2013년 8월 전자공학회 논문지 제 50 권 제 8 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 8, August 2013 http://dx.doi.org/10.5573/ieek.2013.50. 8.143

논문 2013-50-8-18

캐시 버퍼와 읽기 요청을 고려한 낸드 플래시 기반 솔리드 스테이트 디스크의 요청 스케줄링 기법

(A Cache buffer and Read Request-aware Request Scheduling Method for NAND flash-based Solid-state Disks)

방 관 후*, 박 상 훈**, 이 혁 준***, 정 의 영*

(Kwanhu Bang, Sang-Hoon Park, Hyuk-Jun Lee©, and Eui-Young Chung)

요 약

솔리드 스테이트 디스크 (SSD)는 고성능 개인용 컴퓨터나 서버 분야에서 뛰어난 특성과 성능을 바탕으로 입지를 넓혀 나가고 있다. 특히 낸드 플래시 메모리에 기반한 SSD가 주류를 이루며 이미 거대한 시장을 확보하고 있는 낸드 플래시 메모리 시장의 큰 부분을 차지하고 있다. 이러한 낸드 플래시 메모리 기반 SSD에는 보통 낸드 플래시 메모리의 특성을 숨기기 위하여 DRAM으로 제작되는 캐시 버퍼가 장착되는데 이 캐시 버퍼는 보다 높은 성능을 달성하기 위해 나중 쓰기 방식을 활용하고 이는 기존의 낸드 플래시 메모리 만을 고려한 스케줄링 기법들을 I/F에서 효과적으로 활용할 수 없게 한다. 따라서 본 논문에서는 I/F에서 사용할 수 있는 캐시 버퍼를 고려한 스케줄링 기법을 제안하고자 한다. 스케줄링 기법은 크게 두 가지 기준을 가지고 스케줄링을 진행하는데 캐시 버퍼의 적중 여부와 읽기 요청에 대한 우선순위이다. 이는 캐시 버퍼에 적중한 요청들을 먼저 처리하여 처리속도를 증가시키고 시스템 성능에 보다 큰 영향을 끼치는 읽기 요청의 지연시간을 줄이기 위함이다. 실험 결과에 따르면 제안하는 스케줄링 기법을 사용했을 때 약 26% 향상된 읽기 성능을 보여주었다.

Abstract

Solid-state disks (SSDs) have been widely used by high-performance personal computers or servers due to its good characteristics and performance. The NAND flash-based SSDs, which take large portion of the whole NAND flash market, are the major type of SSDs. They usually integrate a cache buffer which is built from DRAM and uses the write-back policy for better performance. Unfortunately, the policy makes existing scheduling methods less effective at the I/F level of SSDs Therefore, in this paper, we propose a scheduling method for the I/F with consideration of the cache buffer. The proposed method considers the hit/miss status of cache buffer and gives higher priority to the read requests. As a result, the requests whose data is hit on the cache buffer can be handled in advance and the read requests which have larger effects on the whole system performance than write requests experience shorter latency. The experimental results show that the proposed scheduling method improves read latency by 26%.

Keywords : 낸드 플래시 메모리, solid-state disk, 스케줄링, 캐시 버퍼

* 정회원, ** 학생회원, 연세대학교
(Department of Electrical and Electronic Engineering, Yonsei University)
*** 정회원, 서강대학교
(Department of Computer Science and Engineering, Sogang University)
© Corresponding Author(E-mail: hyukjunl@sogang.ac.kr)
※ 이 논문은 하이닉스, 서강대학교 교내 신진연구비지원 (No. 201210057) 및 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (2012-047670)
접수일자: 2013년4월1일, 수정완료일: 2013년7월22일

Ⅰ. 서 론

솔리드 스테이트 디스크 (SSD)는 기존 하드 디스크 드라이브 (HDD)의 저장매체인 자기 디스크가 아닌 DRAM이나 낸드 플래시와 같은 메모리 소자를 활용하여 HDD를 대체하고자 하는 디스크이다. SSD는 HDD의 기계적 부분들을 모두 제거하여 반응 속도를 비약적으로 향상시켰으며 여러 가지 장점을 기반으로 고성능

개인용 컴퓨터와 서버 그리고 휴대용 컴퓨터에 이르기까지 많은 부분에서 활용되고 있다. 다만 DRAM으로 구성된 SSD는 지나치게 높은 가격과 전력 유실 문제로부터 자유롭지 못하여 일반적으로는 낸드 플래시를 활용하여 SSD를 구성한다.

그림 1은 일반적인 SSD의 내부 구조를 보여준다. 인터페이스 (I/F)를 통해 SSD 내부로 들어온 데이터는 먼저 보통 DRAM으로 구성된 캐시 버퍼에 적재되며, 낸드 플래시 메모리로 옮겨져야 하는 데이터가 발생하게 되면 flash translation layer (FTL)를 거쳐 내드 플래시 메모리에 저장하게 된다.

그림 1의 모듈들은 모두 SSD 성능 향상에 기여하지만 그 중 다수의 낸드 플래시 메모리로 구성하는 multi-channel/way 구조는 SSD에 병렬성을 부여하여 큰 성능향상 효과를 가지게 된다 [1][2]. 더불어 이런 구조의 효율성을 극대화시키기 위하여 많은 연구들이 병렬성을 활용하기 위한 데이터 맵핑과 스케줄링을 제안하였다. [3][4][5].

위와 같은 스케줄링은 그림 1의 FTL과 함께 효과적으로 활용될 수 있으나 I/F 단에서는 스케줄링에 대해서는 제안된 바가 적다. 만약 이미 연구된 스케줄링 기법이 바로 I/F 단에 적용될 경우 그 효과는 반감될 수 있는데, 이는 캐시 버퍼의 나중 쓰기 방식으로 인하여 캐시 버퍼 전과 후의 접근 패턴이 달라지기 때문이며 결과적으로 캐시 버퍼의 상태까지 고려한 스케줄링이 필요하게 된다.

따라서 본 연구에서는 낸드 플래시 메모리 상태 정보뿐 아니라 캐시 버퍼의 상태를 고려하는 스케줄링 기법을 제안하고자 한다. 제안하는 기법은 캐시 버퍼와는 독립적으로 SSD의 인터페이스 단에 캐시 버퍼와 캐시 버퍼 관리 기법의 수정 없이 적용이 가능하며, 읽기 요청과 캐시 버퍼 적중인 요청들에 우선순위를 두어 SSD 전체의 지연시간 감소를 목표로 한다. 실험결과에 따르면, 제안하는 스케줄링 기법을 사용한 경우, first-in, first-out 방식의 스케줄링보다 읽기 지연시간을 평균 26%, 쓰기 지연시간을 2% 감소시켰으며, 낸드 플래시 메모리의 유휴시간을 활용한 스케줄링 기법보다는 읽기 27%, 쓰기 8%를 감소시켰다.

Ⅱ. 본 론

1. 기존 스케줄링 기법과 캐시 버퍼

앞서 언급한 낸드 플래시 메모리의 병렬성을 활용하고자 하는 다양한 스케줄링과 데이터 맵핑 기법 [3][4][5]이 캐시 버퍼와 조합되었을 때 생길 수 있는 문제점을 그림 2의 간단한 예로 확인할 수 있다. 모든 요청은 쓰기 요청으로 가정하며 괄호 안의 숫자를 통해 해당 요청이 필요로 하는 channel과 way 번호를 표기했다. 만약 0번 channel, 0번 way의 낸드 플래시 메모리가 유휴상태로 파악되어 I/F 단에서 이 요청을 내보낸 경우, 쓰기 요청의 특성상 캐시 버퍼에 기록이 되고 낸드 플래시 메모리에 즉시 전달되지 않을 것이다. 만약 캐시 버퍼에서 미 적중이 발생하여 데이터를 낸드 플래시 메모리로 옮겨 캐시 버퍼의 공간을 확보해야 한다면 옮겨져야 하는 데이터는 캐시 버퍼의 관리 기법에 따라 선택되어질 것이고, 이 데이터의 주소가 요구하는 channel

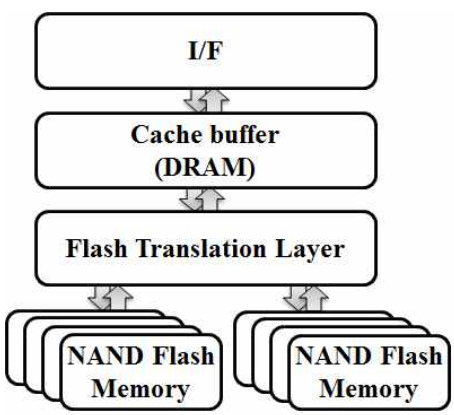


그림 1. 일반적인 SSD 내부 구조
Fig. 1. Generic architecture of SSDs.

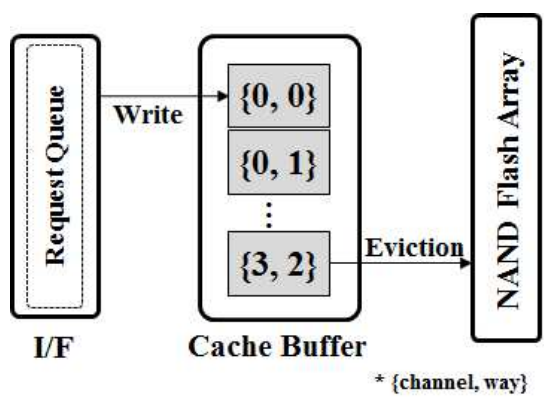


그림 2. 캐시 버퍼를 고려하지 않은 스케줄링 기법
Fig. 2. Scheduling method without consideration of a cache buffer.

과 way는 그림 2에서와 같이 I/F에서 넘어온 요청이 요구하는 channel, way와 다를 수 있다. 결국 I/F 단에서 고려한 최적의 스케줄링에 의한 요청의 순서가 캐시 버퍼 관리 기법에 의해 뒤섞이게 되므로, I/F 단에서의 스케줄링은 캐시 버퍼의 상태를 반드시 고려해야 하는 것이다.

2. 캐시 버퍼 적중과 스케줄링

그림 3은 본 논문에서 제안하고자 하는 캐시 버퍼를 고려한 스케줄링과 그렇지 않은 스케줄링 기법을 그림 3의 (a)와 같은 request queue (RQ)와 캐시 버퍼 (CB) 상황에서 비교하고 있다. RQ와 CB안의 회색 상자로 나타내어진 요청들은 오른쪽으로 갈수록 높은 eviction 우선순위를 가짐을 가정하였다. 각 상자 안의 숫자들은 요청이 필요로 하는 주소로 가정하였다. 모든 요청은 또한 쓰기 요청으로 가정하였다. 또한 그림 1의 SSD와 같은 구조를 가정하여, CB에서 새로운 데이터를 위한 공간 확보 (CB eviction) 를 위해서는 그림 3의 (a)와 같은 우선순위에 따라서 데이터를 낸드 플래시 메모리로 보내주어야 함을 가정하였다. 결국 CB 적중이 발생하지 않는 경우 그림 3의 (b)의 첫 번째 요청 (주소 4)의 예와 같이 처리하는데 CB에 기록하는 (CB write) 것보다 상대적으로 긴 시간을 필요로 하는 낸드 플래시 쓰기 동작 (NAND write) 이후에 CB write 동작을 수행하여 쓰기 요청을 완료할 수 있다. CB와 낸드 플래시 사이의 버퍼 등을 이용하여 이를 줄일 수 있으나, 이 역시 CB와 마찬가지로 제한된 크기만을 가질 수 있으므로, 위와 같은 가정은 현실적이라고 판단할 수 있다.

그림 3의 (b)는 CB의 상태에 대한 고려 없이 SSD에 인가된 순서대로 요청을 내보내는 경우를 가정한다. CB는 그림 3(a)와 같이 가득 차 있고, 처리해야 하는 요청은 CB에서 적중이 발생하지 않았으므로, CB의 eviction 우선순위에 따라 데이터를 낸드 플래시에 보내주어 기록하게 된다. 결론적으로 CB를 구비했음에도 불구하고 쓰기 요청을 처리하는데 낸드 플래시의 느린 동작을 기다려야 하는 것이다. 다수 낸드 플래시의 병렬성 활용을 통하여 이 시간을 줄일 수 있으나, 물리적 공간 등으로 인한 제약은 이 효과를 감소시킨다.

위의 예와는 다르게 그림 3의 (c)의 경우 CB의 상태를 고려하여 CB에서 적중되어 빠른 시간 내에 처리될 수 있는 요청이 있는 경우 인가 순서와 관계없이 먼저 처리하게 된다. 비록 주소 4에 대한 요청이 가장 먼저 RQ에 인가되었으나, 나머지 주소 1~3까지는 CB 적중이 발생하는 주소이므로 때문에 4에 앞서 1~3까지를 먼저

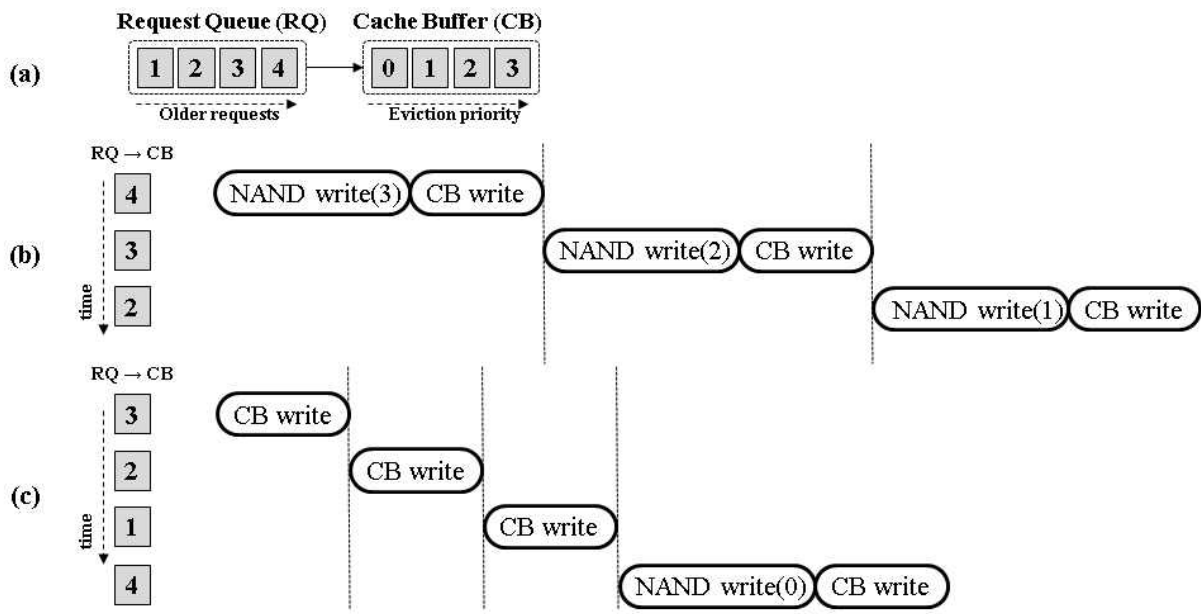


그림 3. (a) 주어진 Request queue 와 cache buffer의 상태에서 (b) 캐시 버퍼 상태를 고려하지 않은 스케줄링 기법과 (c) 캐시 버퍼 상태를 고려한 스케줄링 기법

Fig. 3. Comparison of scheduling methods (b) without consideration of cache buffer and (c) with consideration of cache buffer for (a) the given state of request queue and cache buffer.

처리해 주는 것이다. 이러한 방식은 그림 3(b)에서의 낸드 플래시 쓰기 동작을 필요로 하지 않기 때문에 전체 처리 시간을 획기적으로 단축시킬 수 있게 된다. 다만, 이와 같은 스케줄링을 위해서는 각 요청들 간의 의존도가 없어야 함을 알 수 있다.

3. 읽기 요청 성능 향상의 필요성

읽기와 쓰기 요청 중 기존 SSD에서 보다 많은 성능 저하를 발생 시켰던 요청은 쓰기 요청이었으나, 캐시 버퍼와 나중 쓰기 방식을 적용하여 쓰기 요청에 대한 성능 요구는 상당 부분 만족된 상태이다. 나중 쓰기 방식은 전원 유실에 대한 데이터 손실 위험을 가지고 있기는 하지만 최근에는 개인용 SSD에도 슈퍼 커패시터가 장착되는 등 전원 수준에서의 해결책을 모색하고 있기에 이러한 경향은 계속 될 것으로 보인다.

하지만 다양한 이유로 캐시 버퍼는 쓰기 요청을 위해서만 활용되고 있다. 먼저 낸드 플래시 대비 상대적으로 비싼 DRAM 가격으로 보다 많은 용량을 확보하기 어렵다는 점이다. 또한 읽기 요청에 캐시 버퍼 활용을 위해서는 미리 캐시 버퍼에 데이터를 옮겨야 하는데 이는 시스템이 요청하지 않은 오버헤드로 작용할 수 있기 때문이다. 더불어 쓰기 요청은 캐시 버퍼와 같은 버퍼링을 통하여 성능을 향상시키기 쉽지만, 읽기 요청의 성능은 시스템 전체 성능에 훨씬 큰 영향력을 지니게 되고 일반적으로 쓰기 요청은 읽기 성능은 쓰기 요청보다 시스템 전체 성능에 있어서 큰 영향력을 지니고 있기에 [6][7] 이를 향상시킬 필요성이 존재한다.

SSD 측면에서도 그림 4와 같은 예를 통하여 이러한 필요성이 존재함을 알 수 있다. 그림 4에는 총 3개의 호스트로부터의 요청이 {*R0, W0, R1*}의 순서로 주어짐을 가정하며 *Rn*은 읽기 요청, *Wn*은 쓰기 요청이다. 낸드

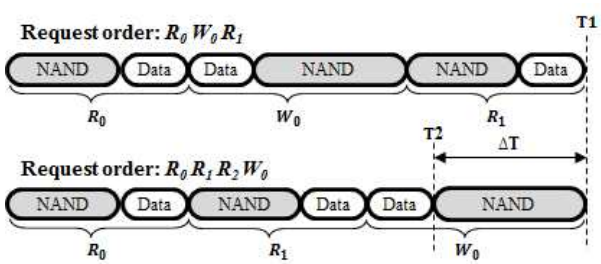


그림 4. 읽기 우선 스케줄링 기법에 성능 향상 예시
Fig. 4. An example of performance improvement by the read-first scheduling method.

플래시의 특성에 따라 읽기 요청은 낸드 플래시에서 데이터를 먼저 읽어온 후 해당 데이터를 호스트에 전달해 주게 되고, 쓰기 요청의 경우 데이터를 먼저 SSD에 전달한 후 낸드 플래시의 처리시간을 기다리게 된다. 따라서 요청들을 순서대로 처리하는 경우 호스트가 관측하는 지연시간은 T1임을 알 수 있다. 하지만 만약 읽기 요청들을 먼저 처리하는 경우, 쓰기 요청은 데이터 전송만으로도 호스트가 다음 작업을 처리할 수 있으므로 T2로 지연시간이 △T만큼 줄어들며, 낸드 플래시가 데이터 기록에 걸리는 시간이 긴 점을 감안하면 그 효과는 매우 클 것이라 예상 가능하다.

결론적으로, 본 논문에서는 I/F 단의 스케줄링 기법에서 읽기 요청에 우선순위를 주어 읽기 요청에 대한 지연시간을 줄이는 방법을 택하고자 한다. 쓰기 요청은 일단 I/F의 버퍼 등에 기록 되면 시스템 측면에서는 해당 요청을 종료시킬 수 있지만 읽기 요청은 모든 데이터가 SSD로부터 전달되어야 하므로 이 지연시간을 줄이는 것은 시스템 영향에 매우 긍정적일 것으로 판단된다.

4. 캐시 버퍼와 읽기 요청을 고려한 스케줄링 기법

본 논문에서 제안하는 스케줄링은 RQ에 머무르는 요청들을 읽기/쓰기 여부와 캐시 버퍼 적중 여부에 따라 4가지로 분류하되 모든 읽기 요청은 쓰기 요청에 대해 우선순위를 가지며, 캐시 버퍼에서 적중한 요청이 그렇지 않은 요청에 대비하여 우선순위를 가진다.

먼저 읽기 요청에 대한 우선순위는 앞서 언급한 바와 같이 쓰기 요청이 나중 쓰기 방식을 사용하기 때문이다. 쓰기 요청의 경우 일단 데이터가 SSD에 들어오면 시스템에서는 다음 요청을 처리할 수 있게 되어 시스템 입장에서는 짧은 지연시간을 가지게 되는데, 그렇지 않은 읽기 요청에 대한 지연시간을 우선순위를 통해 감소시킬 수 있기 때문이다.

캐시 버퍼에 대한 우선순위는 그림 3의 예와 마찬가지로 적중한 데이터를 우선 캐시 버퍼를 통해 처리하여 전체적인 실행 시간을 줄이기 위함이다. 본 논문에서의 캐시 버퍼는 읽기 요청 미 적중 시 공간 할당을 하지 않는 것으로 가정하였고, 적중 시에는 CB에서 직접 데이터를 읽을 수 있도록 하였다.

표 1은 위와 같은 각 요청에 상태에 따른 우선순위를 종합하여 보여주며 숫자가 낮을수록 순위가 높다. 읽기

표 1. 사용된 trace의 특성
Table 1. Characteristics of used traces.

이름	읽기 비율 (%)	평균 읽기 길이	평균 쓰기 길이
fin1	40.8	6.1 KB	4.92 KB
fin2	83.3	2.44 KB	4.98 KB
gen1	85.5	16.49 KB	11.69 KB
gen2	53.7	88.46 KB	8.18 KB
gen3	21.4	14.64 KB	12.67 KB
ftp	38.2	12.96 KB	419.12 KB

```
1: Request RequestScheduling()
2: {
3:     Request req[QUEUE_SIZE];
4:     int req_prior[QUEUE_SIZE];
5:
6:     for( i=0; ∀outstanding requests ; i++)
7:         req_prior[i] = ComputePriority();
8:
9:     for( i=0; ∀outstanding requests ; i++)
10:         for( j=0; ∀outstanding requests ; j++)
11:             if(req[i] depends on req[j])
12:                 req_prior[i] = req_prior[j]+1;
13:
14:    return req[val] s.t. argmin(req_prior[val])
15: }
```

그림 5. 제안하는 스케줄링 기법의 알고리즘
Fig. 5. Algorithm of the proposed scheduling method

요청 중 캐시 버퍼에서 적중하지 못한 요청의 경우 쓰기 요청보다는 높은 우선순위를 가지게 되지만 내부적으로 또 한 번의 우선순위를 판단하게 되며 이 판단의 기준은 읽기 요청이 요구하는 낸드 플래시 메모리의 상태이다. 이는 캐시 버퍼에서 적중하지 못한 읽기 요청은 캐시 버퍼 대신 반드시 낸드 플래시 메모리에 접근해야 하는데 이에 대한 지연시간을 줄이고자 해당하는 낸드 플래시 메모리가 유휴 상태일 경우에 2를 그렇지 않을 경우에 3에 해당하는 우선순위를 할당하였다. 이러한 판단을 위해서는 임의의 요청이 어떤 낸드 플래시 메모리 channel과 way를 필요로 하는지 I/F에서 알 수 있어야 하는데 이를 위해 [8]에서 제안한 것과 같이 논리 주소 기반의 정적 channel, way 할당을 가정하였다.

제안하는 스케줄링 기법은 위와 같은 기준을 가지고 그림 5와 같은 알고리즘에 의해 작동한다. 그림 5의 6~7번째 줄과 마찬가지로 현재 RQ에 존재하는 모든 요청들에 대하여 표 1의 기준을 적용하여 우선순위를 적용한다. 그 후 데이터의 안정성을 위하여 요청 간의 의존성을 확인하는 작업을 9~12번째 줄에서 행한다. 이는 모든 요청들을 비교하여 어떤 요청이 비교 대상인 요청보다 늦게 RQ에 진입하였으며 의존성을 지니고 있는 경우에 해당 요청을 비교 대상 요청보다 늦게 실행시켜 주기 위한 작업이다. 따라서 이와 같은 상황이 발생하는 경우 비교 대상 요청의 우선순위 값에 1을 더하여 해당 요청의 값으로 설정하면, 의존하는 요청보다는 늦게 실행되겠지만, 다른 요청들과는 비교적 공평한 우선순위 비교가 가능할 것이다. 이러한 의존성 비교는 RQ에 처음 요청이 진입할 때 행하며 의존성 종류인 WAR, RAW, RAR, WAW 중 RAR을 제외한 모든 의존성 종류에 대하여 요청의 전체가 아닌 일부분의 주소만 겹치더라도 의존성을 가지는 것으로 간주하였다.

Ⅲ. 실 험

1. 실험 환경

실험을 위하여 그림 1의 구조를 가지는 SSD의 하드웨어와 소프트웨어를 모델링 하였으며, 하드웨어는 SystemC를 활용한 TLM 수준으로 모델링 하였고, 소프트웨어는 C로 작성하여 역시 TLM 수준의 프로세서 모델을 활용하여 실행시켰다. 하드웨어 중 낸드 플래시 메모리 모델은 [9]의 스펙을 참조하였으며 캐시 버퍼는 166 MHz의 동작속도를 가지는 DDR2 메모리로 가정하였다. I/F는 유효 대역폭 3Ghz를 가지고 프로세서와 버스를 포함한 나머지 시스템은 500Mhz의 시스템 클록으로 동작하도록 모델링 하였다. 스케줄링은 I/F에 함께 구현되었으며 총 32개의 요청을 저장 및 스케줄링 할 수 있도록 구현하였으며 낸드 플래시 메모리와 캐시 버퍼의 상태를 알 수 있도록 각 모듈과 연결되어 있다.

I/F 모듈의 호스트 부분에는 표 2와 같은 특성을 가지는 trace를 입력하여 SSD가 이를 실행하는데 걸리는 시간을 측정할 수 있도록 하였다. 앞서 언급한 바와 같이 쓰기 요청은 SSD 내부에 버퍼에 데이터를 기록하면 완료한 것으로, 읽기 요청은 최종 데이터가 SSD 외부로 전송되었을 시에 완료한 것으로 가정하였고, 캐시 버퍼는 읽기 미 적중 시 할당을 하지 않도록 구현되었다. 캐시 버퍼는 LRU 방식으로 동작하며 데이터의 관리 단위는 낸드 플래시 메모리의 페이지 크기와 같은 4KB로 설정하였다. 마지막으로 낸드 플래시 메모리 접근을 위한 FTL은 DFTL [10]을 사용하여 [8]의 방법으로 multi-channel/way 구조에 알맞게 확장하였다.

스케줄링 알고리즘의 성능 비교를 위하여 2가지 비

교군을 추가적으로 구현하였으며, 먼저 들어온 요청을 먼저 내보내는 FIFO 방식과, 현재 유휴 상태의 channel/way로 향하는 요청을 가장 먼저 보내주는 IDLE 방식을 구현, 비교하였다. IDLE 방식은 FTL 수준에서 사용되는 스케줄링 기법을 모사한 것이지만 완전히 동일한 수준은 아니다.

2. 실험 결과

가. 읽기 지연 시간

그림 6은 FIFO 방식에 표준화된 각 방법들의 읽기 지연 시간을 보여준다. 제안하는 방법은 읽기에 우선순위를 두는 만큼 읽기에 대한 성능 향상이 클 것으로 기대되는데 FIFO 대비 약 26%의 읽기 요청 지연시간 감소를 보이는 것을 알 수 있다. 이에 비해 캐시 버퍼를 고려하지 않은 IDLE 기법은 FIFO 보다도 약 2% 정도 나빠진 읽기 지연 시간을 보이는 것을 알 수 있다.

또한 gen2 trace에서의 제안한 스케줄링 기법의 효과가 미비한 것은 이 trace의 평균 읽기 요청의 길이가 매우 길기 때문인 것으로 분석하였다. 본 실험에서 사용한 SSD는 정적인 channel/way 맵핑을 통해 연속된 주소를 요구하는 요청에 대해서는 이미 낸드 플래시 메모리의 병렬성을 충분히 활용하게 되므로 스케줄링의 의한 영향이 적은 것이다.

나. 쓰기 지연 시간

그림 7은 FIFO 방식에 표준화된 각 스케줄링 기법들의 쓰기 지연시간을 보여주고 있다. 제안하는 방법은 FIFO 대비 약 2% 정도의 쓰기 지연 시간 감소를 보였

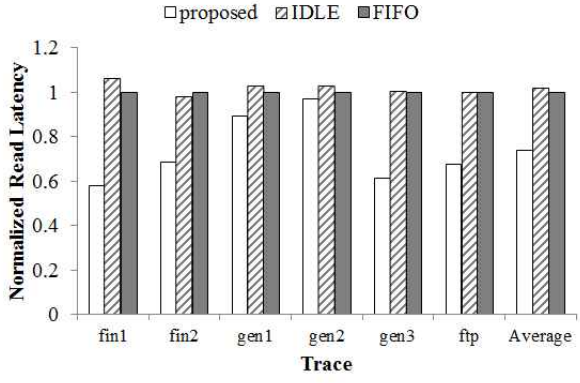


그림 6. FIFO에 표준화된 읽기 지연 시간
Fig. 6. Read latency normalized to that of FIFO.

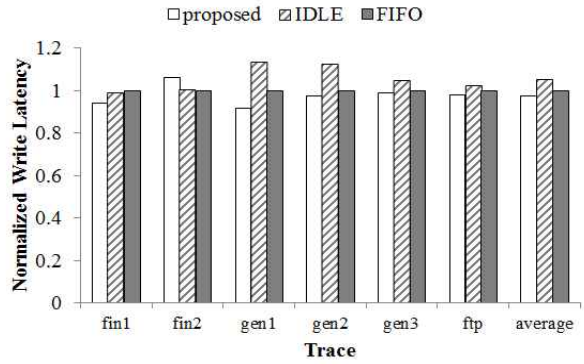


그림 7. FIFO에 표준화된 쓰기 지연 시간
Fig. 7. Write latency normalized to that of FIFO.

으며 IDLE의 경우 5% 정도 쓰기 지연 시간이 증가하여 읽기 지연 시간 만큼의 눈에 띄는 변화는 없는 것으로 관찰 되었다.

IDLE 방법의 경우 읽기 지연 시간과 마찬가지로 RQ에서 내보내는 요청들과 캐시 버퍼에서 낸드 플래시 메모리로 보내지는 요청들의 channel/way에 연관성이 없으므로 대부분의 trace에서 FIFO 보다도 나쁜 성능을 보이게 된다.

제안되는 방법의 경우 쓰기 요청은 읽기 요청보다 항상 낮은 우선순위를 가지게 되므로 다른 작업 없이는 FIFO 보다 낮은 쓰기 지연 시간을 가질 수 없다. 하지만 캐시 버퍼를 고려하여 그림 3과 같은 방식으로 성능 향상을 얻어냈기 때문에 작은 차이지만 평균적으로 FIFO 대비 읽기, 쓰기에서 모두 성능 향상을 얻어낼 수 있었던 것으로 판단된다. 이는 trace 들 중 지역성이 가장 높은 fin1과 gen1 trace에서 가장 눈의 띄는 성능향상을 보여준 것으로 증명할 수 있다. 더불어 fin2 trace의 경우 지역성도 높지 않고 상대적으로 많은 읽기 요청을 가지고 있어 읽기 요청을 우선적으로 처리하기 위해 쓰기 요청이 상당 부분 지연된 것으로 보인다.

Ⅳ. 결 론

본 논문은 캐시 버퍼와 읽기 요청을 고려한 SSD의 요청 스케줄링 기법을 제안하였으며, 이 스케줄링 기법은 기존 SSD를 위한 스케줄링 기법과 크게 두 가지 정도의 차이점을 가진다. 첫 번째로는 다른 스케줄링 기법들은 낸드 플래시 메모리 자체에만 초점을 맞춘데 반해 캐시 버퍼 또한 스케줄링을 위한 고려대상에 포함시

켰다. 따라서 적중 여부에 따른 스케줄링을 가능케 하여 FTL 단계 뿐 아니라 I/F 수준에서의 사용을 목표로 한다. 두 번째로는 읽기 요청에 우선순위를 주어 보다 높은 읽기 성능을 달성하고자 하였다.

총 6개의 입력 trace와 TLM 수준의 시뮬레이터를 사용한 실험결과를 통해서 제안한 스케줄링 기법은 단순 FIFO 기법 대비 성능 향상을 살펴보았으며 읽기 요청 지연시간이 평균 26%, 쓰기 요청 지연시간이 평균 2% 정도 감소한 것을 확인 할 수 있었다. 다만 본 논문에서는 캐시 버퍼의 현재 상태만을 고려하였으며 캐시 버퍼와 낸드 플래시 메모리의 관계를 통해 낸드 플래시로 옮겨지는 데이터를 고려하지 않았는데 이를 고려하면 보다 높은 낸드 플래시 메모리의 활용도를 달성할 수 있을 것이라 기대된다.

REFERENCES

[1] 박현철, 신동군, “Solid State Disk를 위한 주소 매핑 기법,” 전자공학회지, 제27권, 3호, 271-289쪽, 2010년 3월

[2] J. U. Kang, J.-S. Kim, C. Park, H. Park, J. Lee, “A multi-channel architecture for high-performance NAND flash-based storage system,” *Journal of Systems Architecture*, vol. 53, no. 9, pp. 644-658, Sep. 2007.

[3] Y.-B. Chang, L. P. Chang. “A self-balancing striping scheme for NAND-flash storage systems,” In Proceedings of *the 2008 ACM symposium on Applied computing (SAC '08)*, 2008.

[4] Y. Liu, L. Cheng, X. Wang, “Commands scheduling optimized flash controller for high bandwidth SSD application,” *IEEE 11th International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2012.

[5] C. Park, E. Seo, J.-Y. Shin, S. Maeng, J. Lee, “Exploiting Internal Parallelism of Flash-based SSDs,” *Computer Architecture Letters* , vol.9, no.1, pp.9,12, Jan. 2010.

[6] M.K. Qureshi, M.M. Franceschini, “Improving read performance of Phase Change Memories via Write Cancellation and Write Pausing,” *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)* 2010.

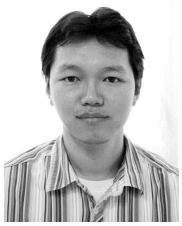
[7] Daniel Pierre Bovet, et al., “Understanding the Linux Kernel”, Chapter 14.2, 2003.

[8] S.-H. Park, S.-H. Ha, K. Bang, E.-Y. Chung, “Design and analysis of flash translation layers for multi-channel NAND flash-based storage devices,” *Consumer Electronics, IEEE Transactions on*, vol.55, no.3, pp.1392-1400, August 2009.

[9] HY27UH08AG5M, Hynix Semiconductor Inc., www.hynix.com.

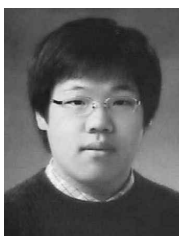
[10] A. Gupta, Y. Kim, and B. Urgaonkar, “Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings,” *SIGPLAN Not.*, vol. 44, pp. 229-240, March 2009.

저 자 소 개



방 관 후(정회원)
2006년 연세대학교 학사 졸업
2008년 연세대학교 석사 졸업
2008년～현재 연세대학교 박사과정
<주관심분야 : 저전력 설계기술, 플래시 메모리 응용, 바이오 응용>

이 혁 준(정회원)
1993년 University of Southern California 학사 졸업
1995년 Standford Unversity 석사 졸업
2001년 Stanford University 박사 졸업
<주관심 분야 : 임베디드 시스템, 바이오 컴퓨팅, 저전력 설계, 메모리 구조>



박 상 훈(학생회원)
2009년 연세대학교 학사 졸업
2009년～현재 연세대학교 통합과정
<주관심분야 : 시스템 구조, 플래시 메모리 응용>

정 의 영(정회원)
1988년 고려대학교 학사 졸업
1990년 고려대학교 석사 졸업
2002년 Stanford University 박사 졸업
<주관심 분야 : 시스템 구조, VLSI 설계, 저전력 설계>