멀티미디어 파일 시스템을 위한 회복 기법의 설계 및 구현

김영철° 김홍연 김병섭 원종호 이미영
한국전자통신연구원 컴퓨터시스템연구부
{kimyc°, kimhy, powerkim, jhwon, mylee}@etri.re.kr

Design and Implementation of the Recovery Method for Multimedia File System

Youngchul Kim° Hongyeon Kim Byungseob Kim Jongho Won Miyoung Lee
Computer System Department, Electronics and Telecommunications Research Institute

요 약

본 논문에서는 차세대 인터넷 서버의 멀티미디어 파일 시스템인 Content Container 파일 시스템(CCFS)의 로그 기반 회복 기법을 설계하고 구현한 내용에 대해 기술한다. CCFS에서는 트랜잭션들이 개별적으로 로깅 할 수 있도록 함으로써 동시성을 높이면서 트랜잭션 수행 중에 로깅으로 인한 오버헤드를 줄인다. 그리고 로그 파일을 순환 구조로 사용할 수 있도록 효율적인 관리를 제공한다. 또한 시스템 고장이 발생하였을 때 로그 파일을 한번만 스캔하면서 완료한 트랜잭션에 대한 로그를 재수행함으로써 시스템의 빠른 회복과 재시작을 보장한다.

1. 서 론

파일 시스템은 크게 사용자 데이터와 메타 데이터로 구성된다. 여기서 메타 데이터는 슈퍼 블록, 비트맵, inode, 디렉토리 등과 같이 사용자 데이터를 조직하고 관리하기 위한 정보로 파일 시스템의 구조를 형성한다.

이러한 메타 데이터는 파일 생성, 삭제, 수정 등의 파일 동작에 의해 빈번하게 변경되기 때문에 파일 시스템 성능에 중요한 영향을 미친다. 또한 시스템 고장이 발생하였을 때 메타 데이터를 일관성 있는 상태로 회복하지 못한다면 추후 파일 동작에 중대한 문제를 발생시키고 파일 시스템 전체를 불능의 상태로도 만들 수 있다.

그러므로 파일 시스템의 중요한 요구사항은 시스템 고장으로부터 안정된 회복을 지원하면서 높은 성능을 제공하는 것이다. 이러한 요구사항은 실시간 멀티미디어 스트리밍 서비스를 제공하기 위한 멀티미디어 파일 시스템 환경에서 보다 중요하다.

본 논문에서는 현재 한국전자통신연구원에서 개발하고 있는 차세대 인터넷 서버의 멀티미디어 파일 시스템인 Content Container 파일 시스템(이하 CCFS)에서 빠른 회복을 제공하기 위한 효율적인 회복 기법에 대해 기술한다. 차세대 인터넷 서버는 HDTV 급의 실시간 멀티미디어 서비스를 제공하기 위하여, 빌딩, 아파트, 학교 등의 지역망을 이용하여 스트리밍 서비스를 가능하도록 네트워킹 기능을 강화한 지역 서버와 데이터 센터용 광역 서버를 포함하는 계층적 구조를 갖는 서비스 시스템이다[1].

이 논문은 다음과 같이 구성된다. 먼저 2장에서는 기존 파일 시스템들에서 연구, 개발되어 온 회복 기법들에 대해 살펴보고, 3장에서 CCFS에 대한 전반적인 소개를 한다. 그리고 4장에서 CCFS의 회복 기법을 설계하고 구현한 내용을 서술하고, 5장에서 결론을 맺는다.

2. 관련연구

파일 시스템의 효율적인 회복과 성능 향상을 지원하기 위한 여러 가지 방법들이 꾸준히 연구, 개발되어 오고 있다.

FFS(Fast File System)에서는 갱신된 메타 데이터들을 고정된 순서에 따라 동기화된 저장(synchronous write)을 하도록 한다. 예를 들어, 파일 생성의 경우, 새로 생성된 파일의 inode를 먼저 저장하고 다음에 그 inode를 포함하는 디렉토리를 저장함으로써 시스템 고장이 발생하더라도 적어도 회복이 가능한 상태를 보장할 수 있다. 하지만 동기화된 저장을 위한 많은 디스크 I/O 비용과 fsck를 이용한 비효율적인 회복은 파일 시스템의 성능을 저하시키는 문제점이 있다.

Soft Updates는 버퍼에 적재된 메타 데이터들 간의 의존성(dependency) 관계를 유지함으로써 메타 데이터의 동기화된 디스크 I/O를 하지 않으면서도 메타 데이터가 회복 가능한 순서로 저장되도록 보장하기 위한 기법이다[8,9].

LFS(Log-Structured Filesystems)는 사용자 데이터와 메타 데이터를 하나의 로그 장치에 모두 저장함으로써 순차적으로 기록되는 로그 장치의 특성을 이용하고자 하였다[11].

최근에 대부분의 파일 시스템들은 데이터베이스 시스템의 로깅(저널링)을 이용한 회복 기법을 채용하고 있다. 즉 메타 데이터의 갱신된 내용만을 로그에 저장하고 시스템 고장이 발생하면 그 로그를 재수행함으로써 메타 데이터의 일관성을 보장한다. 따라서 메타 데이터에 대한 동기화된 저장이 필요 없으며 의존성에 따라 저장되는 순서를 고려하지 않아도 되는 반면 로깅으로 인한 오버헤드를 갖는다. 대표적인 저널링 파일 시스템으로는 Ext3, XFS, JFS, ReiserFS 등을 들 수 있다[3,4,5,6,7]. CCFS에서는 이러한 특징을 반영하여 로그 기반의 회복 기법을 설계하고 구현하였다.

3. Content Container File System

CCFS는 차세대 인터넷 서버의 멀티미디어 파일 시스템으로 여러 가지 특징을 갖는다. 먼저 멀티미디어 스트리밍 서비스를 만족할 수 있는 빠른 파일 입출력을 제공하며 대용량 멀티미디어 파일의 저장과 검색을 지원한다. 또한 빈번하게 생성되거나 삭제되는 멀티미디어 파일에 대한 적합한 저장 공간 관리와 다중 사용자를 위한 효율적인 멀티미디어 파일 접근 방법, 빠른 회복과 재시작, Multithread-Safe하면서 POSIX 표준에 유사한 인터페이스를 제공한다.

CCFS는 사용자 수준의 파일 시스템으로 설계되고 구현되었으며 그림 1에서와 같이 여러 개의 관리기들로 구성된다. 공유 메모리 관리기는 트랜잭션 테이블, 메타 데이터 버퍼와 같이 파일 시스템 동작를 위한 공유 데이터를 관리한다. 그리고 트랜잭션 관리기에서는 인터페이스 수준을 트랜잭션 단위로 하여 파일 동작의 원자성을 제공한다. 메타 데이터에 대한 갱신은 주로 디스크, 파일, 디렉토리, B+ 트리 관리기에서 이루어지며, 트랜잭션들의 메타 데이터 접근에 대한 동시성 제어는 잠금 관리기가 담당한다.

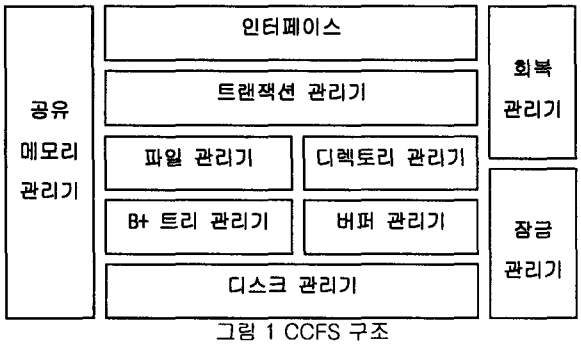

그림 1 CCFS 구조

4. 회복 기법의 설계 및 구현

4.1 개요

디스크에 저장되는 모든 메타 데이터에 대한 갱신은 버퍼에서 이루어진다. 따라서 파일 시스템의 회복은 버퍼 정책과 밀접하게 관련된다고 할 수 있다.

CCFS에서는 갱신된 버퍼의 내용을 디스크에 반영하기 이전에 반드시 해당 로그 레코드를 먼저 저장하도록 하는 WAL(Write Ahead Logging) 프로토콜을 준수한다. 그리고 파일 시스템의 트랜잭션은 매우 빠르게 수행되는 소규모 트랜잭션이라는 특성을 반영하여 메타 데이터 버퍼는 no steal과 no force 정책을 채용한다. 즉 트랜잭션이 완료하기 이전에 변경된 버퍼의 내용이 디스크에 반영되지 않으며, 트랜잭션은 자신이 갱신한 버퍼가 디스크에 반영되었는지 여부와 상관없이 완료할 수 있다. 따라서 디스크상의 메타 데이터는 완료한 트랜잭션들에 의해 반영된 것임을 보장할 수 있으므로 시스템 고장으로 인한 회복 시에 undo 작업은 필요 없으며 단지 완료한 트랜잭션의 redo 만을 수행하면 된다. 그리고 메타 데이터 블록에는 버전 번호를 기록함으로써 회복 시에 재수행 여부를 판단할 수 있다.

현재 CCFS에서는 대부분의 파일 시스템 처럼 시스템 고장에 대한 메타 데이터 회복만을 지원하며 미디어 고장에 대한 회복은 지원하지 않는다.

4.2 로그 파일 관리

로그 파일은 파일 시스템을 포맷할 때 동일한 디바이스에 연속적인 공간으로 할당된 후에 시스템이 수행되는 동안 트랜잭션의 로그 레코드가 순차적으로 저장된다. 하지만 시스템 고장에 대한 회복을 위해서 로그 파일에 로그 레코드를 계속 유지하고 있을 필요는 없다. 따라서 CCFS에서는 로그 파일을 순환 구조(circular structure)로 관리한다. 즉 로그 레코드를 저장하다가 로그 파일의 끝에 도달하면 다시 로그 파일의 처음부터 로그 레코드를 기록하도록 한다.

이를 위해서는 로그 파일에 최소한의 유효한 로그 레코드만을 유지하기 위한 효율적인 관리 방법이 필요하다. 여기서 유효한 로그 레코드란 시스템 고장이 발생하였을 때 시스템을 일관성 있는 상태로 회복하기 위해 필요한 로그 레코드를 말한다. 반면에 해당 트랜잭션이 완료하였거나 롤백 되었고 메타 데이터의 갱신된 부분이 이미 디스크에 반영된 로그 레코드는 더 이상 로그 파일에 유지할 필요가 없다.

CCFS에서는 그림 2와 같이 세 종류의 위치를 조정하면서 로그 파일을 순환 구조로 관리한다. 여기서 LSN(Log Sequence Number)은 파일 시스템 내에서 유일한 값으로 로그 파일에 저장되는 모든 로그 레코드에 할당된다. 이 LSN 값은 로그 파일이 순환한 회수를 가리키는 Sequence Number와 로그 파일에서의 상대적인 변위인 Offset으로 구성된다. 따라서 LSN 값으로 로그 레코드들 간의 생성 순서를 정할 수 있으며 로그 레코드의 로그 파일에서 위치를 알 수 있다.

그림 2에서 Current LSN(4,8)은 최근에 추가된 로그 레코드의 LSN 값이고, Stable LSN(4,4)은 최근에 로그 파일에 저장된 마지막 로그 레코드의 LSN 값이다. 그리고 Effective LSN(3,30)은 유효한 로그 레코드들 중에서 가장 작은 LSN 값이다. 그러므로 그림 2에서 사용 가능한 로그 파일 공간은 Stable LSN과 Effective LSN 사이(어두운 부분)가 된다. 여기서 Effective LSN 값은 주기적 또는 정해진 이벤트에 따라 체크포인트를 수행함으로써 변경된다.

한편 로그 파일에서 재사용되는 공간은 디스크 I/O 비용을 없애기 위해 다시 초기화하지 않는다. 그러므로 시스템 회복을 위해 로그 파일을 읽어 가면서 로그 레코드를 재수행 할 때 로그 파일의 끝을 찾기 위한 방법이 필요하다. 이를 위해 CCFS에서는 로그 파일의 끝을 찾기 위한 효율적인 방법으로 LSN의 Sequence Number를 사용한다.

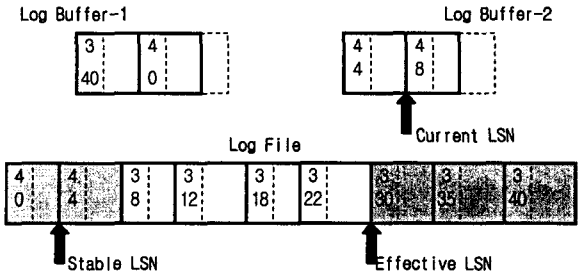

그림 2 로그 파일 관리

4.3 트랜잭션 로깅

CCFS에서는 메타 데이터를 갱신하는 트랜잭션에 한해 그 트랜잭션이 수행 중에 기록할 로그 레코드의 양을 미리 예측하여 개별적으로 로그 버퍼를 할당하고 그곳에 로그 레코드

를 추가하도록 한다. 따라서 트랜잭션이 수행 중에 로그 레코드를 기록할 때 다른 트랜잭션들과의 동시성 제어 비용을 줄일 수 있다.

트랜잭션이 기록하는 로그 레코드는 두 가지 형태를 갖는다. 첫번째는 메타 데이터 블록에서 변경된 데이터만을 기록하는 단일 로그 레코드 형태로 트랜잭션이 수행 중에 자신의 로그 버퍼에 추가한다. 그리고 두번째는 변경된 메타 데이터 블록 전체를 기록하는 블록 로그 레코드 형태로 단일 로그 레코드와 달리 메타 데이터 버퍼에서 직접 전역 로그 버퍼로 추가된다. 단, 이러한 블록 로그 레코드는 트랜잭션에 의해 블록 내의 많은 데이터가 변경된 블록에 대해서만 이루어진다. 예를 들어, 대용량의 파일을 삭제하는 트랜잭션의 경우 데이터 블록에 대한 할당 정보를 갖는 비트맵의 많은 부분을 갱신하게 될 것이다. 따라서 이때 비트맵 블록이 블록 로그 레코드의 대상이 될 수 있다. 이러한 블록 로그 레코드는 시스템 회복 시에 재수행 비용이 작지만 모든 메타 데이터 블록에 대하여 블록 로그 레코드를 기록한다면 트랜잭션 수행에서 로깅의 오버헤드가 커다란 부분을 차지할 수 있다. 그러므로 단일 로그 레코드와 블록 로그 레코드의 효과적인 사용이 필요하다.

트랜잭션은 완료할 때 트랜잭션 로그 버퍼와 블록 로그 레코드를 전역 로그 버퍼에 추가한다. 하지만 트랜잭션이 수행 중에 실패하였다면 단지 자신의 로그 버퍼를 무효화하고 블록 로그 레코드를 기록하지 않는다. 따라서 최대한 완료하지 못한 트랜잭션의 로그 레코드가 로그 파일에 반영되지 못하도록 한다.

로그 파일에 대한 저장은 그림 3과 같이 전역 로그 버퍼 단위로 이루어진다. 이때 전역 로그 버퍼에서는 로그 레코드를 추가한 트랜잭션들 간에 group commit을 제공함으로써 디스크 I/O를 줄일 수 있다. 또한 로그 파일에는 트랜잭션 단위로 로그 레코드가 저장되기 때문에 시스템 회복을 위한 로그 레코드 분석이 용이하다.

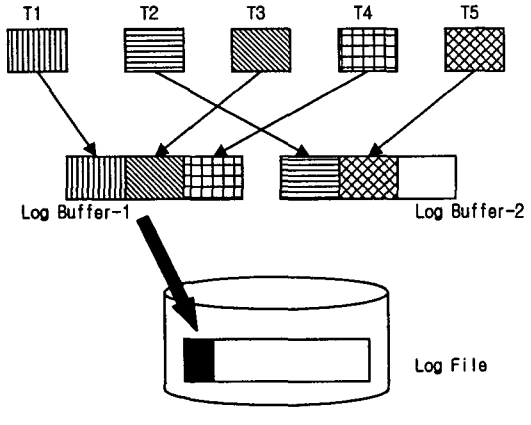

그림 3 트랜잭션 로깅

4.4 회복

트랜잭션이 완료하기 이전에 갱신한 버퍼를 디스크에 반영하지 못하도록 한다는 것은 시스템 고장에 대한 회복을 위하여 undo 작업이 불필요하다는 의미이다. 하지만 시스템 수행 중에 발생할 수 있는 트랜잭션 실패로 인한 롤백은 undo 작업이 필요하다. 왜냐하면 트랜잭션(T1)이 갱신한 버퍼의 내용을 디스크에 반영하지 않고 완료한 이후에, 다른 트랜잭션(T2)가 동일한 버퍼의 내용을 다시 갱신한 다음 완려하지 못하고 실패한다면, 이전의 T1에 의해 갱신된 상태로의 undo가 필요하기 때문이다. 그렇지 않고 단지 변경된 버퍼의 내용을 무시해 버린다면, 이미 완료된 T1에 의해 갱신된 부분이 디스크에 반영되지 못하고 무효화될 수 있다.

한편 시스템 고장에 대한 회복은 먼저 로그 파일에서 가장 작은 LSN 값을 가진 유효한 로그 레코드부터 시작하여 로그 파일을 앞으로 스캔하면서 완료한 트랜잭션의 로그 레코드 중에 갱신된 결과가 디스크에 반영되지 않은 것만을 재수행하도록 한다. 이때 CCFS에서는 버퍼를 이용하여 트랜잭션의 로그 레코드에 대한 분석과 재수행 작업을 한번의 로그 파일 스캔으로 완료함으로써 빠른 재시작을 제공한다.

5. 결 론

이 논문에서는 한국전자통신연구원에서 개발하고 있는 차세대 인터넷 서버의 멀티미디어 파일 시스템인 CCFS의 회복 기법을 설계하고 구현한 내용에 대해 기술하였다.

CCFS는 트랜잭션의 로그 양을 미리 예측하여 트랜잭션별로 로그 버퍼를 할당하고 개별적으로 로그 레코드를 기록함으로써 트랜잭션들 간의 동시성 제어 비용을 줄이고, 트랜잭션 수행 중 로깅으로 인한 오버헤드를 줄인다.

또한 순환 구조를 갖는 로그 파일을 효율적으로 관리하면서 시스템 고장이 발생하였을 때 로그 파일에서 유효한 로그 레코드들을 한번만 스캔함으로써 보다 빨리 시스템을 일관된 상태로 회복시킬 수 있다.

참 고 문 헌

[1] 김명준, 임기욱, " 차세대 인터넷 서버(SMART 서버) 기술 개발," 한국콘텐츠학회지, 제1권, 제1호, 2003.

[2] C.Mohan, et al. " ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM Trans. on Database Systems, Vol.17, No.1, pp.94-162, 1992.

[3] S.C.Tweedie, " Journaling the Linux ext2fs Filesystem," LinuxExpo' 98, 1998.

[4] XFS Homepage, http://oss.sgi.com/projects/xfs/

[5] JFS Homepage, http://www-124.ibm.com/developer/opensource/jfs/

[6] ReiserFS Homepage, http://www.namesys.com/

[7] S.Best, " Journaling File Systems," Linux Magazine, Oct. 2002.

[8] G.R.Ganger, et al. " Soft Updates: A Solution to the Metadata Update Problem in File Systems," ACM Trans. on Computer Systems, Vol.18, No.2, pp.127-153, 2000.

[9] M.I.Seltzer, et al. " Journaling versus Soft Update: Asynchronous meta-data Protection in File Systems," Proc. of USENIX Annual Technical Conference, 2000.

[10] M.Rosenblum, et al. " The Design and Implementation of a Log-Structured File System," ACM Trans. on Computer Systems, Vol.10, No.1, pp.26-52, 1992.