd;
  int i;
  char *map;
  int GB_to_KB = 1048576; //1 GB = 1048576 KB
  int KB_to_Byte = 1024; //1 KB = 1024 Bytes

  int buf_size_in_byte = 4096*1024; //2MB
  char buf[4096*1024] = {0, }; //2MB

  int iter_num = (GB_to_KB*KB_to_Byte)/buf_size_in_byte;

  srand((unsigned int)time(NULL));
  int rand_num = 0;
  for(i = 0; i < buf_size_in_byte; i++){
    rand_num = rand()%64;
    buf[i] = '0' + rand_num;
  }
  if((fd = open("/pmem/file1", O_RDWR|O_CREAT,0644)) < 0){
        printf("failed to open file\n");
        return 0;
  }
  printf("Starting 1G file generating1\n");
  struct timeval startTime, endTime;
  double diffTime;
  gettimeofday(&startTime, NULL);
  for(i = 0; i < iter_num; i++){
        write(fd, buf, buf_size_in_byte);
  }
  gettimeofday(&endTime, NULL);
  if(endTime.tv_usec < startTime.tv_usec){
        endTime.tv_usec += 1000000;
        endTime.tv_sec -= 1;
  }
  printf("The end of file generating operation1 : %ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.
  close(fd);
  return 0;
}
```

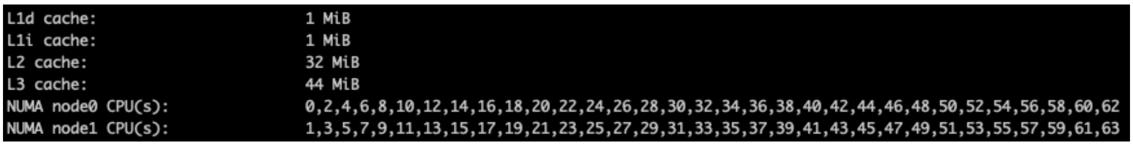
▼ **read 워크로드 코드**

memcpy read 속도가 write에 비해 약 2배 정도 빠른 것을 고려하여 sequential하게 1GB 파일을 총 40번 읽어들여 전체 40GB를 read하는 워크로드를 구상하였다.

```
//Workload for writing to PM as DAX FS mode in sequencial blocks(the amount of sequential write size is restricted due to the limi
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <stdint.h>
#include <immintrin.h>

int main(){
  int fd;
  int i,j,k;
  char *map;
  int n_GB = 40;
  int GB_to_KB = 1048576; //1 GB = 1048576 KB
  int KB_to_Byte = 1024; //1 KB = 1024 Bytes
  int flush_iter = 4096/64;

  int buf_size_in_byte = 4096; //4KB:4096Bytes -> write in bytes
  char buf[4096] = {0, }; //4KB

  int iter_num = (GB_to_KB*KB_to_Byte)/buf_size_in_byte;
  if((fd = open("/pmem/file1", O_RDWR|O_CREAT,0644)) < 0){
    printf("failed to open file\n");
    return 0;
  }

  //map = mmap(NULL, buf_size_in_byte, PROT_READ | PROT_WRITE, MAP_SHARED_VALIDATE|MAP_SYNC, fd, 0);
  map = mmap(NULL, GB_to_KB*KB_to_Byte, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0); //1GB mapping

  //printf("Starting mmap %dGB writing operation1\n", n_GB);
  struct timeval startTime, endTime, syncTime_before, syncTime_after, unmapTime_before, unmapTime_after;
  double diffTime, syncTime, unmapTime;
  gettimeofday(&startTime, NULL);

  for(i=0; i<n_GB; i++){
    for(j=0; j<iter_num; j++){ //1GB sequential write
      char *map_j = map + (buf_size_in_byte*j);
      memcpy(buf, map_j, buf_size_in_byte);
    }
  }

  gettimeofday(&endTime, NULL);
  if(endTime.tv_usec < startTime.tv_usec){
        endTime.tv_usec += 1000000;
        endTime.tv_sec -= 1;
  }
  //printf("The end of mmap write operation1 : %ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.tv_usec)
  printf("%ld.%ld\n", endTime.tv_sec - startTime.tv_sec, endTime.tv_usec - startTime.tv_usec);
  gettimeofday(&syncTime_before, NULL);
  msync(map, buf_size_in_byte, MS_SYNC);
  munmap(map, buf_size_in_byte);
  close(fd);
  return 0;
}
```

▼ **실험 환경**

- Cache size & NUMA information



```
L1d cache:                       1 MiB
L1i cache:                       1 MiB
L2 cache:                        32 MiB
L3 cache:                        44 MiB
NUMA node0 CPU(s):               0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62
NUMA node1 CPU(s):               1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63
```

- PM0로 DAX read 진행
- jbd_2는 노드0의 8번 코어에 배치 (저널링 과정에서의 remote access 방지)
  → read 실험이어서 불필요한 과정이지만, 혹시 모를 요인 배제를 위해 수행
- 코어의 governer는 전부 performance mode로 설정

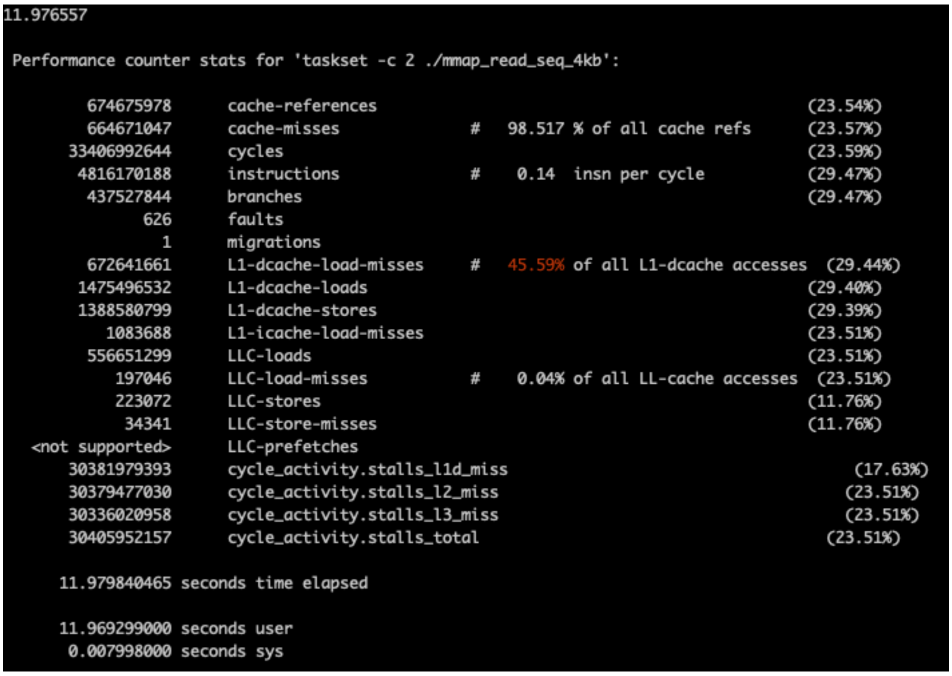
▼ **실험 과정**

PM DAX write 실험과 동일

▼ **실험 결과**

- 4KB 단위 1GB sequential read

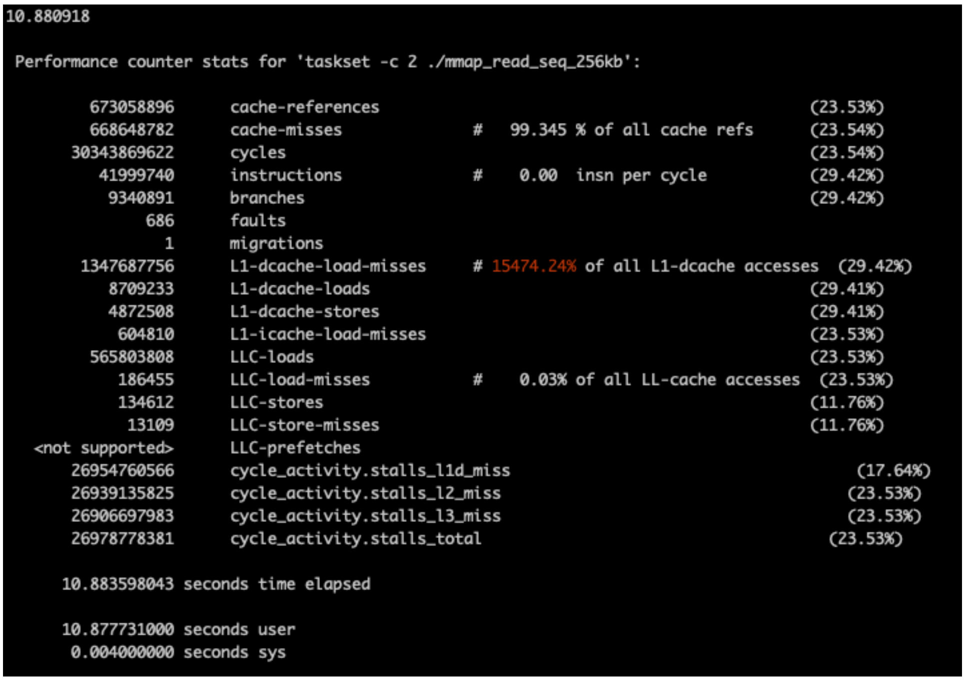
  ▼ perf 결과



```
11.976557

 Performance counter stats for 'taskset -c 2 ./mmap_read_seq_4kb':

        674675978      cache-references                                              (23.54%)
        664671047      cache-misses              #   98.517 % of all cache refs      (23.57%)
      33406992644      cycles                                                        (23.59%)
       4816170188      instructions              #    0.14  insn per cycle           (29.47%)
        437527844      branches                                                      (29.47%)
              626      faults
                1      migrations
        672641661      L1-dcache-load-misses     #   45.59% of all L1-dcache accesses  (29.44%)
       1475496532      L1-dcache-loads                                               (29.40%)
       1388580799      L1-dcache-stores                                              (29.39%)
          1083688      L1-icache-load-misses                                         (23.51%)
        556651299      LLC-loads                                                     (23.51%)
           197046      LLC-load-misses           #    0.04% of all LL-cache accesses  (23.51%)
           223072      LLC-stores                                                    (11.76%)
            34341      LLC-store-misses                                              (11.76%)
   <not supported>      LLC-prefetches
      30381979393      cycle_activity.stalls_l1d_miss                                     (17.63%)
      30379477030      cycle_activity.stalls_l2_miss                                     (23.51%)
      30336020958      cycle_activity.stalls_l3_miss                                     (23.51%)
      30405952157      cycle_activity.stalls_total                                     (23.51%)

     11.979840465 seconds time elapsed

     11.969299000 seconds user
      0.007998000 seconds sys
```

- 256KB 단위 1GB sequential read

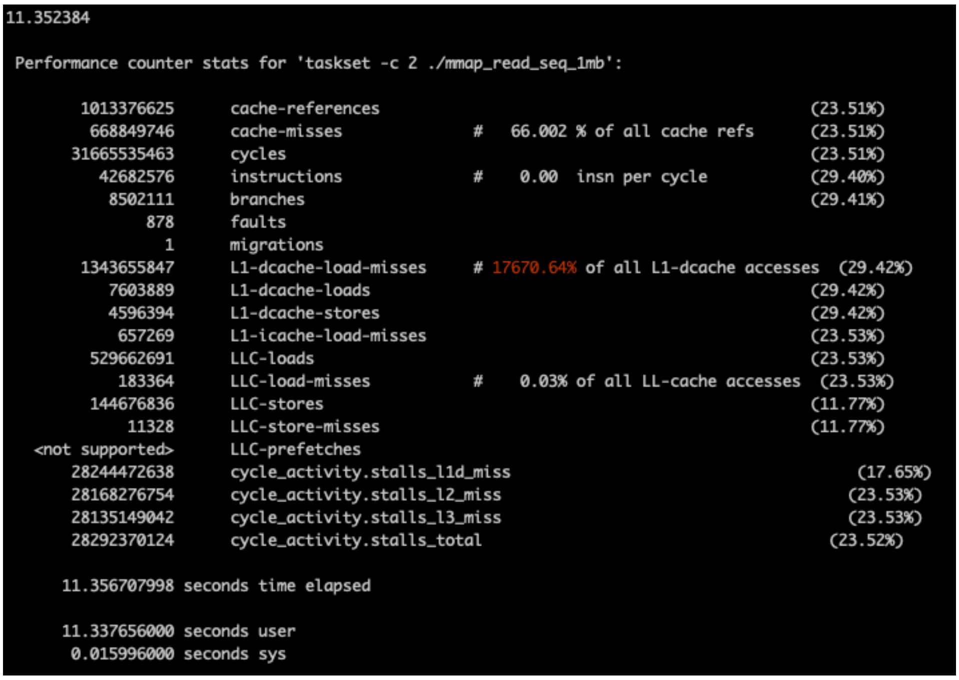
    ▼ perf 결과



```
10.880918

 Performance counter stats for 'taskset -c 2 ./mmap_read_seq_256kb':

        673058896      cache-references                                              (23.53%)
        668648782      cache-misses              #   99.345 % of all cache refs      (23.54%)
      30343869622      cycles                                                        (23.54%)
         41999740      instructions              #    0.00  insn per cycle           (29.42%)
          9340891      branches                                                      (29.42%)
              686      faults
                1      migrations
       1347687756      L1-dcache-load-misses     # 15474.24% of all L1-dcache accesses  (29.42%)
          8709233      L1-dcache-loads                                               (29.41%)
          4872508      L1-dcache-stores                                              (29.41%)
           604810      L1-icache-load-misses                                         (23.53%)
        565803808      LLC-loads                                                     (23.53%)
           186455      LLC-load-misses           #    0.03% of all LL-cache accesses  (23.53%)
           134612      LLC-stores                                                    (11.76%)
            13109      LLC-store-misses                                              (11.76%)
   <not supported>      LLC-prefetches
      26954760566      cycle_activity.stalls_l1d_miss                                     (17.64%)
      26939135825      cycle_activity.stalls_l2_miss                                     (23.53%)
      26906697983      cycle_activity.stalls_l3_miss                                     (23.53%)
      26978778381      cycle_activity.stalls_total                                     (23.53%)

     10.883598043 seconds time elapsed

     10.877731000 seconds user
      0.004000000 seconds sys
```

- 1MB 단위 1GB sequential read

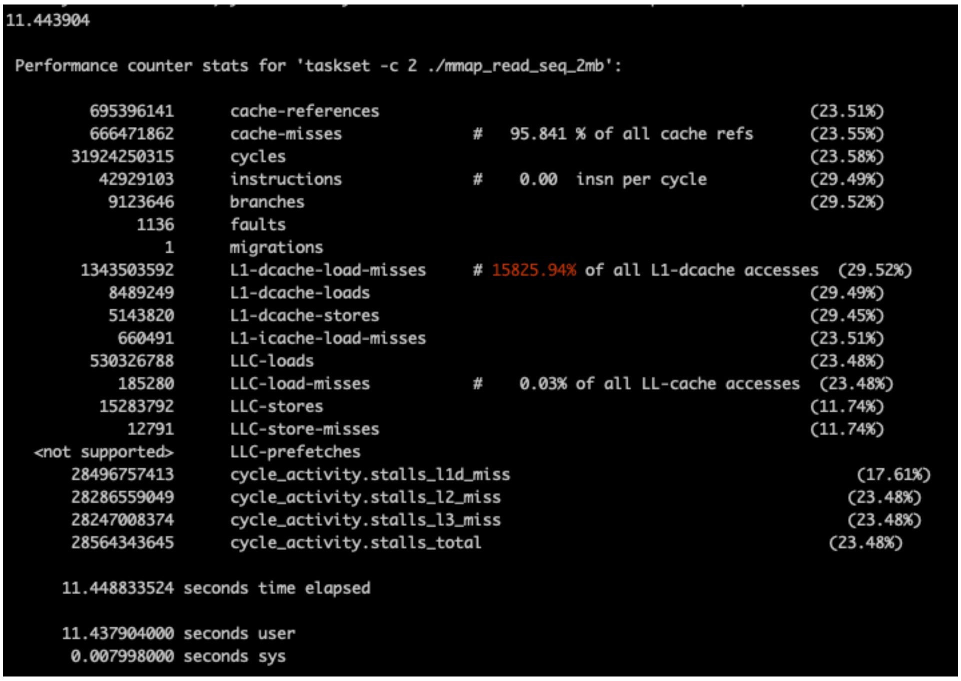
    ▼ perf 결과



```
11.352384

 Performance counter stats for 'taskset -c 2 ./mmap_read_seq_1mb':

        1013376625      cache-references                                              (23.51%)
         668849746      cache-misses              #   66.002 % of all cache refs      (23.51%)
       31665535463      cycles                                                        (23.51%)
          42682576      instructions              #    0.00  insn per cycle           (29.40%)
           8502111      branches                                                      (29.41%)
               878      faults
                 1      migrations
        1343655847      L1-dcache-load-misses     # 17670.64% of all L1-dcache accesses  (29.42%)
           7603889      L1-dcache-loads                                               (29.42%)
           4596394      L1-dcache-stores                                              (29.42%)
            657269      L1-icache-load-misses                                         (23.53%)
         529662691      LLC-loads                                                     (23.53%)
            183364      LLC-load-misses           #    0.03% of all LL-cache accesses  (23.53%)
         144676836      LLC-stores                                                    (11.77%)
             11328      LLC-store-misses                                              (11.77%)
   <not supported>      LLC-prefetches
       28244472638      cycle_activity.stalls_l1d_miss                                     (17.65%)
       28168276754      cycle_activity.stalls_l2_miss                                     (23.53%)
       28135149042      cycle_activity.stalls_l3_miss                                     (23.53%)
       28292370124      cycle_activity.stalls_total                                      (23.52%)

      11.356707998 seconds time elapsed

      11.337656000 seconds user
       0.015996000 seconds sys
```

- 2MB 단위 1GB sequential read

  ▼ perf 결과



```
11.443904

 Performance counter stats for 'taskset -c 2 ./mmap_read_seq_2mb':

         695396141      cache-references                                              (23.51%)
         666471862      cache-misses              #   95.841 % of all cache refs      (23.55%)
       31924250315      cycles                                                        (23.58%)
          42929103      instructions              #    0.00  insn per cycle           (29.49%)
           9123646      branches                                                      (29.52%)
              1136      faults
                 1      migrations
        1343503592      L1-dcache-load-misses     # 15825.94% of all L1-dcache accesses  (29.52%)
           8489249      L1-dcache-loads                                               (29.49%)
           5143820      L1-dcache-stores                                              (29.45%)
            660491      L1-icache-load-misses                                         (23.51%)
         530326788      LLC-loads                                                     (23.48%)
            185280      LLC-load-misses           #    0.03% of all LL-cache accesses  (23.48%)
          15283792      LLC-stores                                                    (11.74%)
             12791      LLC-store-misses                                              (11.74%)
   <not supported>      LLC-prefetches
       28496757413      cycle_activity.stalls_l1d_miss                                     (17.61%)
       28286559049      cycle_activity.stalls_l2_miss                                     (23.48%)
       28247008374      cycle_activity.stalls_l3_miss                                     (23.48%)
       28564343645      cycle_activity.stalls_total                                      (23.48%)

      11.448833524 seconds time elapsed

      11.437904000 seconds user
       0.007998000 seconds sys
```

다른 방향성

turbo mode

- 각 코어별 전력은 제한되어있음 → 만약 PM으로의 I/O가 전력을 덜 소비하면 sibling core에서 돌아가는 task가 더 많은 전력을 소비해서 더 좋은 성능을 낼 수 있는 것 아닐까? → 온도는 HW의 온도 센서가 측정하지만, 전력은 Idle cycle(?)의 비율에 따라 책정되는

데 PM으로의 I/O가 과연 어떤지는 모르겠음

→ 스케줄링으로 연결됨

Noisy neighbor

- 우선 clflush와 다르게 clflushopt와 clwb이 sibling core의 CPU task의 성능을 상당히 저하시키는 현상의 원인은 clflushopt와 clwb이 bus mastering을 하기 때문일수도?(clflushopt와 clwb이 bus mastering을 하는게 확실한 것은 아님. 그냥 가정일 뿐. 확인 필요)
- 지금까지 발견한 사실은 현재 PM DAX I/O stack 상 msync의 마지막 instruction이 clwb인데, 이 clwb이 I/O의 성능은 높일지 몰라도 noisy neighbor가 될 수 있음 (clflush와 다르게 sibling core에 있는 CPU task의 성능을 심각하게 저하시키는 것은 확인된 사실이며, LLC를 공유하는 task들의 성능 또한 저하시킬 여지가 있음 → LLC 공유 관련은 실험 필요)
- 반면, clflush은 sibling core의 CPU task의 성능을 대체로 향상시키지만 I/O task 본인의 성능은 clwb에 비해 상당히 감소됨
- 따라서 우리는 현재 PM I/O stack이 noisy neighbor가 될 수 있음을 발견하였기에 clflush와 clwb 사이의 절충안을 제시하려함

→ Noisy neighbor이므로 sibling core나 LLC를 고려한 스케줄링으로 연결됨
→ clflush와 clwb 사이의 절충안, 즉 flush 및 cache 관련쪽으로 연결됨