2006 한국컴퓨터종합학술대회 논문집 Vol. 33, No. 1(A)

임베디드 기기를 위한 NAND 플래시 파일 시스템의 설계

박송화[O] 이태훈 정기동
부산대학교 컴퓨터공학과
{downy25[O], withsoul}@melon.cs.pusan.ac.kr, kdchung@pusan.ac.kr

Design of a NAND Flash File System for Embedded Devices

Song-Hwa Park[0], Tae-Hoon Lee, Ki-Dong Chung
Dept. of Computer Engineering, Pusan National University

요 약

본 논문은 NAND 플래시 메모리를 기반으로 한 임베디드 시스템에서 빠른 부팅을 지원하는 파일 시스템을 제안한다. 플래시 메모리는 비휘발성이며 기존의 하드디스크와 같은 자기 매체에 비해서 크기가 작고 전력소모도 적으며 내구성이 높은 장점을 지니고 있다. 그러나 제자리 덮어쓰기가 불가능하고 지움 연산 단위가 쓰기 연산 단위보다 크다. 또한 지움 연산 횟수가 제한되는 단점이 있다. 이러한 특성 때문에 기존의 파일 시스템들은 갱신 연산 발생 시, 갱신된 데이터를 다른 위치에 기록한다. 따라서 마운팅 시, 최신의 데이터를 얻기 위해 전체 플래시 메모리 공간을 읽어야만 한다. 이러한 파일 시스템의 마운팅 과정은 전체 시스템의 부팅 시간을 지연시킨다. 본 논문은 임베디드 시스템에서 빠른 부팅을 제공할 수 있는 NAND 플래시 메모리 파일 시스템의 구조를 제안한다. 제안된 시스템은 플래시 메모리 이미지 정보와 메타 데이터 블록만을 읽어 파일 시스템을 구축한다. 메타 데이터가 데이터 위치를 포함하기 때문에 마운팅 시, 전체 플래시 메모리 영역을 읽을 필요가 없으며 파일 데이터 위치 저장을 위한 별도의 자료 구조를 RAM 상에 유지할 필요가 없다. 실험 결과, YAFFS에 비해 76%~85% 마운팅 시간은 감소시켰다. 또한 YAFFS에 비해 64%~75% RAM 사용량을 감소시켰다.

1. 서 론

임베디드 시스템(Embedded system)은 빠른 부팅 시간을 제공해야 한다. 최근 임베디드 시스템의 사용이 증가함에 따라 NAND형 플래시메모리를 저장장치로 사용하는 사례가 증가하고 있다. 이는 플래시 메모리는 물리적인 충격에 강해 휴대가 용이하며 집적도가 높아 소형화 기기에 적합하고, 전력을 적게 소모해 동일 배터리 파워로 장시간 사용할 수 있는 특징을 가졌기 때문이다[1]. 그러나 부팅과정에서 수행되는 플래시 메모리 파일 시스템의 마운팅(mounting) 과정은 전체 부팅 과정에서 비교적 긴 시간을 차지한다. 즉, 임베디드 시스템의 빠른 부팅을 위해서는 플래시 메모리 파일 시스템을 빠르게 마운팅하는 기법을 연구 및 개발할 필요가 있다.

이러한 긴 마운팅 시간은 플래시 메모리의 하드웨어적인 특성과 로그구조를 기반으로 한 파일 시스템의 특성에서 기인한다[2]. 플래시 메모리에서는 쓰기 연산을 위해서는 삭제 연산이 선행되어야 한다. 이 때, 삭제 연산의 단위는 쓰기 연산의 단위인 페이지보다 큰 블록 단위로 이뤄진다. JFFS2[3]나 YAFFS[4]와 같은 기존의 플래시 파일 시스템들은 이러한 제약사항을 극복하기 위해 LFS(Log-Structure File System)[5]에 기반을 두어 외부 갱신(out-place-update) 기법을 통해 해결한다. 즉, 갱신 연산이 발생하면 기존의 데이터 위치에 갱신된 데이터를 기록하지 않고 다른 곳에 갱신된 데이터를 기록한다. 따라서 마운팅 과정에서 가장 최신에 갱신된 데이터를 얻기 위해서는 전체 플래시 메모리 영역을 읽어야 한다.

이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음.

이러한 문제점을 해결하기 위해 본 논문에서는 임베디드 기기를 위한 파일 시스템을 설계한다. 제안하는 파일 시스템은 플래시 메모리 이미지를 언마운팅 시점에 저장하고 마운팅 시, 플래시 메모리 이미지만을 읽어 빠른 마운팅을 지원한다.

본 논문의 구성은 다음과 같다. 2장에서는 플래시 메모리의 특성과 기존의 NAND 플래시 파일 시스템인 YAFFS에 대해 기술한다. 3장에서는 임베디드 기기를 위한 NAND 플래시 메모리 파일 시스템에 대해 설명하며, 4장에서는 제안된 파일 시스템의 성능을 평가하고 5장에서 이 논문의 결론과 향후 과제를 제시한다.

2. 관련연구

플래시 메모리는 EEPROM의 일종으로 셀(cell)을 구성하는 구조에 따라 크게 NOR 플래시 메모리와 NAND 플래시 메모리로 구분된다.

NOR 플래시 메모리는 버스형태의 외부 인터페이스를 가지고 있으므로 임의 접근이 가능하고, CPU에 직접 연결되어 CPU가 수행하는 코드(code)를 직접 실행하는 것도 가능하다. 또한 바이트 단위 프로그래밍이 가능하며 빠른 속도의 접근을 제공하므로 주로 코드를 저장하고 실행하는 용도로 사용된다.

NAND 플래시 메모리는 NOR 플래시 메모리에 비해 블록단위의 삭제 연산속도가 빠르며 쓰기 연산의 속도 또한 NOR 플래시 메모리에 비해 빠르다. 그러나 임의 접근이 느리고 읽기/쓰기 연산이 페이지 단위로 이루어진다는 특성을 가진다. NAND 플래시 메모리는 느린 임의 접근 때문에 음악 파일이나 이미지 파일 등 대용량의 데이터를 저장하는 용도로 주로 이용된다[1].

최근의 많은 응용기기들은 대용량의 저장시스템을 필요로 하는 경우가 많다. NOR 플래시 메모리는 집적도가 낮고 고가이

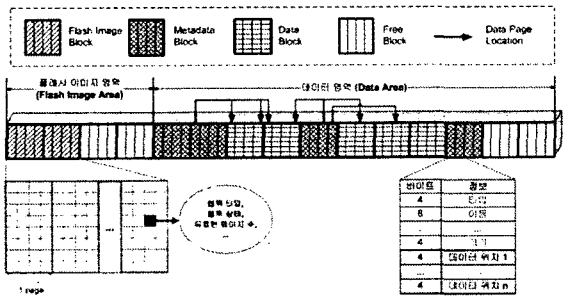
기 때문에 이러한 요구에 적합하지 못하다. 반면에 NAND 플래시 메모리는 단일 칩으로도 대용량이며 비용역시 저렴하기 때문에 NAND 플래시 메모리를 이용하여 응용기기를 만들게 되면 저장용량이 크게 증가하고, 기기의 비용도 낮출 수 있다. 이러한 이유로 NAND 플래시 메모리의 용도가 확대됨에 따라 NAND 플래시의 기술이 활발히 연구되고 있으며, 시장점유율도 증가하고 있다[6].

NAND 플래시 메모리를 위한 대표적인 파일 시스템으로는 YAFFS[4]가 있다. YAFFS는 LFS를 기반으로 하여 플래시 메모리에 대한 갱신 연산을 추가 연산으로 변형하여 처리하는 외부 갱신 기법을 사용한다. 이를 통해 플래시 메모리의 제자리 덮어쓰기(in-place-update)가 되지 않는 문제점을 해결하였다. 하지만 갱신된 데이터가 다른 곳에 기록됨으로써 기존의 데이터가 플래시 메모리 공간에 그대로 남아있게 된다. 또한 갱신 연산이 빈번할 경우, 한 파일에 대한 데이터가 플래시 메모리 공간에 흩어지게 된다. 따라서 YAFFS의 경우, 마운팅 시점에 전체 플래시 메모리 공간을 읽어야만 데이터의 위치를 파악할 수 있으며 최신의 데이터를 얻을 수 있다. 마운팅 과정에서 찾은 각 파일 데이터의 위치는 Tnode라는 자료 구조 형태로 RAM 상에 저장되며 Tnode들은 하나의 트리를 구성하여 파일 연산 시에 사용된다. 이러한 과정은 YAFFS의 마운팅 시간을 크게 지연시키는 요인이다. 또한 마운팅 과정에서 파일 데이터의 위치를 트리 구조로 구성하여 RAM 상에 유지함으로써 파일의 수가 증가하면 메모리 사용량이 급증하게 된다. 다음 장에서는 이러한 문제점을 해결하고 그 외의 몇 가지 새로운 장점을 갖는 NAND 플래시 메모리 파일 시스템에 대해 기술한다.

3. 플래시 메모리 기반의 파일 시스템의 설계

임베디드 시스템은 빠른 부팅시간을 제공해야 하지만 부팅 과정에서 수행되는 플래시 파일 시스템의 마운팅은 비교적 긴 시간을 요구한다. 본 논문에서는 빠른 마운팅을 지원하는 NAND 플래시 파일 시스템의 구조를 제안한다.

3.1 시스템 구조



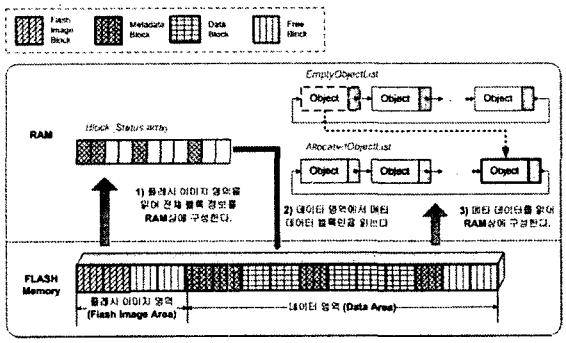
[그림 1] 제안된 플래시 메모리 파일 시스템의 구조

[그림 1]과 같이 전체 플래시 메모리 공간을 플래시 이미지 영역과 메타 데이터와 데이터를 저장하는 데이터 영역으로 나눈다. 플래시 이미지 영역은 고정되어 있으며 라운드 로빈 방식으로 사용된다. 메타 데이터와 데이터는 플래시 이미지 영역을 제외한 데이터 영역에 저장된다.

데이터 영역의 각 블록은 타입을 가진다. 블록이 가질 수 있는 타입은 메타 데이터 타입과 데이터 타입이다. 디렉토리, 하드 링크와 심볼릭 링크 생성 시, 메타 데이터가 메타 데이터 블록에 저장된다. 파일 생성 시에는 먼저 데이터가 데이터 블록에 저장되고 데이터 위치를 포함한 메타 데이터가 메타 데이터 블록에 저장된다. 따라서 YAFFS와 달리 제안된 파일 시스템에서는 Tnode와 같은 파일 위치를 저장하는 별도의 자료구조를 포함하지 않는다.

3.2 빠른 마운팅 기법

제안된 파일 시스템은 언마운팅 시점에 전체 플래시 이미지를 플래시 이미지 영역에 저장한다. 플래시 이미지는 각 블록의 타입, 블록 상태 및 유효한 페이지 수 등 마운팅 과정에서 구축해야 하는 정보들을 포함하고 있다. 마운팅 시, 언마운팅 시점에 저장한 플래시 이미지 정보와 데이터 영역의 메타 데이터 블록만을 읽어 파일 시스템을 구축하게 된다.



[그림 2] 마운팅 과정

[그림 2]는 제안된 파일 시스템에서의 마운팅 과정을 보여주고 있다. 파일 시스템 마운팅 시, 먼저 플래시 이미지 영역에서 가장 최근의 플래시 이미지 정보를 읽어온다. 읽어 온 플래시 이미지 정보는 RAM 상에 블록 정보 배열로 저장된다. 블록 정보 배열은 마운팅 과정에서 뿐만 아니라, 공간 할당 및 garbage collection 등에도 사용된다. RAM 상의 블록 정보를 사용해서 데이터 영역의 메타 데이터 블록들만 읽게 되며, 읽은 메타 데이터는 RAM 상에 오브젝트 리스트로 관리된다. 마운팅 과정에서 각 블록의 상태 변화는 블록 정보 배열에 반영되며, 언마운팅 시점에 블록 정보 배열이 플래시 이미지 정보로 저장된다. 언마운팅 시점에 저장된 플래시 이미지 정보는 다음 마운팅 시점에 사용된다.

4. 실험 및 성능 평가

4.1 실험 환경

연산 종류	읽기	쓰기	지우기
연산 단위	512 B	512 B	16 KB
속도	15 μs	2001 μs	2 ms

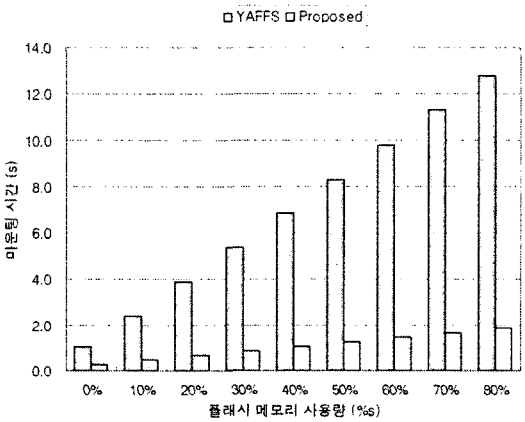
[표 1] NAND 플래시 메모리의 I/O 특성

제안하는 파일 시스템의 성능은 PXA255 200MHz를 장착한 임베디드 보드에서 동작하는 리눅스 시스템을 바탕으로 YAFFS

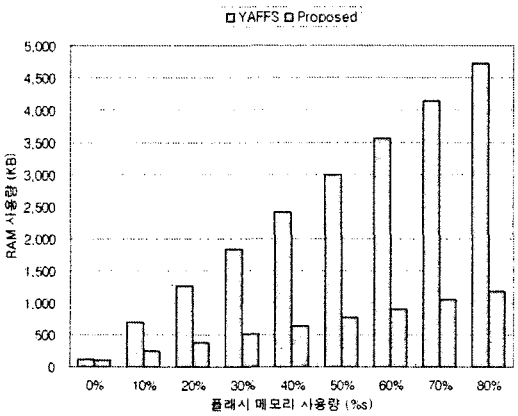
와 비교 평가한다. 실험에 사용된 NAND 플래시 메모리는 64MB이며 특성은[표 1]과 같다. 실험에 사용한 데이터는 [7]의 UNIX 파일 크기 분포를 참고하여 생성되었다.

4.2 실험 결과

본 논문에서는 빠른 마운팅 시간을 제공하는데 중점을 두기 때문에 제안한 시스템의 성능평가를 위해 플래시 메모리 사용량에 따른 마운팅 시간을 기준으로 삼는다. 또한 제안된 시스템에서 데이터 저장을 위한 별도의 자료 구조를 RAM 상에 유지하지 않음으로써 얻는 RAM 사용량의 감소 효과도 실험을 통해 알아본다.



[그림 3] 플래시 메모리 사용량에 따른 마운팅 시간



[그림 4] 플래시 메모리 사용량에 따른 RAM 사용량

[그림 3]은 플래시 메모리 사용량에 따른 마운팅 시간을 보여준다. YAFFS의 경우, 마운팅 시점에 전체 플래시 공간을 읽어 파일 시스템을 구축한다. [그림 3]에서 플래시 메모리의 사용량에 비례하여 YAFFS의 마운팅 시간이 증가하는 것을 볼 수 있다. 제안된 기법의 경우, 플래시 메모리 이미지를 사용하여 메타 데이터 블록만을 읽음으로 YAFFS보다 마운팅 시간이 감소된 것을 볼 수 있다. 전체적으로 약 76%~85% 마운팅 시간이 감소되었다.

제안하는 파일 시스템은 메타 데이터가 파일 데이터의 위치를 포함하고 있기 때문에 데이터 위치 저장을 위한 별도의 자료 구조가 필요하지 않다. YAFFS의 경우, 마운팅 과정에서 파일 데이터의 위치를 Tnode 형태로 RAM 상에 구성한다. [그림 4]에서 YAFFS의 경우 파일 생성으로 플래시 메모리 사용량에 증가함에 따라 Tnode가 생성되어 RAM 사용량이 증가하는 것을 볼 수 있다. 반면, 제안하는 시스템에서는 파일 생성으로 플래시 메모리 사용량이 증가하여도 파일 데이터 위치 저장을 위한 자료 구조를 생성하지 않으므로 YAFFS에 비해 RAM 사용량이 64%~75% 감소되었다.

5. 결론 및 향후과제

본 논문에서는 임베디드 기기를 위해 빠른 마운팅을 지원하는 파일 시스템을 제안하였다. 제안한 플래시 파일 시스템은 전체 플래시 메모리 공간을 플래시 메모리 영역과 데이터 영역으로 나누어 관리하며, 마운팅 시점에 플래시 메모리 영역에서 최신의 플래시 메모리 정보과 메타 데이터 블록만을 읽어 파일 시스템을 구축한다. 실험 결과, YAFFS에 비해 약 64%~76% 마운팅 시간이 감소하였다. 또한 메타 데이터가 데이터 위치 정보를 포함함으로써, 데이터 위치 저장을 위한 자료 구조를 구축하지 않아 YAFFS에 비해 64%~75% RAM 사용량이 감소되었다.

임베디드 시스템의 경우, 갑작스런 시스템 폴트가 발생할 수 있다. 따라서 비정상적인 시스템의 종료에 대비한 저널링 기법 및 빠른 복구 기법이 연구되어야 할 것이다.

참고문헌

[1] F. Douglis, R. Caceres, F. Kaasho, K. Li, B. Marsh, and J.A. Tauber, "Storage Alternatives for Mobile Computers," *In Proceedings of the 1st USENIX Symposium on Operating System Design and Implementation*, pp. 25-37, 1994.

[2] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to Flash Memory," *In Proceedings of the IEEE*, Vol. 91, No. 4, pp. 489-502, April 2003.

[3] D. Woodhouse, "JFFS: The Journaling Flash File System," *In proceedings of the Ottawa Linux Symposium (OLS)*, RedHat Inc., 2001.

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

[5] M.Resenblum and J.K.Ousterhout, "The Design and Implementation of a Log-Structured File System", ACM Transaction on Computer Systems, Vol.10, pp.26-52, 1992

[6] 김정기, 박승민, 김채규, "정보가전을 위한 플래시 파일 시스템에서 등급별 지움 정책과 오류 복구 방법", *제5회 차세대 통신 소프트웨어 학술대회(NCS2001)*, pp.938-941, 2001.

[7] G. Irlam, "Unix File Size Survey," http://www.base.com/gordoni/gordoni.html