특집논문-01-6-3-05

인터넷 홈서버를 위한 스트리밍 전용 파일 시스템

박 진 연*, 송 승 호*, 진 종 현*, 원 유 집*, 박 승 민**, 김 정 기**

File System Support for Multimedia Streaming in Internet Home Appliances

Jinyeon Park*, Seungho Song*, Jonghyun Jin*, Youjip Won*, Seungmin Park** and Jungki Kim**

요 약

최근 급속도로 확장되고 있는 인터넷을 통한 동영상 서비스와 이미 상용 서비스가 시작된 디지털 방송 서비스 등으로 인하여, 가전제품에서 디지털 동영상을 처리하는 데에 관한 관심이 매우 높아지고 있다. 텍스트 기반이나 이미지 기반 데이터와 달리 멀티미디어 데이터는 정보의 출발점으로부터 미리 정해진 시간까지 작업의 목적지에 도달하지 않으면 원래 전달하고자 했던 자료의 의미를 제대로 전달할 수 없다. 멀티미디어 스트리밍 전용 시스템은 데이터를 정해진 시간까지 목표에 전달하는 것을 궁극적인 목표로 하고 설계되어야 한다. 이러한 시간적 제약성 때문에, 멀티미디어 스트리밍 응용은 많은 디스크 대역폭을 필요로 하고, 파일 시스템에 많은 부하를 가하게 된다. 기존에 사용되는 대부분의 범용 파일 시스템은 스트리밍 부하의 특성인 순차적 읽기에 적합하게 설계되어 있지 않다. 따라서, 스트리밍 환경을 위해서 사용되기 위해서는 많은 개선의 여지를 가지고 있다. 본 논문에서는, 멀티미디어 스트리밍 부하의 특성을 분석하고, 이에 최적화된 파일 시스템을 설계한다. 설계된 파일 시스템을 구현하여 범용 파일 시스템과의 성능평가 실험을 수행하였다. 성능 평가 결과 본 논문에서 제시하는 파일 시스템이 순차적 파일 접근의 경우 기존의 유닉스 계열에서 제공되는 파일 시스템 보다 월등한 성능을 보이는 것을 관찰할 수 있었다. 이와 더불어 효율적으로 동영상 자료를 접근할 수 있도록, MPEG-4 압축방식에 특화된 커널 수준의 파일 시스템 API를 제안한다.

키워드 : 멀티미디어, 스트리밍, MPEG-4, 파일 시스템, UFS, 디스크 스케줄링, 운영체제

Abstract

Due to recent rapid deployment of Internet streaming service and digital broadcasting service, the issue of how to efficiently support streaming workload in so called "Internet Home Appliance" receives prime interests from industry as well as academia. The underlying dilemma is that it may not be feasible to put cutting edge CPU, boards, disks and other peripherals into that type of device. The primary reason is its cost. Usually, Internet Home Appliances has its dedicated usage, e.g. Internet Radio, and thus it does not require high-end CPU nor high-end I/O subsystem. The same reasoning applies to I/O subsystem. In Internet Home Appliances dedicated to handle compressed moving picture, it is not equipped with high end SCSI disk with fast rotational speed. Thus, it is mandatory to devise elaborate software algorithm to exploit the available hardware resources and maximize the efficiency of the system. This paper presents our experiences in the design and implementation of a new multimedia file system which can efficiently deliver the required disk bandwidth for a periodic I/O workload. We have implemented the file system on the Linux operating system, and examined itsperformance under streaming I/O workload. The results of the study show that the proposed file system exhibits superior performance than the Linux Ext2 file system under streaming I/O workload. The result of this work not only contribute to advance the state f art file system technology for multimedia streaming but also put forth the software which is readily available and can be deployed.

Ⅰ. 개 요

1. 연구동기

디지털 방송의 상용 서비스가 가시화 되고, 네트웍 대역폭의 확장으로 인해 가정에서도 인터넷 방송을 손쉽게 접할 수 있게 됨에 따라, 동영상을 저장하고 처리하는 정보가전형 스트리밍 서버가 새롭게 관련자들의 주목을 받고 있다. 이와 더불어 TV-anytime[24] 기술도 사용자측에서 동영상을 방송시간에 녹화한 다음 원하는 시간에 재생하는 개념이기 때문에, 저가형 단말기에서 멀티미디어 동영상을 재생하는 것이 필수적이다. 디지털 방송 서비스의 시작, 네트웍 대역폭의 확대, TV-anytime 등장으로 인하여 보다 경제적으로 구성된 정보가전기기(STB[25], PVR[26], 홈서버)에서 가내의 구성원에게 일련의 스트리밍 서비스를 제공하는 것이 매우 중요한 이슈로 등장하고 있다. 현재 급속하게 확산되고 있는 인터넷 정보가전은 일반적인 데스크 탑 PC보다 낮은 성능의 하드웨어로 구성되어 있는데, 이의 가장 큰 이유는 제품의 가격 경쟁력이라 할 수 있겠다. 따라서, 이에 핵심이 되는 사항, 즉 보다 정교하게 시스템의 각종자원을 모델링하여 가용 시스템 자원의 사용율을 극대화하여 효율적으로 멀티미디어 서비스를 제공할 수 있는 기법과 소프트웨어를 개발하는 것은 매우 중요한 사안이다.

원격 교육, 주문형 비디오 등 인터넷을 통한 양방향 멀티미디어 서비스에서 적은 양의 시스템 자원을 효율적으로 사용하는 것은 스트리밍 시스템의 디자인에 있어서 중요한 핵심 중에 하나이다. 이를 위해서는 멀티미디어 재생의 지원을 위한 디스크 대역폭, 디스크의 용량, 입출력 버스의 대역폭 에 대한 정교한 모델링이 수반되어야 한다.

2. 스트리밍 부하의 특성

Ftp, http, telnet 등과 같은 텍스트 기반 입출력 요청과 스트리밍 입출력 요청의 차이점 중 하나는 멀티미디어 데이터의 전송을 위해서는 저장 장치의 대역폭을 "보장"해야 한다는 것이다. 텍스트 기반의 데이터 전송동작(ftp, http, telnet등)은 데이터를 목적지에 "정확히" 전달하는 것이 최종 요구 조건이다. 보다 신속한 데이터의 전송을 통해 사용자에게 좋은 서비스를 제공할 수 있다는 점에는 의문의 여지가 없으나, 도착하는 시간에 따라 자료자체의 의미가 왜곡된다거나 유실되는 현상은 발생하지 않는다. 이와 달리, 스트리밍 서비스의 경우, 각 단위 자료(패킷 혹은 프레임)가 정해진 시간 내에 목적지에 전달되는 것이 강조된다. 자료의 적시성이 자료의 정확성보다 강조되는 경우라 할 수 있겠다. 정해진 시간 내에 요청된 자료가 도착하지 않을 경우, 사용자 측에 화면의 일그러짐, 끊김 등의 현상이 발생하며, 이로 인해 자료의 의미가 사용자에게 제대로 전달되지 못하는 경우가 발생한다. 따라서, 서버가 다수의 사용자에게 스트리밍 서비스를 제공하는 경우 각각의 사용자를 위해 일정량의 시스템 자원(CPU 사용율, 네트웍 대역폭, 기억장치의 대역폭, 주기억장치 버퍼)들을 예약하는 것이 필수적이라 하겠다.

멀티미디어 세션에게 연속적인 자료의 흐름을 제공하기 위해서는, 먼저 서버측 디스크로부터 해당 데이터를 정기적으로 읽어야 한다. 디스크로부터 자료를 효율적으로 읽기 위해서는 (i) 자료를 읽는 과정에서 발생하는 disk arm의 이동거리가 짧고, (ii) 데이터 블록의 위치를 찾는데 필요한 파일 메타데이타의 연산량이 작아야 한다. 디스크에 파일의 블록들을 배치하는 방식과 파일 메타 데이타의 구조 등은 파일 시스템에서 정의하는 사항이다. 따라서, 스트리밍 서버의 효율성을 극대화 하기 위해서는 스트리밍 부하에 최적화된 파일 시스템을 설계하는 것이 가장 중요한 사안 중의 하나인 것이다.

데이터 데이터 블록을 디스크로부터 읽는 데 소요되는 시간을 절약할 수 있다면, 해당 스트리밍 서버는 보다 많은 멀티미디어 스트림을 동시에 서비스할 수 있게 되는 것이다. 디스크 입출력 대기시간을 유발하는 중요한 요소가 검색 오버헤드(seek overhead)이다. 특히 스트리밍 부하와 같이 파일을 순차적으로 접근하는 부하의 경우에는 검색에 따른 시간을 극소화 시키는 것이 절대 중요하다. 스트리밍 전용 파일 시스템의 디자인에서, 데이터 검색에 필요한 디스크 탐색(disk seek) 오버헤드를 최소화 시키는 것이 매우 중요한 사항중의 하나이다.

유닉스 파일 시스템(Unix File System, UFS)이 파일 시스템의 발전역사에서 매우 중요한 위치를 차지하고 있는 것은 누구도 부정할 수 없는 사실이다. 유닉스 파일 시스템의 근본 설계 철학은 텍스트 기반의 비교적 작은 파일을 효과적으로 다루는 동시에 매우 큰 파일까지도 처리하는 것이다. 유닉스 파일 시스템에서 파일은 편향된 다진 트리(skewed n-ary tree)의 leaf에 데이터 블록들이 존재하는 형태이다. 트리의 최대 깊이는 3이다. 이

* 한양대학교 전자전기컴퓨터 공학부
Division of Electrical and Computer Engineering Hanyang University
** 한국전자통신연구원 정보가전연구부
Electronics and Telecommunications Research Institute

런 형태로 구성된 파일을 순차적으로 접근할 경우, leaf node를 검색하는 과정에서 트리의 내부 노드들이 반복적으로 방문 된다. 때문에, 유닉스 계열 파일 시스템의 구조는 스트리밍 환경에 사용되기 위해서는 많은 개선의 여지를 포함하고 있다. 특히, 정보가전과 같이 상대적으로 낮은 성능의 디스크가 사용되는 경우, 스트리밍 부하에 최적화된 파일 시스템을 사용함으로써 디스크의 하드웨어적 성능의 한계를 극복하는 것이 필수적인 사항이라 하겠다.

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 다루는 운영체제 수준에서의 접근방법을 제시한다. 압축과정에서 제시하는 각종 정보들을 효과적으로 사용하는 시스템 모듈들을 정의함으로써 스트리밍 서버의 효율성을 극대화 시키고자 하는 것이다.

3. 관련 연구

멀티미디어 서버 설계에 관련된 다수의 연구결과가 최근에 발표되고 있다[1][2][3]. 멀티미디어 서비스를 위해서는 멀티미디어 데이터를 정해진 시간 내에 전달하기 위하여 디스크 대역폭의 일정량을 멀티미디어 서비스를 위하여 확보 해야 한다. 핵심 논의 사항은 디스크로부터 프레임을 읽을 때, 연속성을 어떻게 제공해 주는가 하는 것이었다. 특히 하나의 서버가 다수의 스트림을 지원할 경우 이들간의 입출력 작업을 어떤 식으로 스케쥴링 하는가에 관한 이슈가 많이 연구되었다. 단일 디스크 관련 이슈와는 별도로 멀티미디어 파일 서버를 위한 디스크 서브 시스템에 관해서도 많은 연구가 있었다[4][5]. [6][7]에서는 디스크 스케줄링 알고리즘을 디스크 배열(array)까지 확장시켰다. 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[8].

현재의 디스크 드라이브 기술은 zoning 기술을 적용하고 있으며, 이 기술은 바깥쪽 실린더가 안쪽 실린더보다 많은 수의 섹터(sector)를 가지고 있는 것이다. Tewari외[9]는 구역 기반(zoned disk) 디스크의 멀티미디어 파일 배치 문제를 연구하였다. 멀티미디어 파일 시스템을 위한 디스크 스케줄링과 zoning에서의 디스크 전송률 변화에 관한 이슈를 구체화한 다수의 연구가 진행되었다[10][11][12][23]. 특히 Nerjes외 [10]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 이루어졌다[13]. Shenoy외 [14]는 미디어 타입에 맞게 최적화되어 분리된 파일 시스템 파티션보다 1개의 파일 시스템 프레임 워크 안에서 이기종간의 workload를 효과적으로 다룰 수 있는 것이 가능하다고 결론지었다. [15]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[16][17]. MMFS[18]은 prefetching, 상태기반 캐싱(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 [19]은 (1) MOSA라는 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. 멀티미디어 데이터 스케쥴링

스트리밍 세션에게 지터(jitter)없는 동영상 서비스를 제공하기 위해서는 데이터 블록이나 프레임이 "일정간격"으로 플레이어에 전달되어야 하며, 이를 위해서는 일정량의 디스크 대역폭이 세션에 할당되어야 한다. 디스크 헤드의 움직임과 I/O 서브 시스템의 작동이 근본적으로 비동기적이다. 멀티미디어 재생 작업의 동기적 성질과

I/O 서브 시스템 작동의 비동기적인 성질의 차이점을 상호 보완하기 위하여 각 세션에 일정량의 메모리 버퍼를 할당해야만 한다. 이를 본 논문에서는 "동기화 버퍼"라고 부르기로 하겠다. 다수의 세션에게 연속적으로 멀티미디어 데이터를 전달하기 위하여, '주기' 기반의 입출력 스케쥴링 기법이 사용된다. '주기' 기반 스케쥴링 기법에서는 시간을 주기로 분할하고, 각 주기마다 미리 정해진 분량의 데이터를 스트리밍 세션에게 공급한다. 주기의 길이는 서비스 하고자 하는 세션의 수 혹은 세션들의 총 대역폭에 의해 결정된다. Won 외[8]에서 이제까지 연구된 '주기'기반 디스크 스케줄링 방법을 자세히 설명을 하고 있다.

스트리밍 서버가 일련의 세션들에게 멀티미디어 서비스를 제공할 수 있는 조건을 다음과 같이 요약할 수 있다. (i) 각 스트림에게 주기 마다 전송되는 데이터 양은 한 주기동안 재생하기에 충분한 양이어야 하고, (ii) 한 주기에서 모든 세션이 필요로 하는 데이터를 읽는데 걸리는 총 시간은 주기의 길이보다 작아야 한다. 조건 (i)은 스트림에서 기아현상(starvation)이 발생하지 않도록 하는 것을 나타낸다. 이를 구체적으로 전개하면 수식 1과 같은 형태의 공식을 도출할 수 있다. 이것은 서버가 m개의 스트림을 지원할 경우, 각 스트림이 매 라운드마다 필요로 하는 데이터 블록의 양을 나타낸다. 이에 대한 구체적인 사항은 Won 외[8]를 참조하기 바란다. , ni 는 한 주기동안 스트림 i 에게 제공되는 데이터 블록의 개수, m은 전체 스트림의 개수, b 는 I/O 유닛의 크기, ri 는 i번째 스트림의 재생 대역폭을 나타낸다. Bmax 는 디스크의 최대 전송 대역폭을 나타낸다.

$$n_i \geq \frac{O(m)r_i}{\frac{b}{B_{\max}}\left(B_{max} - \sum_{i=1}^{m} r_i\right)} \qquad (1)$$

수식 (1)에서 볼 수 있는 바와 같이 버퍼 사이즈가 $O\left(\frac{1}{1-\rho}\right)$의 형태를 가진다. ρ 는 디스크 사용율, 즉 $\frac{\sum_{i=1}^{m} r_i}{B_{\max}}$ 을 의미한다. 여기서 중요한 특성은 디스크 사용율이 100%에 근접할수록, 혹은, 서버에 많은 부하가 걸릴수록, 일련의 스트림을 지원하는 데 필요한 버퍼의 크기가 폭발적으로 증가한다는 것이다. 또한 디스크 헤드의 움직임으로 발생하는 오버헤드, $O(m)$, 역시 동기화 버퍼의 공간에 영향을 미치고 있다. 디스크 오버헤드는 파일 시스템 구조, 파일 배치 전략, 디스크 스케줄링기법등에 의해 결정된다.

Ⅲ. 유닉스 파일 시스템과 스트리밍

스트리밍 서비스를 위한 구체적인 설계방법을 기술하기에 앞서, 먼저 잘 알려진 유닉스 계열의 파일시스템을 소개하고, 유닉스 파일 시스템의 특성과 스트리밍 부하의 관계를 언급하고자 한다. 본 절에서는 리눅스 운영체제에서 가장 많이 사용되는 Ext2 파일 시스템을 기반으로 하여 소개한다. 유닉스 파일 시스템은 전형적으로 디스크 파티션 관리를 위한 정보와 데이터를 엄격하게 분리를 하고 있다. 파일을 위한 정보를 따른 분리해서 저장하고 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.

1. 유닉스 파일 시스템에서 inode 의 구조

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 위치를 나타낸다. 블록의 주소는 4바이트이다. 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(12 직접 주소*4K바이트 데이터 블록). 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 트리구조 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.

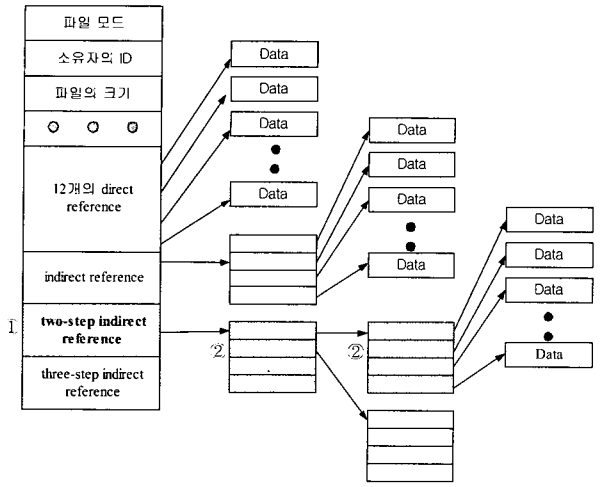

그림 1. Ext2 파일 시스템의 구조
Fig. 1. Structure of Ext2 file system

2. 유닉스 파일 시스템에서 멀티미디어 파일

압축 기법을 사용하더라도 1시간분량의 영화는 수백 메가에서 수 기가까지의 디스크 용량을 필요로 한다. MPEG-1으로압축된 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 1에서의 ①) 과 2개의 간접 주소 블록(그림 1에서의 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. 이점은 본 연구를 통해서 제안된 HERMES 파일 시스템과 Ext2파일 시스템을 비교 분석하는 부분에서 구체적으로 언급하도록 하겠다.
- 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 데이터 블럭을 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[20]. 본 연구에서 제안하는 파일 시스템은 이러한 문제를 성공적으로 해결하였다.
- 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. 본 논문에서는 데이터 블록을 연속적으로 배치하면서 이를 지정하기 위한 주소 블록의 오버헤드를 최소화 하는 기법을 제시한다.

3. 유닉스 시스템의 파일 배치 전략

리눅스, 솔라리스, NetBSD 등 유닉스 호환 운영체제의 파일 시스템의 대부분이 블록 그룹 내지는 실린더 그룹의 개념을 이용하여 데이터 블록의 배치를 최적화시키는 기법을 사용하고 있다. 이들 기법의 궁극적인 목표는 연속한 데이터 블록들을 서로 인접하거나, 이가 불가능할 경우 최대한 가깝게 배치하는 것이다. 리눅스 운영체제에서 가장 보편적으로 사용되고 있는 Ext2 파일 시스템은 블록 그룹(block group) 개념을 도입하여, 파일의 데이터 블록과 inode, 디렉토리 할당을 정교하게 관리한다[21]. 블록 그룹은 일련의 연속된 실린더들의 그룹이다. 블록 그룹은 Ext2 파일 시스템에서 파일의 데이터 블럭을 배치하는 단위라고 볼 수 있다. 블록 그룹의 크기는 파일 시스템 포맷(format)시에 결정되고, 다시 포맷하기 전까지는 변하지 않는다.

Ext2 파일 시스템은 여러 개의 블록 그룹으로 구성이 되어 있으며 각각의 블록은 파일 시스템의 견고성을 유지하기 위한 슈퍼블록(superblock)의 복사본과group descriptor, 블록 비트맵, inode 비트맵, inode 테이블과 데이터 블록을 가지고 있다. 블록 그룹을 사용하여 파일 시스템은 상대적으로 좀더 가까운 실린더(cylinder) 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.

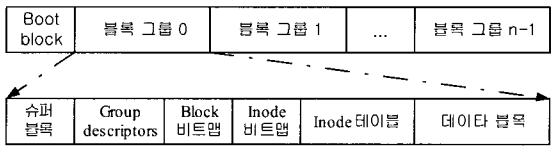

그림 2. Ext2 파일 시스템의 디스크 구조
Fig. 2. Disk layout of Ext2 file system

Ⅳ. HERMES 파일 시스템

1. HERMES 파일 시스템 레이아웃

Ext2 파일 시스템을 포함하여 유닉스 계열의 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. 아래의 그림 4는 본 연구에서 개발한 파일 시스템에서 단일 파티션 상의 레이아웃을 보여주고 있다.

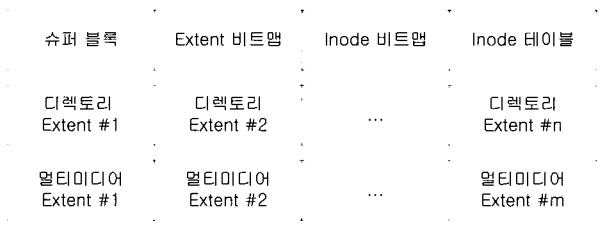

그림 3. HERMES 파일 시스템 레이아웃
Fig. 3. HERMES file system layout

슈퍼블럭(superblock)은 파일 시스템의 전체적인 내용을 저장하고 있다. 디렉토리 extent의개수, 멀티미디어 extent의 개수, inode의 개수, 현재 비어 있는 extent의 개수, extent의 크기, 생성 시간 등의 정보를 저장하고 있다.

Extent는 파일의 가장 기본적인 구성 단위이며, extent는 여러 개의 데이터 블록이 모여서 구성된다. Extent의 크기는 파일 시스템 포맷시에 시스템 관리자가 크기를 설정 할 수가 있다. 멀티미디어 파일은 멀티미디어 extent에 저장이 되며, 디렉토리의내용은 디렉토리 extent에 저장이 된다. 이렇게 멀티미디어와 디렉토리를 따로 분리해서 저장하는 이유는 파일종류에 따라 국부성(locality)을 높여서 디스크 탐색(seek) 시간을 줄이기 위해서다.

Extent 비트맵에는 각각의 extent의 사용 유무를 비트맵 형식으로 저장한다. 새로운 멀티디어 파일이 생성되면, 이 비트맵을 이용하여 사용하지 않는 첫 번째 extent를 찾아가서 저장 하게 된다. 각각의 Inode 는 파일의 메타 정보를 기억하고 있다. 파일 소유자의 ID, 파일 소유자의 그룹 ID, 파일의 모드(예, rwxr-xrw-) 등을 기억하고 있다. 이러한 inode들이 모여 있는 곳이 바로 inode 테이블이다. inode 비트맵에서 각 inode의 사용 유무를 비트맵 형식으로 저장하고 있다. 위와 같은 디스크 배치 전략으로 인해 얻을 수 있는 이점은 다음과 같다.

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

2. HERMES 메타 데이터 블럭의 구조

유닉스 계열 파일 시스템의 i-node 구조는 데이터 용량이 크며, 순차적 접근이 많은 멀티미디어 파일을 위해 적합하지 않기 때문에, inode를 재구성 하였다. 그림 4는 본 연구에서 제안한 멀티미디어 파일 시스템의 inode의 구조를 나타내고 있다. Ext2에서와 같은 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. 이 방식은 [19]에 의해서 제안되었다.

직접 레퍼런스는 멀티미디어 extent의 위치를 저장(포인터) 하고 있다. 각각의 직접 레퍼러스와 함께 연속 extent의 개수도 함께 저장을 한다. Ext2에서는 레퍼런스 한 개가 한 개의 데이터 블록을 가리키고 있어서 많은 레퍼런스 블록이 필요했지만, 위와 같은 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 두어서 이 레퍼런스에 다른 멀티미디어 extent 의 위치를 저장한다. 직접 레퍼런스로 저장할 수 있는 최대 파일 사이즈는 (extent의크기) * (직접 레퍼런스 #1의 연속 개수 + 직접 레퍼런스 #2의 연속 개수 + ... + 직접 레퍼런스 #12의 연속 개수)이다. 간접 레퍼런스에서는 멀티미디어 extent의 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(= 직접 레퍼런스 12개 + Indirect Block 당 512개*8*간접레퍼런스 3개)까지의 파일 사이즈를 저장할 수 있다.

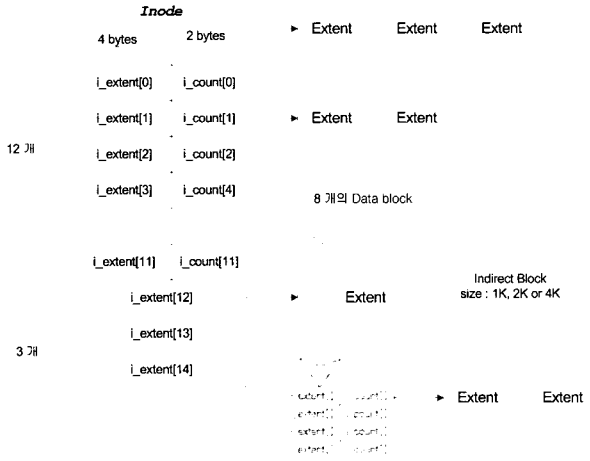

그림 4. inode 레이아웃 과 블록 맵
Fig. 4. Inode Layout and block map

3. 논리적 기본 단위 저장

동영상을 포함한 멀티미디어 자료는 파일의 물리적 기본단위와 논리적 기본 단위가 다르다. 물리적 기본단위는 블록이고, 논리적 기본단위(Semantic Data Unit 혹은 Logical Data Unit)는 압축된 파일의 한 프레임이나 오디오 샘플이다. 화면을 재생하고, 편집하고, 배속 재생 등의 다양한 인터렉티브(interactive)한 동작 등은 모두 논리적 기본단위에 의해 이루진다. 따라서, 멀티미디어 서비스를 지원하기 위한 파일 시스템의 경우는, 디스크의 낭비를 초래한다는 단점은 있지만 파일을 논리적 단위의 집합으로 처리함으로써 위와 같은 장점을 얻을 수 있다. 이를 지원하긴 위한 파일 시스템 API는 5장에서 설명한다.

V. 스트리밍을 위한 커널의 지원

4장에서 제시된 파일 시스템은 각종 포멧의 파일이 저장가능하며, VFS(Virtual File System) 계층의 하부에 위치하도록 설계되었다. 이 접근 방식의 장점은 기존 존재하는 응용 프로그램을 변경없이 그대로 사용할 수 있다는 것이다. 주지하는 바와같이 VFS는 대부분의 파일 시스템이 공통적으로 사용할 수 기능들을 추출하여 가상의 추상화 계층을 구성한 것이다. 따라서, 파일 시스템 개발자와 응용 프로그램 개발자의 분리함으로써 시스템 개발과 사용의 용이성을 제고한다. 그러나, 범용성을 강조함으로써, 특정 화일 형식에 최적화된 기능은 제공하지 않는다. 본 연구의 목적은 홈서버를 위한 위한 스트리밍 파일 시스템의 개발이며, 파일의 형식은 MPEG-4를 목표로 하고 있다. MPEG-4 의 시스템 부분(part 6)에서는 다양한 기능을 제공하는 매우 강력한 형태의 파일형식을 정의하고 있다. 본 논문에서는 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 근간한다. Quicktime 포멧을 기반으로 MPEG-4 표준안에서 추가된 기능을 지원하도록 구성되어있다. 퀵타임 파일[22]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 아톰(atom)으로서, 퀵타임 파일은 이들 QT 아톰들의 집합이다. QT 아톰에는 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 아톰 들은 각각의 트랙에 관한 정보를 기억하고 있다. 아래의 그림 5는 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, 동

기화 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 아톰들은 퀵 타임에서 이미 정의되어 있으나, MPEG-4 Part 6(System Part)에서 정의하고 있는 OD(Object Descriptor), ESD(Elementary Stream Descriptor), BIFS (Binary Format for Scenes)등은 퀵타임 포멧에서는 정의되지 않고 있다. 이를 위하여, MP4파일은 다음과 같은 아톰 포멧을 추가로 정의하고 있다. 이를 요약하면 표 1과 같다.

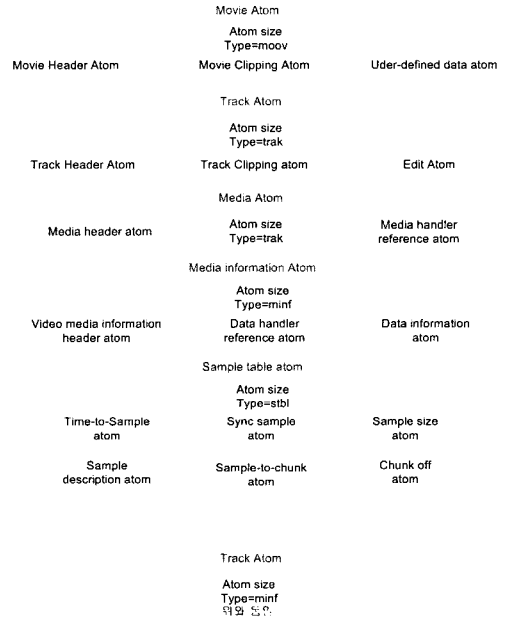

그림 5. Quicktime 파일 구조
Fig. 5. File structure of Quicktime

표 1. MPEG-4 파일에 새로 추가된 아톰들
Table 1. Newly added Atoms in MPEG-4 file

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

2. 스트리밍용 파일 시스템 구조

일반적인 응용프로그램은 VFS에서 제공하는 API를 통하여 파일을 접근한다. 하지만, VFS에서 제공하는 시스템 콜들은 스트리밍 환경에 적합하지 않으므로, 본 연구에서는 VFS가 다루지 않는, 스트리밍 환경에 특화된 핵심적인 파일 시스템 API를 제안한다.

그림 6에서와 같이 스트리밍 파일 시스템에 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.

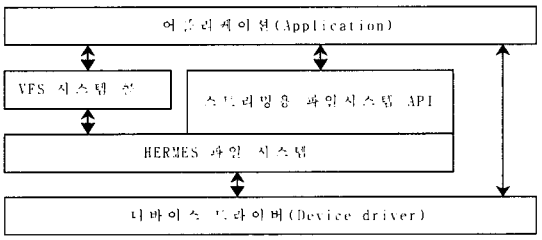

그림 6. 파일 시스템, VFS, 스트리밍용 파일 시스템 API 와의 관계
Fig. 6. Relationship among file system, VFS and file system API for streaming

3. 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 관한 정보를 저장하는 구조체이다. 트랙의 ID, 생성 시간, 변경 시간, 시간 스케일, 트랙의 재생 시간 등에 관한 정보를 저장한다.

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

그림 7. Mp4track 구조체
Fig. 7. Structure of Mp4track

4. 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.

- 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* piodLength)
  MPEG-4 파일의 IOD(Initial Object Descriptor)를 검색하여 initalOD에 저장을 하고, 그 크기를 pIoDLenght 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 논리적 기본단위(Semantic data unit)의 데이터가 저장이 되어 있다. Buffer 에는 논리적 기본단위 데이터의 내용이, size는 논리적 기본단위 데이터의 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 buffer 에 저장이 되며, 크기는 size 에 저장이 된다.

VI. 성능 평가 실험

본 연구에서 개발된 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 있다. 3번째 디스크에 Ext2 파일 시스템을 구성하고, 4번째 하드 디스크에 Intelli 파일 시스템 파티션을 설정하였다. 표2 에서는 성능평가 실험에 사용된 디스크의 물리적 특성을 요약하고 있다.

표 2. SCSI HDD 성능 명세표
Table 2. Specification of SCSI HDD

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

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.

570Mbyte의 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 8은 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.

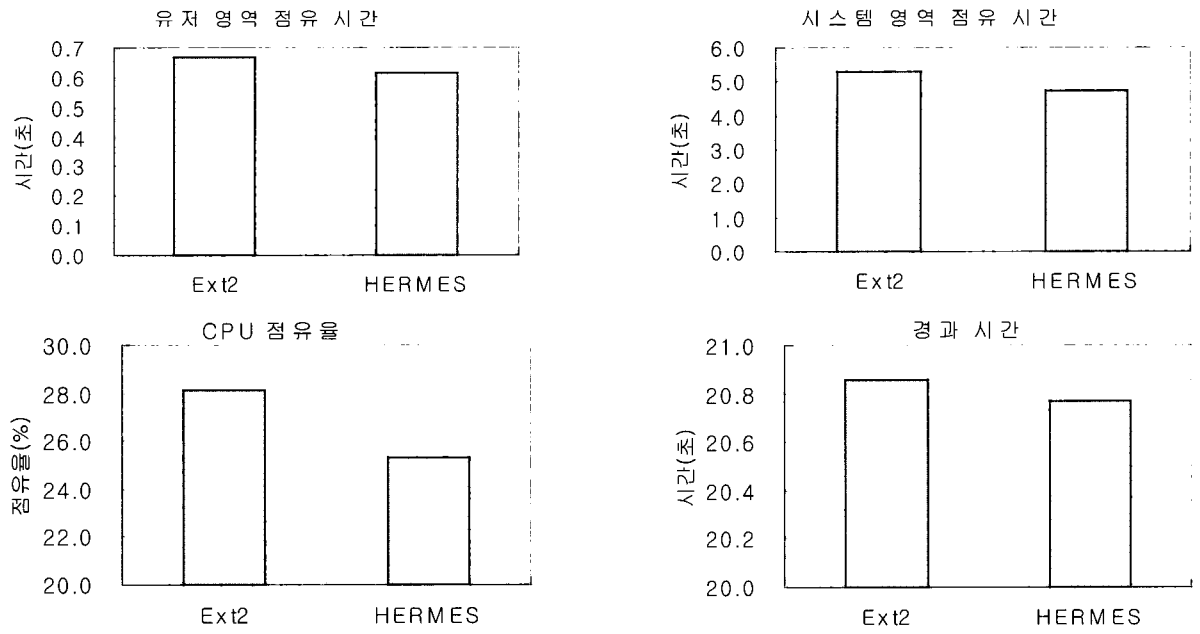

그림 8. 순차적 읽기 시간 측정 비교
Fig. 8. Latency : Ext2 vs. HERMES

2. 스트리밍 부하 실험

멀티미디어 스트리밍 환경을 구현하기 위하여 파일 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.

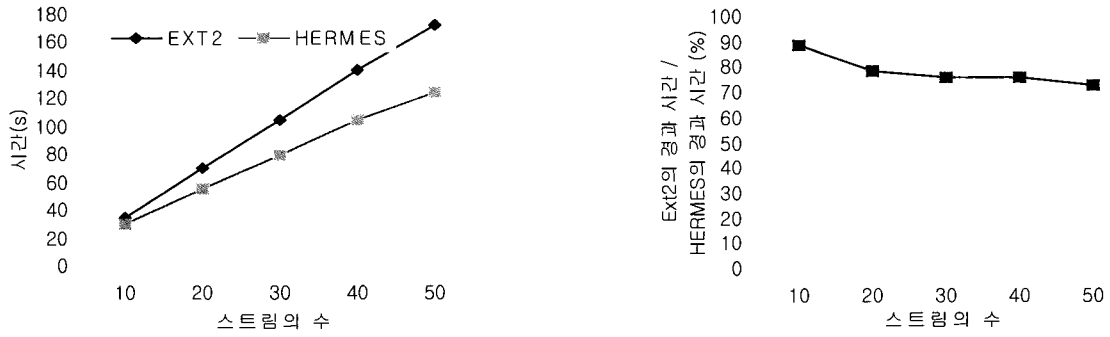

그림 9. 스트림의 수의 증가에 따른 순차 읽기 소요 시간
Fig. 9. Scalability: EXT2 vs. HERMES

3. 응답시간의 편차

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.

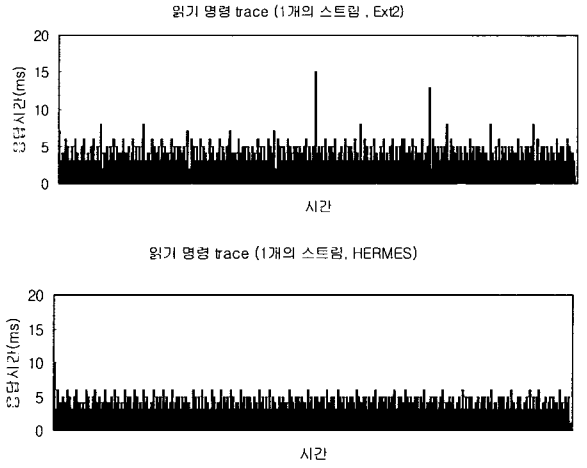

그림 10. 64K를 읽을 때 응답 시간
Fig. 10. Response time in 64kbyte I/O block

그림 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 11에서는 이들의 분산을 비교해 보았다. 이것으로 HERMES가 안정적으로 파일 읽기 명령을 수행한다고 볼 수 있다.

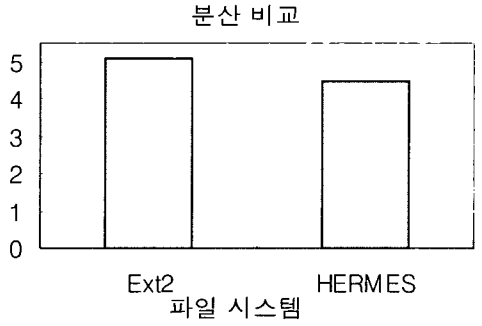

그림 11. Ext2와 HERMES 파일 시스템의 분산 비교
Fig. 11. Latency Vauiance: Ex2 vs. HERMES

Ⅶ. 결 론

멀티미디어 스트리밍 서비스의 급속한 대중화로 인하여 멀티미디어 서비스를 효율적으로 지원할 수 있는 서버의 설계가 매우 강조되고 있다. 이에 근간을 차지하는 핵심중의 하나가 멀티미디어 데이타에 적합한 파일 시스템의 설계이다. 멀티미디어 파일은 2개의 요소, '크기'와 '대역폭'으로 특성 지어질 수 있다. 스트리밍 서비스를 위한 파일 시스템은 대용량의 파일을 효과적으로 다룰 수 있어야 하고, 데이터를 읽어 들이는데 요구되는 오버헤드를 최소화하여, 일정 데이터 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, 전송하는데 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 제안한 HERMES 파일 시스템은 파일 저장 단위를 상대적으로 크기가 큰 extent로 정의를 하고, inode 구조를 멀티미디어 데이터에 맞게 바꾸어, 기존의 유닉스 파일 시스템의 멀티레벨 구조가 가지는 성능 측면에서의 한계를 극복하였다. 또한 MPEG-4 파일 포맷에 최적화 되었으며, 자료의 저장단위를 물리적인 단위에서 멀티미디어 자료의 논리적 단위로 채택함으로써 보다 유연하고 다양한 스트리밍 서비스 환경의 구현을 가능케 하였다.

성능평가 결과 스트리밍 환경에서 기존의 Ext2 파일 시스템 보다 월등한 성능을 보이고 있다. HERMES 파일 시스템은 홈 서버, PVR(Personal Video Recoder), Digital STB(Set Top Box)등 스트리밍 서비스를 목적으로 하는 차세대 정보가전에서 핵심을 담당하는 소프트웨어 컴포넌트로서의 역할을 십분 다할 것으로 확신한다.

참 고 문 헌

[1] 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*, pp.235-242, 1993.

[2] 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.

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

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

[5] 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.

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

[7] Antine Mourad, "Issues in the design of a storage server for video-on-demand," *Multimedia Systems*, 1996(4):70-86, 1996.

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

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

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

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

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

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

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

[15] 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*, pp.297-302, Bristol, UK, 1998.

[16] Roger L. Haskin, "Tiger Shark-a scalable file system for multimedia," *IBM Journal of Research and Development* vol.42, no.2 pp.185-197, 1998

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

[18] T. N. Niranjan, Tzicker Chiueh, and Gerhard A. Schloss. "Implementation and evaluation of a multimedia file system," *International Conference on Multimedia Computing and Systems-Proceedings* Jun. 1997

[19] Chuanbao Wang, Vera Goebel, and 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

[20] Rosenblum, M., "The Design and Implementation of a Log-Structured File System," Kluwer Academic Publishers, 1995.

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

[22] http://www.apple.com/

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

[24] http://www.tv-anytime.org

[25] Eunsam Kim, Hyeongho Son, and Baegeun Kang, "Design and implementation of an enhanced personal video recorder for HDTV," *Consumer Electronics, 2001. ICCE. International Conference on, 2001* pp.316-317

[26] Jaeger, R., "Set-top box software architectures for digital video broadcast and interactive services," *Performance, Computing, and Communications, 2001. IEEE International Conference on., 2001*

저 자 소 개

박 진 연

2000년 2월 : 한양대학교 공과대학 전기전자통신전파공학부 학사
2002년 1월 : 한양대학교 대학원 전자통신전파공학과 재학

송 승 호

2001년 8월 : 연세대학교 공과대학 사회환경시스템 공학부 학사
2002년 1월 : 한양대학교 정보통신대학원 정보처리학과 재학

진 종 현

2001년 2월 : 한양대학교 공과대학 전기전자통신전파공학부 학사
2002년 1월 : 한양대학교 대학원 전자통신전파공학부 재학

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

박 승 민

1981년 : 울산대학교 전자공학과 학사
1983년 : 홍익대학교 대학원 전자공학과(석사)
1998년 3월~현재 : 충남대학교 대학원 전자공학과 박사과정
1983년 3월~1984년 9월 : (주)LG전자
1984년 9월~현재 : 한국전자통신연구원 인터넷정보가전연구부
실시간미들웨어연구팀장/책임연구원

김 정 기

1992년 : 전북대학교 컴퓨터공학과 학사
1994년 : 전북대학교 컴퓨터공학과 석사
1999년 : 전북대학교 컴퓨터공학과 박사
1996년~1998년 : 시스템공학연구소 연구원
1998년~현재 : 한국전자통신연구원 선임연구원