플래시 메모리 파일 시스템 기술 동향 분석

이현구, 김정훈, 엄영익
성균관대학교 정보통신대학
e-mail:hyungooo, myhuni20, yieom@skku.edu

A Technical Trend Analysis of Flash Memory File System

Hyun-Ku Lee, Junghoon Kim, Young-Ik Eom
College of Information and Communication Engineering, Sungkyunkwan University

요 약

현재 낸드 플래시 메모리는 크기가 작고 충격에 강하며 소비전력이 적어 스마트 디바이스의 메인 스토리지로 활용되고 있다. 하지만 읽기, 쓰기 동작 외에 소거 동작의 필요와 메모리 접근 단위의 차이, 쓰기 횟수의 한계 등의 단점이 존재하며 이를 사용하는 파일 시스템은 기존의 파일 시스템과는 다른 복잡한 연산과 다양한 접근 방식이 요구되어지고 있다. 본 논문에서는 이러한 낸드 플래시의 제약을 극복하기 위해 연구되어온 파일시스템들의 특성과 차세대 메모리로 불리는 SCM의 적용으로 얻을 수 있는 장점을 분석하고 향후 연구방향에 대해 모색하도록 한다.

[1]1. 서론

기술의 발전과 더불어 현대인들의 필수품이 되어가고 있는 스마트 디바이스는 날이 갈수록 성능은 좋아지고 크기는 작아지고 있다. 이런 제품들의 이동 가능한 특성상 크기는 작고 충격에 강하고, 소비전력이 적은 스토리지가 필요한데 낸드(NAND) 플래시 메모리는 이러한 요구사항에 적합하고 가격 또한 비교적 저렴하여 현재 가장 널리 사용되고 있는 스토리지 중 하나가 되었다.

낸드 플래시 메모리는 데이터에 랜덤 액세스가 가능하고, 읽고 쓰는 속도도 하드 디스크에 비해 몇 배나 빠른 장점이 있다. 하지만 일반적인 블록 장치와는 달리 읽기, 쓰기 동작 이외에 소거 동작이 필요하며, 읽기 쓰기 동작의 경우 일반적인 블록장치와 같은 페이지 단위로 이루어지는데 반해, 소거 동작은 플래시 메모리의 물리적 특성상 페이지보다 큰 단위인 블록 단위로 이루어져야 한다. 또한 한번 쓰기 연산을 수행한 영역에 덮어 쓰기가 불가능하며 각 블록 당 최대 삭제 횟수가 100만 번 내외로 제한된다.

낸드 플래시 고유의 특성들로 인하여 스마트 디바이스에서 사용하는 파일 시스템은 그동안 사용하였던 블록장치의 파일 시스템보다 복잡한 연산과 다양한 접근 방식이 요구되어지고 있다.

또한 낸드 플래시의 단점을 보완할 수 있는 스토리지 클래스 메모리(Storage Class Memory, 이하 SCM)도[1] 현재 차세대 메모리로 불리며 주요 메모리 제조사에 의해 상용화 단계에 도입하였고 이에 따라 스마트 디바이스 파일 시스템 연구의 방향도 변해가고 있다.

본 논문에서는 플래시 메모리를 위한 파일 시스템 구현 기법과, 현재 상용화중인 SCM과 낸드 플래시 메모리의 특성을 2장에서 설명하고, 3장에서 관련 연구를 소개한 뒤, 4장에서 비교 및 향후 연구 방향을 제시하도록 하겠다.

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10041244, 스마트TV 2.0 소프트웨어 플랫폼]

2. 배경 지식

2.1 플래시 메모리를 위한 파일 시스템 구현 방법

플래시 메모리를 위한 파일 시스템 구현 기법은 크게 FTL(Flash Translation Layer)을 이용한 기존의 파일 시스템을 사용하는 방법과 MTD(Memory Technology Devices)등을 이용하여 플래시 메모리에 특화된 파일 시스템을 사용하는 방법으로 나눌 수 있다.

FTL은 플래시 메모리와 블록 디바이스의 중간에 위치하는 레이어로 데이터의 논리적인 주소와 플래시 메모리의 물리적인 주소의 사상 관계를 저장하고 재사상 함으로써(adress remapping) 플래시 메모리를 블록장치처럼 사용할 수 있게 해주는 기법이다. 주로 Ext4 파일 시스템을 FTL과 함께 사용한다.

플래시 메모리에 특화된 파일 시스템을 사용하는 방법은 리눅스의 경우, MTD를 사용하여 플래시 메모리를 디바이스 장치로 접근하고 LFS(Log-struc- tured File System)의 기법 등을 도입한 JFFS2, YAFFS2 등의 파일 시스템을 플래시 메모리의 메인 파일 시스템으로 사용하는 방법이다. 이 때 LFS는 플래시 디바이스 자체에 트리 구조를 유지하고 플래시 메모리에 대한 갱신 연산을 다른 빈 페이지에 기록하여 플래시 메모리상의 지움 연산을 최소화 하는 방식으로 플래시 메모리의 성능과 수명을 증가시켜주는 기능을 제공한다.

2.2 SCM과 낸드 플래시 메모리의 특성

비휘발성 속성과 DRAM 또는 SRAM의 고속 바이트 단위 랜덤 접근을 지원하는 SCM은 현재 주요 메모리 제조사들에 의해서 PCM(또는 PRAM), FeRAM, MRAM 등이 주도적으로 연구 개발, 상용화되고 있다.

SCM의 특성을 낸드 플래시 메모리, SDRAM과 비교하면 표 1에서와 같이 나타낼 수 있다.

<표 1> Flash, RAM, SCM 메모리 특성

		Flash	RAM	SCM	
		NAND	SDRAM	PCM	FeRAM
휘발성 여부		비휘발성	휘발성	비휘발성	비휘발성
메모리 접근 단위	읽기/쓰기	페이지	바이트	바이트	바이트
	소거	블록	-	-	-
메모리 접근 속도	읽기	12us	50~75ns	DRAM comparable	110ns
	쓰기	200us	50~75ns	Not match DRAM	110ns
	소거	2ms	-	-	-
내구성(쓰기 횟수)		10^5	10^{15}	$<10^{12}$	10^{12}

3. 관련 연구

3.1 JFFS2(Journaling Flash File System 2)

JFFS2는[2] LFS의 원리를 플래시 메모리 파일 시스템 설계에 적용하여 구현한 파일 시스템으로 낸드 플래시 메모리의 특성을 고려한 파일 시스템 데이터 갱신 작업을 수행한다. 파일 시스템의 변경사항을 '로그'로써 '노드'에 기록하여 저널링이 가능하지만 전체 페이지 이용 후 자유 블록을 확보하는 비용이 크고, 메모리 사용량도 많으며, 파일 시스템이 마운트 될 경우 전체 저장 공간을 모두 검색하여 사상 정보를 재구축하기 때문에 메모리의 용량에 따라서 마운트 시간이 길어지게 된다.

3.2 YAFFS2(Yet Another Flash File System 2)

LFS에서 제안된 out-of-place 업데이트 기법을 도입한 YAFFS2는[3] 업데이트 요청이 발생했을 때, 새로운 페이지를 할당받아 업데이트된 데이터를 기록하고, 기존의 페이지는 무효화 시킨 뒤, 매핑 정보를 갱신한다. 무효화된 페이지는 추후 가비지 컬렉션(garbage collection) 작업을 통해 다시 할당 가능한 상태로 바뀌며, 체크 포인트 기능을 추가해 마운트 수행 시 미리 저장된 램 트리 구조를 불러와 전체 시스템을 스캔하지 않고도 파일 시스템 자료구조를 구축할 수 있다. 하지만 마운트 해제가 정상적으로 되지 않은 경우 체크 포인트가 무시되며 결과적으로 파일 시스템 자료구조를 구축하기 위해 메모리 전체를 읽어야 하는 단점이 있다.

3.3 CFFS(Core Flash File System)

파일 시스템을 마운트 할 경우 링크로 연결된 메타 데이터만 읽도록 하여 빠른 초기화가 가능하도록 고안한 CFFS는[4] YAFFS를 개량한 버전으로 메타 데이터 링크 정보가 현재 데이터 상태와 일치한다면 초기화 시간이 비약적으로 빨라지지만, 비정상적인 종료로 인해 불일치한 링크가 생길 경우 CFFS는 데이터 또는 아이노드 블록을 식별할 수 있는 식별 플래그가 없으므로 모든 플래시 영역을 스캔하여 메타 데이터 링크를 구성하고 파일 시스템 자료구조를 새로 구축하여야 한다.

3.4 PFFS(Proposed Flash File System)

파일 시스템 초기화시 파일 시스템 자료구조의 구축에 소요되는 시간과 자주 변경되는 비교적 작은 크기의 메타 데이터를 효율적으로 관리하는 방법을 SCM(PRAM)을 이용하여 해결하도록 고안한 PFFS는[5] 데이터는 낸드 플래시 메모리에, 메타 데이터는 PRAM에 저장하여 잦은 메타데이터 정보의 변경에 따른 낸드 플래시 메모리의 소거 연산을 줄이고 파일 시스템 마운트 시 PRAM에 있는 메타 데이터만 읽으면 되어 초기화가 빠른 장점이 있다.

<표 2> 스마트 디바이스 파일 시스템 비교 분석

	JFFS2	YAFFS2	CFFS	PFFS
시스템 자료구조 생성 방법	모든 노드를 한번 씩 읽음	체크 포인트를 이용하여 자료구조만 읽음	메타 데이터의 링크를 참조	PRAM의 메타 데이터만 읽음
메타 데이터 관리 방법	데이터와 함께 관리	데이터와 함께 관리	데이터와 함께 관리	PRAM에서 따로 관리
저널링 지원 방법	변경사항을 노드로 관리	변경사항을 트리형식으로 저장	변경사항을 트리형식으로 저장	변경사항을 트리형식으로 저장
비정상적 시스템 종료 시	모든 노드를 한번 씩 읽음	모든 노드를 한번 씩 읽음	모든 노드를 한번 씩 읽음	PRAM의 메타 데이터만 읽음

4. 비교 평가 및 향후 연구방향

표 2에서와 같이 낸드 플래시 메모리만을 사용하는 파일 시스템은 메타 데이터를 데이터와 함께 관리하므로 링크를 이용하여 메타 데이터끼리 연결을 시키거나 구축된 파일 시스템 자료구조를 따로 저장하여 빠른 초기화가 가능하도록 해주는 기능을 갖추고 있다. 하지만 비정상적 시스템 종료 시에는 이러한 정보들이 완벽히 저장되지 않아 다시 모든 노드를 재스캔 하여 시스템 정보를 재구축 하여야만 한다. 이러한 단점을 극복하기 위해 SCM에 메타 데이터를 따로 관리하여 비정상적인 시스템 종료 시에도 메타 데이터 영역만 따로 읽어 빠른 초기화가 가능한 파일 시스템이 연구되어지고 있으나 비교적 높은 가격으로 인한 PRAM의 용량제한에 따른 여러 가지 문제점이 있어 이를 효율적으로 해결 할 수 있는 파일 시스템이 향후 연구되어야 하겠다.

참고문헌

[1] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," IBM Journal of Research and Development, vol. 52, no. 4/5, 2008.

[2] David Woodhouse. "JFFS: Journaling Flash File System" In Proceedings of the Ottawa Linux Symposium(OLS), Red Hat, Inc., 2001.

[3] YAFFS, A Flash file system for embedded use. http://www.yaffs.net/.

[4] Seung-Ho Lim, and Kyu-Ho Park, "An Efficient NAND Flash File System for Flash Memory Storage," IEEE trans. on computer, vol.55, no.7, pp.906-912, Jul. 2006.

[5] Y. Park et al., "PFFS: A Scalable Flash Memory File System for the Hybrid Architecture of Phase Change RAM and NAND Flash," In Proceedings of the ACM Symposium. Applied Computing, 2008.