메타데이타를 비휘발성 램에 유지하는 플래시 파일시스템의 설계 및 구현

(Design and Implementation of the Flash File System that Maintains Metadata in Non-Volatile RAM)

도 인 환 † 최 종 무 †† 이 동 희 ††† 노 삼 혁 ††††

(In Hwan Doh) (Jongmoo Choi) (Donghee Lee) (Sam H. Noh)

요 약 램의 특성과 비휘발성 저장매체의 특성을 동시에 가지고 있는 비휘발성 램을 시스템 소프트웨어에서 효과적으로 활용한다면 전체 시스템의 성능 향상에 크게 기여할 수 있다. 비휘발성 램을 활용함으로써 시스템 소프트웨어의 성능을 향상시키고자 하는 노력 중의 하나로, 본 연구에서는 비휘발성 램을 고려하는 MiNV (Metadata in NVram) 파일시스템을 설계하고 구현하였다. MiNV 파일시스템은 모든 메타데이타를 비휘발성 램에 저장, 관리하고 일반 파일데이타는 낸드플래시 메모리에 저장하고 관리한다. 본 논문에서는 MiNV 파일시스템이 기존의 플래시 메모리 기반 파일시스템과 비교해서 얼마나 높은 성능 향상을 얻을 수 있는지를 정량적으로 보여준다. 성능 평가 결과, 비휘발성 램을 활용하는 파일시스템은 극도로 짧은 마운트 시간만을 필요로 한다. 기존의 대표적인 플래시 메모리 파일시스템인 YAFFS와 비교했을 때, MiNV 파일시스템은 동일한 워크로드를 처리하면서 보다 적은 횟수의 플래시 메모리에 대한 페이지 읽기, 쓰기, 그리고 블록 소거 연산을 요청한다. 플래시 메모리 연산 횟수에서의 이득은 MiNV 파일시스템의 수행속도 향상에 그대로 반영되어, 수행속도 측면에서 MiNV 파일시스템은 YAFFS보다 평균 400% 정도의 성능 향상을 보인다.

키워드 : 비휘발성 램, 플래시 메모리, 파일시스템, 메타데이타

Abstract *Non-volatile RAM (NVRAM) is a form of next-generation memory that has both characteristics of nonvolatility and byte addressability each of which can be found in nonvolatile storage and RAM, respectively. The advent of NVRAM may possibly bring about drastic changes to the system software landscape. When NVRAM is efficiently exploited in the system software layer, we expect that the system performance can be significantly improved. In this regards, we attempt to develop a new Flash file system, named MiNVFS (Metadata in NVram File System). MiNVFS maintains all the metadata in NVRAM, while storing all file data in Flash memory. In this paper, we present quantitative experimental results that show how much performance gains can be possible by exploiting NVRAM. Compared to YAFFS, a typical Flash file system, we show that MiNVFS requires only minimal time for mounting. MiNVFS outperforms YAFFS by an average of around 400% in terms of the total execution time for the realistic workloads that we considered.*

Key words : Non-Volatile RAM, Flash memory, File system, Metadata

· 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발] 사업의 일환과 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업으로 수행된 연구임(No. R0A-2007-000-20071-0)

† 학생회원 : 홍익대학교 컴퓨터공학과
ihdoh@cs.hongik.ac.kr
†† 종신회원 : 단국대학교 컴퓨터과학과 교수
choijm@dankook.ac.kr
††† 정 회 원 : 서울시립대학교 컴퓨터과학부 교수
dhl_express@uos.ac.kr
†††† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
samhnoh@hongik.ac.kr

논문접수 : 2007년 10월 2일
심사완료 : 2007년 11월 22일

정보과학회논문지: 시스템 및 이론 제35권 제2호(2008.2)

1. 서 론

비휘발성 램은 램의 특성과 비휘발성 저장매체의 특성을 동시에 가지고 있는 차세대 메모리이다. 현재, 다양한 형태의 비휘발성 램이 존재하며, 대표적인 비휘발성 램으로는 FeRAM(Ferroelectric RAM), PRAM (Phase-change RAM), 그리고 MRAM(Magnetoresistive RAM)이 있다[1-3]. 특히, 삼성전자는 2004년 64Mbit PRAM 개발에 성공한 뒤, 2005년 말 상용화가 가능한 256Mb PRAM을 세계 최초로 개발한 데 이어 2006년에는 512Mb PRAM을 개발하였고, 2007년 말에는 기가 비트 급의 PRAM을 개발해서 상용화 한다는 로드맵을 가지고 있다. 또한, 삼성은 ROM을 대체하는 용도로, 모바일 전화기에 시범적으로 PRAM 샘플을 탑재하여 그 사업성을 확인하고 있다[2]. 이처럼 비휘발성 램은 머지않아 임베디드 시스템을 그 시작으로 해서 컴퓨터 시스템의 일반적인 구성요소로써 그 위치를 확고히 할 것으로 예상된다.

시스템 소프트웨어가 비휘발성 램을 효과적으로 활용한다면 시스템 전체의 성능 향상을 꾀할 수 있을 것이다. 이에, 비휘발성 램이 현재의 시스템에 도입될 경우 시스템 소프트웨어는 어떠한 모습으로 진화되어야 할 것인가라는 근본적인 물음이 존재한다. 이 물음에 대한 한 가지 해결책으로써, 본 연구에서는 비휘발성 램을 활용하는 플래시 메모리 기반의 파일시스템을 제안한다. 파일시스템의 관점에서 보면 자주 참조되고 상대적으로 중요한 데이타인 메타데이타를 빠른 속도로 그리고 안정적으로 관리하는 것이 무엇보다도 중요하다. 따라서 본 연구에서는 메타데이타를 비휘발성 램에 관리하며, 파일데이타를 낸드플래시 메모리에 저장하고 관리하는 MiNV(Metadata in Non-Volatile RAM) 파일시스템(이후, MiNVFS)을 설계하고 구현한다.

본 연구의 목적은 플래시 메모리 기반의 파일시스템에서 메타데이타를 비휘발성 램에서 관리할 경우 얼마나 높은 성능 향상을 얻을 수 있는지를 정량적으로 보여주는 것이다. 성능 평가 결과 MiNVFS는 기존의 플래시 메모리 파일시스템들과 달리 극도의 짧은 마운트 시간만을 필요로 한다. 현실적인 파일시스템 워크로드를 고려하였을 때, MiNVFS가 플래시 메모리에 요청하는 페이지 읽기, 쓰기, 그리고 블록 소거 연산의 횟수는 기존의 대표적인 플래시 메모리 기반 파일시스템인 YAFFS와 비교해서 현저하게 줄어든다. 이는 수행속도 측면에서 최대 600%, 평균 437%의 성능 향상을 이끌어낸다.

본 논문의 구성은 다음과 같다. 2절에서 먼저 관련 연구들에 대해서 살펴보고, 3절에서는 MiNVFS의 설계에 대해서 설명한다. 이어서, 4절에서는 리눅스에서의 MiNVFS 구현과 성능 평가에 대해서 논의한다. 마지막으로, 5절에서는 결론을 맺는다.

2. 관련연구

본 연구와 관련이 있는 연구들은 플래시 메모리를 고려한 파일시스템, 그리고 비휘발성 램을 고려한 파일시스템에 관한 연구들로 구분될 수 있다.

최근 들어 임베디드 시스템에 대한 관심이 높아지면서 다양한 플래시 메모리 기반의 파일시스템들이 등장하고 있다. 플래시 메모리상에서 동작하는 파일시스템을 제공하기 위한 아주 초창기의 노력중의 하나는 Kawaguchi에 의해서 작성된 FTL(Flash Translation Layer) 형태의 플래시 메모리 드라이버이다[4]. 이후 FTL의 성능을 최적화하려는 노력이 있었으며, 그 중의 하나로 블록단위와 페이지단위의 FTL 사상테이블을 혼용해서 사용함으로써 FTL의 성능을 향상시킨 연구가 있었다[5]. 이와는 달리 플래시 메모리에 특화된 파일시스템을 제작하려는 노력이 있었다. 이 파일시스템들은 로그구조 기반 파일시스템을 근간으로 설계, 구현되었다[6]. 대표적인 플래시 메모리 파일시스템으로는 JFFS, JFFS2, 그리고 YAFFS 등이 존재한다[7,8]. 이들 외에도 다수의 플래시 메모리 파일시스템들이 존재하는데 Gal과 Toledo는 이들에 대한 조사를 폭 넓게 진행한 바 있다[9].

기존의 플래시 메모리 파일시스템, 특히 YAFFS, JFFS2 등의 가장 큰 취약점 중의 하나로 긴 마운트 시간을 들 수 있는데, 이러한 문제를 극복하기 위한 연구들도 진행된 바 있다[10,11]. 이들 연구에서는 파일시스템을 언마운트하기 전에 램에 구성되어 있는 파일시스템 메타데이타를 모두 플래시 메모리의 특정 공간에 복사해 둠으로써 빠른 마운트를 제공하고자 한다. 개선된 마운트 속도에도 불구하고, 이러한 기법을 지원하기 위해서는 추가적인 플래시 메모리 공간을 필요로 하며, 파일시스템이 언마운트되기 전에 추가적인 작업을 반드시 필요로 한다. 이는 시스템 성능에 적지 않은 성능 저하를 가져올 수 있다. 지금까지 언급한 모든 파일시스템들과 마운트 속도 향상을 위한 기법들은 비휘발성 램을 전혀 고려하지 않는다는 점에서 본 연구와 차별된다.

본 연구와 밀접하게 관련이 있는 연구로써, 비휘발성 램을 고려한 파일시스템에 대한 연구들이 존재한다. 현재까지 비휘발성 램을 고려한 파일시스템은 MRAM 파일시스템, Conquest 파일시스템, PRAM 파일시스템, 그리고 NEB 파일시스템이 있다[12-15]. MRAM 파일시스템과 Conquest 파일시스템은 본 연구와 마찬가지로 파일시스템의 메타데이타를 비휘발성 램인 MRAM에서 관리하고 데이타를 저장매체에 유지한다는 개념을 사용한다. 또한 같은 연구 그룹에서 메타데이타를 MRAM에

저장할 경우 데이타 보호를 어떻게 처리할 것인가에 대한 연구도 발표되었다[16]. 이들의 연구들은 메타데이타를 비휘발성 램에서 관리할 수 있다는 개념을 담고 있는 최초의 논문으로써 그 가치가 인정된다. 그럼에도 불구하고, 이들은 모두 저장매체로써 하드디스크를 고려하고 있으므로 이들 연구가 플래시 메모리의 특성을 전혀 고려하고 있지 않다는 점에서 본 연구와 차별된다. PRAM 파일시스템과 NEB 파일시스템은 각각 비휘발성 램인 PRAM과 FeRAM를 하드 디스크를 대체하는 주요 저장매체로써 고려한 파일시스템이라는 점에서 본 연구와 차별된다. 이처럼 파일시스템이 비휘발성 램을 고려하는 연구들이 이미 존재하지만 본 연구에서 제안된 MiNVFS와는 모두 차별된다.

3. MiNV 파일시스템의 설계

MiNVFS는 메타데이타를 비휘발성 램에 저장하고 관리하며 일반 데이타를 플래시 메모리에 유지하고 관리하는 파일시스템이다. 본 절에서는 MiNVFS의 설계에 대해서 설명한다. 먼저 MiNVFS가 비휘발성 램에 어떠한 종류의 메타데이타를 저장하는지에 대해서 언급하고, 이어서 이들 메타데이타가 어떻게 관리되는지에 대해서 설명한다.

3.1 비휘발성 램에 저장되는 메타데이타

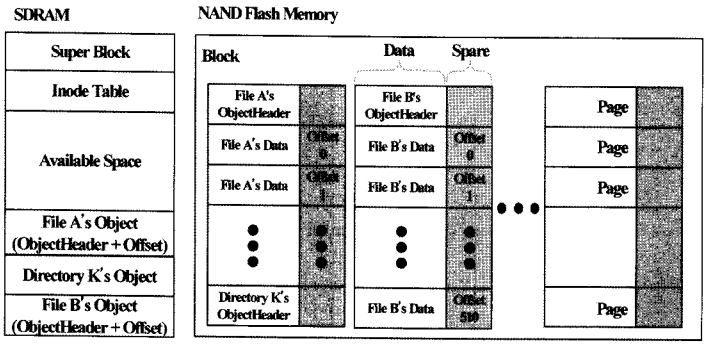
전통적인 파일시스템들 가운데 일부는 메타데이타에 대한 쓰기 연산을 동기적으로 처리함으로써 최소한의 파일시스템 안정성을 보장하고자 노력을 기울인다. 이들 파일시스템은 하나의 파일에 대한 수정을 처리하기 위해서 실제 데이타를 수정하는 작업은 버퍼 캐시(즉, 휘발성 메모리)에서 수행함과 동시에 해당 파일의 메타데이타에 대한 업데이트는 비휘발성 저장매체에 대해서 수행한다. 상대적으로 느린 저장매체에 대해 동기적으로 처리되는 메타데이타의 수정은 파일시스템의 수행속도를 저하시키는 주요한 원인 중의 하나이다.

메타데이타는 파일데이타와 비교했을 때 작은 크기를 가지지만 훨씬 더 빈번하게 업데이트되는 자료이다. 이러한 특성을 가지는 메타데이타를 소량의 빠른 저장장치에 저장하고 관리하였을 때, 직관적으로 파일시스템의 안정성과 수행속도에 대한 향상을 기대할 수 있다. 이러한 점을 고려해서, MiNVFS는 파일시스템을 구성하기 위한 모든 메타데이타 정보를 비휘발성 램에서 유지하고 관리하며, 파일데이타만을 플래시 메모리에 저장하도록 설계된다.

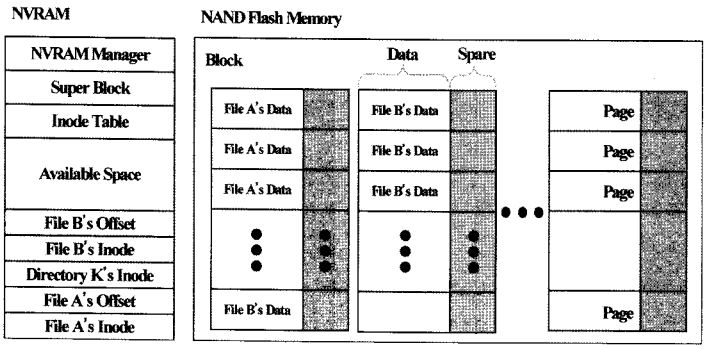
그림 1은 대표적인 플래시 메모리 기반 파일시스템 중의 하나인 YAFFS와 본 연구에서 제안하는 MiNVFS의 메타 데이타와 파일데이타가 여러 형태의 저장매체에서 관리되고 있는 상태를 보여준다. 그림 1(a)는 YAFFS가 메타데이타와 파일데이타를 플래시 메모리에 저장하고 있고, 파일시스템 마운트 이후에 메타데이타 정보를 모두 램으로 올려놓은 상태를 보여준다. YAFFS는 하나의 파일과 관련된 Inode 정보를 나타내는 구조체인 `yaffs_ObjectHeader`를 위해서 플래시 메모리의 1개 페이지를 할애한다. 실제 파일데이타는 페이지 단위로 저장되며, 각각의 페이지가 특정 파일의 어느 위치에 있는 데이타인지를 지시하는 정보인 파일 오프셋 정보는 해당 페이지의 스페어 영역에 기록된다. 그림 1(b)는 MiNVFS의 메타데이타가 저장되어 있는 비휘발성 램과 실제 데이타가 저장되어 있는 플래시 메모리의 레이아웃을 보여준다. 그림 1(a)와 가장 크게 차별되는 점은 MiNVFS는 메타데이타를 비휘발성 램에만 저장하고, 메타데이타를 캐시하기 위한 용도로써 램을 사용하지 않는다는 것이다.

MiNVFS의 비휘발성 램에는 그림 1(b)에서 보는 것처럼 슈퍼블록 정보와 Inode 테이블 정보, 그리고 하나 이상의 Inode 정보가 저장되어 있다. 비휘발성 램의 처음 부분에는 이들 메타데이타 정보를 관리하기 위해서 필요한 비휘발성 램 관리자 정보가 존재한다. 비휘발성 램 관리자 정보에는 슈퍼블록 정보와 Inode 테이블 정보의 시작 위치를 포함한다. 비휘발성 램은 BGET 메모리 할당자를 통해서 동적으로 관리되며 비휘발성 램 관리자 정보는 비휘발성 램을 동적으로 관리하는 데 필요한 정보를 유지한다[17]. 이때, BGET 메모리 할당자를 사용한 이유는 구현상의 편의성 때문이다.

비휘발성 램에 존재하는 메타데이타가 세부적으로 어떤 내용을 담고 있는지 설명하면 다음과 같다. 첫 번째 메타 데이타로써, 슈퍼블록은 파일시스템의 파일데이타가 저장되어 있는 플래시 메모리의 현재 상태 정보를 포함한다. 플래시 메모리의 각 페이지에 대해서 해당 페이지가 사용되고 있는지 여부와 유용한 데이타를 포함하고 있는지 아닌지를 나타내는 2비트 정보를 가진다. 또한 각 블록에 대해서 현재 할당되어 있는지 아니면 사용가능한지, 그리고 해당 블록에서 사용되고 있는 페이지의 개수 등을 나타내기 위해서 한 블록당 8바이트 정보를 유지한다. 두 번째 메타데이타로써, Inode 테이블이 있는데 이는 Inode 정보들을 해시하는 테이블로써 이 테이블에 현재까지 생성된 파일과 디렉터리와 관련된 Inode 정보가 연결되어 있다. 비휘발성 램에 존재하는 이 테이블을 통해서 이미 생성되어 있는 특정 Inode에 대한 접근은 시스템의 재부팅 여부와 관계없이 언제든지 가능하게 된다. 마지막으로, Inode 정보는 각 파일과 디렉터리를 파일시스템이 적절하게 관리할 수 있도록 하는 정보들을 담고 있으며, `Inode` 구조체와 `File_Offset` 구조체로 구성된다. `Inode` 구조체는 파일 혹은



(a) YAFFS



(b) MiNVFS

그림 1 YAFFS와 MiNVFS에 의해서 관리되는 메타데이타

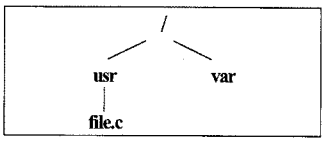
디렉터리에 대한 소유권, 접근권한, 생성, 수정시간 등을 포함하고 있으며, `File_Offset` 구조체는 파일의 특정 오프셋에 해당하는 데이타가 실제 플래시 메모리의 어느 위치에 있는지에 대한 정보를 담고 있다.

3.2 메타데이타의 관리

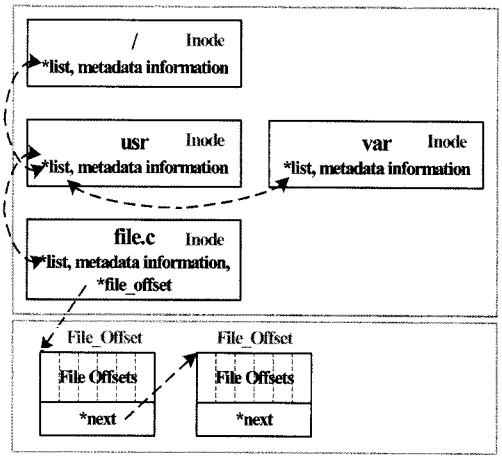
이전 절에서는 MiNVFS가 유지하고 있는 메타데이타에 대해서 살펴보았다. 본 절에서는 논리적인 디렉터리 계층 구조에 대해서 MiNVFS가 메타데이타를 어떠한 방식으로 관리하는지 설명한다.

그림 2(a)는 현재 구성되어 있는 디렉터리 계층과 파일의 구성을 논리적으로 보여주고 있으며, 그림 2(b)는 그 논리적인 모습을 표현하기 위해서 MiNVFS가 관리하고 있는 메타데이타의 구조를 보여준다. 그림 2(a)에는 루트 디렉터리와 하위 2개의 디렉터리, 그리고 하나의 파일이 존재한다. 그림 2(b)에서는 그림 2(a)의 3개 디렉터리들을 표현하기 위해서 각각 하나씩의 `Inode` 구조체가 생성되어 있고, 하나의 파일을 위해서 1개의 `Inode` 구조체가 할당되어 있다. 그리고 이들 `Inode` 구조체들은 디렉터리 계층구조를 표현하기 위해서 서로 리스트로 연결되어 있다.

MiNVFS에서의 디렉터리 계층 구조는 오직 비휘발성



(a) 디렉터리와 파일의 논리적인 구조



(b) MiNVFS에 의해서 관리되는 해당 메타데이타

그림 2 MiNVFS에 의한 메타데이타 관리

램에만 존재하는 Inode 구조체와 이들을 서로 연결하고 있는 연결 리스트만으로 충분히 표현될 수 있다. 반면에 파일은 추가적으로 해당 파일의 실제 데이타가 플래시 메모리상의 어느 곳에 위치하고 있는지에 관한 정보, 즉 파일 오프셋을 관리하기 위한 정보를 가지고 있어야 한다. 이를 위해서 파일을 위한 Inode 구조체는 파일 오프셋을 관리하기 위한 추가적인 구조체인 File_Offset 구조체에 대한 연결을 포함하고 있다.

File_Offset 구조체는 Inode 구조체와 같은 크기를 가지도록 설계되어 있으며, 이 구조체는 고정된 크기이므로 하나의 File_Offset 구조체가 표현할 수 있는 크기 이상으로 파일이 커진다면 추가적으로 File_Offset 구조체를 하나 더 할당해서 이들을 서로 연결하도록 설계되어 있다. 이때 File_Offset 구조체를 Inode 구조체와 같은 크기로 설계한 이유는 비휘발성 램의 할당과 해제 과정에서 외부 단편화의 발생 가능성을 없애기 위해서이다. File_Offset 구조체의 사용은 MiNVFS의 설계와 구현을 간단하게 하려는 의도이다.

File_Offset 구조체에서 사용되는 파일 오프셋의 관리 단위로는 어떠한 크기로도 설정될 수 있다. 이미 언급한 바와 같이, 본 연구에서는 전통적인 플래시 메모리 기반의 파일시스템에서 메타데이타를 비휘발성 램에서 관리할 경우에 얻을 수 있는 성능 향상을 정량적으로 보여주고자 한다. 이러한 이유로, 전통적인 플래시 메모리 파일시스템인 YAFFS의 파일 오프셋 관리 단위인 페이지를 MiNVFS의 파일 오프셋 관리 단위로 사용하였다.

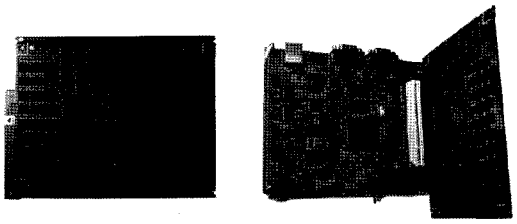
4. 성능 평가

본 연구에서는 MiNVFS를 리눅스에서 구현하였으며, 실제 임베디드 시스템 환경에서 MiNVFS의 성능을 평가하였다. 본 절에서는 리눅스에서의 MiNVFS 구현과 실험환경에 대해서 언급한 후 MiNVFS의 성능 평가 결과에 대해서 설명한다.

4.1 리눅스에서의 구현 및 실험 환경

본 연구에서는 메타데이타를 비휘발성 램에서만 관리하고 실제 데이타를 플래시 메모리에 저장하고 관리하는 MiNVFS를 설계하고 리눅스에서 구현하였다. MiNVFS는 리눅스 버전 2.4.x에서 모듈 형태로 동작하도록 구현되었다. MiNVFS는 앞서 설명했던 설계를 바탕으로 리눅스의 VFS(Virtual File System) 계층과 MTD(Memory Technology Device) 계층 사이에 구현되었다. MiNVFS는 파일시스템을 생성하는 도구를 포함하고 있고, 일반적인 파일시스템들이 제공해 주는 대부분의 기본적인 연산들을 처리할 수 있다.

MiNVFS는 비휘발성 램을 사용하는 플래시 메모리 기반의 파일시스템이다. 그림 3은 MiNVFS의 성능 평가 시 사용된 비휘발성 램과 임베디드 개발 보드의 모습을 보여주는 사진이다. 그림 3(a)는 MiNVFS를 구동하기 위해서 사용되는 비휘발성 램에 대한 사진이다. 이 비휘발성 램은 12MB 용량의 FeRAM이다. 수작업으로 제작된 비휘발성 램 보드는 그림 3(b)처럼 PCI 인터페이스를 통해서 임베디드 개발 보드와 연결된다. 이 비휘발성 램은 개발 보드 상에서 물리 메모리 주소를 통해 CPU로부터 직접 접근이 가능하다.



(a) 12MB FeRAM 보드 (b) EZ-M28 임베디드 보드

그림 3 실험을 위한 시스템 환경

성능 평가 시 사용된 임베디드 보드는 PCI 인터페이스를 지원하는 FALINUX사의 EZ-M28 보드이다[18]. 이 보드에는 운영체제로써 리눅스 버전 2.4.18이 동작한다. EZ-M28보드는 삼성에서 개발한 ARM920T 기반의 S3C2800 프로세서를 탑재하고 있으며, 32MB의 SDRAM과 64MB의 낸드플래시 메모리를 가지고 있다. 성능 평가 과정에서 사용된 낸드플래시 메모리의 용량은 64MB 중에서 32MB이며, 실제 사용된 FeRAM의 용량은 12MB 중에서 최대 501KB이다.

MiNVFS의 성능 비교 대상으로는 현재 가장 잘 알려져 있는 전통적인 플래시 메모리 기반 파일시스템인 YAFFS를 선택하였으며, 본 절의 성능 평가에서는 MiNVFS와 YAFFS의 성능을 각각 측정해서 서로 비교한다. 이때, MiNVFS는 YAFFS가 사용하지 않는 추가적인 12MB의 비휘발성 램을 사용하도록 실험 환경이 설정되어 있다. 이는 불공정한 실험 환경으로 보여질 수 있지만, YAFFS는 비휘발성 램을 사용하도록 설계되어 있지 않을 뿐만 아니라 MiNVFS가 YAFFS 보다 더 적은 양의 플래시 메모리를 사용하고 극히 소량의 SDRAM만을 필요로 한다는 점을 감안할 필요가 있다.

4.2 실험 결과

본 실험에서는 파일시스템의 성능 평가 인자로 마운트 시간과 파일시스템 워크로드를 처리하는 데 소요된 총 수행 시간 및 플래시 메모리 연산 횟수를 고려한다.

4.2.1 마운트 시간

기존의 플래시 메모리 파일시스템들은 플래시 메모리 내에 메타데이타 정보가 산재되어 있으므로 마운트 과

정에서 유효한 데이타가 저장되어 있는 모든 블록들을 스캔 해야만 한다. 이 스캔 과정은 실제로 마운트 시에 상당한 시간적인 오버헤드를 유발한다. 반면에 본 연구에서 제안한 MiNVFS는 비휘발성 램에 메타데이타를 저장하며, 항상 파일시스템이 마운트 된 상태와 일치하는 모습으로 메타데이타를 유지한다. 따라서 MiNVFS는 마운트를 위해서 플래시 메모리를 스캔하는 과정을 수행하지 않는다.

그림 4는 32MB 플래시 메모리의 사용률을 0%에서부터 100%까지 10%씩 증가시켜가면서 MiNVFS와 YAFFS의 마운트 시간을 측정한 결과이다. 이때, 사용률은 플래시 메모리 내의 전체 블록 개수에 대해서 유효한 데이타를 포함하고 있는 블록 개수의 비율을 의미한다. 그림 4에서, YAFFS는 플래시 메모리의 사용률이 증가함에 따라 파일시스템의 마운트 시간이 선형적으로 증가해서 최대 3초 이상의 시간을 소비한다. 이는 앞서 언급한 바와 같이 플래시 메모리의 사용률이 증가함에 따라서 스캐닝 해야 하는 블록의 수가 증가하기 때문이다. YAFFS와는 달리 MiNVFS가 마운트를 위해서 소비하는 시간은 플래시 메모리의 사용률에 관계없이 대략 67 마이크로 초로 거의 일정하다.

기존의 플래시 메모리 기반 파일시스템들의 마운트 시간은 플래시 메모리의 크기가 증가할수록 그 크기에 비례해서 길어지게 된다. 현재 iPod nano MP3 플레이어를 비롯하여 수 기가 바이트 단위의 플래시 메모리를 탑재한 휴대용 장치들이 보편화되고 있는 상황을 고려해 보았을 때, YAFFS와 같은 플래시 메모리 기반 파일시스템의 긴 마운트 시간은 더 이상 간과할 수 없는 문제임이 분명하다. 더불어, 이러한 플래시 메모리 기반 파일시스템이 루트 파일시스템으로 사용될 경우, 파일시스템의 오랜 마운트 시간은 부팅 속도 저하의 문제를 초래하게 된다. 이러한 측면에서 마운트 시간이 극도로 짧은 MiNVFS는 아주 매력적인 대안으로 보여질 수 있다.

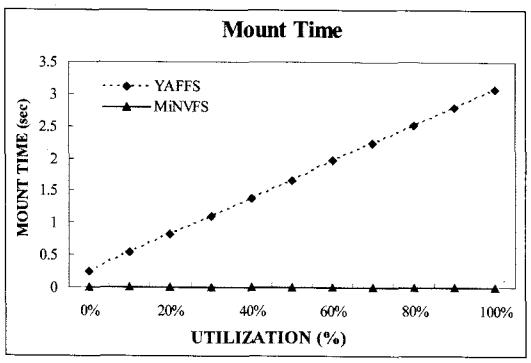


그림 4 YAFFS와 MiNVFS의 마운트 시간

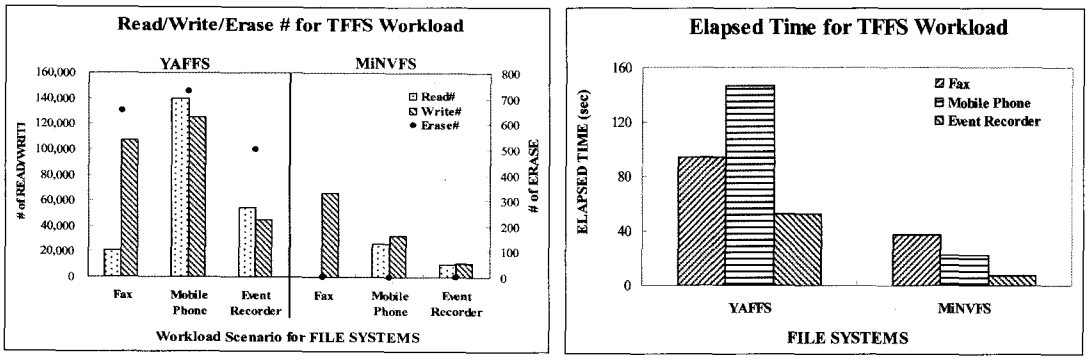
4.2.2 현실적인 시나리오에 대한 수행 성능 평가

현재까지 임베디드 시스템의 특성을 제대로 반영하는 워크로드를 얻고자 하는 노력은 많았으나 아직까지 이와 관련해서 공인된 형태의 워크로드나 벤치마크가 존재하지 않는 것이 현실이다. 본 실험에서는 플래시 메모리 기반 TFFS 파일시스템을 제안한 Gal과 Toledo가 TFFS의 성능을 평가하기 위해서 사용하였던 워크로드 시나리오를 그대로 재현해서 MiNVFS와 YAFFS의 총 수행 시간을 평가한다[19].

TFFS의 워크로드는 팩스와 휴대전화 그리고 이벤트 기록기의 동작과 파일의 생성, 수정에 관한 시나리오를 포함하고 있으므로 조금은 현실성이 있는 플래시 메모리 기반 파일시스템 워크로드로 보여진다. 팩스 워크로드는 팩스뿐만 아니라 자동응답기, 음악 재생기와 같은 비교적 큰 파일들이 저장되는 장치들의 동작을 대변한다. 휴대전화 워크로드는 휴대전화와 더불어 작은 파일들을 저장하는 호출기나 문자 전송 기기와 같은 장치들의 동작을 대변한다. 이벤트 기록기 워크로드는 로그 파일을 생성하고 그 로그 파일에 대한 수정을 필요로 하는 기기들의 동작을 대변할 수 있다.

그림 5는 MiNVFS와 YAFFS가 앞서 언급한 팩스, 휴대전화, 그리고 이벤트 기록기 워크로드 시나리오를 처리하면서 요청한 플래시 메모리 읽기/쓰기/소거 연산의 횟수와 각각의 워크로드를 처리하는데 걸린 총 수행 시간을 보여준다. 그림 5(a)에서, MiNVFS가 모든 워크로드 시나리오에 대해서 YAFFS보다 더 적은 횟수의 플래시 메모리 연산을 요청함을 알 수 있다. MiNVFS는 메타데이타를 비휘발성 램에 유지하기 때문에 메타데이타를 플래시 메모리에 저장하기 위해서 요청되는 쓰기 연산을 필요로 하지 않는다. MiNVFS와 달리 YAFFS는 메타데이타의 갱신 과정에서, 수정될 메타데이타에 대한 읽기 연산과 해당 메타데이타에 대한 무효화 연산 그리고 갱신된 메타데이타에 대한 쓰기 연산을 요청한다. 이러한 이유로 YAFFS는 메타데이타 관리를 위해서 MiNVFS보다 더 많은 플래시 메모리 페이지들을 사용하게 되며, 이는 자연스레 더 많은 블록 소거 연산을 동반한다.

그림 5(b)는 수행속도 측면에서 MiNVFS가 YAFFS보다 모든 워크로드에 대해서 좋은 성능을 나타냄을 보여준다. MiNVFS는 TFFS의 팩스 워크로드에 대해서 152%, 휴대전화 워크로드에 대해서 559%, 그리고 이벤트 기록기 워크로드에 대해서는 600%의 성능 향상을 나타낸다. MiNVFS는 TFFS 워크로드에 대해서 YAFFS에 비해 최대 600%, 평균 437%의 수행속도 향상을 보인다. 이 결과를 통해서, MiNVFS는 자주 업데이트되는 비교적 작은 크기의 파일들이 많이 존재하는 워크로드



(a) 읽기/쓰기/소거 연산 횟수 (b) 총 수행 시간

그림 5 TFFS 워크로드를 수행하면서 드러난 YAFFS와 MiNVFS의 수행 성능 비교

에 대해서 높은 성능 향상을 나타냄을 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 메타데이타를 비휘발성 램에 저장, 관리하고 일반 파일데이타는 낸드플래시 메모리에 저장하고 관리하는 MiNV(Metadata in Non-Volatile RAM) 파일시스템을 설계하고 구현하였다. 논문에서는 MiNVFS가 기존의 플래시 메모리 기반 파일시스템과 비교해서 얼마나 높은 성능 향상을 나타내는지를 정량적으로 보여주었다. 성능 평가 결과, MiNVFS는 극도로 짧은 마운트 시간만을 필요로 하며, 현실적인 파일시스템 워크로드에 대해서 MiNVFS가 YAFFS보다 최대 600%, 평균 437%정도 더 빠른 수행속도를 나타냈다.

본 연구와 관련해서 향후 해결해나가야 할 과제들을 정리하면 다음과 같다. 첫째, MiNVFS는 메타데이타를 비휘발성 램에 저장하고 관리하기 때문에 어렵지 않게 메타데이타를 완벽한 수준으로 안전하게 관리할 수 있다. 그렇지만 비휘발성 램에 존재하는 메타데이타와 플래시 메모리에 존재하는 파일데이타 사이에 일관성이 파괴되는 경우에 대한 대비책을 현재는 가지고 있지 않다. 완벽한 수준의 안정성을 보장하기 위해서는 이러한 문제에 대한 해결책이 반드시 요구된다. 둘째, 본 연구의 성능 평가에서 기존의 파일시스템에 비해서 현저하게 줄어든 MiNVFS의 플래시 메모리 입출력 연산 횟수를 고려한다면, MiNVFS는 직관적으로 플래시 메모리의 마모도 평준화(Wear Leveling)와 저전력 측면에서 기존 시스템과 비교해서 높은 성능을 가질 것으로 추측된다. 향후에 이러한 추측에 대한 확인 작업이 필요하며, MiNVFS를 위한 마모도 평준화 기법과 저전력 기법에 대한 연구가 필요하다. 마지막으로, MP3 플레이어나 휴대전화와 같은 임베디드 시스템 환경에서 MiNVFS가 더 적은 양의 비휘발성 램만을 사용하면서 더 높은 성능을 보일 수 있도록 최적화될 필요가 있다.

참 고 문 헌

[1] Ramtron International - Nonvolatile Memory, Integrated Memory and Microcontrollers, http://www.ramtron.com.

[2] Tech-On News, http://techon.nikkeibp.co.jp/english/NEWS_EN/20070226/128173.

[3] Freescale Semiconductor, http://www.freescale.com.

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

[5] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, A Space-efficient Flash Translation Layer for CompactFlash Systems, *IEEE Transactions on Consumer Electronics, 28*, 2 (May 2002), 366-375.

[6] M. Rosenblum and J. K. Ousterhout, The design and implementation of a log-structured file system, *ACM Transactions on Computer Systems, 10*, 1 (Feb. 1992), 26-52.

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

[8] D. Woodhouse, JFFS: The Journaling Flash File System, Ottawa Linux Symposium, 2001.

[9] E. Gal and S. Toledo, Algorithms and Data Structures for Flash Memories, *ACM Computing Surveys (CSUR), 37*, 2 (Jun. 2005), 138-163.

[10] K. S. Yim, J. Kim, and K. Koh, A Fast Start-Up Technique for Flash Memory Based Computing Systems, In *Proceedings of the 2005 ACM Symposium on Applied Computing(SAC 2005)*, pp. 843-849, Mar. 2005.

[11] C. H. Wu, T. W. Kuo and L. P. Chang, Efficient Initialization and Crash Recovery for Log-based File Systems over Flash Memory, In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006)*, pp. 896-900, Apr. 2006.

[12] A. B. Szczurowska, MRAM-preliminary analysis for file system design, Master's thesis, University

of California, Santa Cruz, Mar. 2002.
[13] A. A. Wang, P. Reiher, G. J. Popek, and G. H. Kuenning, Conquest: Better Performance Through a Disk/Persistent-RAM Hybrid File System, In *Proceedings of the USENIX Annual Technical Corference (USENIX 2002)*, pp. 15-28, Jun. 2002.
[14] CE Linux Public, PramFs, http://www.celinuxforum.org/CelfPubWiki/PramFs.
[15] C. Hyun, S. Baek, S. Ahn, J. Choi, and D. Lee, Experimental Evaluation of an Extent-based File System for Nonvolatile-RAM, In *Proceedings of the 2nd International Workshop on Software Support for Portable Storage (IWSSPS 2006)*, pp. 18-24, Oct. 2006.
[16] K. Greenan and E. Miller, "Reliability Mechanisms for File Systems Using Non-Volatile Memory as a Metadata Store," In *Proceedings of the 2nd International Workshop on Software Support for Portable Storage (IWSSPS 2006)*, pp. 8-17, 2006.
[17] The BGET Memory Allocator, http://www.fourmilab.ch/bget.
[18] FALINUX, EZ-M28, http://falinux.com/zproducts/ez-m28.php.
[19] E. Gal and S. Toledo, A Transactional Flash File System for Microcontrollers, In *Proceedings of the USENIX Annual Technical Conference (USENIX 2005)*, pp. 89-104, Apr. 2005.

도 인 환

2003년 홍익대학교 정보컴퓨터공학부(학사). 2006년 홍익대학교 컴퓨터공학과(석사). 2006년~현재 홍익대학교 컴퓨터공학과 박사과정. 관심분야는 운영체제, 내장형시스템, 차세대저장장치

최 종 무

1993년 서울대학교 해양학과(학사). 1995년 서울대학교 컴퓨터공학과(석사). 2001년 서울대학교 컴퓨터공학과(박사). 2001년~2003년 유비쿼스 주식회사 책임연구원. 2003년~현재 단국대학교 정보컴퓨터학부 조교수. 2005년~2006년 University of California, Santa Cruz 교환교수. 관심분야는 운영체제, 내장형시스템, 스토리지시스템, 차세대저장장치

이 동 희

1989년 서울대학교 컴퓨터공학과(학사). 1991년 서울대학교 컴퓨터공학과(석사). 1998년 서울대학교 컴퓨터공학과(박사). 1998년~1999년 삼성전자 중앙연구소 선임연구원. 1999년~2001년 제주대학교 통신컴퓨터과학부 조교수. 2002년~현재 서울시립대학교 컴퓨터과학부 부교수. 관심분야는 운영체제, 내장형시스템, 플래시메모리스토리지.

노 삼 혁

1986년 서울대학교 컴퓨터공학과 학사 1993년 메릴랜드대학교 컴퓨터과학과 박사. 1993년~1994년 조지워싱턴대학교 객원 조교수. 1994년~현재 홍익대학교 정보컴퓨터공학부 교수. 관심분야는 운영체제, 플래시메모리소프트웨어, 내장형시스템, 차세대저장장치