대용량 플래시 메모리를 위한 선택적 페이지 변환 기법

최병두, 신동군
성균관대학교 정보통신공학부
e-mail : {chee1004, dongkun}@skku.edu

Selective Page-Level Mapping Technique for Large-Scaled Flash Memory

Byung-Doo Choi, Dongkun Shin
School of Information and Communication Engineering, Sungkyunkwan University

요 약

본 논문에서는 플래시 메모리를 효율적으로 관리하기 위하여 페이지 수준 주소 변환과 블록 수준 주소 변환을 선택적으로 적용하는 기법을 제시한다. 페이지 수준 변환을 사용하는 기존 FTL은 대용량의 주소 관리 정보를 플래시 메모리에 저장하여 성능과 사용측면에서 여러 문제점이 있다. 제안된 기법은 일부의 블록만 페이지 수준 주소 변환을 사용하고 나머지 블록은 블록 수준 주소변환을 이용하여 주소 관리 정보에 필요한 메모리 공간을 기존의 20% 수준으로 줄인 동시에 성능을 약 28%향상시켰다.

1. 서론

플래시 메모리는 하드디스크에 비해 높은 안정성과 전력 소모가 적다는 장점 때문에 음향기기, 휴대전화, 디지털 카메라와 같은 휴대용 기기의 저장장치로 사용되고 있으며, 점차 사용되는 분야가 늘어나고 있는 추세이다.

플래시 메모리는 하드디스크와 같은 기존 저장장치와 다른 몇 가지 특징들을 가지고 있다. 플래시 메모리는 읽기, 쓰기, 삭제의 세 가지 연산을 수행한다. 읽기와 쓰기 연산은 페이지 단위로 수행되고 삭제 연산은 블록 단위로 수행된다. 그리고 플래시 메모리는 '삭제 후 쓰기'가 가능하다. 즉, 데이터가 기록된 페이지는 값을 변경할 수가 없으며 다른 값을 기록하기 위해서는 해당 페이지가 포함된 블록을 삭제한 후 다시 쓰기를 수행해야한다. 또한 블록 내의 페이지는 처음부터 끝까지 순차적으로 쓰여야 한다. 블록 내부에서 쓰기를 수행한 페이지보다 앞에 빈 페이지가 존재해도 해당 페이지에 쓰기를 수행할 수 없다.

위와 같은 특징 때문에 플래시 메모리로 구성된 저장장치는 기존의 장치와 다른 파일 시스템을 필요로 한다. 하지만 플래시 변환 계층(Flash Translation Layer, FTL)을 사용함으로써 기존의 파일 시스템을 이용할 수 있다. FTL은 플래시 메모리를 관리하며 하드디스크와 같은 인터페이스를 제공한다. FTL의 주된 역할은 상위 계층의 논리 주소를 실제 플래시 메모리의 물리 주소로 변환하는 것이다. 이를 주소 변환(Address Mapping)이라 한다. 주소 변환 기법에는 페이지 수준 변환, 블록 수준 변환, 하이브리드(hybrid)변환 기법 [1]이 있다.

본 논문에서는 대용량 플래시 메모리에 페이지 단위 변환을 적용할 때 발생하는 문제점을 해결하기 위해 선택적 페이지 변환 기법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 FTL기법의 특징과 단점을 살펴볼 것이며, 3장에서 이러한 단점을 보완한 FTL을 제시하였다. 4장에서는 대용량 플래시 메모리를 위한 선택적 주소 변환방법을 설명하였다. 5장에서는 시뮬레이션을 통해 본 논문에서 제안한 기법을 실험할 것이며 마지막으로 6장에서 결론과 향후 연구 과제에 대해 기술하고자한다.

2. 관련 연구

로그 버퍼 기반 FTL은 로그 블록과 데이터 블록의 연관 관계에 따라 1:1 로그 블록 변환(BAST), 1:N 로그 블록 변환(FAST)[2], N:N+K 로그 블록 변환(Superblock)[3] 등으로 구분할 수 있다. 또 다른 N:K 로그 변환 기법 SAST[4]가 제안 되었으며 지역성을 고려한 변환 방식인 LAST(Locality-aware Sector Translation)[5]가 소개되었다. 이 기법에서는 로그 버퍼를 분류함으로써 가비지 컬렉션 비용을 줄이고 있다.

슈퍼블록 FTL(이하 SB)은 하이브리드 변환 기법의 하나로 3단계의 변환 테이블을 사용한다. 변환 테이블은 플래시 메모리의 스페어 영역에 저장된다. 슈퍼블록 FTL에 쓰기 요청이 도착하면 로그블록을 슈퍼블록에 할당하여 데이터를 기록한다.

SB는 1:N 로그 블록 변환에서 합병 연산에 드는 비용을 줄이기 위해 N:K 로그 블록 변환 기법을 사용하였다. 이 기법에서는 N개의 데이터 블록에 대해 K개의 로그 블록을 연관시킨다. 1:N 방식은 로그 블록의 연관도가 높기

때문에 합병 비용이 매우 커지는 반면 SB에서는 데이터 블록을 N개로 묶어 그룹으로 지정하고 그룹에 속한 데이터 블록들마다 로그 블록을 할당하기 때문에 최대 연관도를 N으로 제한할 수 있다.

SB에서 더 이상 빈 로그 블록이 없어 쓰기를 수행할 수 없을 경우에는 가비지 컬렉션(Garbage Collection)이 발생한다. 로그 블록에 연관된 슈퍼블록의 데이터 블록이 모두 무효화 되었다면 이를 선택하여 삭제하고 로그 블록을 하나 선택하여 데이터 블록으로 변경하는 교체 합병을 수행한다. 만약 교체 합병을 할 수 있는 데이터 블록이 없다면 가장 접근이 오래된 슈퍼블록을 선택하여 해당 슈퍼블록의 데이터 블록의 공간에 따라 부분 합병이나 완전 합병을 수행한다.

완전합병은 빈 블록을 하나 할당받아 슈퍼블록 내의 데이터 블록 중 유효한 페이지 수가 가장 작은 두 개를 선택하여 유효한 페이지를 복사한 뒤 선택된 데이터 블록을 삭제하는 것으로 수행된다. 만약 유효한 페이지의 수가 한 블록을 초과한다면 블록을 하나 더 할당 받아 또 다른 데이터 블록의 유효한 페이지를 복사한 뒤 삭제한다.

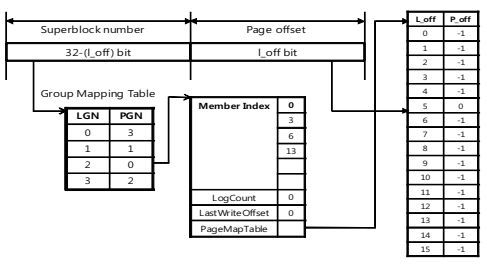
SB는 데이터 블록을 묶어 슈퍼블록 단위로 관리한다. 슈퍼블록 내부는 페이지 단위로 관리되므로 슈퍼블록 변환 테이블과 해당 슈퍼블록 내의 페이지 변환 테이블이 필요하며 이는 플래시 메모리의 스페어 영역에 저장된다. 그러므로 슈퍼블록 내의 페이지 변환 테이블의 크기가 스페어 영역의 크기보다 커질 수 없다. SB는 슈퍼블록의 크기가 커질수록 좋은 성능을 보이는 반면 변환 테이블 크기에 한계가 있기 때문에 성능에 제약이 있다.

3. 슈퍼블록 FTL의 수정

본 논문에서는 페이지 단위 변환 기법의 이점을 최대한 살리기 위해서 SB의 쓰기와 가비지 컬렉션 기법을 수정(이하 SB$^+$)하였다.

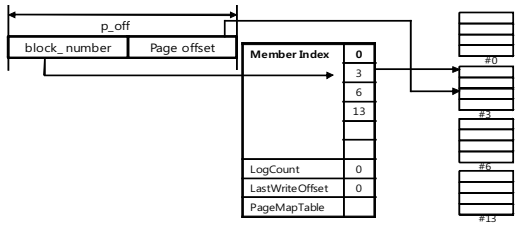
3.1 주소 변환 구조

SB$^+$에서 FTL의 주소 변환 구조는 그림 1과 같다. 상위 계층에서 FTL로 전송된 논리 주소는 슈퍼블록 번호와 페이지 오프셋으로 나누어진다. 먼저 그룹 변환 테이블에서 슈퍼블록번호를 사용하여 슈퍼블록 구조체의 위치를 찾는다. 다음으로 페이지 오프셋을 통해 슈퍼블록에서 데이터가 기록된 페이지의 물리 오프셋을 검색한다.



(그림 1) 주소 변환 구조

그림 2와 같이 물리 오프셋은 다시 두 부분으로 나누어진다. 블록번호는 슈퍼블록에서 데이터가 기록된 블록을 가리키며 페이지 오프셋으로 해당 블록에서의 페이지 위치를 찾아낼 수 있다.



(그림 2) 물리 오프셋에서 물리 주소로의 변환

3.2 쓰기 요청의 처리

SB에서 로그 블록에서만 쓰기 연산이 수행되던 것과는 다르게 SB$^+$는 슈퍼블록 내부의 빈 페이지에서 쓰기를 수행한다. 즉 데이터 블록에 빈 공간이 있는 경우에는 데이터 블록에 쓰기를 수행할 수 있다. 그렇지 않을 때에는 업데이트 블록을 할당 받아 쓰기를 수행한다.

쓰기 요청은 다음과 같은 순서로 이루어진다. 먼저, 해당 슈퍼블록을 찾고 해당 슈퍼블록에 데이터를 쓸 수 있는 빈 페이지가 있는지 확인한다. 빈 페이지가 없을 경우 업데이트 블록을 할당받는다. 업데이트 블록을 모두 사용하여 더 이상 업데이트 블록을 할당할 수 없는 경우에는 가비지 컬렉션을 수행하여 빈 공간을 확보한다.

다음으로 페이지 테이블을 확인하여 해당 논리 페이지가 기록되었는지 확인하여 데이터가 존재하면 그 페이지를 무효화를 시키고 빈 공간에 쓰기 작업을 수행한다.

3.3 가비지 컬렉션

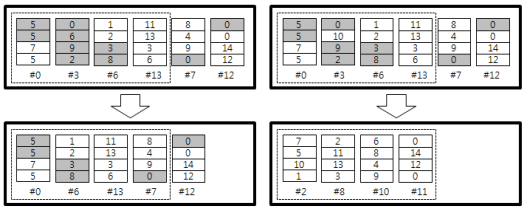
이미 업데이트 블록이 전부 할당되어 더 이상 블록을 할당할 수 없거나 슈퍼 블록이 이미 할당 받을 수 있는 업데이트 블록을 모두 사용 중인 경우에 새로 업데이트 블록 할당이 요청되면 가비지 컬렉션이 발생한다.

가비지 컬렉터는 업데이트 블록을 할당받은 슈퍼블록 중 가장 오래전에 접근되었던 슈퍼블록을 선정한다. 이에 포함된 블록들을 확인하여 모든 페이지가 무효화된 블록이 있을 경우 그림 3의 (a)와 같이 교체합병을 수행한다. 교체합병을 수행할 수 없는 경우에는 완전 합병을 수행한다. 교체 합병은 SB에서 가비지 컬렉션하는 방법과 동일하나 완전 합병은 SB와 다르다. SB에서 완전 합병이 이루어지면 한 개의 빈 블록이 만들어지는데 반해 SB$^+$의 완전합병은 그림 3의 (b)와 같이 모든 유효한 페이지들을 새로운 데이터 블록에 복사하고 이전 데이터 블록을 전부 삭제하여 슈퍼블록에 할당된 모든 업데이트 블록을 제거하여 업데이트 블록 수만큼 빈 블록이 생성된다.

가비지 컬렉션이 수행되면 유효한 페이지만을 복사하고

빈 공간을 확보한다. 완전 합병 연산이 다수 발생한 슈퍼 블록의 내부에서 자주 갱신된 페이지는 계속 업데이트가 발생하여 슈퍼블록 내에서 뒤쪽에 분포하게 된다. 반면 변경이 없는 페이지는 슈퍼블록 내부에서 앞쪽에 모이게 된다. 즉 데이터의 접근 빈도에 따라 슈퍼 블록 내부에서 페이지들이 구분되는 결과를 낳는다.

자주 업데이트 되는 페이지의 경우 무효화된 페이지의 수가 늘어나게 되며 이러한 데이터가 모임으로써 교체 병합 연산의 가능성이 증가한다. 또한 더 이상 접근이 없을 페이지가 가비지 컬렉션에 의해 복사되는 비용의 낭비 역시 줄일 수 있다.



(그림 3) 합병 작업 예제

4. 선택적 주소 변환 기법

점차 플래시 메모리를 사용하는 저장장치의 용량이 늘어나면서 더 큰 메모리 공간이 요구되고 있으며 이를 스페어 영역에 모두 저장하는 것은 불가능하다는 문제가 있다. 예를 들어 16GB의 대용량 플래시 메모리의 경우 슈퍼 블록 변환 테이블의 크기는 약 1MB정도이며 슈퍼블록 내의 페이지 변환 테이블은 32MB의 공간이 필요하다.

그러므로 본 논문에서는 전체 플래시 메모리의 슈퍼 블록 단위 변환 테이블을 유지하며 현재 페이지 단위로 관리되는 일부분의 슈퍼 블록의 페이지 단위 변환 테이블을 관리함으로써 플래시 메모리를 효율적으로 활용한다.

4.1 선택적 주소 변환

고정된 주소 변환 기법을 사용하는 SB와 달리 본 논문에서는 슈퍼블록에 저장된 데이터의 특징에 따라 다른 주소 변환 기법(이하 SSB$^+$)을 제공한다. 한번 기록된 이후 변경되지 않는 데이터를 갖는 슈퍼블록은 블록 수준 변환을 이용하여 관리한다. 크기가 작은 반면 변경이 자주 일어나는 데이터가 속해있는 슈퍼블록은 페이지 수준 변환으로 관리하는 것으로 블록 단위 변환과 페이지 단위 변환 기법의 장점을 활용한다.

일반적으로 대용량의 연속적인 쓰기 요청은 한번 발생한 이후 더 이상 갱신되지 않을 가능성이 높은 반면에 작은 크기의 불규칙적인 쓰기 요청은 자주 갱신되는 데이터인 경우가 많다. 그러므로 본 기법에서는 쓰기 요청의 길이에 따라 동적으로 각 슈퍼 블록의 변환 방식을 변경한다.

4.2 선택적 주소 변환기법에서의 쓰기 작업

슈퍼블록의 주소 변환기법에 따라 쓰기 작업의 처리가 달라진다. 슈퍼블록이 페이지 단위 변환으로 관리될 경우에는 SB$^+$의 쓰기 방법을 사용한다. 그러나 슈퍼블록이 블록 단위 변환으로 관리될 경우 다음과 같은 절차에 따라 쓰기 작업이 처리된다.

데이터 블록의 해당하는 위치에 데이터가 기록되어 있는지 확인한다. 기존 데이터가 기록되어 있다면 일반적인 블록 단위 변환에서 사용하는 쓰기 방법처럼 빈 블록을 할당 받고 기존의 블록에서 유효한 데이터를 빈 블록으로 복사하고 쓰기 작업을 처리한다.

4.3 블록 단위 변환에서 페이지 단위 변환으로 변경

쓰기 요청의 길이가 기준 값(이하 Θ) 이하일 때 해당 슈퍼블록을 페이지 단위 변환으로 관리한다. 블록 단위 변환을 페이지 단위 변환으로 바꾸는 과정은 다음과 같다. 먼저 슈퍼 블록 변환 테이블에서 해당 슈퍼블록 정보를 검색하여 페이지 변환 테이블을 할당하고 각 물리 오프셋을 초기화 시킨다. 그리고 슈퍼 블록 정보의 변환 단위 비트를 페이지 단위로 변경한다.

페이지 변환으로 관리하는 슈퍼블록이 늘어날수록 많은 메모리 공간을 필요하기 때문에 페이지 단위 변환을 사용하는 슈퍼 블록의 개수는 제한(이하 N_{pg})을 둔다. 페이지 변환으로 관리하는 슈퍼 블록의 개수가 N_{pg}과 같을 때 새로운 슈퍼블록이 페이지 단위 변환으로 변경되면 가장 오래전에 접근되었던 슈퍼 블록을 블록 단위 변환으로 변경시키고 새로운 슈퍼 블록의 페이지 단위 변환 테이블을 할당한다.

4.4 페이지 단위 변환에서 블록 단위 변환으로 변경

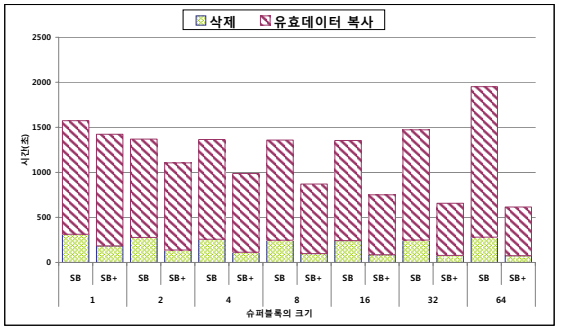
4.3절에서 본 바와 같이 페이지 변환으로 관리하는 슈퍼블록의 개수가 N_{pg}과 같으면 가장 접근이 오래된 슈퍼블록을 블록 단위 변환으로 변경한다. 블록 단위 변환으로의 변경은 다음과 같다. 슈퍼블록의 데이터 블록 개수만큼 새로운 블록을 할당받아 모든 유효한 페이지를 논리적인 순서와 물리적인 순서가 동일하도록 복사한다. 다음으로 기존의 데이터 블록과 로그블록을 모두 삭제한 다음 블록 단위 변환 테이블을 갱신한다. 마지막으로 페이지 단위 변환 테이블을 삭제하고 슈퍼 블록의 정보의 변환 단위 비트를 블록 단위로 변경한다.

5. 실험결과

제안하는 기법을 시뮬레이터로 구현하여 실험하였다. 실험 입력데이터로는 노트북에서 워드프로세서, 동영상 재생, 웹브라우징, 게임 등의 작업을 수행하는 과정에서의 입출력 정보를 수집하였다. 노트북의 운영체제는 마이크로소프트 윈도우XP이고 파일 시스템은 FAT32를 이용하였

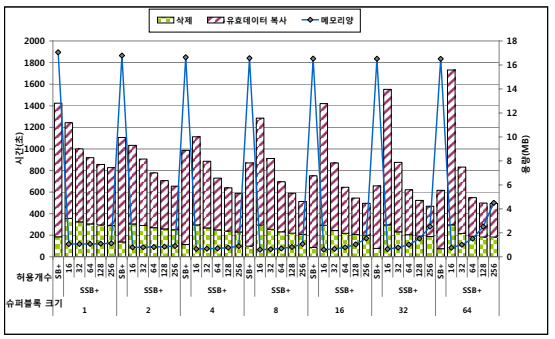
다. 시뮬레이터의 플래시 메모리 크기는 8GB로 설정하여 실험을 진행하였다.

그림 4는 SB와 SB+의 성능을 비교한 것이다. 그림 4와 같이 SB+에서는 슈퍼블록의 크기가 커질수록 성능이 좋아졌다. 그 이유는 SB는 변경되는 데이터가 로그 블록에 기록되는 것과는 달리 SB+의 경우 데이터 블록에 빈 페이지가 있다면 데이터 블록에도 변경된 데이터가 기록될 수 있다. 따라서 슈퍼블록의 크기가 커질수록 데이터 블록의 크기가 커지므로 로그블록이 없어도 데이터 블록 내에서 변경되는 데이터 처리가 가능하므로 합병작업이 적게 발생하여 평균 약 37%의 성능이 향상되었다.



(그림 4) SB와 수정 슈퍼블록 비교

그림 5는 SB+과 SSB+의 성능 비교한 것이다. SSB+에서 θ = 2 로 설정하고, N_{pg} = 16~256개로 변경하며 실험하였다. 좌측 y축은 시간(초)로 성능을 나타내고 우측 y축은 N_{pg}에 따라 필요한 메모리 용량을 나타낸다. 슈퍼블록의 크기가 32이고 N_{pg}가 256개 일 때 가장 좋은 성능이 나타났다. SB+에서 같은 슈퍼블록 크기일 때 성능보다 약 28%의 성능이 향상된다. 그 이유는 선택적 주소변환을 함으로써 유효데이터가 기록된 페이지의 복사가 줄어들어 성능이 향상된 것이다. 하지만 블록 수준 주소 변환의 단점인 블록의 삭제 횟수가 증가하였다.



(그림 5) SB+과 SSB+ 비교

SB+는 슈퍼블록 변환 테이블과 페이지 변환 테이블이 메모리에 존재하므로 약 16MB이상의 메모리 용량을 차지한다. SSB+의 경우 슈퍼블록 변환 테이블과 제한된 페이지 변환 테이블이 메모리에 존재하므로 SB+보다 적은 메모리 용량을 차지한다. 이 때 약 2.5MB의 메모리를 사용하는데 SB+의 16%정도의 메모리 용량으로 SB+의 성능과 비교하여 비슷하거나 그 이상의 성능을 보여줬다.

6. 결론

이 논문에서는 기존의 슈퍼블록 FTL을 수정한 기법을 제시하였다. 기존의 슈퍼블록 FTL에 비하여 슈퍼블록 내부에서 페이지 단위 변환의 장점을 제대로 활용하기 위하여 쓰기 및 가비지 컬렉션의 방법을 수정하였으며 선택적 주소변환을 적용하였다. 제안한 기법의 실험결과 제안한 기법은 적은양의 메모리 공간으로도 좋은 성능을 보여주었다.

향후 연구과제로는 현재 블록 수준 주소 변환에서 페이지 수준 주소 변환으로 기법을 변경하는 기준인 θ 를 입출력 데이터에 따라 동적으로 변경하는 방법과 선택적 주소 변환 기법의 다중채널을 사용하는 FTL에 대한 적용 방법을 계획하고 있다.

참고문헌

[1] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho. "A space-efficient flash translation layer for compactflash systems," IEEE Transactions on Consumer Electronics, 48(2):366 - 375, May 2002.

[2] S. W. Lee, D. J. Park, T. S. Chung, W. K. Choi, D. H. Lee, S. W. Park, H. J. Song. "A log buffer based flash translation layer using fully associative sector translation," ACM Transactions on Embedded Computing System, 6, 3, 2007.

[3] J. U. Kang, H. Jo, J. S. Kim, J. Lee. "A superblock-based flash translation layer for NAND flash memory," International Conference on Embedded Software, 161-170, 2006.

[4] S. Y. Park, W. Cheon, Y. Lee, M. S. Jung, W. Cho and H. Yoon. "A Re-configurable FTL (Flash Translation Layer) Architecture for NAND Flash based Applications," in Proc. of International Workshop on Rapid System Prototyping, pp. 202-208, 2007

[5] S. J. Lee, D. Shin, Y. J. Kim, J. H. Kim. "LAST : Locality-Aware Sector Translation for NAND Flash Memory-Based Storage System," International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED) in conjunction with HPCA-14, 52-59, 2008.