FAT 파일 시스템의 호환성을 유지하며 성능과 안정성을 향상시키는 메타데이터 저널링 기법의 설계

(Temporary Metadata Journaling Scheme to Improve Performance and Stability of a FAT Compatible File System)

현 철 승 [†] 최 종 무 [††] 이 동 희 [†††] 노 삼 혁 [††††]
(Choulseung Hyun) (Jongmoo Choi) (Donghee Lee) (Sam H. Noh)

요 약 FAT 호환 파일 시스템은 다양한 플랫폼에서 데이터가 호환 가능하기 때문에 메모리 카드나 임베디드 시스템에서 널리 사용된다. 최근 임베디드 시스템에서는 갑작스런 전원 정지 시 복구 기법뿐만 아니라 다양한 응용의 요구를 만족시키기 위해 더 나은 파일 시스템의 성능을 요구하고 있다. 이런 요구를 수용하기 위해서는 파일 시스템의 구조에 대해 변경이 필요해 진다. 파일 시스템의 구조에 대한 변경은 데이터의 호환성에 심각한 문제를 발생시키게 된다. *메타데이터 저널링(Metadata Journaling)*은 데이터 호환성 문제를 최소화하면서 뛰어난 성능을 만족시킬 수 있는 기법이다. 이 기법을 FAT 호환 파일 시스템에 구현하여 벤치마크를 수행하였다. 벤치마크 결과는 작은 크기의 불규칙적인 메타데이터 쓰기를 저널 영역에 순차적으로 씀으로써 성능의 향상을 확인할 수 있었다. 뿐만 아니라 제안된 기법은 자연적으로 복구 기법을 제공함으로써 빠른 시간 내에 부팅이 가능하다. 그렇지만 일시적으로 FAT 호환 파일 시스템과 호환이 불가능한 지점이 존재한다. 이런 문제는 파일 시스템이 *언마운트(un-mount)*시점에 저널 영역에 쓰인 내용들을 원래의 위치로 복사함으로써 FAT 파일 시스템과 호환성을 유지시킬 수 있다.

키워드 : 메타데이터 저널링, FAT 파일 시스템, 인스턴트 부팅

Abstract The FAT (File Allocation Table) compatible file system has been widely used in mobile devices and memory cards because of its data exchangeability among numerous platforms recognizing the FAT file system. By the way, modern embedded systems have tough demands for instant power failure recovery and superior performance for multimedia applications. The key issue is how to achieve the goals of superior write performance and instant booting capability while controlling compatibility issues. To achieve the goals while controlling compatibility issues, we devised a temporary meta-data journaling scheme for a FAT compatible file system. Benchmark results of the scheme implemented in a FAT compatible file system shows that it really improves write performance of the FAT file system by converting small random write for meta-data update to a large sequential write in journaling area. Also, it provides natural way to implement the instant booting capability. Nevertheless, the file system compatibility is temporarily compromised by the scheme because it stores updated

· 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업으로 수행된 연구임(No. R0A-2007-000-20071-0)
· 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-314-D00340)
· 이 논문은 2008 한국컴퓨터종합학술대회에서 'FAT 파일 시스템의 호환성을 유지하며 성능과 안정성을 향상시키는 메타데이터 저널링 기법의 설계'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울시립대학교 컴퓨터통계학과
cshyun@uos.ac.kr
†† 종신회원 : 단국대학교 컴퓨터학부 교수
choijm@dankook.ac.kr
††† 정 회 원 : 서울시립대학교 컴퓨터과학부 교수
dhl_express@hanmail.net
(Corresponding author임)
†††† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
samhnoh@hongik.ac.kr

논문접수 : 2008년 8월 25일
심사완료 : 2009년 1월 23일

정보과학회논문지 : 시스템 및 이론 제36권 제3호(2009.6)

meta-data in the temporary journaling area rather than to their original locations. However, the compatibility can be fully recovered at any time by journal-flushing that copies meta-data in journaling area to their original locations. Generally, the journal-flushing is done before un-mounting a memory card so that it can be used in other mobile devices which recognized FAT file system but not the temporary meta-data journaling scheme.

Key words : Metadata Journaling, FAT File System, Instant Booting

1. 서 론

임베디드 시스템에서 플래시 메모리 저장 장치는 비휘발성, 낮은 전력 소비 및 충격에 강한 장점 때문에 널리 사용되고 있다. 현재 휴대용 기기에서 Compact Flash, MMC, SD 같은 플래시 메모리 카드들이 널리 사용되고 있다. 이러한 메모리 카드 내에는 플래시 메모리를 블록 장치(예, 디스크)처럼 동작하게 하는 FTL (Flash Translation Layer)[1,2]이 존재한다. 이 때문에 기존의 FAT 호환 파일 시스템이나 Ext2, Ext3 파일 시스템이 메모리 카드상에서 동작할 수 있다.

물론, JFFS2[3]나 YAFFS[4]같은 플래시 메모리 전용 파일 시스템이 제안되었지만 PC와 디바이스 장치의 데이터 호환성 때문에 FAT 호환 파일 시스템[5]이 더 많이 사용되고 있으며 임베디드 시스템에서 사실상의 표준으로 인식되고 있다.

그렇지만, 플래시 메모리 저장 장치상에서 동작하는 FAT 파일 시스템은 메타데이터에 대한 불규칙적인 쓰기로 인해 성능이 떨어지는 단점이 있다. 이런 단점이 있지만 데이터 호환이 가능하다는 커다란 장점 때문에 MP3 플레이어나 휴대폰 같은 임베디드 시스템에 널리 사용되고 있다. 최근 복잡하고 다양한 응용들을 만족시키기 위해 더 나은 성능의 FAT 호환 파일 시스템이 요구되고 있다. 뿐만 아니라 갑작스런 전원 정지에 따른 복구 기법도 요구되고 있다.

이처럼 임베디드 시스템에서 보다 좋은 쓰기 성능과 복구 기법이 요구되고 있지만 이런 요구를 만족시키기 위한 설계들은 파일 시스템의 호환성 문제를 일으킬 수 있다. 제안된 메타데이터 저널링 기법은 설계의 가장 중요한 이슈인 데이터 호환성 문제를 적절히 제어하면서 좋은 성능을 얻을 수 있는 기법이다. 이 기법은 플래시 메모리 저장 장치에서 메타데이터 변경에 대한 불규칙적인 쓰기 동작을 저널 영역에 순차 쓰기로 변경함으로써 성능 이득을 얻을 수 있다. 뿐만 아니라 이러한 순차 쓰기를 통해 전원 정지 시에도 저널 영역에서 마지막으로 쓰여진 위치를 찾음으로써 효과적인 복구가 가능해진다. 그렇지만 파일의 메타데이터를 원래의 위치가 아닌 저널 영역에 쓰는 것은 일시적으로 호환성 문제를 일으키기 된다. 이런 문제는 저널 영역에 쓰여진 메타데이터를 원래의 위치로 쓰는 저널 *플러싱(Journal-Flushing)*을 수행함으로써 원래의 호환성을 유지할 수 있다. 일반적으로 PC에서 메모리 카드를 제거하기 전에 저널 플러싱을 수행함으로써 제안된 기법이 적용된 FAT 파일 시스템과 원래의 FAT 호환 파일 시스템간의 호환성을 제공할 수 있게 된다.

2. 관련 연구

NAND 플래시 메모리는 휴대용 기기에서 저장 장치로 널리 사용되고 있다. 그 결과로 Log-Structured 파일 시스템[6]의 변종인 JFFS2나 YAFFS같은 플래시 메모리 전용 파일 시스템이 제안되었다. 이 파일 시스템들은 플래시 메모리 저장 장치를 위해 설계되었지만 SD, MMC나 CompactFlash 카드상에는 동작하지 않는다. 이런 카드들의 내부에는 플래시 메모리 칩을 블록 장치처럼 동작하게 하는 FTL 소프트웨어가 있다. 그렇기 때문에 블록 장치를 기반으로 하는 FAT 파일 시스템과 같은 일반적인 파일 시스템들이 메모리 카드상에서 동작할 수 있다.

FAT 호환 파일 시스템은 PC와 휴대용 기기간에 데이터 호환성 때문에 임베디드 시스템에서 널리 사용되어 사실상의 표준으로 자리잡았다. 그러나 이러한 호환성 문제는 저널 기법의 도입을 어렵게 한다. 일반적인 저널 기법은 파일 시스템의 구조에 변경을 가해야 한다. 이런 변경은 호환성에 문제를 일으키게 되기 때문에 적용하기가 용이하지 않다.

현재, FAT 호환 파일 시스템의 안정성 향상과 일관성 복구를 위해 제안된 기법은 [7-9]가 있다. KFAT 파일 시스템[9]은 파일 연산에 대한 메타데이터 변경을 로그 파일에 기록하고 이후에 기록된 정보를 바탕으로 파일 시스템의 일관성을 복구하는 기법을 사용하였다.

FATTY[8] 파일 시스템은 메타데이터의 변경을 동기적으로 저장 장치에 쓰는 동작과 하나의 파일 시스템 연산에 대해 저장 장치에 반영되는 순서를 유지하는 Soft-updates[10] 기법을 사용한다. 이렇게 변경된 파일 시스템의 내용을 순서를 유지하면서 저장 장치에 동기적으로 씀으로써 파일 시스템의 일관성 문제를 해결할 수 있다. 그러나 동기적인 쓰기는 파일 시스템의 성능을 하락시키는 원인이 됨으로 [8]에서는 디스크 캐시 내에서 변경된 연산을 조합하여 한 번에 쓰는 *지연된 순차 쓰기(DSW: Delayed Sequential Write)*를 활용하여

성능을 향상 시켰다.

마이크로소프트사에서 제안된 TFAT(Transactional FAT)[7]은 파일 시스템에서 현재 수행되고 있는 연산을 처리하기 위해 FAT1을 사용하고 FAT0는 변경되기 이전의 안정적인 상태를 저장하는데 사용한다. FAT1에 변경되는 내용들을 반영하고 변경이 완료되면 FAT0에 FAT1의 내용을 복사하는 방식으로 동작한다. 이런 기법은 안정성을 향상시킬 수 있지만 약간의 성능 저하가 발생하는 문제점을 갖고 있다.

저널 기법은 파일 시스템의 안정성뿐만 아니라 메타데이터 연산이 많은 작업부하에서 성능을 높이기 위해 널리 사용되었다. 저널 영역에 어떤 데이터를 저장해야 하는지는 파일 시스템에서 제공하는 저널 기법의 수준과 구현 방법에 따라 다르다. 일반적인 저널 파일 시스템은 메타데이터와 파일 데이터에 대한 변경을 저장 장치에 쓰기 전에 수행할 연산의 종류와 그 연산에 관련된 메타데이터를 저널 영역에 쓴다. 후에 갑작스런 전원 정지로 인해 파일 시스템의 일관성이 유지되지 않은 상태에서도 재부팅되는 시점에 저널 영역에 저장된 데이터를 기반으로 완료되지 않은 연산을 다시 수행하거나 취소함으로써 일관성을 복구할 수 있다. 이렇게 파일 시스템이 저장된 데이터에 대한 변경이나 새로 쓰여질 데이터에 대해 저널을 수행함으로써 파일 시스템의 안정성이 증가되며 가끔은 메타데이터 연산에 대해 성능향상을 얻을 수 있게 된다. 제안하는 기법은 기존의 저널 기법과 유사하지만 현재 존재하는 파일 시스템에서 사용할 수 있는 기법이며 이 기법을 사용하는 파일 시스템과 원래의 파일 시스템간의 호환성 문제를 제어하는데 초점을 맞추고 있다.

3. 메타데이터 저널링 기법의 설계

3.1 설계 이슈

임베디드 시스템의 특수한 환경 때문에 갑작스런 전원 정지에 따른 데이터의 비일관적인 상태를 빠른 시간에 복구할 수 있는 기법이 요구된다. 또한 다양한 멀티미디어 응용들을 안정적으로 수행하기 위해 보다 높은 성능의 파일 시스템을 필요로 하고 있다. 임베디드 시스템에서의 저장 장치는 전력 소비가 적고 충격에 강한 특징을 가지고 있는 플래시 메모리가 많이 사용된다. 이런 플래시 메모리는 읽기 성능이 쓰기 성능보다 우수하기 때문에 쓰기 성능을 높이는 것이 주요 요구 사항이 되고 있다.

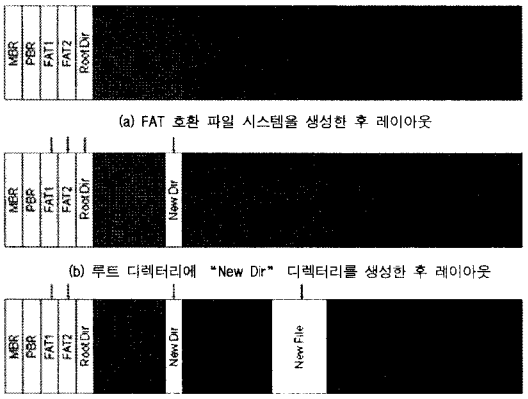
파일 시스템에서 성능을 향상시키기 위해서는 기존 시스템에 대한 변경이 필요하지만 파일 시스템의 호환성 문제는 계속 요구되고 있다. 이렇게 현재 발생하고 있는 요구들을 수용하기 위한 설계는 호환성에 대한 제어를 제공하면서 빠른 복구와 쓰기 성능을 향상시키는 데 주목해야 한다. 즉, 호환성 문제와 새로운 기법을 어떻게 조화시킬 것인가가 설계에서 가장 중요한 점이다.

플래시 메모리나 하드 디스크에서 순차 쓰기가 반복 쓰기나 불규칙적인 쓰기보다 우수한 성능을 보이는 것은 잘 알려진 사실이다. 이를 기반으로 파일 시스템의 쓰기 요청을 순차적으로 변환함으로써 성능을 향상시킬 수 있다. 그러나 Log-Structured 파일 시스템[6]의 동작 방식처럼 메타데이터와 파일 데이터에 대한 쓰기를 순차적으로 한다면 파일 시스템의 구조를 다시 설계해야 하며 호환성도 잃게 된다. 또한 파일 데이터에 대한 저널링은 파일 시스템의 성능을 저하시키는 요인으로 작용할 수 있다[11]. 그래서 파일 시스템의 변경과 호환성 위반을 최소화하기 위해 메타데이터의 쓰기 요청만 저널 영역에 순차적으로 변경하였다. 일반적인 FAT 호환 파일 시스템에서 클러스터 할당 정책은 순차적으로 수행된다. 플래시 메모리를 저장 장치로 사용하는 파일 시스템에서 순차적인 할당 정책을 사용하면 메타데이터에 대한 반복적인 쓰기와 불규칙적인 쓰기가 성능 저하의 주요 원인이 된다. 따라서 메타데이터에 대한 쓰기를 순차적으로 변경함으로써 성능향상을 얻을 수 있게 된다. 뿐만 아니라 메타데이터의 순차 쓰기는 갑작스런 전원 정지나 예상치 못한 오류가 발생하는 상황에서도 자연스럽게 데이터의 일관성을 제공해 줄 수 있게 된다.

3.2 메타데이터 저널링의 동작 방식

그림 1은 원래의 FAT 호환 파일 시스템의 레이아웃이다. 디스크의 시작 섹터에 *마스터 부트 레코드(MBR)*이 존재한다. 이 레코드는 최대 네 개까지의 파티션을 지원해 줄 수 있으며 파티션의 위치와 크기를 테이블로 저장하고 있다. 각 파티션의 시작 섹터는 *파티션 부트레코드(PBR)*, FAT1, FAT2가 존재하고 그 다음으로 루트 디렉터리가 있다. 원래의 FAT 파일 시스템에서는 파일 또는 디렉터리가 생성/확장될 때 새로운 클러스터를 할당하게 된다. 클러스터 할당으로 인해 적절한 위치의 FAT1과 FAT2가 변경된다. 또한 부모 디렉터리와 관련된 디렉터리 엔트리가 파일이나 디렉터리의 변경된 상태를 반영하기 위해 그림 1의 (b)와 (c)에서 화살표로 표시된 것처럼 바뀌게 된다. 이런 메타데이터는 파일 시스템이 관리하는 영역 전체에 걸쳐 퍼져 있으며, 이에 대한 변경은 파일 시스템의 성능에 있어 주요 장애가 된다. 게다가 메타데이터의 변경은 저장 장치에 변경된 순서를 유지하면서 쓰여야만 파일 시스템에 문제가 발생한 후 재부팅시에 *fsck*같은 일관성 검사 툴을 사용하여 일관성을 복구할 수 있다.

이렇게 파일 시스템 영역 전체에 흩어져 있는 메타데이터에 대해 불규칙적이고 동기적으로 발생하는 쓰기를 제거하기 위해 메타데이터 저널링 기법을 사용할 수 있

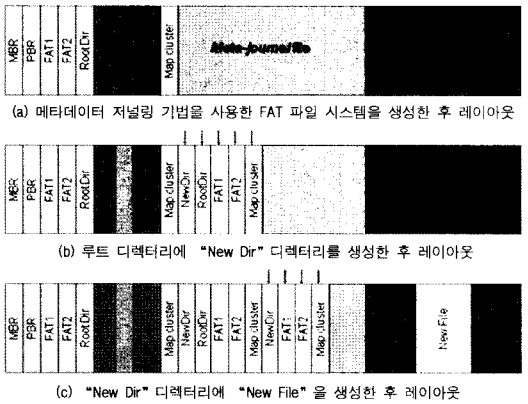


(a) FAT 호환 파일 시스템을 생성한 후 레이아웃

(b) 루트 디렉터리에 "New Dir" 디렉터리를 생성한 후 레이아웃

(c) "New Dir" 내에 "New File" 을 생성한 후 레이아웃

그림 1 FAT 호환 파일 시스템의 레이아웃



(a) 메타데이터 저널링 기법을 사용한 FAT 파일 시스템을 생성한 후 레이아웃

(b) 루트 디렉터리에 "New Dir" 디렉터리를 생성한 후 레이아웃

(c) "New Dir" 디렉터리에 "New File" 을 생성한 후 레이아웃

그림 2 메타데이터 저널링 기법을 적용한 파일 시스템의 레이아웃

다. 이 기법은 변경된 메타데이터를 포함하는 클러스터들을 저널 영역에 순차적으로 쓰기를 수행한다. 그림 2의 (a)는 파일 시스템을 생성한 후 메타데이터 저널링을 위한 저널 파일을 생성한 모습을 보여주고 있다. 이 저널 파일은 FAT 호환 파일 시스템에서 일반 파일로 다룰 수 있기 때문에 호환성에 문제가 되지 않는다. 메타데이터 저널링 기법이 적용된 파일 시스템에서 디렉터리나 FAT 엔트리 같은 메타데이터의 변경이 발생하면, 이 변경과 연관된 디스크 캐시내의 클러스터가 변경된다. 그리고 디스크 캐시 내에 변경된 모든 클러스터들은 원래의 위치가 아닌 저널 파일에 순차적으로 쓰여지게 된다.

메타데이터의 클러스터가 원래의 위치가 아닌 저널 파일에 임시로 쓰여지기 때문에 원래의 위치와 저널 파일 내에서 위치를 연결시켜야 원래의 위치로 환원시킬 수 있다. 저널 파일을 구성하고 있는 클러스터는 맵 클러스터(*Map Cluster*)와 메타데이터 클러스터로 나뉜다. 그림 3은 파일 시스템의 메타데이터의 위치와 저널 파일 내의 위치를 연결시키기 위한 맵 클러스터의 구조를 보여주고 있다. 맵 클러스터의 크기는 FAT 파일 시스템이 생성될 때 제공되는 클러스터의 크기와 동일하다. 또한 *헤드(Head)*, *맵핑 엔트리(Mapping Entry)*, *테일(Tail)*로 구성된다. 헤드와 테일 영역은 저널 파일 내에서 메타데이터 클러스터와 맵 클러스터를 구분할 수 있는 패턴 정보를 저장하기 위한 식별자, 가장 최신의 맵 클러스터를 찾기 위한 시퀀스(Sequence) 번호, 그리고 헤드와 테일 영역의 유효성을 검사하기 위해 체크섬(Checksum) 필드로 구성된다. 맵핑 엔트리 영역에서 각 엔트리는 저널 파일을 구성하는 클러스터 번호가 증가하는 순으로 메타데이터 클러스터의 원래 위치를 저장하고 있다. 예를 들어 저널 파일을 위해 클러스터가 64번부터 127번까지 할당되었다면 첫 번째 엔트리가 64번 클러스터의 원래 위치를, 그 다음 엔트리가 65번 클러스터의 원래 위치를 저장하게 된다.

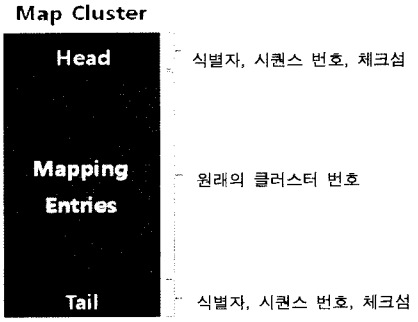


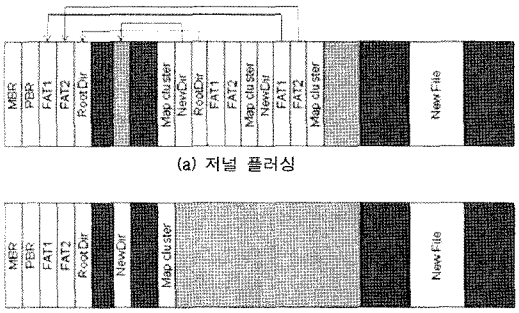
그림 3 맵 클러스터의 구조

저널 파일에 대한 연산의 예로 메타데이터 클러스터 270번과 271번이 저널 파일 내에 두 번째와 세 번째 클러스터에 쓰여졌다면 이를 원래의 위치와 연결시키기 위해 맵 클러스터를 구성하고 있는 맵핑 엔트리 영역내의 두 번째 엔트리에는 270, 세 번째 엔트리에는 271이 각각 저장된다. 또한, 맵 클러스터 내의 헤드와 테일 영역에 시퀀스 번호 필드의 값을 1만큼 증가시키고 저널 파일 내의 네 번째 클러스터에 쓰게 된다. 그 후 270, 271번 클러스터에 대한 모든 요청은 저널 파일 내의 두 번째와 세 번째 클러스터로 변경되게 된다. 시간이 지난 후 기존의 270번과 또 다른 272번 클러스터가 변경되면 저널 파일의 다섯 번째와 여섯 번째 클러스터에 각각 쓰여지게 된다. 뿐만 아니라 위에 설명한대로 맵 클러스터가 변경되고 일곱 번째 클러스터에 쓰여지면 모든 270, 272번에 대한 요청은 저널 파일 내의 그 위치로 변경된다.

파일 시스템이 *마운트(mount)*되는 시점에 저널 파일 내의 모든 맵 클러스터를 검색하고 그 중에 가장 큰 시퀀스 번호를 갖는 맵 클러스터를 찾는다. 이렇게 마지막으로 쓰여진 맵 클러스터를 찾은 후 맵핑 엔트리 영역의 정보를 바탕으로 모든 요청이 저널 파일 내의 클러스터로 변경된다. 또한 순차적으로 메타데이터 클러스터가 쓰여지고 마지막에 맵 클러스터가 쓰여짐으로써 자연스럽게 복구 기법이 제공되게 된다. 파일 시스템이 디스크 캐시에 존재하는 여러 개의 메타데이터와 파일 클러스터들을 변경하는 연산을 수행하였다고 가정해 보자. 연산이 완료되거나 추가의 다른 요청을 처리하기 위해 디스크 캐시에 가용 공간이 없으면 파일 시스템은 디스크 캐시의 변경된 내용을 저장 장치에 써야 한다. 이 경우 제안된 기법이 구현된 파일 시스템에서 파일 데이터 클러스터는 원래의 위치에 쓰여지고 메타데이터 클러스터는 저널 파일에 기록된다. 마지막으로 메타데이터 클러스터의 원래 위치와 저널 파일 내의 위치를 연결한 후 맵 클러스터를 쓴다. 이 시점에서 시스템에 문제가 발생하면 다시 시스템이 시작되어도 맵 클러스터가 저널 파일 내에 쓰여졌기 때문에 아무런 문제도 발생하지 않는다. 이처럼 메타데이터에 대한 변경은 맵 클러스터가 쓰여짐과 동시에 완료되게 된다. 반면, 맵 클러스터가 쓰여지기 전에 시스템 문제로 인해 다시 부팅되면 이전의 변경은 무효화된다. 이런 형태의 저널링과 맵핑 동작은 파일 시스템 구조에 대한 변경을 트랜잭션화 할 수 있게 하여 시스템 붕괴 후 빠른 부팅을 가능하게 한다.

위에 언급하였지만 저널 파일 내에는 메타데이터 클러스터와 다양한 시퀀스 번호를 갖는 맵 클러스터들이 혼재한다. 이 경우 파일 시스템은 맵 클러스터를 찾기 위해 메타데이터 클러스터와 맵 클러스터를 구분할 수 있어야 한다. 이를 위해 맵 클러스터는 헤드와 테일 영역에 식별자를 포함하고 있다. FAT 호환 파일 시스템에서 메타데이터 클러스터는 디렉터리 엔트리와 FAT을 저장하고 있는 클러스터이다. 이를 이용해 유효한 FAT과 디렉터리 엔트리에 존재할 수 없는 패턴을 식별자 필드에 저장함으로써 저널 파일 내에서 맵 클러스터를 찾을 수 있다. 예를 들어, FAT 파일 시스템 표준에 따르면 유효한 FAT 엔트리는 클러스터 번호가 1일 수 없다. 따라서 FAT12인 경우 0x001, FAT16인 경우 0x0001, FAT32인 경우 0x00000001은 FAT 엔트리에 나타날 수 없다. 또한 체크섬 필드는 부분적으로 쓰여진 맵 클러스터를 구분하기 위해 사용된다.

저널 파일 내에 모든 클러스터가 사용되면 메타데이터 변경을 저널 파일에 저장하기 위해 *저널 플러싱(Journal-Flushing)*이 이루어진다. 이 동작은 LFS에서의 동작과 유사하지만 구현의 용이성을 위해 간결하게 설계하였다. 첫 단계로 맵 클러스터에서 맵을 저장하고 있는 맵핑 엔트리 영역의 정보를 참고하여 유효한 클러스터들을 원래의 위치로 복사한다. 복사가 끝나면 맵핑 엔트리 영역의 각 엔트리는 0xFFFFFFFF로 채워지고 시퀀스 번호를 1만큼 증가시킨다.



(a) 저널 플러싱

(b) 저널 플러싱후 파일 시스템 레이아웃

그림 4 저널 플러싱 동작

그 후 저널 파일의 첫 번째 클러스터 위치에 맵 클러스터를 씀으로써 저널 플러싱 동작이 완료된다. 이런 쓰레기 수집동작이 완료된 후 위에 설명한 대로 파일 시스템이 동작하게 된다. 비록 현재 구현된 저널 플러싱 기법이 간단하지만 구현된 시스템에서 우수한 성능을 보여주었으며, 후에 좀더 정교하게 설계된 기법들도 효율적으로 구현할 수 있도록 하였다.

메타데이터 저널링 기법은 변경된 메타데이터 클러스터가 원래의 위치가 아닌 저널 파일 내에 존재할 경우 일시적으로 FAT 호환 파일 시스템과 호환성을 잃게 된다. 그렇지만 원하는 시점에 저널 플러싱을 수행함으로써 FAT 호환 파일 시스템과의 호환성을 완벽히 복구할 수 있다. 파일 시스템의 상위 계층의 응용이나 운영 체제에서 저널 플러싱을 요청할 수 있지만, 대부분의 경우 *언마운트(un-mount)*되는 시점에 호출되어 다른 시스템에서 FAT 호환 파일 시스템으로 저장된 데이터를 인식할 수 있게 된다.

4. 구현 및 실험 결과

메타데이터 저널링 기법을 MP3 플레이어나 네비게이션 시스템에서 사용되고 있는 FAT 호환 파일 시스템[12]에 구현하였다. 구현된 파일 시스템은 Windows CE나 Linux 운영 체제에서 동작한다. 이렇게 다양한 시스템에서 동작할 수 있도록 파일 시스템을 구현하였기 때문에 현재 파일 시스템은 자체적으로 캐시를 관리하고 있다. 그래서 Linux 운영 체제인 경우 디스크 캐시를 일부만 사용하고 있기 때문에 성능이 다소 떨어진다.

원래의 FAT 호환 파일 시스템[12]은 파일 시스템 복구를 위해 Intent-logging 기법을 사용하고 있다. 이 기

법은 파일 시스템 구조를 변경하기 전에 연산의 종류와 메타데이터를 로그 파일에 저장한다. 연산이 성공적으로 완료되면 그 연산을 위해 로그 파일에 쓰였던 내용은 삭제된다. 그렇지 않고 갑작스런 전원 정지 같은 시스템 붕괴가 일어나면 재부팅될 때 로그 파일 내의 로그 정보를 바탕으로 연산을 완료시키거나 취소시킴으로써 일관성을 복구한다. 실험 결과에서 아무런 기법이 적용되지 않은 파일 시스템을 "standard", intent-logging 기법이 적용된 파일 시스템을 "intent-logging", 메타데이터 저널링 기법이 적용된 파일 시스템을 "metadata-journaling"이라 표현하였다.

각 기법이 적용된 파일 시스템의 성능을 평가하기 위해 200MHz의 EP9315 평가보드를 사용하였고 벤치마크는 Postmark를 사용하였다. EP9315 평가보드는 64MB의 RAM, 16MB의 NOR 플래시 메모리가 탑재되어 있다. Postmark는 트랜잭션의 형태나 회수, 생성할 파일과 디렉터리 개수, 파일 크기 같은 파라미터들을 변경함으로써 다양한 작업 부하를 생성할 수 있다. 실험에 사용된 파라미터는 표 1과 같고 작업 부하는 메타데이터 연산이 주로 일어날 수 있도록 하였다. 또한 실험에 사용한 플래시 메모리 저장 장치는 SD, CompactFlash, USB드라이브이며, 표 2의 내용과 같다.

그림 5는 각 저장 장치에서 벤치마크의 수행 결과이다. 여기서 메타데이터 저널링 기법이 적용된FAT 파일 시스템의 벤치마크 수행시간은 저널 플러싱 동작이 포함된 시간이다. 그림 5의 결과는 메타데이터 저널링 기법이 적용된 파일 시스템에서 FAT 호환 파일 시스템이나 intent-logging 기법이 적용된 파일 시스템보다 월등한 성능 향상이 있음을 보여주고 있다. 이런 결과는 플래시 메모리에서 순차 쓰기가 불규칙적인 쓰기에 비해 좋은 성능을 내기 때문에 이를 활용한 메타데이터 저널링 기법이 파일 시스템 안정성뿐만 아니라 성능 향상에도 기여할 수 있음을 보여 준다. 또한 기존에 구현된 FAT 호환 파일 시스템이나 intent-logging 기법이 적

표 1 메타데이터 작업부하를 위한 Postmark 파라미터

파라미터	작업 부하
파일 생성 개수	1000
디렉터리 생성 개수	1000
파일 크기	0.5 - 20KB
트랜잭션 회수	1000

표 2 실험에 사용된 저장 장치

종류	제품 번호	용량
SD	PLEOMAX SMSD600S	2G
USB	SUM-M4GUB	4G
CompactFlash	PLEOMAX SCF100A	4G

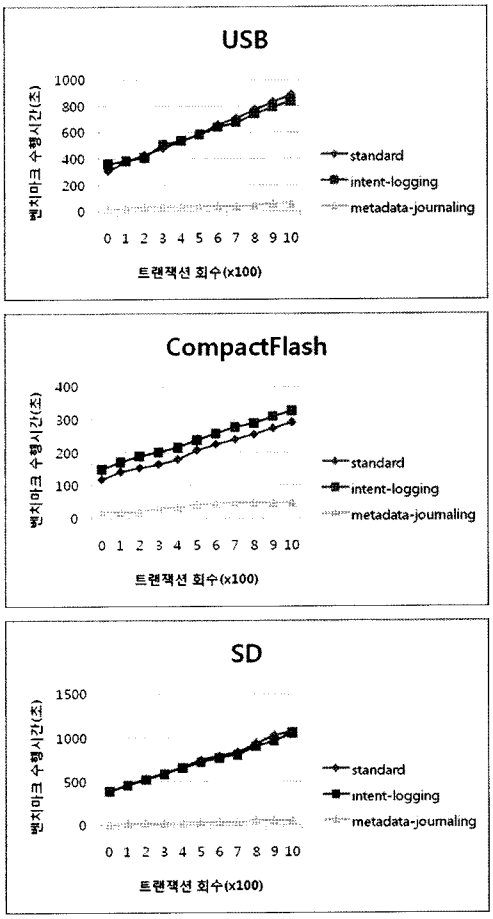


그림 5 SD, CompactFlash, USB 저장 장치에서 벤치마크 수행 결과

용된 파일 시스템에서의 메타데이터 갱신은 파일 시스템의 안정성 향상을 위해 동기적인 쓰기가 반드시 필요하다. 그렇지만 메타데이터 저널링 기법인 경우 메타데이터 연산에서 맵 클러스터가 체크포인트 역할을 수행함으로써 지연된 쓰기가 가능해진다. 이런 지연 쓰기는 메타데이터 저널링 기법이 적용된 파일 시스템에서 성능 향상에 많은 도움을 준다.

그림 6은 벤치마크가 수행되는 동안 저장 장치로의 요청에 대한 트레이스이다. 이 트레이스를 살펴보면 FAT 호환 파일 시스템이나 intent-logging 기법이 적용된 파일 시스템인 경우 FAT 영역과 메타데이터 영역에 굉장히 많은 불규칙적인 쓰기가 발생함을 확인할 수 있다. 반면 메타데이터 저널링 기법이 적용된 FAT 파일 시스템에서는 블록번호가 22000부터 24000까지(그래프에서 사각형으로 표시된 "가" 부분)의 저널 영역에 순차적으로 쓰기가 발생함을 확인할 수 있다. 순차 쓰기가 종료

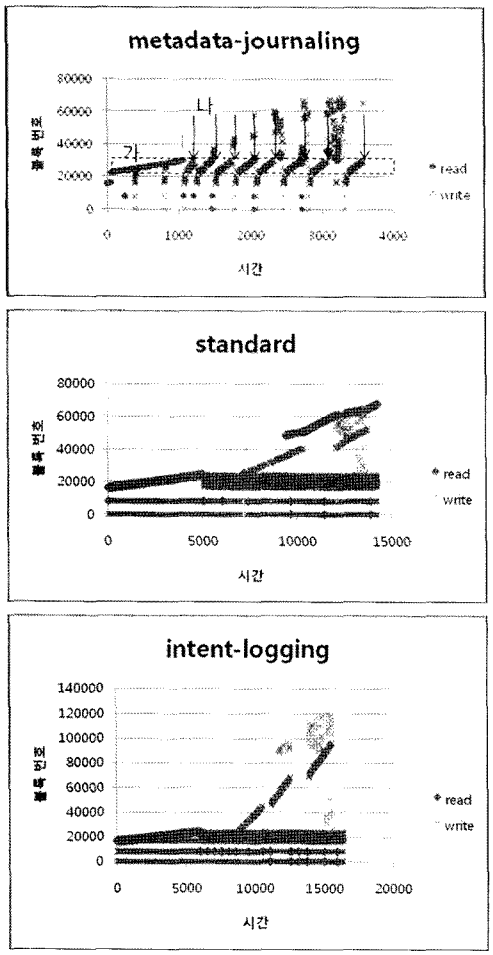


그림 6 벤치마크 수행중 저장 장치로 요청되는 파일 시스템 트레이스

되는 부분에서는 저널 플러싱이 동작한다(그래프에서 화살표로 표시된 "나" 부분). 이에 따라 블록번호가 20000번지 이하에 위치한 FAT 영역과 루트디렉터리에 대한 쓰기 연산이 발생하고 있으며 블록번호가 30000번지 이상에 위치한 디렉터리 블록에 대해서도 저널 플러싱에 의한 쓰기 연산이 발생하고 있다. 저널 플러싱 동작이 불규칙적인 쓰기를 발생시키고 있지만 기존의 기법이 적용된 FAT 파일 시스템에 비해 불규칙적인 쓰기 동작이 순차적으로 변경되고 있고 동기적이 쓰기가 획기적으로 감소했음을 확인할 수 있다. 이로 인해 메타데이터 저널링 기법이 적용된 FAT 파일 시스템이 기존의 파일 시스템보다 월등한 성능을 보인다.

5. 결론 및 향후 연구

최근의 임베디드 시스템에서는 다양한 응용을 수행하기 위해 사실상의 표준인 FAT 파일 시스템의 성능 개선 요구가 많다. 뿐만 아니라 임베디드 시스템 특성상 갑작스런 전원 정지시에 빠른 시간내에 일관성을 복구할 수 있는 기법도 요구되고 있다. 본 논문에서는 이러한 요구를 만족할 수 있는 메타데이터 저널링 기법을 제안하였다. 이 기법을 실제 시스템에 구현하였고, 벤치마크를 수행한 결과 표준 FAT 파일 시스템이나 Intent-logging 기법이 적용된 FAT 파일 시스템에 비해 월등한 성능 향상이 있음을 확인할 수 있었다. 또한 맵 클러스터의 쓰기를 통해 여러 개의 메타데이터 변경을 트랜잭션화하여 갑작스런 전원 정지에도 효과적으로 일관성을 복구가 가능하다. 그리고 저널 파일 내의 메타데이터 클러스터들을 원래 위치로 환원하여 기존의 FAT 파일 시스템과 완벽하게 호환될 수 있도록 저널 플러싱 연산을 둠으로써 호환성을 제어할 수 있도록 하였다.

이 기법은 저널 플러싱이 적절히 동작하지 않으면 기존 파일 시스템과의 호환성을 잃을 수 있다. 또한 저널 플러싱이 완전히 종료되지 않고 표준 FAT 파일 시스템으로 마운트할 경우 데이터가 손실될 위험이 있다. 이런 문제를 해결하기 위해 향후 정교한 저널 플러싱 동작에 대한 연구가 필요하다.

참 고 문 헌

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

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

[3] D. WoodHouse, "JFFS: The Journaling Flash File System," in *Ottawa Linux Symposium 2001*, 2001.

[4] "YAFFS (Yet Another Flash File System) Version 0.3," 2002.

[5] "FAT32 File System Specification," Microsoft Coporation, 2000.

[6] M. Rosenblum and J. K. Ousterhout, "The Design and Implementation of a Log-Structured File System," *ACM Transactions on Computer Systems*, Vol. 10, pp. 26-52, 1992.

[7] "Transaction-Safe FAT File System," Microsoft Co., 2008.

[8] L. Alei, L. Kejia, L. Xiaoyong, and G. Haibing, "FATTY: A Reliable FAT File System," in *10th Euromicro Conference on Digital System Design Architecture, Methods and Tools(DSN 2007)*: IEEE, 2007, pp. 390-395.

[9] M. S. Kwon, S. H. Bae, S. S. Jung, D. Y. Seo, and C. K. Kim, "KFAT: Log-Based Transactional FAT Filesystem for Embedded Mobile Systems," in *Information & Communication Technology Symposium(ICTS)*, 2005.

[10] M. K. McKusick and G. R. Ganger, "Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem," in *1999 USENIX Annual Technical Conference* Monterey, California, USA: USENIX Association, 1999.
[11] Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, "Analysis and Evolution of Journaling File Systems," in *Proceedings of the USENIX 2005 Annual Technical Conference*, pp. 105-120, 2005.
[12] "ZFS," Zeen Information Technologies, Inc.

현 철 승
2001년 제주대학교 통신컴퓨터공학부(학사). 2007년 서울시립대학교 컴퓨터통계학과(석사). 2007년~현재 서울시립대학교 컴퓨터통계학과 박사과정. 관심분야는 운영체제, 내장형시스템, 저장장치

최 종 무
1993년 서울대학교 해양학과(학사). 1995년 서울대학교 컴퓨터 공학과(석사). 2001년 서울대학교 컴퓨터 공학과(박사). 2001년~2003년 유비쿼스 주식회사 책임 연구원. 2003년~현재 단국대학교 공과대학 컴퓨터학부 컴퓨터공학 전공 조교수 2005년~2006년 UC Santa Cruz 방문 교수. 관심분야는 운영체제, 임베디드시스템, 차세대 저장장치

이 동 희
1989년 서울대학교 컴퓨터공학과(학사)
1991년 서울대학교 컴퓨터공학과(석사)
1998년 서울대학교 컴퓨터공학과(박사)
1998년~1999년 삼성전자 중앙연구소 선임연구원. 1999년~2001년 제주대학교 통신컴퓨터공학부 조교수. 2002년~현재 서울시립대학교 컴퓨터과학부 부교수. 관심분야는 운영체제, 플래시메모리소프트웨어, 내장형시스템

노 삼 혁
정보과학회논문지 : 시스템 및 이론
제 36 권 제 1 호 참조