Performance Evaluation of Solid-State Disks through Black-Box Approach

Junkil Ryu† Chanik Park‡

Department of Computer Science & Engineering, POSTECH
{lancer† , cipark‡ }@postech.ac.kr

Abstract Due to recent technology advancement in NAND flash memory, SSDs (Solid-State Disks) are replacing the existing HDDs (Hard Disk Drives) in computer systems. However, SSDs have different performance characteristics from HDDs. For example, SSD has higher random I/O performance due to NAND flash memory chips used as underlying storage media, and the performance of an SSD is highly dependent on its SSD controller firmware whereas the performance of an HDD is not due to the underlying mechanical components in the HDD. In order to optimally configure OS policies in buffer cache and prefetching for SSDs, we need to know their performance characteristics. In this paper, we propose a black-box approach to retrieve an important parameter of an SSD called SSD management block. We have shown that our proposed approach is useful to analyze the performance characteristics of commercially available SSD products.

Key words : Solid-State Disk (SSD), Black-box approach

1. 서 론

SSD (Solid-State Disk)는 여러 개의 NAND 플래시 메모리 칩들을 이용하여 구현된 대용량 저장 매체로, 빠른 데이터 처리가 가능하고, 전력소모, 발열, 소음 등이 낮아 빠르게 모바일 시장에 적용되고 있다. SSD는 그림 1처럼 호스트 인터페이스, 프로세서, SRAM, DRAM, 플래시 메모리 컨트롤러, 및 NAND 플래시 메모리 칩들로 구성된다. 특히 플래시 메모리 컨트롤러는 다중 채널에 있는 여러 개의 NAND 플래시 메모리 칩들을 병렬적으로 사용할 수 있도록 해준다. 따라서 SSD의 경우, HDD와 달리 동일한 하드웨어 설계에 기반을 두더라도 소프트웨어 설계에 따라서 성능 특성이 확연히 달라 질 수 있다.

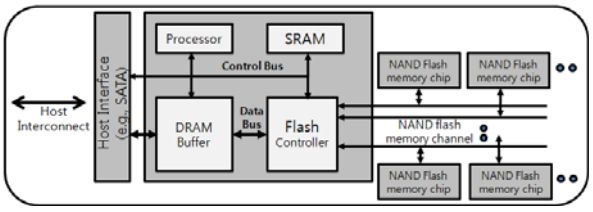


그림 1 SSD 하드웨어 설계

그림 2 SSD A의 성능 측정 (iozone benchmark)

그림 2와 3은 시중에 판매 되고 있는 SSD A와 SSD C의 성능을 구입 초기에 측정하고(start_XX), 3개월 후에 동일한 성능 측정 실험(3month_XX)을 한 것을 보여주는 것이다. 그림 2와 3에서 표기는 start와 3month은 실험시기를, 뒤의 RR에서 첫 번째 R은 I/O 패턴으로 S이면 순차, R이면 임의이고 두 번째 R은 I/O 종류를 표현하는 것으로 R이면 읽기, W이면 쓰기이다. 구입 초기 성능 측정 실험을 보게 되면 SSD A의 순차 쓰기와 임의 쓰기간의 성능패턴에 비해서 SSD C의 순차 쓰기와 임의 쓰기간의 성능패턴이 작은 명령어 크기부터 확연히 다른 것을 알 수 있다. 또한 SSD A와 C의 3 개월 후의 성능은 전체적으로 초기 성능에 비해서 저하되는 것을 알 수 있다. 특이한 것은 SSD C의 경우, SSD A와 달리 순차 읽기와 임의 읽기의 성능이 3개월 후에 확연히 저하된다는 것이다. 참고로 SSD A와 SSD C의 절대적인 성능 차이는 플래시 메모리 채널의 수에서 발생한다.

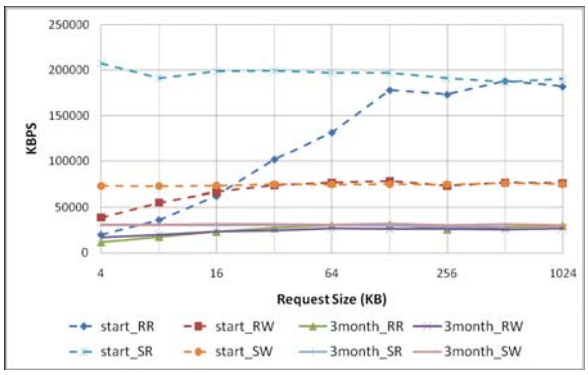


그림 3 SSD C의 성능 측정 (iozone benchmark)

이러한 SSD들 간의 서로 다른 성능특성을 이해하기 위해서, 본 논문에서는 내장 소프트웨어 설계를 모르는 상황에서 간단한 성능 테스트를 통해서 SSD의 소프트웨어 설계 및 성능 특성을 이해할 수 있는 방법을 제시하고자 한다.

2. 본 론

SSD의 저장 매체는 NAND 플래시 메모리로서 out-place update만 지원하고, 기존 HDD의 호스트 인터페이스 (ATA, SATA 등)에서 적용되는 디스크 I/O 블록의 크기 (512 B)와 NAND 플래시 메모리의 내부 페이지 크기 (2 KB 또는 4KB)가 다르다 [1, 2]. 따라서 고성능 SSD를 위해서는 여러 개의 페이지들로 구성된 SSD 관리 블록을 일반적으로 정의하는 데, 성능을 위해서는 다음과 같은 고려들을 해야 한다[1-3].

- NAND 플래시 메모리의 out-place update로 인한 쓰기 성능 저하 문제를 최소화시켜야 함
- 페이지 단위로 맵핑 테이블을 지원하게 되면 맵핑 테이블의 크기가 커지고 요구 블록 검색에 많은 시간을 필요로 하기 때문에 성능 저하를 초래할 수 있음
- SSD 내부의 다중 플래시 메모리 채널 상에 있는 NAND 플래시 메모리 칩들을 병렬적으로 이용할 수 있도록 해야 함.

외부 I/Os를 통해서 SSD의 성능 특성을 알아보기 위해서, SSD 관

리블록의 크기와 SSD 관리 블록 배치가 중요하다. 따라서 본론에서는 이를 알 수 있는 간단한 기법을 제안한다. 제안 기법 도출과 도출된 기법의 검증을 위해서 시중에 판매되고 있는 SSD A, B, C를 이용하였다. SSD A, B, C의 하드웨어 상세는 그림 4와 같다.

	SSD A	SSD B	SSD C
NAND flash memory type	SLC	SLC	MLC
Memory buffer size	32 MB	16 MB	16 MB
Physical page size	4KB	2KB	4KB
Number of channels	4	4	10
Number of flash memory chips	8	16	20

그림 4 각 SSD의 하드웨어 명세

SSD 관리 블록의 크기: SSD 관리 블록이 요구 명령의 크기보다 크게 되면 데이터 복제 오버헤드가 발생한다. 그림 5는 데이터 복제 오버헤드가 발생하는 형태를 보여주고 있다. 그림 5의 (1)은 갱신되는 데이터의 크기가 하나의 관리 블록보다 작고, 하나의 관리 블록 안에서 갱신이 이루어지는 것을 보여준다. 그림 5의 (2)는 갱신되는 데이터 크기가 하나의 관리 블록 보다 작지만 두 개의 관리블록들에 걸쳐있는 경우이다. 이러한 경우에는 (1) 경우보다 2 배 이상의 데이터 복제 오버헤드가 발생한다. 이러한 특성을 이용하여 SSD 내부의 관리 블록의 크기를 추정할 수 있다.

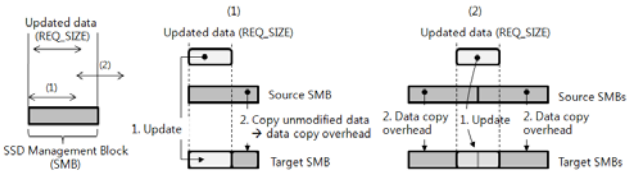


그림 5 SSD에서 데이터 갱신 시 발생하는 오버헤드

SSD들의 관리 블록 크기를 추정하기 위해서 그림 6의 pseudo code를 이용하여 간단한 프로그램을 작성하였다. 일정한 크기의 쓰기 명령을 일정 크기씩 이동해가면서 수행함으로써, 그림 5 (1)과 (2)의 오버헤드 차이를 이용하여 SSD 관리 블록의 크기를 추정한다. 그 pseudo code에서 Chunk는 SSD 내부 버퍼 및 로그영역으로 인한 실험 오차를 줄이기 위해서 SSD를 여러 개로 구분한 것으로 SSD에 포함된 Chunk의 수는 최대 (SSD Capacity)/(chunk size)이다. 실험 오차를 줄이기 위해서 chunk size는 크게 설정하고 여러 chunk에서 반복적으로 실험하도록 하였다. Pseudo code는 T_{max}, T_{min}, T_{avg}을 구하는 부분과 SSD 관리 블록 크기를 추출하는 부분으로 나누어진다. 그 함수에 있는 $T_{mon}(i, k)$은 k번째 chunk에서 i × shift size만큼 이동한 위치에 쓰기 명령을 내렸을 때 측정되는 반응 시간을 나타내고, T_i는 각 chunk에서 i × shift size만큼 이동한 위치에 명령을 내렸을 때의 반응시간으로 $Chunk_{num}$개의 Chunks에서 테스트한 결과의 평균이다. T_{max}, T_{min}, T_{avg}는 chunk에서 이동 위치에 상관없이 쓰기 명령의 반응시간 최대값, 최소값, 평균값을 의미한다. SSD 관리 블록의 크기는 T_{max}, T_{min}, T_{avg}의 관계를 이용하여 추출하는데, $T_{max} - T_{avg}$가 $T_{avg} - T_{min}$보다 크다는 것은 쓰기 명령이 대체적으로 ⌈ write request size / SSD management block size ⌉ 개의 SSD 관리 블록들에 맞아 떨어지다가, ⌈ write request size / SSD management block size ⌉ +1개의 SSD 관리 블록들에 걸치는 것을 의미한다. 이러한 패턴을 보이는 i 들을 모두 모아놓고 그것의 최대공약수를 구하게 되면 SSD 관리 블록 크기를 구할 수 있다. $Chunk_{num}$은 실험에 사용한 chunk 수로서 Garbage collection 등의 영향을 피하기 위해서 SSD에 있는 chunk 수의 반만을 이용한다.

테스트 프로그램을 이용하여 추정하는 SSD A, B, C의 관리 블록 크기는 16 KB, 1 MB, 4 KB이다. 그림 7은 각 이동 위치에서 쓰기 명령에 따른 반응시간을 보여주고 있다. 4 KB 쓰기 명령의 경우 매 16 KB 마다 반응시간이 커지는 것이 보이며, 16 KB 및 32 KB 쓰기 명령의 경우에는 매 16 KB마다 반응 시간이 떨어지는 것으로 나타난다. 4 KB 쓰기 명령 패턴에서 반응 시간이 올라가는 것은 두 개의 관리 블록들에 쓰기 명령이 걸치는 경우로 보이며, 16 KB 및 32 KB 쓰기 명령에서 반응 시간이 내려오는 패턴을 보이는 것은 관리 블록(들)의 경계에 맞는 경우에 데이터 복제 오버헤드가 작기 때문이다. 다른 SSD 들의 경우에도 테스트 프로그램이 예측한 크기에 일치하는 패턴을 보여준다.

SSD 관리 블록의 다중 플래시 채널 상의 배치: SSD 관리 블록 배치는 그림 8과 같이 디스크 I/O 블록들을 SSD 관리 블록에 할당하는 단계와 SSD 관리 블록을 다중 플래시 메모리 채널 상의 플래시 페이지들에 할당하는 단계로 구성되어진다. 파일 시스템에서 4401XX에 512 B를 쓰는 명령이 SSD에 내려오면 SSD 내장 소프트웨어는 해당 디스크 I/O 블록이 속한 SSD 관리 블록과 새로 사용되어질 SSD 관리 블록을 할당해야 한다. SSD 관리 블록이 할당되어지면 SSD 관리 블록에 속한 플래시 메모리 페이지를 할당한다. 이러한 관계를 나누어서 설명하면 다음과 같다.

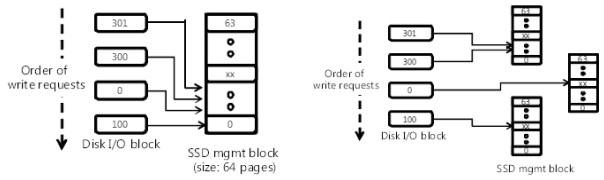
input : write request size, shift_size, $Chunk_{num}$ chunks
output: SSD management block size
begin
  // Obtain $T_{max}, T_{avg}, T_{min}$
  $T_{max} \leftarrow 0$; $T_{avg} \leftarrow 0$; $T_{min} \leftarrow \infty$; $T_{sum} \leftarrow 0$; $T_{total} \leftarrow 0$;
  MinPB $\leftarrow \emptyset$; MaxPB $\leftarrow \emptyset$;
  for $i \leftarrow 0$; $i < \frac{chunk\_size}{shift\_size}$; i++ **do**
    for $k \leftarrow 0$; $k < Chunk_{num}$; k++ **do**
      $T_{sum} \leftarrow T_{sum} + T_{mon}(i, k)$;
      //$T_{mon}(i, k)$ is response time of a write request
      //at i × shift_size position in k-th chunk
    end
    $T_i \leftarrow \frac{1}{Chunk_{num}} \times T_{sum}$; $T_{total} \leftarrow T_{total} + T_i$;
    if $T_i < T_{min}$ **then**
      $T_{min} \leftarrow T_i$;
    else if $T_i > T_{max}$ **then**
      $T_{max} \leftarrow T_i$;
    end
  end
  $T_{avg} \leftarrow \frac{shift\_size}{chunk\_size} \times T_{total}$;
  //Detect the size of SSD management block
  for $i \leftarrow 0$; $i < \frac{chunk\_size}{shift\_size}$; i++ **do**
    for $k \leftarrow 0$; $k < Chunk_{num}$; k++ **do**
      $T_{sum} \leftarrow T_{sum} + T_{mon}(i, k)$;
    end
    $T_i \leftarrow \frac{1}{Chunk_{num}} \times T_{sum}$;
    if $T_{max} - T_{avg} > T_{avg} - T_{min}$ *and* $T_i > \frac{1}{2} \times (T_{avg} + T_{max})$ **then**
      MaxPB $\leftarrow$ MaxPB $\cup$ i;
    else if $T_{max} - T_{avg} < T_{avg} - T_{min}$ *and* $T_i < \frac{1}{2} \times (T_{avg} + T_{min})$ **then**
      MinPB $\leftarrow$ MinPB $\cup$ i;
    end
  end
  if $MinPB \neq \emptyset$ **then**
    SSD management block size $\leftarrow$ shift_size × gcd ($\forall$ x $\in$ MinPB);
  else
    SSD management block size $\leftarrow$ shift_size × gcd ($\forall$ x $\in$ MaxPB);
  end
end

그림 6 SSD 관리 블록 크기를 추정하는 pseudo code

그림 7 SSD A의 shift write

[From 디스크 I/O 블록 To SSD 관리 블록] 그림 9는 디스크 I/O 블록이 SSD 관리 블록으로 할당되어 지는 방법을 보여준다. 디스크 I/O 블록을 SSD 관리 블록으로의 할당은 쓰기 명령에 의해서 이루어지는 것으로 할당 알고리즘은 크게 order-based allocation 알고리즘과 address-based allocation 알고리즘으로 구분되어진다. Order-based allocation 알고리즘은 디스크 I/O 블록 주소와 관계없이 쓰기 명령 순서에 따라 SSD 관리 블록에 할당한다. 그림 9 (a)처럼 디스크 I/O 블록의 주소가 100, 0, 300, 301 순서로 쓰기 명령이

내려오게 되면 크기가 허용하는 한 하나의 SSD 관리 블록에 순차적으로 할당되어진다. 이 알고리즘을 사용하게 되면 쓰기 패턴에 관계없이 쓰기 성능을 최대화 할 수 있지만 쓰기 패턴에 따라 데이터 배치가 정해지기 때문에 읽기 성능은 나빠질 수 있다.



(a) Order-based allocation according to write requests' order (b) Address-based allocation according to write requests' address

그림 8 디스크 I/O 블록의 SSD 관리 블록으로 맵핑

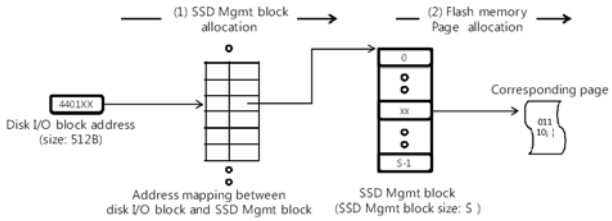


그림 9 디스크 I/O 블록, SSD 관리 블록, 플래시 메모리 페이지의 관계

Address-based allocation 알고리즘은 그림 9 (b)와 같이 디스크 I/O 블록 주소에 따라 SSD 관리 블록을 할당한다. 이 알고리즘을 적용하게 되면 임의 쓰기의 성능은 나쁘게 되지만 쓰기패턴에 관계없이 데이터 배치는 일정하게 되므로 읽기 성능은 order-based allocation 알고리즘보다 대체적으로 낮게 된다.

SSD에 적용된 알고리즘을 확인하기위한 방법은 다음과 같다. (1) 플래시 메모리 페이지 크기 단위로 임의 쓰기를 한다. 해당 쓰기 동작은 디스크 I/O 블록 주소 0부터 SSD 용량의 1/2정도(오차를 없앨 정도의 크기면 됨)에 해당하는 주소까지 범위에 대해서 수행한다. (2) 해당 범위에서 4 x 플래시 메모리 페이지 크기를 이용하여 순차 읽기 성능을 측정한다 (T1). (3) 해당 범위에서 플래시 메모리 페이지 크기를 이용하여 순차쓰기를 한다. (4) 해당 범위에서 4 x 플래시 메모리 페이지를 이용한 순차 읽기 성능을 측정한다 (T2). (5) T1과 T2의 성능 비교를 한다. 성능 격차가 크게되면 order-based allocation 알고리즘이며 그렇지 않으면 address-based allocation 알고리즘이다.

Iometer benchmark tool[4]을 이용하여 각 SSD에 대해서 T1과 T2를 비교해보면 SSD A와 B는 거의 성능 차이가 없지만, SSD C의 경우 15%의 차이를 보였다. 그렇다고 SSD C가 여기에서 말하는 order-based allocation 알고리즘을 사용한다고 말할 수는 없다. SSD C의 관리 블록의 크기가 4 KB이기 때문이다. 그러나 이를 통해서 SSD C가 order-based allocation 알고리즘과 같은 방식으로 SSD 관리 블록 (SSD C의 경우 플래시 메모리 페이지에 해당)을 4 KB 쓰기를 할당해주는 것을 알 수 있고, 이를 통해서 order-based allocation 알고리즘이 적용될 경우 성능 특성을 살펴볼 수 있다.

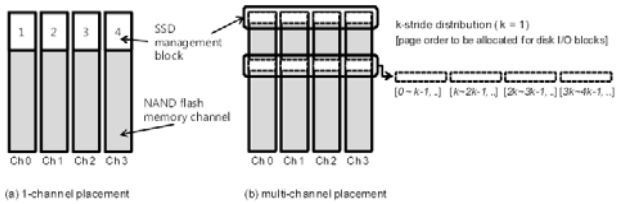


그림 10 SSD 관리 블록의 플래시 메모리 상의 배치

[From SSD 관리 블록 To 플래시 메모리 페이지] 그림 10은 SSD 관리 블록을 물리적인 플래시 메모리 채널 상의 플래시 페이지들에 배치하는 방법을 보여준다. 그림 10의 (a) 1-channel placement 방법은 SSD 관리 블록을 하나의 채널 상에 존재하는 플래시 메모리 칩 내에 배치하는 것이다. 그림 10의 (b) multi-channel placement 방법은 SSD 관리 블록 구성을 위해서 여러 채널상의 플래시 메모리 칩들을 이용하여 구성한다. 여러 개의 채널 상에 있는 플래시 메모리 칩들에 있는 플래시 페이지들을 이용할 경우, SSD 관리 블록을 구성하는 페이지 순서를 정하는 방법은 그림 10처럼 라운드 로빈 방식으로 각 채널에서 k개의 플래시 페이지들을 할당한다. 성능에 중요한 것은 k의 값을 정하는 것으로 k값이 크면 요구 명령의 서비스 시간이 길어지게 된다.

$$RT_{(a)} = \lfloor \frac{S_{read}}{N_{CH} \times S_{SMB}} \rfloor \times (\frac{S_{SMB}}{S_{page}} \times T_{page} + (N_{CH} - 1) \times \epsilon') + A + \alpha \quad (1)$$

$$S'_{read(a)} \equiv S_{read} - \lfloor \frac{S_{read}}{N_{CH} \times S_{SMB}} \rfloor \times (N_{CH} \times S_{SMB})$$

$$A = \begin{cases} \frac{S'_{read(a)}}{S_{page}} \times T_{page} & \text{if } S'_{read(a)} \le S_{SMB}, \\ \frac{S_{SMB}}{S_{page}} \times T_{page} + \lfloor \frac{S'_{read(a)}}{S_{SMB}} - 1 \rfloor \times \epsilon' & \text{if } S'_{read(a)} > S_{SMB} \end{cases}$$

$$RT_{(b)} = \lfloor \frac{S_{read}}{N_{CH} \times S_{page} \times k} \rfloor \times (\frac{S_{page} \times k}{S_{page}} \times T_{page} + (N_{CH} - 1) \times \epsilon) + B + \alpha \quad (2)$$

$$S'_{read(b)} \equiv S_{read} - \lfloor \frac{S_{read}}{N_{CH} \times S_{page} \times k} \rfloor \times (N_{CH} \times S_{page} \times k)$$

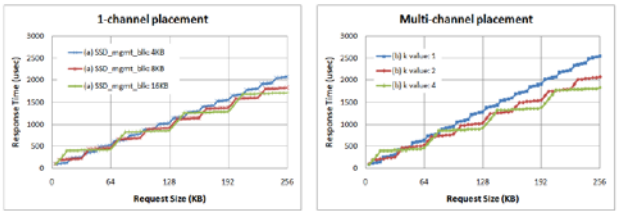
$$B = \begin{cases} \frac{S'_{read(b)}}{S_{page}} \times T_{page} & \text{if } S'_{read(b)} \le S_{page} \times k, \\ \frac{S_{page} \times k}{S_{page}} \times T_{page} + \lfloor \frac{S'_{read}}{S_{page} \times k} - 1 \rfloor \times \epsilon & \text{if } S'_{read(b)} > S_{page} \times k \end{cases}$$

그림 11 SSD 관리 블록을 플래시 메모리 페이지들에 배치방법을 위한 수식들

SSD 관리 블록을 플래시 메모리 페이지들에 배치하는 정보를 검출하기 위해서 배치 방법에 따른 성능 특성 그래프를 수식을 통해서 구하고, 실제 실험을 통해서 얻은 성능 특성 그래프와 비교하여 배치 정보를 검출한다. 그림 11에 있는 수식 (1, 2) 는 다음과 같은 특성을 이용하여 고안되어졌다. 여러 개의 채널을 이용하는 경우, 요구 명령을 각 채널의 플래시 메모리 칩에 동시에 보내고 각 채널에 있는 데이터를 동시에 전달 받을 수 없기 때문에 그림 10 (b)의 경우, 데이터 읽기 시 동기화 오버헤드가 나타날 수 있다. 플래시 메모리 칩의 페이지 당 읽기 성능이 프로그래밍 성능보다 훨씬 좋기 때문에 읽기 성능에서 동기화 관련 오버헤드는 확연히 드러날 수 있지만 쓰기의 경우 버퍼링으로 인해서 외부에 동기화 오버헤드가 드러나지는 않을 것이다. 수식(1, 2)을 위해서 다음과 같은 표식들을 사용하였다. SSD 관리블록 크기는 S_{SMB}, 채널의 수는 N_{CH}, 읽기 요구 명령의 크기는 S_{read}, 플래시 메모리의 기본 페이지 크기는 S_{page}, 플래시 메모리 페이지 칩에서 하나의 페이지를SSD 내부의 DRAM 버퍼로 전송하는 시간은 T_{page}, 각 채널에 있는 데이터를 이동시키는 데 요구되는 동기화 오버헤드는 $\varepsilon(\varepsilon')$로 표현하였다. 디자인 (b)의 경우 모든 채널을 이용하여 SSD 관리 블록을 구성하고 있기 때문에 디자인 (a)의 동기화 오버헤드 (ε')이 디자인 (b)의 동기화 오버헤드 (ε)보다 작을 것이다. 그림 10에서 디자인 (a)의 읽기 요구명령에 따른 반응시간은 수식 (1)로, 디자인 (b)의 읽기 요구 명령에 따른 반응시간은 수식 (2)로 표현된다. 수식에서 α는 읽기 명령을 수행하기 위해서 요구되는 기본적인 시스템 오버헤드이다. 예를 들면 디스크 I/O 블록을 SSD 관리 블록으로, SSD 관리 블록을 물리적인 페이지 주소로 전환하는 데 소요되는 오버헤드도 여기에 포함된다. 수식 (2)에서 k는 그림 10의 (b)의 k를 말한다.

그림 12는 수식을 이용하여 얻은 결과를 그래프로 보여준 것이다. 이 수식 결과를 위해서, SSD가 4개의 채널을 사용하고 있고 각 플래시 메모리의 페이지 크기는 4 KB, 하나의 페이지를 읽는 데 소요되는 시간을 100 µsec (SLC NAND 플래시 메모리로 가정) , 그림 10 (a)의 동기화 오버헤드 (ε', 10 µsec)을 그림 10 (b)의 동기화 오버헤드 (ε, 20 µsec)의 반으로 설정하였다. 성능 비교를 간단하게 보여주기 위해서 시스템 오버헤드 α의 값은 0으로 하였다. 1-channel placement에서 SSD 관리 블록의 크기가 플래시 페이지보다 클 경우, multi-channel placement에서 k > 1인 경우 계단형 패턴이 뚜렷하게 보인다. SSD 관리 블록의 크기가 플래시 페이지와 같거나 k = 1인 경우 선형 패턴에 가깝게 보인다. 그림 12에서 읽기성능 만을 고려해볼때, 각 채널에서 같은 양을 읽을 경우에는 (a) 1-channel placement 알고리즘의 성능이 알고리즘 (b) multi-channel

placement 알고리즘 보다 나음을 알 수 있다. 그리고 각 알고리즘에서 SSD management block (또는 k)의 크기가 작을수록 작은 명령 크기에서 나은 성능을 보이지만 요구 명령 크기가 커질수록 성능이 상대적으로 나빠지는 것을 알 수 있다. 이것은 동기화 오버헤드로 인해서 발생하는 것으로 큰 크기의 읽기 명령에서 성능이 좋게 만들려면 SSD 관리 블록의 크기나 k의 크기를 크게 하는 것이 낫다. 이것과 관련하여 iozone[5]을 이용한 성능 측정에서도 확인할 수 있었다. 그림 13의 결과에서 4 KB 읽기요구에서 SSD B가 SSD A보다 나은 성능을 보이는 이유는 SSD B에서 사용하고 있는 플래시 메모리 칩이 2 KB 페이지를 사용하고 있기 때문이다. 이것은 k의 값을 작게 하는 것과 같은 효과를 보인다. 그러나 명령의 크기가 커질수록 SSD B의 성능은 SSD A보다 나쁘다는 것을 알 수 있다.



(a) 1-channel placement (b) Multi-channel placement

그림 12 플래시 메모리 채널의 수는 4개, 4 KB 플래시 페이지 하나를 읽어오는 걸리는 시간을 100 usec라고 가정했을 때, 요구 명령 크기에 따른 반응시간

	Random Read					Random Write				
Request Size (KB)	4	8	16	32	64	4	8	16	32	64
A	19045	34822	49910	70103	90915	1699	2962	4734	7108	10941
B	22828 (1.20)	35904 (1.03)	50179 (1.01)	62035 (0.88)	70064 (0.77)	607 (0.36)	1225 (0.41)	2445 (0.52)	4752 (0.67)	8965 (0.82)
C	19685 (1.03)	35940 (1.03)	61933 (1.24)	101541 (1.45)	131501 (1.45)	38756 (22.81)	55070 (18.59)	66421 (14.03)	73405 (10.33)	75905 (6.94)

그림 13 iozone을 이용한 성능 측정, ()의 값은 SSD A에 대한 상대적인 값임.

그림 14는 SSD B의 명령 크기에 따른 읽기 성능을 보여주는 그래프이다. 실험은 호스트 시스템의 캐시 효과를 배제하기 위해서 Linux 시스템의 SCSI Generic 인터페이스를 이용하여 읽기명령의 크기에 따른 반응 시간을 측정하였다. SSD 내부에서 발생할 지도 모를 미리 읽기 및 캐시 효과를 배제하기 위해서 SSD의 영역을 여러 개로 구분하여 여러 영역에서 구한 반응시간을 평균화한 값이다. 이러한 실험은 SSD A와 C에 대해서도 수행되어졌으며, HDD (시게이트 사의 ST350063 모델)의 성능은 시각적인 비교를 위해서 추가한 것이다.

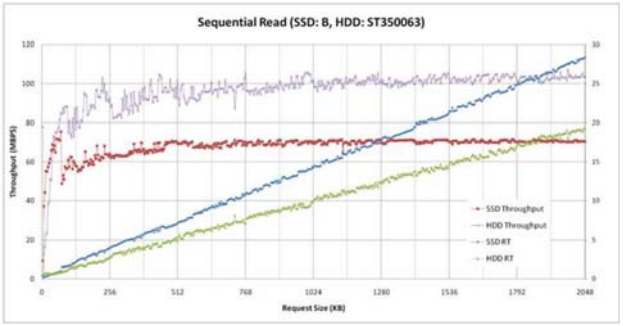


그림 14 SSD B에서 요구 명령의 크기에 따른 성능. 오른쪽 수치는 반응시간으로 단위는 msec임

그림 12의 패턴과 비교해보았을 때, SSD B는 k값이 1인 multi-channel placement 알고리즘에 해당한다. SSD 관리 블록의 크기가 1 MB로 플래시 메모리 페이지보다 크고 패턴이 선형으로 보이기 때문이다.

3. 결 론

본 논문에서는 내장 소프트웨어 설계를 모르는 상황에서 간단한 성능 테스트를 통해서 SSD의 소프트웨어 설계 및 성능 특성을 이해할 수 있는 기법을 제시하였다. SSD 성능에 영향을 미치는 내장 소프트웨어 설계 요소로서 SSD 관리 블록의 크기, 배치, 그리고 디스크 I/O 블록의 SSD 관리 블록으로의 할당을 들었다. 이러한 요소들은 간단한 I/O 테스트를 통해서 들어날 수 있는 것들인 동시 SSD의 성능 특성을 명확하게 보여주는 것이다.

- SSD 관리 블록의 크기가 크게 되면 데이터 복제 오버헤드로 인해서 임의 쓰기의 성능이 떨어지게 된다. 그러나 특이하게도 장기간에 걸친 쓰기 성능 저하는 SSD 관리 블록의 크기가 클수록 작다는 것은 장기간의 걸친 실험의 결과로 알 수 있었다. 이것은 맵핑 테이블 관리가 용이하고 garbage-collection이 수월하기 때문일 것으로 보인다.

- 디스크 I/O 블록의 SSD 관리 블록으로의 할당은 간단하게 보면 로그기법으로 쓰기를 할 것인가 아니면 SSD 관리 블록 상의 일정한 논리적 위치를 고려할 것인가의 문제이다. 로그기법의 경우, 임의 쓰기 성능을 향상시킬 수 있는 방안이기는 하나 관리가 제대로 이루어지지 않을 경우, 순차 읽기 및 임의 읽기 성능을 저하시키는 원인이 된다. 이것은 로그기법을 사용한다고 여겨지는 SSD C의 성능을 보여주고 있는 그림 2를 보면 알 수 있다.

본 논문의 내용으로 보아서 미루어 알 수 있는 것은 SSD의 성능 특성은 내장 소프트웨어 설계로 나타나는 것들로 아직 SSD의 성능 향상을 위해서 해야될 작업이 많다는 것이다.

본 논문의 내용과 달리 SSD의 내장 소프트웨어 설계 요소를 고려하여 성능 특성을 보이지 않는 대신, 각 SSD의 성능차이를 보일 수 있도록 더 많은 I/O 명령 패턴 경우를 고려하여 성능 결과를 볼 수 있도록 제안된 연구[6]는 최근 발표되었다. 해당 연구에서는 I/O 명령이 내려오는 시간으로 (consecutive, pause, burst), I/O 명령의 크기, I/O 명령 패턴 (sequential, random, ordered, partitioned), 그리고 I/O 형태 (읽기, 쓰기)를 이용하여 성능 측정 프로그램을 제안하였다.

Acknowledgement

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음. (IITA-2009-C1090-0902-0045)

참고 문헌

[1] COOKE, J., Flash memory technology direction. In *Proceedings of Microsoft WinHec 2007.*

[2] COOKE, J., Introduction to flash memory. In *Proceedings of Flash Memory Summit 2008.*

[3] AGRAWAL, N., PRABHAKARAN, V., WOBBER, T., DAVIS, J. D., MANASSE, M., AND PANIGRAHY, R., Design tradeoffs for ssd performance. In *Proceedings of 2008 USENIX Annual Technical Conference.*

[4] IOMETER. Iometer benchmark tool. http://www.iometer.org.

[5] IOZONE Iozone filesystem benchmark tool. http://www.iozone.org.

[5] L. Bouganim, B. P. Josson, and P. Bonnet, uFLIP: Understanding Flash IO Patterns, 4th Biennial Conference on Innovative Data Systems Research (CIDR), January 2009.