NAND 플래시 메모리 파일 시스템의 설계

박송화° 이주경 정기동
부산대학교 컴퓨터공학과
{downy25°, jklee}@melon.cs.pusan.ac.kr, kdchung@pusan.ac.kr

Design of a File System on NAND Flash Memory

Song-hwa Park° Joo-Kyong Lee Ki-Dong Chung
Dept. of Computer Engineering, Pusan National University

요 약

본 논문은 임베디드 시스템에서 대용량 저장시스템에 적합한 NAND 플래시 메모리 기반의 파일 시스템을 제안한다. 플래시 메모리는 비휘발성이며 기존의 하드디스크와 같은 자기 매체에 비해서 크기가 작고 전력소모도 적으며 내구성이 높은 장점을 지니고 있다. 그러나 제자리 덮어쓰기(update-in-place)가 불가능하고 메모리 셀에 대한 초기화 횟수가 제한되는 단점이 있다. 또한 NAND 플래시 메모리는 바이트 단위의 입출력이 불가능하다. 이러한 특성 때문에 NAND 플래시 메모리를 저장장치로 사용할 경우 기존의 저장 장치 관리 방법과는 다른 방법을 요구한다. 본 논문은 임베디드 시스템에서 대용량 저장장치를 위한 NAND 플래시 메모리 기반의 파일 시스템의 구조를 제시하고, 대용량 파일 지원을 위한 메타 데이터 구조와 데이터 수정 기법을 제안한다.

1. 서 론

임베디드 시스템(Embedded system)은 간편하게 조작하고 쉽게 데이터를 보관할 수 있는 저장 매체를 요구한다. 휴대가 용이하고 빠른 접근 시간을 요구하는 임베디드 시스템의 특성상 부피가 크며 소비전력이 큰 하드디스크를 저장매체로 사용하는 것은 비효율적이다. 이와 같은 이유로 다음과 같은 장점을 가진 플래시 메모리의 사용이 증가하고 있다. 첫째, 플래시 메모리는 비휘발성(non-volatile)이며 ROM과 달리 재사용이 가능하다. 둘째, 플래시 메모리는 크기가 작기 때문에 소형화 기기에 적합하고 대용량 데이터 처리가 가능하다. 셋째, 플래시 메모리는 임의 접근(random access)이 가능하기 때문에 입출력 속도가 빠르다.

그러나 플래시 메모리를 저장 시스템으로 사용하기 위해서는 두 가지 단점이 있다. 첫째, 플래시 메모리는 데이터 수정 시 본래 주소에 덮어쓰기(update-in-place)가 불가능하다. 플래시 메모리의 각 비트가 단방향으로만 토글링(toggling)되기 때문에 쓰기 연산 시 초기화 연산을 선행해야 한다. 즉, 쓰기 연산 전에 초기화된 메모리 공간의 확보가 필요하다는 것을 의미한다. 둘째, 플래시 메모리의 각 블록은 초기화 연산의 횟수가 제한되어 있기 때문에 플래시 메모리의 전체 공간이 균등하게 사용되지 못하는 경우에는 사용가능한 메모리 공간이 급격히 줄어들게 된다[1].

본 논문에서는 플래시 메모리의 특성과 관련한 파일 시스템을 설계하고, 설계한 파일 시스템의 메모리 공간을 효율적으로 사용하기 위한 메타 데이터 구조 및 데이터 수정 기법을 제안한다. 제안된 데이터 수정 기법은 플래시 메모리에 쓰는 양을 줄임으로써 플래시 메모리의 수명을 연장한다.

본 논문의 구성은 다음과 같다. 2장에서는 플래시 메모리의 상세한 내용을 기술하고, 3장에서는 대용량 저장 매체를 위한 플래시 메모리 파일 시스템 구조를 제안한다. 4장에서는 제안된 파일 시스템의 성능을 평가하고 5장에서 이 논문의 결론과 향후 과제를 제시한다.

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

2. 플래시 메모리

플래시 메모리는 일종의 EEPROM(Electrically Erasable and Programmable ROM)으로 셀(cell)을 구성하는 구조에 따라 크게 NOR 플래시 메모리와 NAND 플래시 메모리로 구분된다. [표 1]은 NOR 플래시 메모리와 NAND 플래시 메모리의 연산 속도를 비교한 것이다[2].

[표 1] NOR 플래시 메모리와 NAND 플래시 메모리의 비교

종류	접근 시간		
	읽기	쓰기	지우기
NOR 플래시	150ns (1B) 14.1μs (512B)	211μs (1B) 3.53ms (512B)	1.2s (128KB)
NAND 플래시	10.2μs (1B) 35.9μs (512B)	201μs (1B) 226μs (512B)	2ms (16KB)

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.

최근의 많은 응용기기들은 대용량의 저장시스템을 필요로 하

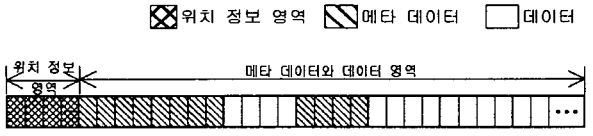
는 경우가 많다. 예를 들어, 휴대전화나 PDA와 같은 이동형 기기들의 멀티미디어 데이터에 대한 수요가 늘어남에 따라 이를 저장하기 위해 많은 저장 공간을 필요로 하고 있으며 다양한 기능을 내장하기 위해 보다 많은 시스템 자원을 저장할 저장 장치의 필요성이 증가하고 있다. NOR 플래시 메모리는 집적도가 낮고 고가이기 때문에 이러한 요구에 적합하지 못하다. 반면에 NAND 플래시 메모리는 단일 칩으로도 대용량이며 비용 역시 저렴하기 때문에 NAND 플래시 메모리를 이용하여 응용기기를 만들게 되면 저장용량이 크게 증가하고, 기기의 비용도 낮출 수 있다. 이러한 이유로 NAND 플래시 메모리의 용도가 확대됨에 따라 NAND 플래시의 기술이 활발히 연구되고 있으며, 시장점유율도 증가하고 있다[3].

3. 플래시 메모리 기반의 파일 시스템의 설계

플래시 메모리를 사용하는 파일 시스템에서는 제자리 덮어쓰기가 불가능하므로 데이터를 변경하고자 하는 경우, 변경하고자 하는 데이터가 들어있는 블록을 무효화하고 초기화된 블록을 할당받아 수정된 데이터를 저장한다. 이때 크기가 큰 파일의 데이터를 변경할 경우 전체 데이터를 다시 써주어야 하므로 플래시의 메모리 공간이 낭비되며 플래시의 수명이 단축 된다. 본 논문에서는 이러한 상황이 발생하는 것을 막기 위한 플래시 파일 시스템의 구조와 데이터 수정 기법을 제안한다.

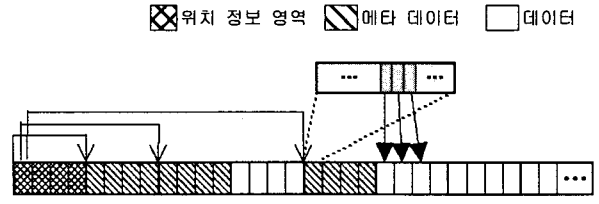
3.1 시스템 구조

본 연구에서는 로그 구조 파일 시스템(Log-Structured File System, LFS)[4]에 기반을 두고 저장 시스템을 설계하였다.



[그림 1] 제안된 플래시 메모리 파일 시스템의 구조

[그림 1]과 같이 전체 플래시 메모리 공간을 위치 정보를 저장하는 영역과 메타 데이터와 데이터를 저장하는 영역으로 나눈다. 위치 정보 영역은 고정되어 있으며 메타 데이터와 데이터는 위치 정보 영역을 제외한 플래시 메모리의 공간을 각각 세그먼트와 블록 단위로 할당받는다. 읽기/쓰기 연산은 페이지 단위로 수행하며, 삭제 연산은 블록 단위로 수행한다.



[그림 2] 데이터 쓰기 과정

제안하는 파일 시스템에서 데이터 쓰기 과정은 [그림 2]와 같다. 데이터 쓰기 연산이 발생하면 플래시 메모리에 메타 데이터를 먼저 기록한 후, 실제 데이터를 기록한다. 데이터 수정 시, 제자리 덮어쓰기가 불가능한 플래시 메모리의 특성으로 인해 메타 데이터의 위치가 빈번히 바뀌게 되므로 메타 데이터에 대한 위치 정보를 위치 정보 영역에 기록한다. 이것은 파일 시스템 마운트 시, 메타 데이터 영역의 위치 정보를 제공하여 파일 시스템의 마운트 시간을 감소시킨다. 또한 메타 데이터의 데이터 위치 정보를 사용하여 파일 시스템 마운트 시 각 파일의 위치 정보를 메모리상에 유지할 수 있다. 파일의 위치 정보를 메모리에 유지함으로써 파일 연산 발생 시 파일에 빠르게 접근할 수 있다.

3.2 메타 데이터 구조

제안된 시스템에서의 메타 데이터의 구조는 [표 2]와 같다. 메타 데이터는 파일에 대한 정보를 가지며 데이터의 페이지 수만큼 데이터의 위치에 대한 정보를 포함한다. 메타 데이터는 하나의 페이지에 기록되므로 하나의 메타 데이터가 나타낼 수 있는 데이터의 크기는 한정되어, 결과적으로 지원하는 파일의 크기에 제한을 두게 된다. 이러한 문제점을 해결하기 위해 하나의 파일에 대한 메타 데이터들을 링크드 리스트(linked-list) 구조로 유지하며 지원 가능한 파일의 최대 크기는 4GB 이다.

[표 2] 제안된 시스템에서의 메타 데이터의 구조

Byte	정보
4	파일 아이디(id)
4	파일 타입
4	상위 디렉토리(directory)
2	파일 이름의 체크 합(check sum)
8	파일 이름
4	파일 권한
4	파일 소유자의 사용자 아이디
4	파일 소유자의 그룹 아이디
4	파일의 마지막 접근 시간
4	파일의 마지막 수정 시간
4	파일의 마지막 변경 시간
4	파일 오프셋(offset)
4	파일 크기
4	동일 파일 아이디
1	링크(link) 수
8	파일 별칭(alias)
4	이전 메타 데이터의 위치
4	파일의 데이터 위치 정보 1
…	…
4	파일의 데이터 위치 정보 n

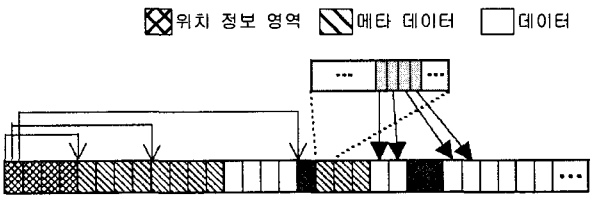
3.3 데이터 수정

제안된 파일 시스템에서의 데이터 수정은 다음과 같은 절차로 수행한다.

(1) 수정되기 이전의 파일 데이터를 읽어온다.
(2) 수정하려는 파일의 데이터와 (1)에서 읽어온 데이터를 비교하여 수정되는 데이터의 위치와 크기를 조사한다.
(3) 수정되는 데이터를 기록할 영역을 할당한다.
(4) 메타 데이터의 정보를 갱신하여 플래시 메모리에 저장한다. 이때, 메타 데이터를 위해 새로운 세그먼트를 할당받은 경우에는 위치 정보 영역에 갱신된 메타 데이터의 위치를 기록한다.
(5) (3)에서 할당받은 데이터 영역에 수정된 데이터 기록이 완료되면 수정되기 이전의 데이터를 무효화한다.

[그림 3]은 [그림 2]에서 파일의 일부분에 데이터를 수정하여 일부 데이터가 변경되고, 새로운 데이터가 추가되어 파일의

크기가 커지는 경우 일어나는 연산의 과정을 보여준다. 이러한 경우, 수정된 데이터와 추가된 데이터 및 갱신된 데이터만 플래시 메모리에 기록한다.



[그림 3] 데이터 수정 과정

4. 실험 및 성능 평가

4.1 시뮬레이션과 작업부하의 설계

성능 분석을 위해 본 논문에서는 저장 매체로서 40개의 블록으로 구성된 64MB 용량의 플래시메모리를 기반으로 본 논문에서 제안한 파일 시스템을 시뮬레이션 하였다. 각 블록은 32개의 페이지로 구성하였고, 나머지 실험 인자는 [표 3]에 기술한다.

[표 3] 시뮬레이션 파라미터

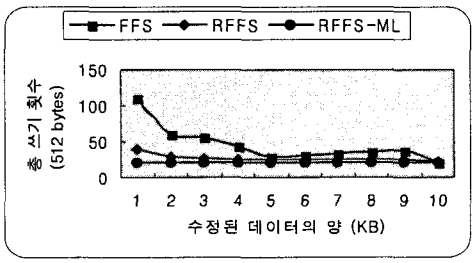
플래시 메모리 관련 파라미터	
플래시 메모리 용량	64 Mbytes
세그먼트 크기	64 Kbytes
블록 크기	16 Kbytes
페이지 크기	512 bytes

본 실험에서는 수정된 데이터만 플래시 메모리에 쓰는 경우의 실험결과를 평가한다. 실험은 두 가지 유형의 작업부하(workload)를 사용하였다. 한 유형은 파일 생성 후, 1KB 단위로 데이터를 수정하여 파일의 크기가 10KB가 될 때 총 쓰기 횟수를 측정하였다. 나머지 하나의 유형은 파일 생성 후, 10 킬로바이트 단위로 데이터를 수정하여 파일의 크기가 100KB가 될 때까지의 총 쓰기 횟수를 측정하였다.

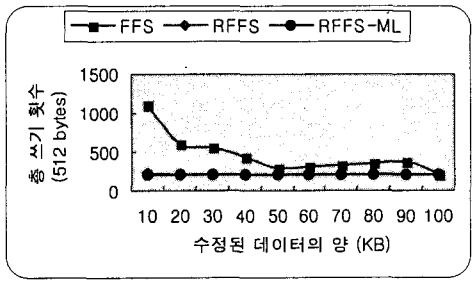
4.2 실험 결과

본 논문에서는 플래시 메모리의 쓰기 양을 감소시키는데 중점을 두기 때문에 제안한 기법의 성능평가를 위해 데이터 수정 시 모든 데이터를 다시 쓰는 기법을 기준으로 삼는다. 이 기법을 FFS로 명명한다. 제안된 기법은 수정된 데이터만을 기록한 후, 메타 데이터와 위치 정보 데이터를 기록한다. 제안된 기법을 RFFS로 명명한다. 또한 제안된 기법에서 메타 데이터와 위치 정보 데이터 쓰기의 영향을 알아보기 위해 데이터 수정 시 수정된 데이터만 기록하고 메타 데이터와 위치 정보 데이터는 기록하지 않는 경우도 비교한다. 이 기법을 RFFS-ML이라고 한다.

[그림 4]는 각 작업부하에 대해 여러 기법을 수행했을 때의 총 쓰기 횟수를 보여준다. 그림 (a)는 파일의 끝에 추가되는 데이터가 1KB 단위이고, 파일의 크기가 10KB가 될 때까지의 총 쓰기 횟수를 보여준다. 그림 (b)는 파일의 끝에 추가되는 데이터가 10KB 단위이고, 파일의 크기가 100KB가 될 때까지의 총 쓰기 횟수를 보여준다. 두 경우 모두 본 논문에서 제안한 기법이 우수한 성능을 보였다. 또한 RFFS와 RFFS-ML을 비교하면 메타 데이터와 위치 정보 데이터로 인한 오버헤드가 크지 않다는 것을 확인할 수 있다. 그림 (a)와 그림 (b)에서 수정되는 데이터의 양이 많을수록 메타 데이터와 위치 정보 데이터로 인한 오버헤드가 상대적으로 낮아짐을 확인할 수 있다.



(a) 수정된 데이터의 단위: 1KB



(b) 수정된 데이터의 단위: 10KB

[그림 4] 수정된 데이터의 양에 따른 총 쓰기 횟수

5. 결론 및 향후과제

본 논문에서는 플래시 메모리의 수명 연장을 위해 메타 데이터와 데이터를 수정하는 새로운 기법을 제안하였다. 플래시 메모리를 저장 매체로 사용할 때의 중요한 문제는 플래시 메모리의 쓰기 양을 줄임으로써 수명을 연장시키는 것인데 이는 데이터 수정 시 수정되는 데이터만 기록하고 파일의 데이터의 위치를 메타 데이터에 기록함으로써 해결한다. 결과적으로 제안된 데이터 수정 기법은 플래시 메모리에 쓰기 양을 줄임으로써 플래시의 수명을 연장시킬 수 있다.

향후에는 좀 더 효율적인 데이터 수정 기법을 위한 연구가 진행되어야 할 것이다.

참고문헌

[1] 김한준, 이상구, "신뢰성 있는 플래시메모리 저장시스템 구축을 위한 플래시메모리 저장 공간 관리 방법", 정보과학회 논문지(시스템 및 이론), 27(6), pp.567-582, 2000

[2] A Space-efficient Flash Translation Layer for CompactFlash Systems, Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, IEEE Transactions on Consumer Electronics, May 2002

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

[4] 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