플래시 메모리 기반 저장 시스템의 저널링 영향 분석

장보길*, 임승호*
*한국외국어대학교 디지털정보공학과
e-mail : xearo1@naver.com, slim@hufs.ac.kr

Analysis of Journaling Effect for Flash Memory Storage System

Bogil Jang*, Seung-Ho Lim*
*Dept. of Digital Information Engineering, Hankuk University of Foreign Studies

요 약

낸드 플래시 메모리는 저장장치로서 널리 사용되는 소자이다. 플래시 메모리 기반의 저장장치 시스템에서 파일 시스템의 저널링 기능을 사용할 때, 플래시 메모리의 특징을 고려해주는 것이 필요하다. 본 연구에서는 플래시 메모리 기반의 저장장치 시스템에서 파일 시스템의 저널링 연산이 미치는 영향에 대해서 분석해 보고, 오버헤드가 될 만한 부분을 찾아본다. 이러한 오버헤드가 될 만한 부분을 찾아서 제거해줌으로써 플래시 메모리의 사용성을 증대시킬 수 있다.

1. 서론

플래시 메모리, 특히 낸드 플래시의 급격한 기술 발전은 마그네틱 기반의 저장장치 디스크로부터 낸드 플래시 메모리 기반의 저장장치의 사용을 점점 증대시키고 있다. 그런데, 낸드 플래시 메모리 기반의 저장장치는 물리적인 특성상 마그네틱 기반의 저장장치와는 많은 다른 특징을 가지고 있기 때문에 기존의 마그네틱 기반의 저장장치의 성능을 향상시키기 위한 많은 기술들이 낸드 플래시 메모리 기반의 저장장치 시스템에는 맞지 않는 성향이 있다.

특히, 파일 시스템의 운영 방법은 저장장치의 성능과 밀접한 연관이 있다. 대부분의 파일 시스템은 데이터의 빠른 복구와 일관성 유지를 위해서 저널링 기능을 사용하고 있는데, 이 저널링 방법이 낸드 플래시 메모리의 특징을 활용하지 못하고 있기 때문에 성능을 저하시키는 요인이 된다. 본 논문에서는 낸드 플래시 메모리 기반의 저장 시스템에서 사용되는 파일 시스템의 저널링 방법에 대해서 분석하고, 오버헤드가 될 만한 요소를 도출해보도록 한다. 본 논문은 다음과 같이 구성되어 있다. 2 장에서는 플래시 메모리의 특징을 간략히 정리하고, 3 장에서는 플래시 메모리 기반의 저장 시스템에서 저널링 기법을 분석하고 그 오버헤드에 대해서 정리한다. 4 장에서는 본 논문의 결론을 맺도록 한다.

2. 낸드 플래시 메모리 및 플래시 변환 계층

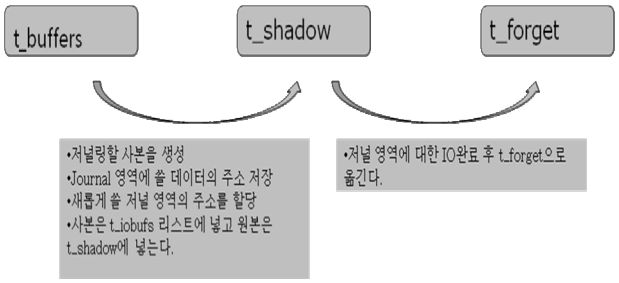
낸드 플래시 메모리의 기본 동작방식은 읽기, 쓰기, 지우기 연산으로 구성된다. 읽기와 쓰기는 페이지(Page)라고 하는 낸드 플래시 메모리 단위로 이루어지며, 지우기는 블록(Block)이라고 하는 단위로 이루어진다. 일반적으로 페이지의 크기는 2KB, 4KB, 8KB 정도의 크기를 가지며, 한 블록은 64 개 또는 128 개의 페이지로 구성되어 128KB, 256KB, 512KB 정도의 크기이다. 소자의 특성상 낸드 플래시 메모리는 In-Place-Update 가 되지 않기 때문에 데이터 쓰기 연산은 항상 지워져 있는 프리 영역에 이루어진다. 만약, 프리 영역이 없을 경우에는, 기존에 데이터가 쓰여진 영역 중 한 블록을 축출하여, 유효한 데이터만 복사를 한 후, 해당 블록을 지워줌으로써 프리 영역을 확보해야 한다. 이러한 과정을 가비지 콜렉션(GC, Garbage Collection)이라고 한다. 그렇기 때문에 플래시 소프트웨어는 데이터의 논리 주소와 물리주소를 매핑 시켜주는 매핑 알고리즘과 GC 알고리즘을 이용하여 낸드 플래시 메모리 저장장치를 관리한다. 이러한 관리 소프트웨어를 플래시 변환 계층(FTL)이라고 한다. 낸드 플래시 메모리 기반의 저장장치는 플래시 변환계층이 필수적인 소프트웨어 이다.

3. 저널링 파일 시스템 및 저널링 분석

저널링 파일 시스템은 파일 시스템의 일정 부분에 수정된 데이터를 로그로 기록하여 백업 복구 시에 사용되도록 만든 복구 능력이 있는 파일 시스템을 말한다. 사용자가 파일 시스템의 파일이나 내용을 입력 또는 수정하면 그 내용을 물리적인 디스크에 기록하기 바로 전에 관련 내용을 저널 로그(log)라는 별도 영역에 기록한다. 만약 로그 영역에 데이터를 기록 중에 비정상적인 종료가 이루어진 것을 확인하면, 다시 부팅할 때 로그에 기록된 내용을 참고로 하여 데이터를 복구하는 방법이다. 이러한 방법을 통해서 복구 시간을 최소화 시켜줄 수 있으며, 기존의 데이터를 로그 영역으로부터 복구 시킬 수 있기 때문에 파일 시스템의 신뢰성을 보장시켜 준다.

저널링 파일 시스템의 쓰기 단위는 트랜잭션이라는

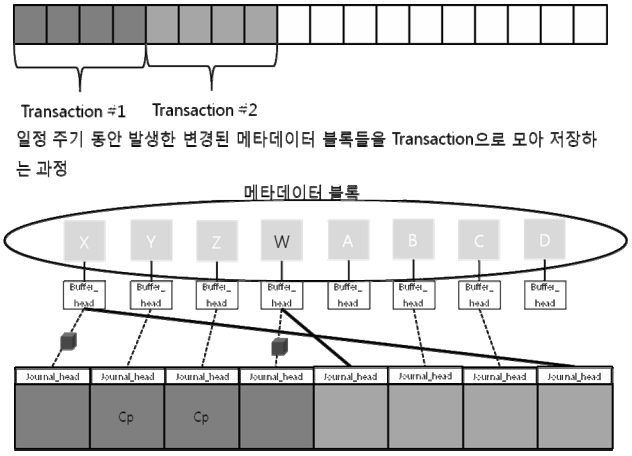
단위를 사용하며, 하나의 트랜잭션은 Running, Commit, Checkpoint 라는 주기를 통해서 actomic 하게 로그 영역에 쓰이게 된다. 트랜잭션의 주기 중에서 Running 은 해당 트랜잭션에 데이터를 모으는 과정이며, Commit 은 실제 로그 영역에 데이터 쓰기를 수행하는 단계이고, Checkpoint 는 로그 영역 쓰기 완료 후, 관련 데이터가 홈 영역에 쓰이면, 로그 영역에서 해제되는 단계이다. 트랜잭션의 단계 중에서, 실제 데이터 쓰기가 발생할 때 journal header 의 데이터 버퍼들이 실제 저널 트랜잭션 데이터 구조에서 이동하는 처리 과정을 그림 1 에 나타내었다. 그림에서와 같이 Commit 단계에서 해당 트랜잭션의 데이터 버터들은 t_buffers, t_shadows, t_forget 단계로 이동하면서 로그 영역에 쓰이게 되고, 각각의 관련 저널 블록 넘버, 홈 영역 블록 넘버 등에 대한 기록을 남기게 된다.



(그림 1) 저널링 Commit 시 Journal_head 의 Process 과정

그림 2 는 트랜잭션의 commit 시 발생하는 overhead 를 분석한 그림이다. 저널 트랜잭션의 한 주기는 running, commit, checkpoint 이지만, 저널 로그 영역의 전체 주기를 봤을 때는, 다수 개의 트랜잭션이 running 과 commit 을 반복하다가, 저널 로그 영역이 다 차게 되면, 비로소 checkpoint 를 수행해 주게 된다. 이러한 다수개의 running, commit 이 발생할 때 commit 된 블록이 반복해서 계속 발생하는 현상이 발생한다. 그림 2 는 그 과정을 도식화 한 것인데, 그림 2 에서와 같이 두 개의 연속된 트랜잭션 commit 에서 같은 데이터 블록이 연속적으로 포함되어 있는 것을 발견할 수 있다. 그림 2 에서는 Cp 로 표시된 블록이다. 이러한 블록은 다음 Commit 에도 포함되어 있기 때문에 전혀 사용되지 않는 데이터 블록이다. 즉, commit 이 되었기 때문에 블록은 유효한 데이터를 포함하고 있는 듯이 표시하고 있지만, 실제로는 쓸모 없는 데이터를 포함하고 있는 블록이다.

이렇게 저널 구조상에서는 무효한 데이터이지만, 실제 블록 레벨 아래의 저장장치 단에서는 유효한 데이터를 가지는 것처럼 보이는 블록들은 저장장치에 안 좋은 영향을 미친다. 게다가 이는 플래시 메모리에서는 더 안 좋은 결과를 낳게 된다. 플래시 메모리에서는 이러한 블록들이 유효한 데이터를 포함하고 있다고 간주하기 때문에 가비지 콜렉션 시에 유효한 페이지로 간주하여 데이터 복사가 발생한다. 그러나 실제로는 저널 로그상에서 유효한 데이터가 아니기 때문에 가비지 컬렉션시 유효한 카피를 하지 않고 무효한 데이터로 표시할 수 있을 것이다. 실제로 저널링 파일 시스템과 플래시 메모리 저장장치로 이루어진 시스템의 데이터 로그를 분석해 보면, 이러한 저널 로그 블록들이 많이 발생함을 확인 할 수 있었다. 바꾸어서 말하면, 이러한 블록들을 제거해 주면 많은 성능 향상이 일어날 수 있을 것으로 기대할 수 있다. 구체적으로는, 플래시 메모리의 가비지 컬렉션 성능이 향상될 수 있으며, 플래시 메모리 쓰기 연산 횟수의 감소로 인해서 플래시 메모리의 수명을 연장 시킬 수 있으며, 저널링 commit 시의 응답속도를 높일 수 있을 것으로 기대된다.



(그림 2) Journal Commit 시 발생하는 Duplication Overhead

4. 결론

플래시 메모리 기반의 저장장치를 기본 저장 시스템 및 파일 시스템으로 사용하는 시스템에서는 저널링 영역의 트랜잭션으로 인한 오버헤드가 발생할 수 있다. 본 연구에서는 저널링 트랜잭션 시에 발생하는 연산을 분석하고, 플래시 메모리에서 오버헤드가 될 만한 부분을 고찰해보았다. 저널링 연산의 분석을 통해서 저널링 영역의 commit 시 발생하는 중복된 데이터를 발견할 수 있었으며, 이를 제거해줌으로써 플래시 메모리의 사용성을 증대시킬 수 있음을 알 수 있다.

참고문헌

[1] “"Memory Technology Device (MTD) subsystem for Linux.”", http://www.linux- mtd.infradead.org.
[2] Intel Corporation, “"Understanding the flash translation layer(FTL) specification”", http://developer.intel.com/.
[3] A. Ban, “"Flash file system,”", United States Patent, no. 5,404,485, April 1995.
[4] S. Tweedie, “Journaling the Linux ext2fs filesystem", LinuxExpo'98, 1998.
[5] S. Best, D. Gordon and I. Haddad, “IBM's journaled filesystem", Linux Journal, 2003.