ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 2, pp. 177-185, 2016. 2
http://dx.doi.org/10.5626/JOK.2016.43.2.177

이중 쓰기 버퍼를 활용한 SSD의 성능 향상 및 수명 연장 기법

(Dual Write Buffer Algorithm for Improving Performance and Lifetime of SSDs)

한 세 준 † 강 동 현 † 엄 영 익 ††
(Se Jun Han) (Dong Hyun Kang) (Young Ik Eom)

요 약 본 논문에서는 NVRAM과 DRAM으로 구성된 SSD의 쓰기 버퍼 구조 및 제안된 쓰기 버퍼 구조에 적합한 이중 쓰기 버퍼 알고리즘을 제안한다. 읽기/쓰기 작업이 혼합된 일반적인 워크로드에서 저장 장치의 성능을 향상시키기 위해서 읽기 작업에 의해 참조되는 페이지 또한 고려하였다. 그리고, NVRAM에 저장되는 쓰기 작업에 의해 참조된 페이지를 효율적으로 관리하여 낸드 플래시 메모리에서 발생하는 삭제 연산의 횟수를 감소시켜 SSD의 수명을 연장하였다. 우리는 실험을 통해 제안하는 쓰기 버퍼 알고리즘이 버퍼 적중률을 최대 116.51% 향상시켰으며, 낸드 플래시 메모리에서의 삭제 연산의 횟수를 최대 56.66% 감소시킬 수 있었다.

키워드: 낸드 플래시 메모리, SSD, 쓰기 버퍼 알고리즘, NVRAM

Abstract In this paper, we propose a hybrid write buffer architecture comprised of DRAM and NVRAM on SSD and a write buffer algorithm for the hybrid write buffer architecture. Unlike other write buffer algorithms, the proposed algorithm considers read pages as well as write pages to improve the performance of storage devices because most actual workloads are read-write mixed workloads. Through effectively managing NVRAM pages, the proposed algorithm extends the endurance of SSD by reducing the number of erase operations on NAND flash memory. Our experimental results show that our algorithm improved the buffer hit ratio by up to 116.51% and reduced the number of erase operations of NAND flash memory by up to 56.66%.

Keywords: NAND flash memory, SSD, write buffer algorithm, NVRAM

· 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.R0126-15-1082, (ICBMS-총괄) ICBMS(IoT, 클라우드, 빅데이터, 모바일, 정보보호) 핵심 기술 개발 사업 총괄 및 엑사스케일급 클라우드 스토리지 기술 개발)
· 이 논문은 2015 한국컴퓨터종합학술대회에서 'SSD의 성능과 수명 향상을 위한 하이브리드 쓰기 버퍼 관리 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 정보통신대학
sejun.han@skku.edu
kkangsu@skku.edu
†† 종신회원 : 성균관대학교 정보통신대학 교수(Sungkyunkwan Univ)
yieom@skku.edu
(Corresponding author임)

논문접수 : 2015년 8월 5일
(Received 5 August 2015)
논문수정 : 2015년 10월 14일
(Revised 14 October 2015)
심사완료 : 2015년 10월 22일
(Accepted 22 October 2015)

정보과학회논문지 제43권 제2호(2016. 2)

1. 서 론

최근, 낸드 플래시 메모리 기반의 저장장치인 SSD (Solid-State Drive)는 지금까지의 주요 저장장치인 HDD (Hard Disk Drive)에 비해서 빠른 I/O 응답성, 저전력 그리고 내구성을 바탕으로 HDD를 빠르게 대체하고 있다. 하지만, SSD는 낸드 플래시 메모리 특성으로 인해 읽기 연산이 쓰기 연산보다 느리며 제한된 수명을 가지고 있다. 이러한 단점들을 완화시키기 위해 SSD는 내부에 DRAM과 같은 빠른 속도의 메모리를 쓰기 버퍼(앞으로 쓰기 버퍼는 버퍼라고 칭한다.)로 사용하고 있다. 버퍼를 관리하는 버퍼 알고리즘은 플래시 메모리로 수행되는 읽기/쓰기 연산을 효율적으로 관리하여 플래시 메모리의 성능 및 수명을 개선하고 있다. 하지만, SSD 내부의 버퍼는 SSD의 용량에 비해 적은 용량이기 때문에 버퍼를 효율적으로 관리하기 위해 다양한 버퍼 알고리즘들이 제안되어 왔다[1-5]. 하지만 이들은 워크로드의 일부 특성과 낸드 플래시 메모리의 일부 특성만을 고려하고 있다.

최근 PRAM(Phase-Change RAM)과 STT-RAM (Spin Transfer Torque RAM)과 같은 다양한 형태로 NVRAM (Non-Volatile Memory)이 연구되고 있다. NVRAM은 바이트 단위의 접근과 같은 메모리의 특성과 전원이 공급되지 않아도 저장된 데이터가 유지되는 저장장치의 대표적인 특성을 갖추고 있다. 또한, NVRAM에서의 읽기 연산의 속도가 메모리에서 수행되는 읽기 연산의 속도와 유사하여 DRAM을 대신할 차세대 메모리로 여겨지고 있다.

본 논문에서는 새로이 연구되고 있는 NVRAM을 활용하는 버퍼 구조와 버퍼 알고리즘을 제안한다. 제안하는 구조는 소량의 NVRAM을 DRAM과 함께 버퍼로 사용한다. 그리고, 기존의 버퍼 알고리즘들과 달리 읽기 작업에 의해 요청된 페이지들도 버퍼 공간에 저장하는 방법을 통해 읽기/쓰기 요청이 혼합된 일반적인 상황을 고려 하였다. 또한, DRAM에 위치한 수정된 페이지들을 효율적으로 NVRAM으로 이동시켜 데이터 일관성을 높일 수 있도록 하였다. 마지막으로, NVRAM에서는 낸드 플래시 메모리의 쓰기 특성을 고려한 페이지 관리를 통하여 SSD의 수명을 향상시킬 수 있도록 하였다.

본 논문은 다음과 같은 구성으로 되어 있다. 2장에서는 기존의 버퍼 알고리즘에 관련된 연구를 살펴본다. 본 논문에서 제안하는 버퍼 알고리즘은 3장에서 자세히 설명하고, 4장에서는 성능을 평가한다. 마지막으로 5장에서는 연구 결과를 요약한다.

2. 관련 연구

버퍼 관리 알고리즘은 일반적으로 버퍼에 저장된 데이터를 관리 하는 단위에 따라 페이지 기반 알고리즘, 블록 기반 알고리즘 그리고 하이브리드 알고리즘으로 분류된다.

페이지 기반 버퍼 관리 기법은 시간지역성(Temporal Locality)을 고려하여 설계된 LRU(Least Recently Used) 및 이를 유사하게 구현한 CLOCK 알고리즘, 그리고 2Q 알고리즘[6] 들이 있다. 하지만, 이러한 페이지 기반의 알고리즘들은 낸드 플래시 메모리의 특성을 고려하고 있지 않고 있기 때문에 낸드 플래시 메모리로 구성된 SSD의 버퍼 보다는 메인 메모리의 버퍼 캐시를 관리하는 알고리즘으로 보다 적합하다.

FAB[1]과 LB-CLOCK[2]은 블록 기반의 버퍼 관리 기법으로 전통적인 페이지 기반의 버퍼 관리 기법과 달리 낸드 플래시 메모리의 특성을 고려하였으며, 공간지역성(Spatial Locality)에 포커스를 맞추고 있다. FAB[1]은 대용량의 순차 쓰기 패턴(Sequential Write Pattern)이 자주 발생하는 PMP(Portable Media Player)를 대상으로 제안되었으며, 블록 기반의 LRU로 버퍼를 관리한다. 버퍼의 여유 공간이 없는 경우 가장 많은 페이지를 포함하고 있는 블록을 희생 블록(Victim Block)으로 선정하여 블록에 포함된 모든 페이지들을 동시에 교체한다. 이와 같은 과정을 통해 FAB은 낸드의 가비지 컬렉션(Garbage Collection) 비용을 감소시키고 순차 쓰기 패턴의 비율을 높인다. LB-CLOCK[2]은 기존의 페이지 기반의 CLOCK 알고리즘을 플래시 메모리의 특성을 반영하여 블록 기반으로 확장한 것이다. LB-CLOCK은 전통적인 CLOCK 알고리즘과 유사하게 블록에 포함된 페이지중 하나라도 접근이 되면 블록의 참조 비트(Reference Bit)을 설정한다. LB-CLOCK은 희생 블록을 선정할 때 블록들의 최신성(Recency)과 이용률(Utilization)을 고려하여 최근에 접근되지 않은 블록들 중에서 이용률이 높은 블록을 희생 블록으로 선정한다. 하지만, 이러한 블록 기반의 버퍼 알고리즘들은 임의 페이지 쓰기와 순차 쓰기가 혼합된 일반적인 컴퓨팅 환경에서는 적합하지 않다.

하이브리드 버퍼 관리 기법은 앞서 설명한 버퍼 알고리즘들과 달리 버퍼를 관리할 때 하나 이상의 단위로 페이지들을 관리하는 기법이다. 대표적인 하이브리드 기법으로는 BPAC[3]과 CBM[7]을 예로 들 수 있다. BPAC[3]은 워크로드의 참조 지역성을 고려하기 위해 자주 접근되는 페이지 리스트와 그렇지 않은 리스트로 구분하여 관리한다. 그리고 실시간으로 워크로드의 페이지 참조를 분석하여 동적으로 페이지들을 이동시킨다. 하지만, 이와 같은 실시간 모니터링을 하기 위해 많은 연산과정이 필요하다. CBM[7]은 앞서 설명한 알고리즘과 달리 소량의 NVRAM을 DRAM과 함께 사용하여

버퍼를 구성한다. 또한, 기존의 버퍼 기법들과 달리 읽기 작업을 통해 참조된 페이지들을 DRAM에 저장을 하여 응답성을 높인다. 그리고 NVRAM에서 희생 블록을 선정하여 낸드 플래시로 쓰기 작업이 진행될 때, NVRAM에서 선정된 희생 블록과 동일한 논리 블록 번호를 갖는 DRAM의 페이지들을 병합하여 순차 쓰기의 비율을 높인다. 하지만, CBM은 소량의 NVRAM에 쓰기 작업에 참조되는 페이지만 저장하고 있기 때문에 쓰기 작업이 빈번히 발생하는 워크로드에서는 낮은 버퍼 적중률을 보일 수 있으며, DRAM에 위치한 페이지의 병합으로 인해 낸드 플래시 메모리에 불필요한 쓰기를 유발하여 낸드 플래시 메모리의 수명에 좋지 않은 영향을 미칠 수 있다.

이와 같이 버퍼를 활용하여 낸드 플래시 메모리 기반 저장장치의 성능을 높이는 방법에 대해 많은 연구가 진행되었다. 하지만, NVRAM을 이용하는 하이브리드 기법은 아직까지 충분히 연구가 진행 되어 있지 않아 연구가 필요한 실정이다.

3. 구조 설계

이 장에서는 제안하는 구조의 세부적인 설계와 동작 방식에 대해서 알아볼 것이다. 최근 다양한 형태로 연구되고 있는 NVRAM은 고유의 특성으로 인해 DRAM과 같이 병렬적으로 사용될 가능성이 높다. 그래서, 우리는 최근 활발히 연구되는 NVRAM을 DRAM과 함께 SSD 내부의 버퍼로 활용하는 그림 1과 같은 구조를 소개하며, 이를 기반으로 NVRAM과 DRAM의 페이지를 효율적으로 관리하기 위한 버퍼 알고리즘을 제안한다. 그리고, 본 논문에서는 제안 기법을 DWB(Dual Write Buffer)라고 명칭한다. DWB는 NVRAM과 DRAM의 페이지들을 두 개의 수정된 CLOCK 알고리즘을 통하여 각각 관리한다.

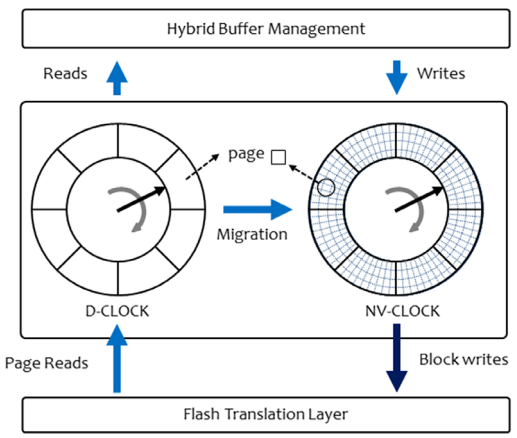

그림 1 제안된 이중 쓰기 버퍼 구조
Fig. 1 Overview of Dual Write Buffer architecture

3.1 DWB 알고리즘

DWB는 DRAM의 페이지들을 관리하는 D-CLOCK과 NVRAM의 페이지들을 관리하는 NV-CLOCK으로 구성된다.

3.1.1 D-CLOCK 알고리즘

임의 페이지 쓰기 작업은 낸드 플래시 메모리에게 좋지 않은 영향을 미치기 때문에 대부분의 SSD 버퍼 알고리즘들은 임의 페이지 쓰기를 순차 쓰기로 변경하고, 낸드 플래시 메모리에서 발생하는 쓰기 횟수를 줄이는 데 많은 노력을 기울이고 있다. 이와 같이 기존의 버퍼 알고리즘들은 쓰기 요청에 의한 페이지 접근에만 집중한 것에 반하여, DWB는 다음과 같은 이유로 읽기 요청에 의한 페이지 접근 또한 고려하고 있다. 읽기 요청은 실시간성을 띄고 있으며, 읽기/쓰기 작업이 혼합된 워크로드에서는 읽기 요청에 의해 쓰기 작업이 지연되는 상황이 발생할 수 있기 때문이다.

DWB는 버퍼에서 읽기 요청에 의한 페이지 접근이 발생하면, 우선적으로 DRAM과 NVRAM으로 구성된 버퍼 영역에서 해당 페이지가 있는지 확인을 한다. 만약, 버퍼 공간에 요청된 페이지가 존재하지 않는 경우라면, 낸드 플래시 메모리에서 요청된 페이지를 읽어 DRAM 버퍼 공간에 새로운 페이지를 할당하여 저장을 하고 호스트 시스템에 반환한다. 이와 반대로, 시스템의 쓰기 요청에 의해 페이지 접근이 발생하면, 우선적으로 버퍼에서 해당 페이지를 찾아 업데이트 한다. 만약, 버퍼에 접근된 페이지가 버퍼 영역에 존재하지 않는다면, 우선적으로 DRAM에 새로운 공간을 할당하여 수정된 페이지를 저장한다.

DWB는 페이지 기반의 CLOCK 알고리즘을 수정하여 DRAM의 페이지들을 관리하며, 이를 D-CLOCK이라고 명칭한다. D-CLOCK은 기존의 CLOCK 알고리즘과 달리 쓰기 작업에 의해 참조된 페이지들을 우선적으로 희생 페이지로 선정하기 위해 참조 비트를 사용한다. 읽기 작업에 의해 DRAM버퍼 영역에 저장된 페이지들에 대해서는 참조 비트를 설정한다. 그리고, DRAM 버퍼 영역에서 쓰기 작업에 의해 저장되거나 수정된 페이지들은 참조 비트를 설정하지 않는다. 그리고 DRAM의 여유 공간이 부족할 때 쓰기에 의해 참조된 페이지들이 우선적으로 희생 페이지로 선정되도록 하며 이를 'dirty first migration'이라고 명칭한다.

DWB는 DRAM에서 여유 공간이 부족한 경우에는 다음과 같은 이유로 쓰기 요청에 의해 할당된 페이지를 우선적으로 희생 페이지로 선정하여 NVRAM으로 이동시키는 'dirty first migration' 기법을 사용한다. 첫째로,

```
1   void D_CLOCK(page *p)
2     if(p is not in the DRAM) {
3         if(the DRAM is full) {
4           v = select_dramvictim();
5           if(v.status is clean)
6             evict v;
7           else
8             migrate v to NVRAM;
9           }
10          insert p into DRAM based on page number;
11          if(p status is clean)
12            p.reference_bit = true;
13      }
14    }
```

그림 2 DRAM 에서의 희생 페이지 선정

Fig. 2 Victim selection on DRAM

```
1   void NV_CLOCK(page *p) {
2     if(the NVRAM is full) {
3       v_nvram = select_nvramvictimblock();
4       v_dram = find_drampage(v_nvram.block_number);
5       if(v_dram.status is not clean) {
6         evict v_nvram;
7         flush v_dram.data;
8         v_dram.status = clean;
9       }
10      else
11        evict v_nvram;
12    }
13    insert p into NVRAM base on page number;
14  }
```

그림 3 NVRAM 에서의 희생 블록 선정

Fig. 3 Victim selection on NVRAM

NVRAM은 비휘발성 메모리인 동시에 읽기 요청에 의한 페이지 접근 시 DRAM과 유사한 속도로 빠르게 응답을 보이기 때문에 플래시 메모리에서의 페이지 접근보다 응답시간을 단축시킬 수 있다.

둘째로, 비휘발성의 NVRAM에서 페이지들을 그룹핑을 할 수 있기 때문에 임의 페이지 쓰기를 낸드 플래시 메모리에 유리한 순차 쓰기로 변경할 수 있다. DWB는 DRAM에서 희생 페이지로 선정된 페이지가 쓰기에 의해 수정된 페이지인 경우에는 NVRAM으로 이동시켜 여유 공간을 확보하며, 읽기 요청에 의해 참조된 페이지라면 DRAM에서 바로 삭제하여 여유 공간을 확보한다.

3.1.2 NV-CLOCK 알고리즘

NVRAM의 페이지를 관리하는 NV-CLOCK도 전통적인 CLOCK 알고리즘을 수정하여 적용하고 있다. NVRAM의 페이지들을 관리하는 NV-CLOCK은 NVRAM에서 낸드 플래시 메모리로 순차 쓰기를 수행하기 위해 페이지들의 논리 주소로 그룹핑하여 페이지들을 블록 단위로 관리한다. 그리고 NV-CLOCK은 블록 기반으로 CLOCK 알고리즘을 적용하여 NVRAM에서 여유 공간이 부족할 경우 희생 블록을 선정한다. 그리고, 선정된 희생 블록에 속한 여러 개의 페이지를 한번에 낸드 플래시 메모리로 쓰기를 수행한다. 블록 기반으로 관리되는 NV-CLOCK은 NVRAM에 저장된 페이지의 최신성을 유지하기 위해서 블록에 속한 하나의 페이지가 접근되더라도 블록의 참조 비트를 설정한다. 그리고 NVRAM에서 희생 블록으로 선정된 블록과 같은 논리 블록 번호를 갖는 DRAM의 페이지들에서 쓰기에 의해 수정된 페이지들이 있다면, NV-CLOCK으로 선정된 희생 블록의 페이지들이 낸드 플래시로 쓰기가 수행될 때 동시에 쓰기 작업을 수행하여 순차 쓰기의 비율을 높여준다.

3.2 예 제

그림 4와 그림 5는 제안된 버퍼 알고리즘인 DWB 의 동작을 설명하고 있다. 그림 4는 DRAM의 페이지를 관리하는 D-CLOCK의 동작 상황을 (a) - (e) 단계를 통해 보여주고 있으며, 그림 4(a)는 D-CLOCK의 초기 상태를 나타낸다. 페이지 5, 6 그리고 7은 읽기에 의해 임시로 저장된 페이지를 나타내며, 페이지 4, 13 그리고 15는 쓰기에 의해 저장된 페이지를 나타낸다. 그리고 D-CLOCK의 포인터는 시계 방향으로 이동하며, 현재는 페이지 5를 가리키고 있다.

그림 4(b)는 쓰기 동작에 의해 요청된 페이지 10이 버퍼 영역에 존재하지 않은 상황을 보여주고 있다. 현재 DRAM의 6개의 페이지를 모두 사용하고 있기 때문에

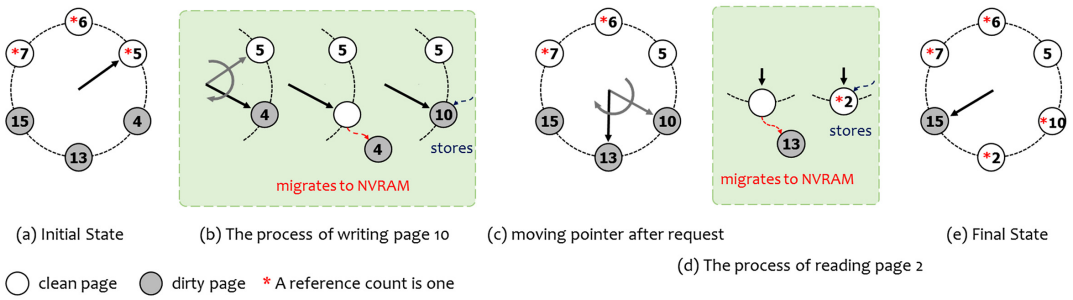

그림 4 DRAM의 공간이 6 페이지일 경우의 D-CLOCK 동작 예제

Fig. 4 An example of D-CLOCK operations, with a maximum of 6pages in D-CLOCK

D-CLOCK은 포인터를 시계 방향으로 이동시키면서 희생 페이지를 찾는다. 페이지 5는 참조 비트가 설정되어 있기 때문에 참조 비트를 설정을 해제시키고, 포인터를 페이지 4로 이동시킨다. 페이지 4는 참조비트가 설정되어 있지 않기 때문에 희생 페이지로 선택이 된다. 그리고, 페이지 4는 쓰기 요청에 의해 수정된 페이지이기 때문에 NVRAM으로 이동시키고 DRAM의 공간을 확보한 뒤 페이지 10을 DRAM에 저장을 한다. 그리고 D-CLOCK의 포인터를 이동시킨다(그림 4(c)).

그림 4(d) - 그림 4(e)는 읽기 동작에 의해 접근된 페이지가 버퍼 공간에 없을 경우의 동작에 대한 예시이다. 앞서 설명한 쓰기 동작에 요청된 페이지가 버퍼 영역에 없을 경우와 기본적으로 같은 프로세스로 동작을 한다. 하지만, 제안된 쓰기 버퍼 기법에서는 읽기에 의해 참조된 페이지를 최대한 DRAM에서 유지시키기 위해서 읽기에 의해 참조된 페이지를 DRAM에 저장할 때 참조 비트를 설정하여 희생 블록 선택을 최대한 지연시킨다(그림 4(d)).

그림 5는 NVRAM의 페이지를 관리하는 NV-CLOCK의 동작을 설명하고 있다. DRAM과 NVRAM모두에 여유 공간이 없는 상황에서 DRAM에 새로운 페이지를 저장하기 위해 공간을 확보하는 것으로 시작한다. 우선 DRAM에 새로운 페이지를 저장하기 위해서 D-CLOCK은 페이지 15를 희생 페이지로 선택한다. 희생 페이지로 선택된 페이지 15는 쓰기에 의해 내용이 변경된 페이지이기 때문에 NVRAM으로 이동시킨다(그림 5(a)). 이 때, NVRAM 또한 여유 공간이 없기 때문에 NV-CLOCK은 희생 블록을 선택한다. 앞서 설명하였듯이 NV-CLOCK은 블록의 최신성과 사용률을 고려하여 블록 2를 희생 블록으로 선정한다(그림 5(c)). 이때, DRAM에 위치한 페이지 중 NVRAM에서 희생 블록으로 선정된 '블록 2'와 동일한 논리 블록 번호를 갖고 쓰기에 의해 참조된 페이지를 찾는다. 그림 5에서는 페이지 10이 희생 블록과 동일한 논리 블록 번호를 갖고 쓰기에 의해 참조된 페이지이기 때문에 블록 2와 함께 낸드 플래시 메모리로 쓰기를 수행한다(그림 5(d)). 그리고, DRAM에 위치한 페이지 10은 낸드 플래시 메모리의 페이지 10과 동일한 데이터를 갖고 있기 때문에 읽기 요청에 의해 저장된 페이지와 동일하게 설정을 변경해준다.

4. 실험 및 성능 평가

4.1 실험 환경

제안하는 버퍼 알고리즘의 성능을 평가하기 위해서 SSD 패치[8]가 적용된 DiskSim 시뮬레이터[9]에 버퍼 알고리즘을 적용하였다. 실험을 위한 시뮬레이터 설정은 SSD의 용량을 64GB로 한 것을 제외하고는 CBM[7]과 동일하게 설정하였다. 그리고 DRAM과 NVRAM으로 구성된 하이브리드 버퍼 관리 기법인 CBM과 DWB의 NVRAM 용량을 전체 버퍼 용량의 10%로 할당하였으며, 나머지 공간인 버퍼의 90%를 DRAM 용량으로 할당하였다 그리고, FAB은 버퍼를 DRAM만을 고려하고 있기 때문에 버퍼 전체 용량을 DRAM으로 할당하였다. 그리고 하이브리드 버퍼 알고리즘은 CBM과 DWB에서 NVRAM의 용량을 전체 버퍼 용량의 10%로 설정한 이유는 아직까지 NVRAM의 제조 공정은 아직 안정화되지 않았기 때문에 DRAM에 비해 제조 비용이 아직 높기 때문이다.

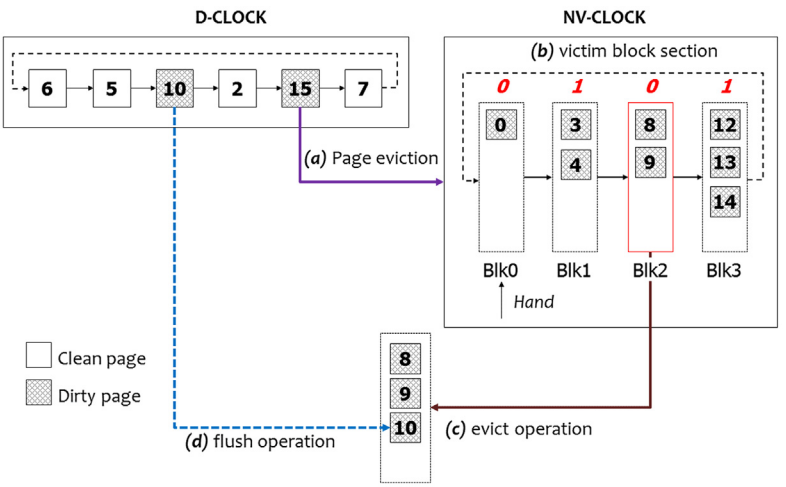

그림 5 NVRAM의 공간이 8 페이지일 경우의 NV-CLOCK의 동작 예제

Fig. 5 An example of NV-CLOCK operations, assuming a maximum of 8 pages in NV-CLOCK and 4 pages in a block

표 1 사용된 워크로드 특성
Table 1 Characteristics of the workloads

	쓰기 작업 비율	워크로드 크기
Financial [10]	18%	1.8 GB
CAMWEBDEV [11]	92%	0.52 GB
Filebench [12]	50%	2.81 GB

성능평가 실험에서 사용된 워크로드는 읽기 작업의 비율을 달리하여 선정하였다. 읽기/쓰기 작업의 비율에 따른 제안된 버퍼의 성능 향상 여부를 확인하기 위해 읽기/쓰기 작업의 비율을 달리 하여 워크로드를 선정하였다. 성능 평가에 사용된 워크로드의 특징은 표 1과 같다.

4.2 버퍼 관리 기법에 따른 성능 비교

우리는 실험에서 버퍼 적중률과 SSD에서 발생하는 삭제 연산의 발생 횟수를 측정하였다. 버퍼 적중률은 버퍼 알고리즘 비교에서 대표적으로 사용되는 성능 평가 지표 중의 하나이다. HDD와 달리 물리적인 페이지에 제자리 덮어 쓰기(In-place-update)가 불가능한 낸드 플래시 메모리 기반의 저장장치는 데이터를 업데이트 하기 위해서 제한된 삭제 연산을 수행한다. 이와 같이 낸드 플래시 메모리에서의 수행된 삭제 연산의 횟수는 수행 가능한 쓰기 작업 횟수를 예측할 수 있기 때문에 낸드 플래시 메모리 기반의 저장장치인 SSD의 수명을 예측하는 중요한 지표이다.

4.2.1 버퍼 적중률

Financial 워크로드는 쓰기 작업보다 읽기 작업이 많이 발생하는 OLTP(Online Transaction Processing) 작업의 대표적인 워크로드이다. OLTP 작업의 워크로드는 주로 순차 읽기/쓰기 작업 보다는 임의 읽기/쓰기 작업이 주로 발생하는 워크로드로 블록 기반의 쓰기 버퍼 알고리즘에서 낮은 성능을 보여주고 있다.

그림 6에서 확인할 수 있듯이 DWB가 다른 버퍼 알고리즘인 FAB에 비하여 최대 116.51% 그리고 CBM에 비해서 최대 8.91% 향상된 결과를 보여주고 있다. 다른 기법에 비해 순차 쓰기의 비율이 높은 워크로드에서 높은 성능을 발휘하는 블록 기반의 알고리즘인 FAB은 다른 버퍼 알고리즘과 달리 낮은 성능을 보여주고 있다. 그 이유로는 임의로 접근되는 페이지들이 서로 다른 블록에 포함되었을 때 FAB의 희생 블록 선정 알고리즘에 의해 재참조되기 전에 희생 블록으로 선정되어 방출 되는 문제가 발생할 수 있기 때문이다.

CAMWEBDEV 워크로드는 쓰기가 읽기보다 월등히 많이 발생하는 워크로드이며 임의 페이지 참조보다 순차 페이지 참조가 많이 발생한다. 쓰기의 비율이 월등히 높은 CAMWEBDEV 워크로드에서도 제안된 버퍼 알고리즘인 DWB는 FAB에 비해서 30.77% 그리고 CBM에 비해서 109.15% 향상된 버퍼 적중률을 보임을 그림 7을 통해서 확인할 수 있다. CABWEBDEV 워크로드에서 CBM이 다른 버퍼 알고리즘에 비해서 낮은 성능을 보이는 이유는 CBM은 적은 양의 NVRAM 만을 이용해 쓰기 작업에 의해 수정된 페이지를 저장하고 있기 때문에 지속적으로 쓰기 작업이 발생하는 상황에서는 DRAM에 여유 공간이 있는 상황에서도 NVRAM에서 여유 공간을 확보하기 위해 지속적으로 희생 블록(혹은 페이지)을 방출하기 때문이다. 이와 같은 이유로 버퍼에 저장된 페이지가 재참조될 때까지 충분히 머무를 수 없기 때문에 다른 버퍼 알고리즘에 비해 낮은 버퍼 적중률을 보이고 있다.

마지막으로 읽기/쓰기 작업이 비슷한 비율로 발생하는 상황에서 DWB가 버퍼 적중률 향상에 어떤 영향을 미치는지 확인하기 위해 Filebench 벤치마크를 이용하여 읽기/쓰기 작업이 비슷한 비율로 발생하는 워크로드

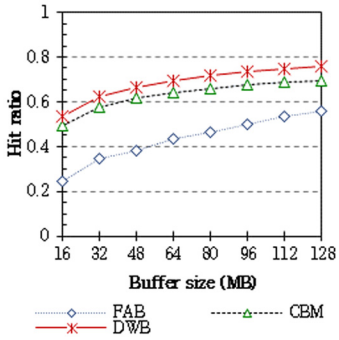

그림 6 Financial 워크로드의 버퍼 적중률
Fig. 6 Buffer hit ratio of Financial workload

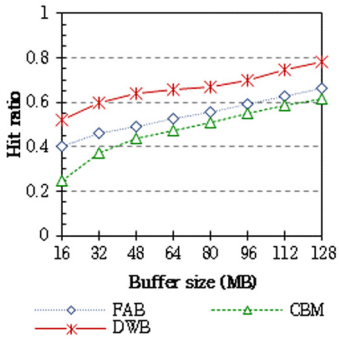

그림 7 CAMWEBDEV 워크로드의 버퍼 적중률
Fig. 7 Buffer hit ratio of CAMWEBDEV workload

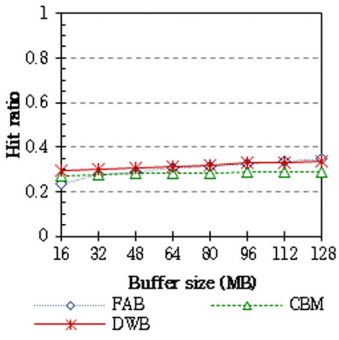

그림 8 Filebench 벤치마크 워크로드 버퍼 적중률
Fig. 8 Buffer hit ratio of Filebench benchmark workload

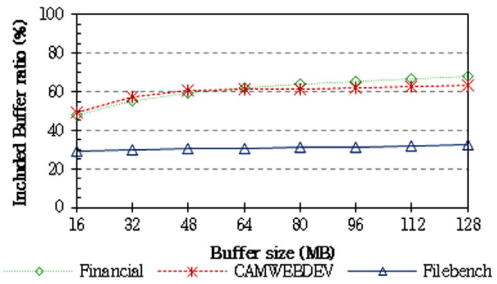

그림 9 동일 페이지 재접근시 버퍼 사이즈 내 비율
Fig. 9 The ratio of the distance between accesses to each page included in the buffer size

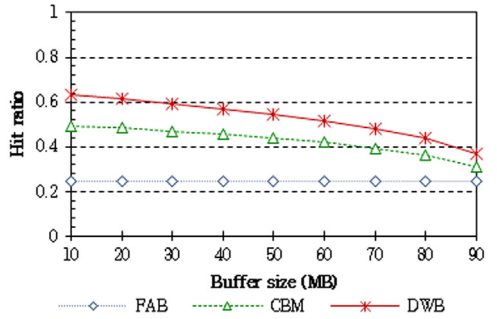

그림 10 NVRAM 비율 변화에 따른 버퍼 적중률
Fig. 10 Buffer hit ratio according to the NVRAM size

를 생성하여 실험을 진행하였다.

그림 8에서 확인할 수 있듯이 읽기/쓰기가 비슷한 비율로 발생하는 상황에서는 실험에 사용된 버퍼 알고리즘들이 비슷한 버퍼 적중률을 보여주고 있지만, 제안된 버퍼 기법인 DWB는 FAB과 CBM 대비 최대 23.38% 그리고 15.35% 향상되었다. 이와 같이 버퍼 알고리즘에 따라 성능 차이를 크게 보이지 않는 이유는, 그림 9에서 확인할 수 있듯이, Filebench 벤치마크로 생성된 워크로드의 경우 다른 워크로드와 달리 동일한 페이지에 대한 새로운 접근 요청시에 요청 간격이 버퍼 사이즈에 포함되는 비율이 상당히 낮기 때문임을 알 수 있다. 이는 Filebench 벤치마크로 생성된 워크로의 경우 읽기/쓰기 요청이 전체 주소 영역에 걸쳐 고르게 분산된 임의 페이지 접근이라는 것을 의미한다. 이와 같이 페이지 참조가 발생하는 상황에서는 버퍼 알고리즘이 버퍼 적중률에 큰 영향을 끼치지 않음을 알 수 있다.

또한, 우리는 버퍼 사이즈를 고정시킨 상황에서 전체 버퍼에서 차지하는 NVRAM용량의 비율을 변경하면서 버퍼 관리 알고리즘의 성능을 비교하였다. 그림 10은 전체 버퍼 사이즈를 16MB로 고정시키고 전체 버퍼에서 차지하는 NVRAM의 비율을 10%에서 90%로 변경하면서 버퍼 적중률을 측정하였으며, 이를 위해 Financial 워크로드를 사용하였다. CBM과 DWB의 NVRAM은 쓰기 작업에 의해 변경된 페이지만을 저장하기 때문에 읽기 작업의 비율이 높은 Financial 워크로드에서는 NVRAM의 비율이 증가함에 따라 전체적인 버퍼 적중률이 낮아지는 것을 보이고 있다. 하지만, DWB가 항상 높은 버퍼 적중률을 보이는 것을 알 수 있다.

4.2.2 낸드 플래시 메모리 수명 예측

제한된 횟수의 삭제 연산이 수행되는 낸드 플래시 메모리로 구성된 스토리지 디바이스는 낸드 플래시 메모리에서 수행되는 삭제 연산 횟수를 감소시킴으로써 수명 연장이 가능하다. 우리는 SSD 패치[8]가 적용된 DiskSim 시뮬레이터[9]를 이용한 시뮬레이션을 통해 SSD에서 발생할 수 있는 삭제 연산 횟수를 측정하였다.

DWB는 실험에 사용된 버퍼 알고리즘들과 달리 읽기 작업에 의해 참조된 페이지들을 낸드 플래시 메모리로 쓰기를 수행하지 않기 때문에, 읽기 작업의 비율이 높은 Financial 워크로드를 사용했을 경우 최대 48.31% 감소된 삭제 연산 횟수를 그림 11을 통해 확인할 수 있다.

그림 12를 통해 쓰기 작업이 높은 비율로 발생하는 CAMWEBDEV 워크로드를 확인해보면 CBM이 다른 버퍼 알고리즘에 비해서 높은 삭제 횟수를 보이고 있다. 적은 양의 NVRAM 만을 버퍼로 사용하는 CBM은 다른 버퍼 알고리즘들에 비해서 쓰기에 의해 참조되는 페이지들이 재참조되기 전에 버퍼에서 낸드 플래시 메모리로 쓰기가 지속적으로 발생하여 높은 삭제 연산 횟수를 보이고 있다. DWB는 쓰기의 비율이 높은 워크로드에서도 최대 56.66% 감소된 삭제 연산 횟수를 보여주었다.

다른 워크로드에 비해 페이지에 대한 접근이 분산된 Filebench 워크로드에서도 DWB를 버퍼 알고리즘으로 적용한 경우에도 그림 13과 같이 최대 20.26% 감소된 삭제 연산 횟수를 보여줬다.

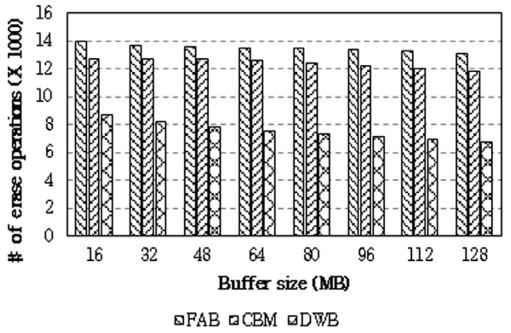

그림 11 Financial 워크로드의 지우기 작업 횟수
Fig. 11 The number of erase operations for Financial workload

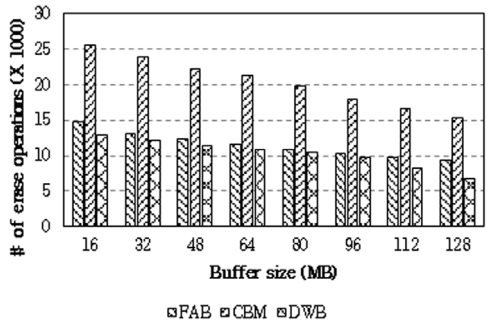

그림 12 CAMWEBDEV 워크로드의 지우기 작업 횟수
Fig. 12 The number of erase operations for CAMWEBDEV workload

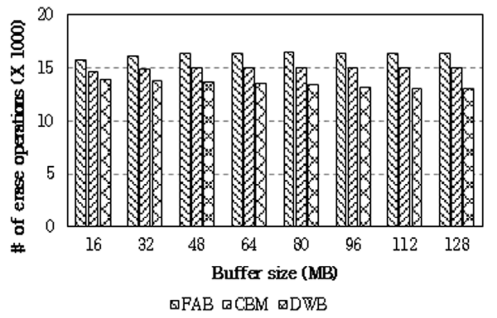

그림 13 Filebench 벤치마크 워크로드 지우기 작업 횟수
Fig. 13 The number of erase operations for Filebench benchmark workload

5. 결 론

기술발전으로 NVRAM의 상품화가 진행됨에 따라 기존의 SSD의 버퍼에도 NVRAM이 적용이 예상된다. 하지만, 아직까지 NVRAM은 기존에 버퍼로 사용되는 DRAM을 모두 대체하기에는 비용/가격 적인 측면에서 비효율적이다.

본 논문에서는 소량의 NVRAM을 적용한 새로운 구조의 버퍼 구조를 소개하며, 이를 기반으로 새로운 버퍼 알고리즘을 제안한다. 그리고 버퍼 적중률과 낸드 플래시 메모리의 삭제 연산 횟수를 기존의 버퍼 알고리즘들과 비교하였다. 제안된 버퍼 알고리즘은 전반적으로 향상된 버퍼 적중률을 보여주었으며, 시뮬레이션을 통해 낸드 플래시 메모리에서의 삭제 연산의 수행 횟수가 감소됨을 확인하였다.

향후 연구로는 바이트 단위로 접근이 가능한 NVRAM의 특성을 고려하여 버퍼 알고리즘에 대해 연구를 진행할 예정이다.

References

[1] H. Jo, J.-U. Kang, S.-Y. Park, J.-S. Kim, and J. Lee, "FAB: Flash Aware Buffer Management Policy for Portable Media Players," *IEEE Trans. On Consumer Electronics*, Vol. 52, No. 2, pp. 485-493, May. 2006.

[2] B. Debnath, S. Subramanya, D. Du, and D. J. Lilja, "Large Block CLOCK (LB-CLOCK): A Write Caching Algorithm for Solid State Disks," *Proc. of IEEE International Symposium on 17th Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pp. 1-9, 2009.

[3] G. Wu, B. Eckart, and X. He, "BPAC: An Adaptive Write Buffer Management Scheme for Flash-based Solid State Drives," *Proc. of IEEE Symposium on 26th Mass Storage Systems and Technologies*, pp. 1-6, 2010.

[4] H. Kim and S. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," *Proc. of 6th USENIX Conference on File and Storage Technologies*, pp. 1-14, 2008.

[5] J. Hu, H. Jiang, L. Tian, and L. Xu, "PUD-LRU: An Erase-Efficient Write Buffer Management Algorithm for Flash Memory SSD," *Proc. of IEEE International Symposium on 18th Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pp. 69-78, 2010.

[6] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," *Proc. of VLDB*, pp. 439-450, 1994.

[7] Q. Wei, C. Chen, and J. Yang, "CBM: A Cooperative Buffer Management for SSD," *Proc. of IEEE Symposium on 30th Mass Storage Systems and Technologies*, pp. 1-12, 2014.

[8] N. Agrwal, V. Prabhakaran, T. Wobber, J.D. Davis, M.S. Manasse, and R. Panigrahy, "Design Tradeoffs for SSD Performance," *Proc. of USENIX Annual Technical Conference*, pp. 57-70, 2008.

[9] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. Ganger, DiskSim 4.0 [Online]. Available: http://www.pdl.cmu.edu/DiskSim (downloaded 2015, Mar. 10)

[10] OLTP Application I/O Trace [Online]. Available: http://traces.cs.umass.edu/index.php/Storage/Storage (downloaded 2015. Mar. 10)

[11] MSR Cambridge Traces [Online]. Available: http://iotta.snia.org/traces/list/BlockIO (downloaded 2015, Mar. 10)

[12] Filebench benchmark [Online]. Available: http://filebench.sourceforge.net/ (downloaded 2015, Mar. 10)

한 세 준

2005 충남대학교 학사. 2014년 성균관대학교 IT융합학과 석사. 관심분야는 운영체제, 스토리지 시스템, 가상화 기술

강 동 현

2007년 한국산업기술대학교 컴퓨터공학과 학사. 2010년 성균관대학교 전자전기컴퓨터공학과 석사. 2013년~현재 성균관대학교 전자전기컴퓨터공학과 박사과정 관심분야는 스토리지 시스템, 운영체제

엄 영 익

1983년 서울대학교 계산통계학과 학사 1985년 서울대학교 전산과학과 석사. 1991년 서울대학교 전산과학과 박사. 1993년~현재 성균관대학교 정보통신대학 교수. 관심분야는 시스템 소프트웨어, 운영체제, 가상화, 파일 시스템