플래시 메모리 성능향상을 위한 핫 페이지 관리 기법을 이용한 버퍼교체 정책

김대영, 김정한, 조현진, 엄영익
성균관대학교 정보통신공학부
e-mail: {dykim14, gtgkjh, hjcho, yieom}@ece.skku.ac.kr

A Buffer Replacement Policy using Hot Page Management Scheme for Improving Performance of Flash Memory

Daeyoung Kim, Junghan Kim, Hyun-jin Cho, and Young Ik Eom
School of Information and Communications Engineering, Sungkyunkwan University

요 약

플래시 메모리는 우리 생활에 널리 사용되고 있는 휴대용 저장장치 중의 하나이다. 빠른 입출력 속도와 저전력, 무소음, 작은 크기 등의 장점을 가지나 덮어쓰기가 불가능하고 읽기/쓰기의 속도에 비해 소거 연산의 속도가 매우 느리다는 단점이 있다. 이를 보완하기 위해, 호스트와 플래시 메모리 사이에 버퍼 캐시를 두어 사용하고 있으며, 버퍼 캐시에 사용되는 교체 정책에 따라 플래시 메모리 장치의 성능이 크게 영향을 받는다. 본 논문에서는 블록 단위의 LRU 기법의 단점을 개선한 HPLRU 기법을 제안한다. HPLRU 기법은 최근에 자주 참조되었던 페이지인 핫 페이지 들을 모아 리스트를 만들어 관리하고, 이를 통해 페이지 적중률을 향상시키고 다른 페이지들로 인해 핫 페이지들이 소거되는 현상을 개선하였다. 이 알고리즘은 임의 데이터 패턴에 좋은 성능을 보이며 쓰기 발생 횟수를 많이 감소시키는 결과를 보였다.

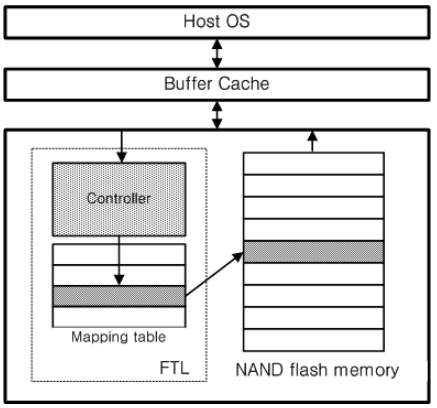
1. 서론

플래시 메모리는 작고 가벼우면서도 많은 용량을 제공하며 저전력, 무소음의 특징을 가진 저장 매체이다. 이러한 장점이 있는 반면에 플래시 메모리에는 몇 가지 물리적인 제한들이 존재한다. 그 중의 하나가 '쓰기 전 소거(erase-before-write)'라는 제약으로, 한번 사용한 페이지 위에 데이터를 쓰기 위해서는 반드시 소거 연산(erase operation)을 해주어야 한다. 이로 인해 플래시 메모리는 동일한 주소에 소거 연산 없이 덮어 쓰기 연산(overwrite operation)을 하는 것이 불가능하며, 이것은 플래시 메모리의 전반적인 쓰기 성능을 크게 저하시킨다. 그리고 각 플래시 메모리 셀에 대하여 소거 횟수가 제한되어 있어서 소거 한도를 초과할 경우 해당 메모리 셀은 더 이상 사용할 수 없게 된다. 또 다른 제한 요소는 입출력 연산의 단위와 처리 속도의 비대칭성이다. 읽기/쓰기 연산이 페이지 단위로 이루어지며 각각의 동작 속도가 15, 200 μs인데 반해, 소거 연산은 블록 단위로 동작하며 처리 속도는 10 ms로 앞의 두 연산과는 큰 차이가 있다[1].

이와 같은 플래시 메모리의 물리적인 제한에도 불구하고, 우리는 흡사 하드 디스크를 사용하듯이 플래시 메모리를 사용하고 있다. 이런 일이 가능한 것은 FTL(Flash Translation Layer)이라고 하는 소프트웨어 계층이 호스트의 파일 시스템과 플래시 메모리 장치 사이에서 중계 역할을 하고 있기 때문이다[2]. FTL은 파일 시스템의 논리 주소와 플래시 메모리의 물리 주소간의 매핑 정보(mapping information)를 관리하며, 가비지 콜렉션(garbage collection)과 마모도 평준화(wear leveling) 관리의 역할을 한다. 버퍼 캐시는 전체 디스크 블록의 일부를 주기억장치에 저장하는 공간으로서, 디스크의 입출력 연산의 발생 횟수를 줄여 시스템의 성능을 향상시킨다. 운영체제가 페이지 요청을 할 때, 요청한 페이지가 버퍼 캐시 내에 존재하는 확률을 페이지 적중률이라 한다. 적중률(hit ratio)이 높을수록 디스크에 접근하지 않고 버퍼 캐시를 통해 입출력 연산을 수행하게 되므로 전체 디스크 성능이 향상된다. 그러므로 버퍼 캐시 교체 정책의 목적은 페이지 적중률을 최대화하는 것이다. 플래시 메모리는 입출력 연산의 속도가 비대칭적이므로 플래시 메모리를 위한 버퍼 캐시 알고리즘은 적중률과 더불어 입출력 연산의 속도 차이를 고려해야 한다.

본 논문은 플래시 메모리 저장 장치의 성능을 향상시키기 위해서 버퍼 캐시를 두 개의 리스트로 나누어 관리하는 HPLRU(Hot Page indication LRU) 기법을 제시한다. 제안된 정책은 버퍼 캐시 내에서 재 참조된 페이지인 핫 페이지(hot page)로 구성된 핫 페이지 리스트를 이용하여 페이지 적중률을 향상시킨다.

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음(IITA-2008-C1090-0801-0027)



(그림 1) NAND 플래시 메모리 저장 장치의 구조

논문의 구성은 다음과 같다. 2장에서는 관련 연구와 배경을 살펴보고, 3장은 본 논문에서 제안한 정책의 세부 사항을 알아보고, 4장은 제안한 정책의 성능 평가, 마지막으로 5장에서는 결론과 함께 향후 연구에 대하여 기술한다.

2. 관련 연구 및 배경

2.1 Flash Translation Layer(FTL)

FTL의 주요 역할은 플래시 메모리 상에 기존의 파일 시스템을 사용할 수 있게 하는 것이다. 이를 위해 FTL은 매핑 테이블(mapping table)을 이용하여 기존 파일 시스템의 논리 주소를 플래시 메모리에서 사용하는 물리 주소로 매핑한다. 플래시 메모리에서 기존 파일 시스템처럼 같은 페이지에 덮어 쓰기를 하기 위해서는 상대적으로 긴 시간이 걸리는 소거 연산을 먼저 수행해야 한다. 하지만 FTL은 이미 소거된 페이지에 덮어 쓰기를 하도록 매핑 정보를 변경함으로서 소거 연산을 피해갈 수 있다.

덮어 쓰기로 인해 다른 블록의 갱신된 페이지만을 가지는 블록을 로그 블록이라고 한다. 로그 블록을 사용한 FTL인 BAST(Block-Associative Sector Translation)는 로그 블록에 하나의 데이터 블록만이 갱신된 페이지를 저장할 수 있다[3]. BAST는 제한된 숫자의 로그 블록을 사용했기 때문에, 임의 패턴의 쓰기 연산 시에는 성능이 나빠진다. 이러한 단점을 고치기 위해 제안된 FAST (Fully-Associative Sector Translation)는 어떤 블록이라도 로그 블록에 갱신된 페이지를 저장할 수 있어서 로그 블록의 공간 낭비를 줄이면서 임의 패턴에 대한 쓰기 성능을 향상시켰다[4].

2.2 버퍼 캐시 알고리즘(Buffer Cache Algorithm)

플래시 메모리를 위한 버퍼 캐시 알고리즘은 입출력 연산의 성능을 향상시키기 위해 플래시 메모리의 블록 일부를 버퍼에 저장하여 페이지 적중률을 높이는 것이 목적이다. 또한 읽기 연산보다 쓰기 연산이 약 10배 이상 느리기 때문에, 입출력 성능을 향상시키기 위해서는 가능한 한 적은 수의 쓰기 연산이 수행되는 것이 좋다.

FAB(Flash-Aware Buffer) 관리기법은 대용량의 순차 파일을 효과적으로 처리하므로 쓰기 연산의 요청이 순차적인 휴대용 멀티미디어 장치들에게 유용한 버퍼 교체 기법이다[5]. FAB 기법은 블록 단위로 페이지들을 관하며, 한 블록은 같은 블록 번호를 가지는 페이지들로 구성된다. 버퍼 캐시의 공간이 모자랄 경우, 가장 페이지 수가 많은 블록을 희생 블록(victim block)으로 선택하여 버퍼에서 제거한다. 페이지 수가 많은 블록을 제거할수록, 로그 블록 FTL 상에서는 합병해야 할 블록의 수가 줄어들게 되므로 연산의 비용이 줄어든다. FAB 기법은 순차 쓰기 패턴을 가지는 대용량 멀티미디어 파일의 처리에 적합하나, 블록 크기보다 큰 파일이나 프로세스가 빈번하게 참조되는 경우에는 그리 적합하지 않다.

BPLRU(Block Padding LRU) 기법은 블록 단위의 LRU 기법을 기반으로 하며, SSD(Solid State Disk)의 쓰기 버퍼 정책에 사용되는 알고리즘이다[6]. LRU 기법에 따라 블록들의 순서가 정해지지만, 블록내의 모든 페이지가 버퍼 안에 있을 경우에는 그 블록이 우선적으로 희생 블록이 된다. 이것은 쓰기 패턴이 순차적인 파일의 처리에 유리하다. 그리고 희생 블록이 플래시 메모리에 쓰이기 전에, 참조되지 않았던 희생 블록의 페이지 모두를 읽어서 희생 블록의 공간에 저장하는 것으로 교체 합병 연산의 수행을 유도한다. BPLRU 기법은 순차 쓰기 패턴과 지역성이 강한 쓰기 패턴에 매우 좋은 성능을 보여주며, 임의 쓰기 연산 또한 그에 못지않은 훌륭한 성능을 보여준다.

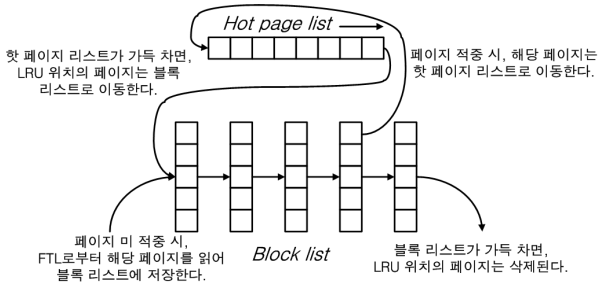
BPLRU 기법과 같은 블록 단위의 LRU 기법은 다수의 콜드 페이지(cold page)로 구성된 블록이 재 참조된 소수의 페이지로 인해 오랫동안 버퍼에 머무르게 되는 경우, 버퍼 공간의 낭비를 초래하게 되는 단점이 있다. 본 논문의 알고리즘은 블록 단위로 정책이 수행되면서 핫 페이지를 따로 관리하므로 높은 적중률과 그로 인한 쓰기 연산 횟수의 감소 효과를 가진다.

3. HPLRU(Hot Page indication LRU)

HPLRU 기법은 일반적인 플래시 메모리 장치의 버퍼 캐시 교체 정책으로 사용될 수 있는 기법으로, 블록 단위의 리스트와 페이지 단위의 리스트로 구성된다. 버퍼 캐시의 페이지 적중률을 높게 하기 위하여, 블록 리스트에서 한번 이상 재 참조된 페이지인 핫 페이지들을 뽑아 페이지 단위의 리스트에 넣어 특별히 관리한다. 그리고 HPLRU 기법은 희생 블록의 삭제시 해당 블록에서 아직 읽혀지지 않았던 페이지들을 읽어와서, FTL상에서 일어나는 합병 연산의 비용을 최소화할 수 있게 만들어 입출력의 성능을 높인다.

3.1 HPLRU의 구조

HPLRU 기법은 그림 2처럼 버퍼 캐시를 두 개의 리스트로 나누어 관리한다.



(그림 2) HPLRU의 구조

블록 리스트는 블록 단위로, 핫 페이지 리스트는 페이지 단위로 해당 리스트 안에 있는 페이지들을 관리한다.

블록 리스트는 블록 번호가 동일한 페이지들로 한 블록을 구성하는데, 각 블록들은 버퍼에 들어온 순서대로 정렬된다. 페이지들의 참조는 블록의 순서에 영향을 주지 않으며 블록들은 처음 들어온 순서대로 버퍼에서 삭제된다. 처음 버퍼 캐시에 들어온 페이지는 블록 리스트에 먼저 저장이 되고, 블록 리스트의 공간이 모자랄 경우 가장 먼저 들어온 블록이 삭제된다.

핫 페이지 리스트는 버퍼 안에 있는 페이지들 중에서 최근에 재 참조된 페이지들로 구성된다. 블록 리스트와는 달리 LRU 기법으로 페이지들을 관리하며, 리스트의 공간이 모자랄 경우 LRU 위치에 있는 페이지가 삭제되고 블록 리스트에 재삽입이 된다. 리스트 내의 페이지들은 새로운 핫 페이지가 리스트에 삽입이 되지 않는 한 삭제되지 않는다. 이것은 핫 페이지 리스트의 중요한 특성으로 대용량 순차 파일뿐만 아니라 크기가 작은 파일이 다수 들어와도 재참조가 되지 않는 한, 리스트의 내용은 보존된다.

블록 리스트와 핫 페이지 리스트의 적정 비율은 시뮬레이션을 통해 구했다. 핫 페이지 리스트의 비율이 커질수록 페이지 단위 LRU 기법의 성능과 비슷해지는 결과가 나타났다. 그리고 블록 리스트의 비율이 커질수록 블록 단위 LRU 기법의 성능과 비슷해지는 대신, 핫 페이지의 인식 효과가 낮아져 성능이 저하되었다. 블록 단위 LRU 기법의 특성을 유지하면서 핫 페이지의 인식 효과를 최대한으로 유지할 수 있는 적정 비율은 블록 리스트가 99%, 핫 페이지 리스트가 1%일 때로 나타났다.

3.2 HPLRU의 정책

HPLRU 기법에서는 버퍼 캐시로 들어오는 모든 페이지들은 블록 리스트를 거치게 되며, 재 참조된 페이지들은 핫 페이지 리스트로 삽입된다. 블록 리스트는 선입선출의 방식으로 운영되므로 페이지가 재 참조되더라도 해당 블록의 위치는 변하지 않는다. 이를 통해 블록 리스트의 LRU 위치에 있는 블록이 삭제될 때, 콜드 페이지는 삭제되며, 핫 페이지는 버퍼 캐시 내에 그대로 유지될 수 있다.

블록 단위의 리스트를 사용하는 이유는 FTL에서 발생하는 합병 연산의 비용을 최소화하기 위해서이다. 합병 연산은 데이터 블록과 로그 블록에서 가장 최근에 갱신된 페이지들을 추출해 새로운 데이터 블록을 만들고 나머지 블록들을 소거해 저장 공간을 확보하는 연산이다. 합병 연산에 참가하는 데이터 블록의 수가 작을수록 소거되는 블록의 수도 작아지고, 합병 연산의 비용 역시 작아진다.

그래서 합병 연산의 이상적인 경우는 매핑 정보만 바꾸는 것으로 로그 블록을 데이터 블록으로 사용할 수 있는 경우이며 이를 교체 합병(switch merge) 연산이라고 한다. 교체 합병 연산이 발생되기 위해서는 로그 블록이 중복되는 페이지 없이 온전한 한 블록으로 구성되어야 하는데, 이것을 결정하는 것은 바로 버퍼 캐시에서 나오는 페이지들이다.

버퍼 캐시에서 FTL로 전달되는 페이지들을 온전한 한 블록으로 만들기 위해, BPLRU 기법에서 사용된 페이지 패딩(page padding) 기법을 이용했다. 이 방법은 버퍼 캐시의 희생 블록이 확정되고 삭제가 이루어 질 때, 희생 블록에서 누락된 페이지들을 플래시 메모리로부터 읽은 후 채워넣는 것으로 약간의 읽기 연산을 추가하는 것으로 소거 연산의 횟수를 줄일 수 있다.

그리고 페이지들이 순차적으로 들어와 한 블록을 온전히 채운 경우, 이것을 대용량 순차 파일로 생각하고 이런 파일의 대부분이 가까운 시간 내로 재 참조되지 않기 때문에 해당 블록을 리스트의 LRU 위치에 넣어준다. 이 정책을 통해 대용량 순차 파일이 버퍼 캐시의 입력으로 된다고 해도 버퍼의 내용은 거의 유지가 되고 페이지 적중률이 유지된다.

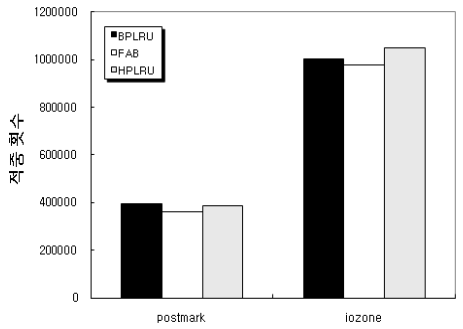
4. 성능 평가

본 논문에서 제안하는 HPLRU 기법과 BPLRU 기법, FAB 기법의 블록 쓰기 발생 횟수와 적중 횟수를 비교한다. 성능 평가의 방법으로는 트레이스 기반의 시뮬레이션을 사용되었으며 시뮬레이션의 데이터는 요청된 디스크의 번호와 읽기/쓰기 여부로 구성된다. 성능 측정에 사용된 두 개의 트레이스는 각각 윈도우와 리눅스 운영체제를 기반으로 하여 벤치마크 프로그램인 IOzone과 Postmark를 이용하여 만들어졌으며, 랜덤 읽기/쓰기 패턴으로 동작한 것을 기록하였다[7, 8].

4.1 버퍼 적중 횟수

그림 3은 각 버퍼 교체 알고리즘의 버퍼 적중 횟수를 비교하여 보여준다.

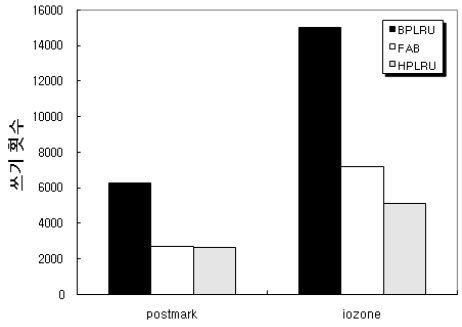
버퍼 적중 횟수는 페이지 단위로 측정하였으며 모든 알고리즘이 큰 차이 없이 비슷한 횟수를 기록하고 있다. 근소한 차이지만 FAB의 적중 횟수가 가장 적게 나왔으며, LRU 기법을 포함하고 있는 HPLRU와 BPLRU의 적중 횟수는 비교적 높게 측정되었다. 이것은 데이터의 시간 지역성(time locality)을 잘 반영하는 LRU 기법의 특성 때문이다.



(그림 3) 버퍼 적중 횟수

4.2 쓰기 발생 횟수

그림 4는 쓰기 발생 횟수에 대한 시뮬레이션 결과를 보인다.



(그림 4) 블록 쓰기 횟수

쓰기 발생 횟수가 작게 나올수록 뛰어난 성능을 내는 것이다. 특히 쓰기 연산은 FTL에서 합병 연산과 바로 연결되므로 소거 연산까지 간접적으로 포함한다고 볼 수 있다. HPLRU와 FAB이 비슷한 성능을 보여주는 데 반해, BPLRU는 이들의 거의 두 배에 가까운 쓰기 횟수를 보여주고 있다. 이전 섹션에서 적중한 페이지의 개수가 세 알고리즘이 비슷하게 나왔으나, 블록 쓰기 횟수에서 차이가 나는 것은 핫 페이지를 얼마만큼 잘 유지했느냐의 차이로 분석된다. 왜냐하면 FAB은 희생 블록을 선정할 때 페이지의 개수가 가장 많은 블록을 선택하므로 핫 페이지를 효과적으로 유지하는 것은 아니지만, 다른 페이지들을 읽어들일 공간 확보가 용이하므로 어느 정도 보상이 되며, HPLRU는 핫 페이지 리스트를 통해 핫 페이지를 위한 공간을 확보하였고 페이지 단위의 관리를 통해 핫 페이지가 버퍼에 머무르는 시간이 다른 알고리즘들에 비해 길어지기 때문이다. 하지만 BPLRU의 경우, 블록 단위의 LRU 정책을 사용하므로 핫 페이지를 유지할 수 있는 시간이 상대적으로 짧기 때문에 적중률에 비해 쓰기 횟수가 많이 나오게 된 것으로 분석된다.

5. 결론

본 논문에서는 플래시 메모리를 위한 버퍼 교체 정책으로 HPLRU 기법을 제시하였다. HPLRU는 핫 페이지로 이루어진 리스트를 사용하여, 블록 단위 LRU 기법을 사용하였을 때 콜드 페이지가 필요이상으로 버퍼에 체류하는 현상을 없애고, BPLRU가 블록 크기보다 작은 파일들 다수가 버퍼로 들어올 경우, 핫 페이지가 버퍼 내에서 삭제되어 버리는 점을 개선하였다. 그 결과, 임의 데이터 패턴에서의 페이지 적중률이 높아졌고 블록 쓰기 횟수가 감소되어 전체 입출력 성능이 향상되었다. 이는 플래시 메모리를 사용하는 휴대용 저장장치에서의 성능 향상을 기대할 수 있다. 하지만 핫 페이지의 잦은 교체 시 전체적인 성능이 나빠지는 점을 개선해야 할 필요가 있고, 핫 페이지 리스트의 크기가 다양한 워크로드에 대해 충분한 실험을 통한 검증이 필요하다.

향후 연구로는 다양한 핫 페이지의 활동 패턴에 대해 적중률을 유지하는 방안과 개선된 정책을 실제 플래시 메모리를 이용한 시스템에 적용하여 다양한 워크로드로 테스트를 수행하는 것이다. 이를 통해 여러 데이터 패턴에 대한 최적의 성능을 발휘할 수 있는 페이지 리스트의 적절한 비율을 찾아내고 그 성능을 비교분석 할 계획이다.

참고문헌

[1] "Nand Flash Memory & Smartmedia data book," Samsung Electronics, 2005.

[2] Understanding the flash translation layer(FTL) specification, http://developer.intel.com

[3] J. S. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. K. Cho, "A Space-efficient Flash Translation Layer for CompactFlash Systems," IEEE Trans. on Consumer Electronics, Vol. 48(2), pp. 366-375, 2002.

[4] S. W. Lee, D. J. Park, T. S. Chung, D. H. Lee, S. W. Park, and H. J. Song, "A Log Buffer-Based Flash Translation Layer using Fully-Associative Sector Translation," ACM Trans. on Embedded Computing Systems, Vol. 6(3), pp. 18-44, 2007.

[5] H. S. Jo, J. U. Kang, S. Y. Park, J. S. Kim, and J. W. Lee, "FAB : Flash-Aware Buffer Management Policy for Portable Media Players," Consumer Electronics, IEEE Trans. on, Vol. 52(2), pp. 485-493, 2006.

[6] H. J. Kim and S. J. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," FAST '08: 6th USENIX Conf. on File and Storage Technologies, pp. 239-252, 2008.

[7] W. D. Norcott, and D. Capps, "IOzone Filesystem Benchmark," http://www.iozone.org

[8] NetApp Corp., "PostMark : A New File System Benchmark," http://www.netapp.com