리눅스 멀티미디어 파일 시스템

손정수° 이민석
amamos@dallext.hansung.ac.kr, mslee@hansung.ac.kr

Multimedia File System for Linux

Jungsoo Sohn, Minsuk Lee
School of Computer Engineering, Hansung University

요 약

최근 개인용 멀티미디어 시스템이 늘어나면서 그 운영 체제로 내장형 리눅스가 많이 사용되고 있다. 기존 리눅스 파일 시스템들은 상대적으로 작은 파일의 입출력에 최적화 되어 있어 대용량, 순차적 읽기 중심, 파일의 재사용 가능성이 낮은 개인용 멀티미디어 시스템에는 적합하지 않다. 본 논문에서는 멀티미디어 시스템을 위한 새로운 리눅스 파일 시스템을 구현하였다. 새 파일 시스템에 대한 성능 측정 결과, 읽기의 경우 Ext2, Ext3 보다 각각 62.31%, 62.22% 높은 성능을 나타내었고, 쓰기/읽기 동시 수행에 있어서는 Ext2, Ext3 보다 각각 32.83%, 35.72% 높은 성능을 나타내었다. 또한 불시의 전원 차단에 대한 파일 시스템의 안정성도 매우 높았다.

1. 서 론

오늘날 IT 발전이 화상전화, 영상회의, 개인용 비디오 레코더 등 멀티미디어 시스템에 집중되면서 그 운영체제로 리눅스가 많이 사용되고 있다. 그러나 기존 리눅스 파일 시스템들은 상대적으로 작은 파일의 입출력에 최적화 되어 있어 대용량, 다중 미디어의 융합, 순차적 파일 접근, 빠른 읽기 속도 요구, 불시의 전원 차단이라는 독특한 특징의 개인용 멀티미디어 시스템에는 적합하지 않다. 따라서 본 논문에서는 개인용 멀티미디어 시스템을 위한 새로운 리눅스용 멀티미디어 파일 시스템(이하 PMFS, Personal Multimedia File System)을 구현하였다. 또한 이에 대한 성능 평가를 실시하여 PMFS를 검증하였다.

본 논문의 구성은 2장에서는 관련 연구로 리눅스 파일 시스템의 종류와 멀티미디어 시스템 지원의 문제점, 3장에서는 PMFS의 구현 내용, 4장에서는 PMFS 대한 성능 평가를 실시하며 5장에서는 결과 분석, 6장에서는 본 연구의 결론을 기술하므로 논문을 마치고자 한다.

2. 관련 연구

2.1 리눅스 파일 시스템

오늘날 널리 사용되는 리눅스 파일 시스템은 Ext2, Ext3,JFS,XFS,ReiserFS이 있다. Ext2는 Way Davidson에 의해 디자인 된 리눅스 표준 파일 시스템으로 일반 파일, 디렉토리, 디바이스 파일, 심볼릭 링크의 표준 Unix 파일 타입을 지원한다[1]. Ext3는 Ext2에 저널링을 도입한 Ext2의 확장 파일 시스템이며[2] ReiserFS는 Hans Reiser가 개발한 것으로 작은 파일 의 성능 향상에 초점을 맞추어 디자인 되었다[3]. JFS는 IBM사에 의해 제작된 것으로 e-business파일 서버, 인트라넷과 같은 서버 환경에서의 높은 처리 양을 위해 디자인되었고 [4] SGI에 의해 개발된 XFS는 대용량의 서버를 위해 구현되었으며 완전한 64-bit 파일 시스템으로 9 Exabytes까지 지원 가능하고 이전 파일 시스템에 비해 Bandwith가 월등히 좋다[5].

2.2 개인용 멀티미디어 시스템을 위한 파일 시스템

2.2.1 멀티미디어 파일의 특징

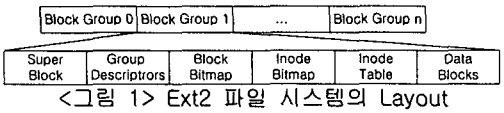
멀티미디어는 텍스트, 그래픽, 이미지, 오디오, 비디오, 애니메이션 등 여러 미디어 데이터 정보가 융합되어 디지털화 된 형태를 말한다[6]. 이 멀티미디어 파일은 대용량, 파일에 대한 순차적 접근, 읽기 연산 중심이라는 독특한 특징을 가진다. 대표적인 멀티미디어 파일인 영화, 에니메이션, 음악 파일의 경우 수 메가에서 수 기가의 크기를 가지며 대개는 한 번 디스크에 기록 된 후, 계속해서 읽혀 재생된다.

2.2.2 개인용 멀티미디어 시스템을 위한 기존 파일 시스템의 문제점

현재 가장 많이 사용되고 있는 리눅스 파일 시스템은 Ext2, Ext3이며 여기서는 Ext2의 문제점만을 기술한다. 유닉스 파일 시스템에 많은 영향을 받은 Ext2는 텍스트 기반의 비교적 작은 파일의 처리 성능 향상을 중심으로 디자인되었다. 이로 인해 개인용 멀티미디어 시스템에 Ext2를 적용하는데 심각한 문제점이 발생하게 된다[7].

1) 디스크 Layout으로 인한 문제

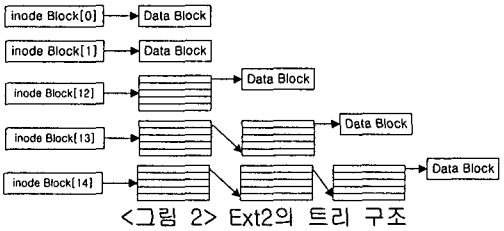
Ext2는 <그림 1>과 같이 블록그룹 메커니즘을 사용하는데 만일 블록 그룹의 사이즈 보다 큰 멀티미디어 파일을 다룰 경우 데이터블록은 여러 개의 다른 블록그룹으로 나누어져 저장되며 디스크 탐색 오버헤드를 발생시킨다.



<그림 1> Ext2 파일 시스템의 Layout

2) 데이터의 트리 구조 관리로 인한 문제

Ext2는 <그림 2>와 같은 트리 구조로 모든 객체를 관리한다. 그러나 순차적 읽기 쓰기를 수행하는 멀티미디어 파일의 경우 이러한 구조는 불필요한 디스크 검색 오버헤드를 발생시킨다.



<그림 2> Ext2의 트리 구조

3) 버퍼캐쉬 사용으로 인한 문제

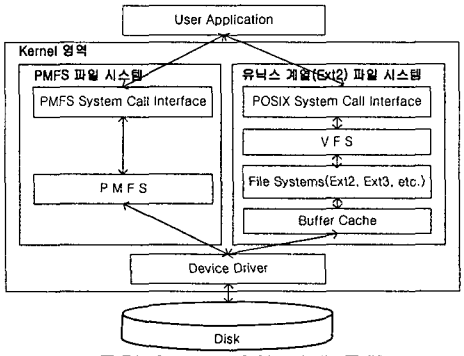
파일의 재사용 가능성을 고려하여 버퍼캐쉬를 사용한다. 하지만 단일 프로그램이 순차적으로 파일을 읽을 경우 버퍼캐쉬는 오히려 메모리 낭비와 많은 스왑핑으로 인한 시스템의 성능 저하를 가져온다. 또, 개인용 멀티미디어 시스템은 불시에 전원을 끄는 사례가 빈번하게 발생하므로 버퍼캐쉬를 사용할 경우 데이터의 무결성이 보장되지 못한다.

3. 개인용 멀티미디어 시스템을 위한 PMFS 구현

본 연구에서는 위와 같은 문제들을 해결한 새 리눅스 파일 시스템인 PMFS 구현하였다.

3.1 PMFS의 전체 구조

PMFS는 <그림 3>과 같이 자체 시스템 콜 인터페이스를 제공하며 응용 프로그램은 이를 통해 PMFS를 디스크내의 파일에 접근하게 된다. 이때 PMFS는 일반 Ext2와 달리 버퍼캐쉬를 사용하지 않는다.



<그림 3> PMFS의 전체 구조

3.2 디스크 Layout

PMFS는 멀티미디어 특성을 반영하여 디자인되었으며 하나의 슈퍼블록과 이를 위한 백업 그리고 다수의 Chunk 블록들로 구성된다. 슈퍼블록은 파일 시스템 아이디, Chunk 사이즈, 처음 Chunk 위치, 전체 Chunk 수, 할당된 Chunk 수, 빈 Chunk 수, '채널 정보'가 있으며 채널 정보는 미디어들에 대한 타입 정보, 채널에 할당된 Chunk수 등이 기록된다. 하나의 Chunk는 16MB 크기이며 Chunk Head와 실제 데이터 블록영역으로 구분된다. 실제 각 Chunk의 데이터는 두 번째 디스크 섹터에서 시작하며 한 Chunk에 기록할 수 있는 최대 데이터 양은 16MB - 512B = 16,776,704 바이트이다.

3.3 데이터의 순차 관리

PMFS는 순차적 연산을 수행하는 멀티미디어 파일의 특성을 반영하여 데이터를 디스크에 순차적으로 기록한다. 이는 파일의 크기가 아무리 크다 하더라도 블록그룹을 사용할 때와 같이 데이터가 흩어지는 것이 없으며 일정한 방향으로 데이터를 읽게 되어 헤더의 이동시간은 현저하게 줄어들게 된다.

3.4 PMFS의 메타 데이터 구조

PMFS는 많은 데이터를 요구 할 뿐 아니라 디스크 탐색 오버헤드를 발생시키는 아이노드의 트리 구조를 사용하지 않는다. 대신 스트림 인덱스 채널, Search Log 인덱스 채널을 구현하였고 응용 프로그램에서 이 채널에 대한 인덱싱을 수행하도록 하였다. Search Log 인덱스 채널은 데이터의 유무를 기록하는 Log로서 스트림 인덱스 채널과 같이 주기적으로 해당 채널에 기록된다.

3.5 데이터 무결성 보장

파일 시스템은 불시에 전원 차단이 된 후에도 데이터의 무결성을 보장해야 한다. 이를 위해 PMFS는 버퍼 캐시를 사용하지 않고 바로 디스크에 접근하게 디자인되었고 전원의 불시 차단에 대해 높은 데이터 무결성을 보장해 준다.

3.6 채널 데이터 구조

기존 리눅스 파일 시스템들은 여러 데이터를 아이노드 테이블(inode table)을 가지고 관리했다. 그러나 이러한 구조는 멀티미디어 파일과 같은 다중 미디어를 관리하는데 있어서 적합하지 않다. 이를 위해 PMFS는 비디오, 오디오, 이미지, 텍스트 등 각 미디어에 따라 파일 대신 채널 별로 데이터를 구분하여 관리한다. 이러한 구조는 채널에 대한 접근은 곧 미디어에 대한 개별적 접근을 하게 하므로 응용 프로그램이 미디어에 대한 프로그램 구현을 편리하게 한다.

3.7 대용량의 블록 Size 구조

멀티미디어 파일 시스템은 다중 미디어 데이터 프레임들이 한 블록에 여러 채널 별로 유지되어야 하기 때문에 블록 크기가 충분히 커야 한다. PMFS는 이를 위해 하나의 블록을 16MB chunk 단위로 나누어 관리한다.

3.8 Read-ahead
멀티미디어 시스템에서는 쓰는 속도 보다 읽기 속도가 더 중요하다. PMFS는 이를 위해 Read-ahead를 기존 파일 시스템 보다 자주, 많이 수행함으로 읽기 속도를 증가 시켰다.

4. MFS의 성능 평가 실험

본 실험에서는 Ext2, Ext3를 PMFS와의 성능 평가 비교 대상으로 선정하였으며 성능 평가 기준은 시험 프로그램에 대한 응답 시간으로 하였다.

4.1 실험 환경 및 Work-Load 설정
성능 평가를 실시할 개인용 시스템 환경은 MIPS 300MHz, 메모리는 32MB, 리눅스 커널 2.4.18 이다. Work-Load는 메모리 상에 존재하는 32KB의 일정한 데이터를 10개의 쓰레드가 각각 70MB씩, 총 700MB를 해당 파일 시스템에 쓰기, 읽기를 수행 하게된다.

4.2 테스트 프로그램 구현
테스트 프로그램은 멀티미디어 파일에 대한 성향을 잘 반영하고 있어야 한다. 개인용 멀티미디어 시스템의 경우 주로 순차적인 쓰기, 읽기 연산이 대부분이기 때문에 테스트 프로그램도 이와 같은 기능을 갖도록 구현되었다. 버퍼 크기(32KB)의 데이터를 각각의 파일 시스템에 쓰고, 읽는다. 단, PMFS의 경우에는 파일이 아닌 해당 채널에 접근한다.

4.3 측정 방법
본 실험에서 사용한 시간 측정 전략으로는 마이크로초 단위까지 측정이 가능한 *gettimeofday()* 함수를 사용하여 디스크 I/O를 수행하는 쓰레드의 시작과 끝 시간의 차를 구하였다. 또한 버퍼캐쉬의 효과를 없애기 위해 매 실험하기 전 시스템을 재부팅 하였다.

5. 결과 분석 및 평가

성능 측정 실험 결과는 <표 5>와 같다.

<표 5>성능 평가 결과

Time / FS	Response Time		
	Write	Read	Write/Read
Ext2	101,890,000	157,880,000	132,920,000
Ext3	101,670,000	157,510,000	138,900,000
PMFS	109,790,000	59,500,000	89,280,000

쓰기의 경우 PMFS가 Ext2, Ext3보다 각각 7.75%, 7.99%가 느린 것으로 나타났다. 쓰기 성능이 떨어진 이유는 PMFS가 버퍼캐쉬를 사용하지 않고 데이터를 바로 디스크에 기록하기 때문이다. 그러나 비록 쓰기 속도는 느려졌으나 시스템의 안정성은 월등히 증가하였다. 반면 읽기의 경우 PMFS가 Ext2, Ext3보다 각각 62.31%, 62.22% 높게 나타났다. 쓰기와 읽기를 동시에 수행했을 때에도 PMFS가 Ext2, Ext3 보다 각각 32.83%, 35.72% 높게 나타났다. 이처럼 읽기 속도가 빠른 이유는 PMFS가 훨씬 긴 순차 블록을 엑세스하고 Read-ahead를 보다 효과적으로 하기 때문이다.

6. 결론

본 연구에서는 개인용 멀티미디어 파일 시스템을 지원하는 새로운 파일 시스템인 PMFS를 구현하였다.
PMFS의 성능을 측정한 결과 쓰기 속도는 Ext2, Ext3 보다 느린 것으로 나타났다. 그러나 읽기, 쓰기/읽기 수행에 있어서는 월등히 높은, 매우 우수한 성능을 나타내었으며 시스템의 안정성 역시 증가하였다.
본 논문에서는 이 같은 실험을 통해 PMFS가 개인용 멀티미디어 시스템에 매우 적합하며 안정성 및 성능 또한 우수함을 보였다.

위 결과는 개인용 멀티미디어 시스템의 경우 PMFS를 사용하는 것이 올바르며 이는 시스템의 성능과 안정성을 크게 높일 수 있음을 말해 준다.

참고 문헌

[1] Remy Card, Theodore Ts'o, and Stephen Tweedie ,"Design and Implementation of the Second Extended Filesystem", First Dutch International Symposium on Linux, ISBN 90-367-0385-9, 1994
[2] Dr. Stephen Tweedie, "EXT3, Journaling Filesystem", the Ottawa Linux Symposium, html document of OLS, 20 July, 2000
[3] Daniel Robbins, "고급 파일시스템 개발자 가이드, 저널링과 ReiserFS"
http://www-903.ibm.com/developerworks/kr/linux/library/l-fs.html#7
[4] Journaled File System Technology for Linux, html document of IBM
http://oss.software.ibm.com/jfs/
[5] Adam Sweeney, Doug Doucette, Wei Hu, Curtis Anderson, Mike Nishimoto, and Geoff Peck, "Scalability in the XFS File System", 1996 USENIX conference, html document of SGI, 1996
http://oss.sgi.com/projects/xfs/papers/xfs_usenix/#fn1
[6] 멀티미디어와 정보화사회, 한국과학기술진흥재단
http://211.40.179.13/book_file/ke28/ke028-index.htm
[7] 원유집,박진연, "정보가전용 멀티미디어 파일 시스템 기술"