Journal of KIIT. Vol. 16, No. 4, pp. 129-139, Apr. 30, 2016. pISSN 1598-8619, eISSN 2093-7571 129
http://dx.doi.org/10.14801/jkiit.2016.14.4.129

플래시 메모리 기반의 데이터베이스 시스템을 위한 BS-CLOCK 버퍼 교체 알고리즘

이미경*, 이두기*, 박상현**

BS-CLOCK: A Buffer Replacement Algorithm for Flash Memory-based Database Systems

Mi-Kyung Lee*, Du-Ki Lee*, and Sang-Hyun Park**

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2015R1A2A1A05001845).

요 약

최근 플래시 메모리의 특성을 이용한 버퍼 교체 알고리즘들이 계속해서 연구되고 있다. 이러한 연구들 중에서도 Spatial Clock 알고리즘은 희생자 페이지를 선정할 때 시간적 지역성뿐만 아니라 공간적 지역성도 고려한다는 점에서 좋은 성능을 보인다. 본 논문에서는 Spatial Clock 알고리즘과 본 논문에서 입증한 실험 결과를 활용하여 플래시 메모리 기반의 데이터베이스 시스템을 위한 BS-CLOCK 버퍼 교체 알고리즘을 제안한다. BS-CLOCK 알고리즘은 버퍼 교체 시 공간적으로 인접해있는 더티 페이지들을 플래시 메모리로 미리 써서 수행시간을 감소시키고, 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수를 제한하여 불필요한 쓰기 연산을 방지한다. 그 결과, BS-CLOCK 알고리즘은 Spatial Clock 알고리즘에 비해 캐시 적중률의 손실이 거의 없이 최대 88%의 수행시간 감소율을 보였다.

Abstract

Buffer replacement algorithms using the characteristics of flash memory is being studied recently. Among these studies, Spatial Clock shows good performance in that it considers spatial locality as well as temporal locality on victim selection. In this paper, we suggest a new buffer replacement algorithm for flash-based database systems, called BS-CLOCK, by using Spatial Clock and an experimental result in this paper. BS-CLOCK can reduce I/O time by writing dirty pages with a sequential pattern to flash memory ahead and avoid unnecessary write operations by limiting the maximum number of pages that could be written to flash memory. Experimental results show that BS-CLOCK reduces I/O time by up to 88% with minimal loss in cache hit ratio, compared to Spatial Clock.

Keywords
buffer replacement algorithm, page replacement, virtual paging, flash memory, trace-driven simulation

* 연세대학교 컴퓨터과학과
** 연세대학교 컴퓨터과학과 교수(교신저자)
· 접 수 일: 2015년 11월 19일
· 수정완료일: 2016년 02월 14일
· 게재확정일: 2016년 02월 17일
· Received: Nov. 19, 2015, Revised: Feb. 14, 2016, Accepted: Feb. 17, 2016
· Corresponding Author: Sang-Hyun Park
Dept of Computer Science, Yonsei University, Sinchon-dong, Seodaemun-gu Seoul, 120-749, Korea
Tel.: +82-2-2123-5714, Email: sanghyun@cs.yonsei.ac.kr

Ⅰ. 서 론

하드 디스크나 플래시 저장장치와 같은 비휘발성 저장장치를 보조기억장치로 사용한 데이터베이스 시스템에서 성능 부족이나 수명 문제와 같은 보조기억장치의 물리적 한계점이 시스템의 성능 저하로 연결되는 것은 자명한 일이다. 이러한 문제를 해결하기 위하여 메인 메모리를 버퍼로 사용한 버퍼 교체 알고리즘에 관한 연구가 계속해서 수행되어 왔다[1]-[3].

버퍼 교체 알고리즘은 버퍼에 새로운 페이지를 적재할 빈 프레임이 없을 때 그림 1과 같이 동작한다. 즉, 버퍼에 있는 페이지들 사이에서 희생자 페이지를 선정한 후 버퍼에서 내보내고, 그 자리에 새로운 페이지를 적재할 수 있도록 한다. 버퍼 교체 알고리즘에서 가장 고려해야 될 점은 희생자 페이지로서 차후에 가장 참조 되지 않을 것 같은 페이지를 얼마나 정확하게 예측하는가이다. 이를 위해 대부분의 버퍼 교체 알고리즘들은 시간적 지역성(Temporal Locality)을 고려하여 희생자 페이지를 결정한다.

그러나 플래시 메모리[4]가 등장한 이후 플래시 메모리의 장점들을 이용한 버퍼 교체 알고리즘에 관한 연구들이 활발히 진행되면서, 이전처럼 시간적 지역성만을 바탕으로 한 기존의 버퍼 교체 알고리즘들이 플래시 메모리를 기반으로 한 시스템의 성능을 저하시킬 수 있음이 밝혀졌다[5]-[9]. 이러한 현상은 플래시 메모리는 제자리 덮어쓰기(In-Place Update)가 불가능하여 쓰기 연산 이전에 소거 작업이 필요하다는 점과 쓰기 연산의 성능이 읽기 연산에 비해 현저히 좋지 않은 점, 그리고 수명이 제한적인 점 등과 같은 단점들을 보유하고 있기 때문에 나타나는 것으로 보여진다. 이와 같은 플래시 메모리의 단점들로 인한 시스템의 성능 저하를 개선하기 위하여 플래시 메모리가 하드 디스크보다 임의적 읽기(Random Read) 연산의 속도가 굉장히 빠르고 쓰기 연산중에서도 순차적 쓰기(Sequential Write) 연산의 속도가 임의적 쓰기(Random Write) 연산의 속도보다 빠르다는 점을 기반으로 한 버퍼 교체 알고리즘 연구들이 등장하고 있다[10]-[13].

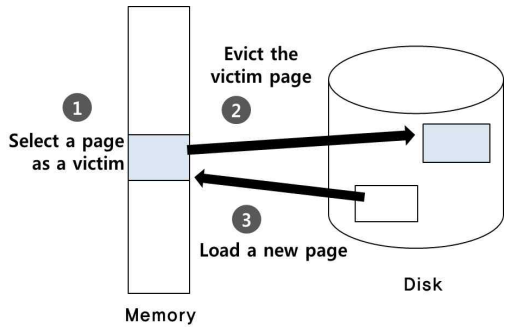


그림 1. 버퍼 교체 알고리즘의 동작 과정
Fig. 1. Mechanism of page replacement algorithm

이처럼 플래시 메모리의 특징들을 고려한 버퍼 교체 알고리즘에 관한 연구들 중 하나인 Spatial Clock 알고리즘[14]은 다른 알고리즘들 사이에서도 성능 면에서 두각을 나타내고 있는데, 이는 Spatial Clock 알고리즘이 희생자 페이지를 결정할 때 시간적 지역성뿐만 아니라 공간적 지역성(Spatial Locality)도 고려하기 때문이다. 다시 말해서 Spatial Clock 알고리즘은 플래시 메모리로 보내지는 쓰기 연산들이 다른 버퍼 교체 알고리즘들보다 순차적으로 정렬되어 있기 때문에 수행시간을 감소시킬 수 있다.

한편, 본 논문에서는 같은 개수의 페이지들을 플래시 메모리로 쓰더라도 페이지들을 여러 개씩 묶어서 쓰는 경우 플래시 메모리의 처리량을 높일 수 있다는 것을 입증하였다. 표 1은 입증을 위해 실험에 사용된 플래시 저장 장치들[15][16]을 나타낸다.

표 1. 실험에 사용된 플래시 저장 장치
Table 1. Flash storages used in experiments

저장 장치 유형	메모리 칩 제조사	클래스	용량 (GB)
microSDHC	Samsung	10	16
	Sandisk	10	16

그림 2에서 보는 것과 같이 총 64MB를 마이크로 SD 카드로 쓸 때 1MB씩 쓰는 경우의 수행시간이 4KB씩 쓰는 경우의 수행시간보다 삼성 마이크로 SD 카드(Samsung‘s Micro SD Card)와 샌디스크(Sandisk‘s) 마이크로 SD 카드 각각에서 35%, 77.5% 감소된 것을 확인할 수 있다.

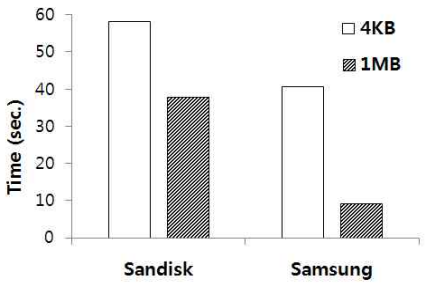


그림 2. 플래시 메모리로 쓰는 양에 따른 I/O 수행시간
Fig. 2. I/O time by the amount written to flash memory

실제 하드 디스크에서 동일한 환경으로 실험했을 때 수행시간의 차이가 없었다는 결과를 감안한다면, 플래시 메모리에 여러 개의 페이지들을 한꺼번에 쓰는 것이 플래시 메모리의 특징을 더 잘 활용 한다는 것을 알 수 있다.

이와 같은 실험 결과를 바탕으로 본 논문에서는 BS-CLOCK(Bulk-write Spatial Clock) 버퍼 교체 알고리즘을 제안하고자 한다. BS-CLOCK 알고리즘은 Spatial Clock 알고리즘을 확장한 알고리즘으로, 공간적으로 인접해있고 본 논문에서 제시한 조건에 부합되는 페이지들을 희생자 페이지들로 선정하고 선정된 희생자 페이지들을 플래시 저장장치로 미리 써서 알고리즘의 성능을 향상시킨다.

논문의 구성은 다음과 같다. 2장에서는 기존 저장 장치 시스템과 플래시 메모리를 기반으로 한 저장 장치 시스템에서 각각 동작하는 대표적인 알고리즘들을 소개하고, BS-CLOCK 알고리즘의 토대가 되는 Spatial Clock 알고리즘에 대해 설명한다. 3장에서는 본 논문에서 제안하는 BS-CLOCK 알고리즘에 대해 상세히 살펴보고, 4장에서 BS-CLOCK 알고리즘의 우수한 성능을 Spatial Clock 알고리즘과 비교하여 살펴본다. 마지막으로 5장에서 결론과 차후에 다룰 만한 연구를 제시한다.

II. 관련 연구

하드 디스크를 기반으로 한 기존의 버퍼 교체 알고리즘들은 플래시 메모리의 독특한 특성들 때문에 플래시 메모리가 사용된 저장 장치 시스템에서 다소 효율적이지 못하다. 따라서 하드 디스크의 물리적 특성을 고려한 버퍼 교체 알고리즘들을 플래시 메모리의 특성에 맞게 개선하는 연구가 진행되어 왔다. 이번 장에서는 대표적으로 사용된 하드 디스크 기반의 버퍼 교체 알고리즘들과 이를 개선하여 플래시 메모리에 최적화한 버퍼 교체 알고리즘들에 대해 설명하고자 한다.

LRU(Least Recently Used) 알고리즘은 페이지 참조에 대하여 시간적 지역성을 토대로 동작한다[17][18]. 즉, 최근에 참조된 페이지는 가까운 미래에 다시 참조될 가능성이 높다는 것을 전제로 희생자 페이지를 선정한다. 따라서 LRU 알고리즘은 버퍼 내의 페이지들이 참조된 시간을 기준으로 정렬된 상태를 유지해야 한다. 또한 페이지 교체 시 오랫동안 참조되지 않은 페이지를 희생자 페이지로 선정해야 한다. 그러나 LRU 알고리즘은 페이지들을 항상 참조 시간 순서대로 유지해야 되기 때문에 구현에 있어서 오버헤드가 크다.

이를 개선하기 위하여 등장한 CLOCK 알고리즘은 LRU 알고리즘과 근접하다[19]. 그러나 LRU 알고리즘과 같이 버퍼 내의 모든 페이지들을 참조된 시간 순서대로 유지하지 않고, 페이지마다 존재하는 참조 비트(Reference bit)를 가지고 희생자 페이지를 결정하기 때문에 페이지들이 항상 정렬된 상태를 유지하지 않아도 된다. 그 대신 CLOCK 알고리즘은 페이지들을 순환적 형태의 리스트로 유지한다. CLOCK 알고리즘은 페이지가 참조될 때 하드웨어에 의해 페이지의 참조 비트가 1로 초기화된다. 그러나 페이지 교체 시, CLOCK 포인터를 사용하영 페이지들의 참조 비트를 검사한다. 참조 비트가 0인 경우 CLOCK 알고리즘은 이 페이지를 희생자 페이지로 간주하여 버퍼에서 삭제한다. 참조 비트가 1인 경우 CLOCK 포인터가 참조 비트가 0인 페이지를 가리킬 때까지 페이지들의 참조 비트들을 0으로 변경한다. 이와 같은 동작 방식은 페이지들이 참조된 순서대로 희생자 페이지가 결정되지 않지만 참조 비트를 사용함으로써 LRU 알고리즘보다 구현을 위한 오버헤드가 적고 LRU 알고리즘과 근접한 성능을 내기 때문에 가상 메모리 시스템에서 많이 사용된다.

CFLRU(Clean-First Least Recently Used) 알고리즘은 LRU 알고리즘을 플래시 메모리에 맞게 개선한 알고리즘으로, 플래시 메모리의 쓰기 연산의 비용이

읽기 연산의 비용보다 훨씬 크다는 점을 고려하여 설계되었다[20]. CFLRU 알고리즘은 LRU 리스트를 워킹 영역(Working Region)과 클린-퍼스트 영역(Clean-First Region)으로 구분지어 동작한다. LRU 리스트의 헤드(Head) 부분에 위치한 워킹 영역은 최근에 참조된 페이지들로 구성되며 따라서 주로 이 구간에서 캐시 적중(Cache Hit)이 발생된다. 리스트의 꼬리(Tail)에 위치한 클린-퍼스트 영역은 오랜 기간 동안 참조되지 않은 페이지들이 모여 있고, 버퍼에 있던 페이지가 새로운 페이지로 교체될 때 이 구간에 있던 페이지들 사이에서 희생자 페이지가 결정된다. CFLRU 알고리즘이 교체할 희생자 페이지를 선택하는 과정은 다음과 같다. 먼저 CFLRU 알고리즘은 버퍼에 새롭게 참조된 페이지가 삽입될 비어있는 프레임이 없는 경우, 플래시 메모리의 쓰기 연산의 비용을 절감하기 위해 클린-퍼스트 영역에 있던 클린 페이지(Clean Page)를 우선적으로 희생자 페이지로 선정하여 버퍼에서 삭제한다. 만약 이 구간에 클린 페이지가 없으면 클린-퍼스트 영역에 존재하는 더티 페이지(Dirty Page)를 희생자 페이지로 간주한다. 클린-퍼스트 영역의 크기인 윈도우 사이즈(Window Size)는 미리 결정된 워크로드에서 여러 번 실험을 수행한 후 가장 잘 나온 평균값으로 결정되거나 주기적으로 워크로드의 연산들에 대한 정보를 얻어 동적으로 결정하는 방법이 있다. 일반적으로 윈도우 사이즈 값이 커질수록 캐시 적중률 또한 감소되지만 버퍼에서 내보내진 더티 페이지 개수 역시 감소되므로 플래시 메모리의 쓰기 연산의 횟수가 감소될 수 있다.

FAB(Flash-Aware-Buffer Management) 알고리즘 역시 LRU 알고리즘을 개선한 알고리즘으로, LRU 알고리즘이나 CFLRU 알고리즘과 달리 버퍼 내 페이지들을 블록 단위로 묶어 관리한다[21]. 또한 FAB 알고리즘은 시간적 지역성보다 플래시 메모리의 순차적 쓰기 성능을 중요시 여기기 때문에 페이지 교체 시, 가장 많이 순차적 쓰기를 생성할 것 같은 블록을 희생자로 선정하여 플러시(Flush)한다. 이 알고리즘은 '큰 순차 쓰기'를 시간적 지역성보다 중요시하므로 다른 버퍼 교체 알고리즘에 비해 캐시 적중률이 저하되는 문제가 발생할 수 있다.

Spatial Clock 알고리즘은 CLOCK 알고리즘을 확장한 알고리즘으로, CLOCK 알고리즘과 같이 버퍼에 있는 페이지들이 순환적 형태의 리스트로 유지되고 참조 비트를 사용하여 희생자 페이지를 결정한다. 한 가지 CLOCK 알고리즘과 차별화된 점은 버퍼에 있는 페이지들이 논리적 섹터 번호(Logical Sector Number)를 기준으로 정렬된 상태를 유지한다는 점이다. 따라서 페이지 교체가 발생되는 동안 시간적 지역성뿐만 아니라 공간적 지역성(Spatial locality) 역시 고려된다. 그러므로 Spatial Clock 알고리즘은 다른 버퍼 교체 알고리즘에 비해 순차적으로 정렬된 쓰기 연산들을 생성하기 때문에 I/O가 분산되어 있는 워크로드에서 두각을 드러낸다.

III. BS-CLOCK

이번 장은 본 논문에서 제안하는 새로운 버퍼 교체 알고리즘을 소개한다. BS-CLOCK 알고리즘은 Spatial Clock 알고리즘을 확장한 알고리즘으로, 서론에서 제시 했던 실험 결과를 바탕으로 동작한다. 따라서 BS-CLOCK 알고리즘은 버퍼에 있는 페이지들을 논리적 섹터 번호를 기준으로 정렬된 상태를 유지할 수 있도록 하고, 여러 개의 페이지들을 한꺼번에 플래시 메모리로 쓰면 더 효율적이라는 점을 감안하여 희생자로 선정된 여러 개의 페이지들을 한꺼번에 플래시 메모리로 쓰는 것을 허용한다.

그림 3은 BS-CLOCK 알고리즘에서 희생자 페이지를 선정하는 과정을 보여준다. 버퍼에 새롭게 참조된 페이지를 할당 할 빈 프레임이 없을 때, BS-CLOCK 알고리즘은 CLOCK 포인터가 가리키는 페이지의 참조 비트를 검사한다. 참조 비트가 1인 경우, CLOCK 포인터가 참조 비트가 0인 페이지를 가리킬 때까지, 페이지들의 참조 비트들을 0으로 변경한다. 그러나 참조 비트가 0이고 해당 페이지가 클린 페이지인 경우에는 버퍼에서 해당 페이지를 삭제하고, 더티 페이지인 경우에는 해당 페이지의 다음 페이지들 역시 참조 비트가 0이고 해당 페이지와 순차적인 패턴을 이루는 더티 페이지들인지 검사한다. 조건을 만족하면 이 페이지들을 모두 희생자 페이지들로 간주하고 플래시 메모리로 한꺼번에 쓴다. 또한, 캐시 적중률이 저하되는 것을 막기 위하여 선정된 희생자 페이지들 중 본래 CLOCK 포

인터가 가리켰던 첫 번째 페이지만을 버퍼에서 내보내고 나머지 희생자 페이지들은 더티 비트(Dirty Bit)를 이용하여 클린 페이지로 변경한다.

그림 4와 그림 5는 Spatial Clock 알고리즘과 BS-CLOCK 알고리즘이 희생자 페이지를 선정하는 과정을 예시로 보여주고 있다. 두 알고리즘 모두 페이지들이 논리적 섹터 번호 순서대로 이미 정렬되어 있고 논리적 섹터 번호가 16인 페이지부터 검사를 시작한다.

```
Procedure chooseVicitimPage(Page *p, int key)
Input
    p   : page address that should be checked
    key : page number that would be inserted
Output
    page_state: CLEAN or DIRTY
    /*
     Check if dirty pages are in a sequential order
    */
 1: if p->dirtied == 0
 2: then page_state = CLEAN;
 3: else
 4:   page_state = DIRTY;
 5:   nr_pages = 0; i = 0; q = p;
 6:   do
 7:     r = q;
 8:     if nr_pages >= 1
 9:     then array[i++] = r->lsn; endif
10:     q = choose_next_page(q->lsn);
11:     nr_pages++;
12:   while(q->referenced == 0 and
            q->dirtied == 1 and
            q->lsn == r->lsn + 1 and
            i < SEQ_DIRTIED_PAGES)
13: endif
14: evict_page(p->lsn);

    /*
     Turn the state of victim pages into clean
    */
14: if page_state == DIRTY and
       nr_pages >= 2 then
15:   if nr_pages >= SEQ_DIRTIED_PAGES then
16:     for i = 0; i < SEQ_DIRTIED_PAGES; i++
17:       r = search_page(root, array[i]);
18:       r->dirtied = 0;
19:     end for
20:     nr_pages = SEQ_DIRTIED_PAGES;
21:   else
22:     for i = 0; array[i] != -1; i++
23:       r = search_page(root, array[i]);
24:       r->dirtied = 0;
25:     end for
26:   endif
27: endif
```

그림 3. BS-CLOCK 알고리즘의 희생자 페이지 선별 과정
Fig. 3. Victim selection of BS-CLOCK

논리적인 섹터 번호가 16인 페이지는 참조 비트가 0이므로 Spatial Clock 알고리즘에서는 이 페이지를 희생자 페이지로 간주하여 버퍼에서 내보내고 플래시 메모리로 쓰기 연산을 수행한다. 반면 BS-CLOCK 알고리즘에서는 논리적인 섹터 번호가 24인 페이지의 참조 비트와 더티 비트, 순차 패턴을 검사한다. 논리적인 섹터번호가 16인 페이지부터 40인 페이지까지 모두 참조 비트가 0이고 더티 페이지이며 공간적으로 인접해있으므로 이 페이지들을 희생자로 간주한다. 또한 첫 번째 희생자 페이지인 논리적 섹터 번호가 16인 페이지만 버퍼에서 내보내고, 그 외의 논리적 섹터 번호가 24인 페이지부터 40인 페이지까지 모두 클린 페이지로 변경한다.

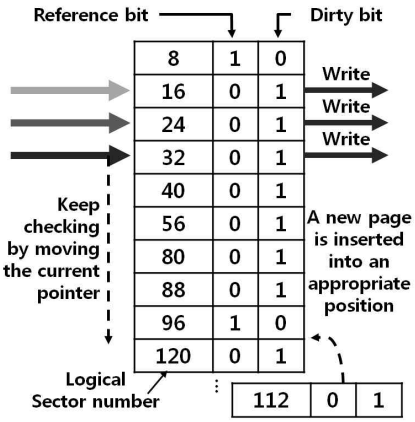


그림 4. Spatial Clock 알고리즘의 희생자 선정 과정
Fig. 4. Victim selection of spatial clock

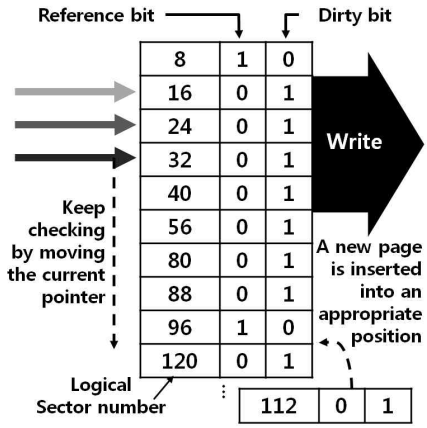


그림 5. BS-CLOCK 알고리즘의 희생자 선정 과정
Fig. 5. Victim selection of BS-CLOCK

이와 같은 방식으로 수행하다 보면 Spatial Clock 알고리즘은 다른 알고리즘에 비해 순차적으로 정렬된 쓰기 연산들을 더 많이 만들어내기 때문에 좀 더 좋은 성능을 보이지만, 이미 대부분이 순차적 쓰기 패턴들로 구성된 워크로드에서는 다소 효율적이지 못하다. 반면 BS-CLOCK 알고리즘은 페이지 교체 시 조건에 만족하는 페이지들을 한꺼번에 플래시 메모리로 쓰기 때문에 순차적 패턴의 워크로드뿐만 아니라 다른 워크로드에서도 수행시간을 크게 감소시킬 수 있다. 또한 여러 개의 페이지들을 플래시 메모리로 썼지만 버퍼에서 한 개의 페이지만 삭제하였기 때문에 캐시 적중률이 크게 감소되고, 나머지 희생자 페이지들의 경우 클린 페이지로 변경하였기 때문에 차후 쓰기 연산의 개수를 줄일 수 있다. 그러나 BS-CLOCK 알고리즘은 플래시 메모리로 여러 개의 페이지들을 한꺼번에 미리 썼기 때문에 불필요한 쓰기 연산들이 발생할 수도 있다. 이와 같은 문제가 BS-CLOCK 알고리즘의 장점들을 뛰어넘지 못하지만, 본 논문에서는 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수를 제한하여 불필요한 연산을 줄이고자 한다. 만약 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수가 너무 커지면 가까운 미래에 다시 참조될 가능성이 높은 페이지들까지 플래시 메모리로 쓰게 되는 불필요한 연산이 발생할 수 있고, 이와 반대로 값이 너무 적어지면 여러 개의 페이지를 한꺼번에 플래시 메모리로 쓰는 방식의 장점을 충분히 활용할 수 없게 된다.

Ⅳ. 성능 평가

이번 장에서는 실험을 통해 플래시 메모리에 최대로 쓸 수 있는 적절한 페이지의 개수를 찾고, 실험을 통해 찾은 최댓값을 적용하여 BS-CLOCK 알고리즘의 성능을 Spatial Clock 알고리즘과 비교하고자 한다. 먼저 실험을 위하여 두 종류의 트레이스(Trace)를 사용하였다.

캐시-이전-트레이스(Before-Cache-Trace)는 구글사의 Android reference phone, Nexus-One with Andorid Open Source Project 2.3.7, Gingerbread와 같은 스마트폰에서 여러 가지 어플리케이션들을 실행시켜 버퍼로 접근되는 I/O 연산들의 집합으로, I/O 연산들은 '연산 종류, 페이지 번호' 순서쌍으로 나타낸다. [22]에서 얻은 캐시-이전-트레이스는 표 2와 같이 웹 브라우징(Web Browsing), 비디오 스트리밍(Video Streaming), 혼합 어플리케이션(Mixed Apps) 워크로드로 분류할 수 있다.

표 2. 워크로드를 구성하는 연산의 개수
Table 2. The number of I/O operations in workloads

워크로드 유형	읽기 연산의 개수	쓰기 연산의 개수	총 개수
웹 브라우징	23,383	47,346	70,729
비디오 스트리밍	67	387,701	387,768
혼합 어플리케이션	134,910	105,796	240,706

웹 브라우징 워크로드는 수 시간 동안 웹 사이트에 접속하거나 웹 사이트에서 파일을 다운받는 등의 작업들을 수행하면서 얻은 트레이스로, 작은 크기의 파일들을 계속해서 읽고 쓰기 때문에 랜덤(Random)한 경향의 읽기 또는 쓰기 연산들로 구성되어 있다. 비디오 스트리밍 워크로드는 수 시간 동안 유튜브(YouTube)와 같은 사이트에서 동영상 서비스를 시청하면서 얻은 트레이스로, 워크로드 중에서도 쓰기 연산의 개수가 가장 많다. 혼합 어플리케이션 워크로드는 스마트폰의 어플리케이션들을 단독으로 실행하거나 동시에 실행시켜 얻은 트레이스다. 이와 같은 캐시-이전-트레이스들을 트레이스-드라이븐 시뮬레이션(Trace-Driven-Simulation)에 돌리면 알고리즘들의 캐시 적중률을 구할 수 있다.

캐시-이후-트레이스(After-Cache-Trace)는 버퍼 교체 알고리즘에 의해 필터링(Filtering)되어 플래시 저장 장치로 접근되는 I/O 연산들의 집합으로서 I/O 연산들은 '연산 종류, 페이지 번호, 플래시 메모리로 쓰는 양'으로 나타낸다. 캐시-이후-트레이스를 통하여 플래시 저장장치에서 실제 I/O 연산들이 수행되는 시간을 측정할 수 있으며 이를 위하여 3.2.0-25 버전의 리눅스 커널에서 O_DIRECT 방식을 사용하였다. 또한 실험을 위하여 사용한 플래시 저장장치들은 서론에서 소개되었던 표 1과 같다.

4.1 플래시 메모리에 최대로 쓸 수 있는 페이지 개수

플래시 메모리에 최대로 쓸 수 있는 페이지의 개수에 대한 적절한 값을 구하기 위해 패이지 한 개의 크기를 4KB로, 버퍼의 크기를 페이지 교체가 빈번히 일어나는 4MB로, 그리고 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수를 표 3과 같이 설정하여 실험을 수행하였다. 표 3은 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수와 페이지 개수에 따른 플래시 메모리에 최대로 쓸 수 있는 양을 보여준다.

그림 6에 따르면 BS-CLOCK 알고리즘은 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수와 관계없이 모든 워크로드에서 캐시 적중률의 변화가 거의 없음을 확인할 수 있다.

표 3. 플래시 메모리로 쓸 수 있는 최대 페이지 개수와 양
Table 3. The maximum number of pages and amount that could be written to flash memory

페이지 개수	페이지 개수에 따른 양 (= 페이지 개수 * 페이지 한 개의 크기)
2	8K
4	16K
8	32K
16	64K
32	128K
64	256K
128	512K
256	1M
512	2M
1024	4M

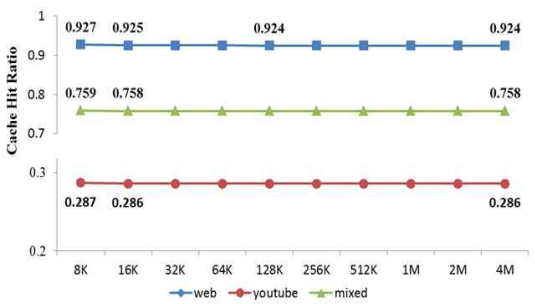
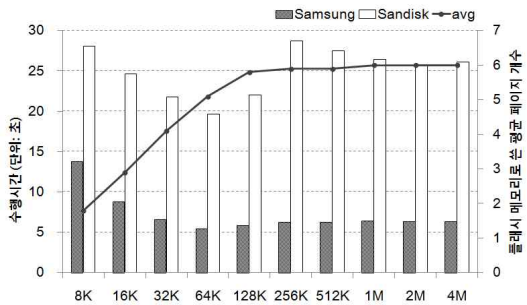
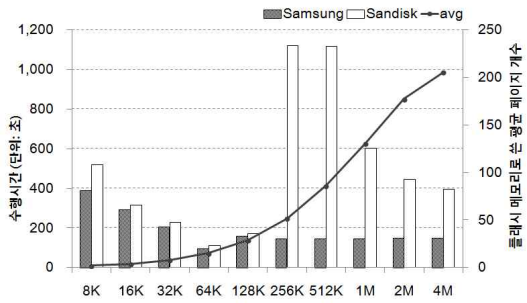


그림 6. 플래시 메모리에 최대로 쓸 수 있는 페이지 개수에 따른 캐시 적중률
Fig. 6. Cache hit ratio by the maximum number of pages that could be written to flash memory

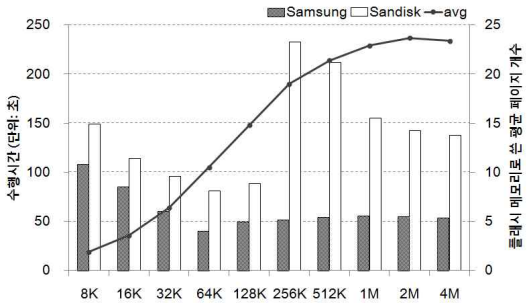
특히 플래시 메모리에 최대로 쓸 수 있는 양이 4MB일 때에는 8KB일 때보다 캐시 적중률이 0.3% 밖에 감소되지 않았다. 이와 같은 결과는 BS-CLOCK 알고리즘이 페이지 교체 시 희생자 페이지들 중 첫 번째 희생자 페이지만 버퍼에서 내보내기 때문이다. 따라서 그림 6을 통해 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수를 구하고자 할 때 캐시 적중률은 크게 고려할 필요가 없음을 알 수 있다.



(a) 워크로드 1: 웹 브라우징
(a) Workload 1: Web browsing



(b) 워크로드 2: 비디오 스트리밍
(b) Workload 2: Video streaming



(c) 워크로드 3: 혼합 어플리케이션
(c) Workload 3: Mixed apps

그림 7. 플래시 저장 장치에서 측정한 I/O 수행시간과 플래시 메모리로 쓴 평균 페이지 개수
Fig. 7. I/O time measured at flash storage and the average number of pages written to flash memory

그림 7은 그림 6과 같이 BS-CLOCK 알고리즘에서 플래시 메모리에 최대로 쓸 수 있는 양을 8KB에서 4MB까지 변화시키면서 측정한 플래시 메모리로 실제 쓰게 되는 평균 페이지 개수와 어플리케이션 레벨에서 측정한 실제 플래시 저장장치에서의 수행시간을 나타낸 그래프이다. 그림 7에서 보는 것과 같이 전반적으로 플래시 메모리로 최대 쓸 수 있는 양이 커질수록 수행시간은 감소된다. 이와 같은 결과는 플래시 메모리가 여러 개의 페이지들을 한꺼번에 쓰는 것이 더 효율적이기 때문에 발생된다. 또한 플래시 메모리로 최대 쓸 수 있는 양이 128KB에서 256KB인 구간에서는 수행시간이 오히려 증가된 것은 플래시 메모리로 최대 쓸 수 있는 양이 커질수록 평균 페이지 개수가 비례하여 증가하지 않기 때문에 나타나는 현상이다. 그러나 플래시 메모리로 최대 쓸 수 있는 양이 128KB에서 256KB인 구간에서 마이크로 SD 카드의 종류에 따른 수행시간 증가량의 격차가 큰지는 좀 더 연구해볼 필요가 있다.

4.2 성능 비교

이번 장에서는 4.1장에서 살펴본 실험 결과들을 참고로 플래시 메모리에 최대로 쓸 수 있는 양을 64KB로 즉, 플래시 메모리에 최대로 쓸 수 있는 페이지의 개수를 16개로 고정하여 실험을 수행하였다.

표 4는 BS-CLOCK 알고리즘과 Spatial Clock 알고리즘이 플래시 메모리로 쓴 총 페이지의 개수를 나타낸다. 표 4에서 보는 것과 같이 BS-CLOCK 알고리즘은 Spatial Clock 알고리즘보다 플래시 메모리로 더 많은 페이지를 쓰고 있는 것을 알 수 있다.

그럼에도 불구하고 그림 8과 그림 9에서 보는 것과 같이 플래시 메모리에서 수행된 I/O 연산의 수행시간을 측정했을 때 BS-CLOCK 알고리즘의 수행시간이 Spatial Clock 알고리즘의 수행 시간보다 확연히 낮은 것을 알 수 있다. 특히 삼성의 마이크로 SD 카드에서는 BS-CLOCK 알고리즘의 수행시간이 Spatial Clock 알고리즘에 비해 웹 브라우징 워크로드, 비디오 스트리밍 워크로드, 혼합 어플리케이션 워크로드에서 각각 60.8%, 85.6%, 67.6%로 감소된 것을 알 수 있다. 샌디스크의 마이크로 SD 카드에서도 BS-CLOCK 알고리즘의 수행시간이 Spatial Clock 알고리즘보다 동일한 워크로드에서 각각 25.9%, 88.7%, 65.3% 감소된 것을 알 수 있다. 이와 같이 BS-CLOCK 알고리즘이 플래시 메모리로 쓰는 총 페이지 개수가 Spatial Clock 알고리즘보다 훨씬 많음에도 불구하고 수행시간이 확연히 감소된 것은 BS-CLOCK 알고리즘이 플래시 메모리가 순차적 쓰기 성능이 좋고 여러 개의 페이지들을 한 번에 플래시 메모리로 쓰는 것이 페이지 한 개씩 쓰는 것보다 훨씬 더 효율적임을 잘 활용했기 때문이다.

표 4. 플래시 메모리로 쓴 총 페이지 개수
Table 4. The total number of pages written to flash memory

워크로드 유형	BS-CLOCK	Spatial Clock
웹 브라우징	5,391	3,856
비디오 스트리밍	275,744	275,557
혼합 어플리케이션	44,632	43,921

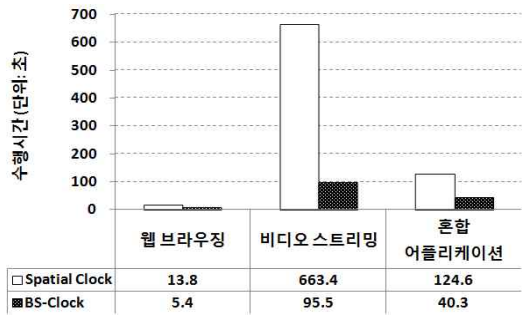


그림 8. 삼성 마이크로 SD 카드에서 측정한 I/O 수행시간
Fig. 8. /O time measured at Samsung's microSD card

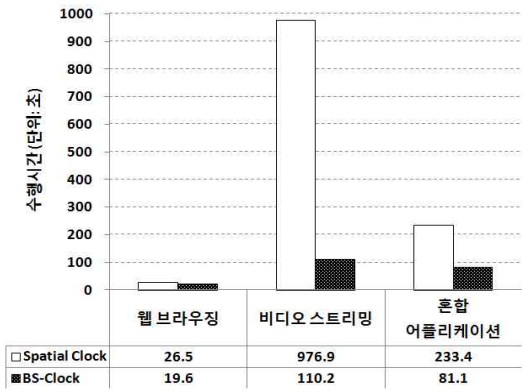


그림 9. 샌디스크 마이크로 SD 카드에서 측정한 I/O 수행시간
Fig. 9. /O time measured at Sandisk's microSD card

표 5. 캐시 적중률 비교
Table 5. Cache hit ratio comparison

워크로드 유형	BS-CLOCK	Spatial Clock	캐시 적중률 감소율(%)
웹 브라우징	0.925	0.93	0.54
비디오 스트리밍	0.286	0.287	0.35
혼합 어플리케이션	0.758	0.759	0.13

또한 표 5에 따르면 BS-CLOCK 알고리즘의 캐시 적중률은 Spatial Clock 알고리즘에 비해 약 0.54%, 0.35%, 0.13%밖에 감소되지 않은 것을 확인할 수 있다. 이와 같은 결과는 BS-CLOCK 알고리즘이 수행시간을 확연히 감소시키는 성능 향상을 얻었음에도 불구하고 캐시 적중률을 크게 저하시키지 않았다는 것을 알 수 보여준다.

V. 결론 및 향후 과제

버퍼 교체 알고리즘이 플래시 기반의 저장 장치 시스템 상에서 적절한 효율성을 내기 위해서는 플래시 메모리의 독특한 특징들을 잘 고려해야 한다. 또한 플래시 메모리는 이미 자주 쓰이는 저장 장치 메모리로 자리매김하였으므로 하드 디스크 기반의 기존 버퍼 교체 알고리즘들을 개선해야 할 필요가 있다. 본 논문에서는 플래시 메모리의 특징들을 고려하여 플래시 메모리에 최적화된 새로운 버퍼 교체 알고리즘을 소개하고 있다. BS-CLOCK 알고리즘은 Spatial Clock 알고리즘을 확장한 알고리즘으로, 버퍼에 새롭게 참조된 페이지를 할당할 빈 프레임이 없는 경우, 참조 비트가 0이고 순차적인 패턴을 가진 더티 페이지들을 희생자 페이지로 간주한다. 또한 선정된 희생자 페이지들은 플래시 저장 장치로 한 번에 쓴 다음, 첫 번째 희생자 페이지만 버퍼에서 삭제하기 때문에 캐시 적중률의 저하를 걱정할 필요가 없다. 또한 첫 번째 희생자 페이지를 제외한 나머지 희생자 페이지들을 클린 페이지로 변경하기 때문에 차후에 발생할 쓰기 연산의 횟수를 감소시키려 한다. 더 나아가 플래시 메모리로 최대 쓸 수 있는 양을 제한하여 불필요한 쓰기 연산을 자제한다. 특히 BS-CLOCK 알고리즘은 실험에 사용된 두 마이크로 SD 카드에서 측정한 평균 수행시간 감소율이 Spatial Clock 알고리즘보다 웹 브라우징 워크로드, 비디오 스트리밍 워크로드, 혼합 어플리케이션 워크로드에서 각각 43.4%, 87.2%, 66.4%인 것을 알 수 있었다. 버퍼 교체 알고리즘에서 수행시간과 캐시 적중률 모두 향상시키기는 어렵지만 BS-CLOCK 알고리즘은 캐시 적중률의 손실을 최소화하면서도 수행시간을 크게 줄였다.

추후 연구로는 BS-CLOCK 알고리즘을 다양한 플래시 저장 장치에서 실험하여 성능의 우수성을 더욱 견고히 입증하는 것이고, 플래시 메모리로 최대 쓸 수 있는 양가 256KB 구간인 곳에 수행시간이 갑자기 증가한 원인을 밝혀내는 것이다. 이러한 점들을 밝혀내기 위한 연구를 본 논문의 향후 과제로 한다.

References

[1] James L. Peterson and Abraham Silberschatz, "*Operating system concepts (2nd ed.)*", Addison-Wesley Longman Publishing Company, pp. 201-256, Jan. 1985.

[2] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum, "The LRU-K page replacement algorithm for database disk buffering", Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 297-306, June 1993.

[3] Song Jiang, Feng Chen, and Xiaodong Zhang, "CLOCK-Pro: An Effective Improvement of the CLOCK Replacement", Proceedings of USENIX Annual Technical Conference, pp. 323-336, Feb. 2005.

[4] Doo-Hwan Yoo and Il-Hoon Shin, "Implementing Greedy Replacement Scheme using Multiple List for Page Mapping Scheme", Journal of KIIT, Vol. 9, No. 6, pp. 17-23, June 2011.

[5] Song Jiang, Xiaoning Ding, and Feng Chen, "DULO: An Effective Buffer Cache Management Scheme to Exploit Both Temporal and Spatial Locality", Proceedings of the 4th USENIX

Conference on File and Storage Technologies (FAST), pp. 101-114, Dec. 2005.

[6] Baichuan Shen, Xin Jin, Yong Ho Song, and Sang Sun Lee, "APRA: Adaptive Page Replacement Algorithm for NAND Flash Memory Storages", Proceedings of Computer Science-Technology and Applications (IFCSTA), pp. 11-14, Dec. 2009.

[7] Ioannis Koltsidas and Stratis D. Viglas, "Data Management Over Flash memory", Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 1209-1212, June 2011.

[8] Yun-Seok Yoo, Hyejeong Lee, Yeonseung Ryu, and Hyokyung Bahn, "Page replacement algorithms for NAND flash memory storages", Proceedings of the International Conference on Computational Science and its Applications, pp. 201-212, Aug. 2007.

[9] Hung-Wei Tseng, Han-Lin Li, and Chia-Lin Yang, "An Energy-Efficient Virtual Memory System with Flash Memory as the Secondary Storage", Proceedings of the International Symposium on Low Power Electronics and Design, pp. 418-423, Oct. 2006.

[10] Bo-Kyeong Kim and Dong-Ho Lee, "LSF: a new buffer replacement scheme for flash memory-based portable media players", *Journal of IEEE Transactions on Consumer Electronics,* Vol. 59, No. 1, pp. 130-135. Apr. 2013.

[11] Dong Hyun Kang, Changwoo Min, and Young Ik Eom, "TS-CLOCK: temporal and spatial locality aware buffer replacement algorithm for NAND flash storages", Proceedings of the ACM international conference on Measurement and modeling of computer systems (SIGMETRICS), pp. 581-582, June 2014.

[12] Sung Kyu Park, Youngwoo Park, Gyudong Shim, and Kyu Ho Park, "CAVE: channel-aware buffer management scheme for solid state disk", Proceedings of the ACM Symposium on Applied Computing, pp. 346-353, Mar. 2011.

[13] Xufeng Guo, Jianfeng Tan, and Yuping Wang, "PAB: parallelism-aware buffer management scheme for NAND-based SSDs", Proceedings of the 21st International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 101-110, Aug. 2013.

[14] Hyojun Kim, Moonkyung Ryu, and Umakishore Ramachandran, "What is a good buffer cache replacement scheme for mobile flash storage?", Proceedings of the 12th ACM SIGMETRICS/ PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems (SIGMETRICS), pp. 235-246, June 2012.

[15] http://www.samsung.com/sec/consumer/it/storage/ memory/MB-MP16DB/KR, [Accessed: Jan. 10, 2016].

[16] https://www.sandisk.com/home/memory-cards, [Accessed: Jan. 10, 2016].

[17] L. A. BELADY, "A study of replacement algorithms for virtual storage computers", IBM Systems Journal, Vol. 5, No. 2, pp. 78-101, Apr. 2010.

[18] Mr. C. C. Kavar and Mr. S. S. Parmar, "Performance Analysis of LRU Page Replacement Algorithm with Reference to different Data Structure", Journal of Engineering Research and Applications (IJERA), Vol. 3, No. 1, pp. 2070-2076, Feb. 2013.

[19] Victor F. Nicola, Asit Dan, and Daniel M. Dias, "Analysis of the generalized clock buffer replacement scheme for database transaction processing", Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, pp. 35-46, June 1992.

[20] Seon-yeong Park, Dawoon Jung, Jeong-uk Kang,

Jin-soo Kim, and Joonwon Lee, "CFLRU: A Replacement Algorithm for Flash memory", Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems (CASES), pp. 234-241, Oct. 2006.

[21] Heeseung Jo, Jeong-Uk Kang, Seon-Yeong Park, Jin-Soo Kim, and Joonwon Lee, "FAB: Flash-aware Buffer management policy for portable media players", *Journal of IEEE Transactions on Consumer Electronics*, Vol. 52, No. 2, pp. 485-493, May 2006.

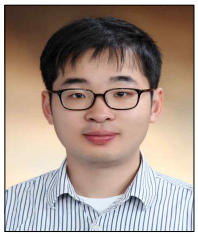
[22] https://wiki.cc.gatech.edu/epl/index.php/S-Clock, [Accessed: Dec. 22, 2015].

저자소개

이 미 경 (Mi-Kyung Lee)

2014년 2월 : 동덕여자대학교 컴퓨터과학과 졸업(학사)
2014년 3월 ~ 현재 : 연세대학교 컴퓨터과학과 석사과정
관심분야 : 데이터베이스, 버퍼 교체 알고리즘, 플래시 메모리

이 두 기 (Du-Ki Lee)



2003년 : 포항공과대학교 컴퓨터공학과 졸업(학사)
2005년 : 포항공과대학교 컴퓨터공학과 졸업(석사)
2013년 ~ 현재 : 연세대학교 컴퓨터과학과 박사과정
관심분야 : 데이터베이스, 플래시 메모리

박 상 현 (Sang-Hyun Park)

1989년 : 서울대학교 컴퓨터공학과 졸업(학사)
1991년 : 서울대학교 대학원 컴퓨터공학과 졸업(공학석사)
2001년 : UCLA 대학원 컴퓨터과학과 졸업(공학박사)
1991년 ~ 1996년 : 대우통신 연구원
2001년 ~ 2002년 : IBM T. J. Watson Research Center Post-Doctoral Fellow
2002년 ~ 2003년 : 포항공과대학교 컴퓨터공학과 조교수
2003년 ~ 2006년 : 연세대학교 컴퓨터과학과 조교수
2006년 ~ 2011년 : 연세대학교 컴퓨터과학과 부교수
2011년 ~ 현재 : 연세대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 데이터마이닝, 바이오인포매틱스, 적응적 저장장치 시스템, 플래시 메모리 인덱스, SSD