모바일 플래시 저장장치를 위한 SWSC(Sequential Write Spatial Clock) 버퍼 교체 알고리즘

이미경, 이두기, 신민철, 박상현
연세대학교 컴퓨터과학과
e-mail : dlalrud228@naver.com

SWSC(Sequential Write Spatial Clock) Buffer Replacement Algorithm For Mobile Flash Storage

Mikyung Lee, Duki Lee, Mincheol Shin, Sanghyun Park
Dept of Computer Science, Yonsei University

요 약

지난 몇 년간 스마트폰은 굉장히 빠른 속도로 발전하면서 생활 속에서 큰 비중을 차지하고 있다. 이러한 스마트폰에는 에너지 효율, 크기, 속도 면에서 모바일 기기에 적합한 Flash storage가 탑재되고 있다. 이 논문에서는 스마트폰에 탑재된 Flash storage를 기반으로 한 버퍼 교체 알고리즘들 가운데 Spatial Clock 알고리즘에 초점을 맞추고 있다. 그리고 이 알고리즘이 Video Streaming workload에서 성능 발휘를 하지 못한다는 점을 해결하기 위해 SWSC(Sequential Write Spatial Clock) 알고리즘을 제안하였다. 이 알고리즘은 dirty 페이지들이 연속적인 경우 sequential write를 수행한다. 따라서 write 수행시간을 줄일 수 있고 결과적으로 Video Streaming workload에서도 좋은 성능을 발휘할 수 있다.

1. 서론

모바일 장치들이 등장하면서 우리 생활 속에서 큰 영향력을 발휘하는 장치로 자리 잡았다. 특히 지난 몇 년간 스마트폰은 의사소통의 매개체 역할뿐만 아니라 여러 장치를 제어하는 역할까지 수행하면서 굉장히 빠른 속도로 발전해왔다. Flash storage는 기존의 저장 장치인 하드디스크에 비해 전력 소비가 낮고 크기와 무게가 작으며 속도가 빠르다. 따라서 스마트폰에는 Flash storage가 탑재되고 있다. 학계에서도 Flash storage를 사용하여 성능을 개선하려는 아이디어들이 많이 등장하고 있다. 특히 하드디스크 대신 Flash storage를 기반으로 한 버퍼 교체 알고리즘들이 많이 등장하였다 [2, 3, 4, 5]. 이 논문에서는 스마트폰 환경에서 Flash storage를 사용하여 버퍼 교체 알고리즘의 성능을 높이려고 한다. 따라서 Flash 기반의 버퍼 교체 알고리즘들을 살펴보고 이 중에서도 Spatial Clock 알고리즘 [2]의 성능 개선에 초점을 맞추고 있다.

Spatial Clock 알고리즘은 새로운 페이지가 추가될 때마다 페이지들을 논리적 섹터 번호 순서대로 정렬시킨다. 따라서 페이지가 교체되는 동안 Spatial locality가 고려된다. 그러나 [1]에 나왔듯이 Video Streaming workload에서 이 알고리즘으로 인한 성능 향상을 거의 얻을 수 없었다. 그러므로 본 논문에서는 이와 같은 workload에서도 좋은 성능을 얻기 위해 SWSC(Sequential Write Spatial Clock) 버퍼 교체 알고리즘을 제안하고 있다. Spatial Clock 알고리즘은 페이지 교체가 발생할 때 dirty 페이지들이 연속적으로 나타나도 한 페이지만 write 한다. 그러나 SWSC 알고리즘에서는 이 페이지들을 sequential write 하므로 write 수행시간이 확연히 감소할 수 있다. 그리고 이 페이지들을 버퍼에서 삭제하지 않고 clean 페이지로 바꾸어 상주시기 때문에 캐시 적중률이 줄어들지 않고서도 성능을 높일 수 있다.

본 논문은 다음과 같은 구성을 가진다. 2장에서는 Flash storage를 바탕으로 한 버퍼 교체 알고리즘들을 소개한다. 3장에서는 2장에서 소개한 알고리즘들 중 Spatial Clock 알고리즘을 향상한 SWSC 알고리즘을 소개한다. 4장에서는 SWSC의 성능을 예측하였고 5장에서는 SWSC와 관련하여 추후 어떤 연구가 이루어질 수 있을지 설명한다.

2. 관련 연구

Flash storage가 나타나면서 학계에서 Flash storage를 사용하여 성능 향상을 시도하는 아이디어들이 많이 등장했다. 이 중에는 Buffer management 성능 향상을 위한 버퍼 교체 알고리즘들도 포함되어 있다. 가장 많이 활용되고 있는 알고리즘이 하드디스크를 기반으로 한 LRU [1]과 Clock이다. 두 알고리즘의 페이지 교체 방식은 다음과 같다. LRU(Least Recently Used)는 가장 오랫동안 접근되지 않은 페이지는 앞으로도 접근되지 않을 것이라는

Temporal locality를 고려한다. 그러므로 페이지 교체 시 가장 오랫동안 접근되지 않은 페이지를 교체한다.

Clock은 페이지를 적재한 프레임들이 순환적 형태의 리스트로 유지된다. 각 프레임은 논리적인 섹터 번호와 reference bit를 가지고 있고 reference bit는 교체할 페이지를 선정하는 데 사용된다. 페이지가 프레임에 처음 적재되거나 참조되면 프레임의 reference bit 값은 1로 설정된다. 이 알고리즘은 [1, 3, 4, 5] 알고리즘들처럼 프레임들을 이동시키지 않고 교체할 프레임을 가리키는 포인터를 사용하여 동작한다. 포인터를 시계 방향으로 이동시키면서 포인터가 가리키는 프레임 중 reference bit 값이 0인 프레임 상의 페이지를 교체한다. 교체된 프레임을 가리키던 포인터는 다음 프레임을 가리키게 된다. reference bit 값이 1인 경우 reference bit 값을 0으로 변경하고 포인터는 다음 프레임을 가리킨다. 모든 프레임의 reference bit 값이 1인 경우 FIFO와 같은 방식으로 동작한다.

Flash storage의 등장 이후, Flash storage를 기반으로 하여 LRU와 Clock을 변형한 알고리즘들이 등장하였다. BPLRU[3], FAB[4], FAB[5]는 LRU를, Spatial Clock[2]은 Clock을 변형한 알고리즘이다.

BPLRU(Block Padding Least Recently Used)는 버퍼 내부에 블록들이 리스트로 유지되며 각 블록은 페이지들로 이루어져 있다. 리스트의 한쪽 끝은 가장 오랫동안 참조되지 않은 블록이 차지하는 자리이며 리스트의 나머지 한쪽 끝은 가장 최근에 참조된 블록이 차지하는 자리이다. 따라서 페이지가 참조되면 해당 페이지가 속해있는 블록은 리스트의 가장 최근에 참조되는 블록 자리로 이동한다.

CFLRU(Clean-First Least Recently Used)는 BPLRU와 달리 리스트를 이루는 단위가 페이지이다. 이 페이지들을 최근에 참조된 페이지들이 모여있는 Working region과 오랫동안 참조되지 않은 페이지들이 모여있는 Clean-first region으로 리스트를 구별하였다. 교체될 페이지를 선정하는 과정은 Clean-first region 안에서 이루어진다. Write 횟수를 감소하기 위해 Clean-first region 안에 있는 clean 페이지가 dirty 페이지보다 교체될 수 있도록 한다.

FAB(Flash-Aware Buffer management)는 MP3나 휴대용 PMP에서 사용되었던 알고리즘이다. 버퍼에는 블록들이 리스트로 유지된다. 그리고 블록마다 페이지들이 리스트로 유지된다. 버퍼 교체가 발생한 경우 페이지 개수가 가장 많은 블록을 교체한다.

Spatial Clock은 Clock을 바탕으로 두고 있기 때문에 프레임의 구성이 같고, 프레임들이 순환적 형태의 리스트로 유지된다. 버퍼에 새로운 페이지가 추가될 때마다 페이지 프레임들이 논리적인 섹터 번호를 기준으로 정렬된다. 따라서 페이지 교체가 이루어지는 동안 Spatial locality가 고려되기 때문에 Clock보다 write 성능이 좋다. 그러나 프레임들의 reference bit 값이 연속적으로 0이다 라는 보장이 없으므로 sequential write를 보장할 수 없다.

3. SWSC 버퍼 교체 알고리즘

Spatial Clock 알고리즘은 Spatial locality를 고려하므로 Web browsing workload과 Mixed Apps workload에서는 높은 성능을 얻는다. 하지만 Video Streaming workload에서는 다른 workload들보다 I/O 수행시간이 길고 이 알고리즘으로 인한 성능 향상이 거의 없다 [2]. 이러한 현상은 Video Streaming workload에서는 write request의 대부분이 이미 sequential pattern을 많이 갖고 있기 때문이다. 그러므로 이 논문에서는 Video Streaming workload에서도 성능 향상을 얻기 위해 SWSC(Sequential Write Spatial Clock) 알고리즘을 제안한다.

SWSC은 Spatial Clock과 마찬가지로 프레임 구성이 같고, 새로운 페이지가 추가될 때마다 페이지 프레임들은 논리적인 섹터 번호 순서대로 정렬된다. Spatial Clock과 다른 점은 교체 시 dirty 페이지가 처리되는 방식과 페이지의 개수이다.

페이지 부재가 발생했지만 버퍼 내에 빈 프레임이 없는 경우, 알고리즘 1이 호출된다. 이 알고리즘에서는 교체될 페이지가 어떻게 선정되는지를 보여준다. reference bit 값이 0이고 clean인 페이지가 교체될 페이지로 선정된 경우 이 페이지는 버퍼에서 삭제된 후 새로운 페이지로 교체된다 (알고리즘 1 Line 9~10, 23~25). reference bit 값이 0이고 dirty인 페이지가 교체될 페이지로 선정된 경우, 현재 페이지 프레임의 다음 프레임을 검사한다. 만약 다음 프레임의 페이지가 현재 프레임의 페이지처럼 dirty하고 순차적인 경우 아까와 마찬가지로 또 다음 프레임을 검사한다. 이 검사는 reference bit 값이 1이거나 dirty bit 값이 0이거나 또는 순차적이지 않은 페이지가 나타날 때까지 반복 된다 (알고리즘 1 Line 13~16). 검사가 끝난 후 순차적인 dirty 페이지들은 Flash memory에 sequential write를 수행한 후 이 페이지들 중 가장 첫 번째 페이지만을 삭제한다. 그리고 이 페이지를 제외한 나머지 페이지들은 버퍼에서 삭제하지 않고 dirty bit 값을 이용하여 clean 페이지로 바꾸어 버퍼에 상주시킨다. 만약 dirty 페이지들이 순차적이지 못한 경우에는 한 페이지만 Flash memory에 write 한 후, 버퍼에서 삭제한다.

그림 1은 SWSC의 동작 과정을 보여준다. 페이지 프레임 리스트는 페이지 프레임들로 구성된다. 페이지 1개의 크기는 8개의 섹터와 같다고 가정하고 있다. 프레임은 논리적인 섹터 번호, reference bit, dirty bit로 구성된다. 프레임들은 논리적인 섹터 번호를 기준으로 정렬된다. 따라서 새로운 페이지가 추가될 때 적절한 위치를 찾아 삽입된다. 교체될 페이지를 찾기 위해 포인터를 사용하여 프레임들을 검사한다. 포인터가 가리키는 논리적인 섹터 번호가 16인 프레임은 reference bit 값이 1이므로 bit 값을 0으로 변경시킨 후 다음 프레임을 검사한다. 논리적인 섹터 번호가 24인 프레임은 reference bit 값이 0, dirty bit 값이 1이므로 다음 프레임을 검사한다. 논리적인 섹터 번호가 32인 이 프레임은 reference bit 값이 0, dirty bit 값이

1이며 이전 프레임의 페이지와 순차적이므로 sequential

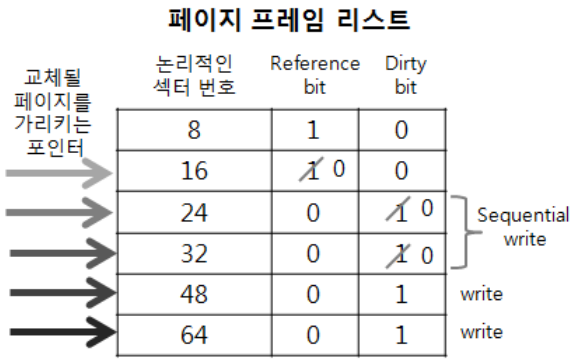
알고리즘 1

```
void replacePage(Frame *curFrame, Frame *newFrame)
    Description
    교체할 페이지를 검사한 후, 새로운 페이지와 교체한다.
    Input
     curFrame: 교체할 페이지 프레임을 가리키는 포인터
     newFrame: 새로운 페이지 프레임을 가리키는 포인터
    Output  없음

1   Frame *nextFrame = curFrame->next
2   int sectorNum = pageSize/sectorSize
3   int delNum      /* 교체할 프레임의 개수 */

4   WHILE (1) DO
5     IF curFrame->reference ==1 THEN
6       curFrame->reference = 0
7       curFrame = curFrame->next
8     ELSE
9       IF curFrame->dirty==0 THEN
          /* 포인터가 가리키는 페이지를 버퍼에서 삭제*/
10        evictPage(curFrame)
11      ELSE
12        delNum=1
          nextFrame = curFrame->next
          /* nextFrame 이 가리키는 프레임의 reference bit
          값이 0, dirty bit 값이 1, 이전 프레임의 페이지와
          순차적인 경우 */
13        WHILE (nextFrame->reference == 0 &&
                  nextFrame->dirty == 1 &&
                  nextFrame->lsn ==
                  nextFrame->prev->lsn+sectorNum)
          DO
14          delNum++
15          nextFrame = nextFrame->next
16        END WHILE
          /* 포인터가 가리키는 프레임을 Flash memory
          에 write */
17        writeToFlash(curFrame, delNum)
          /* 포인터가 가리키는 프레임을 버퍼에서
          삭제 */
18        evictPage(curFrame)
19        IF delNum != 1 THEN
            /* 포인터가 가리키는 프레임들의 dirty bit
            값을 0 으로 변경 */
20          cleanFrames(curFrame->next, delNum-1)
21        END IF
22      END IF

        /* 포인터를 사용하여 새로운 페이지를 적절한
        위치에 삽입한다. 프레임들은 정렬된 상태를 유지
        한다*/
23      insertAndSort(curFrame, newPage)
24      curFrame = curFrame->next
25      break
26    END IF
27  END WHILE
```

(알고리즘 1) 교체될 페이지를 선정하는 알고리즘

write가 가능한 조건을 갖추고 있다. 마찬가지로 다음 페이지를 검사해보면 reference bit 값이 0, dirty bit 값이 1 이지만 논리적인 섹터 번호가 48이므로 sequential pattern을 이루지 못다. 따라서 논리적인 섹터 번호가 24인 페이지와 32인 dirty 페이지를 sequential write 한 후, 논리적인 섹터 번호가 4인 페이지를 버퍼에서 삭제하고 나머지 페이지의 dirty bit 값을 0으로 변경한다.

이처럼 순차적인 dirty 페이지들을 sequential write 하므로 sequential write request가 write request의 대부분을 차지하는 workload에서 수행시간을 감소시킨다. 그리고 sequential write 한 페이지들을 clean 페이지로 변경하여 버퍼에 계속 상주시키기 때문에 적중률이 떨어질 우려가 없고, 나중에 페이지 교체가 발생해도 Flash memory에 write 할 필요가 없어진다.



(그림 1) SWSC 동작 과정

4. 성능 예측

이 논문에서는 SWSC 구현이 이루어지지 않았다. 따라서 이 알고리즘이 실제 모바일 Flash storage의 성능에 어떤 영향을 끼칠지 예측하고자 한다.

4.1 Video Streaming workload에서의 성능 예측

Video Streaming workload는 write request가 순차적이다. 그러므로 이 workload에서 Spatial Clock으로 인한 성능 향상이 거의 없다 [2]. 그러나 SWSC는 dirty 페이지들이 순차적이면 sequential write 하기 때문에 write 수행시간이 줄어든다. 따라서 Video Streaming workload에서도 좋은 성능을 기대할 수 있다.

4.2 Mobile Flash Storage에서의 성능 예측

eMMC는 random write 성능이 sequential write보다 약 2~30배 정도 낮고 microSD는 약 65배 정도 낮다 [2]. Spatial Clock은 페이지가 교체되는 동안 Spatial locality를 고려하므로 microSD와 eMMC의 낮은 random write 성능을 보완해준다 [2]. SWSC 역시 Spatial locality가 고려되기 때문에 두 장치의 write 성능을 보완해줄 수 있다.

4.3 Spatial Clock vs. WL-Spatial Crlock

이 장에서는 Spatial Clock과 SWSC 성능을 write 수행시간과 적중률을 기준으로 비교한다.

4.3.1 Write 수행시간

두 알고리즘 모두 페이지 교체가 이루어지는 동안 Spatial locality가 고려되므로 write 수행시간을 줄일 수 있다. 그러나 SWSC는 순차적인 dirty 페이지들을 sequential write 하므로 Spatial Clock보다 수행시간을 더 감소시킬 수 있을 것이다.

4.3.2 적중률

SWSC는 순차적인 dirty 페이지들을 sequential write 한 후 버퍼에서 삭제하지 않는다. 다만 dirty bit를 사용하여 이 페이지들을 clean 페이지로 변경하여 버퍼에 상주시킨다. 따라서 SWSC에서도 Spatial Clock과 같이 적중률이 크게 떨어지지 않을 것으로 예측된다.

5. 결론 및 향후 연구

모바일 기기의 사용이 증가하고 Flash storage가 하드디스크를 대체하는 저장장치로 대두하면서 스마트폰에서 Flash storage가 기본적으로 사용되고 있다. 본 논문에서는 스마트폰에서 Flash storage를 사용하여 Buffer management의 성능을 높이려는 버퍼 교체 알고리즘들 중 Spatial Clock을 살펴보았다. 이 알고리즘은 Video Streaming workload에서 성능 향상이 거의 없다. 이를 해결하기 위해 SWSC 알고리즘을 제안하였다. 이 알고리즘은 Flash storage가 sequential write 성능이 높은 점을 이용하였다. 순차적인 dirty 페이지들을 sequential write 하므로 Video Streaming workload에서 수행시간을 확실하게 감소시킬 수 있다. 또한, 이 페이지들을 clean 상태로 바꾸어 버퍼에 계속 상주시키기 때문에 캐시 적중률이 감소하지 않고서도 성능을 높일 수 있을 것으로 예측된다.

그러나 write request의 대부분이 random pattern인 workload에서는 Spatial Clock과 성능의 차이가 크게 나타나지 않을 것으로 보인다. 따라서 이를 보완하기 위해 일단 이 알고리즘을 실제 기기에서 구현하여 다른 버퍼 교체 알고리즘들과 성능을 비교하고 분석하려 한다. 또한 Flash storage의 특성 중 중요한 요소로 꼽히는 write 횟수를 줄일 수 있도록 수단을 취하고자 한다.

참고문헌

[1] Asit Dan, Don Towsley "An approximate analysis of the LRU and FIFO buffer replacement schemes" 1990. SIGMETRICS

[2] Hyojun Kim, Moonkyung Ryu, Umakishore Ramachandran "What is a Good Buffer Cache Replacement Scheme for Mobile Flash Storage?" 2012. SIGMETRICS

[3] Hyojun Kim, Seongjun Ah "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage" 2008.
FAST: 6th USENIX Conference on File and Storage Technologies.

[4] Seon-yeong Park, Dawoon Jung, Jeong-uk Kang, Jin-soo Kim, Joonwon Lee "CFLRU: A Replacement Algorithm for Flash Memory" 2006. CASES

[5] Heeseung Jo, Jeong-Uk Kang, Seon-Yeong Park, Jin-Soo Kim, and Joonwon Lee "FAB; Flash-Aware Buffer management policy for Portable Media Players" 2006.IEEE