플래시 메모리 파일 시스템을 위한 효율적인 소거 횟수 평준화 기법

배영현[O]* 최종무** 이동희*** 노삼혁**** 민상렬*
*서울대학교 컴퓨터공학부 **단국대학교 정보컴퓨터학부
***서울시립대학교 컴퓨터과학부 ****홍익대학교 정보컴퓨터공학부
{yhbae[O], symin}@archi.snu.ac.kr, choijm@dku.edu
dhlee@venus.uos.ac.kr, samhnoh@hongik.ac.kr

An Efficient Wear-leveling Scheme for Flash Memory File System

Young Hyun Bae[O]*, Jongmoo Choi**, Donghee Lee***, Sam H. Noh****, Sang Lyul Min*
*School of Computer Engineering, Seoul National University
**Division of Information and Computer Science, Dankook University
***School of Computer Science, University of Seoul
****School of Information and Computer Engineering, Hongik University

요 약

이동 기기의 저장 장치로 널리 사용되는 플래시 메모리는 데이터를 기록하기 전에 해당 블록이 미리 소거되어 있어야 하는 제약이 있다. 또한 각 블록은 소거 횟수의 한계를 가지고 있기 때문에 특정 블록이 집중적으로 사용되는 경우에는 일부 블록의 수명이 일찍 다하게 되어 저장 장치로서의 수명도 짧아지게 된다. 따라서 플래시 메모리 파일 시스템은 고속의 데이터 입출력 성능뿐만 아니라 기록과 소거 동작이 특정 블록에 집중되지 않도록 하여 저장 장치의 내구성을 개선하는 소거 횟수 평준화 기능을 제공해야 한다. 기존에 제안된 소거 횟수 평준화 기법은 복잡한 계산을 필요로 하며 각 블록의 소거 횟수를 유지해야 하는 비용 등으로 인해 자원이 부족한 소형 이동 기기에서 구현하기에는 비효율적이다. 본 논문에서는 플래시 메모리 파일 시스템에서 구현과 동작이 단순하고 어떠한 데이터 접근 형태에 대해서도 평준화 성능이 우수한 효율적인 소거 횟수 평준화 기법을 제안한다. 그리고 제안된 기법을 구현하여 기존 플래시 메모리 파일 시스템의 소거 횟수 평준화 성능과 비교 평가 한다.

1. 서 론

플래시 메모리는 비휘발성 반도체 소자로서 기존 디스크 저장 장치에 비해 빠른 데이터 접근 성능을 보장하며, 부피와 소비 전력이 매우 작다. 또한 외부 충격 등에 대해서 높은 신뢰성을 보장하기 때문에 최근 휴대 전화기나 개인 정보 단말기(PDA)와 같은 소형 이동 기기에서 데이터 저장 장치로 널리 사용되고 있다. 페이지(page)의 집합인 블록(block)으로 구성된 플래시 메모리에 데이터를 기록(program)하기 위해서는 해당 블록이 미리 소거(erase)되어 있어야 하는 제약이 있다. 그리고 소거의 단위인 블록과 기록의 단위인 페이지의 크기가 동일하지 않다. 이에 따라 플래시 메모리에 저장된 기존 데이터를 직접 갱신(in-place update)하는 것이 불가능하기 때문에 변경된 데이터를 기록하기 위해서는 새로운 페이지 혹은 블록을 확보하고 변경된 저장 위치를 갱신하는 작업이 추가로 필요하다. 따라서 플래시 메모리를 기반으로 하는 파일 시스템은 데이터의 논리 주소와 물리적 위치 사이의 관계를 관리하는 고유의 주소 사상 (address mapping) 기법과 자유 블록 확보 (free block reclamation) 기법을 포함하고 있다.

플래시 메모리는 이와 같은 동작 제약뿐만 아니라 각 블록이 소거될 수 있는 한계 횟수를 가지고 있다. 즉, 특정 블록에 대해 기록과 소거 동작이 집중되면 일부 블록의 수명이 일찍 다하게 되어 저장 장치로서의 내구성이 떨어지게 된다. 따라서 플래시 메모리 파일 시스템은 특정 블록이 집중적으로 사용되어 소거 횟수가 급증하지 않도록 소거 횟수 평준화 (wear-leveling) 기능을 제공해야 한다. 기존의 플래시 메모리 파일 시스템에서는 각 블록의 사용 빈도나 소거 횟수에 대한 정보를 바탕으로 자유 블록을 확보하는 과정에서 소거 횟수를 평준화하는 블록 관리 기법을 구현하고 있다. 그러나 기존 소거 횟수 평준화 기법들은 전체 블록을 검색하는 등의 복잡한 계산을 필요로 하며 블록의 소거 횟수와 관련된 정보를 유지해야 한다. 시스템 자원이 비교적 작은 소형 이동 기기에서는 구현이나 동작이 단순하여 소거 횟수 평준화에 따른 성능의 저하를 최소화할 수 있는 소거 횟수 평준화 기법이 필수적이다. 또한 파일 시스템의 파일 접근 형태에 관계없이 효과적으로 소거 횟수를 평준화할 수 있어야 한다.

본 논문에서는 플래시 메모리 파일 시스템의 주소 사상 기법에 따라 관리되는 플래시 메모리의 각 블록 영역을 일정한 주기에 따라 이동시킴으로써 특정 블록에 대해 기록과 소거가 집중되는 현상을 근본적으로 방지하는 효율적인 소거 횟수 평준화 기법을 제안한다. 이 기법에서는 각 블록의 소거 횟수를 유지할 필요가 없고 전체 블록의 일정한 소거 횟수 주기에 따라 평준화 작업을 수행하기 때문에 계산 비용이 최소화된다. 또한 모든 블록이 공평하게 영역 이동의 대상이 되기 때문에 파일 접근 형태에 관계없이 효과적으로 소거 횟수 평준화 기능을 수행한다.

2. 기존 연구

소거 횟수 평준화를 통해 플래시 메모리 저장 장치의 내구성을 개선하려는 연구는 초기의 블록 관리 기법의 개발과 함께 있어 왔다. eNVy 시스템에서는 블록 삭제 비용과 삭제 빈도의 곱으로 표시되는 삭제 색인(cleaning index)을 이용하여 블록 삭제 작업을 수행한다[1]. 이때 삭제 색인이 큰 블록이 우선적으로 선택되기 때문에 삭제 빈도가 큰 블록에 기록과 소거 동작이 집중되는 문제가 발생한다. eNVy 시스템은 소거 횟수가 큰 블록과 소거 횟수가 작은 블록에서 유효한 데이터를 서로 교환함으로써 문제를 해결하였다. 그러나 매번 삭제 작업 때 마다 삭제 색인을 계산하여 전체 블록과 비교해야 하는 계산 비용이 많이 든다. Greedy 정책과 cost-benefit 정책에 따라 소거할 블록을 선택하는 Kawaguchi의 방법에서도 age 값에 의해 사용 빈도가 집중되는 것을 방지하지만 순수한

소거 횟수 평준화를 위한 것이 아니며 cost-benefit 값을 계산하기 위한 비용이 많이 든다[2].

로그 구조 파일 시스템(Log-structured File System, LFS)에 기반한 JFFS2(Journaling Flash File System 2)와 YAFFS(Yet Another Flash File System)는 데이터의 기록과 갱신을 플래시 메모리의 저장 공간에서 순차적으로 수행한다[3, 4]. 그리고 저장 공간의 끝까지 사용한 후에는 처음부터 다시 빈 공간을 확보하면서 데이터를 기록한다. 이러한 LFS 처리 방식은 자연스럽게 플래시 메모리의 전체 블록을 균등하게 사용하기 때문에 우수한 소거 횟수 평준화 기능을 제공한다. 그러나 파일 시스템에서의 파일 접근 형태가 균일하지 않은 경우에는 파일의 갱신 빈도에 따라 해당하는 블록의 소거 횟수의 편차가 커지는 문제가 있다.

3. 플래시 메모리 파일 시스템

본 논문에서는 [5]에서 설계한 트랜잭션 기반 플래시 메모리 파일 시스템(Transactional Flash Memory File System, TFFS)을 위한 효율적인 소거 횟수 평준화 (wear-leveling) 기법을 제안하고 구현한다. TFFS는 그림 1에서 보는 것과 같이 파일 시스템 모듈과 주소 사상 모듈로 구성된 플래시 메모리 파일 시스템이다. 파일 시스템 모듈은 파일 및 디렉터리의 생성, 쓰기, 갱신, 읽기 등의 기능을 사용자에게 제공한다. 주소 사상 모듈은 정적인 주소 사상을 기반으로 플래시 메모리를 관리하여 효율적인 데이터 저장 기능을 제공한다.

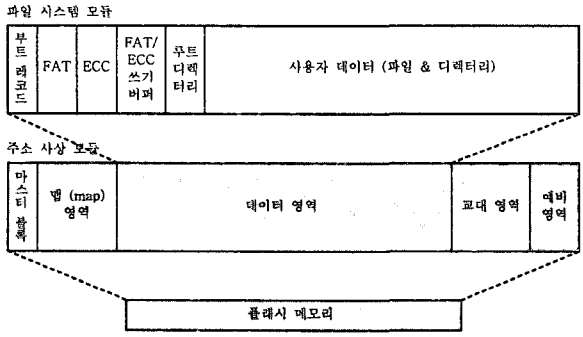

그림 1 TFFS의 구조

그림에서 보는 것처럼 파일 시스템 모듈은 기존 FAT 파일 시스템과 유사한 구조를 가지고 있다. 주소 사상 모듈은 전체 플래시 메모리에 다섯 개의 영역을 블록 단위로 할당하여 파일 시스템 모듈에서 요청하는 데이터 접근을 처리한다. 파일 시스템에서 파일의 새로운 데이터를 기록할 때에는 해당 논리 주소의 기존 데이터를 유지할 필요가 없기 때문에 정적 사상(static mapping)에 따라 할당된 데이터 영역(data area)의 블록을 직접 소거하고 기록한다. 이때는 새로운 블록을 확보하거나 주소 사상 정보를 갱신하는 추가 비용 없이 고속으로 쓰기 요청을 처리할 수 있다. 반면에 파일의 데이터나 파일 시스템의 부가 정보(meta-information)를 갱신할 때는 새로운 블록에 변경된 데이터를 기록한 다음 주소 재사상을 통해 안전하게 처리해야 한다. TFFS는 교대 영역(alternate area)를 이용하여 변경된 데이터를 기록하고 맵 영역에 변경된 주소 사상 정보를 기록하는 교대 사상 (alternate mapping) 기법을 포함하고 있다. 따라서 주소 사상 모듈은 파일 시스템 모듈의 데이터 특성에 따른 제어에 의해 선택적으로 주소 사상 기법을 적용한다. 즉, TFFS는 플래시 메모리 상에서 데이터의 안전한 갱신을 보장함과 동시에 플래시 메모리의 제약에 따르는 추가 비용 없이 고속으로 쓰기 요청을 처리할 수 있다.

이와 같은 주소 사상 기법에 따라 플래시 메모리를 사용하면 특정 영역의 블록들에 대한 사용 빈도가 상대적으로 높아진다. 즉, 교대 사상을 통한 쓰기 요청을 처리하기 위해서 매번 제한된 크기의 교대 영역과 맵 영역의 블록이 사용되기 때문에 데이터 영역의 블록들에 비해 소거 횟수가 상대적으로 커지게 된다. 또한 정적인 사상으로 데이터 영역이 할당되었기 때문에 파일 시스템에서의 데이터 접근 형태에 따라 데이터 블록들 사이의 소거 횟수도 큰 차이를 보일 수 있다. 따라서 TFFS에서는 저장 장치의 내구성 향상을 위한 효율적인 블록 소거 횟수 평준화 기법이 필수적이다.

4. 영역 이동 기법 (area shift scheme)

소형 이동 기기에서 주로 사용되는 플래시 메모리 파일 시스템을 위한 소거 횟수 평준화 기법은 계산 복잡도가 낮고 추가로 유지하는 정보를 최소화하여 효율적으로 구현되어야 한다. 영역 이동 (area shift) 기법은 각 블록의 소거 횟수 정보를 별도로 유지하지 않으며 소거할 블록을 선택하기 위해 복잡한 계산이나 검색이 필요 없는 블록 소거 횟수 평준화 기법이다. TFFS에서 상대적으로 사용 빈도가 높은 맵 영역과 교대 영역의 위치를 주기적으로 이동시킴으로써 특정한 블록들이 맵과 교대 영역으로서 사용이 집중되는 것을 방지한다. 이때 영역 이동을 결정하는 주기는 블록들의 전체 소거 횟수에 의해서만 결정되기 때문에 계산 복잡도와 플래시 메모리에 유지해야 하는 추가 정보는 최소화된다. 또한 영역 이동에 의해 데이터 블록들의 상대적인 위치도 강제적으로 이동되기 때문에 파일 시스템 모듈에서의 데이터 접근 형태와 관계없이 데이터 블록들의 소거 횟수도 균일하게 유지할 수 있다.

영역 이동에 의해 변경되는 교대 영역과 데이터 영역의 위치는 맵 영역에 기록되며, 변경되는 맵 영역의 위치는 별도의 고정된 블록에 기록한다. 이를 위해 TFFS의 주소 사상 모듈의 구조는 그림 2와 같이 두 개의 맵 영역을 할당하여 설계된다. L1 맵 영역의 위치는 고정되며 영역 이동이 수행될 때만 주소 사상 정보와 함께 변경된 맵 영역의 위치 정보가 기록된다. 그리고 L2 맵 영역에는 기존과 같이 교대 사상을 통해 변경되는 주소 사상 정보를 기록한다. 따라서 TFFS가 새로 기동할 때는 L1 맵 영역을 검색하여 현재 L2 맵 영역의 위치를 식별한 다음 L2 맵 영역까지 검색하여 최신의 주소 사상 정보를 읽어낸다. 여기서 L1 맵 영역의 위치는 고정되지만 영역 이동이 수행될 때만 갱신되기 때문에 영역 이동되는 블록들에 비해 소거 횟수가 크지 않게 유지된다.

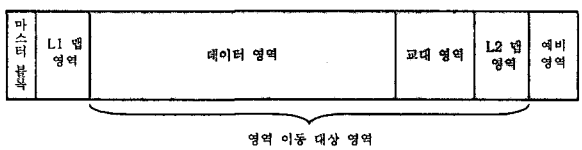

그림 2 영역 이동 기법을 위한 주소 사상 모듈의 구조

그림 3은 TFFS에서 초기화 후 첫 번째 영역 이동이 수행되는 과정을 설명한다. 교대 영역과 L2 맵 영역의 위치가 뒤쪽으로 한 블록씩 이동되며 L2 맵 영역의 마지막 블록은 데이터 영역의 첫 번째 블록으로 이동된다. 이때 데이터 블록은 영역 이동되어 비워질 교대 영역의 첫 번째 블록 위치로 먼저 이동된다. 영역 이동의 대상이 되는 블록들이 사용 중인 경우에는 실제 데이터 복사를 통해 기존 데이터를 유지해야 한다. 마지막으로 변경된 주소 사상 정보와 각 영역의 위치 정보를

L1 맵 영역에 기록한다.

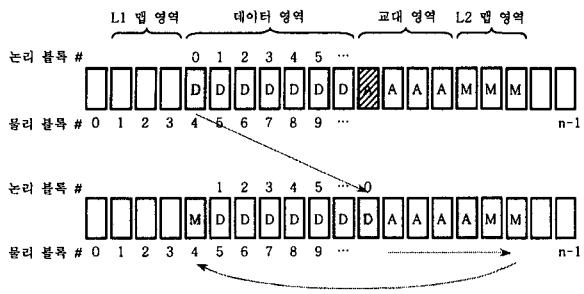

그림 3 영역 이동 기법의 동작

영역 이동을 수행한 후에 블록들의 전체 소거 횟수가 일정한 값에 도달하면 다시 동일한 방법으로 영역 이동을 수행한다. 영역 이동을 결정하는 전체 소거 횟수의 임계값이 작을수록 소거 횟수 평준화 성능은 우수해지지만 영역 이동 빈도가 높아져 파일 시스템의 데이터 기록 성능은 저하된다.

5. 성능 평가

TFFS에 구현된 영역 이동 기법의 소거 횟수 평준화 성능을 비교 평가하기 위하여 기존의 YAFFS와 함께 플래시 메모리 에뮬레이션 환경에서 표 1과 같은 테스트 작업을 수행하였다. 에뮬레이션 되는 플래시 메모리의 크기는 16MB로서 1024개의 블록으로 구성된다. 작업 내용은 다섯 가지 크기 별로 20개씩 100개의 파일을 생성한 후에 파일을 임의로 선택하여 갱신 혹은 삭제/생성 동작을 수행한다. 그리고 백만 번 반복 수행 후 각 블록의 소거 횟수를 측정하였다. 다양한 파일 접근 형태를 반영하기 위하여 동일한 크기의 20개 파일의 선택은 균등 분포(uniform distribution)와 지수 분포(exponential distribution)에 따라 각각 실험을 수행하였다.

표 1 테스트 작업의 구성

파일 크기	1024, 4096, 16384, 65536, 524288
파일 개수	각 크기마다 20개 (총 12MB)
파일 접근 형태	임의로 파일 선택 후 갱신, 삭제, 생성 (동일한 크기의 파일들에 대해 균등 분포 혹은 지수 분포에 따라 선택)
파일 접근 회수	1,000,000 번

그림 4는 동일한 크기의 20개 파일을 균등하게 선택하여 접근한 테스트 작업을 수행한 결과이다. 영역 이동 기법을 적용한 TFFS와 LFS에 기반하여 저널링 방식으로 동작하는 YAFFS 모두 우수한 소거 횟수 평준화 성능을 보여준다. 그러나 그림 5에서 보는 바와 같이 테스트 작업에서의 파일 선택을 지수 분포에 따라 수행하는 경우에 YAFFS는 각 파일의 접근 빈도에 따라 해당 파일이 저장된 블록들의 소거 횟수가 매우 불규칙적으로 나타나는 것을 볼 수 있다. 반면에 TFFS는 여전히 비교적 우수한 소거 횟수 평준화 성능을 보여준다. 즉, 파일 시스템의 파일 접근 형태와 관계없이 우수한 소거 횟수 평준화 기능을 제공한다. 따라서 영역 이동 기법은 저장 장치의 전체 공간에 대해 균등한 사용이 보장되지 않는 실제 응용에서 보다 효과적이다.

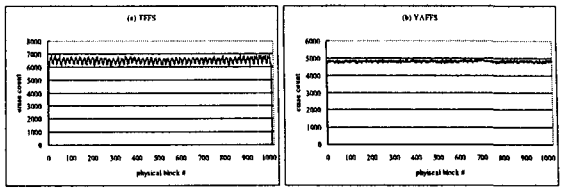

그림 4 블록 소거 횟수 (균등 분포에 따른 파일 접근)

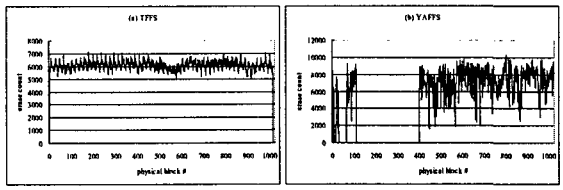

그림 5 블록 소거 횟수 (지수 분포에 따른 파일 접근)

6. 결론

플래시 메모리는 각 블록의 소거 횟수 한계를 가지고 있기 때문에 일부 블록에만 사용이 집중되면 일찍 수명이 다한 블록들로 인해 전체 플래시 메모리의 안정적인 사용이 어렵게 된다. 따라서 플래시 메모리 파일 시스템은 고속의 데이터 저장 기능뿐만 아니라 전체 블록의 소거 횟수를 균등하게 유지하여 저장 장치의 내구성을 개선하는 소거 횟수 평준화 기법을 제공해야 한다. 본 논문에서 제안한 영역 이동 기법은 기존 기법들에 비해 평준화 작업을 수행하는 계산 복잡도가 매우 낮고 플래시 메모리에 추가로 유지하는 정보가 최소화된 효율적인 소거 횟수 평준화 기법이다. 또한 전체 데이터 블록이 영역 이동의 대상이 되기 때문에 파일 시스템의 데이터 접근 형태에 관계없이 효과적인 소거 횟수 평준화 기능을 제공한다.

참고문헌

[1] M. Wu, and W. Zwaenepoel, "eNVy: A Non-Volatile, Main Memory Storage System," In *Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-6)*, pp. 86-97, Oct. 1994.

[2] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Flash-Memory Based File System," In *Proceedings of the USENIX 1995 Winter Technical Conference*, pp. 155-164, Jan. 1995.

[3] D. Woodhouse, "JFFS: The Journalling Flash File System," *Ottawa Linux Symposium*, 2001.

[4] Aleph One Company, "YAFFS (Yet Another Flash File System)," http://www.aleph1.co.uk/yaffs/

[5] Y. H. Bae, J. Choi, D. Lee, S. H. Noh, and S. L. Min, "Design and Implementation of Flash Memory File System for Mobile Devices," Technical Report, School of Computer Engineering, Seoul National University, June 2004.