스트리밍 서버를 위한 멀티미디어 파일 시스템 최적화

(Optimizing a Multimedia File System for Streaming Severs)

박 진 연 † 김 두 한 †† 원 유 집 ††† 류 연 승 ††††
(Jinyoun Park) (Doohan Kim) (Youjip Won) (Yeonseung Ryu)

요 약 전통적인 텍스트기반의 입출력과 달리, 멀티미디어 자료의 재생은 저장장치에서 일정한 대역폭을 보장을 필요로 한다. 대용량 서버에서 가장 많이 사용되는 유닉스 계열의 파일 시스템은 대역폭 보장이 필요하고, 순차적 접근특성을 가지고 있는 멀티미디어 자료 재생에 많은 개선의 여지를 가지고 있는 것이 사실이다. 본 논문에서는 유닉스 계열 파일 시스템의 단점을 극복하고 동영상 실시간 재생에 적합한 파일 시스템 구조를 연구 개발한 결과를 기술하고자 한다. 본 파일 시스템은 세 가지 설계 목표를 가지고 개발되었다. 첫 번째는 순차적 접근 부하에 대한 효과적 지원이다. 순차적 접근 특성을 효과적으로 지원하기 위해서는 트리기반의 데이타 블록구성이 아닌 연결리스트 기반의 데이타 블록 구성방식을 채택한다. 두 번째는 파일 단편화 방지이다. 순차적 읽기에 있어서 과도한 디스크 탐색(Seek) 작업은 디스크의 효율성에 부정적인 영향을 미친다. 이를 효과적으로 극복하기 위하여 파일은 데이타 유닛 그룹(Data Unit Group)이라 불리는 단위의 집합으로 구성되며, 데이타 유닛 그룹은 연결리스트를 이용하여 구성되었다. 세 번째는 논리적 유닛에 기반한 파일 접근방식의 지원이다. 멀티미디어 파일은 비디오 프레임이나 오디오 샘플들의 집합으로 구성되어 있으며, 이들은 각기 다른 크기를 가지고 있다. 따라서, 이들에 대한 임의접근(Random Access)를 지원하기 위해서 각 논리적 유닛의 위치를 나타내는 인덱스를 파일 메타구조에 포함하였다. 이 부분은 트리구조를 이용하여 구성한 것이다. 실험을 통해서 파일 시스템의 성능을 리눅스 기반의 EXT2 파일 시스템, SGI 사에서 개발한 XFS 파일 시스템과 비교하였으며, 본 논문에서 제안하는 파일 시스템이 기존 리눅스 기반의 EXT2 그리고 SGI 사의 XFS 파일 시스템 보다 더 우수한 성능을 나타내는 것으로 입증되었다.

키워드 : 멀티미디어, 스트리밍, 파일 시스템, UFS, 디스크 스케줄링

Abstract In this paper, we describe our experience in the design and implementation of the SMART file system to handle multimedia workload. Our work has three design objectives: (i) efficient support for sequential workload, (ii) avoiding disk fragmentation, (iii) logical unit based file access. To achieve these three objectives, we develop a file system where a file consists of linked list of Data Unit Group. Instead of tree like structure of the legacy Unix file system, we use single level file structure. Our file system can also access the file based upon the logical unit which can be video frame or audio samples. Data Unit Group is a group of logical data units which is allocated continuous disk blocks. At the beginning of each Data Unit Group, there exists an index array. Each index points to the beginning of logical data units, e.g. frames in the Data Unit Group. This index array enables the random access and sequencial access of semantic data units. SMART file system is elaborately tailored to effectively support multimedia workload. We perform physical experiments and compare the performance of SMART file system with EXT2 file system and SGI XFS file system. In this experiment, SMART file system exhibits superior performance under streaming workload.

Key words : Multimedia, Streaming, File System, UFS, Scheduling

· 본 연구는 2003~2006년 한국과학재단의 젊은과학자연구활동지원연구(R08-2003-000-11104-0)지원을 받아 수행되었습니다.

† 비 회 원 : 지티이퓨처텔
jypark@futuretel.co.kr
†† 비 회 원 : 한양대학교 전자통신전파공학과
lissom33@ece.hanyang.ac.kr
††† 정 회 원 : 한양대학교 공과대학 전자전기컴퓨터 공학부 교수
yjwon@ece.hanyang.ac.kr
†††† 비 회 원 : 명지대학교 컴퓨터소프트웨어학과
ysryu@mju.ac.kr

논문접수 : 2001년 10월 8일
심사완료 : 2004년 5월 13일

1. 서 론

1.1 연구 동기

최근 컴퓨터 통신망 속도, CPU 속도, 디스크 등 저장장치 성능 등의 급격한 신장에 힘입어 원격 교육, 주문형 비디오 등 인터넷을 통한 양방향 멀티미디어 서비스가 빠른 속도로 대중화되고 있다. 멀티미디어 서버를 설계하는데 있어서 핵심적인 사항은 가능한 적은 양의 각종 시스템 자원, 예를 들어 주기억장치, CPU 클럭, 디스크 대역폭 등을 사용하면서, 비교적 많은 개수의 스트리밍 세션(Streaming Session)을 지원하는 것이다. 이러한 자원 이용의 효율성은 멀티미디어 서비스의 단가와도 밀접한 영향을 가진다. ftp, http, telnet 등과 같은 텍스트 기반 데이타의 전송 서비스와 달리 멀티미디어 서비스는 시스템의 자원 요구에 있어서 특별한 성질을 가지고 있다. 각각의 패킷 혹은 프레임은 미리 정의된 시간 안에 목적지에 도달하여야 하며, 이 조건이 만족되지 않을 경우, 목적지에 도착했음에도 불구하고, 정보의 의미 자체를 잃을 수 있는 것이다. 이러한 자료 입출력의 실시간성을 보장하기 위해서 데이타 블록을 전송하는데 필요한 각종 시스템 자원들을 미리 예약해 놓아야 한다. 따라서, 멀티미디어 서비스의 효과적 지원을 위해서는 일정량의 시스템 자원들이 서비스에 할당되어야 한다. 관련된 시스템 자원으로서는 디스크 저장장치, 네트웍 카드, 주기억장치, CPU 등이다. 멀티미디어 자료를 효과적으로 정해진 시간까지 목적지에 전달하기 위해서는 위에서 언급한 자원들을 효과적으로 사용할 수 있어야 한다. 다수의 사용자에게 멀티미디어 서비스를 제공하는 서버 입장에서는 각각의 서비스 세션이 필요로 하는 자원요구량을 정교하게 모델링 하여, 단위 서버가 다룰 수 있는 세션의 개수를 극대화 하는 것이 관건이라 하겠다. 본 논문에서는 멀티미디어 서비스를 제공하는 데 관련된 다수의 자원 중에서 디스크 자원을 효율적으로 사용하는데 필요한 기법을 집중적으로 다루고자 한다.

멀티미디어 세션(Session)에 대해서 데이타 블록을 시간에 맞추어 전송하기 위해서는 일정한 데이타 전송 용량이 보장되어야 한다. 서버의 입장에서는 디스크 저장장치로부터 자료를 읽는 전송대역폭과 네트웍 카드의 전송대역폭, CPU 싸이클 등이 전송 용량을 보장하는 자원에 해당된다. 추상적인 자료구조인 "파일"로부터 디스크에 저장되는 물리적인 데이타 블록들을 사상시키는 역할을 하는 "파일 시스템"은 멀티미디어 스트리밍 서비스를 효과적으로 수행하는 데 있어서 핵심적인 역할을 한다. 만약 프레임 데이타나 특별한 데이타 블록을 디스크 저장장치로부터 보다 빠른 속도로 가져올 수 있다면, 멀티미디어 서버는 미디어 간의 동기화를 위해 사용되는 버퍼의 양을 적게 사용 할 수 있고, 더 많은 멀티미디어 스트림을 서비스 할 수 있다. 디스크 입출력 지연 시간(I/O Latency)을 유발하는 주요인자는 디스크 탐색 오버헤드(Overhead)이다. 일반적인 랜덤 디스크 입출력의 경우 80%정도의 시간이 탐색 오버헤드이다. 따라서, 멀티미디어 파일 시스템을 디자인하는 데 있어서, 데이타 검색에 사용되는 디스크 탐색(Disk Seek) 오버헤드를 최소화시키는 것이 핵심 관건이라 하겠다.

멀티미디어 자료는 여러 가지 다양한 환경에서 사용된다. 자료의 편집(Editing), 자료의 검색, 실시간 재생 등이 그것이다. 데이타가 어떤 목적으로 사용되느냐에 따라 자료를 저장하는 데 있어서의 접근하는 방식이 달라져야 한다. 동영상 검색이 주가 되는 시스템에서는 해당 자료에 대한 인덱스구조와 인덱스 방식, 키 프레임을 추출하는 방법 등이 매우 정교하게 설계되어야 한다. 멀티미디어 동영상의 편집은 동영상, 음성, 텍스트 등을 통합하고 동기화 정보를 추가하여 하나의 파일로 만드는 작업이 주를 이루며 교육용 자료를 제작하는 데 많이 응용된다. 이러한 경우에는 각 미디어에 속한 자료를 손쉽게 추출, 변경 저장할 수 있는 저장기법이 필요하다. 동영상 실시간 재생의 경우에는 주로 파일을 처음부터 끝까지 순차적으로 읽는 동작이므로 이를 효과적으로 지원할 수 있는 데이타 블록 배치 방법, 파일구조 등이 필요하다. 검색이나 편집 작업에서는 멀티미디어 파일을 다루는데 있어서는 주어진 데이타를 신속하게 입출력하는 실시간성이 그리 강조되지 않는다. 그러나 동영상 실시간 재생을 위한 시스템의 경우 자료를 실시간에 입출력하는 것이 관건이므로 해당 파일 시스템이 동영상 실시간 재생의 부하에 특화되도록 설계되어야 한다. 본 논문에서는 멀티미디어 실시간 재생 환경에 최적화된 새로운 파일 시스템을 연구 개발하고, 멀티미디어 파일의 실시간 재생 환경하에서 성능을 측정하였다.

1.2 관련 연구

최근까지 멀티미디어 서버에 관한 많은 연구가 진행되어 왔다[1-3]. 멀티미디어 스트림은 데이타 블록 흐름의 연속성을 보장해 주기위해 디스크 대역폭이 보장되어야 한다. 초기의 주요 주제는 단일 디스크로부터 데이타 블록을 읽어들일 때 본래의 멀티미디어 데이타의 실시간 재생 속도에 맞추어 블록을 읽는 기법의 개발이었다. 단일 디스크 시스템과 관련된 여러 가지 연구와 별도로, 멀티미디어 파일 서버를 위한 디스크 서브 시스템에 관해서도 많은 연구가 진행되었다[3,4]. Rompogiannakis외[5]는 디스크 스케줄링 알고리즘을 디스크 배열(Array)까지 확장시켰다. 일반적인 파일 시스템의 경우 디스크에서 버퍼 캐쉬로 탑재된 페이지는 LRU등

의 방식을 통하여 교체된다. 그러나 멀티미디어 서버의 경우에는 탑재된 페이지가 얼마나 자주 요청되는 비디오 파일에 속한 페이지냐에 따라서 교체의 우선순위를 결정하는 것이 가장 이상적이다. 이를 이용하면 버퍼 캐쉬의 페이지 미스 확률을 효과적으로 줄일 수 있으며, 따라서 디스크 입출력 부하를 줄일 수 있다. 스트리밍 서버에서 자료를 디스크로부터 읽어들이는 오버헤드를 줄이기 위하여 많은 기법이 제안되었는데, 그 중에 하나가 탑재된 데이타 블록을 다시 사용하는 것이다[4,6].

하나의 파일 시스템 파티션에서 텍스트, 멀티미디어, 이미지등 다양한 형식의 미디어를 효과적으로 지원하는 방식에 대한 연구들도 진행되었다[7]. Shenoy외[8]는 미디어 타입별로 파일을 분리하여 각각의 파일 시스템 파티션에 저장하는 것보다 한 개의 통합된 파일 시스템 프레임 워크에서 여러 종류의 미디어 I/O 요청을 다루는 것이 훨씬 더 효율적이라는 연구결과를 제시하고 있다. Rompogiannakis외[5,9]는 멀티미디어 재생을 하면서, 동시에 산발적으로 발생하는 재생과 상관없는 I/O 요청을 효과적으로 다룰 수 있는 디스크 스케줄링 알고리즘을 개발하였다.

최근 멀티미디어 데이타 처리를 위하여 최적화된 파일 시스템 프로토타입(Prototype)이 제안되었다[10,11]. Niranjan외[12]는 새로운 미리 읽기 방법(Prefetching), 상태 기반 캐싱(State-Based Caching), 우선순위 디스크 스케줄링, 멀티 스트림간의 동기화를 구현하여 양방향 재생의 성능을 높였다. VCR과 같은 재생 모드(빠른 재생, 뒤로 재생)의 응답 시간을 개선하였으며, 여러 개의 멀티미디어 스트림들 간에 동기가 어긋나는 것을 최소화 하도록 설계하였다.

Minorca 멀티미디어 파일 시스템[13]은 (1) MOSA라는 디스크 레이아웃과 데이타 배치 기술을 제안했고, 시간적인 특성을 갖는 멀티미디어 데이타와 그렇지 않은 데이타들을 한 개의 파일 시스템에서 동시에 지원해 줄 수 있게 하며 (2) I/O 요청 큐(Queue)를 최적화시키기 위해서 새로운 미리 읽기 방법을 제안하고 있다. 이러한 기술을 사용하는 이유는 디스크 접근의 지역성(Locality)를 높이고 디스크 탐색 오버헤드를 줄이는데 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존 유닉스 파일 시스템의 구조와 문제점을 제시한다. 3장에서는 본 논문에서 제안하는 멀티미디어 파일 시스템의 디스크 구성 및 구조에 대해서 설명하고, 4장에서 멀티미디어 스트리밍에 최적화된 리눅스 기반 파일 시스템의 성능 측정 결과를 보여준다. 5장에서는 결론 및 향후 연구과제를 언급한다.

2. 유닉스 파일 시스템

유닉스 파일시스템(UFS)는 파일 시스템 분야에서 큰 축을 이루는 훌륭한 업적이다. 본 논문이 제안하는 파일 시스템을 다루기에 앞서 유닉스 파일 시스템을 소개하기로 한다.

2.1 유닉스 파일 시스템 구조

유닉스 파일 시스템은 파일 시스템을 관리하기 위한 파일 시스템 메타 정보와 실제 데이타를 분리하여 관리/저장하고 있다. 파일에 관련된 정보, (예를 들어 소유자, 생성일, 접근 권한, 크기 등)를 저장하고 있는 자료구조를 inode라고 한다. 리눅스 파일 시스템에서 가장 널리 쓰이고 있는 EXT2 파일 시스템의 inode 구조는 파일의 관리를 위한 파일 메타 정보(파일의 모드, 파일 소유자의 ID, 파일 크기 등)와 데이타 레퍼런스로 이루어져 있다. 데이타 레퍼런스는 12개의 직접 레퍼런스와 1개씩의 1차 간접 레퍼런스, 2차 간접 레퍼런스, 3차 간접 레퍼런스로 구성되어 있다. 레퍼런스는 4바이트 포인터이다. 데이타를 저장하는 최소 단위를 데이타 블록이라 하며, 리눅스의 EXT2 파일 시스템의 경우 파일 시스템을 초기화 할 때 그 크기를 1K바이트, 2K바이트, 혹은 4K바이트로 지정할 수 있다. EXT2 파일 시스템에서 데이타 블록이 4K바이트라고 가정하면, 직접 레퍼런스를 사용해서 나타낼 수 있는 최대 파일 사이즈는 48K바이트이다(12 직접 레퍼런스*4K바이트 데이타 블록). 1개의 4K바이트 블록은 1024개의 레퍼런스를 가지게 되고 1차 간접 레퍼런스까지 사용하면 4M바이트의 데이타까지 저장할 수 있다. 2차 간접 레퍼런스까지 사용하면 1개의 파일에서 4G바이트까지 저장할 수 있다.

2.2 유닉스 시스템의 파일 배치 방법

리눅스, 솔라리스, NetBSD등 현재의 유닉스 호환 운영체제 파일 시스템 대부분이 데이타 배치를 최적화시키기 위해 여러가지 기법을 사용하고 있다. 이러한 기법의 궁극적인 목표는 데이타 블록을 서로 연속적으로 배치시키거나, 가능한 한 가깝게 배치하여 입출력 지연 시간(I/O Latency)을 최소화하는 데 있다.

EXT2 파일 시스템은 블록 그룹(Block Group) 개념을 사용하여 파일의 데이타 블록과 inode, 디렉토리 할당을 정교하게 관리할 수 있도록 제공하고 있다[9]. 블록 그룹은 연속적인 실린더들의 모임이다. 파일 시스템을 초기화 할 때(Disk Format) 블록 그룹 사이즈가 정해지며, 다시 포맷하기 전까지는 변하지 않는다. 하나의 EXT2 파일 시스템은 여러개의 블록 그룹으로 구성이 되어있으며 각각의 블록 그룹은 파일 시스템의 견고성(Robustness)을 위해 복사된 슈퍼 블록(Super Block)과 그룹 디스크립터, 블록 비트맵, inode 비트맵, inode

테이블과 데이타 블록으로 이루어져 있다. 블록 그룹 개념의 사용으로 인해 파일 시스템은 상대적으로 좀더 가까운 실린더(Cylinder) 위치에 파일의 데이타 블록을 놓을 수 있다. 하지만, 블록 그룹 기반 정책에서도 여전히 블록 그룹의 크기(EXT2 파일 시스템의 최대 블록 그룹 크기 = 128M바이트)를 넘는 파일은 여러 개의 다른 블록 그룹에 나누어서 저장되고 이것은 디스크 탐색에 있어 과도한 오버헤드를 유발하게 된다.

이제까지 소개한 유닉스 파일 시스템이 텍스트 기반 데이타를 위한 파일 시스템으로는 어느 여타 파일 시스템보다 기술 발전에 많은 공헌을 한 것은 누구도 부정할 수 없는 사실이다. 유닉스 파일 시스템은 작은 크기의 파일부터 매우 큰 파일까지 다양한 크기의 파일을 효율적으로 다룰 수 있게 설계되어 있다. 그러나 이제까지 기술한 바와 같이 기존의 유닉스 파일 시스템은 스트리밍 작업에 있어서는 많은 개선의 여지를 남기고 있는 것이 사실이다.

3. 멀티미디어 파일 시스템 설계

3.1 유닉스 파일 시스템과 멀티미디어 파일

멀티미디어 자료의 경우 영화 한편이 수백 M바이트에서 수 G바이트까지의 디스크 용량을 필요로 한다. MPEG-1으로 압축된 1.5Mbits/sec 비디오의 경우 90분짜리 영화 1편에 약 1G바이트의 저장 공간을 필요로 한다. 이러한 대용량의 파일을 저장하기 위해서는 EXT2 파일 시스템의 경우 inode에서 2차 간접 혹은 3차 간접 레퍼런스가 사용된다. 예를 들어 1G바이트 용량의 파일에서 마지막 데이타 블록을 읽기 위해서는 그 전에 추가로 3개의 블록에 접근해야만 한다. 이해를 돕기 위하여 그림 1을 보도록 하자. 그림 1에서 2차 간접 레퍼런스에 의해 지정된 데이타 블록을 읽기 위해서는 ①에서 inode 테이블 블록과 ②에서 2개의 간접 블록을 먼저 읽어야 한다.

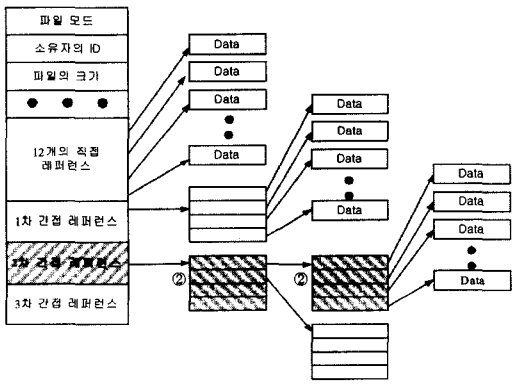


그림 1 EXT2 파일 시스템의 inode 구조

기존의 유닉스 파일 시스템, 특히 EXT2 파일 시스템은 스트리밍 환경에서 다음과 같은 단점을 가지고 있다. 첫 번째 단점으로 비효율적인 순차적 접근 방식이다. EXT2 파일 시스템에서의 트리(Tree)구조 기반 inode 구성은 다양한 크기의 파일들을 매우 효율적으로 다룰 수 있다. 반면에, 멀티미디어 파일과 같은 대용량의 파일을 순차적으로 접근하는 경우에는 데이타 블록을 접근하기 위해서 반드시 한 개의 inode 블록과 여러 개의 간접 레퍼런스 블록을 읽어야 하므로, 추가의 오버헤드를 유발하게 된다. 이러한 inode 블록과 포인터 블록이 버퍼 캐쉬(Buffer Cache)에 이미 로드(Load)가 되었더라도, 메모리 접근은 CPU 싸이클(Cycle)의 상당 부분을 소비하게 된다. 두 번째로, 과도한 디스크 탐색 시간을 들 수 있다. 포인터 블록과 데이타 블록은 디스크 플래터상에 연속적으로 저장되어 있지 않기 때문에, 디스크 헤드는 포인터 블록과 데이타 블록을 읽기 위해서 디스크 플래터(Platter) 사이를 움직여야만 한다. 이러한 디스크 탐색 시간은 디스크 지연의 상당 부분을 차지하고 있다. 마지막으로 디스크 단편화(Disk Fragmentation)로 인한 오버헤드가 있다. 여러번의 파일 생성과 삭제 과정에서 파일 단편화가 발생하고, 이것은 디스크 탐색 시간을 증가시킨다. 우리는 편향 트리 형태의 파일구조[9]와 연결리스트 기반의 파일구조의 장단점을 분석하고 이를 효과적으로 결합하여 새로운 형태의 파일 시스템을 개발한다. 개발 목표중의 하나는 바이트 기반의 접근 방식뿐 아니라 논리적 단위기반의 접근방식까지도 지원하는 것이다[7,14].

3.2 SMART 파일 시스템 설계

우리는 앞에서 지적한 유닉스 기반 파일 시스템의 문제점을 효과적으로 극복하면서 멀티미디어 스트리밍을 효율적으로 지원해 주는 파일 시스템을 개발하였다. SMART 파일 시스템은 데이타 블록이 다계층 트리 형태로 조직되는 것을 피하면서, 다양한 크기의 파일을 지원할 수 있도록 설계되었으며, 디스크 탐색 시간을 최소화하는 데 중점을 두었다.

그림 2는 본 논문에서 제안하는 SMART 파일 시스템의 전체 구조를 보여주고 있다. SMART 파일 시스템은 슈퍼 블록(Super Block), inode, 데이타 유닛 그룹

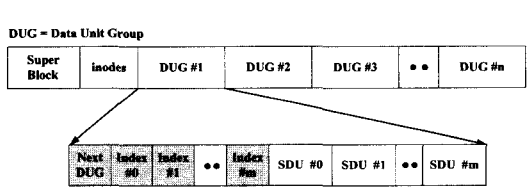


그림 2 SMART 파일 시스템 구조

(Data Unit Group)으로 구성되어 있으며, 데이타 유닛 그룹은 다음 데이타 유닛 그룹을 가리키는 포인터(NextDUG), SDU들의 위치를 저장하고 있는 인덱스(Index), 그리고 SDU의 집합으로 구성되어 있다.

SMART 파일 시스템은 저장 단위를 결정하는 데 있어서, 기존 유닉스 파일 시스템에서 사용되는 물리적 크기에 중점을 둔 저장 단위가 아닌, 자료의 논리적 단위(Semantic Unit)를 저장 단위로 하였다. SDU (Semantic Data Unit)는 비디오 파일에서 1개의 프레임(Frame) 혹은 GOP(Group of Picture)나 오디오 샘플들의 그룹으로 해석될 수 있다. SDU 기반의 데이타 저장 방식, 즉 프레임이나 오디오 샘플 단위로 디스크 플래터(Platter)상에서 연속적으로 저장하는 것은 SDU에 해당하는 여러 개의 데이타 블록이 서로 떨어져서 디스크에 저장되는 것을 방지할 수 있다. 본 논문에서 수행된 실험 결과에 의하면 MPEG-2 방식으로 압축되어 9Mbits/sec의 평균 재생 속도를 가진 비디오 파일에서 I 프레임 크기는 약 150K바이트 정도이다. 그러므로 1개의 I 프레임을 저장하기 위해서는 기존 유닉스 파일 시스템에서 4K바이트 데이타 블록 수십 개로 구성이 되며, 이것은 디스크 상에서 단편화 될 수 있다.

데이타 유닛 그룹은 기존의 파일 시스템의 '블럭'에 해당하며, [14]에서 제안되었다. 데이타 유닛 그룹은 그림 2에서 보는 바와 같이 여러개의 SDU의 집합으로 이루어져있다. 데이타 유닛 그룹의 크기는 포맷(Format) 시에 정해진다. 이와같이 여러개의 SDU를 연속으로 저장하는 큰 단위의 데이타 유닛 그룹의 집합으로 파일을 구성하여 단편화를 방지할 수 있다. 데이타 유닛 그룹 내에서 비디오와 오디오를 분리해서 저장할 수 있으며, 비디오와 오디오 데이타를 갈아가며(Interleaving) 저장할 수 있다.

3.3 데이타 유닛 그룹

그림 3에서 보는 바와 같이 하나의 파일은 하나 이상의 데이타 유닛 그룹의 집합으로 이루어져 있다. 한 파일에 속한 데이타 유닛 그룹들은 연결리스트를 이용하여 연결되어 있으며, 파일 메타 데이타를 포함하고 있는 inode가 파일에 속한 첫 데이타 유닛 그룹의 주소를 저장하고 있다. 데이타 유닛 그룹은 다음 데이타 유닛 그룹을 가리키는 포인터(NextDUG), SDU들의 위치를 저장하고 있는 인덱스(Index)들이 연결리스트로 저장 되어있는 영역, 그리고 SDU 영역으로 구성되어 있다. 따라서, i번째 SDU를 접근하기 위해서는 데이타 유닛 그룹의 앞부분에 존재하는 i번째 포인터를 따라가면 된다. 데이타 유닛 그룹을 구성하는 방법에는 트리구조, 연결리스트, 해쉬방법 등 여러 가지 있을 수 있으며 본 파일 시스템에서는 연결리스트 방법을 사용하였다. 연결리스트로 구성한 이유는 멀티미디어 응용 프로그램은 주로 자료를 순차적으로 접근하기 때문이다.

SMART 파일 시스템에서는 논리 단위, 즉 비디오 프레임이나 오디오 샘플단위의 접근이 가능하며 이것의 크기는 가변적이다. 예를 들어 비디오 프래임의 크기는 프래임의 종류, 즉 I,P,B에 따라 크기가 다르며 MJPEG, MPEG 등의 인코딩 방식에도 영향을 받는다. 논리적인 단위로 데이타를 저장함으로써, 파일 시스템은 멀티미디어 데이타를 프레임 단위 혹은 오디오 샘플 그룹 단위로 조작할 수 있게 된다. 최소 크기의 SDU는 한 개의 데이타 블록의 크기다. SDU의 사이즈가 한 개의 데이타 블록보다 크다면, SDU는 (SDU의 크기)/데이타 블록의 크기 개수만큼의 데이타 블록을 차지하게 된다. 각 프레임의 크기가 가변적인 상황에서 임의의 프레임을 찾기 위해서는 파일을 순차적으로 읽어야 한다. 이러한 문제를 해결하기 위하여 최근에 각광받고 있는 MPEG-4 파일 포맷에서는 각 프레임의 위치 정보를 가지고 있는 배열을 파일에 추가하는 기법을 사용하고 있다. 본 논문에서는 그 정보를 파일 시스템 수준에서 지원하고자 하는 것이다. 데이타 유닛 그룹에서 SDU의 검색은

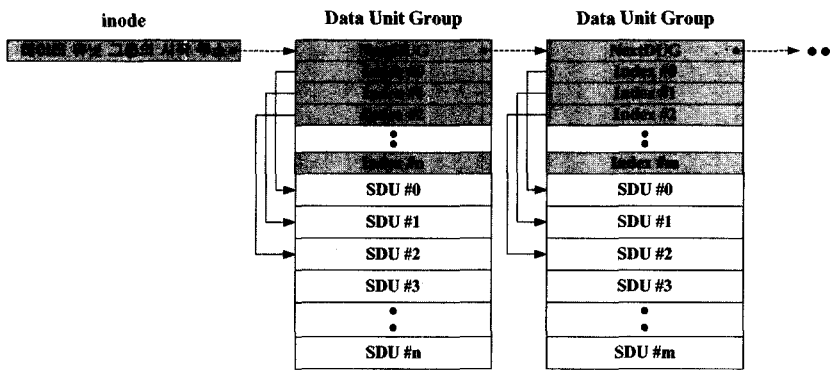


그림 3 inode와 데이타 유닛 그룹(Data Unit Group)

I/O 지연을 상당히 개선할 수 있으며, 이러한 이점은 멀티미디어 스트리밍 응용 프로그램에서 현격한 성능의 차이를 가져온다.

유닉스 파일 시스템에서는 데이타 블록과 포인터 블록이 따로 존재한다. SMART 파일 시스템에서는 데이타 유닛 그룹에 포인터들과 데이타 블록들을 연속적으로 저장하고 있다. 이는 포인터 블록과 데이타 블록을 접근하는 데 발생할 수 있는 디스크 헤드의 움직임을 최대한 줄여보고자 하는 노력에서 시작되었다. 포인터들과 데이타 블록을 인접해서 저장하는 파일 시스템의 접근방식은 BeFS, HERMES 등에서도 사용되고 있다[9,15].

3.4 메타 데이타 구조

슈퍼 블록(Super Block)은 파일 시스템 파티션의 첫 번째 블록에 위치한다. 이 슈퍼 블록은 (1) 데이타 유닛 그룹의 수 (2) 데이타 유닛 그룹의 사이즈 (3) 블록 사이즈 (4) 비어 있는 데이타 유닛 그룹의 수 (5) inode 비트맵 (6) 데이타 유닛 그룹 비트맵을 저장하고 있다. 블록 사이즈는 512바이트로 설정되어 있으며 SDU의 최소 크기는 블록 사이즈와 같다. 새로운 멀티미디어 파일이 생성되면, 'First-Fit' 알고리즘 기반으로 inode 비트맵에서 쓰지 않고 있는 inode를 찾아서 할당한다. 할당된 inode에 데이타가 저장될 경우에도 'First-Fit' 알고리즘 기반으로 데이타 유닛 그룹 비트맵에서 쓰지 않고 있는 데이타 유닛 그룹을 찾아서 할당한다.

SMART 파일 시스템은 멀티미디어 자료만을 저장하기 위한 목적으로 고안되었기 때문에 불필요한 파일 시스템의 메타 정보는 가지고 있지 않다. 따라서, 간단하면서도 멀티미디어에 최적화된 inode를 만들었다. inode에는 (1) 데이타 유닛 그룹의 이름 (2) 데이타 유닛 그룹의 시작 주소 (3) 데이타 유닛 그룹의 생성 시간 (4) SDU의 개수 (5) 인코딩 방법 등에 관한 정보를 저장한다. 데이타 유닛 그룹의 이름은 기존 파일 시스템의 '파일 이름'에 해당한다. 또한 inode에 저장된 파일의 인코딩 방법을 저장하고 있다. 즉 MPEG-1, MPEG-2, MPEG-4, JPEG등의 인코딩 방법을 저장하여 파일 시스템에서 멀티미디어 파일의 특성을 확인 할 수 있다. 본 파일 시스템은 연결리스트 형태를 가지고 있다. 가장 큰 이유는 유닉스 파일 시스템에서 사용되는 복잡한 계층적 구조를 배재하고 순차적으로 접근하기 위함이다. 그림 3에서 보는 바와 같이 모든 데이타 블록들은 모두 직접 연결 포인터에 의해서 연결되어 있다. 여기에서의 데이타 블록은 논리적 저장 단위이다.

표 1 슈퍼 블록의 구조체 구조

설 명
데이타 유닛 그룹의 수
데이타 유닛 그룹의 사이즈
블록 사이즈
비어 있는 데이타 유닛 그룹의 수
inode 비트맵
데이타 유닛 그룹 비트맵

표 2 inode 구조체의 구조

설 명
데이타 유닛 그룹의 이름
데이타 유닛 그룹의 시작 주소
데이타 유닛 그룹의 생성시간
SDU의 개수
인코딩 방법

SMART 파일 시스템에서는 기존의 read와 write 외에 자료를 논리적인 단위로 저장 및 읽을수 있는 파일 시스템 인터페이스를 제공하고 있다(Swrite: 하나의 SDU를 저장하는 파일 시스템 인터페이스, Sread: 하나의 SDU를 읽는 파일 시스템 인터페이스, SreadN: N개의 SDU를 읽는 파일 시스템 인터페이스). 예를 들어 비디오 데이타를 프레임 별로 저장하는 것이 그 예라고 할 수 있다. 멀티미디어 샘플의 크기는 미디어의 종류(오디오 혹은 비디오), 프레임의 종류(I, P, 혹은 B), 인코딩 방식, 대역폭등에 많은 영향을 받는다. 특별한 정보가 주어지지 않을 경우, 파일 내 임의의 샘플을 찾기 위해서는 파일을 순차적으로 검색해야하고, 이는 매우 비효율적인 작업이다. 이 문제점을 해결하기 위하여 SMART 파일 시스템에서는 데이타 유닛 그룹 시작 부분에 각 샘플의 주소를 테이블 형태로 저장한다. 이를 통해서 파일 시스템은 멀티미디어 데이타를 프레임 단위 혹은 오디오 샘플 그룹 단위로 조작할 수 있게 된다. i번째 SDU에 접근하기 위해서는 데이타 유닛 그룹의 앞부분에 존재하는 i번째 포인터를 따라가면 된다. 이러한 기능은 정보가전용 멀티미디어 기기에서 자주 쓰이는 양방향 다배속 재생(2배속, 3배속)을 지원하기 위한 필수적인 사양이다. 샘플별 임의 접근을 지원하기 위한 또 하나의 접근 방법으로 파일의 일부분에 각 샘플에 대한 위치를 저장하는 것이다. MPEG-4 파일 형식이나 Quicktime 파일 시스템이 이를 지원한다[16].

4. 성능 평가

4.1 실험 환경

본 논문에서는 SMART 파일 시스템의 성능을 실제 실험을 통하여 검증한다. SMART 파일 시스템은 리눅스 운영체제에서 구현되었으며, 파일 시스템의 성능은 멀티미디어 스트리밍 서버에 대응하는 입출력 부하를 발생시켜 실험하였다.

실험은 512M바이트의 주기억장치를 가진 Dual Pen-

tium III(660Mhz) 서버에서 실행되었다. 실험에 사용된 하드디스크는 IBM DPSS-309170이다. 이 디스크의 대역폭은 31MByte/sec - 50MByte/sec이다. 본 실험에서는 세 가지 파일 시스템의 성능을 비교 분석하였다.: (i) SMART 파일 시스템 (ii) EXT2 파일 시스템 그리고 (iii) XFS 파일 시스템[17]. XFS는 SGI사에서 대용량 멀티미디어 서버에 사용되는 것을 목표로 하여 개발된 파일 시스템이다. XFS 파일 시스템은 대용량 파일(수백 G바이트 크기)을 효율적으로 처리하며, 하나의 디렉토리 에 많은 수의 파일들을 효율적으로 생성/삭제/검색 할 수 있도록 설계되었다. XFS는 기존의 파일 시스템에서 크게 개선된 자유공간 관리 기법, inode 구조, 파일 매타 구조, 디렉토리 구조 등을 가지고 있다. 보다 구체적인 기술적 내용에 대해서는 [17]를 참고하기 바란다. 본 실험에서는 SGI사가 리눅스 용으로 개발한 XFS 버전을 사용한다.

우리는 이 실험에서 크게 두 가지 측면을 보고자 한다. 첫째는 데이타 블록의 입출력에 소요되는 지연 시간(I/O Latency)이다. 입출력 지연 시간은 응용프로그램이 인지하는 지연 시간으로서 입출력 명령의 직전과 직후의 시간을 측정하여 그 차이를 입출력 지연 시간으로 정의한다. 두 번째는 데이타 블록의 입출력 시간의 분산이다. 멀티미디어 자료를 읽는데 있어서 각 블록을 읽는데 걸리는 시간이 비교적 일정하게 유지되는 것이 매우 중요하다. 하지만, 편향 다계층 형태를 가진 유닉스 기반의 파일 시스템 구조하에서는 데이타 블록의 위치에 따라 입출력 지연 시간의 차이가 크게 나타날 수 있다. 따라서, 성능 평가부분에서 각 파일 시스템의 성능을 입출력 지연 시간의 분산 측면에서 검증해보고자 한다.

본 실험에서는 두 가지의 핵심 요소를 변화시키면서 실험을 진행한다. 먼저 파일 입출력의 단위, 즉 'I/O 유닛(Unit)'의 크기를 변화하면서 세 개의 파일 시스템, 즉 SMART 파일 시스템, EXT2 파일 시스템, 그리고 XFS 파일 시스템의 성능을 비교하였다. 둘째로 파일 시스템에 걸리는 부하에 따른 응답시간의 변화를 측정하기 위하여 멀티미디어 스트림의 수를 증가시키면서 응답시간을 분석한다. 이 실험은 해당 파일 시스템이 어느 정도의 확장성(Scalability)을 가지고 있는가를 실험하는 것이 목적이다.

유닉스 계열의 입출력 시스템의 경우 디스크 서브 시스템에서 데이타 블록이 먼저 버퍼 캐쉬에 탑재가 되고, 이후 동일한 블록에 대한 입력 요청은 디스크 서브 시스템이 아닌 버퍼 캐쉬에서 서비스된다. 본 실험의 목적은 개발된 파일 시스템이 멀티미디어 자료를 효과적으로 "디스크"에 저장하느냐에 대한 검증이다. 이를 위해서는 보다 정확하게 디스크로부터 자료를 읽는데 걸리는 시간, 즉 I/O 요청의 시작부터 완료까지의 경과 시간(I/O Response Time)을 측정하는 것이 매우 중요하다. 따라서, 버퍼 캐쉬에 의한 데이타 블록 서비스를 배제하여 실험이 진행되었다. 이러한 현상, 즉 버퍼 캐쉬 히트(Hit)를 방지하기 위하여, 버퍼 캐쉬 작동은 정지시켰다. 60개의 30M바이트 크기의 MPEG-1 파일을 각 파일 시스템(SMART, EXT2 그리고 XFS)에서 생성을 하여 실험을 진행하였다.

파일의 단편화는 파일 시스템의 성능을 크게 저하시킬 수 있다. 본 논문에서는 파일을 연속적으로 저장하기 위하여 파일 시스템을 모두 초기화한 후 실험에 사용될 파일들을 생성하였다. 따라서, 각 파일 시스템에서 모든 파일은 연속적으로 저장되었다고 가정한다.

4.2 스트리밍 환경 하에서 I/O 지연 시간

먼저 파일 시스템에서 데이타 블록을 읽어 들이는데 걸리는 평균 I/O 지연 시간을 측정하였다. I/O 지연 시간과 I/O 유닛(Unit)의 크기와의 상관 관계를 알아 보기 위해 3개의 다른 I/O 유닛 크기(32K바이트, 64K바이트, 128K바이트)에서 실험을 수행했다. 그림 4, 5, 6은 각각 I/O 유닛 크기가 32K바이트, 64K바이트, 128K바이트일 때의 실험 결과를 보여주고 있다.

I/O 유닛 크기가 작을 때는, SMART 파일 시스템과 EXT2 파일 시스템, XFS 파일 시스템의 성능은 그다지 큰 차이를 보이지 않고 있다. 그러나, SMART 파일 시스템의 I/O 유닛 크기가 증가함에 따라, 성능 향상이 두드러지게 나타남을 볼 수 있다. 현재 EXT2 파일 시스템의 입출력 단위는 최대 4K바이트 이다. 하나의 120K바이트 프레임을 읽기 위해서 최소한 30개의 I/O 명령이 처리되어야 한다. 따라서, 멀티미디어 응용프로그램과 같이 파일을 순차적으로 읽는 경우에는 입출력 데이타의 단위를 크게 할수록 소요시간이 줄어들게 된다. 물론, 이러한 장점은 Web 서버, OLTP등 랜덤 입출력 성격을 가진 부하에서는 그다지 유리하지 않을 수도 있다. XFS 역시 EXT2 보다는 월등한 성능을 나타내고 있다. 성능의 차이는 한번에 읽는 데이타 블록의 크기가 증가할 수록 두드러진다. SMART 파일 시스템은 EXT2는 물론 XFS보다도 월등한 성능을 나타내고 있다. 입출력 단위를 32K바이트로 설정했을 경우에는 SMART 파일 시스템과 XFS 파일 시스템에서 별 차이가 없었으나 입출력 단위가 64K바이트인 경우, 128K바이트인 경우 SMART 파일 시스템에서의 입출력 응답시간은 XFS에서의 입출력 응답시간의 각기 75%, 50%정도이다.

그림 4에서 제시하고 있는 또한가지 자료는 '확장성'(Scalability)이다. 그림 4의 x축은 입출력 세션의 개수를 나타낸다. 여기에서 주의깊게 보아야 할 것은 세션의 개수가 증가함에 따라, 입출력 응답시간이 어떻게 변

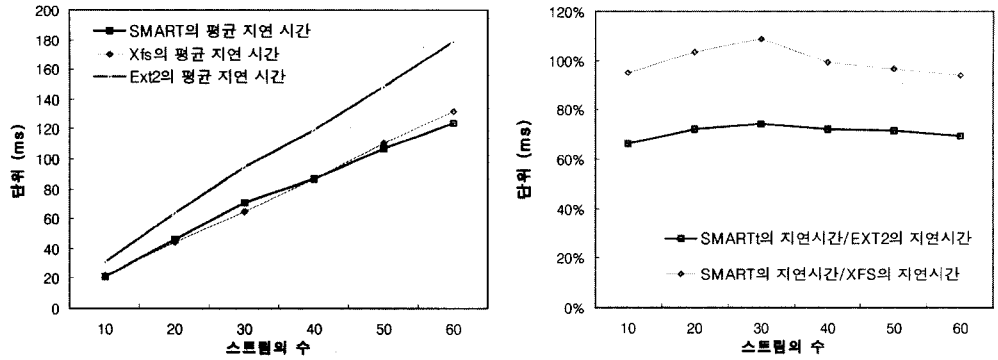


그림 4 32K바이트의 I/O 유닛에서의 읽기 지연 시간

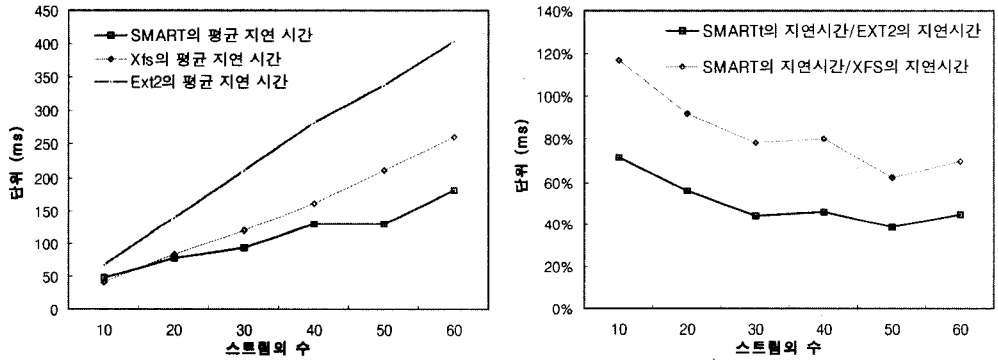


그림 5 64K바이트의 I/O 유닛에서의 읽기 지연 시간

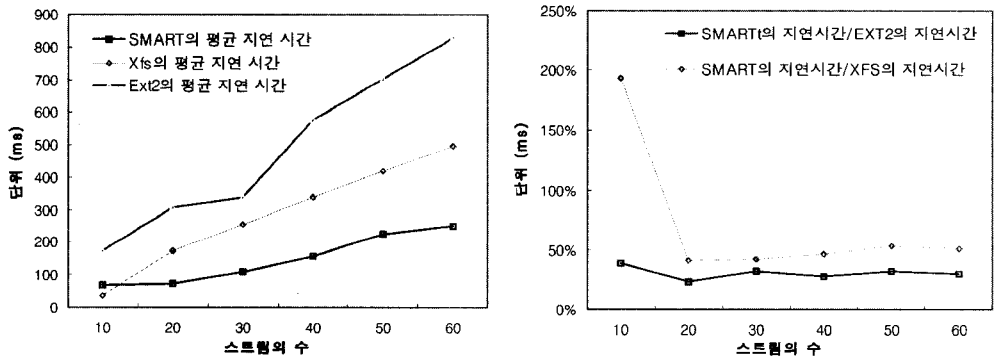


그림 6 128K바이트의 I/O 유닛에서의 읽기 지연 시간

화하는가 이다. 디스크에 많은 작업들이 도착 할수록, 개개의 작업은 보다 많은 시간을 기다려야 한다. 중요한 것은 그래프의 기울기라 하겠다. 32K바이트의 경우에는 EXT2보다 우수한 확장성을 나타내고 있다. 그림 5의 경우에는 좀 더 큰 단위로 입출력이 이루어진다. 이 경우 SMART 파일 시스템이 XFS보다 우수한 확장성을 나타낸다. 확장성에 관한 상대적 우수성은 입출력 단위의 크기를 보다 크게 설정 할수록 두드러지게 나타나고 있다. 그림 6에서는 SMART 파일 시스템이 EXT2 파일 시스템이나 XFS 파일 시스템보다 훨씬 우수한 확장성을 보이고 있다. 이는 동시에 읽는 스트리밍 세션이 증가 할수록 기존 리눅스 파일 시스템에서 2차, 3차 간접 레퍼런스 블록을 읽는데 걸리는 오버헤드가 I/O 지연 시간의 상당 부분을 차지하기 때문이다. 이에 반하여, SMART 파일 시스템은 레퍼런스와 블록 그룹을 제거하여 디스크 탐색 시간을 최소함으로써, 디스크 부하

가 존재하지 않는 플랫(Flat) 구조이므로, 메타 데이타를 읽는데 소요되는 오버헤드가 입출력 단위가 커지면서 상대적으로 줄어들게 된다.

스트리밍 서비스에서는 순차적 읽기가 주된 작동이므로 위의 실험 결과를 통해서 볼 때 SMART 파일 시스템은 스트리밍 서비스 환경에서 기존 유닉스 파일 시스템 계열 보다 우수한 성능을 보인다.

4.3 I/O 지연 시간의 변화

멀티미디어 전용 화일 시스템으로서의 적합성을 평가하는 데 중요한 두 가지 요소로 평균 입출력 지연 시간과 입출력 지연 시간의 분산을 들 수 있다. 멀티미디어 서비스에서 데이타 흐름의 연속성을 보장하기 위해서는 각 데이타가 일정시간 간격으로 목적지에 도달 하여야 한다. 여기서 목적지는 어떤 부분을 기준으로 보느냐에 따라 서버 주기억장치가 될 수도 있고, 클라이언트의 버퍼가 될 수도 있겠다. 이러한 연속성을 보장하는 데 있어서 가장 중요한 요소는 응답시간의 예측 가능성이다. 평균 응답시간은 빠르다 할지라도, 각 응답시간간의 편차가 매우 크다면 멀티미디어 서비스용으로는 적합하지 않을 수 있다. 때문에, 입출력 지연 시간의 분산이 멀티미디어 서비스에 적합하게 설계되었는가를 결정하는 핵심인자가 될 수 있는 것이다. 응답시간을 결정하는 데는 디스크 스케줄링 방법, 파일 시스템 구조, 파일 배치 방법 등 여러 가지 요소들이 있다. 데이타 블록은 읽는데 걸리는 시간을 정확하게 예측하는 것은 매우 복잡한 과제다. 스트리밍 환경에서는 각각의 데이타가 미리 정의된 시간 안에 전달되어야 하고, 따라서 읽기 작업 지연 시간의 분산이 작은 파일 시스템은 스트리밍 서버의 효율성과 스케줄링 능력을 개선할 수 있는 것이다.

이번 실험에서는 각 파일 시스템(SMART, EXT2, XFS)에서 30M바이트 크기의 파일을 순차적으로 읽는 실험을 30회 반복한다. 32K바이트 블록을 읽는 시간을 측정하고, 측정된 시간의 분산을 보고자 한다. 파일 시스템의 특성에 따라 입출력 시간이 균일하게 나타날 수도 있고, 아니면 같은 크기의 블록을 읽더라도 블록의 파일 내 위치 혹은 디스크상의 위치에 크게 다르게 나타날 수도 있다. 이는 파일의 구조, 디스크상에서의 블록 할당방법 등에 영향을 받는다. 이 실험에서는 입출력 지연 시간이 어느정도 균일하게 나타나는 지를 측정하고자 하는 것이 목적이다. 스트리밍 서버가 일정간격으로 비디오 프레임이나 오디오 샘플을 전송하는 데 있어서, 의도한 시간내에 자료가 디스크에서 읽혀지는 것이 매우 중요하다. 스트리밍 서버가 디스크의 입출력 지연 시간을 정확히 예측하는 것은 거의 불가능하다. 때문에 디스크의 입출력 지연 시간이 비교적 균일하게 분포하는 것이 스트리밍 서버의 입출력 스케줄링에 매우 중요한 역할을 한다. 따라서, 입출력 지연 시간의 분산이 적을수록 멀티미디어 서비스에 적합한 파일 시스템이라 하겠다. 요청된 데이타 블록이 버퍼 캐쉬에서 서비스되지 않도록 하였다.

32K바이트이고 각각의 I/O요청에 대한 지연 시간을 측정하고 측정된 시간의 분산을 계산 하였다. 요청된 데이타 블록이 버퍼 캐쉬에서 서비스되는 경우 정확한 입출력 지연 시간을 측정할 수 없기 때문에, 각 프로세스는 서로 다른 파일을 읽었다.

그림 7은 30M바이트의 크기를 가진 파일을 32K바이트로 읽는데 걸리는 시간의 평균과 분산을 나타내고 있다. 하나의 멀티미디어 세션이 존재하는 경우, SMART 파일 시스템에서의 지연 시간이 EXT2 파일 시스템에서의 지연 시간보다 적고, XFS 파일 시스템과는 비슷하게 측정되었다. 그림 7에서 보듯이 하나의 I/O 요청을 처리하는데 걸리는 분산은 SMART 파일 시스템이 EXT2 파일 시스템보다는 훨씬 작게 나타났다. XFS 파일 시스템은 멀티미디어 전용 파일 시스템답게 매우 우수한 지연 시간의 분산을 보이고 있다. 따라서, XFS 역시 매우 잘 설계된 파일 시스템이라는 것을 알 수 있다.

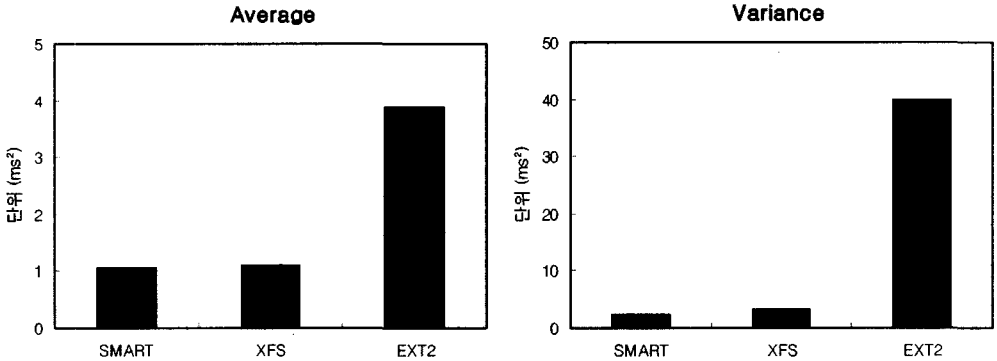


그림 7 32K 바이트의 I/O 유닛 사이즈에서의 읽기 시간의 평균과 분산

SMART 파일 시스템은 XFS 파일 시스템보다 약 30% 적은 분산을 보인다. EXT2 파일 시스템에서는 멀티 레벨을 갖는 데이타 블록 레퍼런스 구조와 블록 그룹의 배치 전략으로 인해 파일에 따라 읽기 지연 시간의 차이가 매우 다르게 나타났다. 반대로 SMART 파일 시스템에서 데이타 블록은 단일 계층 구조를 가지고 있어, 파일의 메타 데이타는 항상 데이타 블록과 인접해 있다. 제안된 파일 시스템에서 모든 파일은 동일한 구조를 가지며, 디스크 플래터(Platter)상에서 비슷한 배치 구조를 가지게 된다. 이러한 파일시스템의 구조적 특성으로 인하여 전체 파일을 읽는데 걸리는 시간이 매우 균일하게 나타난다.

5. 결 론

온라인 교육, 주문형 비디오 등의 분야를 중심으로한 멀티미디어 스트리밍 서비스의 급속한 대중화로 인하여 멀티미디어 서비스를 효율적으로 지원할 수 있는 서버의 설계가 매우 강조되고 있다. 최근에 부각되고 있는 PVR, Set-Top Box등의 정보가전이나 새로운 응용분야로 등장한 자동차용 멀티미디어 시스템 모두 멀티미디어 부하에 적합하게 설계된 파일 시스템이 필수적이다. 멀티미디어 파일의 특징은 2개의 인자, 즉 '크기'와 '대역폭'으로 특징지어 질 수 있다. 스트리밍 서비스를 위한 파일 시스템은 대용량의 파일을 효과적으로 다룰 수 있어야 하고, 데이타를 읽어들이는 데 요구되는 오버헤드를 최소화하여 일정한 데이타의 전송 대역폭을 유지하며, 자료를 전송해 주어야만 한다. 기존 파일 시스템은 다양한 크기의 파일을 효율적으로 관리하는데 초점이 맞추어져 있고, 이로 인하여 다소 복잡한 파일 구조를 갖는다. 때문에, 기존 파일 시스템은 멀티미디어 자료를 효과적으로 저장, 전송하는 데 비효율적인 요소를 가지고 있다. 본 논문에서는 멀티미디어 자료를 효과적으로 저장 및 전송하는 목적으로 최적화된 파일 시스템을 연구, 개발하고, 이의 성능 측정과 분석에 초점을 맞추었다. 본 논문에서는 파일의 구조를 보다 단순하게 설계하여 기존의 유닉스 파일 시스템의 멀티레벨 구조가 가지는 성능 측면에서의 한계를 극복하고, 자료의 저장단위를 물리적인 단위에서 멀티미디어 자료의 논리적 단위로 채택함으로써 멀티미디어 서비스를 제공하는 데 요구되는 최적의 파일시스템을 개발하였다. 본 논문에서 제안하는 SMART 파일시스템은 기존 유닉스 계열의 EXT2 파일 시스템과 XFS 파일 시스템과 비교하여 빠른 응답속도와 비교적 안정적인 응답 등 많은 장점을 가지고 있는 것으로 분석되었다. 본 연구는 A/V 시스템이나 차세대 정보가전에서 효율적으로 사용될 수 있는 멀티미디어 전용 파일 시스템의 개발이라는 측면에서 매우 큰 의미를 가지고 있다.

참 고 문 헌

[1] D.R. Kenchammana-Hosekote and J. Srivastava. "Scheduling Continuous Media on a Video-On-Demand Server," In *Proc. of International Conference on Multi-media Computing and Systems*, Boston, MA, May 1994. IEEE.

[2] P. Rangan, H. Vin, and S. Ramanathan. "Designing an on-demand multimedia service," *IEEE Communication Magazine*, 30(7):56-65, July 1992.

[3] J. Gemmell. "Multimedia Network File Servers: Multi-Channel Delay Sensitive Data Retrieval," In *Proc. of 1st ACM Multimedia Conf. ACM*, Oct. 1993.

[4] B Ozden, A. Biliris, R. Rastogi, and Avi Silberschatz. "A Low-Cost Storage Server for Movie on Demand Databases," In *Proc. of VLDB '94*, 1994.

[5] Y. Rompogiannakis, G. Nerjes, P. Muth, M. Paterakis, P. Triantafillou, and G. Weikum. "Disk scheduling for mixed-media workloads in a multimedia server," In *Proceedings of ACM Multimedia '98*, pages 297-302, Bristol, UK, 1998.

[6] Youjip Won and Jaideep Srivastava. "SMDP: Minimizing buffer requirements for continuous media servers," *ACM/Springer Multimedia Systems Journal*, 8(2):pp. 105-117, 2000.

[7] Shenoy, P., Goyal, P., Rao, S., S., and Vin, H., "Symphony: An Integrated Multimedia File System," Technical Report TR-97-09, Department of Computer Science, Univ. of Texas at Austin, Mar 1997.

[8] Prashant Shenoy, Pawan Goyal, Harrick M. Vin. "Architectural considerations for next generation file systems," *Proceedings of the seventh ACM international conference on Multimedia*, 1999.

[9] Dominic Giampaolo, "Practical File System Design," Morgan Kaufmann, 1999.

[10] Roger L.Haskin. "Tiger Shark - a scalable file system for multimedia," *IBM Journal of Research and Development* v 42 n 2 p185-197, 1998.

[11] William J.Bolosky, Robert P.Fitzgerald, John R.Douceur. "Distributed schedule management in the Tiger video file server," *Operating Systems Review (ACM)* 1997.

[12] T.N. Niranjan, Tzicker Chiueh, Gerhard A.Schloss. "Implementation and Evaluation of a Multimedia file system," *Proc. of Int'l Conference on Multi-media Computing and Systems*, Jun 3-6, 1997.

[13] Chuanbao Wang, Vera Goebel, Thomas Plagemann. "Techniques to increase disk access locality in the Minorca multimedia file system," *Proc. of the seventh ACM Multimedia Conference*, 1999.

[14] Wonjun Lee, Difu Su, Duminda Wijesekera,

Jaideep Srivastava, Deepak Kenchammana-Hosekote, Mark Foresti. "Experimental Evaluation of PFS Continuous Media File System," Proceedings of *CIKM '97*, p.246-253, Las Vegas, Nevada, USA.

[15] Youjip Won, Jinyoun Park, Sangback MA, "HERMES: File System Support for Multimedia Streaming in Information Home Appliance," Lecture Note in Computer Science, Vol. 2510, pp. 172-179, Oct. 2002.

[16] http://www.quicktime.com

[17] http://www.sgi.com/software/xfs

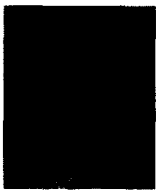
[18] Michael Beck, Harald Bohme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus. "Linux Internals," Addison-Wesley Pub Co, ISBN: 0201331438.

박 진 연

2000년 한양대학교 전자통신전파공학과 학사. 2002년 한양대학교 전자통신전파공학과 대학원 석사. 2002년~현재 지티이 퓨처텔 근무. 관심분야는 멀티미디어 시스템, QoS, 운영체제

김 두 한

2002년 경원대학교 공과대학 전자공학과 학사. 2002년~현재 한양대학교 전자통신전파공학과 석사과정. 관심분야는 멀티미디어 시스템, 운영체제, 실시간 시스템, QoS



원 유 집

1990년 서울대학교 자연과학대학 계산통계학과 학사. 1992년 서울대학교 자연과학대학 계산통계학과 전산학 석사. 1997년 University of Minnesota 전산학 박사. 1997년~1999년 Server Performance Analyst, Intel Corp. 1999년 3월~현재 한양대학교 공과대학 전자전기컴퓨터 공학부 교수. 관심분야는 멀티미디어 시스템, 멀티미디어 네트웍, 인터넷 프로토콜, 성능평가이론, 데이타베이스, 운영체제

류 연 승

1990년 서울대학교 자연과학대학 계산통계학과 학사. 1992년 서울대학교 자연과학대학 전산과학과 석사. 1996년 서울대학교 자연과학대학 전산과학과 박사. 1996년~2000년 삼성전자 선임연구원. 2000년~2003년 한림대학교 조교수. 2003년~현재 명지대학교 조교수. 관심분야는 운영체제, 실시간 시스템, 멀티미디어 시스템