MLC 플래시 메모리에서 LSB 백업을 줄이기 위한 요청 분할 플래시 변환 계층

황연성°, 신동군
성균관대학교 정보통신대학
sami0708@skku.edu dongkun@skku.edu

RSFTL: Request Striping Flash Translation Layer for Reducing LSB Backup in MLC Flash Memory

Yeonseong Hwang°, Dongkun Shin
School of Information and Communication Engineering, Sungkyunkwan University

요 약

낸드 플래시 메모리가 널리 사용됨에 따라 보다 많은 용량을 제공하기 위하여 한 메모리 셀이 1비트를 표현하는 싱글레벨 셀(Single-level cell)보다 메모리 셀 당 2비트의 정보를 저장하여 용량 집적도를 높인 멀티레벨 셀(Multi-level Cell)이 보편적으로 사용되게 되었다. 멀티레벨 셀에 저장되는 2개의 비트는 대응 페이지(paired page) 구조로 쌍을 이룬다. 이러한 구조에서 하나의 비트를 프로그램 하는 경우, 쌍을 이루는 다른 비트도 손상을 입을 수 있는 가능성이 있다. 따라서 페이지를 안전하게 저장하기 위해 LSB 백업이라는 기법이 제안되었다. 그러나 LSB 백업으로 인해 추가적인 쓰기 연산이 필요하여 플래시 메모리에서 최대 15%의 성능저하가 나타난다. 이 논문에서는 블록(block)을 분할하여 요청 단위로 기록하여 LSB 백업 양을 최대로 줄일 수 있는 기법을 제안한다.

1. 서 론

낸드 플래시 메모리는 비휘발성 저장매체로서 전력 소모가 작으며 충격에 강하고 이동성이 좋아서 모바일 기기의 주 저장매체로 널리 쓰이고 있다.

낸드 플래시 메모리는 한 메모리 셀 당 1비트를 저장하는 SLC플래시 메모리와 한 메모리 셀 당 2비트를 저장하는 MLC플래시 메모리로 구분된다. MLC플래시 메모리는 한 셀 내에 두 개의 대응 페이지를 저장하여 프로그램 할 수 있다. 그 중 상위 페이지를 MSB (Most Significant Bit) 페이지, 하위 페이지를 LSB (Least Significant Bit) 페이지라 한다. MLC 기반 메모리는 SLC에 비해 상대적으로 느리고 수명이 짧지만 가격이 싸고 용량이 크기 때문에 최근 대부분의 플래시 기반 시스템들은 MLC 플래시 메모리를 사용하고 있다.

그러나 MLC 기반 메모리는 대응 페이지 간섭이 일어날 수 있다[1]. 두 개의 대응 페이지는 같은 메모리 셀에 있기 때문에, 만약에 MSB 페이지를 프로그램 하던 도중에 전원이 나가는 상황이 발생하게 되면 이미 프로그램 되어 있던 대응 LSB 페이지도 손상을 입게 된다. 따라서 MLC 기반 메모리는 MSB 페이지에 대한 프로그램 수행 중 전원 공급이 중단되더라도 LSB 페이지를 복구할 수 있는 방법을 제공해야 한다. LSB 백업은 MSB 페이지를 쓰기 전에 LSB 페이지를 백업함으로써 전원 공급이 중단되었을 경우 백업한 페이지를 통해 복구가 가능하도록 한다. 하지만 LSB 백업을 위하여 추가적인 쓰기가 필요하다. 그림 1은 LSB 백업으로 인해 성능이 최대 15%가 줄어들었음을 보여준다.

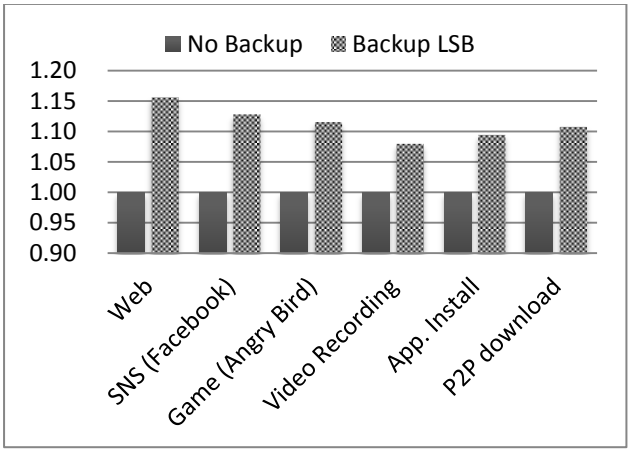

그림 1 FPM에서 LSB 백업에 따른 수행시간 차이

본 논문에서는 페이지 사상을 이용하는 플래시변환계층(FPM, Full Page Mapping)에서 LSB 백업으로 인한 성능 저하를 줄이기 위해 블록을 분할하여 크기가 작은 요청을 따로 기록하는 RSFTL(Request Striping Flash Translation Layer)을 제안한다. 실험결과 LSB 페이지를 최대한 무효화시킴으로써 LSB 백업으로 인한 성능저하를 줄여 쓰기 성능에 효과적임을 확인했다.

· 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. 2012-0006417)

2. 관련연구

2.1 FPM

플래시 변환 계층은 파일 시스템과 플래시 메모리 사이에서 파일 시스템에서 사용하는 논리주소를 플래시 메모리에서 사용하는 물리주소로 사상하는 역할을 한다. 플래시 변환 계층 가운데 FPM은 현재 가장 널리 사용되고 있는 기법으로 모든 페이지의 사상 정보를 사상 테이블에 저장하여 관리한다. 따라서 입력한 데이터를 플래시 메모리 내의 페이지를 선택하여 사상할 수 있기 때문에 유연성이 높다. 하지만 모든 페이지의 사상정보를 저장해야 하므로 메모리 사용량이 크다는 단점이 있다. 그림 2는 FPM에서 쓰기 요청이 들어왔을 때, 데이터를 어떻게 저장하는지 보여준다. FPM은 쓰기 요청을 순차적으로 저장한다. 따라서 같은 요청이 계속 들어오지 않는 이상 LSB 백업을 수행하기 때문에 LSB 백업으로 인한 오버헤드가 크다.

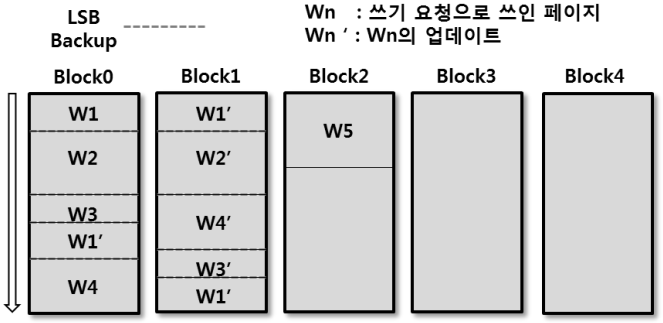

그림 2 FPM에서 쓰기 요청이 들어왔을 경우의 데이터 저장 방법

2.2 MLC 기반 메모리에서의 LSB 백업

MSB 페이지를 프로그램 하는 동안에 대응 LSB 페이지가 손상을 입는 것을 방지하기 위해 MLC 기반 메모리는 LSB 백업 기법을 사용한다. MSB 페이지를 프로그램 하기 전에 LSB 페이지를 백업 블록에 복사해 둠으로써 갑자기 전원이 나갔을 경우 LSB 페이지를 용이하게 복구시킬 수가 있다. MSB 페이지의 프로그램 시간은 LSB 페이지보다 오래 걸리기 때문에 성능 저하를 줄이기 위해 MSB 페이지는 백업하지 않는다. 일반적으로 n번째 페이지와 (n+2)번째 페이지가 대응 페이지이다.

대응 페이지 간섭은 같은 쓰기 요청 내에서는 일어나지 않는다. 쓰기 요청을 처리하는 도중에 전원이 나갔을 경우 부분적으로 쓰여진 데이터들은 무효화되어 다시 쓰여져야 하기 때문에 LSB 백업은 쓰기 요청이 클수록 적게 일어난다. 또한, 대응 MSB 페이지가 쓰여지기 전에 LSB 페이지가 무효화되는 경우에도 LSB 백업을 할 필요가 없다.

3. RSFTL 설계 및 구현

RSFTL은 FPM보다 LSB 백업 양을 줄이기 위해 제안되었다. MSB 페이지가 쓰여지기 전에 LSB 페이지

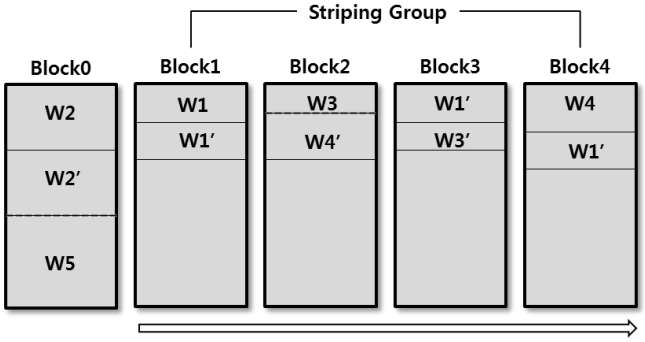

그림 3 RSFTL에서 쓰기 요청이 들어왔을 경우의 데이터 저장 방법

가 무효화되면 LSB 백업을 할 필요가 없다는 점을 이용하여, RSFTL은 블록들을 하나의 블록 그룹으로 묶어서 그룹에 속한 블록들에 요청 단위로 차례대로 쓴다. 블록 그룹의 개수가 늘어날수록, 이전의 요청이 다시 들어왔을 경우에 무효화되어 있을 가능성이 높아진다. 따라서 블록 그룹의 개수가 늘어날수록 성능은 좋아진다. 하지만 아무런 정보 없이 차례대로 쓸 경우, 무효화되는 데이터들이 흩어져서 가비지 컬렉션의 오버헤드가 늘어나게 된다. 따라서 크기가 작은 요청은 시간적 지역성(Temporal locality)이 높다는 점에 착안하여[2], 일정 크기 이하의 쓰기 요청에 한해 블록 그룹에 쓰면, 작고 자주 업데이트 되는 요청들이 블록 그룹에 속하게 되므로, 무효화되는 데이터들을 모을 수 있게 된다. 그리고 일정 크기 이상의 쓰기 요청에 대해서는 그룹으로 묶이지 않은 하나의 블록을 할당하여 기록하고, 블록을 다 쓰게 되면 새로운 블록을 다시 할당 받게 된다. 그림 3은 그림 2에서 FPM과 동일한 쓰기 요청이 차례로 들어왔을 경우, RSFTL에서 데이터를 어떻게 저장하는지 보여준다. W2와 W5는 크기가 큰 요청이므로 그룹으로 묶이지 않은 블록에 저장하고, 나머지 W1, W3, W4에 대해서는 블록 그룹에 차례로 저장한다.

RSFTL의 가비지 컬렉션은 FPM과 동일하게 적용된다. 블록 그룹은 연속하여 저장되지 않기 때문에 각 블록에 대해서 가장 오버헤드가 적은 희생 블록(victim block)을 결정하여 가비지 컬렉션을 수행하게 된다.

4. 실 험

4.1 실험 환경

본 논문의 실험에서는 소프트웨어 시뮬레이터를 만들어 사용하였으며, 대상으로 한 플래시 메모리 스토리지는 다음과 같다.

- Samsung KFXXGH6X4M flash memory[3]
  - 읽기 속도: 403us/page
  - 쓰기 속도: 994us/page
  - 지우기 속도: 872us/block

또한 실험에 사용한 워크로드는 갤럭시S3 스마트폰에서 웹 서핑, 앱 설치, 게임, 카메라, 캠코더 등의 실

제 워크로드를 수집한 것과 8KB의 랜덤 쓰기를 계속하여 발생시킨 워크로드를 사용하였다.

RSFTL의 비교대상으로는 FPM기법을 사용하였으며 LSB 백업을 하지 않은 FPM과 LSB 백업을 하는 FPM, 그리고 RSFTL에 대해서 LSB 백업의 양, 쓰기 성능, 수행시간을 측정하고 이를 통해 가장 좋은 성능을 내는 블록 그룹에 기록할 요청의 크기를 탐색하였다.

4.1 LSB 백업과 수행시간에 관한 실험

그림 4는 갤럭시S3에서 FPM과 RSFTL에 대해 LSB 백업의 양과 수행 시간을 측정한 것이다. LSB 백업을 하는 FPM보다 RSFTL에서 블록 그룹이 64개일때, 최대 1.6%의 성능시간을 줄였고, 전체 LSB 백업의 양은 51%로 줄어들었다. 그림 5는 8KB 랜덤 쓰기 워크로드에서 LSB 백업의 양과 수행 시간을 측정한 것이다. 일반적으로 8KB 이하의 크기로 쓰기 연산을 블록 그룹에 진행했을 때 LSB 백업의 양과 가비지 컬렉션의 오버헤드가 최소화됨을 확인하였다. 또한 8KB 랜덤 쓰기 워크로드에 대해 최대 17%의 성능효과가 있음을 확인하였다.

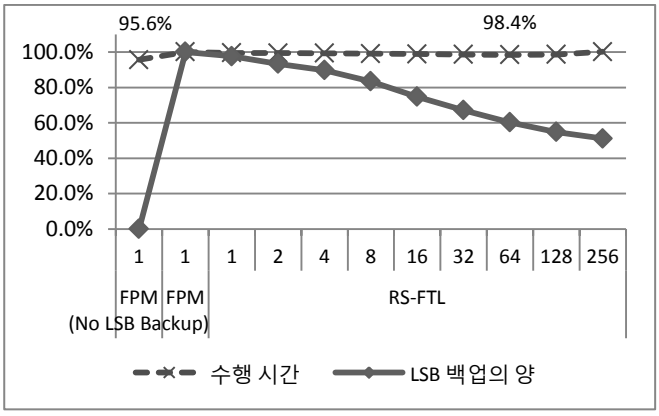

그림 4 gs3_trace에서의 성능

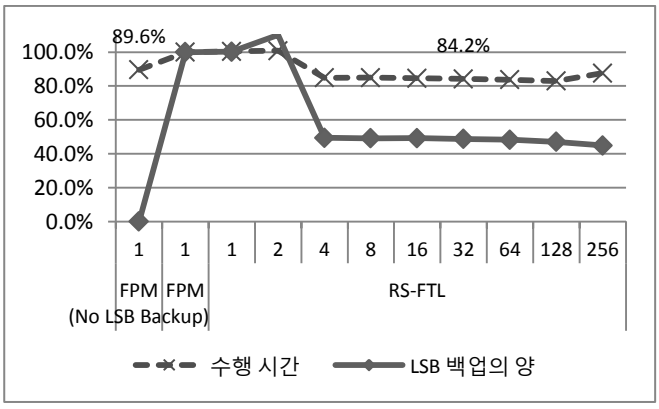

그림 5 syntrace에서의 성능(8KB 랜덤 쓰기)

4.2 쓰기 성능에 관한 실험

그림 6과 7은 각각 갤럭시S3와 8KB 랜덤 쓰기 워크로드에서 쓰기 성능을 측정한 것이다. 8KB 랜덤 쓰기 워크로드에서 블록 그룹이 128일 때, 최대 62%의 성능효과가 있음을 확인하였다.

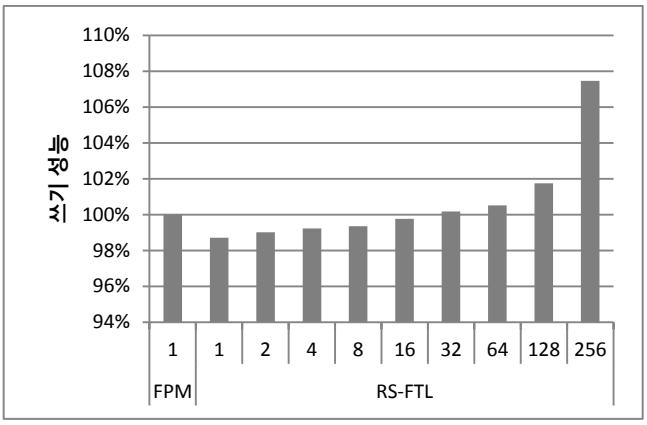

그림 6 gs3_trace에서의 성능

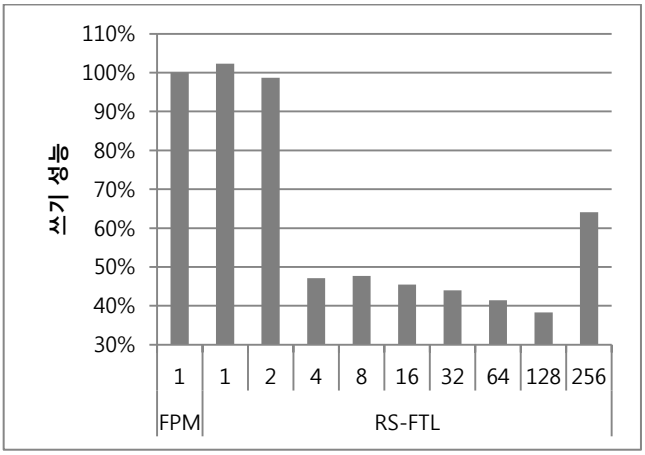

그림 7 syntrace에서의 성능(8KB 랜덤 쓰기)

5. 결 론

우리는 이 논문을 통해 LSB 백업의 양을 줄이는 새로운 FTL을 제안하였다. 그리고 시뮬레이터를 이용하여, LSB 백업의 양, 무효화된 LSB 페이지의 수, 전체 수행시간을 측정하였으며, FPM과 비교하여 쓰기 성능, 수행시간 측면에서 좋은 정책임을 검증하였다.

참고문헌

[1] Lee, Ki Yong, et al. "Design and implementation of MLC NAND flash-based DBMS for mobile devices," in *Journal of Systems and Software*, 2009.

[2] S. Lee et al, “LAST: locality-aware sector translation for NAND flash memory-based storage systems,” in *ACM SIGOPS Operating Systems Review*, 2008.

[3] Lee, Sungjin, et al. “FlexFS: a flexible flash file system for MLC NAND flash memory,” in *Proceedings of the USENIX Annual Technical Conference, San Diego, CA*, 2009.