파일시스템 기능을 지원하는 FAT 호환 플래시 변환 계층※

김유미, 백승재, 최종무
단국대학교 컴퓨터학과
e-mail:{raspberi, ibanez1383, choijm}@dankook.ac.kr

File System Featured FAT Compatible Flash Translation Layer

Yumi Kim, Seungjae Baek, Jongmoo Choi
Dept. of Computer Science and Engineering, Dankook University

요 약

저 전력, 내구성, 소형, 빠른 속도 등의 장점을 가지고 있는 플래시 메모리는 생산 기술 발전에 힘입어 센서 노드, 휴대폰, MP3, PMP등의 소형 전자 제품의 저장장치에서부터 SSD형태로 노트북이나 서버에 이르기 까지 그 활용범위가 더욱 확장되어 가고 있다. 다양한 시스템에서 사용될 수 있는 플래시 메모리의 특성상 이에 저장된 데이터의 호환성은 중요한 고려사항이다. 이를 위해 플래시 메모리의 고유한 특성을 숨기고 일반적인 블록장치로 에뮬레이션 해주는 소프트웨어인 FTL과 FAT 파일시스템이 플래시 메모리 관리를 위한 사실상 표준 소프트웨어로써 사용되고 있다. 그러나 범용 컴퓨터를 기반으로 개발된 FTL과 FAT 파일시스템을 열악한 하드웨어로 구성된 시스템에서 구동하는 경우 많은 제약이 발생한다. 따라서 본 논문에서는 이러한 제약사항을 극복하기 위해 최소한의 파일시스템 기능을 제공하는 FAT 표준 호환 FTL을 제안한다. 제안된 기법은 리눅스 운영체제에 동적으로 적재 가능한 모듈형태로 구현되었으며, 실험을 통해 본 논문에서 제안한 기법이 기존 기법 대비 32%의 메모리 공간을 절약할 수 있으며, 동시에 완벽한 FAT 호환성을 제공함을 확인할 수 있었다.

1. 서론

저 전력, 충격에 대한 내구성, 소형, 빠른 속도 등을 특징으로 하는 플래시 메모리는 생산 기술 발전으로 인해 지속적인 대용량화 및 용량 대비 가격 하락이 가속화 되고 있다. 이에 따라 센서 노드(sensor node)나 휴대폰, MP3, PMP등의 소형 전자제품의 저장장치에서부터, 서버나 노트북 등에 사용되는 SSD(Solid-State Disk)에 이르기까지 그 활용 범위 또한 더욱 넓어지고 있다.

플래시 메모리는 기존 저장장치와 달리 덮어쓰기 제약, 삭제연산의 필요성, 연산 수행 단위와 시간의 비대칭성, 제한된 횟수의 삭제 연산 가능, 배드 블록(bad block)의 발생 가능 등 기존 저장장치와는 다른 특징을 가지고 있다.

이러한 플래시 메모리를 저장장치로써 활용하기 위해 플래시 메모리의 특성을 고려한 플래시 메모리 전용 파일시스템에 대한 연구나 [1, 2], 플래시 메모리 고유의 특성을 감추고 상위 수준에 하드디스크와 같은 일반적인 블록장치로 보여줌으로써 기존의 파일시스템을 사용할 수 있도록 해주는 플래시 변환 계층(FTL: Flash Translation Layer)에 대한 연구 등이 진행된바 있다 [3-6].

USB 플래시 드라이브나 SD(secure digital) 카드와 같이 다양한 시스템에서 사용될 수 있는 플래시 메모리 기반 저장 장치의 특성상 플래시 메모리에 기록된 데이터의 호환성은 매우 중요한 고려 사항이다. FAT 파일 시스템은 다양한 플랫폼에서 데이터의 호환이 가능하기 때문에 플래시 메모리를 저장장치로 사용하는 시스템에서 FTL과 함께 사실상 표준으로 사용되고 있다 [8].

그러나 데이터의 호환성을 위해 범용 컴퓨터 기반으로 개발된 FTL과 FAT 파일 시스템을 낮은 성능의 CPU와 최소한의 메모리로 구성되는 센서노드와 같은 열악한 하드웨어 환경에서 운용하기에는 많은 제약이 따른다.

본 논문에서는 이러한 제약조건을 극복하기 위해 최소한의 파일시스템 기능을 제공하는 FAT 표준 호환 플래시 변환 계층(File System featured FAT compatible Flash Translation Layer, 이하 FSFTL)을 제안한다. FSFTL은 열악한 하드웨어 자원을 가지는 시스템에서도 원활히 동작할 수 있도록 최소의 메모리 사용량을 필요로 하며, FAT 표준과 호환되는 최소한의 파일시스템 연산(operations)을 지원한다.

제안된 기법은 400MHz로 동작하는 Xscale CPU, 64MB SDRAM, 64MB NAND 플래시 메모리 등이 장착되어 있는 임베디드 보드 상에서 [9], 리눅스의 VFAT[10] 파일시스템과, NFTL[11]을 기반으로 구현되었다. 또한 실험을

※ 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2008-314-D00340)

통해 제안된 FSFTL이 기존의 VFAT과 NFTL 대비 32% 적은 용량의 메모리 자원을 필요로 하며, 동시에 효율적으로 FAT 표준 호환 파일시스템 연산을 제공함을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해 살펴보며, 3장에서는 FSFTL의 구조에 대해 설명한다. 4장에서는 실험결과를 보이며, 끝으로 5장에서는 결론 및 향후 연구계획에 대해 기술한다.

2. 관련연구

플래시 메모리 기반 저장장치를 효율적으로 관리하기 위한 기법은 크게 둘로 나뉜다. 첫째, 플래시 메모리의 특성을 고려한 전용 파일 시스템을 사용하는 방법이다. LFS(Log-Structured File System)[14]의 동작 구조를 이용한 JFFS[1], YAFFS[2]등의 파일 시스템이 대표적인 예이다.

둘째, 기존 하드디스크와 다른 플래시 메모리의 다양한 특성을 숨기고 일반적인 블록 장치로 에뮬레이션 해주는 FTL과 FTL 상위에 전통적인 파일시스템을 구축하여 사용하는 방법이다. 이때 논리적인 페이지 번호와 물리적인 페이지 번호를 연결하는 단위에 따라 블록 맵핑 기법 FTL [3]과 페이지 맵핑 기법 FTL [4]으로 나뉠 수 있다. 또한 이들의 장점을 취한 혼합형 매핑 FTL도 존재한다.

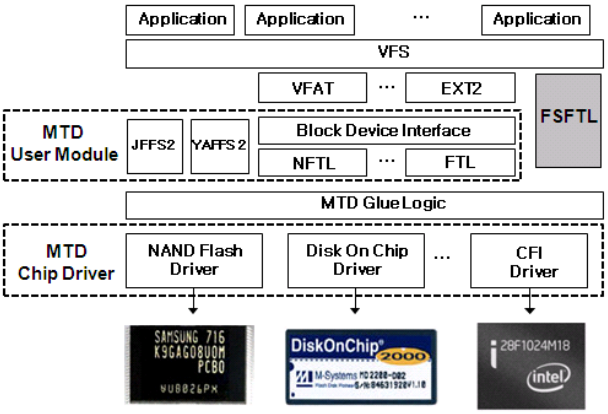
[5]의 연구에서는 기본적으로 모든 데이터 블록에 대해 블록 레벨 매핑 정보를 유지하며, 페이지 레벨로 매핑 정보가 유지되는 소수의 로그블록을 두고 각 블록 별로 갱신된 데이터들을 별도의 로그블록에 기록한다. 로그 블록이 가득 차는 경우 병합(switch)연산을 통해 새로운 로그블록을 생성한다.

[5]의 연구를 바탕으로, 로그블록을 순차 로그블록(sequence log block)과 임의 로그블록(random log block)으로 구분한 뒤, 논리 블록 당 하나씩의 로그블록이 지정되던 [5]와 달리 순차 데이터가 아닌 경우에는 완전 사상(fully-associative)되는 임의 로그블록에 데이터를 기록한 뒤 추후 병합하는 기법도 연구된 바 있다 [6]. 또한 [7]에서는 성능을 저해하는 완전병합(full merge)연산을 줄이기 위해 데이터의 지역성을 고려하였다. 구체적으로 임의 로그블록을 Hot/Cold 영역(partition)으로 구분하고, 지역성 검사기(locality detector)를 이용해 데이터를 다른 영역에 저장함으로써 성능향상을 추구하였다.

3. FSFTL: 파일 시스템 기능을 지원하는 FAT 호환 플래시 변환 계층

본 논문에서는 열악한 하드웨어 환경에서도 원활히 동작하며, FAT 호환 파일시스템 기능을 제공하는 플래시 변환 계층인 FSFTL을 제안한다. 본 논문에서 제안하는 FSFTL은 리눅스 커널 2.6.21상에 적재 가능한 모듈 형태로 실제 구현되었다.

(그림 1)에서 볼 수 있듯이 리눅스의 플래시 메모리 관리 소프트웨어 구조는 크게 VFS(Virtual File System)계층, 파일 시스템 계층(예: Ext2, VFAT 등), MTD(Memory Technology Device)계층의 3단계의 계층 구조를 이루고 있다. MTD계층은 MTD 사용자 모듈 계층과 MTD 글루 로직(glue logic) 계층, MTD칩 드라이버 계층으로 다시 나누어진다. MTD 사용자 모듈 계층은 블록 수준 매핑을 제공하는 NFTL, 페이지 수준 매핑을 하는 FTL 등의 오픈 소스 FTL 소프트웨어들을 포함한다. 본 논문에서 제안하는 FSFTL은 MTD 글루 로직 상위에, FTL과 파일시스템의 기능을 동시에 수행할 수 있도록 VFAT, FAT 그리고 NFTL을 기반으로 구현되었다.



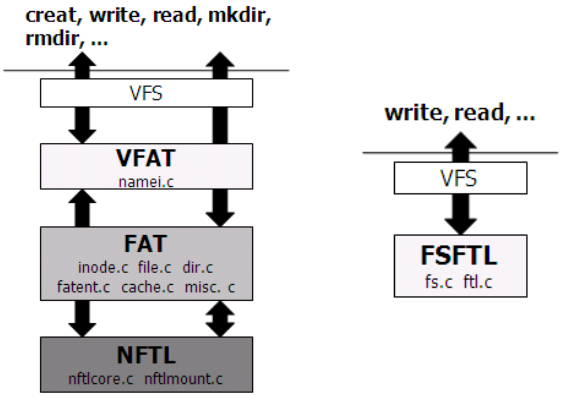
(그림 1) FSFTL과 Linux의 MTD 구조

(그림 2)는 FSFTL의 기본 개념을 보여준다. 리눅스 상에서 VFAT 파일 시스템을 통해 플래시 메모리를 저장장치로 사용하기 위해서는 총 세 개의 모듈(VFAT, FAT, NFTL)을 필요로 한다. NFTL은 상위 수준에 일반적인 블록 읽기/쓰기 인터페이스를 제공하며, VFAT은 FAT 파일 시스템이 제공하는 기본적인 FAT 파일 시스템 관련 인터페이스를 이용하여 VFAT 고유의 기능을 제공한다. 그러나 FSFTL은 단일 소프트웨어 모듈을 통해 FTL의 기능과 파일시스템의 기능을 동시에 제공한다.

한편 FSFTL은 상위 VFS 수준에 제공할 FAT 호환 파일시스템 관련 기능들이 설정(configuration) 가능하도록 구현되었다. 따라서 FSFTL이 운용될 환경에 따라 보다 많은 파일시스템 기능을 제공하도록 설정되거나, 혹은 가장 작은 메모리 용량을 요구하도록 설정될 수 있다.

4. 실험 결과

구현된 FSFTL은 400MHz로 동작하는 Intel PXA255 CPU와 64MB DRAM, 64MB NAND 플래시 메모리 등의 장치가 부착되어 있는 임베디드 개발 보드 상에서



(a) 기존 시스템 구조 (b) FSFTL 구조

(그림 2) 기존 시스템과 FSFTL의 구조 비교

실험되었다. 비교대상으로는 Linux의 VFAT[10] 파일시스템과 MTD(Memory Technology Device)내에 존재하는 NFTL[3]을 사용하였다.

리눅스 상에서 VFAT 파일 시스템을 통해 플래시 메모리를 저장장치로 사용하기 위해 필요한 세 모듈의 총 크기는 <표 1>에 나타낸 바와 같이 93.71Kbyte이다. 반면 FSFTL의 코드 크기는 기존 모듈의 크기 대비 32% 작아진 64.42Kbyte에 불과함을 알 수 있다. 이는 FSFTL이 열악한 하드웨어 환경을 가진 시스템에서도 효율적으로 수행될 수 있음을 의미한다.

<표 1> 모듈 파일 사이즈 비교

	기존 시스템		FSFTL
크기	VFAT	14.15 KByte	64.42 KByte
	FAT	62.52 KByte	
	NFTL	17.04 KByte	
합계		93.71 KByte	64.42 KByte

(그림 3)은 FSFTL의 실제 동작 과정을 통해 FAT 호환성 지원 여부를 보여준다. (그림 3)에서는 우선 nftl.ko, fat.ko, vfat.ko 세 개의 모듈을 'insmod' 명령을 이용하여 커널에 적재한다. 그런 뒤 'nftl_format' 과 'mkdosfs' 명령어를 사용하여 NFTL 수준의 포맷 및 FAT32 형식으로 파일시스템 수준의 포맷을 수행하였다. 그런 뒤 mount 명령을 사용하여 블록 장치를 마운트 하고, a.txt 파일을 생성하여 test_string 이라는 테스트 문자열을 파일에 기록하였다. 이후 같은 블록 장치를 fsftl.ko 모듈을 통해 마운트 하여 a.txt 파일에 대한 연산을 수행함으로써 FSFTL이 완벽한 FAT 호환 파일시스템 기능을 제공함을 확인할 수 있었다.

또한 Postmark[12] 벤치마크 프로그램을 이용하여 구현된 FSFTL의 오버헤드를 측정하였다. <표 2>는 FSFTL과 기존 소프트웨어의 벤치마크 수행 결과를 보여준다. <표 2>에서 볼 수 있듯이 본 논문에서 제안된 FSFTL은 기존 기법에 대비 부가적인 오버헤드 없이 FAT호환성을 지원함을 알 수 있다.

```
[root@falinux ~]$ ./nftl_format /dev/mtd2
                    :
[root@falinux ~]$ insmod fat.ko
[root@falinux ~]$ insmod vfat.ko
[root@falinux ~]$ insmod nftl.ko
                    :
[root@falinux ~]$ fdisk /dev/nftla
                    :
[root@falinux ~]$ ./mkdosfs -F 32 /dev/nftla1
mkdosfs 2.11 (12 Mar 2005)
[root@falinux ~]$ mount /dev/nftla1 /mnt/temp
[root@falinux ~]$ cd /mnt/temp
[root@falinux temp]$ touch a.txt
[root@falinux temp]$ echo "test string" >>a.txt
[root@falinux temp]$ ls
a.txt
[root@falinux temp]$ cat a.txt
test string
[root@falinux temp]$
[root@falinux temp]$ cd ..
[root@falinux mnt]$ umount /mnt/temp
                    :
[root@falinux ~]$ insmod inflaware.ko
                    :
[root@falinux ~]$ mount /dev/nftla1 /mnt/temp
[root@falinux ~]$ cd /mnt/temp/
[root@falinux temp]$
[root@falinux temp]$ ls
a.txt
[root@falinux temp]$ cat a.txt
test string
[root@falinux temp]$
```

(그림 3) FSFTL의 FAT 호환성 확인

<표 2> Postmark 수행 결과

	벤치마크 수행시간(초)
VFAT+FTL	5
FSFTL	5

5. 결론 및 향후 연구 계획

본 논문에서는 열악한 하드웨어 자원을 가지는 시스템에서도 플래시 메모리를 저장장치로써 원활히 사용하기 위한 FSFTL을 제안하였다. 제안된 기법은 리눅스 2.6.21에 실제 구현되었으며, 실험을 통해 FSFTL이 기존 기법 대비 적은 메모리 자원을 필요로 하며, 적은 오버헤드로 FAT 호환 파일시스템 기능을 지원함을 확인하였다.

향후 본 연구는 FSFTL이 제공하는 파일시스템 기능을 다양하게 설정함으로써 성능, 기능성, 자원요구량 간의 트레이드오프 분석 및 이를 통한 플래시 메모리 관리 소프트웨어의 스펙트럼 연구로 확장해 나갈 것이다.

참고문헌

[1] D. Woodhouse, "JFFS: The journaling Flash file system", Ottawa Linux Symposium, 2001, http://sources.redhat.com/jffs2/jffs2.pdf

[2] Aleph One, "YAFFS: Yet another flash file system", http://www.yaffs.net/

[3] M-Systems, "Flash-Memory translation layer for NAND flash (NFTL)", 1998.

[4] Intel Corporation, "Understanding the Flash Translation Layer (FTL) Specification," 1998.

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

[6] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park, and Ha-Joo Song, "A Log Buffer based Flash Translation Layer using Fully Associative Sector Translation", ACM Transactions on Embedded Computing Systems, vol. 6, Feb. 2007.

[7] Sungjin Lee, Dongkun Shin, Young-Jin Kim, Jihong Kim, "LAST: Locality-Aware Sector Translation for NAND Flash Memory-based Storage Systems", SIGOPS Oper. Syst. Rev. vol. 42, no. 6, pp. 36-42, 2008.

[8] 현철승, 최종무, 이동희, 노삼혁, "FAT 파일시스템의 호환성을 유지하며 성능과 안정성을 향상시키는 저널링 기법의 설계", 정보과학회 학술발표논문집, 제 35권 1호 (A), pp. 319~320, 2008.

[9] EZ-X5 Evaluation Board, "http://www.falinux.com"

[10] Linux VFAT File System, http://bmrc.berkeley.edu/people/chaffee/vfat.htmll

[11] MTD subsystem for Linux, http://www.linux-mtd.infradead.org/archive/index.htm

[12] J. Katcher, "PostMark: A New File System Benchmark", Technical Report TR3022, Network Appliance Inc., 1997.