신뢰성 향상을 위한 NAND플래시 메모리에서의 저널링 파일시스템

김태훈*O 이태훈** 정기동*
부산대학교 컴퓨터공학과*
{rider7979*O withsoul**}@melon.cs.pusan.ac.kr, kdchung* @pusan.ac.kr

Design of Journaling File System on NAND Flash for advanced reliability

TaeHoon Kim*O , TaeHoon Lee**, Kidong Chung*
Dept. of Computer Engineering, Pusan National University*
Pusan National University**

요 약

플래시 메모리는 다양한 임베디드 시스템에서의 사용 빈도가 높으며, 특히 소형 정보기기의 보조기억장치로 빈번히 쓰이고 있으며, 이동형 정보기기의 경우 배터리를 사용하고 이동성이 중시되기 때문에 사용 중 전력 중단이나 외부의 충격, 환경에 많은 영향을 받게 된다. 플래시 메모리의 효율적인 오류관리를 위해서는 파일 시스템의 사용이 필요한데, 오류 발생시에 빠른 마운팅과 복구는 필수적인 요소가 된다. 본 논문에서는 기존의 JFFS[1]의 다른 NAND 플래시 파일 시스템에 적합한 구조를 설명하고, 실험을 통해 성능을 평가함으로써 NAND 플래시 메모리에 적용하기에 적합한 저널링 방법을 제안한다.

1. 서 론

현재 모바일 기기는 누구나 가지고 있을 정도로 일반화 되어 있으며, 모바일 장치에 사용되는 보조기억장치에는 플래시 메모리의 사용이 일반화 되어있다. 플래시에 대한 연산은 하드디스크와는 달리 전기적으로 일어나기 때문에, 파일의 분산에 따른 접근 시간의 차이가 없어 읽기 속도가 빠르고 비교적 자유롭다. 하지만 쓰기 연산은 복잡한 과정을 거치게 된다. 먼저 삭제 블록의 단위로 삭제 연산을 수행하게 되고 연산이 수행되면 삭제 후 원하는 곳에 쓰기 연산을 수행된다. 또한 경량이며 충격에 강하기 때문에 소규모 임베디드 시스템의 보조기억장치로 적합하다. 하지만, 모바일 기기는 휴대용 배터리를 사용하고 외부 환경에 쉽게 노출되기 때문에 갑작스러운 전력중단, 불안정한 네트워크의 연결이 발생할 수 있고, 이로 인해 데이터의 유용성이 떨어질 수 있기 때문에 발생할 수 있는 다양한 상황에 대한 효율적인 대처가 필요하다.

갑작스러운 전력중단이나 오류가 발생했을 경우, 빠른 마운팅과 빠른 오류복구 지원은 최소한의 손실을 가져오는 효율적인 방법이 될 수 있다

플래시 메모리는 치명적인 약점을 가지고 있는데, 기존에 사용되는 디스크 매체와 달리 데이터가 존재하는 영역에는 덮어쓰기가 불가능하다. 기존에 데이터가 있던 영역을 쓰기 위해서는 삭제 연산을 수행하여 플래시 메모리의 모든 비트값을 1로 바꾸는 작업이 선행되어야 한다.

그리고 플래시 메모리 셀에 데이터를 쓰기 전에 수행하는 초기화 연산의 횟수가 제한적이다. 일반적으로 그 수명은 10만번~100만번 정도로 제한되어 있다. 따라서 특정 위치에 대한 집중적인 사용은 플래시 메모리의 수명을 단축시키는 결과를 가져올 수 있다. 따라서 가비지 컬렉션(garbage collection)과 쓰기 평준화(wear leveling)과 같은 작업이 요구된다..

기존 대부분의 리눅스 파일 시스템에서는 오류가 발생할 경우 fsck를 이용하여 파일 시스템의 일관성을 검사하고 복구한다. 하지만, fsck는 수행시간이 전체 파일 시스템의 크기와 파일 및 폴더의 수와 비례하기 때문에 복구 시간이 길어지며, 그에 따른 오버헤드도 크기 때문에 빠른 마운팅이 필요한 임베디드 장치에는 적합하지 않다. 이러한 단점을 극복하기 위한 방법으로 최근 트랜잭션과 로깅 기능을 적용한 저널링 파일 시스템[2][3]이 개발되었다. 저널링 파일 시스템은 수행된 연산에 대해 로그 정보를 남기고 오류 발생시 로그를 추적하여 빠른 복구가 가능하게 된다. 이를 위해 로그 저장을 위해 디스크의 일정 영역을 저널영역으로 예약하여 사용하고 있다. JFFS2는 저널링을 플래시 파일시스템 적용한 파일시스템이지만, 마운트를 위해 전체 플래시 메모리를 스캔하기 때문에 실질적으로 빠른 복구를 구현하지 못한다. 따라서, 플래시 메모리에서는 기존의 저널링 방법을 그대로 적용할 경우 플래시 메모리가 가지는 제약 때문에 저널링을 적용하기 위해서는 플래시 메모리에 적합한 새로운 구조가 필요하다.

본 논문에서는 신뢰성 향상을 위한 빠른 복구를 지원하기 위한 플래시 파일 시스템의 구조(Reliable Flash FileSystem)를 제안하고 실험을 통해 이를 입증한다.

본 논문의 구성은 다음과 같다. 2장 관련연구에서 저널링 파일시스템에 대해 설명한다. 3장에서는 본 논문에서 제안하는 플래시 파일 시스템에 저널링을 적용하기 위한 구조를 제안하고 4장에서 실험 및 성능 평가에 대해 기술하고 5장에서 결론을 맺는다.

이 논문은 교육인적 자원부 지방연구중심대학 육성사업 (차세대물류IT 기술 연구 사업단)의 지원에 의하여 연구되었다."

2. 관련연구

2.1 저널링 파일 시스템(Journaling File System)

파일 시스템에서는 모든 데이터를 조직화하고 엑세스가 가능한 상태로 관리하게 된다. 기존의 리눅스에서는 시스템에 장애가 발생하면 메타 데이터의 일관성이 깨지게 될 수 있고, 이 경우 파일 시스템의 일관성을 회복하기 위한 방법으로 리눅스에서는 전통적으로 fsck를 사용해 왔다. 하지만, fsck는 5단계로 이루어지는 복잡한 과정을 거치며 파일 시스템과 파일의 크기에 비례해서 수행시간이 길어지는 오버헤드를 가지고 있다.

fsck의 단점을 보완하기 위해 저널링 파일시스템이 설계되었다. 저널링 파일 시스템은 log라고 불리는 파일을 디스크의 일정한 영역에 유지한다. 파일 시스템이 update되거나 record되는 연산이 수행될 때마다 그 내용을 로그로 기록하게 된다. 하나의 작업을 위해 수행되는 원자적인 모든 작업을 모아 하나의 트랜잭션으로 관리하며, 수행이 완료되면 commit 연산이 이루어지고, 실제 수행된 내용이 디스크에 쓰이게 된다. 따라서, 파일 시스템에 장애가 발생해도 로그를 추적하여 replay함으로써 빠르게 파일 시스템의 일관성을 유지하고 복구를 수행할 수 있다. 로그를 기록하기 위해 전체 파일 시스템의 크기와 비례하여 디스크의 일정영역을 저널 영역으로 사용하게 되는데, 마운트시에 저널 영역을 할당하며, 정상적으로 언마운트가 될 때 저널 영역을 삭제를 하게 된다.

현재 저널링 파일 시스템을 지원하는 파일 시스템으로는 SGI의 XFS, IBM의 JFS, Namesys의 ResierFS, Ext2에 저널링을 적용한 Ext3가 있으며, 플래시 메모리의 특성에 맞게 수정하여 적용된 JFFS 등이 있다.

3. 시스템 설계

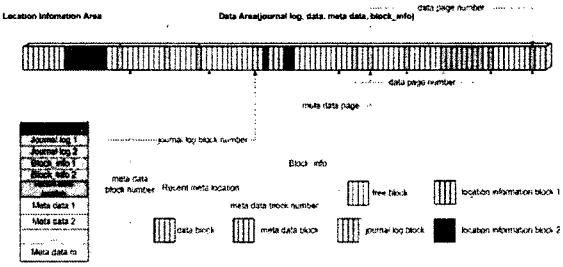
3.1 구조

기존의 저널링 파일 시스템과 같은 신뢰성과 빠른 성능을 제공하기 위해서는 빠른 마운트가 가능해야 하고, 전체적인 로그 검색이 빠른 시간안에 가능해야 한다. 하지만, 일반적으로 기존의 JFFS2나 YAFFS[4]와 같은 파일 시스템은 로그구조 기반의 파일 시스템 구조[5][6]를 가지고 있다. 따라서 로그가 플래시 메모리 전체에 흩어져 있게 된다. 따라서 이 로그를 빠른 검색이 용이 하도록 일정한 위치에 연속적으로 모아서 쓸 수 있어야 한다. 또한 빠르게 파일 시스템 마운트가 이루어질 수 있어야 한다.

따라서, 본 논문에서 제안하는 저널링을 적용하기 위한 NAND 플래시 파일 시스템은 플래시 메모리의 영역을 용도에 따라 나누어서 사용하게 된다.

위치정보(Location Information)영역, 로그(Log)영역, 메타데이터영역, 데이터영역 그리고, 블록정보(Block Information)영역으로 구성되며, 그 구성은 아래의 [그림1]와 같다.

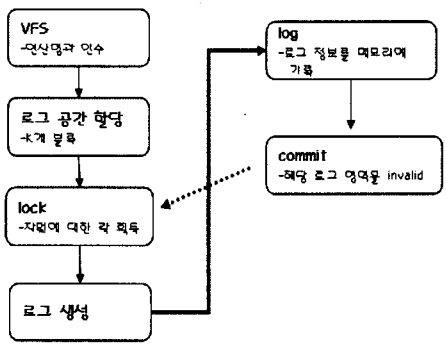
위치정보 영역은 로그 영역과 메타데이터 영역, 블록정보 영역의 위치정보를 가지고 있고, 정상적인 언마운트 여부를 확인할 수 있는 언마운트 플래그를 가지고 있다. 정상적인 언마운트 수행 후에 파일 시스템이 마운트 될 때는 위치정보 영역에서 메타데이터의 포인터를 통해 메타 데이터 영역의 정보만을 스캔해서 파일 객체 정보를 유지하고 블록 정보 영역을 통해 각 블록의 상태를 유지한다. 만약, 마운트시에 언마운트 플래그를 통해 비정상적인 언마운트가 수행됐음이 확인되면 위치정보 영역에서 로그 포인터를 통해 로그 정보를 확인하여 복구 과정을 수행하게 된다. 로그 영역의 할당은 K개의 블록 단위로 이루어진다. 여기서 K는 삭제의 단위인 세그먼트를 이루는 블록의 개수를 의미한다. K개 블록 단위로 할당하는 이유는 로그 파일이 기록될 공간을 확보해두지 않을 경우 데드락(dead lock)과 같은 문제가 발생할 수 있기 때문에, 예약된 공간과 같이 사용하기 위해서이다. 만약, K개의 블록이 모두 소비되었을 경우, 새롭게 K개의 블록을 할당 받게 되는데, 위치정보 영역에 새롭게 할당되는 K블록의 포인터가 업데이트 되고 연속된 K개의 블록을 할당받게 된다. 로그 영역은 파일 시스템이 마운트 될 때에 할당받게 되고, 언마운트 될 때는 가비지 컬렉션을 수행하여 삭제해주게 된다. K개 단위로 할당받기 때문에 플래시 메모리의 일정 영역에 쓰기가 집중되지 않아 데이터의 지역성을 방지할 수 있다. 또한 세그먼트 단위로 할당되기 때문에 로그 영역에 대한 삭제가 용이하게 된다.



[그림3] RFFS의 플래시 영역 구성

3.2 로깅 기법

RFFS는 오류가 발생했을 경우 빠른 복구를 위해 로깅기법을 사용한다. 파일 시스템에서 임의의 연산일 수행될 경우, 수행되는 연산에 대해 로그를 남김으로써 오류 발생 시에도 빠른 오류회복이 가능하다. 로깅이 수행되는 흐름을 [그림2]와 같다.



[그림4] 로깅 흐름도

[그림2]에서와 같이 파일 시스템과 로깅은 각각 독자적으로 수행되게 되고, 실제적인 파일 시스템의 연산이 수행되기 전에 로그 영역에 로깅(logging)이 완료된 후에 사용자에 의한 연산이 수행된다. 따라서, 로깅연산은 파일 시스템의 다른 연산에 간섭받지 않고 로그 데이터에 대한 무결성을 보장하게 된다.

만약, 전원오류(power fail)와 같은 문제가 발생할 경우 마운트 단계에서 로그를 참조하여 빠르게 복구를 수행하게 된다. RFFS에서의 복구 수행 과정은 다음과 같다

1) 위치 정보 영역의 unmount_flag가 0인 경우 비정상적 종료로 판단하고 복구 작업 수행한다.
2) 위치정보 영역의 log 포인터를 통해 로그 영역을 찾고, 로그 영역을 검색(scan)하여 최근에 실패한 연산을 찾는다
3) 로그의 meta필드를 통해 메타데이터 영역을 찾아 데이터를

읽어온다. 메타 데이터가 가리키는 데이터영역과의 일관성을 검사한다.

4) 모두 유효할 경우 이전 데이터는 무효(invalid)상태로 설정하고 새 데이터를 파일 시스템 구성에 포함한다. 유효하지 않거나 완료되지 않은 트랜잭션은 폐기하고 이전의 상태로 롤백(roll back)한다.

4. 실험 및 성능평가

4.1 실험환경

본 논문에서 제안한 구조의 성능을 평가하기 위한 환경은 다음 [표2]과 같다. 실험은 사용한 하드웨어는 휴인스에서 개발한 PXA255-pro3 교육용 보드에 NAND 플래시 확장보드를 사용 했다. PXA255-pro3 보드는 arm 코어를 사용하며 128MB의 메인 메모리를 가지고 있다. 확장보드의 플래시 메모리는 삼성의 64Mb 플래시 메모리가 장착되어 있으며, 16kb의 블록을 사용하며, 각 블록은 512byte단위의 페이지 32가 하나의 블록을 구성한다.

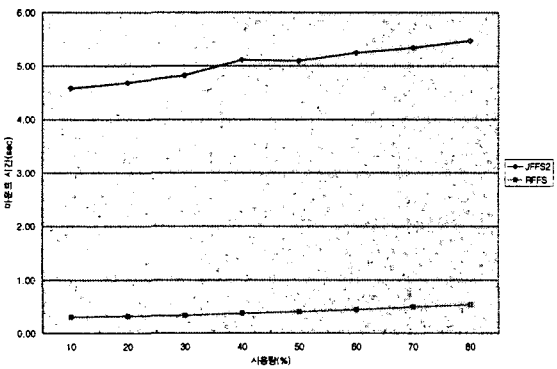
표 2 실험환경

플래시 메모리	Samsung K9F1208U0B(64Mb/chip)
실험 보드	PXA255-pro3
메인 메모리	SDRAM 128MB
Kernel	Linux 2.4.19
CPU	ARM-core 400Mhz

4.2 성능 비교

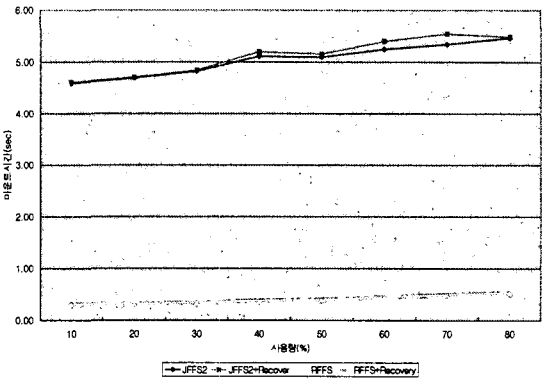
대표적인 플래시 전용 파일 시스템은 JFFS2와 본 논문에서 제안한 파일 시스템간의 마운트 속도와 인위적으로 오류가 발생하도록 하고 마운트를 수행하여 복구 속도를 측정한다. 특히, JFFS2는 저널링이 적용된 대표적인 플래시 파일 시스템으로 본 논문에서 제안한 저널링 구조와 비교가 가능하다.

첫 번째 실험은 사용량을 10%를 기준으로 단계적으로 사용량을 늘려 가면서 마운트 속도를 측정했다. 전체 64Mb의 플래시 메모리를 30M씩 나누어 각각 JFFS2와 RFFS 마운트하여 마운트 시간을 측정했다. 실험결과 JFFS2는 마운트를 위해 전체 플래시 메모리를 스캔하기 때문에 파일 시스템 구축을 위해 많은 시간을 소비하게 된다. 따라서, [그림]에서와 같이 JFFS2에 비해 RFFS가 크게 빠른것을 알 수 있다.



[그림3] JFFS2와 RFFS의 마운트 시간 비교

두 번째 실험은 사용량을 10%씩 늘려가며 오류가 발생한 상황에서의 마운트 속도를 측정했다. 전원중단과 같은 오류가 발생했을 경우, 저널링 파일 시스템은 마운트 단계에서 오류를 회복하므로 오류 발생시의 마운트 시간을 측정하면 복구에 소요되는 시간을 측정할 수 있다. 실험 결과 두 파일 시스템 모두 저널링을 적용하기 때문에 복구에 소요되는 시간은 아주 짧은 시간이 소요됨을 볼 수 있다. 하지만, JFFS2는 빠른 복구를 수행하더라도 전체적인 마운트 시간이 길기 때문에 저널링으로 인해 얻는 이점은 적다고 할 수 있다.



[그림4] JFFS2와 RFFS의 복구 시간 비교

5. 결론

본 논문에서는 NAND플래시 메모리에 효율적인 저널링 기능을 적용하기 위한 구조를 제안했다. 플래시 메모리는 일정영역을 예약하여 사용하기 불가능하기 때문에 기존의 저널링 파일 시스템에서 사용하는 저널링 기법과 달리 플래시 메모리에 적합한 구조로 변경되어 적용되어야 한다. JFFS는 저널링 플래시 메모리에 적용한 파일 시스템이다. 하지만, 빠른 복구를 통해 빠른 파일 시스템 마운트가 가능하게 하는 저널링의 특징에 부합되지 않는다. 본 논문에서 제안한 시스템에서는 로그 영역을 위한 할당 블록을 세그먼트 단위로 구성함으로써 삭제가 용이하게 하고, 빠른 마운트를 위한 구조를 제안하였다. 또한 트랜잭션이 지원을 통한 원자적 연산을 지원함으로써 신뢰성 있는 파일 시스템의 구축이 가능하도록 하였다.

참고문헌

[1] David Woodhouse,"JFFS:The Journalling Flash File System" Technical Paper of RedHat inc. Oct. 2001.

[2] Ricardo Galli "Journal File System in Linux" 2001

[3] Richard Menedetter "Journaling Filesystems for Linux"

[4] "YAFFS Flash File System" http://www.aleph1.co.uk/yaffs/

[5] Margo Seltzer, Keith Bostic "An Implementation of a Log-Structured File System for UNIX" Winter USENIX, January 25-29, 1993

[6] Mendel Rosenblum "The Design and Implementation of a Log-Structured File System"