主題

정보가전용 멀티미디어 파일시스템 기술

원 유 집, 박 진 연

차 례

Ⅰ. 서론

1. 정보가전과 멀티미디어 파일 시스템

앞으로는 TV · 오디오 · VCR · 냉장고 · 세탁기 등 기존 제품들에 컴퓨터 및 정보통신 기술을 접목, 가전제품 고유의 기능을 수행하면서 인터넷을 통해 가정에서 필요한 각종 정보를 얻을 수 있는 디지털 정보 가전이 주류를 이룰 것으로 기대되고 있다. 정보가전의 몇 가지 예를 들면[1] (i) 직접 책을 사지 않고, 소설책 등의 책을 인터넷을 통해서 전자책에 다운로드 받아 다니거나, 의사나 간호사는 진료표 등을 기록하여 언제든지 볼 수 있는 전자 책 (ii) WAP (Wireless Application Protocol)를 통해서 장소와 시간에 구애 받지 않고 언제든지 인터넷에 접속하여 전자 상거래, 인터넷 서핑이 가능한 인터넷 가능 핸드폰 (iii) 양방향성 기능이 부여 되어, TV 화면을 통해 자신이 원하는 정보를 얻을 수 있는 WebTV 와 홈씨어터 (iv) 무선 기능이 장착된 PDA (v) 현재 우리는 웹서버를 통해서 전자상거래, 주식 거래, 경매 참가, 카메라로 집 감시하기 등의 일을 할수 있지만, 앞으로는 이러한 기능이 집으로 이동하여 가정 기기등을 통합하여 관리하고, 멀티미디어 자료등을 저장 할 수 있는 홈 서버 등을 들 수 있다.

급격한 압축기술의 발달, 통신 대역폭의 확장, CPU 속도의 고속 신장등이 상호 결합하여 TV 서비스 방식에도 일대 변혁이 일어나고 있다. TV로 인터넷의 각종 콘텐츠를 즐기는 인터넷TV 서비스가 시작됐으며 우리나라도 이르면 올 연말부터 지상파 디지털 TV 본 방송과 디지털 위성방송이 서비스될 예정이다. 또한 바쁜 직장생활로 TV를 볼 수 없더라도 양방향서비스를 기반으로 한 디지털방송이 본격화하면 정해진 방송시간이 아니더라도 사용자가 원하는 시간에 TV 프로그램을 시청할 수 있는, 정교한 형태의 서비스가 가능해진다.

이들 정보 가전의 특징은 제품 설계 초기부터 해당 기기의 사용목적이 잘 정의된다는 것이다. 때문에 범

용 PC에 비해 효과적으로 시스템을 구성할 수있다. 인터넷 검색만을 위한 기기라면 CD-ROM같은 장치는 필요성이 떨어진다. 동영상 상영을 위한 기기의 경우에는 리모콘등의 제어장치를 제공하며, 고속 수치 연산을 위한 CP U라던지, 대용량 시스템을 위한 운영체제등은 필요없을 것이다. 불필요한 요소들을 시스템 구성에서 제외시킴으로써 해당 제품의 가격 경쟁력을 강화시시킬 수 있다.

컴퓨터를 이용한 동영상 자료의 저장/처리가 점점 대중화되면서, 정보가전 기기에서 멀티미디어 정보를 효율적으로 저장/처리하는 것이 매우 중요한 사안으로 등장하고 있다. 특히, 차세대 지능형 방송(혹은 대화형 방송)서비스에 필수적으로 요구되는 PVR(Personal Video Recorder) 혹은 Set-top Box에서는 동영상 서비스를 효율적으로 지원할 수 있는 기능이 매우 강조되고 있다. 이들 정보 가전은 기본적으로 CBR 및 VBR 압축방식에 의해 압축된 파일들을 지원해야 하며, 다양한 형식의 데이터(멀티미디어, 텍스트, 이미지)를 상호 간섭 현상 없이 처리할 수 있어야 한다. 이를 위해서는 보다 정교한 프로세스 스케줄링 및 자원할당 기능을 가진 운영체제가 탑재되어야 한다. 일반적인 데스크 탑 PC와는 달리 정보 가전기기는 고사양의 컴포넌트들로 구성되지 않는다. 가장 큰 이유는 가격 경쟁력 유지 때문이라고 할 수 있다. 저속의 하드 디스크, 비교적 낮은 사양의 CPU등으로 구성되기 때문에 주어진 자료를 처리하는 소프트웨어 시스템이 매우 효율적으로 설계되어야 한다. 가장 중요한 기능중의 하나로 방송 프로를 녹화하면서 저장된 프로를 재생하는 것이다. 이를 효과적으로 지원하기 위해서는 디스크 사용율을 극대화해야 하고, 파일 시스템을 포함한 운영체제의 해당 부분이 멀티미디어 파일에 최적화 되어야 한다. 일예로, Tivo[2]는 MFS(Media File System)와 별도의 디스크 스케줄링 서브시스템을 분리하여 저사양의 디스크를 이용하여 높은 대역폭의 멀티미디어 파일을 처리하고 있다.

2. 스트리밍 부하의 특성

Ftp, http, telnet 등과 같은 텍스트 기반 입출력과 스트리밍 자료를 위한 입출력의 가장 큰 차이는 대역폭의 보장이다. 텍스트 기반의 데이터 전송동작(ftp, http, telnet등)은 데이터를 목적지에 정확히 전달하는 것이 최종 요구 조건이다. 이와 달리, 스트리밍 서비스의 경우, 각 단위 자료(패킷 혹은 프레임)가 정해진 시간 내에 목적지에 전달되는 것이 중요하다. 자료의 "적시성"이 자료의 "정확성"보다 강조되는 경우라 할 수 있겠다. 정해진 시간 내에 요청된 자료가 도착하지 않을 경우, 사용자 측에 화면의 일그러짐, 끊김 등의 현상이 발생하며, 이로 인해 원 자료의 의미가 사용자에게 제대로 전달되지 못하는 경우가 생기는 것이다. 따라서, 서버가 다수의 사용자에게 스트리밍 서비스를 제공하는 경우 각각의 사용자를 위해 일정량의 시스템 자원(CPU 사용율, 네트워 대역폭, 기억장치의 대역폭, 주기억장치 버퍼)들을 예약하는 것이 필수적이라 하겠다.

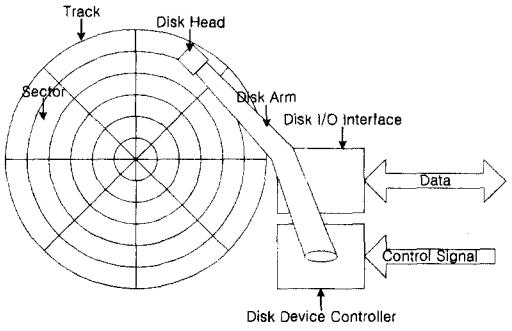


그림 1. 하드 디스크 구조

그림 1은 디스크의 구성요소를 도식적으로 표시하고 있다. 디스크로부터 데이터를 읽는 시간은 (i) 디스크 헤드를 움직이는 시간(seek time), (ii) 디스크 원판이 회전하여 해당 데이터 블록의 위치를 찾는 시간(rotational latency), 그리고 (iii) 실제 데이터를 읽는 시간(transfer time)으로 구성된다.

멀티미디어 데이터를 효과적으로 읽고 쓰기 위해서는 검색 시간과 회전 지연 시간, 그리고 데이터 블록의 위치를 찾는 데 필요한 파일 메타 데이타의 연산량을 줄이는 것이 핵심이라 할 수 있다. 디스크 헤드 이동 시간은 데이터 블록의 배열에 따라 결정된다. 데이터 블록을 배치하는 방식과 파일 메타 데이타의 구조 등은 파일 시스템에서 정의하는 사항이다. 따라서, 디스크의 효율성을 극대화하기 위해서는 스트리밍 부하의 특성을 정확히 파악하여 이에 최적화 된 파일 시스템을 설계하는 것이 가장 중요하다. 스트리밍 전용 파일 시스템의 디자인에서, 데이터 검색에 필요한 디스크 탐색(disk seek) 오버헤드를 최소화 시키는 것이 매우 중요한 사항이라 할 수 있겠다.

유닉스 파일 시스템이 파일 시스템의 발전 역사에서 매우 중요한 위치를 차지하고 있는 것은 누구도 부정할 수 없는 사실이다. 유닉스 파일 시스템의 근본 설계 철학은 텍스트 기반의 비교적 작은 파일의 처리 성능을 중점적으로 향상시키면서, 동시에 매우 큰 파일까지도 다루는 것이다. 때문에, 유닉스 파일 시스템에서 파일은 편향된 다진 트리(skewed n-ary tree)의 leaf에 파일의 데이터 블록들이 존재하는 형태를 가진다. 이러한 유닉스 파일 시스템의 구조는 스트리밍 환경에 사용되기 위해서는 많은 개선의 여지를 포함하고 있는 것이 사실이다.

본 논문은 크게 두 가지를 주제를 다루고자 한다. 먼저 스트리밍 환경에 최적화된 파일 시스템을 제안한다. 파일 시스템은 디스크 검색 시간을 최소화하고, 데이터 블록의 위치를 계산하는 데 관련된 파일 메타데이터 연산작업을 최소화 하는 데 초점을 맞추고 있다. 두번째로, MPEG-4 압축방식으로 압축된 파일을 효과적으로 다루기 위한 파일 시스템의 요구조건을 정의하고 이를 위한 지원방법을 제시하고자 한다. 압축과정에서 제시하는 각종 정보들을 효과적으로 사용하는 시스템 모듈들을 정의함으로써 스트리밍 서버의 효율성을 극대화 시키고자 하는 것이다.

3. 관련 연구

멀티미디어 서버 설계 관련된 다수의 연구결과가 최근에 발표되고 있다[3,4,5]. 멀티미디어 데이터를 정해진 시간 내에 전달하기 위하여 디스크 대역폭의 일정량이 멀티미디어 서비스를 위하여 확보 되어야 한다. 이 부분의 핵심 논의 사항은 디스크로부터 프레임을 읽을 때, 데이터 흐름의 연속성을 어떻게 제공해 주느냐 하는 것이다. 단일 디스크 관련 이슈와는 별도로 멀티미디어 파일 서버를 위한 디스크 서브시스템에 관해서도 많은 연구가 있었다[6,7]. [8]은 디스크 스케줄링 알고리즘을 디스크 어레이(array)까지 확장시켰다. 많은 버퍼를 요구하는 것은 스트리밍 서비스를 제공할 때 상당히 많은 버퍼 오버헤드를 보인다. 스트리밍 서비스에서 디스크 오버헤드를 줄이기 위해 수많은 접근이 제안되었고, 제안된 연구의 대부분은 전에 읽혀진 스트림에 의해 로드(load)된 데이터 블록을 다시 사용하는 방법에 대한 것들이다[9].

현재의 디스크 드라이브 기술은 zoning 기술을 적용하고 있으며, 이 기술은 바깥쪽 실린더가 안쪽 실린더보다 많은 수의 섹터(sector)를 가지고 있는 것이다. Tewari외[10]는 구역 기반(zoned disk) 디스크의 멀티미디어 파일 배치 문제를 연구하였다. [11,12,13,14]에서는 멀티미디어 파일 시스템을 위한 디스크 스케줄링과 zoning 에서의 디스크 전송률 변화에 관한 이슈를 구체화하였다. 특히 Nerjes외[11]는 VBR 부하 하에서 정교한 디스크 스케줄링 모델과 서비스 요청 수락 모듈에 대한 알고리즘을 제안했다.

최근에는 여러 종류의 다양한 미디어들을 스트리밍 서비스에서 서로 구애 받지 않고 전송해 주는 문제에 관한 연구가 행해지고 있다[15]. Shenoy외[16]는 미디어 타입에 맞게 최적화되어 분리된 파일 시스템 파티션보다 1개의 파일 시스템 프레임 워크 안에서 이기종간의 workload를 효과적으로 다룰 수

있는 것이 가능하다고 결론지었다. [17]에서는 멀티미디어 재생뿐만 아니라 재생과 상관없이 산발적으로 발생하는 I/O 요청을 효과적으로 다룰 수 있는 디스크 스케줄링 알고리즘을 개발하였다.

또한 멀티미디어 데이터를 특별히 다룰 수 있는 파일 시스템의 프로토타입(prototype)이 제안되었다 [18,19]. MMFS[20]은 prefetching, state-based caching, 우선 순위 디스크 스케줄링, 동기화된 멀티 스트림을 이용하여 양방향 재생의 성능을 높였다. VCR과 같은 재생 모드(빠른 재생, 뒤로 재생)의 응답 시간을 개선하였으며, 여러 개의 멀티미디어 스트림간에 동기화 skew를 최소화 하도록 설계되었다.

Minorca 멀티미디어 파일 시스템[21]은 (1) MOSA 라는 디스크 레이아웃과 새로운 데이터 배치 기술을 제안했고, 이것은 크기가 큰 멀티미디어 파일에게는 연속적으로 디스크를 할당하였고, 동시에 파일 사이즈가 작으면서 연속성 없는 데이터들도 1개의 파일 시스템에서 지원해 주고 있으며 (2) I/O 요청 큐(queue)를 최적화 하기 위해서 새로운 read-ahead 방법을 제안하고 있다. 이러한 기술의 목표는 디스크 접근의 국지성(locality)를 높이며 디스크 탐색 오버헤드(overhead)를 줄이는데 있다.

Ⅱ. 유닉스 파일 시스템과 스트리밍

스트리밍 서비스를 위한 구체적인 설계방식을 소개 하기에 앞서, 먼저 유닉스 계열의 파일시스템을 소개하고, 유닉스 파일 시스템의 특성과 스트리밍 부하의 관계를 언급하고자 한다. 본 절에서는 리눅스 운영체제에서 가장 많이 사용되는 Ext2 파일 시스템을 소개한다.

1. 유닉스 파일 시스템에서 inode 의 구조

유닉스 파일 시스템은 전형적으로 디스크 파티션 관리를 위한 정보와 데이터를 엄격하게 분리를 하고 있다. 파일에 관련된 정보를 저장하고 있는 데이터 블록을 "i-node"라고 한다. i-node 에는 해당 파일에 관련된 각종 메타 정보가 수록되어 있으며, 실제 데이터 블록의 주소(레퍼런스)를 가지고 있다. Ext2 파일 시스템의 inode는 파일 모드(예를 들어 rwxrw-r--), 소유자의 id, 크기 등 관리를 위한 정보와 데이터 블록의 주소를 가지고 있다. i-node는 총 15개의 데이터 블록 주소를 가지고 있다. 12개의 주소는 데이터 블록의 주소이다. 나머지 3개의 주소는 간접 주소(indirect reference) 블록, 2중 간접 주소(two step indirect reference), 3중 간접 주소(three step indirect reference) 블록의 위

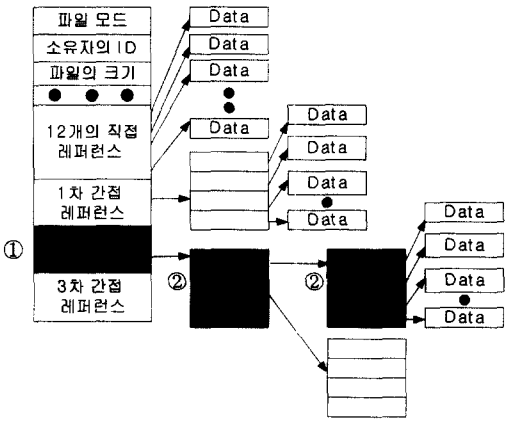


그림 2. Ext2 inode의 구조

치를 나타낸다. Ext2 파일 시스템에서는 데이터 블록의 크기를 1K, 2K, 4K로 조정할 수 있다. 데이터 블록의 크기는 파일 시스템 파티션을 포멧할 때 지정한다. 데이터 블록이 4K바이트라고 가정하면, 직접 주소(direct reference)를 사용해서 나타낼 수 있는 최대 파일 사이즈는 48Kbytes 이다 (12개의 직접 주소 * 4K바이트 데이터 블록). 주소를 나태내기 위해서는 4바이트가 필요하므로, 1개의 4K바이트의 블록은 1024개의 블록 포인터를 가지게 되고, 따라서 간접 주소 블록까지 사용할 경우 최대 4Mbytes의 파일을 저장할 수 있다. 이중 간접 주소

(Two-step indirect reference)까지 사용하면 1개의 파일에서 4G바이트까지 저장할 수 있다.

이러한 트리 구조 기반 파일 구성은 주로 작은 크기의 파일을 파일 시스템이 다루면서 매우 큰 파일까지도 지원하고자 할 때 상당히 유용하게 사용될 수 있다. 그러나, 스트리밍 부하에서와 같이 순차적으로 파일을 읽고자 할 때에는 불필요한 오버헤드(overhead)를 유발하게 된다.

2. 유닉스 파일 시스템에서 멀티미디어 파일

압축 기법을 사용하더라도 1시간분량의 영화는 수백 메가에서 수 기가까지의 디스크 용량을 필요로 한다. MPEG-1으로 압축된 비디오 재생 대역폭을 1.5Mbps이라 할 경우, 90분 길이의 영화는 약 1기가 바이트의 저장 공간을 필요로 한다. MPEG-2 압축방식은 고화질 영상을 위해 제안되었으며, 4-20Mbps의 미디어 재생 대역폭을 가지고 있으므로, 더 많은 저장장소를 필요로 한다. 이러한 대용량의 파일을 저장하기 위해서는 이중 혹은 3중 간접 주소 블록까지 사용해서 파일을 구성해야 한다. 1기가 바이트 파일의 마지막 데이터 블록을 읽기 위해서는 3개의 블록을 통해서 접근이 이루어진다. inode 테이블 블록(그림 2에서의 ①) 과 2개의 간접 주소 블록(그림 2에서의 두개의 ②)이 이에 해당된다. 요약하자면, 기존의 유닉스 계열의 파일 시스템, 특히 EXT2 파일 시스템은 스트리밍 부하를 위한 환경에서 사용되기 위해서는 다음과 같은 개선점들을 안고 있다.

- 다중 간접주소 블록을 통한 접근 : 하나의 데이터 블록을 읽을 경우 inode블록과 다수의 간접 주소 블럭들을 먼저 읽어야 함으로 이에서 유발되는 입출력의 부하가 매우 크다. 간접 주소 블록이 버퍼 캐쉬에 존재한다 할지라도 데이터 블록 주소를 계산하는 과정이 CPU에 많은 부하를 가할 수 있다.
- 간접 주소 블록과 데이터 블록의 배치: Ext2 파일 시스템에서는 데이터 블록의 주소를 가지고 있는 간접 주소 블록과 데이터 블록이 따로 저장되어 있다. 포인터 블록과 데이터 블록이 연속적으로 저장 되어있지 않기 때문에, 디스크 헤드는 inode와 포인터 블록을 읽기 위해 디스크 플래터(platter) 상을 이동한다. 디스크 탐색 시간은 디스크로부터 데이터블럭을 읽는 시간 중에서 상당히 많은 부분을 차지하고 있으므로, 파일에 관련된 데이터 블록과 간접 주소 블럭을 연속적으로 디스크에서 나열하며 헤드의 움직임을 최소화하는 것이 매우 중요하다. 평균 디스크 검색 시간은 임의의(random) 검색 패턴으로 512바이트 섹터를 읽어 오는데 총 시간의 67% 를 차지한다[22].
- 데이터 블록의 단편화 : 파일의 생성과 삭제과정이 반복될 경우 데이터 블록들이 디스크 플래터 상에서 흩어져 배치될 수 있고, 따라서, 이로 인한 디스크 탐색 시간의 증가가 발생할 가능성이 있다.

3. 유닉스 시스템의 파일 배치 전략

리눅스, 솔라리스, NetBSD 등 유닉스 파일 시스템의 대부분이 블록 그룹 내지는 실린더 그룹의 개념을 이용하여 데이터 블록 배치를 최적화하고 있다. 이들 기법의 궁극적인 목표는 연속한 데이터 블록들을 서로 인접하거나, 이가 불가능할 경우 최대한 가깝게 배치하는 것이다. Ext2 파일 시스템은 블록 그룹 개념을 도입하여, 파일의 데이터 블록과 inode, 디렉토리 할당을 정교하게 관리한다[23]. 블록 그룹은 일련의 연속된 실린더들의 그룹이다. 블록 그룹의 크기는 파일 시스템 포맷 시에 결정되고, 다시 포맷하기 전까지는 변하지 않는다.

Ext2 파일 시스템은 여러 개의 블록 그룹으로 구성이 되어 있으며 각각의 블록은 파일 시스템의 견고성을 유지하기 위한 수퍼블록(superblock)의 복사본과 그룹 descriptors, 블록 비트맵, inode 비트

맵, inode 테이블과 데이터 블록을 가지고 있다. 블록 그룹을 사용하여 파일 시스템은 상대적으로 좀더 가까운 실린더 위치에 파일의 데이터 블록을 배치한다. 데이터 블록을 배치할 때, 블록 그룹 내에 배치하는 것에 우선순위를 부여함으로써, 데이터 블록배치의 효율성을 향상시킨다. 하지만, 데이터 블록 기반 정책은 여전히 블록 그룹의 크기를 넘는 파일을 여러 개의 다른 블록 그룹에 파일을 나누어서 저장하게 되고 이것은 디스크 탐색에 있어 오버헤드를 유발하게 된다.

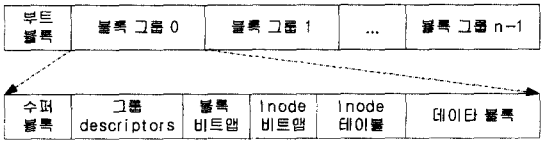


그림 3. Ext2 파일 시스템의 디스크 구조

Ⅲ. 멀티미디어 데이터 저장 형식

차세대 인터넷 스트리밍을 위한 표준으로 주목받고 있는 압축방식은 MPEG-4 이다. 이를 저장하는 파일 형식인 mp4 파일 포멧에 대하여 소개하고, 이를 효과적으로 지원하기 위한 파일 시스템 수준의 전략을 제시하고자 한다.

1. MPEG-4 파일 구조

MPEG-4 파일(*.mp4)은 애플사의 스트리밍 전용 파일 포멧인 퀵 타임(Quicktime) 포맷을 근간하여 MPEG-4 표준안에서 추가된 기능을 지원하도록 구성되어있다. 퀵타임 파일[24]은 미디어에 대한 명세와 미디어 데이터를 분리하여 저장한다. 미디어에 대한 명세 혹은 메타 데이터는 트랙의 개수, 비디오 압축 방식, 시간 정보등에 관한 정보들 뿐만 아니라, 미디어 데이터에 저장된 위치에 관한 정보도 기억하고 있다. 미디어 데이터는 비디오 프레임, 오디오 샘플 등이 여기에 해당한다.

퀵타임 파일 포멧의 정보를 저장하는 기본 단위는 QT 아톰으로서, 퀵타임 파일은 이들 QT 아톰들의 집합이다. QT 아톱에는 아톰의 사이즈, 타입, 아톰의 내용 순으로 저장되어 진다. 또한 아톰은 다른 아톰을 포함하는 계층적 구조를 가질 수 있다. 우리가 흔히 접할 수 있는 일반 영화의 경우를 생각해 보도록 하자. 영화는 일단 영상 트랙, 사운드 트랙, 그리고 자막 부분 등으로 구성되어 있을 것이다. 퀵 타임 포멧에서는 이 영화가 하나의 영화 아톰, 즉, 무비(movie) 아톰으로 표현된다. 무비 아톰의 헤더 부분은 해당 영화의 전체적인 정보를 기억하고 있으며, 영상 트랙, 사운드 트랙, 그리고 자막 부분이 각기 하나의 미디어 트랙 아톰을 이루어 무비 아톰에 포함되게 된다. 이 트랙 아톰들은 각각의 트랙에 관한 정보를 기억하고 있다.

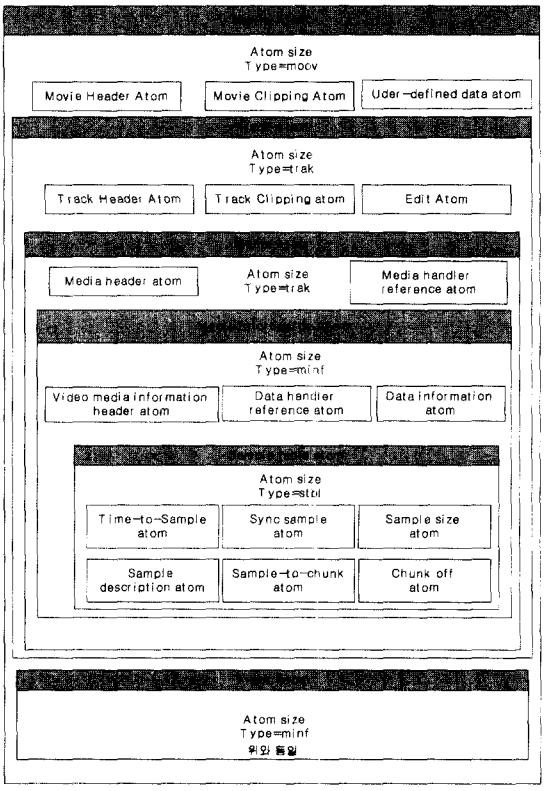


그림 4. Quicktime 파일 구조

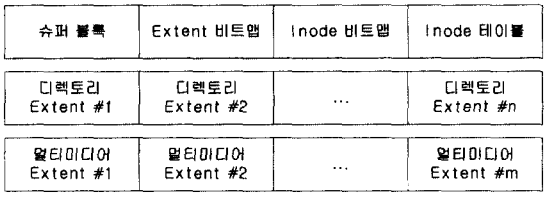


그림 5. 리눅스 기반 MPEG-4 파일 시스템 레이아웃

2. MP4 파일의 세부 구조

그림 4는 2개의 트랙으로 이루어진 멀티미디어 파일의 아톰 구성도를 나타내고 있다.

무비에 2개의 트랙이 존재하므로 무비 아톰은 하위에 2개의 트랙 아톰을 가지고 있다. 1개의 트랙 아톰에는 미디어 아톰이 존재하며, 미디어 아톰에는 샘플 테이블 아톰이 존재한다. 샘플 테이블 아톰에는 시간-샘플에 관한 아톰, 동기화 샘플 아톰, 샘플-청크 아톰, 청크 오프셋 아톰 등이 저장되어 있다. 이 아톰들을 통해서 트랙의 특정 샘플에 접근을 할 수 있다.MP4파일 포멧에서 필요로 하는 대부분의 atom들은 퀵 타임에서 이미 정의되어 있으나, MPEG-4 Part 6 :System Part[25] 에서 정의하고 있는 OD(Object Descriptor), ESD (Elementary Stream Descriptor), BIFS (Binary Format for Scenes)등은 퀵타임 포멧에서는 정의되지 않고 있다. 이를 위하여, MP4 파일은 아톰 포멧을 추가로 정의하고 있다. 이를 요약하면 표 1과 같다.

아톰 이름	타입코드
MP4CopyrightAtom	cprt
MP4ESDAtom	esds
MP4ObjectDescriptorAtom	iods
MP4ObjectDescriptorMediaHeaderAtom	odhd
MP4ODTrackReferenceAtom	mpod
MP4SceneDescriptionMediaHeaderAtom	sdhd

표 1. MPEG-4 파일에 새로 추가된 아톰들

Ⅳ. HERMES : 정보가전용 멀티미디어 파일 시스템

이번 장에서는 스트리밍 전용 시스템을 위하여 한양대학교 분산멀티미디어 연구실에서 연구 개발된 HERMES 파일 시스템[26]을 개괄적으로 소개하도록 하겠다. HERMES 파일시스템은 이제까지 멀티미디어 시스템분야에서 학계에서 제시된 최신 기법들과 자체 개발된 이론들을 총체적으로 통합 구현한 스트리밍 전용 파일 시스템이다. 유닉스 계열 파일시스템은 데이터 블록 단위로 파일을 배치하기 때문에 동영상 파일처럼 상대적으로 큰 용량의 파일에 대해서는 디스크 단편화가 발생할 수 있다. 이를 극복하기 위하여 HERMES 파일 시스템에서는 Extent를 파일 배치의 기본 단위로 한다. Extent는 연속된 블록들의 집합이며 크기는 1-2MByte 정도로 해당 파티션을 포맷할 때 포멧 작업의 인자로 결정된다.

그림 5는 본 연구에서 개발한 파일 시스템에서 단일 파티션 상의 레이아웃을 보여주고 있다. Extent는 파일의 가장 기본적인 구성 단위이며, extent는 여러 개의 데이터 블록이 모여서 구성된다. 멀티미디어 파일은 멀티미디어 extent에 저장이 되며, 디렉토리의 내용은 디렉토리 extent에 저장이 된다. 이렇게 멀티미디어와 디렉토리를 따로 분리해서 저장하는 이유는 파일 종류에 따라 지역성(locality)을 높여서 디스크 탐색(seek) 시간을 줄이기 위해서다.

Extent 비트맵은 각각의 extent의 사용 유무를 비트맵 형식으로 저장한다. 새로운 멀티미디어 파일이 생성되면, 이 비트맵을 이용하여 사용하지 않는 첫 번째 extent를 찾아가서 저장된다. 각각의 inode는 파일의 메타 정보를 기억하고 있다. 파일 소유자의 ID, 파일 소유자의 그룹 ID, 파일의 모드

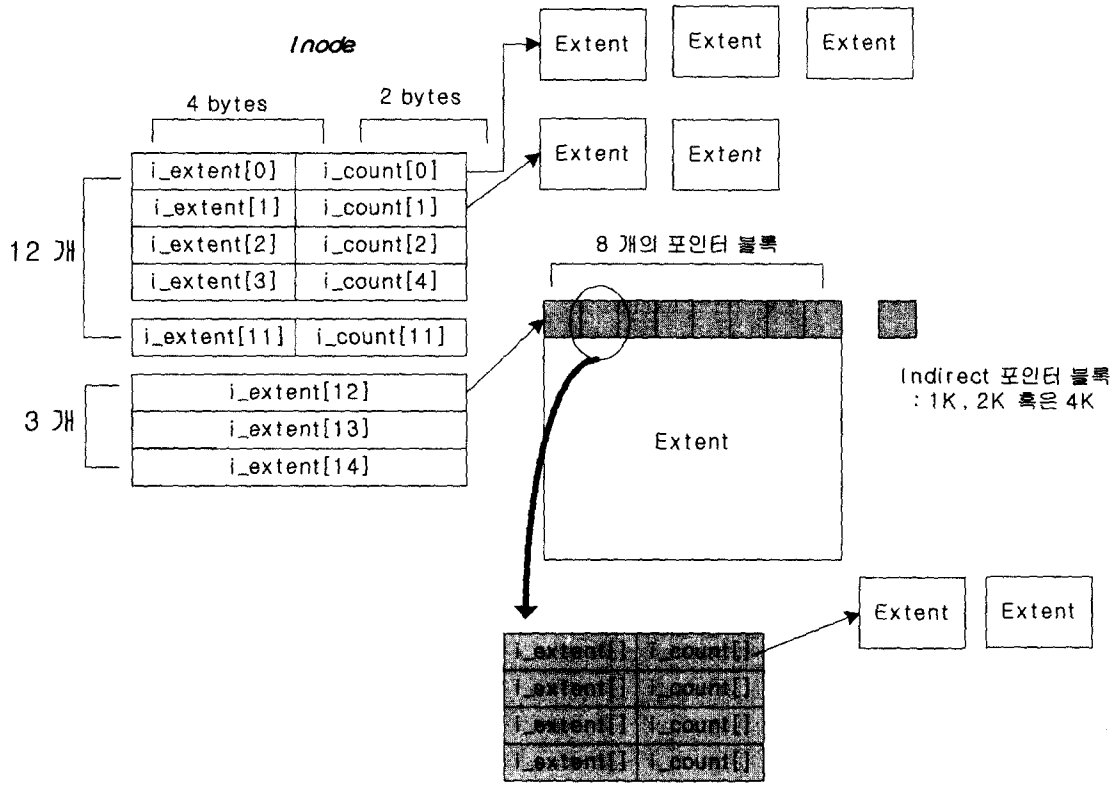


그림 6. *inode* 구조와 블록 맵

(예, rwxr-xrw-) 등이 그것이다. inode 테이블에서는 각 inode 사용 유무를 비트맵 형식으로 저장하고 있다. 위와 같은 디스크 배치 전략으로 인해 얻을 수 있는 이점은 아래와 같다.

- ▶ 멀티미디어 스트리밍 환경의 최적화
- ▶ 임베디드 시스템에 적합한 견고성
- ▶ 사용자 QoS 보장
- ▶ I/O latency 의 최소화
- ▶ 다양한 파일 사이즈에 대한 지원

1. HERMES 메타 데이터 블럭의 구조

유닉스 계열 파일 시스템의 i-node 구조는 데이터 용량이 크며, 순차적 접근이 많은 멀티미디어 파일에는 적합하지 않기 때문에, inode를 재구성 하였다. 아래 그림은 본 연구에서 제안한 멀티미디어 파일 시스템의 inode 구조를 나타내고 있다. 멀티 레벨의 간접 레퍼런스를 사용하지 않고 단일 계층 레퍼런스 구조 및 연속된 extent의 개수도 저장을 하여, 디스크 탐색 시간을 최소화 했을 뿐만 아니라, 저장 공간도 절약하였다.

직접 레퍼런스는 멀티미디어 extent의 위치를 저장(포인터) 하고 있다. 각각의 직접 레퍼런스와 함께 연속 extent의 개수도 함께 저장을 한다. 이 방식은 처음으로 [21]에서 제안되었다.

이러한 방식의 도입으로 인해 저장 공간의 절약 뿐만 아니라, 디스크 검색 시간도 상당히 줄일 수 있다. 만약 멀티미디어 파일이 커서 직접 레퍼런스로 가리킬 수 있는 extent의 한계가 넘어가면, 간접 레퍼런스에서 가리키는 extent 앞에 레퍼런스 블록을 두어

서 이 레퍼런스에 다른 멀티미디어 extent의 위치를 저장한다. 직접 레퍼런스로 저장할 수 있는 최대 파일 사이즈는 (extent의 크기) * (직접 레퍼런스 #1의 연속 개수 + 직접 레퍼런스 #2의 연속 개수 + ... + 직접 레퍼런스 #12의 연속 개수) 이다.

i-node 블록은 12개의 직접 주소(direct reference)와 3개의 "통합" 일차 간접 주소필드를 갖는다. 간접 레퍼런스에서는 멀티미디어 extent 의 일정 부분을 레퍼런스 블록으로 사용하여, 레퍼런스 블록과 멀티미디어 자료가 인접하게 하였다. 이것은 간접 레퍼런스를 사용하더라도 디스크 헤드의 움직임을 최소화 하기 위해서다.

디스크 단편화 현상이 심화되어 extent를 연속적으로 배치할 수 없다 할지라도, extent의 크기 * 12300 (= 직접레퍼런스 12개 + Indirect Block 당 512개*8*간접레퍼런스 3개) 까지의 파일 사이즈를 저장할 수 있다.

2. 논리적 파일 저장단위

멀티미디어 자료의 경우 파일의 물리적 기본단위와 의미적 기본 단위가 다르다. 물리적 기본단위는 블록이고, 의미적 기본단위(Semantic Data Unit 혹은 Logical Data Unit)는 압축된 파일의 한 프레임이나 오디오 샘플이다. 화면을 재생하고, 편집하고, VCR의 되감기, 빨리돌리기 등의 다양한 동작 등은 모두 의미적 기본단위에 의해 이루진다. 따라서, 멀티미디어 서비스를 지원하기 위한 파일 시스템의 경우는, 파일을 논리적 단위의 집합으로 처리함으로써 많은 장점을 얻을 수 있다.

본 논문에서 개발한 파일 시스템에서는 멀티미디어 데이터를 의미적 기본단위로 저장한다. 이것은 인터랙티브한 사용자 요청을 매우 효과적으로 처리할 수 있다. 예를 들어, 사용자가 특정 배속(예, 2배속, 3배속, 뒤로 보기 등)의 속도를 요청 하거나 프레임을 건너 뛰어서 읽을 때, 특정 시간부터 영화를 보려 할 때 유리하다.

V. 스트리밍을 위한 운영체제 커널

VFS(Virtual File System)은 대부분의 파일 시스템이 공통적으로 가지고 있는 개념을 추출하여 개발된 파일 시스템 모델이다. HERMES 파일 시스템은 VFS 하부에 위치하도록 구현되었으며, 따라서 응용프로그램은 VFS에서 제공하는 API(예, open, read, write,close 등)를 이용하여 파일을 사용할 수 있다. 즉, 스트리밍 응용 프로그램이 파일시스템의 구조에 대해 독립적으로 존재할 수 있다. 하지만, 스트리밍 응용에 적합하게 시스템의 각종 자원들을 사용하기 위해서는 운영체제 수준에서 VFS가 다루지 않는 스트리밍 환경에 특화된 핵심적인 파일 시스템 API들의 사용이 필요하다. 따라서 VFS 이외에 스트리밍에 최적화된 커널 수준의 API를 제공함으로써 스트리밍 소프트웨어가 스트리밍 부하를 위하여 정제된 환경에서 작동하는 것을 가능케 한다. 추가로 개발된 API및 자료구조는 (i)파일을 논리적인 단위로 다루는 기능, (ii) 파일의 QoS 정보를 파일 시스템에게 넘겨주는 기능, (iii)파일의 스트리밍과 관련된 메타데이타를 파일시스템에게 전달하는 기능을 가능케 하는 것을 목적으로 한다. 새로운 시스템 콜은 mp4로 시작한다.

1. 스트리밍을 위한 세부 정보의 전달

mminfo 구조체는 멀티미디어 재생에 관한 속도 및 재생 방향에 관해서 설정을 한다. 파일 시스템 수준에서 디스크 스케줄링 방법을 결정하는 데 사용된다. 초기 재생시의 재생 속도 및 재생 방향을 설정하며, 파일이 정상적으로 열기가 되면 파일 디스크립터(descriptor) 번호가 리턴이 되지만, 에러 발생시에는 -1을 리턴한다. mp4track 구조체는 트랙에 관

```
struct mminfo {
    int direction;
    int speed;
}
struct mp4track {
   unsigned int TrackID;
   unsigned int CreationTime;
   unsigned int ModificationTime;
   unsigned int Duration;
   double TimeScale;
}
```

한 정보를 저장하는 구조체이다. 트랙의 ID, 생성 시간, 변경 시간, 시간 스케일, 트랙의 재생 시간등에 관한 정보를 저장한다.

2. 스트리밍 전용 커널 서비스

최적화된 환경의 스트리밍을 위하여 HERMES 파일 시스템은 다음과 같은 커널 서비스를 제공한다. 이들 커널 서비스는 스트리밍 서버가 보다 효율적으로 시스템의 자원을 관리할 수 있게 한다.

Ⅵ. 성능 평가

HERMES 파일 시스템은 파일 메타 데이터 구조, 블록 비트맵, inode 비트맵등 모든 기능이 처음 개념설정, 설계부터 구현까지 자체적으로 개발되었다. 파일 시스템 프로토타입은 리눅스 커널 2.4에서 구현되었다. 개발된 파일 시스템의 성능과 기존의 유닉스 계열의 파일 시스템의 성능을 비교하는 실험을 수행하였다. 실험에 사용된 컴퓨터는 Pentium III 746MHz 듀얼 CPU를 가진 서버로서 IBM DPSS-309170M SCSI 디스크 4개가 장착되어 있

```
int mp4open(const char *pathname, int flags, struct mminfo *mminfo);
   특정 경로명(pathname)에 지정된 파일을 연다. Flags 는 열기 모드에 관해서 설정을 한다.
int mp4close(int fd)
   열려진 파일을 닫는다.
int mp4GetMovieIOD( int fd, char* initialOD, int* plodLength)
   MPEG-4 파일의 IOD(Initial Object Descriptor)를 검색하여 initalOD에 저장을 하고,
   그 크기를 plodLenght 에 저장을 한다.
int mp4GetTrackCount(int fd, int *nCount)
   MPEG-4 파일의 총 트랙수를 검색하여 nCount에 저장을 한다.
 int mp4GetMovieTrack( inf fd, unsigned int trackID, struct mp4track *pTrack)
   MPEG-4 파일의 특정 트랙의 트랙 정보를 얻어 온다.
 int mp4read(int fd, unsinged int trackID, unsigned int sampleNo, void *buffer, size_t size)
   특정 트랙의 특정 샘플을 읽는다. 이 샘플에는 멀티미디어의 LDU가 저장이 되어 있다. Buffer
   에는 LDU의 내용이, size는 LDU의 크기가 저장이 된다.
 int mp4write(int fd, unsigned int trackID, unsigned int sampleNo,  void *buffer, size_t size)
   특정 트랙의 특정 샘플을 파일에 저장을 한다. 저장할 내용은 buffer에 저장이 되며, 크기는
   size에 저장이 된다.
```

다. 3번째 디스크에 Ext2 파일 시스템을 구성하고, 4번째 하드 디스크에 HERMES 파일 시스템 파티션을 설정하였다. 표2에서는 성능평가 실험에 사용된 디스크의 물리적 특성을 요약하고 있다.

실험은 2가지 측면에서 진행되었다. 첫 번째는 대용량 파일을 순차적으로 읽는데 걸리는 시간과 시스템 사용율을 측정해 보았고, 두 번째는 동시에 여러 개의 파일을 읽었을 때의 응답 시간을 측정하였다.

1. 대용량 파일의 순차적 읽기 시간 측정

약 570MB의 파일을 각각의 파일 시스템에 7개씩 쓴 후에, 이 파일들을 순차적으로 읽었을 때의 시간을 측정하였다. cat 명령어를 실행하여 읽었을 때의 시간과 CPU 사용율을 측정하였다. Ext2 파일 시스템에서도 파일은 단편화 없이 디스크의 바깥쪽 실린더부터 순차적으로 배치하였다. 570 Mbyte의 파일은 ext2 파일 시스템에서는 12 개의 단일 간접 주소, 143개의 이중 간접 주소 블록으로 이루어 진다.

실험의 신뢰성을 확보하기 위하여 같은 실험을 7회 반복하였다. 그림 7은 실험 결과의 평균값을 나타낸 것이다. 대용량 파일을 읽었을 때 MPEG-4 멀티미디어 파일 시스템이 적은 시스템 영역 점유 시간을 보인다. CPU 사용율이 현저하게 낮은 것을 볼 수 있다. CPU 사용율이 낮은 이유는 Ext2 파일 시스템에 비하여 HERMES 파일 시스템의 블록 맵 구조가 상대적으로 단순하여 데이터 블록의 위치를 계산하는 소요되는 CPU 오버헤드가 작기 때문이라고 예측된다. 두 파일 시스템 모두 파일 시스템의 초기화 직후에 배치하였기 때문에, 단편화는 발생하지 않는다. 따라서, 성능의 차이는 ext2에서 발생하는 단편화에서 연유한다고 볼 수는 없다. 파일 시스템의 응답시간에서 HERMES파일 시스템이 좋은 성능을 보이는 것은, 데이터 블록을 읽을 때, extent의 일부분을 간접주소 블록으로 사용하는 방식을 사용함으로써, 멀티미디어 데이터 블록의 지역적 국부성(spatial locality)을 극대화 시킨 결과로 판명된다.

2. 스트리밍 부하 실험

멀티미디어 스트리밍 환경을 구현하기 위하여 파일

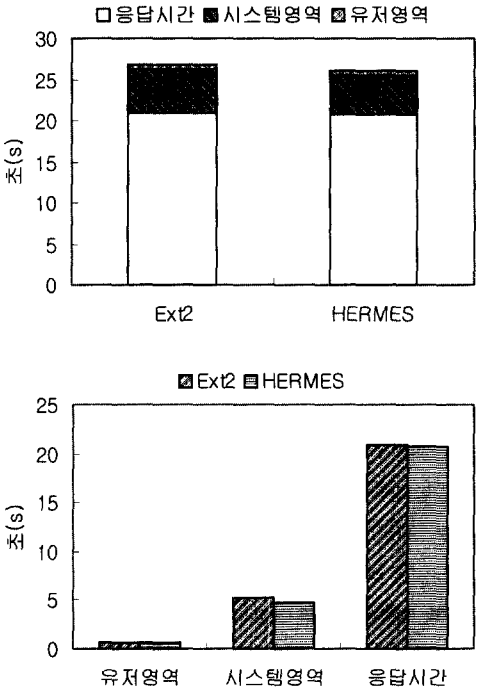


그림 7. 순차적 읽기 성능 측정 비교

표 2. SCSI HDD 성능 명세표

용량(Capacity)	9.1GB	인터페이스	Ultra160 SCSI
섹터 크기	512 bytes	회전 속도	7200 RPM
미디어 전송 속도	248 -400 Mbits/sec	평균 탐색 시간	6.8ms
트랙간 탐색 시간	0.6ms	전체 트랙 탐색 시간	15.0ms

시스템에 스트리밍 부하를 가할 수 있는 환경과 시뮬레이션 프로그램을 제작하였다. 약 50MB크기의 파일 50개를 Ext2 파일 시스템과 MPEG-4 멀티미디어 파일 시스템에 생성하였다. 시뮬레이션 프로그램은 각 파일 시스템의 멀티미디어 파일을 64K바이트씩 읽으며, 동시에 여러 개의 쓰레드가 각기 다른 파일을 순차적으로 읽는다. 동시에 읽는 파일의 개수를 증가 시키면서 쓰레드가 단일 파일을 순차적으로 읽는데 소요되는 총 시간을 측정하였다.

동시 진행중인 스트리밍 프로세스가 증가 할수록 HERMES 파일 시스템의 I/O 지연 시간이 ext2 파일 시스템의 지연 시간보다 작게 나타난다. 이것은 동시에 읽는 스트리밍 세션이 증가할수록 기존 리눅스 파일 시스템에서 two-step, three-step의 indirect 블록을 읽는데 걸리는 오버헤드(overhead)가 I/O 지연 시간의 상당 부분을 차지하기 때문이라고 판단된다. 이와 반대로 MPEG-4 멀티미디어 파일 시스템은 간단한 파일 시스템 구조 및 블록 그룹을 제거하여 디스크 탐색 시간을 최소화 함으로써, 디스크 부하가 높아질수록 상대적인 효율성이 증가한다.

3. 응답시간의 분산

스트리밍 서비스의 핵심은 스트리밍 사용자에게 데이터 블록을 "일정" 간격, 정규적으로 배달하는 것이다. 따라서, 파일 시스템이 입출력 요청을 균일한, 내지는 비교적 적은 오차범위의 응답시간 내에 처리할 수 있다면, 스트리밍 서버의 효율성을 극대화 할 수 있다. 이를 평가하기 위하여 64Kbyte의 데이터 블록을 읽었을 때의 읽기 명령 응답시간을 분석하였다.

그림 9,10에서 볼 수 있는 바와 같이, HERMES 파일 시스템에서 입출력 요청의 응답 시간이 비교적 적게 걸릴 뿐만 아니라 응답 시간의 변이도 작은 것을 볼 수 있다. 그림 10은 스트림수 증가에 따른

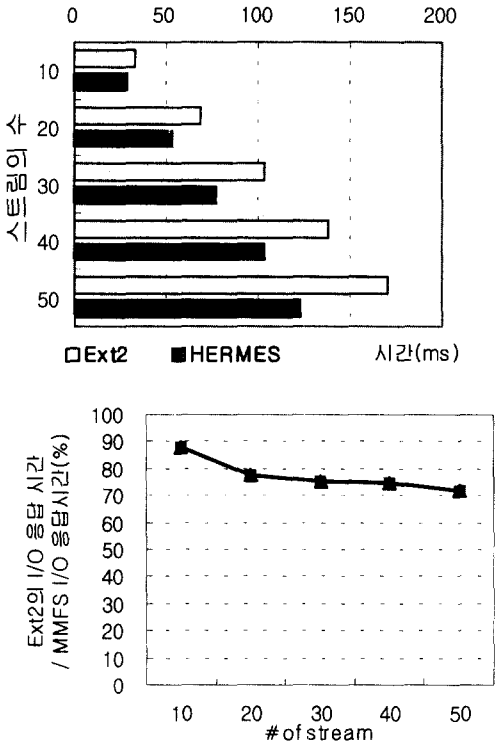


그림 8. 스트림의 수의 증가에 따른 순차 읽기 소요 시간

Ext2와 HERMES 파일 시스템의 분산(σ^2)을 나타내고 있다. 이것으로 HERMES가 안정적으로 파일 읽기 명령을 수행한다고 볼 수 있다.

Ⅵ. 결론

정보가전은 컴퓨터와 가전제품이 융합된 새로운 형태의 기기이다. 전통적인 컴퓨터가 가지는 연산/저장/처리능력과 전통적인 가전제품이 가지는 사용 용이성, 신뢰성, 견고성 및 가격경쟁력등의 특성을 동시에 만족해야하는 것이 정보가전이다. 본 연구에서는 멀티미디어 처리를 위한 정보가전 기기의 요구사항을 정의하고 이를 효율적으로 처리할 수 있는 파일 시스템 설계에 대한 내용을 소개하였다. 마지막으로

멀티미디어 정보가전용 파일 시스템의 실제 개발 사례를 소개하였다. 멀티미디어 자료가 점차 대중화 됨에따라 이를 효과적으로 처리할 수 있는 시스템 컴포넌트의 개발 및 핵심 소프트웨어 개발이 차세대의 정보가전 기기시장에서 우위를 선점하는 중요한 관건이라 할 수 있겠다.

※참고문헌

[1] Want, R.: Borriello, G. "Survey on information appliances", *IEEE Computer Graphics and Applications, Volume: 20 Issue: 3* , May-June 2000 Page(s): 24 -31

[2] http://www.tivo.com

[3] Mon-Song Chen, Dilip D. Kandlur, and Philip S. Yu. "Optimization of the grouped sweeping scheduling(gss) with heterogeneous multimedia streams", In *ACM Multimedia* '93, pages 235-242, 1993.

[4] 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.

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

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

[7] 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.

[8] Lougher P. and Shepherd D. "The design of a storage server for continuous media", *The Computer Journal*, 36(1): 32-42, 1993.

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

[10] Renu Tewari and Richard King and Dilip Kandlur and Daniel M. Dias. "Placement of Multimedia Blocks on Zoned Disks", In *Proceedings of SPIE West* '96, 1996.

[11] Guido Nerjes, Peter Muth, and Gerhard Weikum. "Stochastic service guarantees for continuous data on multi-zone disks", In *Proceedings of the 16 th Symposium on Principles o Database Systems, Tucson*, Arizona, 1997.

[12] Ghandeharizadeh, Shahram and Kim, S. and Shahabi, C. "Continuous Display of Video OBjects Using Multi-Zoned Disks", Technical report, University of Southern California, 1995.

[13] P.K.C. Tse and C.H.C. Leung. "Improving multimedia systems performance using constant-density recording disks", *Multimedia Systems*, 8(1): 47-56, January 2000.

[14] Youjip Won and K. Cho, "Minimizing the Impact of Starting New Session in Zoned Disk", *In Proc. of ACM Multimedia Conference '01*, Sep. 30 - Oct. 4, 2001, Ottawa, Ontario, Canada

[15] Youjip Won and Y.S. Ryu. "Handling Sporadic Tasks in Multimedia File System", In *In Proc. of ACM Multimedia Conference '00, Los Angelses, CA, USA, Oct.* 2000

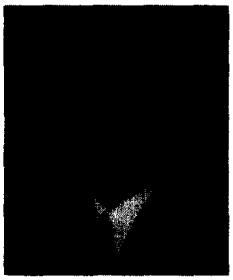
[16] Prashant Shenoy, Pawan Goyal, Harrick M. Vin. "Architectural considerations for next generation file systems", *Proceedings of the seventh ACM international conference on Multimedia*, 1999

[17] 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.

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

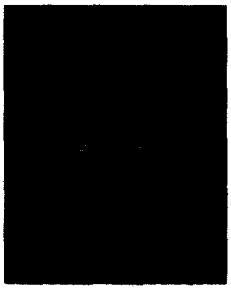
[19] William J. Bolosky, Robert P. Fitzgerald, John R. Douceur. "Distributed schedule

management in the Tiger video fileserver", *Operating Systems Review (ACM)* 1997
[20] T.N. Niranjan, Tzicker Chiueh, Gerhard A.Schloss. "Implementation and evaluation of a multimedia file system", *International Conference on Multimedia Computing and Systems -Proceedings* Jun 3-6 , 1997
[21] Chuanbao Wang, Vera Goebel, Thomas Plagemann. "Techniques to increase disk access locality in the Minorca multimedia file system", *Proceedings of the seventh ACM international conference (part 2) on Multimedia (Part 2)*, 1999
[22] Rosenblum, M., "The Design and Implementation of a Log-Structured File System", *Kluwer Academic Publishers*, 1995.
[23] Michael Beck, Harald Bohme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus, Harold Bohme. "Linux Internals", Addison-Wesley Pub Co. ISBN: 0201331438
[24] http://www.apple.com/
[25] ISO/IEC JTC1/SC29/WG11 N2611. MPEG-4 systems version 2 WD 5.0, December 1998.
[26] 정보 가전용 멀티미디어 파일 시스템 개발에 관한 연구, 연구보고서, 한국 전자통신 연구원



원 유 집

1990년 서울대학교 전산과학과(학사), 1992년 서울대학교 전산과학과(석사), 1997년 미네소타 주립대학교(박사), 1997년-1999년 Performance Analyst, Intel Corp. 1999년 - 현재 한양대학교 공과대학 전자전기컴퓨터 공학부 교수 〈관심분야〉 멀티미디어 시스템, 초고속 네트웍, 성능평가이론, 데이터 베이스, 분산시스템



박 진 연

2000년 한양대학교 전자전자통신전파공학과(학사), 2000년 - 현재 한양대학교 전자통신전파공학과 석사 재학중 〈관심분야〉 멀티미디어 시스템, 운영체제, 데이터베이스, 인터넷