플래시 메모리 관련 최근 기술 동향†

서울대학교 이수관 · 민상렬* · 조유근*

1. 서 론

플래시 메모리는 기계적인 동작이 없는 순수 반도체로 이루어진 비휘발성 저장매체로서 속도가 빠르고 전력소모가 작으며 또한 부피가 작고 충격에 강하다는 장점이 있다. 플래시 메모리는 초기에 주로 재기록 가능한 펌웨어 및 임베디드 시스템의 데이터 저장 매체로 사용되었지만 유비쿼터스 컴퓨팅을 지향하는 최근의 컴퓨터 산업 발전 추세에 따라 이동전화기, 개인정보단말기(PDA), MP3 재생기, 이동 멀티미디어 재생기(PMP) 등의 다양한 개인용 멀티미디어 정보기기로 적용범위를 넓혀가고 있으며, 최근에는 하이브리드 하드디스크, 플래시 메모리 SSD(Solid State Disk)에 대한 연구도 활발하게 진행되고 있다.

우리나라는 세계 최고 수준의 플래시 메모리 생산 기술을 보유하고 있다. 플래시 메모리 시장규모는 최근 몇 년간 급속도로 커져 왔으며 그 중에서도 가장 큰 성장을 기록한 NAND 플래시 메모리 시장에서 삼성전자와 하이닉스는 60% 이상의 시장 점유율을 보이고 있다. 하지만 우리나라의 플래시 메모리 하드웨어 국가경쟁력과 비교할 때 플래시 메모리 소프트웨어의 국제경쟁력은 상대적으로 낮기 때문에 플래시 메모리 소프트웨어 기술 발전의 중요성이 점차 커지고 있다.

본 논문에서는 플래시 메모리의 특성, 제약 조건 등에 대하여 설명하고 이를 극복하기 위한 소프트웨어 및 하드웨어 지원 기술에 대하여 살펴본 후 상용 솔루션 및 최근 연구결과에 대하여 알아본다.

† 본 논문은 정통부 선도기반기술개발사업("Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발") 참여 연구원들간의 토의에 기반하여 작성되었습니다. 토의에 적극적으로 참여하여 주신 여러 연구원분들께 감사드립니다.

† 본 논문은 정통부 및 정보통신연구진흥원의 정보통신선도기반기술개발사업의 연구결과로 수행되었습니다.

* 종신회원

2. 플래시 메모리의 특성

플래시 메모리는 내부 소자들의 구성 형태에 따라 NOR, DINOR, AND, NAND 등의 다양한 종류가 있다. 본 장에서는 이 중 현재 가장 많이 사용되고 있는 NOR 플래시 메모리와 NAND 플래시 메모리의 특성 및 응용에 대하여 살펴본다.

2.1 NOR 플래시 메모리

초기에 개발된 NOR 플래시 메모리는 전원공급이 끊겨도 데이터를 잃지 않는 비휘발성 메모리로서, 바이트 단위로 읽기 가능한 RAM 형태의 인터페이스를 가진다. 하지만 쓰기의 경우에는 해당 바이트의 1인 비트를 0으로 바꾸는 동작만 가능하다. 0인 비트를 쓰기가 가능한 1인 상태로 바꾸는 동작을 소거(erase)라고 하며 이는 읽기, 쓰기 보다 훨씬 큰 영역인 소거 블록(erase block)단위로만 가능하다. NOR 플래시 메모리는 읽기 성능은 비교적 좋지만 쓰기와 소거 성능은 매우 낮다.

상기한 동작 특성 외에 중요한 특성으로는 소거 횟수에 제한이 있다는 것이다. 일반적으로 100,000번 정도의 소거 싸이클을 지원하며 이 후에는 정확한 동작을 보장할 수 없게 된다. 이런 현상을 마모(wear)된다고 표현한다.

2.2 NAND 플래시 메모리

NOR 플래시 메모리보다 나중에 개발된 NAND 플래시 메모리는 집적도를 높일 수 있는 구조적인 특성으로 인해 보다 적은 비용으로 대용량의 저장장치를 구성할 수 있게 되었다. 하지만 NOR 플래시 메모리와 같은 바이트 단위의 접근은 가능하지 않으며 일정 영역(페이지) 단위로만 읽기 및 쓰기가 가능하다. 이런 특성으로 인해 일반적인 RAM 인터페이스를 이용할 수 없고 별도의 인터페이스 장치가 필요하며 읽기 성능 또한 NOR 플래시 메모리보다 낮다. 하지만 쓰기 및 소거 성능은 NOR 플래시 메모리보다 현저하게 높

다. 일반적인 페이지의 크기는 512B, 2KB 등이다. 또한 한번 소거된 후 페이지 당 쓰기 횟수에 제한이 있는데 이를 NOP(Number Of Program)라 하며 1~4 정도의 값을 가진다.

NOR 플래시 메모리와 마찬가지로 NAND 플래시 메모리도 소거 동작이 필요하고 소거 횟수에 제한이 있다(보통 100,000번). NOR 플래시 메모리와 달리 NAND 플래시 메모리는 수율(yield)을 높이기 위하여 공장 출하 시 소수의 초기 배드 블록을 허용한다. 이러한 초기 배드 블록은 스패어 영역의 특정 위치에 표시되며, 플래시 메모리의 첫 번째 블록, 즉 0번 블록은 배드 블록이 아님이 보장된다.

삼성전자는 최근 NAND 플래시 메모리와 제어 로직을 결합해 RAM 인터페이스를 사용할 수 있도록 한 OneNAND 플래시 메모리[1]를 개발, 생산하고 있으며 이는 대용량, 고성능이 필요한 모바일 디바이스 분야에서 점차 많이 사용되고 있다.

2.3 NOR/NAND 응용

일반적인 컴퓨팅 환경에서의 메모리 계층구조에서 코드 실행은 휘발성이지만 고속 동작이 가능한 DRAM 또는 SRAM 등을 이용하게 되며 대량의 데이터 저장에는 가격대비 용량이 큰 하드디스크가 주로 이용된다. 이와 달리 내장형 시스템에서는 환경적인 제한(크기, 충격, 온도, 무게 등), 전력 소모, 실시간성 등의 여러 가지 제한으로 인해 기계적인 동작이 필요한 하드디스크보다는 플래시 메모리가 많이 사용되어져 왔다. NOR 플래시 메모리는 바이트 단위로 읽기 동작이 가능하기 때문에 주로 코드영역을 저장하는데 사용되며 이 경우 코드 수행이 필요할 때 DRAM/SRAM을 통하지 않고 플래시 메모리 상에서 직접 프로그램 코드를 수행할 수 있다(XIP: eXecute In Place). 대용량의 데이터 저장이 필요한 경우 NOR 플래시 메모리보다 가격대 용량이 큰 NAND 플래시 메모리가 많이 사용된다.

표 1 NOR 및 NAND 플래시 메모리 비교

	NOR	NAND
페이지 크기	1B, 2B, 8B, 32B	512B, 2KB
소거 블록 크기	8KB~128KB(다양)	16KB, 128KB
NOP	제한 없음	1~4
소거 횟수	100,000	100,000
배드 블록	실행시간 배드 블록	초기 및 실행시간 배드 블록
집적도	낮다	높다
주요 응용	코드 저장	데이터 저장
주요 생산	Intel, AMD	삼성전자, 하이닉스, 도시바

표 1은 NOR 및 NAND 플래시 메모리의 특성 및 제약 사항에 관해 정리한 것이다.

3. 플래시 메모리 관리 기법

2장에서 설명된 쓰기 전 소거의 필요성, 소거 횟수 제한, 초기 및 실행시간 배드 블록 등의 제약 조건들로 인해 플래시 메모리는 하드디스크와 같은 블록 디바이스로 직접 사용할 수 없고 하드웨어 및 소프트웨어적인 지원이 반드시 필요하다. 본 장에서는 이러한 지원 기법으로 많이 사용되는 FTL(Flash Translation Layer), Flash-aware 파일시스템, 마모도 평준화, 배드 블록 관리, 하드웨어 및 소프트웨어 에러 검출 및 정정, 가상메모리 적용 등에 대하여 설명한다.

3.1 FTL / Flash-aware 파일시스템 및 마모도 평준화 기법

NAND 플래시 메모리는 앞에서 설명한 바와 같이 하드디스크와 같은 제자리 갱신(in-place update)이 불가능하고 페이지 쓰기 동작 전에 해당 페이지가 속해 있는 블록을 소거하는 작업이 선행되어져야 한다. 이러한 제약 조건을 극복하기 위해 크게 두 가지 소프트웨어적인 기법을 사용한다.

첫 번째 방법은 블록 디바이스 계층과 플래시 메모리 드라이버 사이에 매핑 계층(mapping layer)을 두고 쓰기 요청의 경우 이미 소거되어진 영역을 이용해 처리한 후 매핑 테이블을 변경해주는 기법이다. 일반적으로 FTL(Flash Translation Layer)이라고 불리는 이 기법을 이용할 경우 일반적인 블록 디바이스를 이용하는 기존의 파일시스템들을 수정 없이 직접 사용할 수 있게 된다. 매핑은 보통 플래시 메모리의 소거 블록 단위 또는 페이지 단위로 이루어지며 각각 블록 레벨 매핑, 페이지 레벨 매핑으로 불린다[2,3]. 블록 레벨 매핑의 경우는 매핑 테이블의 크기가 작지만 매핑이 변경되는 경우, 쓰기 요청이 속해 있는 소거 블록의 쓰기 요청에 포함되어 있지 않은 블록 내의 다른 모든 페이지가 복사되어져야 하는 추가적인 부담이 있다. 페이지 레벨 매핑의 경우는 효율적으로 매핑 변경을 처리할 수 있지만 매핑 테이블이 커지기 때문에 매핑 테이블 저장을 위한 플래시 메모리 공간이 많이 필요하며 메모리 사용량도 커지는 문제가 있다. 블록 레벨 및 페이지 레벨 매핑의 장점을 취합하여, 쓰기 요청의 패턴에 따라 큰 순차(large sequential) 요청의 경우는 블록 레벨 매핑을 이용하여 처리하고, 작은 크기의 비순차(small random) 요청의 경우는 페이지 레벨 매

핑을 이용하여 처리하는 하이브리드 형태의 FTL도 사용된다[4].

두 번째 방법은 플래시 메모리의 제약 조건을 고려하여(Flash-aware) 파일시스템을 설계하는 것이며 제자리 갱신이 불가능한 제약을 피하기 위하여 일반적으로 로그 구조 파일시스템(log structured file system)[5]의 형태를 가지게 된다. 이러한 Flash-aware 파일시스템은 플래시 메모리의 전체 영역을 하나의 원형 로그로 보고 쓰기 요청은 항상 로그의 시작 부분에 기록된다. 로그의 시작과 끝 사이가 쓰기 가능한 자유영역이 되며, 쓰기가 진행됨에 따라 자유영역의 크기가 일정 수준 이하로 작아질 경우 소거 동작을 통해 충분한 자유영역을 확보하게 된다. 또한 모든 쓰기 동작 및 소거 동작은 로그의 시작과 끝에서 단방향으로 일어나기 때문에 자연스럽게 마모 평준화의 효과를 얻을 수 있다는 장점이 있다. 하지만 슈퍼블록이나 아이노드와 같은 파일시스템 메타데이터의 위치 역시 고정되지 않고 계속 변하기 때문에 마운트시 플래시 메모리 전체 영역을 스캔하여 메모리에 파일시스템의 전체 구조를 구축하는 작업이 필요하며 이로 인해 마운트 시간이 길어지고 메모리 사용량이 커지는 단점이 있다.

다음 그림은 FTL을 통해 일반적인 블록 디바이스 기반 파일시스템을 사용할 수 있는 구조와 FTL 없이 직접 플래시 메모리를 접근하는 Flash-aware 파일시스템에 대한 간략한 구성도이다

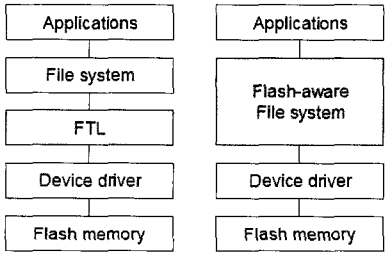

그림 1 FTL 기반 파일시스템 및 Flash-aware 파일시스템

3.2 배드 블록 관리 및 에러 정정

일반적으로 플래시 메모리는 사용 중에 블록 내의 일부 비트가 제대로 동작하지 않는 실행시간(run time) 배드 블록이 발생할 수 있으며, NAND 플래시 메모리의 경우는 공장 출하 시에도 초기 배드 블록이 존재한다. NAND 플래시 메모리를 사용할 경우에는 이러한 초기 및 실행시간 배드 블록들을 탐지하여 사용하지 않도록 하는 배드 블록 관리 기법이 필요하다. 이는 보통 배드 블록 테이블로 구현되며, 공장 출하 시 처음 사용될 때 전체 플래시 메모리의 스패어 영역의 초기 배드 블록 표시 위치를 조사하여 매핑에서 제외시키고 테이블을 플래시 메모리에 기록해 놓는 방법을 사용한다. 이 때 실행시간 배드 블록 발생시 교체 목적으로 일부 블록들을 확보해 놓고 실행시간 배드 블록 발생시 미리 확보해둔 블록들 중 하나를 선택하여 교체해준 후 테이블을 갱신하여 다시 플래시에 기록하게 된다.

NAND 플래시 메모리 사용 시 1비트 에러를 배드 블록으로 처리할 경우 실행시간 배드 블록의 개수가 너무 많아지게 되는 문제가 있다. 그러므로 일반적으로 페이지 당 1비트의 에러를 검출하고 수정할 수 있는 기법(error detection/correction)을 사용해야 하며 Reed-Solomon, BCH 기법 등이 주로 이용되고, 이를 위해 페이지의 스패어 영역이 사용된다. 이러한 기법은 소프트웨어적으로 구현되거나 성능향상을 위해 플래시 메모리 제어 하드웨어를 통하여 구현된다.

3.3 가상 메모리

플래시 메모리를 사용자 데이터를 저장하기 위한 용도 뿐 아니라 코드 실행을 위한 용도로도 사용하여 DRAM 및 SRAM의 기능을 전체 또는 일부분 대체하기 위한 연구도 진행되고 있다. 임베디드 시스템 환경에서 플래시 메모리를 이용한 메모리 시스템 구조는 비용과 에너지 소모, 성능 등의 요구조건에 따라 다양하게 발전되어 왔다. 초기에는 바이트 단위의 읽기가 가능한 NOR 플래시 메모리 상에 있는 코드를 RAM으로 옮기지 않고 직접 실행하는 XIP(eXecute-In-Place) 방식을 사용함으로써 코드 실행을 위해 요구되던 RAM의 크기를 줄일 수 있었다. 그러나 NOR XIP 방식을 이용하는 경우, NOR 플래시 메모리로부터의 읽기 속도가 RAM에서의 읽기 속도보다 느리기 때문에 RAM으로부터 실행하는 경우에 비해 성능이 저하되는 문제가 발생한다. 이에 따라 코드를 NAND나 NOR 플래시 메모리에 저장하고 있다가 부팅 시에 RAM에 모두 복사하여 RAM으로부터 코드를 실행시키는 shadowing 방식이 개발되었다. 하지만 shadowing 방식은 RAM의 사용량을 다시 증가시켰고, 그 결과 생산 단가와 에너지 소모가 증가되었다.

이러한 단점을 극복하기 위해 최근에는 임베디드 시스템에서도 플래시 메모리를 저장 매체로 사용하는 가상메모리를 활용하려는 연구가 활발히 진행되고 있다. 과거에는 생산 단가를 줄이기 위해 임베디드 프로세서에 가상메모리를 위해 필요한 메모리 관리 장치(MMU: Memory Management Unit)를 포함시키지 않았지만, 최근에 출시되는 Intel PXA255, Motorola 68060,

ARM 920T, PowerPC 440GX 등의 프로세서들은 MMU를 탑재하고 있기 때문에 임베디드 시스템에서의 가상메모리 활용 가능성이 높아지고 있다. 가상메모리를 사용하게 되면, 모든 프로그램 코드와 데이터를 RAM에 올려둘 필요가 없기 때문에 소량의 RAM으로도 시스템을 구성할 수 있다. 결국 이러한 가상메모리 기법을 이용해 시스템의 생산 단가와 에너지 소모를 효과적으로 줄일 수 있게 된다.

사용할 수 있는 메인 메모리가 매우 부족한 경우에는 코드 오버레이(code overlay) 기법을 사용할 수 있다. 코드 오버레이는 프로그래머가 직접 수작업으로 동일한 함수 호출 체인에 존재하지 않는 프로그램 내의 함수들이 실행 중에 메인 메모리 영역을 공유하도록 하는 기법이며 코드 오버레이를 자동으로 생성하는 컴파일러도 존재한다. 하지만 코드 오버레이는 코드의 종류에 따라 성능이 다르고 코드 실행을 위한 메모리 크기가 많아야 대략 20%정도 감소한다. 이와 달리 요구 페이징(demand paging) 기법을 적용하면 성능의 감소를 감수할 때 이론적으로 코드 실행을 위해 필요한 메모리의 크기를 한 페이지까지 줄일 수 있다. 플래시 메모리를 위한 요구 페이징을 지원하는 가상메모리 시스템은 임베디드 리눅스와 같은 운영체제에 이미 구현되어 있으며, 가상메모리를 응용프로그램에 특화된 형태로 자동 생성하는 기술에 대한 연구도 진행 중이다.

4. 플래시 메모리 소프트웨어 기술 현황

이 장에서는 플래시 메모리를 기반으로 하는 FTL, 파일시스템 및 가상 메모리 시스템 등에 대해 실제 사용되는 소프트웨어 및 솔루션 중심으로 기술 현황을 살펴본다.

4.1 FTL / 파일시스템 기술 현황

4.1.1 TrueFFS

M-systems의 주력 FTL인 TrueFFS[6]는 계층적인 구조를 가진다. 하드웨어로는 M-systems 사에서 제작한 DiskOnChip이 사용되며, Reed-Solomon 알고리즘에 의해 비트 에러 검출 및 정정을 수행하고, 쓰레기 수집(garbage collection)은 소거 연산을 최소화하는 것에 중점을 두고 설계되었다. TrueFFS는 FAT 테이블의 변화를 모니터링 함으로써 마모 평준화 알고리즘의 효율을 높일 수 있는 FAT 필터를 구현한다. FAT 필터는 FAT 테이블의 모든 변화를 검사하며 FAT 엔트리의 상태가 "할당" 상태에서 "가용" 상태로 바뀐 것을 감지하면 TrueFFS의 다른 계층에 해당 클러스터와 연관된 섹터들이 삭제되었음을 알린다. 이러한 정보를 통해 보다 효율적인 마모 평준화 작업을 수행할 수 있다.

4.1.2 FlashFX, Reliance 파일시스템

FlashFX와 FlashFX Pro[7]는 Datalight 사에 의해 개발된 FTL로서 100 종류 이상의 NOR 및 NAND 플래시 메모리 칩을 지원하기 위해 FIM(Flash Interface Module) 모듈이 존재한다. 이 모듈은 각각의 칩의 동작특성에 맞도록 byte 단위(x8), word 단위(x16), interleaved byte-wide 단위(2x8 또는 4x8) 또는 interleaved word-wide 단위(2x16) 등 다양한 방식으로 동작하도록 설정 가능하다. FlashFX Pro는 멀티 쓰레드로 동작하며 동 회사(Datalight)에서 제공하는 고성능의 플래시 메모리에 최적화된 파일시스템인 Reliance에 직접 적용 가능하다.

Reliance 파일시스템[8]은 내장형 시스템에 최적화되어있고, 예기치 않은 전력 단절 시에도 사용자 데이터 및 메타 데이터를 손실 없이 보존한다. 이를 위해 Reliance 파일시스템은 "committed state"와 "working state"를 가지는 two state 파일시스템 구조를 지닌다. 모든 파일시스템의 변경사항(사용자 데이터 및 메타 데이터 모두)은 현재 working state에만 기록되고 해당 연산이 완전히 commit된 후에 transaction point를 만들고 "committed state"로 변경하기 때문에 파일시스템의 신뢰성이 향상된다.

4.1.3 JFFS, JFFS2

JFFS(Journaled Flash File System)[9]는 리눅스 플래시 메모리 파일시스템으로 MTD(Memory Technology Device)[10] 계층 위에서 동작하는 로그 구조 파일시스템(log-structured file system)으로 NOR 플래시 메모리만을 지원한다. 모든 파일시스템 변경사항은 원형 로그의 형태로 기록되며 자유공간의 크기가 일정 수준 이하로 떨어질 경우 쓰레기 수집기(garbage collector)가 전체 로그를 스캔하며 유효한 노드만 모으는 작업을 통해 자유공간의 크기를 증가시킨다. JFFS2는 NAND 플래시 메모리까지 지원하며 데이터를 압축해서 저장하는 기능을 제공한다.

두 파일시스템은 모두 마운트 시에 전체 플래시 메모리를 읽어 inode chain을 구성해야 하므로 구동 시간이 오래 걸리며 메모리 사용량이 높고, 쓰레기 수집 시 성능의 급격한 저하가 발생하는 단점이 있다

4.1.4 YAFFS

YAFFS(Yet Another Flash File System)는 초기 설계단계에서부터 NAND 플래시 메모리만을 대상으로 개발된 파일시스템이다[11]. YAFFS 역시

JFFS2처럼 기본적으로 로그 구조 파일시스템이며 일반적으로 JFFS2보다 높은 성능을 보인다. 현재 리눅스와 WinCE에서 동작할 수 있으며, 실시간 운영체제(RTOS)로 쉽게 포팅할 수 있도록 한 YAFFS/direct 버전이 있다. 최근 많이 사용되는 대블록 NAND 플래시 메모리를 위해 YAFFS2 버전이 개발되었다.

YAFFS에서 모든 사용자 데이터는 일정 크기의 chunk 단위로 기록되며(보통 NAND 플래시 메모리의 한 페이지와 같은 크기인 512바이트), 해당 페이지의 스패어 영역에 파일 id와 chunk 번호가 기록된다. 마운트 시에는 역시 전체 플래시 메모리 영역을 스캔하는 작업이 필요하지만 JFFS2와 달리 단지 스패어 영역만 검사하므로 마운트 시간이 훨씬 빠르다.

4.1.5 PocketStore/Unistore/TFS4, ZFS/Z-Flash

우리 나라는 NAND 플래시 메모리 생산 및 판매량에서는 우위를 점하고 있지만 소프트웨어 지원은 아직 미비한 실정이다. 국내에서 플래시 메모리 파일시스템 솔루션을 자체 개발하여 상용화에 성공한 기업으로는 삼성전자와 지인정보기술이 잘 알려져 있다. 삼성전자는 FTL에 해당하는 PocketStore(Windows CE 기반)[12], Unistore(Symbian 기반)[13] 및 FAT 호환 파일시스템인 TFS4[14] 제품군을 가지고 있다. 지인정보기술의 ZFS 파일시스템 솔루션[15]은 FTL과 파일시스템의 기능이 분리된 구조를 지니며 저널링 기법을 사용하여 고신뢰성 파일시스템을 제공한다. Z-Flash 파일시스템 솔루션은 FTL과 파일시스템의 기능이 통합된 구조이며, 로그 구조 파일시스템의 문제점(느린 부팅 시간, 높은 메모리 사용량)을 극복하며 신뢰성을 보장하기 위해 트랜잭션 기반으로 동작하는 플래시 메모리 파일시스템이다.

4.2 가상메모리 기술 현황

NOR XIP 모델과 shadowing 모델은 상용화 되어 실제 제품에 이용되고 있다. 최근에는 NAND 플래시 메모리 시장에서 높은 시장 점유율을 차지하고 있는 삼성전자와 하이닉스를 중심으로 NAND 플래시 메모리에 저장된 코드를 요구 페이징을 통해 수행하는 기술에 대해 부분적으로 연구가 진행되고 있는 것으로 알려져 있다.

현재 플래시 메모리를 위한 가상메모리 시스템을 응용프로그램에 특화된 형태로 자동 생성하는 기술은 아직까지 존재하지 않는다. 하지만 삼성전자와 서울대학교의 공동작업으로 NAND 플래시 메모리를 위한 FTL에 특화된 요구 페이징 시스템을 자동으로 생성하여 프로그램에 내장하는 기법을 개발하고, MMU가 없는 시스템에서 FTL에 적용한 사례가 있다[16].

5. 최근 동향

플래시 메모리는 개발 초기에는 소형 임베디드 장비의 데이터 저장소나 재기록 가능한 펌웨어등에 많이 사용되었다. 하지만 미세 공정 및 기술 발전에 따라 급속하게 용량이 증대되고 있으며, 수요 증대에 따른 대량 생산으로 인해 가격 역시 급속하게 하락하고 있는 추세다. 또한 PMP(Personal Media Player), UMPC(Ultra-Mobile PC) 등이 빠르게 보급되고 있기 때문에 대량의 멀티미디어 데이터 저장을 위해 플래시 메모리가 많이 이용되고 있다. 이렇게 달라진 환경에서 플래시 메모리의 적용 범위는 점점 커지고 있다. 본 장에서는 노트북 또는 일반 데스크탑 컴퓨팅 환경에서의 플래시 메모리 적용 연구를 소개하고, 대용량 플래시 메모리 생산에 많이 사용되는 MLC(Multi-Level Cell) 플래시 메모리 및 이에 따른 소프트웨어, 하드웨어적 고려사항을 설명한다.

5.1 하이브리드 하드디스크

최근에는 NAND 플래시 메모리와 하드디스크를 결합하여 성능을 향상시키고 전력 소모를 줄이는 연구가 진행 중이다. 하이브리드 하드디스크(Hybrid hard disk)는 하드디스크 내부에 소량의 NAND 플래시 메모리를 추가한 형태로 구성되며[17] 이를 통해 다음과 같은 장점을 얻게 된다.

5.1.1 성능향상

하드디스크는 큰 순차 접근(large sequential access)는 높은 성능을 얻을 수 있지만 작은 비순차 접근(random small access)의 경우에는 디스크 헤드의 탐색 시간(seek time)이 커지기 때문에 낮은 성능을 보이게 된다. 하이브리드 하드디스크에서는 작은 임의 접근 데이터를 NAND 플래시 메모리에 저장할 수 있기 때문에 전체적인 성능 향상을 기대할 수 있다.

5.1.2 디스크 스핀다운 시간 증대에 따른 전력 소모 감소

일반적으로 전력 소모를 줄이기 위해, 일정 시간 이상 하드디스크에 대한 요청이 없을 경우 디스크 플래터의 회전을 멈추게 된다(spin down). 하이브리드 하드디스크는 디스크가 멈춘 상태에서도 플래시 메모리에 자유 영역(free space)이 남아있는 동안은 쓰기 요청을 처리할 수 있기 때문에 하드디스크 스핀다운 상태를 장시간 유지할 수 있고 결국 전력 소모를 더욱 줄일 수 있다. 또한 스핀다운 상태를 오래 유지하는 것은 기계적인 동작을 적게 하는 것이기 때문에 장기적

으로 MTBF(Mean Time Between Failures)를 높이는 부가적인 효과도 얻게 된다.

5.1.3 부팅 시간 감소

시스템 종료 시 운영체제는 확장된 ATA 코맨드를 이용하여 다음 부팅 시 필요한 데이터들을 하이브리드 하드디스크 내의 플래시 메모리에 위치시키게 할 수 있다. 이를 통해 다음 부팅 시 디스크가 완전히 스핀업 되지 않은 상태에서도 필요한 데이터들을 플래시메모리에서 즉시 읽어드릴 수 있게 되고 시스템 부팅 시간을 단축할 수 있게 된다. 비슷한 방법으로 하이버네이션 상태에서 빠르게 깨어나는 것이 가능해진다.

하이브리드 하드디스크는 삼성전자와 마이크로소프트에 의하여 주도적으로 개발되고 있으며 ReadyDrive란 이름으로 2007년 초 출시 예정인 Windows Vista에 의해 지원될 예정이다. 하이브리드 하드디스크를 위한 ATA 확장 명령어 세트(ATA extended command set)인 "NV Cache Feature Set"은 ATA 표준을 담당하는 T13 위원회에 의해 ATA8 표준에 포함되었다. 삼성전자와 마이크로소프트는 2006년 WinHEC을 통해 256MB 크기의 OneNAND 플래시 메모리를 사용한 하이브리드 하드디스크를 시연하였다.

5.2 플래시 메모리 SSD(Solid State Disk)

플래시 메모리 SSD는 하드디스크와 달리 기계적인 동작이 없는 순수 반도체 장치인 플래시 메모리만으로 구성되기 때문에 전력소모가 적고 소음이 적으며, 충격에 강하다. 일반적으로 SATA 및 PATA 인터페이스를 지원하기 때문에 호스트 시스템은 플래시 메모리 SSD를 일반적인 하드디스크로 인식하게 되고 섹터 단위 읽기 및 쓰기 요청을 하게 되며 이를 처리하기 위해 내부적으로 FTL을 사용하게 된다. 플래시 메모리의 특성상 작은 비순차 접근 요청에 대한 응답 시간이 매우 짧으며, 큰 순차 접근 시의 성능 향상을 위해 다수의 칩과 다수의 플래시 버스를 구성하고 칩 레벨 인터리빙 및 버스 레벨 인터리빙을 통해 성능을 극대화하는 방법이 이용된다.

아직까지는 비용적인 면에서 하드디스크가 유리하기 때문에, 플래시 메모리 SSD는 우선 고사양 노트북이나 UMPC 등의 모바일, 저전력 환경에서의 대용량 저장 장치로 먼저 적용될 것으로 예상되며, 플래시 메모리의 가격이 여전히 큰 폭으로 하락하고 있기 때문에 가격경쟁력도 더욱 커질 것으로 예상된다.

msystems, Adtron, Memtech, BiTMICRO 등이 플래시 메모리 SSD를 생산하는 주요 업체들이며 하드디스크에 비해 아직도 가격차가 크기 때문에, 진동에 약한 하드디스크를 사용할 수 없는 군사용 장비 등에 주로 사용되고 있다. 2006년에 삼성전자는 16GB 플래시 메모리 SSD를 개발하여 소니 및 자사의 노트북 및 UMPC 일부 모델에 장착하기 시작했다.

5.3 MLC(Multi Level Cell) 플래시 메모리

최근에 등장한 MLC(Multi Level Cell) 플래시 메모리는 한 셀로 1비트를 표현하는 기존의 SLC (Single Level Cell) 플래시 메모리와 달리 한 셀로 2비트 이상을 표현하는 구조를 가진다. 이러한 구조를 통해 MLC 플래시 메모리는 SLC 플래시 메모리에 비해 집적도를 높일 수 있기 때문에 가격대 용량 면에서 유리하지만 동작속도가 SLC 플래시 메모리에 비해 다소 낮다. 또한 하나의 메모리 셀로 네 가지 상태(state) 이상을 표현하는 구조상 비트 에러가 일어날 가능성이 크기 때문에 안정성 면에서도 불리한 면이 있다. 소거 가능 횟수에서도 SLC NAND 플래시 메모리가 일반적으로 100,000 번까지의 소거 횟수를 보장하는 반면에 MLC NAND 플래시 메모리는 10,000 번까지의 소거 횟수를 보장한다. 그러므로 안정적인 동작을 위해서는 페이지 당 최대 4비트 에러까지 정정할 수 있는 ECC 알고리즘을 사용하여야 하며 더욱 정교한 마모도 평준화 정책이 필요하다. 또한 SLC NAND 플래시 메모리에서와 같은 내부 버퍼를 이용해 데이터를 복사하는 고급기능인 copyback 기능이 지원되지 않는다.

플래시 메모리가 SSD와 같은 대용량 저장장치로 사용되기 위해서는 MLC 플래시 메모리의 사용이 더욱 증대될 것으로 보이며 SLC 플래시 메모리와의 차이점을 고려한 FTL 및 파일시스템 설계가 필요하다.

6. 결 론

플래시 메모리는 유비쿼터스 컴퓨팅을 지향하는 최근의 컴퓨터 산업 발전 추세에 따라 초기의 소형디바이스의 저장매체로서의 역할뿐 아니라 다양한 임베디드 및 모바일 장치들의 저장 매체로서도 광범위하게 사용되고 있으며 최근에는 하드디스크를 대체할 수 있는 대용량 저장장치로서도 충분한 적용 가능성을 보여주고 있다.

본 논문에서는 현재 가장 많이 사용되고 있는 NOR 플래시 메모리 및 NAND 플래시 메모리의 특성 및 제약 사항들을 설명하고 이에 따른 하드웨어 및 소프트웨어 지원 기법 및 실제 상용 솔루션들을 살펴보았다. 또한 최근 활발히 연구되고 있는 하이브리드 하드디스크 및 플래시 메모리 SSD에 대하여 설명한 후, MLC

플래시 메모리의 특성 및 그에 따른 소프트웨어 설계 시의 고려 사항에 대하여 알아보았다.

우리나라는 세계 최고의 플래시 메모리 생산국이지만 소프트웨어 지원 면에서는 세계 수준에 많이 뒤처져 있다. 플래시 메모리 산업의 국가 경쟁력을 더욱 높이고 균형 있게 발전하기 위해서는 소프트웨어 지원 기술의 발전이 매우 중요한 시점이다.

참고문헌

[1] B. Kim, S. Cho, Y. Choi, and Y. Choi, "OneNAND(TM): A High Performance and Low Power Memory Solution for Code and Data Storage," Proc. 20th Non-Volatile Semiconductor Workshop, 2004.

[2] A. Ban, "Flash File System," US Patent, no. 5,404,485, Apr. 1995.

[3] Intel Corp., "Understanding the Flash Translation Layer (FTL) Specification," http://developer.intel.com/, 1998.

[4] 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, Vol. 48, No. 2, pp.366-375, 2002.

[5] M. Rosenblum, J. K. Ousterhout, "The design and implementation of a log-structured file system," ACM Transactions on Computer Systems, pp. 26-52, Feb. 1992.

[6] msystems, http://www.m-systems.com/site/en-US/Technologies/Technology/TrueFFS_Technology.htm

[7] Datalight, http://datalight.com/products/flashfx/

[8] Datalight, http://datalight.com/products/reliance/

[9] D. Woodhouse, "JFFS: The Journalling Flash File System," Proc. Ottawa Linux Symp., 2001.

[10] MTD, "Memory Technology Device (MTD) Subsystem for Linux," http://www.linux-mtd.infradead.org

[11] Aleph One, "YAFFS: Yet another flash filing system," http://www.aleph1.co.uk/yaffs/, 2002.

[12] Samsung Electronics, http://www.samsung.com/Products/Semiconductor/FlashSW/pocketstore.htm

[13] Samsung Electronics, http://www.samsung.com/Products/Semiconductor/FlashSW/unistore.htm

[14] Samsung Electronics, http://www.samsung.com/Products/Semiconductor/FlashSW/tfs4.htm

[15] Zeen, http://zeen.snu.ac.kr/

[16] Chanik Park, Junghee Lim, Kiwon Kwon, Jaejin Lee, Sang Lyul Min, "Compiler-assisted demand paging for embedded systems with flash memory," In Proceedings of the 4th ACM International Conference on Embedded Software (EMSoft '04), pp. 114-124, Pisa, Italy, September 2004.

[17] J. Creasey, "Hybrid Hard Drives with Non-Volatile Flash and Longhorn," WinHEC 2005.

이 수 관

2000 서울대학교 컴퓨터공학과(학사)
2002~현재 서울대학교 컴퓨터공학과 석박사 통합과정
관심분야 : 운영체제, 내장형 시스템, 플래시 메모리
E-mail : sklee@ssrnet.snu.ac.kr

민 상 렬

1983 서울대학교 컴퓨터공학과(학사)
1985 서울대학교 컴퓨터공학과(석사)
1989 University of Washington (전산학박사)
1989~1990 IBM T.J.Watson Research Center 객원 연구원
1990~1992 부산대학교 컴퓨터공학과 조교수
1992~현재 서울대학교 전기컴퓨터공학부 교수
관심분야 : Computer Architecture, Parallel Processing, Computer Performance Evaluation
E-mail : symin@archi.snu.ac.kr

조 유 근

1971 서울대학교 건축공학과 학사
1978 미네소타대학교 컴퓨터과학 박사
1979~현재 서울대학교 컴퓨터공학부 교수
1984~1985 미네소타대학교 교환 교수
1993~1995 서울대학교 중앙교육연구 전산원장
1999~2001 서울대학교 공과대학 부학장
2001~2002 한국정보과학회 회장
관심분야 : 운영체제, 알고리즘 설계 및 분석, 암호학
E-mail : ykcho@snu.ac.kr

• 한국소프트웨어공학 학술대회 •

• 일 자 : 2007년 2월 22~24일
• 장 소 : 용평리조트
• 내 용 : 논문발표 등
• 주 최 : 소프트웨어공학연구회
• 상세안내 : http://selab.skku.ac.kr/~kcse07