낸드 플래시 메모리의 특성을 활용하여 부가적인 쓰기 연산을 줄이는 저널링 기법을 위한 FTL

박대준°, 신동군
성균관대학교 전자전기컴퓨터공학과
pdaejun@skku.edu, dongkun@skku.deu

Zero-overhead Journaling Technique for NAND Flash Memory based Storage

Daejun Park°, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

저널링 기법은 파일시스템에서 일관성을 유지해주기 위해 사용하는 기법이다. 하지만 이 기법은 하드디스크를 저장장치로 하는 시스템에 최적화 되어 있어, 낸드 플래시 메모리를 기반으로 하는 저장장치에 적합하지 않은 부분이 존재한다. 저널링 기법에서는 파일시스템의 일관성을 보장해 주기 위해 새로 바꿀 데이터를 미리 저널영역에 기록한 후, 원래 쓰고자 하는 위치에 데이터를 쓰게 된다. 그런데 저널영역에 미리 쓰는 과정이 낸드 플래시 메모리에서 추가적인 쓰기 연산을 요구하기 때문에 성능을 하락시키고, 수명을 단축시키는 원인이 된다. 이러한 문제를 해결하기 위해 저널영역에 미리 기록하는 기법을 통해 원자성을 보장하는 것이 아니라, FTL상에서 매핑의 관리를 통해 원자성을 보장하는 기법을 제안하고 있다. 제안한 기법을 통해 낸드 플래시 메모리에 쓰이는 페이지 수가 효과적으로 줄어드는 것을 확인 할 수 있었다.

1. 서 론

낸드 플래시 메모리를 기반으로 한 저장장치가 점차 대중화됨에 따라, 시스템의 저장장치로 하드디스크를 대체하는 경우가 늘어나고 있다. 특히 스마트폰의 대중화로 낸드 플래시 메모리를 기반으로 한 저장장치인 eMMC가 많이 쓰이고 있다. 운영체제에서 저장장치를 관리하는 파일시스템은, 오랜 시간동안 하드디스크를 기반으로 최적화 되었다. 그런 이유로 현재의 파일시스템이 하드디스크와 다른 낸드 플래시 메모리로 이루어진 저장장치의 특성을 효율적으로 활용하는지 살펴볼 필요가 있다.

저널링 기법[1]은 파일시스템에서 일관성을 유지하기 위하여 사용하는 기법인데, 덮어쓰기로 데이터를 갱신하는 저장장치인 하드디스크에 적합한 기법이다.

한편 낸드 플래시 메모리에서 데이터를 갱신하는 과정을 살펴보면, 물리적 특성에 의해 기존의 공간에 데이터를 덮어쓰지 못하기 때문에, 새로운 공간을 할당하여 데이터를 쓴 후에 기존의 매핑을 변경하고 있다.

이러한 차이점이 있기 때문에 저널링 기법이 낸드 플래시 기반에서 효율적인지 관찰 할 필요가 있다.

저널링 기법에서는 변경하고자 하는 메타데이터를 저널영역에 미리 씀으로써 원자성을 획득한다. 이를 통해 메타데이터를 덮어쓰던 도중에 시스템에 문제가 생기더라도, 복구를 할 수 있게 된다.

하지만 낸드 플래시 기반의 저장장치에서는 저널영역에 변경하고자 하는 메타데이터를 미리 쓰지 않더라도, 변경되기 이전의 메타데이터가 지워지지 않는다면, 메타데이터의 변경 중에 시스템에 문제가 생기더라도 이전의 메타데이터로 복구가 가능할 수 있다.

낸드 플래시 기반의 저장장치인 eMMC를 사용하는 갤럭시 S3에서 관찰되는 저장장치의 쓰기 패턴을 보면, 보통의 경우 저널영역에 기록하는 양이 전체 기록양의 약 16%를 차지하고, 유휴상태일 경우에는 약 71%을 차지한다. 낸드 플래시 메모리 기반의 저장장치에서 저널영역 쓰기로 인해 생기는 추가적인 쓰기는 저장장치의 쓰기 성능에 영향을 끼치게 된다. 또한 낸드 플래시 메모리는 PE(Program Erase) cycle이 제한되어 있기 때문에 저널영역에 추가적인 쓰기를 수행하게 되면 저장장치의 수명을 줄이게 된다. 본 논문에서 이러한 문제점을 해결하기 위해 FTL의 변경과 그에 따른 파일시스템의 수정을 통해, 저널 영역에 데이터를 미리 쓰지 않고도 저널링을 수행 할 수 있는 방법을 제안하겠다.

2. 관련 연구

저널 영역에의 부가적인 쓰기 연산을 피하기 위하여 여러 논문에서 다른 기법을 통해 접근하였다. 저널 영역에 쓰는 데이터를 이용하여 추가적인 쓰기를 줄이거나, 저널 영역에 쓰지 않고 원자성을 획득하는 방법이 있었다.

그림 1은 JFTL[2]에 대한 설명이다. JFTL은 저널영역에

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음
[KI0018-10041244, 스마트TV 2.0 소프트웨어 플랫폼]

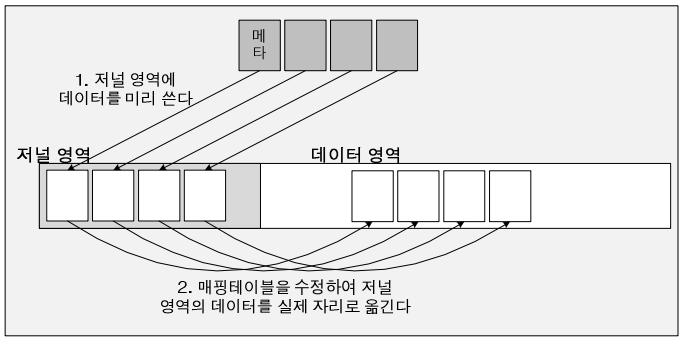

그림 1 JFTL의 동작 원리

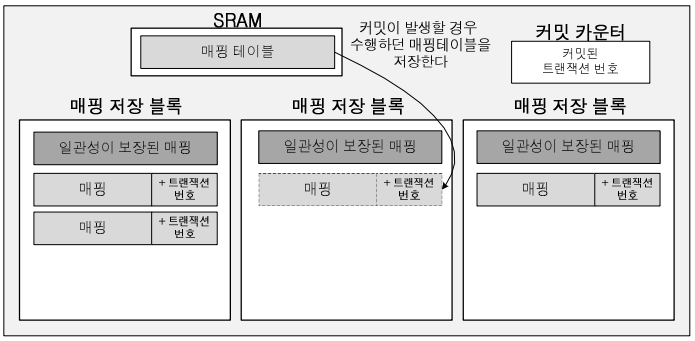

그림 2 T-FTL의 동작 원리

쓴 데이터를 기록에 이용하는 기법이다. 저널영역에 미리 쓰이게 되는 데이터를 활용하여, 데이터의 원래 위치에 쓰게 될 때는 실제로 데이터를 쓰는 것이 아니라, 저널영역에 있는 데이터에 대한 매핑을 수정하여 옮기는 방법을 사용한다. 이렇게 되면 저널 영역에 쓴 데이터가 더 이상 부가적인 기록으로 남지 않는다.

하지만 JFTL은 낸드 플래시 메모리의 페이지 크기가 메타데이터의 크기보다 클 경우 사용할 수 없다. 페이지가 메타데이터의 크기보다 크다면, 매핑을 수정하는 단순한 작업이 아니라, 페이지에서 메타데이터를 제외한 영역에 속하는 데이터를 추가로 기록하는 과정이 필요하기 때문이다.

Atomic write[3]는 저널 영역에 데이터를 쓰지 않고, 데이터를 쓸 때, FTL에서 원자성을 보장해주는 기법이다.

어플리케이션에서 Atomic write에 해당하는 데이터를 특정 명령을 통해 FTL에게 전해주게 되면, FTL은 이러한 데이터를 원자적으로 쓰는 것을 보장해 준다.

이때 데이터를 원자적으로 쓰기 위해 각 페이지를 쓰면서 로그를 남기게 되는데, 시스템에 문제가 생겼을 때 로그를 통해 복구를 할 수 있다. 그러나 로그를 남겨서 복구에 쓰는 방법은 저장장치에 쓰이는 매핑을 이용하지 못한 방법이다.

본 논문에서는 저널 영역에 미리 쓰지 않고, 데이터를 쓸 때 FTL에서 원자성을 보장해 주는 방식을 사용하고 있다. 그러나 Atomic write기법과는 달리 매핑이 저장장치에 쓰인다는 점에 착안하여, 로그를 사용하는 것이 아니라 매핑을 저장할 때 추가적인 정보를 함께 저장하여 원자성을 제공한다.

3. T-FTL

3.1 기존 저널링 기법

저널링 기법은 파일시스템의 일관성을 유지하기 위해 사용하는 기법이다.

메타데이터를 변경하기 전에, 변경되는 메타데이터를 미리 저널영역에 기록한다. 이때 메타데이터를 저널영역에 모두 기록한 상태를 커밋이라고 한다. 그 후 메타데이터를 원래 변경하고자 하는 위치에 쓴다. 이런 과정을 수행하는 메타데이터의 집합을 트랜잭션이라고 한다. 저널링 기법은 트랜잭션을 단위로 파일시스템의 일관성을 유지하게 된다. 시스템에 문제가 생겨 트랜잭션에 해당하는 메타데이터를 저장장치에 기록하다 중단되더라도, 저널영역에 미리 쓰인 데이터가 있기 때문에 이를 이용하여 복구 할 수 있다.

3.2 T-FTL의 동작 설명

기존 저널링 기법에서는 트랜잭션에게 원자성을 부여하기 위해 저널영역에 트랜잭션에 해당하는 메타데이터를 미리 기록하는 방식을 사용하고 있다.

그러나 제안하는 기법에서는 트랜잭션에게 원자성을 부여하기 위해, FTL에서 해당하는 트랜잭션에 대한 정보를 매핑 정보와 함께 저장하는 방식을 사용한다.

일반 데이터를 적을 때는 기존 FTL과 같은 동작을 수행한다. 하지만 매핑을 저장장치에 기록하게 될 때 수행중인 트랜잭션의 번호를 매핑과 함께 적어주게 된다.

그리고 트랜잭션이 모두 저장장치에 저장된 커밋 상태일 때, 파일시스템에서 특정 커맨드를 통해 알려주게 된다. 이때 FTL은 현재 SRAM에 로딩 되어 있는 매핑을 저장장치에 트랜잭션의 번호와 함께 기록한다. 또한 FTL에서 커밋 카운터를 관리하여, 해당 트랜잭션 번호를 카운터에 갱신시켜 주게 된다. 이 동작을 통해 커밋 카운터에 저장된 트랜잭션 번호에 해당하는 메타데이터가 모두 저장장치에 저장되었다는 것을 보장할 수 있다.

이렇게 커밋이 이루어진 상태의 매핑을 일관성이 보장된 매핑이라고 할 수 있는데, FTL에서 일관성이 보장된 매핑을 관리하여, 해당 매핑에서 가리키는 페이지들이 무효화되지 않도록 막는다. 이를 통해 마지막으로 커밋된 트랜잭션의 메타데이터가 가비지컬렉션에 의해 삭제되지 않는 것을 보장하고, 이를 문제가 생겼을 때 이용하여, 일관성이 유지되는 상태로 돌아갈 수 있게 된다.

만일 파일시스템에서 트랜잭션에 해당하는 모든 데이터를 저장장치에 저장하지 못하고 문제가 생겼을 경우, FTL이 복구를 수행하게 된다.

복구 과정은 저장되어있는 커밋 카운터를 읽는 것에서 시작한다. 그 다음 매핑이 저장되어 있는 매핑 저장 블록을 검색하여, 매핑과 함께 저장된 트랜잭션 번호가 커밋 카운터보다 높을 경우에는 모두 제거 시킨다. 이를 제거하는 이유는, 커밋이 일어나지 않은 트랜잭션에 해당하는 매핑이기 때문이다. 이러한 동작을 통해 마지막으로 커밋된 트랜잭션 번호에 해당하는 메타데이터를 가

리키고 있는 매핑만 저장장치에 남아있게 되어서, 트랜잭션에 대한 원자성을 보장해 주고 이로 인해 파일시스템은 일관성을 가지게 된다.

3.3 T-FTL을 위한 파일 시스템의 변경

제안하는 기법은 저널 영역에 미리 메타데이터를 적지 않고, 트랜잭션 정보와 함께 바로 원래의 위치에 메타데이터를 적게 된다. 이때 각 메타데이터에 해당하는 트랜잭션의 번호를 FTL에게 알려주게 된다. 이를 통해 FTL에서는 트랜잭션을 구분하여 원자적으로 쓸 수 있게 된다. 저널링 기법에서는 트랜잭션에 해당하는 메타데이터를 모두 기록하게 되었을 때 커밋 기록을 저장장치에 저장하게 된다. 제안하는 FTL을 위해 수정된 파일시스템에서는 커밋 상태를 커밋된 트랜잭션 번호와 함께 FTL에게 알려준다.

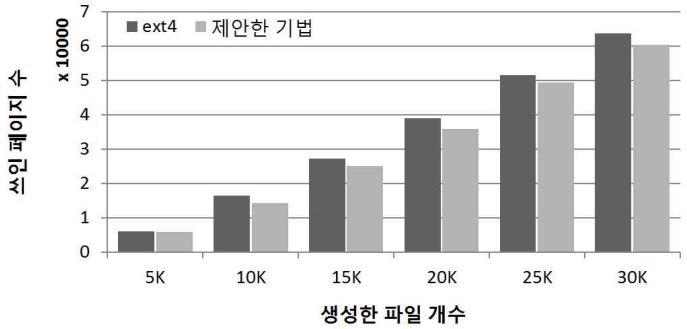

그림 3 postmark 수행 후 낸드 플래시에 쓰인 페이지 개수 비교

4. 실험

4.1 벤치마크 실험 결과 비교

실험을 위해 갤럭시 S4에서 입출력 트레이스를 추출한 후 eMMC 시뮬레이터에서 수행을 하였다. 시뮬레이터에서 낸드 플래시의 페이지 크기는 4KB로 설정 하였다. 제안하는 FTL에 적합하도록 커널의 파일시스템을 수정하고, 시뮬레이터 상에 제안하는 FTL을 구현하였다. 제안하는 FTL과 비교하기 위해 파일시스템은 기존 ext4를 사용하였고 저널링은 메타데이터에 대해서만 수행하였다. FTL은 전체 페이지 매핑을 시뮬레이터에 적용하였다. 그림 3은 postmark[4]를 사용하여 파일을 5000개부터 5000개 단위로 30000개까지 생성하는 것을 논문에서 제안한 기법과 기존 ext4에서 사용하는 저널링 기법에서 수행한 결과이다. 실험 결과를 살펴보면 제안하는 기법에서는 저널 영역에 추가적인 데이터를 쓰지 않기 때문에, 페이지 수가 더욱 적게 쓰이는 것을 관찰 할 수 있다. 페이지 수가 적어지므로, 제안한 기법에서는 낸드 플래시의 수명향상을 기대 할 수 있다. 또한 쓰기 시간이 읽기 시간에 비해 매우 큰 낸드 플래시에서 쓰기 횟수가 줄어들게 되면 전체적인 입출력 성능향상을 기대 할 수 있다.

4.2 실제 워크로드에서 결과 비교

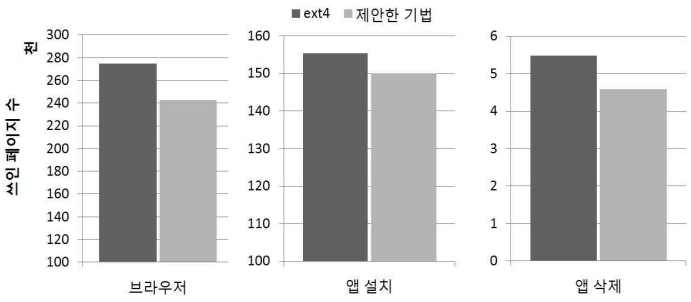

그림 4 실제 워크로드에서 낸드 플래시에 쓰인 페이지 개수 비교

실제 워크로드에서의 성능 평가를 위해 벤치마크 실험과 환경에서 사용자가 겪을 수 있는 여러 시나리오를 수행하면서 페이지 쓰기 횟수를 관찰하였다.

브라우저는 인터넷을 2시간 동안 웹페이지들을 방문하고 각 웹페이지는 10초간 보는 것을 가정한 워크로드이고, 앱 설치는 평균 16MB 크기의 16개의 어플리케이션을 설치한 워크로드이다. 마지막으로 앱 삭제는 앱 설치를 통해 설치한 어플리케이션을 삭제한 워크로드이다.

그림 4를 보면 제안한 기법이 기존 ext4에 비해 쓰기 횟수가 줄어든 것을 확인 할 수 있다. 이는 제안한 기법에서 저널영역에 추가 쓰기를 하지 않으면서도 일관성을 보장해주고 있기 때문이다.

5. 결론

기존 저장장치와는 다른 특성을 가진 낸드 플래시 기반의 저장장치가 대중화 되면서, 기존 저장장치를 위해 설계된 파일시스템과는 다른 접근방식이 불가피하다.

본 논문에서는 낸드 플래시 메모리가 데이터를 덮어쓰지 않고, 데이터에 대한 매핑을 바꾸는 방식으로 갱신한다는 점에 착안하여 매핑에 추가적인 정보를 기록하여 트랜잭션에 대해 원자성을 가지게 하였다.

이를 통해 낸드 플래시 메모리 기반의 저장장치에 적합하지 않은 저널 영역 쓰기를 제거하여 추가적인 쓰기 연산 없이 일관성을 제공하였다.

향후 연구로는 제안한 FTL을 위한 파일시스템에서 성능을 높이기 위해, 메타데이터들을 논리 주소에 연속되도록 할당하는 방법에 대하여 연구할 계획이다.

참고문헌

[1] Tweedie, Stephen C. “Journaling the Linux ext2fs filesystem.” The Fourth Annual Linux Expo. 1998.

[2] Choi, Hyun Jin, et al. “JFTL: A flash translation layer based on a journal remapping for flash memory.” ACM Transactions on Storage, 2009

[3] Ouyang, Xiangyong, et al. “Beyond block I/O: Rethinking traditional storage primitives.” High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on. IEEE, 2011.

[4] Katcher, J. Postmark: A new file system benchmark. Technical Report TR3022, Network Appliance, 1997.