멀티미디어 기기를 위한 지능형 입출력 서브시스템

(Intelligent I/O Subsystem for Future A/V Embedded Device)

장 형 규 † 원 유 집 †† 류 재 민 †
(Hyungkyu Jang) (Youjip Won) (Jaemin Ryu)
심 준 석 ††† 세르게이 볼데브 †††
(Junseok Shim) (Serguei Boldyrev)

요 약 지능형 디스크는 이전에 호스트 프로세서에서 수행되던 입출력 관련 연산 작업을 디스크 상에서 수행함으로써 전체적인 입출력 성능을 향상 시킬 수 있다. 하지만, 현 시점에서 입출력 시스템이 가지는 한계와 하위 호환성 문제로 인하여 지능형 디스크를 직접적으로 현실화 시키기는 어려워 보인다. 본 논문에서는 기존의 입출력 시스템과 하위 호환성을 유지할 수 있도록 물리적인 섹터 정보만을 이용하여 멀티미디어 부하를 판별하고 이를 기반으로 디스크의 동작을 멀티미디어 재생에 동적으로 최적화 시키는 방법을 제안한다. 다양한 입출력 부하로부터 멀티미디어 부하를 지능적으로 분류하기 위해 기계 학습 분야에서 사용되고 있는 부스팅 알고리즘을 사용하였다. 부스팅 알고리즘을 통해 구축된 최종 학습기를 이용하여 최근에 발생한 입출력 부하가 멀티미디어 부하라면, 디스크는 보다 많은 섹터를 미리 읽음으로써 멀티미디어 부하에 대한 디스크 활용율을 극대화 한다. 이러한 지능형 입출력 서브 시 스템을 차후에 멀티미디어 기기에 탑재되는 디스크 드라이브의 내부에 존재시킴으로써 호스트에 부가되는 부하없이 멀티미디어 데이타 재생에 대해 보다 효율적으로 디스크를 구동 시킬 수 있다. 또한, 이러한 결과로 저자원 모바일 기기에서 보다 고화질의 멀티미디어를 재생할 수 있게 된다.

키워드 : 디스크, 학습기, 입출력 최적화, 멀티미디어

Abstract The intelligent disk can improve the overall performance of the I/O subsystem by processing the I/O operations in the disk side. At present time, however, realizing the intelligent disk seems to be impossible because of the limitation of the I/O subsystem and the lack of the backward compatibility with the traditional I/O interface scheme. In this paper, we proposed new model for the intelligent disk that dynamically optimizes the I/O subsystem using the information that is only related to the physical sector. In this way, the proposed model does not break the compatibility with the traditional I/O interface scheme. For these works, the boosting algorithm that upgrades a weak learner by repeating learning is used. If the last learner classifies a recent I/O workload as the multimedia workload, the disk reads more sectors. Also, by embedding this functionality as a firmware or a embedded OS within the disk, the overall I/O subsystem can be operated more efficiently without the additional workload.

Key words : *disk, learner, I/O, optimization, multimedia*

† 비 회 원 : 한양대학교 전자전기컴퓨터
hkjang@ece.hanyang.ac.kr
deflowers@ece.hanyang.ac.kr
†† 종신회원 : 한양대학교 전자전기컴퓨터 교수
yjwon@ece.hanyang.ac.kr
††† 비 회 원 : 삼성종합기술원 스토리지랩 연구원
shim.junseok@samsung.com
Serguei.Boldrev@samsung.com
논문접수 : 2004년 12월 17일
심사완료 : 2005년 10월 21일

1. 개 요

1.1 연구 동기

일반적으로 컴퓨터 시스템은 응용 프로그램과 운영체제, 파일 시스템, 디바이스 드라이버 그리고 디바이스의 계층적인 구조로 이루어진다. 시스템 구조를 계층화 시킴으로써 얻게 되는 가장 큰 장점은 각 계층간의 의존성을 최소화 할 수 있다는 점이다. 각 계층은 사전에 정

의된 인터페이스를 통해서만 인접한 다른 계층과 상호 작용이 일어나기 때문에 인터페이스 체계만 준수한다면 계층간의 상호 간섭 없이 독립적으로 발전할 수 있게 된다. 하지만 계층간 서로 다른 이름 공간을 사용하는 디스크 입출력 서브 시스템에서는 이러한 계층적인 구조로 인해 성능 향상에 제한을 가질 수 밖에 없게 된다. 즉, 응용 프로그램에서 디스크까지 입출력 관련 작업이 진행됨에 따라 파일의 특성에 대한 정보들이 손실되어 결과적으로 디스크는 응용 프로그램이 발생하는 부하에 상관없이 동일한 입출력 동작을 수행할 수 밖에 없다는 한계를 가지고 있다.

이러한 입출력 서브 시스템을 개선하기 위한 지금까지의 연구는 주로 새로운 인터페이스를 추가하거나 디스크에 지능형 모듈을 추가하는 방향으로 진행되어 왔다. 예로, 마이크로프로세서와 메모리를 탑재하고 있는 지능형 디스크는 기존의 호스트 프로세서에서 수행되던 입출력 관련 연산을 디스크 단에서 직접 수행함으로써 호스트 프로세서와 디스크 사이에 발생하는 트래픽을 감소시켜 전체적인 시스템의 성능을 향상 시키고자 하였다. 이러한 지능형 디스크는 파일 시스템 단계에서 사용할 수 없는 정확한 헤드 위치, 블록 매핑 정보와 같은 디스크의 물리적인 정보를 활용할 수 있다는 장점을 가지고 있기는 하지만, 현재 실행중인 응용 프로그램의 특성이나 운영체제의 동작에 대해서는 여전히 알 수 없으며 기존의 입출력 서브 시스템에서 사용하는 인터페이스 체계와 호환성의 결여로 인하여 직접적으로 현실화 시키기는 어려워 보인다.

본 연구에서는 이제까지와는 달리 입출력 인터페이스 체계의 변화 없이 디스크 내부에서 학습을 통하여 현재 입출력 서브 시스템의 상태를 추론할 수 있는 새로운 연구 방향을 제시한다. 이러한 목적을 위해, 다양한 응용프로그램이 발행하는 입출력 부하중에서 멀티미디어 부하를 특성화 하고 이를 기반으로 멀티미디어 응용 프로그램이 실행될 경우 입출력 시스템을 동적으로 최적화 시키는 과정을 보인다. 다양한 입출력 부하중에서 멀티미디어 부하를 특성화 시키기 위해서 미리 얻어진 입출력 패턴을 학습하여 학습기를 생성하고, 최종적으로 만들어진 학습기를 이용하여 일정 시간마다 입출력 부하의 유형을 분류한다. 만약 입출력 부하가 멀티미디어 부하로 추론되면 디스크 입출력 서브 시스템을 멀티미디어 응용 프로그램에 최적화 시킨다. 이런 기능은 차후 지능형 디스크에 펌웨어나 임베디드 디스크 운영체제와 같은 형태로 탑재하여 입출력 서브 시스템이나 디스크의 동작이 동적으로 특정 응용 프로그램에 최적화 될 수 있다.

2. 관련 연구

운영체제가 관리하는 버퍼 캐쉬는 응용프로그램에게 블랙박스로 보인다. [1]는 응용프로그램이 운영체제의 버퍼 캐쉬 정책에 대한 정보를 미리 가지도록 함으로써 버퍼 캐시의 상태를 응용프로그램이 추론할 수 있도록 한다. 즉, 블랙 박스로 간주되던 버퍼 캐쉬를 그레이 박스로 간주되도록 만드는 것이다. [2]는 운영체제가 알 수 없는 디스크의 실리더, 트랙, 섹터와 같은 물리적인 정보를 이용하여 상관도가 높은 데이타를 디스크의 같은 트랙에 할당하고 접근하는 방법을 보여준다. 이를 통해서 대부분의 탐색 시간과 회전 지연을 피한다.

[3]는 응용프로그램에게 운영체제의 디스크 스케쥴링에 대한 정보를 제공함으로써 디스크가 소비하는 에너지를 줄이기 위한 입출력 서브시스템을 보여준다. 이와는 달리 [4]에서는 디스크 관련 정보를 미리 알려주지 않고 정확한 디스크 프로파일링[5-7]을 통해서 디스크 관련 정보를 직접 알아낸다. 그 정보를 바탕으로 본래 비선점의 특징을 가지는 디스크 입출력 작업을 선점이 가능하도록 만드는 실시간 응용 프그램을 위한 디스크 스케쥴링을 만든다.

디스크와 관련된 물리적인 정보를 바탕으로 디스크 입출력 서브시스템을 지능화 시키는 것과는 달리 지능형 디스크 내부의 연산 기능을 이용하는 연구도 진행되고 있다. [8]는 디스크에서 물리적인 정보를 이용하여 연산된 결과를 바탕으로 입출력 서브 시스템이 보다 향상된 성능을 보일 수 있도록 한다. [9, 10]는 전통적으로 파일 시스템에서 수행되던 일부 기능을 디스크로 옮겨와 현재 디스크 헤드의 정확한 위치를 알아낸다. 이를 통해 현재 디스크 헤드가 위치한 곳의 사용되지 않는 섹터에 작은 양의 동기화 쓰기를 수행함으로써 회전 지연을 줄일 수 있다는 것을 보여준다. [11-13]에서는 디스크에서 발생하는 회전 지연 동안에 경유하는 섹터들을 공격적으로 읽어들여 디스크 메모리에 저장한다. 이를 통해서 회전 지연을 가까운 미래에 사용될지 모르는 데이타를 미리 읽는 유용한 입출력 작업으로 변경한다. 이를 자유 블록 스케쥴링이라고 말하고 있다.

이제까지 디스크 입출력 서브시스템에서 지능화 연구와 관련하여 시도된 접근 방식은 결정적으로 제공된 디스크 관련 정보나 새로운 인터페이스의 추가에 의해 현재의 입출력 시스템의 성능을 개선하도록 하였다. 본 연구에서는 이제까지와는 달리 지능형 디스크가 기존의 입출력 서브시스템의 큰 변화 없이 디스크 내부에서 학습을 통하여 현재 입출력 서브시스템의 상태를 추론하는 새로운 연구 방향을 제시한다.

3. 블록 기반 추상화의 한계

3.1 기존 입출력 서브시스템의 구성

입출력 서브 시스템을 바라보는 데에는 다양한 시각이 존재할 수 있다. 따라서, 입출력 서브 시스템이 가지는 한계에 대해서 살펴보기 앞서 입출력 서브 시스템을 바라보는 총체적인 시각을 정의해야 할 필요가 있다. 그림 1은 입출력 서브 시스템을 소프트웨어와 디스크를 포함하는 하드웨어를 거시적인 측면에서 바라본 것으로서, 크게 응용프로그램, 운영체제(파일시스템, 디바이스 드라이버), 디스크로 구분하고 있다.

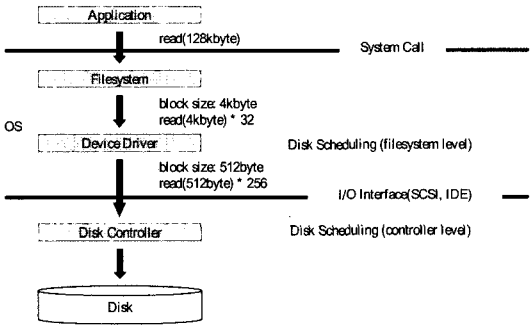


그림 1 입출력 서브시스템 구성도

이와 같은 입출력 서브 시스템을 구성하는 각 계층 사이에 정보가 전달되는 과정은 그림 2에 보이는 것과 같다. 여기서, 디스크는 저장 공간을 실린더, 트랙, 섹터라는 공간적인 구성요소로 구성하는 반면 파일 시스템은 이러한 공간적인 성격을 가지는 디스크를 논리 블록의 선형적인 배열로 보이도록 구조화 시킨다[14].

전통적인 유닉스 파일시스템의 경우에 각 파일은 데이타 블록의 위치, 최근 수정 시간, 접근 모드 등과 같은 그 파일에 관련된 정보를 가지는 아이노드라고 불리는 데이타 구조체와 연결되어 있다. 유닉스 파일시스템에서는 이러한 아이노드를 가지고서 각 파일을 구별하고 관리한다. 파일시스템마다 파일을 구별하고 관리하는 방법이 조금씩 다를 수도 있지만, 어떤 파일시스템도 운영체제가 파일을 관리하기 위해 사용하는 파일디스크립터를 가지고 파일을 관리하지 않는다. 따라서, 운영체제는 파일을 관리하기 위해서 사용하는 파일디스크립터로부터 그 파일에 대한 아이노드를 찾아서 파일시스템에 전달한다. 파일시스템은 전달 받은 아이노드로부터 파일의 데이타 블록의 위치와 관련된 정보를 파일시스템에서 가져와서 입출력 작업에 사용한다. 파일과 관련이 있는 데이타 블록을 구성하는 방법은 각각의 파일시스템에 의존적이다.

디바이스 드라이버는 실제 디바이스와 운영체제를 연결해주는 소프트웨어 계층이다. 디스크 디바이스 드라이버와 같은 경우, 파일시스템의 연산 작업의 결과로서 나온 데이타 블록의 번호가 디스크 디바이스 드라이버에 전달된다. 물리적인 디스크는 논리적인 데이타 블록 번호를 알 수 없기 때문에 디스크 디바이스 드라이버는 논리적인 데이타 블록 번호를 실린더, 트랙, 섹터의 물리적인 공간 정보로 변환하여 디스크에게 전달한다. 디스크는 이렇게 받은 정보들을 가지고 물리적인 입출력 작업을 수행한다.

3.2 기존 입출력 서브시스템의 한계

현재의 입출력 서브 시스템의 큰 한계는 앞서 기술한 입출력 서브 시스템을 구성하는 각 구성 요소들이 서로

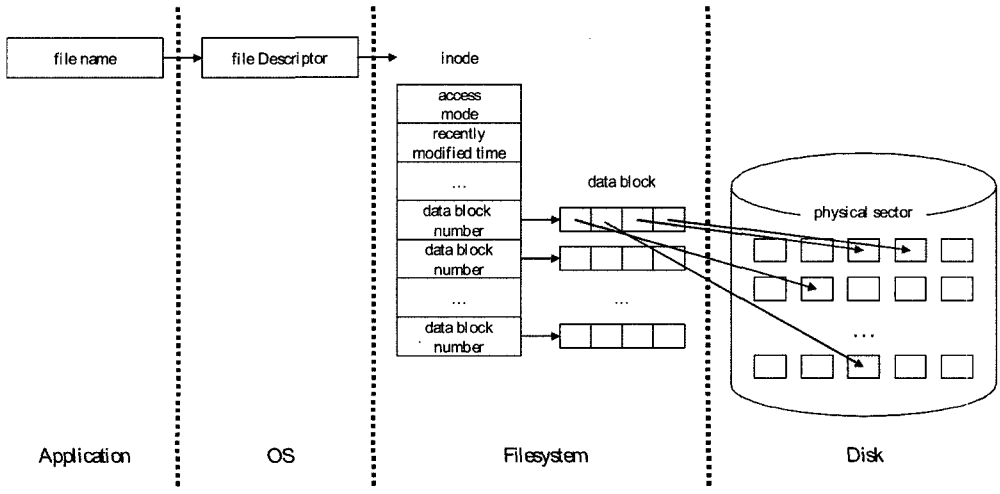


그림 2 입출력 서브시스템의 이름 공간

다른 이름 공간(namespace)을 사용한다는 점에서 부터 비롯된다. 디스크에 저장되어 있는 파일을 읽거나 쓸 경우, 응용 프로그램은 파일 이름을, 운영체제는 파일 디스크립터를, 파일 시스템은 논리 블록을, 디스크는 물리적인 섹터를 사용한다. 또한, 텍스트 편집기, 데이타베이스, 멀티미디어 응용 프로그램, 게임, 컴파일러와 같은 다양한 응용 프로그램은 서로 다른 형태의 입출력 동작을 요구한다. 하지만, 파일 시스템은 디스크에 섹터 번호, 섹터 갯수, 입출력 작업의 종류와 같은 단순한 정보밖에 전달할 수 없기 때문에, 최종적으로 디스크의 입장에서 보면 응용 프로그램이 요구하는 입출력 동작과는 상관없이 모든 데이타를 동일시 취급하게 된다[14].

4. 지능형 입출력 서브 시스템의 구조

지금까지 제안된 지능형 디스크는 응용 프로그램이나 운영체제가 알 수 없는 디스크의 물리적인 정보를 활용함으로써 입출력 시스템의 성능을 개선시킬 수 있는 많은 일을 수행할 수 있었다. 하지만, 이들 역시 현재 응용 프로그램이나 운영체제가 어떤 작업을 수행하고 있는지 알 수 없으며, 입출력 인터페이스의 변경을 요구하기 때문에 기 개발된 시스템과의 하위 호환성을 유지할 수 없다는 문제를 가지고 있다.

본 연구를 통해 제시하고자 하는 지능형 입출력 서브 시스템은 디스크 내부에서 현재 실행중인 응용 프로그램이 발생하는 부하를 학습함으로써 다음에 발생할 부하를 추론하기 때문에 기존의 입출력 서브 시스템의 체계를 변화 시키거나 응용 프로그램이 부가적인 정보를 디스크에 전달하지 않고도 디스크의 동작을 현재 실행중인 응용 프로그램의 부하에 최적화 시킬 수 있게 된다. 이와 같이 응용 프로그램이 발생하는 부하를 학습하고 다음에 발생될 부하의 유형을 추론하기 위해 본 연구에서는 workload monitor, workload analyzer system optimizer로 이루어진 지능형 모듈을 개발하였다.

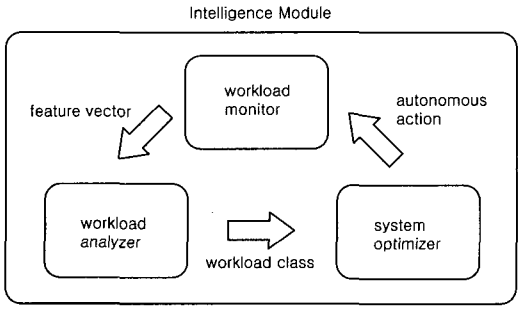


그림 3 지능형 입출력 서브 시스템 구성 모듈

이들 각 구성요소간의 동작 관계는 그림 3에 보이는 것과 같다. 여기서, workload monitor는 최근 T초동안 응용 프로그램에서 발생한 I/O 요청을 수집하여 내부의 트레이스 큐에 유지하고 workload analyzer는 트레이스 큐 내의 I/O 요청을 분석하여 특성화 벡터를 추출한 후 현재 발생한 부하의 유형을 분석한다. workload analyzer가 추출하는 특성화 벡터는 <*디스크로부터 데이타를 읽은 회수, 사이 시간, 사이 시간의 표준편차, 사이 시간의 중간값, 사건 사이 시간의 범위, 디스크로부터 읽은 총 데이타의 양*>과 같은 원소로 이루어지며 이 특성화 벡터가 부스팅 알고리즘의 입력으로 주어져 현재 발생한 부하의 유형을 결정하게 된다. 특성화 벡터를 결정하는 과정과 입출력 요청의 유형을 결정하는 자세한 사항은 다음절에 기술하였다. 또한, workload analyzer에 의해 현재 발생한 부하의 유형이 결정되면 이 결과에 따라 system optimizer는 디스크의 동작을 현재 발생한 부하에 최적화 시킨다.

5. 입출력 데이타의 특성화

이번 절에서는 우리는 멀티미디어 데이타의 특성들에 기초하여 시스템에서 발생하는 입출력 부화중에서 멀티미디어 부하를 특성화 시킬 수 있음을 보여주려 한다. 즉, 다양한 입출력 패턴으로 부터 멀티미디어 부하의 특성을 가장 잘 설명할 수 있는 파라미터를 추출한다. 입출력 패턴에서 파라미터를 추출하는 과정은 미리 알려진 다양한 유형의 응용 프로그램을 실행 시키고 각 응용 프로그램이 발생하는 입출력 요청을 디바이스 드라이버 단에서 수집한다. 이렇게 수집된 입출력 요청들을 통해 멀티미디어 데이타가 가지는 특성을 가장 잘 설명할 수 있는 요소들을 추출해 낼 수 있다.

5.1 입출력 패턴의 추출

입출력 패턴을 추출하기 위해 사용된 데이타의 종류는 표 1과 같다. 우리는 각 멀티미디어 응용 프로그램에서 여섯 종류의 멀티미디어 파일을 재생하면서 각각의 경우로부터 10개 이상의 입출력 패턴을 추출하고 디스크 입출력 동작에 대한 통계적 자료와 입출력 패턴의 형태를 비교 하였다. 이 과정에서 동일한 멀티미디어 응용프로그램과 멀티미디어 파일의 경우에 추출한 입출력 패턴으로부터는 각각을 구별할 수 있는 차이점을 찾을 수 없었기 때문에 각각의 경우에 대해서 두 개의 입출력 패턴만을 취하였다. 표 2는 모바일 기기에서 자주 사용되는 다른 응용 프로그램에 대한 입출력 패턴으로 추출한 입출력 패턴의 대상은 텍스트 파일의 읽기, 쓰기, 수정, 게임 등의 응용프로그램 실행한 경우에 대한 것이다.

표 1 응용프로그램 입출력 패턴

Application	File	# of I/O Pattern
mpeg2dec	news(8.76Mbps, high)	2
mpeg2dec	news(0.38Mbps, low)	2
mpeg2dec	music video1(1.18Mps, high)	2
mpeg2dec	music video1(0.67Mbps, low)	2
mpeg2dec	music video2(2.60Mbps, high)	2
mpeg2dec	music video2(1.16Mbps, low)	2
xine	news(8.76Mbps, high)	2
xine	news(0.38Mbps, low)	2
xine	music video1(1.18Mps, high)	2
xine	music video1(0.67Mbps, low)	2
xine	music video2(2.60Mbps, high)	2
xine	music video2(1.16Mbps, low)	2
mplayer	news(8.76Mbps, high)	2
mplayer	news(0.38Mbps, low)	2
mplayer	music video1(1.18Mps, high)	2
mplayer	music video1(0.67Mbps, low)	2
mplayer	music video2(2.60Mbps, high)	2
mplayer	music video2(1.16Mbps, low)	2

표 2 다른 응용프로그램 입출력 패턴

Application	Operation	# of I/O Pattern
text editor	read, write, and modify	3
game	run	3

5.2 입출력 패턴의 형태 및 특성

그림 4부터 그림 9는 이러한 멀티미디어 응용 프로그램의 입출력 패턴으로, 우리는 이로 부터 고화질의 멀티미디어 파일일수록 즉, 초당 재생률 클수록 그 주기가 짧아지고 모든 멀티미디어 데이타는 멀티미디어 응용프로그램에 상관없이 공통적으로 주기적인 입출력 패턴을 보이고 있음을 알 수 있다. 또한, 멀티미디어 파일은 일반적으로 디스크에 연속적으로 저장되는 경향이 크며 연속한 두 개의 입출력 요청은 각각 연속한 248, 8개의 논리 섹터를 가진다. 이상의 특성화 과정 통해 얻은 결과와 멀티미디어 부하가 가지는 일반적인 특성을 연관지어 보면, (1)멀티미디어 데이타는 디스크에 연속적으로 저장되는 경향이 크며 (2)파일에 명시된 초당 재생률에 따라 주기적인 입출력 부하를 발생시키며 (3) 매 순간 읽어 들이는 데이타의 양은 규칙적이다.

우리는 이러한 사실에 입각하여 멀티미디어의 특성을 가장 잘 설명할 수 있는 변수로 각 요청에 대해 <*디스크로부터 데이타를 읽은 회수, 사이 시간, 사이 시간의 표준편차, 사이 시간의 중간값, 사건 사이 시간의 범위, 디스크로부터 읽은 총 데이타의 양*>과 같은 벡터를 산출할 수 있다. 이렇게 산출된 벡터는 부스팅 알고리즘에 기반한 추론 엔진에 입력으로 사용되어 다음번에 발생할 부하를 예측하게 된다.

6. 특성화 부하의 지능적 분류

6.1 학습을 통한 입출력 부하 추론

미리 얻어진 입출력 패턴을 학습하여 입출력 부하를 지능적으로 추론하는 학습기를 만들기 위해 기계 학습 분야에서 널리 알려진 부스팅 알고리즘을 사용하였다.

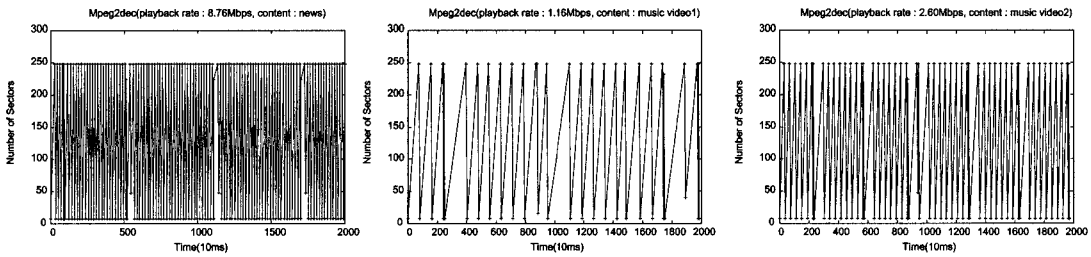


그림 4 Mpeg2dec 입출력 패턴(고화질)

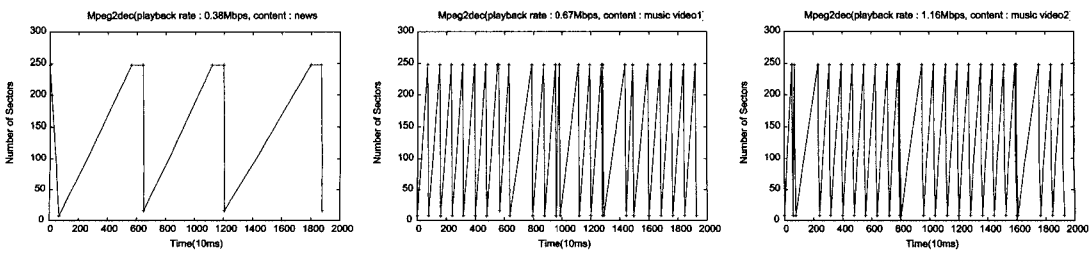


그림 5 Mpeg2dec 입출력 패턴(저화질)

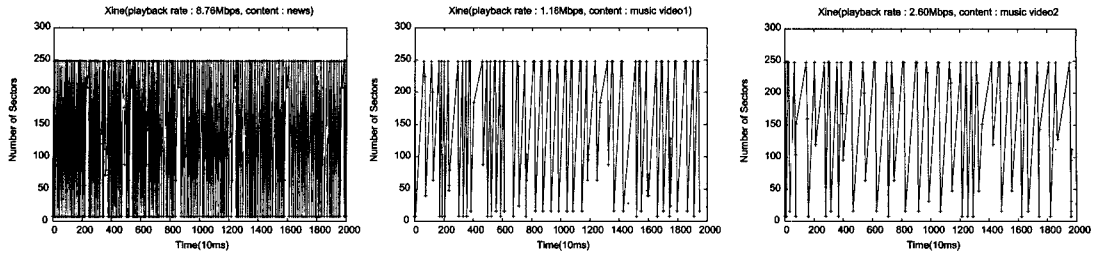


그림 6 Xine 입출력 패턴(고화질)

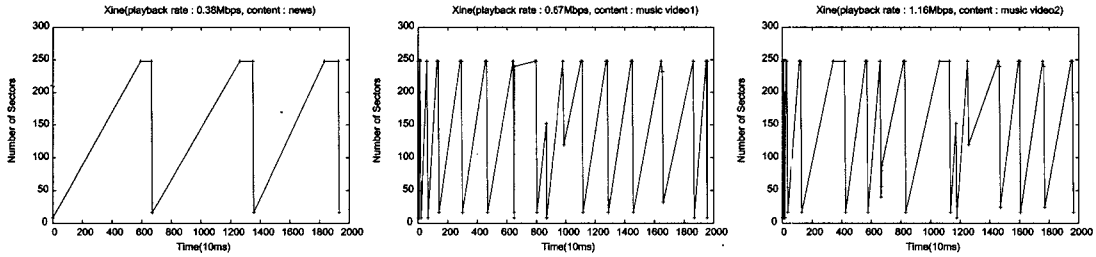


그림 7 Xine 입출력 패턴(저화질)

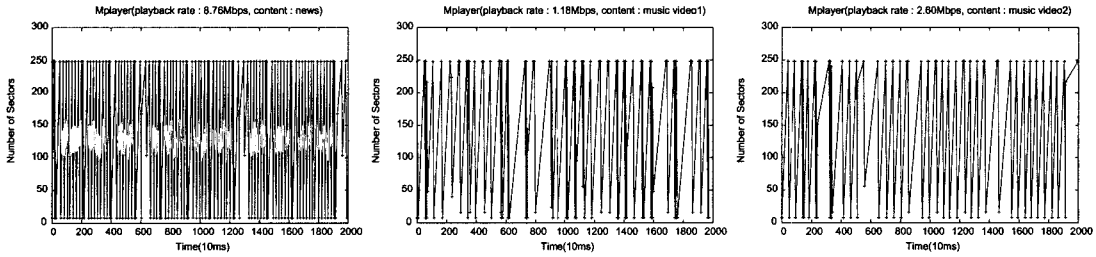


그림 8 Mplayer 입출력 패턴(고화질)

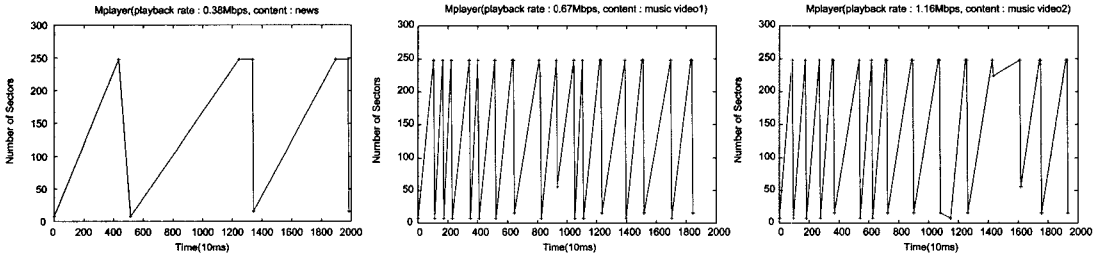


그림 9 Mplayer 입출력 패턴(저화질)

부스팅 알고리즘은 다음과 같은 기본 모형을 기자는 분류(classification) 문제에 사용된다.

n개의 학습자료 $(x_1, y_1),(x_2, y_2),\cdots(x_n, y_n)$이 주어졌다고 가정하자. x_i는 p차원의 입력변수이고, 즉, $x_i = x_{i1}, xx_{i2}, x_{i3}, \cdots, x_{ip}$이고, 출력변수 y_i는 학습자료가 속하는 그룹(class)을 나타낸다. 2개의 그룹이 있을 때, y_i는 1과 -1 중 하나의 정수값을 가진다. 분류문제의 목적은 n개의 학습자료를 이용하여 입력변수로 출력변수를 가장 잘 설명하는 관계를 찾는 것이다. 다시 말하면, 학습자료를 이용하여 최적의 $HR^p \rightarrow \{-1,1\}$함수를 만드는 것이다. 그리고, 새로운 입력변수 x가 주어지면, 이 입력변수를 그룹 $H(x)$로 할당한다.

부스팅 알고리즘은 이전의 학습보다 다음 학습에서 보다 더 좋은 결과가 나올 수 있도록 가중치를 갱신한다. 즉, 분류하기 어려운 입력변수를 가지는 학습자료의 가중치를 높인다. 최종적으로 매 학습마다 만들어진 기

표 3 부스팅 알고리즘

1. 초기화 : n 개의 가중치 $w_1, w_2, w_3, \cdots w_n$ 을 $w_i = 1/n$ 으로 놓는다.
2. 다음을 $m = 1, \cdots, M$ 번 반복한다.
   A. n 개의 학습자료 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ 과 가중치 $w_1, w_2, w_3, \cdots w_n$ $를 이용하여 주어진 입력변수 x_i 에 대하여 출력변수 y_i 가 1일 확률 $p_m(x_i) = \hat{p}_w(y_i = 1 | x_i)$ 를 기본 학습기를 이용하여 추정한다.
   B. $f_m(x_i) = \frac{1}{2} \ln p_m\left(\frac{x_i}{1 - p_m(x_i)}\right)$ 로 놓는다.
   C. 새로운 가중치를 $w_i = w_i \exp(-y_i f_m(x_i))$, $i = 1,2,3,\cdots,n$ 으로 구한 후, $\sum_{i=1}^{M} w_i = 1$ 이 되도록 표준화 한다.
3. 최종 앙상블 모형을 $H(x) = \sum_{i=1}^{M} f_m(x)$ 로 하여 새로운 입력변수 x 에 대하여 $H(x) > 0$ 이면, 그룹 2에, $H(x) < 0$ 이면, 그룹 1에 할당한다.

본학습기를 결합하여 추론 능력이 우수한 학습기를 만들게 된다. 표 3은 부스팅 알고리즘에 대한 자세한 설명이다.

6.2 최종학습기의 성능

멀티미디어 부하 특성화를 위해 추출한 멀티미디어 응용프로그램과 다른 응용프로그램의 입출력 패턴으로부터 학습자료를 만든다. 멀티미디어 부하의 특성을 잘 파악할 수 있는 학습자료를 만드는 것이 무엇보다 중요한 열쇠 이다. 여기에서는 다 36개의 멀티미디어 응용프로그램의 입출력 패턴과 9개의 다른 응용프로그램의 입출력 패턴으로부터 멀티미디어 부하에 대한 학습 자료 2200개, 게임 실행, 텍스트 읽기, 쓰기, 편집과 같은 다른 입출력 부하에 대한 학습 자료 800개를 만든다. 부스팅 알고리즘에서 사용하는 기본학습기는 최종노드가 3개인 의사결정트리(decision tree)이고, 수행하는 학습의 회수는 200으로 하였다.

만들어진 최종학습기의 성능을 평가하기 위하여, 학습자료를 2개의 부분으로 랜덤하게 나눈 후, 하나의 학습자료를 이용하여 학습기를 만들고, 다른 자료를 이용하여 구축된 최종학습기의 추론 능력을 구한다. 이러한 작업을 랜덤하게 100번을 반복하여 최종학습기의 추론 능력을 최종적으로구한다. 다음의 표 4는 이와 방법을 이용하여 측정한 최종학습기의 입출력 부하 추론에 대한 성능이다. False positive는 멀티미디어가 부하가 아닌 경우를 멀티미디어 부하로 추론 한 경우이며, False Negative는 멀티미디어 부하인 경우를 멀티미디어 부하가 아니라고 추론한 경우이다.

표 4 최종학습기의 성능

False Positive	False Negative	Total
0.0180	0.0218	0.0207

7. 지능형 입출력 서브시스템의 구현

지능형 입출력 서브시스템에 대한 구현은 현재 디스크 내부에서 이루어지는 것이 불가능하다. 하지만, 차후에 디스크 내부에 구현 가능한 기능을 운영체제 내부에 구현함으로써 간접적으로 그 성능을 평가할 수 있다. 본 연구에서는 그림 10에 보이는 것과 같이 리눅스 커널 2.4.22에 지능형 모듈을 추가함으로써 지능형 입출력 서브시스템을 구현하였다.

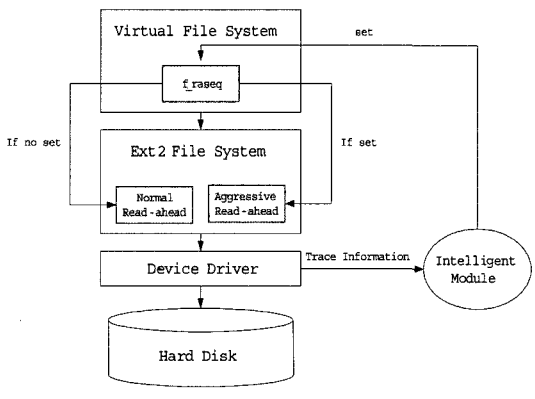


그림 10 의사 결정 모듈

여기서 지능형 모듈은 그림 3에서 보이는 것과 같이 workload monitor, workload analyzer, system optimizer로 이루어져 있다. workload monitor는 최근 5초 동안의 입출력 패턴을 커널에 새로 추가된 트레이스 큐에 유지하고, workload analyzer는 큐로부터 입출력 패턴을 가져와 입력 벡터를 생성한 후 최종 학습기에 전달하여 최근 5초 동안의 입출력 부하가 멀티미디어 부하인지 아닌지를 결정하여 입출력 서브시스템의 최적화 여부를 system optimizer에게 알려준다.

7.1 Workload Monitor

리눅스 커널에 입출력 패턴을 저장하기 위한 새로운 자료 구조를 추가한다. 이 자료 구조는 최근 5초 동안의 입출력 패턴을 유지하는 큐이다. 큐가 가지는 입출력 패턴은 디바이스 드라이버로부터 디스크에 전달되는 입출력 요청으로부터 다음과 같은 데이타를 가져온다.

- 읽기 혹은 쓰기 플래그
- 입출력 요청의 섹터 개수
- 디스크에 입출력 요청이 전달되는 시간

커널은 모듈을 초기화할 때, 미리 주어진 일정양의 빈(free) 입출력 패턴을 가지는 풀(pool)을 생성한다. 입출력 요청이 디스크에 전달될 때, 커널은 입출력 요청으로부터 빈 입출력 패턴을 채우고, 큐에 이것을 삽입한다. 그리고, 이전에 큐에 들어 있는 입출력 패턴으로부터 입출력 요청이 전달된 시간을 조사하여 최근 5초 이전의 입출력 패턴을 큐로부터 제거한다. 제거된 입출력 버퍼는 풀에 추가되어 나중에 다시 큐에 삽입되기를 기다린다. 풀을 사용함으로써 커널 내부에서 메모리 할당 및 해제에서 발생할 수 있는 오버헤드를 최소화한다.

7.2 Workload Analyzer

디바이스 드라이버로부터 디스크에 입출력 요청이 전달되면, 커널은 항상 큐에 최근 5초 동안의 입출력 패턴을 유지시킨다. 큐에 들어 있는 입출력 패턴으로부터 최종학습기의 새로운 입력 벡터를 만든다. 새로운 입력 벡터를 구성하는 요소들은 <디스크로부터 데이타를 읽은 회수, 사이 시간, 사이 시간의 표준편차, 사이 시간의 중간값, 사건 사이 시간의 범위, 디스크로부터 읽은 총 데이타의 양>이며, 최종학습기 생성에 사용된 학습 자료와 동일하다.

최근 5초 동안의 입출력 패턴으로부터 얻은 새로운 입력 벡터는 최종 학습기의 입력으로 사용된다. 앞절에 기술된 부스팅 알고리즘을 통해 이미 생성되어진 최종 학습기는 새로운 입력 벡터를 가지고 최근 5초 동안의 입출력 부하가 멀티미디어 부하인지 아닌지를 분류한다. 커널은 분류 결과가 멀티미디어 부하이면, 입출력 서브시스템의 최적화 여부를 system optimizer에게 알려준다.

7.3 System Optimizer

멀티미디어 파일은 주로 디스크에 연속되게 저장되기 때문에 최근 입출력 부하가 멀티미디어 부하로 분류되면 더욱 공격적으로 미리 읽기를 수행 한다. 즉, 다음번 발생할 부하가 멀티미디어 부하로 판단되면, 다음 번에 요청될 것 같은 데이타 블록을 미리 읽어놓음으로써 다음 번에 데이타 블록에 대한 입출력 요청이 발생했을 때, 이를 디스크에 접근하지 않고 버퍼 캐쉬에 읽어오게 함으로써 디스크에 접근하는 횟수를 줄이고 추가적으로

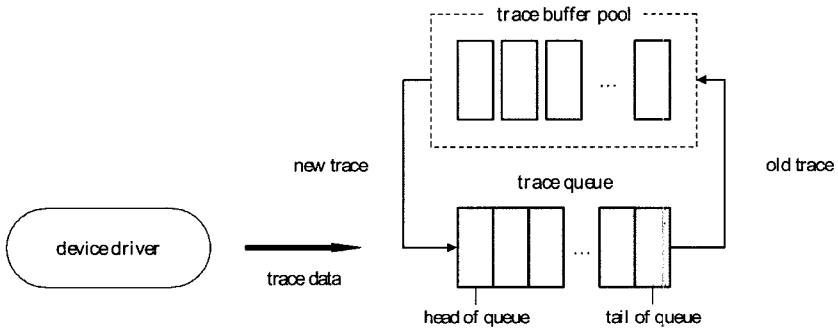


그림 11 Workload Monitor

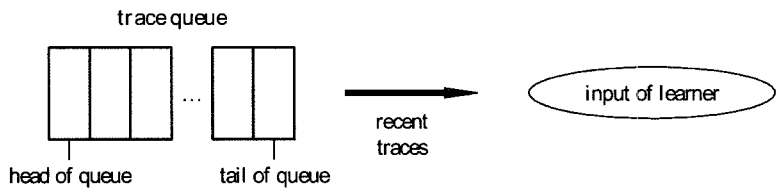


그림 12 Workload Analyzer-1

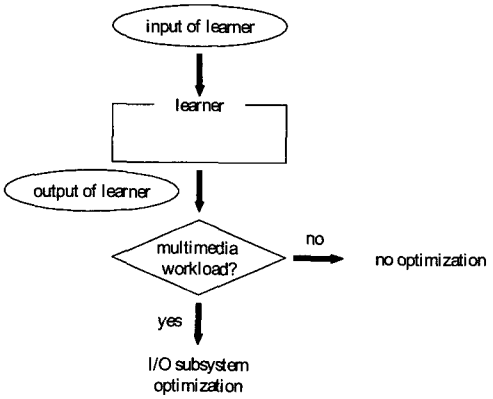


그림 13 Workload Analyzer-2

발생하는 디스크의 회전지연, 탐색 시간, 전송시간과 같은 물리적인 오버헤드를 감소시킬 수 있게 된다.

8. 실험 결과

우리는 지금 모바일 기기에서 지능형 입출력 서브시스템이 디스크에서 발생하는 물리적인 오버헤드를 감소시키는 정도를 보여주고자 한다. 실험에서 사용한 리눅스 커널의 미리 읽기 동작은 31개의 데이타 블록을 미리 읽는 것이다. 리눅스 커널에 구현한 지능형 입출력 서브시스템은 최근 5초 동안의 입출력 부하를 지능적으로 분류하여 그 분류 결과가 멀티미디어 부하이면, 미리 읽기 동작을 61개의 데이타 블록을 미리 읽도록 변화시킨다. 기존의 입출력 서브시스템과 지능형 입출력 서브시스템에서 Mplayer, Xine, Mpeg2dec로 입출력 부하 추출에 사용한 6개의 멀티미디어 파일을 재생하였다. 이들로부터 멀티미디어 파일 재생 동안에 발생하는 입출력 부하를 추출하여 디스크 시뮬레이션 툴인 DiskSim[15]으로 디스크에서 발생한 물리적인 오버헤드 정도를 비교 및 분석하였다. 그림 15, 16, 17은 Mplayer, Xine, Mpeg2dec에서 멀티미디어 파일을 재생하는 동안에 디스크에서 발생하는 정착 시간(탐색 시간 + 회전 지연), 전송 시간, 접근 시간의 비율을 보여준다. 그림 18, 19, 20은 지능형 입출력 버스 시스템과 기존의 입출력 서브시스템의 성능 비교를 보여준다. 지능형 입출력 서브시스템은 기존의 입출력 서브시스템과 비교하여 멀티미디어 파일 재생 동안에 정착 시간은 37-51%, 전송 시간은 2-27% 정도를 감소시켰다. 이를 통해서 호스트의 접근 시간은 17-30% 정도 감소하였다.

그림 21, 22, 23은 호스트가 디스크에 요청하는 하나의 입출력 요청당 정착 시간, 전송 시간, 접근 시간을 보여준다. 평균 입출력 요청의 크기가 미리 읽기 크기를 크게 함으로써 커지기 때문에 이에 대한 각각의 처리 시간은 증가한다. 하지만, 그림 24, 25, 26를 통해 알 수 있듯이 하나의 섹터 당 정착 시간, 전송 시간, 접근 시간은 감소한다. 이는 실제로 디스크로부터 데이타를 읽을 때, 디스크로부터 호스트로 데이타를 읽어오기 위해

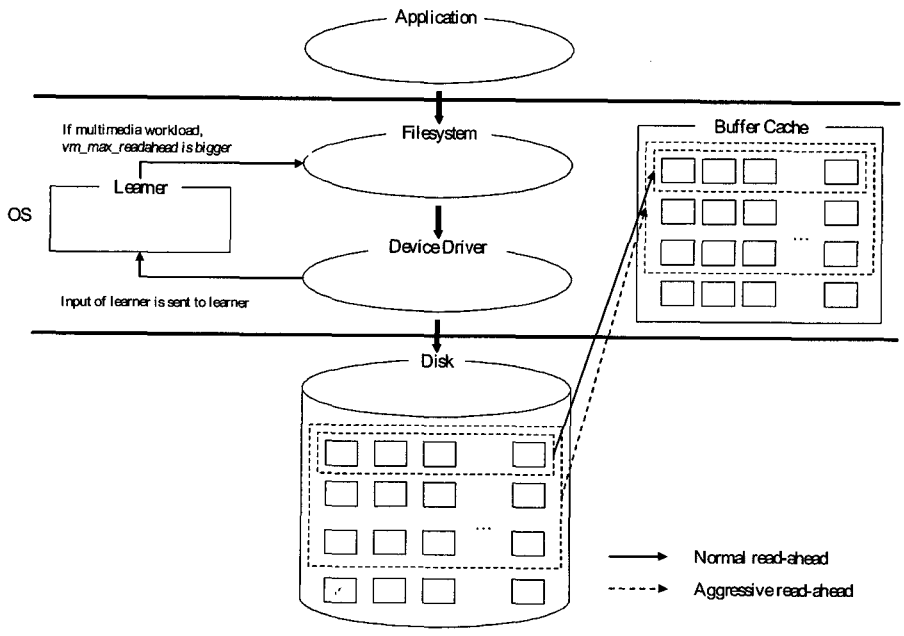


그림 14 System Optimizer

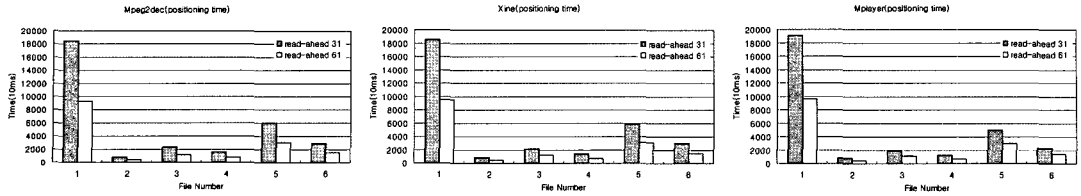


그림 15 Positioning Time

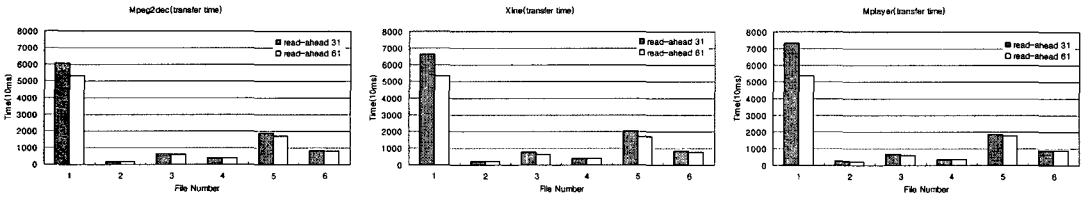


그림 16 Transfer Time

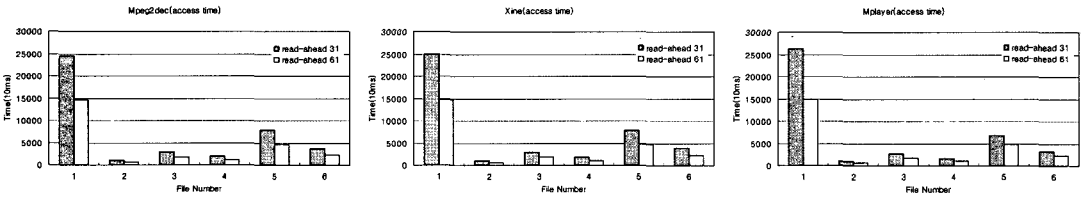


그림 17 Access Time

걸리는 시간의 평균이 줄었다는 것을 보여준다.

표 5 멀티미디어 파일의 종류

File Number	Content	Playback Rate
1	news	8.76Mbps
2	news	0.38Mbps
3	music video1	1.18Mbps
4	music video1	0.67Mbps
5	music video2	2.60Mbps
6	music video2	1.16Mbps

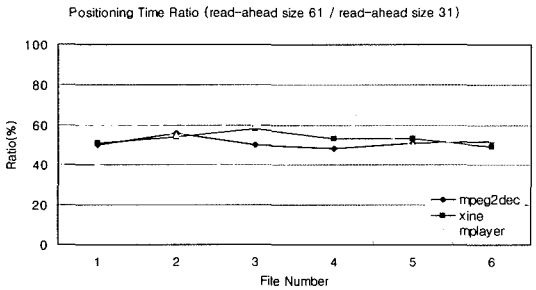


그림 18 Positioning Time Ratio

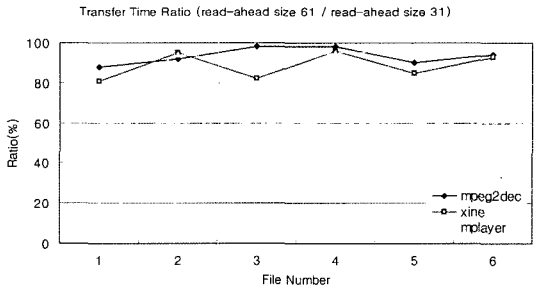


그림 19 Transfer Time Ratio

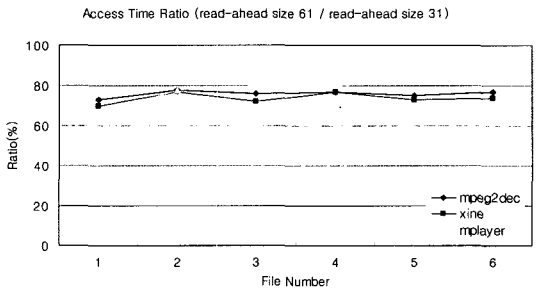


그림 20 Access Time Ratio

9. 결 론

현재 입출력 서브 시스템의 한계는 각 계층이 서로 다른 이름 공간을 사용한다는 점에서 비롯된다. 따라서 응용 프로그램에서 디스크까지 입출력 관련 작업이 진

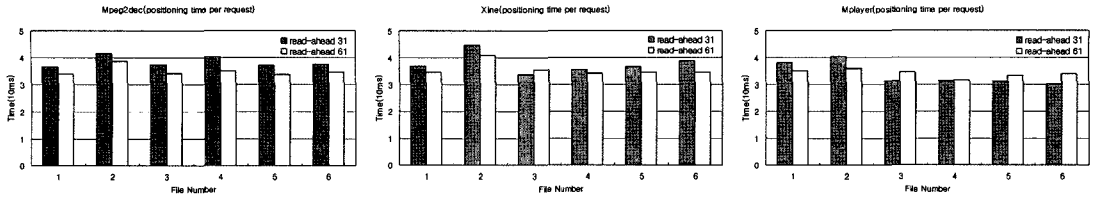


그림 21 Positioning Time per Request

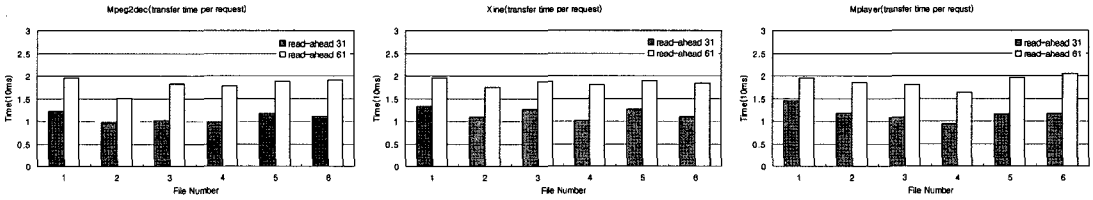


그림 22 Transfer Time per Request

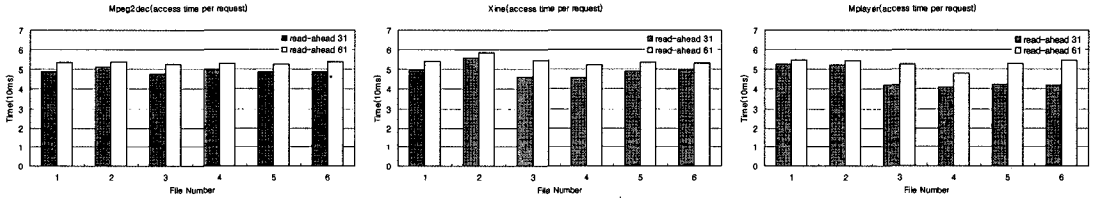


그림 23 Access Time per Request

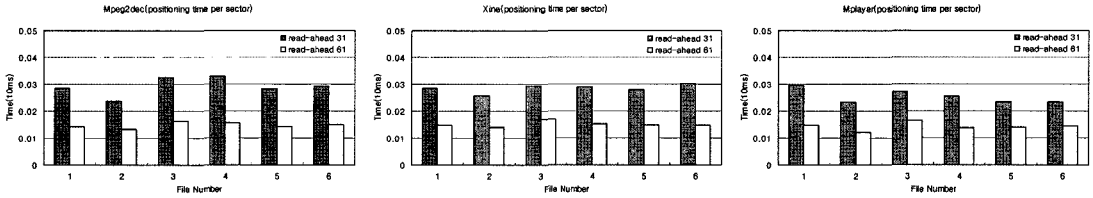


그림 24 Positioning Time per Sector

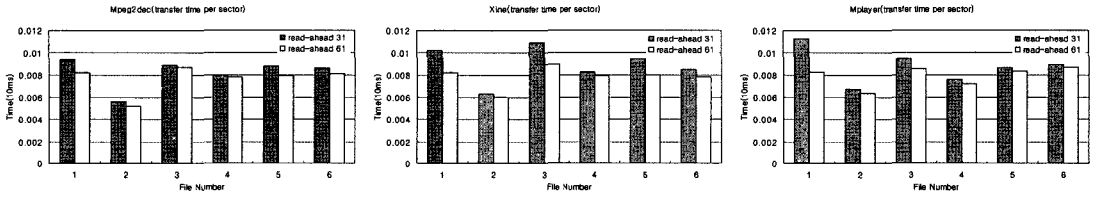


그림 25 Transfer Time per Sector

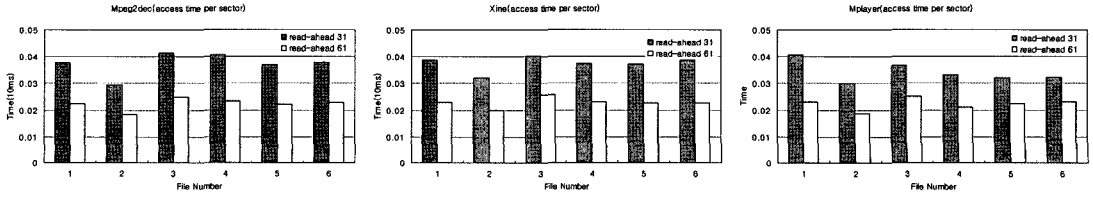


그림 26 Access Time per Sector

행됨에 따라 파일의 특성에 대한 정보들이 손실되게 되며 결과적으로 디스크는 응용 프로그램이 발생하는 부하에 상관없이 동일한 입출력 동작을 수행하게 된다.

이러한 입출력 서브 시스템을 개선하기 위해 지금까지 이루어졌던 연구는 주로 블랙 박스로 간주되는 다른 구성요소의 정보를 응용프로그램이 직접 이용하거나, 새로운 인터페이스를 추가하는 접근 방식을 취하고 있었다. 본 논문에서는 이러한 기존의 연구와는 달리 입출력 서브시스템의 변화 없이 호스트와 디스크 간에 발생하는 입출력 부하를 학습하여 디스크가 현재 어떤 종류의 입출력 부하가 발생하였는지를 알게 함으로써 입출력 부하에 따라 디스크의 동작을 동적으로 최적화 시킬 수 있는 방법을 제시하였다. 입출력 부하를 특성화 시키는 과정에서 보여준 것처럼 본 논문은 여러 입출력 부하 중에서 멀티미디어 부하를 특성화 시키고, 특성화된 속성을 입력 변수로 하는 부스팅 알고리즘을 통해 2% 정도의 오차를 가지고 해당 부하를 분류해 내었으며, 만약 분류된 입출력 부하가 멀티미디어 부하로 판결 날 경우 현재 입출력 작업 대상과 파일과 관련된 연속한 데이타 블록을 더 많이 읽어옴으로써 디스크에서 발생하는 물리적인 오버헤드를 감소시켰다.

참 고 문 헌

[1] N. Burnett, J. Bent, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. Exploiting gray-box knowledge of buffer-cache management, 2002.

[2] J. Schindler, J. Griffin, C. Lumb, and G. Ganger. Track-aligned extents: matching access patterns to disk drive characteristics, Jan.

[3] Andreas Weissel, Bjoern Beutel, and Frank Bellosa. Cooperative I/O - a novel I/O semantics for energy-aware application.

[4] Zoran Dimitrijevic, Raju Rangaswami, and Edward Chanf. Design and analysis of semi-preemtible IO. Proceedings of the Second Usenix FAST, Mar 2003.

[5] Mohamed Aboutabl, Ashock K. Agrawala, and Jean-Dominique Decotignie. Temporally determinate disk access: An experimental approach (extended abstract). In Measurement and Modeling of Computer Systems, pages 280-281, 1998.

[6] Raju Rangaswami David. Diskbench: User-level disk feature extraction tool zoran dimitrijevic.

[7] Bruce L. Worthington, Gregory R. Ganger, Yale N. Patt, and John Wilkes. On-line extraction of SCSI disk drive parameters. Technical Report CSE-TR-323-96, 1996.

[8] Anurag Acharya, Mustafa Uysal, and Joel H. Saltz. Active disks: Programming model, algorithms and evaluation. In Architectural Support for Programming Languages and Operating Systems, pages 81-91, 1998.

[9] Robert M. Loge: A self organizing disk controller. Technical Report HPL-91-179, Dec 1991.

[10] Randlph Y. Wang, Tomas E. Anderson, and David A. Patterson. Virtual log based file systems for a programmable disk. In Operating Systems Design and Implementation, pages 29-43, 1999.

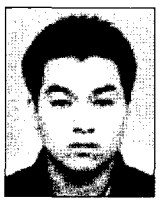
[11] Christopher Lumb, Jiri Schindler, Gregory R. Ganger, Erik Riedel, and David F. Nagle. Towards higher disk head utilization: Extracting "free" bandwidth from busy disk drives. Pages 87-102.

[12] Eric Riedel, Christos Faloutsos, Gregory R. Granger, and David F. Nagle. Data mining on an OLTP system (nearly) for free. Pages 13-21, 2000.

[13] C. Lumb, J. Schindler, and G. Ganger. Freeblock scheduling outside of disk firmware, an 2002.

[14] Muthian Sivathanu, Vijayan Prabhakaran, Florentina I. Popovici, Timmothy E. Denehy, Andrea C. Arpaci-Dusseau, and Remzi H. Aepaci-Dusseau. Semantically-smart disk systems. In Second USENIX Conference on File and Storage Technologies, Mar 2003.

[15] John S. Bucy, Gregory R. Granger, and et al. The disksim simulation environment version 2.0 reference manual.



장 형 규

2003년 한양대학교 전자공학과 졸업
2003년~현재 한양대학교 전자통신 컴퓨터공학 석사과정

원 유 집

1990년 서울대학교 통계학과 졸업(학사) 1992년 서울대학교 전산통계학 졸업(석사). 1997년 미네소타 주립대학 컴퓨터과학과 졸업(박사). 1999년~현재 한양대학교 전자통신컴퓨터 공학부 교수

류 재 민

2003년 한라대학교 전자공학과 졸업
2003년~현재 한양대학교 전자통신 컴퓨터공학 석사과정

심 준 석
현재 삼성 종합기술원, 스토리지 랩 전문
연구원

Serguei Boldyrev
현재 삼성 종합기술원, 스토리지 랩 전문
연구원