빠른 마운트와 복구를 지원하는 NAND 플래시 파일 시스템 설계

진종원°, 이태훈, 정기동
부산대학교 컴퓨터공학과
e-mail:{waller°, withsoul}@melon.cs.pusan.ac.kr, kdchung@pusan.ac.kr

Design of NAND Flash File System for Fast Mount and Recovery

Jong-Won Jin°, Tae-Hoon Lee, Ki-Dong Chung
Dept. of Computer Engineering, Pusan National University

요 약

플래시 메모리는 비휘발성, 저 전력, 빠른 입출력, 충격에 강함 등과 같은 많은 장점을 가지고 있으며 모바일 기기에서의 저장 매체로 사용이 증가 되고 있다. 뿐만 아니라 기존 하드디스크를 대체하는 용도로도 사용하고 있다. 하지만 제자리 덮어쓰기가 불가능하고 지움 연산의 단위가 크다는 제약 및 블록의 지움 횟수 제한이 있다. 이러한 제약을 극복하기 위해 YAFFS와 같은 로그 구조 기반의 플래시 파일 시스템들이 개발 되었다. YAFFS와 같은 로그 구조 기반의 플래시 파일 시스템은 마운팅시에 시스템에 필요한 데이터들을 얻기 위해 전체 플래시 메모리를 읽어야 한다. 이러한 파일 시스템의 마운팅 과정은 전체 시스템의 부팅 시간을 지연시킨다.

본 논문에서는 위와 같은 문제점 해결을 위하여 빠른 부팅을 제공 할 수 있는 NAND 플래시 파일 시스템 구조 및 제안한 구조에서의 시스템 일관성 유지를 위한 빠른 복구 방법들을 제안한다.

1. 서론

PDA, PMP와 RFID 리더 등의 모바일 기기에서의 저장매체로 플래시 메모리 사용이 증가하고 있다. 플래시 메모리는 EEPROM의 일종으로 비휘발성, 저 전력, 충격에 강함 및 빠른 처리 속도 등의 장점이 있고 또한 용량이 커지고 가격은 낮아짐으로서 더욱 활용성이 증가 하였다.

본 논문에서 대상으로 하고 있는 저장 매체인 플래시 메모리는 NAND 플래시 메모리이다. 플래시 메모리는 블록으로 구성이 되며 각 블록은 다시 페이지로 구성이 된다. 그리고 페이지는 data영역과 spare영역으로 구분되어 구성이 되고 읽기 속도와 쓰기 속도는 [표 1]과 같다.[1]

[표 1]에서 보는 것과 같이 플래시 메모리는 읽기와 쓰기 처리시간이 각각 다르고 지움 시간이 존재한다.

이 논문은 2단계 두뇌한국21사업에 의하여 지원되었음.
This work was supported by the Brain Korea 21 Project in 2007.

[표 1] 저장 매체별 처리속도

저장매체	처리시간(512Bytes)		
	Read	Write	Erase
DRAM	2.56us	2.56us	N/A
NOR flash	14.4us	3.53ms	1.2s(128KB)
NAND flash	35.9us	226us	2ms(16KB)
DISK	12.4ms	12.4ms	N/A

그러나 플래시 메모리는 제자리 덮어쓰기가 불가능하고 블록의 지움 횟수에 제한이 있다. 또한 본 논문에서 대상으로 하고 있는 NAND 플래시 메모리는 지움 연산의 단위가 읽기·쓰기 단위보다 크다는 단점이 존재한다.

이러한 NAND 플래시 메모리의 단점을 극복하기 위해 YAFFS[2]와 같은 NAND 플래시 메모리 전용 파일 시스템이 개발되었다. YAFFS는 로그 구조 기반의 파일 시스템으로 NAND 플래시 메모리 사용을 위해 개발되었다. YAFFS는 로그 구조를 기반으로 하여 갱신 연산 발생시, 갱신된 데이터를 다른 위치에 기록하는 외부 갱신

기법(Out-place Update)방법을 사용한다. 따라서 마운팅 시, 시스템에 필요한 데이터를 얻기 위해 전체 플래시 메모리 공간을 읽어서 정보를 유지하게 된다. 이러한 파일 시스템의 마운팅 과정은 전체 시스템의 부팅 시간을 지연시킨다. 또한 YAFFS는 플래시 메모리의 블록별 균등화 된 사용을 위한 고려를 하지 않고 있어 플래시 메모리의 균등화된 사용을 지원하여야 한다.

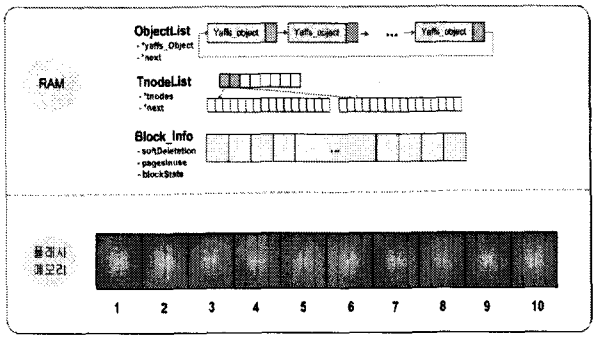
이러한 문제점을 해결하기 위해 본 논문에서는 로그 구조 기반의 NAND 플래시 메모리 파일시스템의 빠른 부팅을 제공하고 균등화된 사용을 지원 할 수 있는 NAND 플래시 파일 시스템의 구조를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 플래시 메모리의 특성과 기존의 NAND 플래시 파일 시스템인 YAFFS에 대해 기술하고, 3장에서는 제안하는 NAND 플래시 파일시스템의 구조, 마운팅 기법 및 복구 기법에 대해 설명하며, 4장에서는 제안한 기법들의 성능을 실험을 통해 비교·분석하고 5장에서 결론을 맺는다.

2. 관련연구

NAND 플래시 메모리를 위한 대표적인 파일 시스템으로는 YAFFS(Yet Another Flash File System)가 있다. YAFFS는 로그 구조를 기반으로 하여 플래시 메모리에 대한 갱신 연산을 외부 갱신 기법을 사용한다. 이를 통해 플래시 메모리의 제자리 덮어쓰기가 되지 않는 문제점을 해결하였다[3]. 하지만 갱신된 데이터가 다른 곳에 기록 됨으로써 기존의 데이터가 플래시 메모리 공간에 그대로 남아 있게 된다. 또한 갱신 연산이 빈번히 발생할 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 있게 된다. 이러한 과정은 YAFFS의 마운팅 시간을 크게 지연시키는 요인이다.

[그림 1]은 YAFFS가 사용하는 자료 구조를 보여 주고 있다. 플래시 메모리에는 데이터들이 쓰기, 갱신 작업에 따라 외부 갱신 방법으로 데이터들이 쓰여지게 되고 파일시스템은 RAM상에 파일시스템내 생성된 오브젝트를 관리하는 Obejct list, 파일의 물리적 데이터 위치를 저장하는 Tnode 및 각 블록별 정보를 저장하는 BlockInfo 등의 자료구조를 통하여 플래시 메모리상의 데이터들을 유지, 관리 한다. 이러한 정보들은 파일시스템의 마운트 과정에서 플래시 메모리의 전체영역을 읽어서 구축을 하게 된다. 마운트 과정에서는 최초의 블록부터 마지막 블록까지 각 블록별로 읽기를 시도 하고 각 블록은 페이지별로 데이터를 구분하여 오브젝트의 경우 오브젝트를 추가하여 램상에 유지하고, 파일의 데이터 경우 해당 객체가 ObejctList에 있으면 해당 객체의 TnodeList에 파일 데이터의 위치를 추가하여 정보를 유지하게 되고 해당 객체가 ObecjtList에 없으면 객체를 등록하고 TnodeList에 파일 데이터의 위치를 추가하는 과정을 수행한다. 이러한 마운팅 작업시 램상의 정보를 구축하기 위해서 플래시 메모리 전체를 읽어야 하므로 부팅시간을 지연시키는 원인이 된다.



[그림 1] YAFFS 구조

본 논문에서는 이러한 문제점을 해결하기 위해 로그 구조 기반의 NAND 플래시 파일 시스템의 빠른 연산을 위한 관리 기법을 제안한다.

3. 제안하는 파일 시스템의 설계

플래시 메모리는 점차 용량이 증대되고 있다. YAFFS의 경우 플래시 메모리의 용량이 증가 할수록 마운트 시간이 길어지게 된다. 본 논문에서는 빠른 마운팅을 지원하는 NAND 플래시 파일 시스템의 구조를 제안한다.

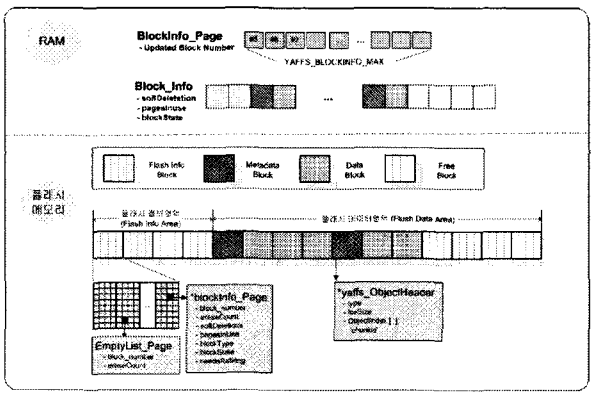
3.1. 시스템의 구조

[그림 2]와 같이 전제 플래시 메모리의 물리적인 공간을 플래시 전체 블록의 정보를 저장하고 있는 플래시 정보영역과 플래시의 메타데이터와 데이터를 저장하는 플래시 데이터영역으로 구분한다.

먼저 플래시 정보영역은 플래시 메모리의 크기에 따라 가변적으로 시스템 마운팅시 정해지게 된다. 이 영역에 유지되는 정보는 여러 개의 Allocated_BlockInfo 정보와 EmptyBlock 정보로 구성 된다. Allocate_BlockInfo 정보는 BlockInfo 객체가 한 page에 들어 갈 수 있는 최대의 개수으로 구성이 된다. BlockInfo 객체는 해당 블록의 상태, 타입 및 사용하는 페이지수 정보를 포함한다.

그리고 EmptyBlock 정보는 빈 블록에 관해서 블록 번호와 해당 블록의 지움 횟수에 관한 데이터를 기록한 것으로 파일시스템 언마운트 시에 RAM상에 유지하고 있는 BlockInfo에 구성되어 있는 블록정보 중 빈 블록의 정보를 모아서 정보를 유지하게 된다. 플래시 정보영역에서 새로운 블록을 할당 받을 시에는 기존의 플래시 정보영역에 분산 되어 적혀있는 BlockInfo에 대한 병합 작업을 수행한다.

플래시 데이터영역은 플래시 정보영역을 제외한 나머지 영역으로 구성된다. 이 영역은 메타데이터 및 데이터를 저장하는데 사용되고 메타데이터와 데이터는 블록 별로 구분하여 저장한다. 디렉토리, 하드 링크와 심볼릭 링크 생성 시, 메타 데이터가 메타 데이터블록에 저장된다. 파일 생성 시에는 데이터가 데이터블록에 저장되고 데이터 위치를 포함한 메타 데이터가 메타 데이터블록에 저장된다.

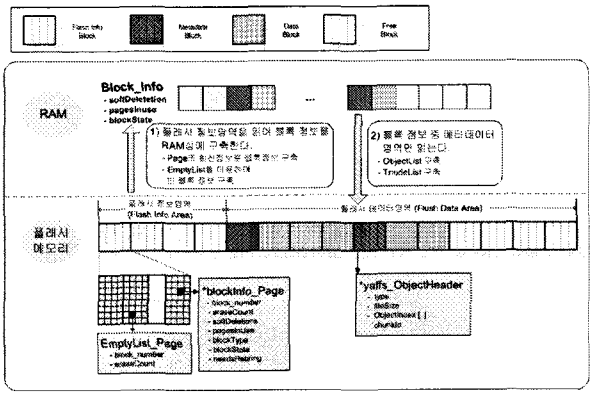


[그림 2] 제안한 플래시 파일 시스템의 구조

3.2. 빠른 마운팅 기법

제안한 파일 시스템에서 플래시 정보영역의 한 page에는 여러개의 BlockInfo 객체로 구성되어 있는 Allocated_BlockInfo가 있다. Allocated_BlockInfo가 블록정보가 한 페이지에 적히게 될 만큼의 BlockInfo 객체를 유지하게 되면 플래시 정보영역에 쓰기 작업을 수행한다. 그리고 언마운팅 시점에는 Allocated_BlockInfo로 유지하는 블록정보 쓰기 작업을 수행하고 EmptyBlock 정보를 기록한다.

[그림 3]은 제안된 파일 시스템에서의 마운팅 과정을 보여 주고 있다. 파일 시스템 마운팅 시, 최신의 빈 블록 정보를 가지고 있는 EmptyBlock을 통하여 빈 블록의 정보를 유지한다. 그리고 플래시 정보영역에서 최신 플래시 정보를 읽어온다. 읽어 온 플래시 정보는 해당 블록의 타입이 메타블록일 경우 해당 메타블록을 찾아가서 메타 정보를 읽어 RAM 상에 각 데이터 정보를 등록을 하고 데이터블록의 경우는 블록정보만 등록을 한다. 플래시 정보 영역은 순차적으로 갱신이 되기에 마운트과정에서는 최신 정보를 얻기위해 해당 블록의 마지막 페이지부터 앞 방향으로 스캔한다.



[그림 3] 제안된 구조의 마운팅 과정

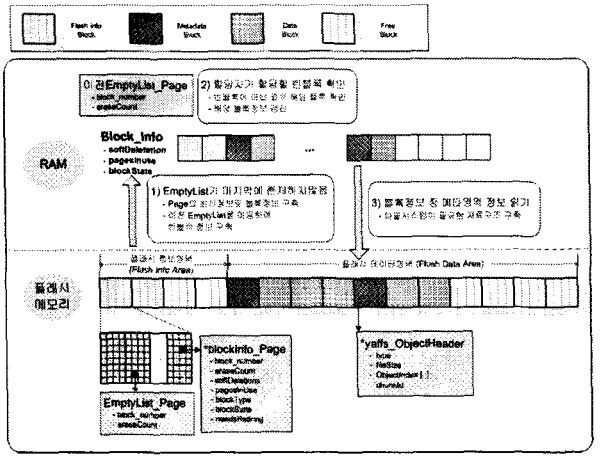
플래시 정보영역에 전체 블록정보를 적지 않고 갱신된 블록의 정보만을 적음으로서 플래시 정보영역의 사용을 최소화 할 수 있다. 또한 이전의 블록정보에 대한 기록이 남아 있어 복구 기법에도 활용 한다. 그리고 빈 블록정보를 따로 적어 지움횟수를 유지함으로서 균등화된 사용을 위한 정보를 제공할 수 있다. 또한 이러한 빈 블록정보는 할당 시에 정보를 구축하는데 용이하게 한다.

3.3. 빠른 복구 기법

제안된 파일 시스템은 플래시 메모리의 부분영역을 읽어서 마운팅을 수행한다. 그러나 이러한 방법은 시스템이 비정상 종료와 같은 문제가 생겼을 경우 블록정보인 Allocated_BlockInfo의 RAM상에 유지하고 있는 정보와 빈 블록의 정보를 가지고 있는 EmptyBlock 정보가 적히지 않게 된다.

[그림 4]는 제안한 구조에서의 복구 기법을 보여 주고 있다. 마운팅시에 플래시 정보영역에서 Emptyblock에 대한 최신정보가 없는 경우에 복구 모드로 들어가게 된다. 먼저 이전에 기록된 Allocated_BlockInfo와 EmptyBlock에 기록 된 정보를 이용하고 RAM상에 유지할 정보를 구축을 한다. 그리고 이전 기록된 EmptyBlock을 통해 구성된 할당자는 빈 블록이 아닌 블

록을 찾아내는 과정을 수행한다. 정상 종료의 경우 할당자는 빈 블록의 리스트로 유지 된다. 시스템의 비정상 종료로 인해 플래시 정보영역에 확인되지 않은 블록정보를 갱신한다. 이러한 과정을 통하여 시스템의 비정상 종료로 인한 기록되지 않은 블록정보를 부분적인 블록의 확인으로 복구가 가능하게 한다.



[그림 4] 제안한 구조의 복구 기법

4. 실험 및 성능 평가

4.1. 실험 방법

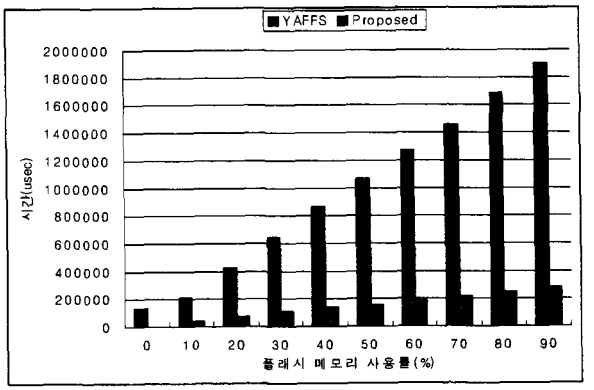
제안하는 기법들을 실제 실험을 통해 성능을 비교하고 분석하겠다.

실험을 위해 휴인스에서 개발된 PXA255-III 임베디드 보드를 사용하였다. 삼성에서 개발된 64MB NAND 플래시 메모리를 사용하였으며 YAFFS와 비교 평가한다.

4.2. 실험 결과

본 논문에서는 빠른 마운팅 시간을 제공하는 것을 목표로 하기 때문에 제안한 구조의 성능을 평가하는 방법은 플래시 메모리의 사용량에 따른 마운팅 시간을 YAFFS파일 시스템과 비교 하여 평가하겠다.

[그림 5]는 플래시 메모리 사용량에 따른 마운팅 시간을 보여준다. YAFFS의 경우, 마운팅 시점에 전체 플래시 공간을 읽어 파일 시스템을 구축한다. [그림 5]에서 플래시 메모리의 사용량에 비례하여 YAFFS의 마운팅 시간이 증가하는 것을 볼 수 있다. 제안된 기법의 경우, 플래시 정보영역을 사용하여 메타 데이터 블록만을 읽음으로 YAFFS보다 마운팅 시간이 감소 된것을 볼 수 있다.



[그림 5] 플래시 메모리 사용량에 따른 마운트 시간

5. 결 론

로그구조로 기반의 플래시 파일 시스템은 마운팅시에 필요한 데이터들을 얻기 위해 전체 플래시 메모리를 읽어야 한다. 이러한 과정은 전체 시스템의 부팅 시간을 지연시킨다.

본 논문에서는 전체 플래시 메모리 영역을 읽는 과부하를 제거하기 위하여 플래시 정보영역을 사용하여 마운팅시에 필요한 정보만을 읽도록 구조를 제안 하였고 이러한 구조에서 시스템의 비정상 종료에 관한 복구기법을 통하여 비정상 종료시에서 빠른 마운팅을 지원 할 수 있도록 하였다. 실험 결과, YAFFS에 비해 약 83%~86% 마운팅 시간이 감소하였다.

참고문헌

[1] 백승재, 최종무, "플래시 메모리 파일 시스템을 위한 순수도 기반 페이지 할당 기법에 대한 연구", 정보처리학회논문지 A 제13-A권 제5호, 2006.10

[2] Yaffs Spec,
http://www.aleph1.co.uk/taxonomy/term/31

[3] 박송화, 이태훈, 정기동, "임베디드 기기를 위한 NAND 플래시 파일 시스템의 설계", 한국컴퓨터종합학술대회, 논문집(A) 제 33권 1호, pp.151-153, 2006.