사용자 수 증대와 초기 대기시간 감소를 위한 비디오 저장 서버의 설계

마 평 수[†] · 조 창 식[††] · 진 윤 숙[††] · 신 규 상[†]

요 약

동시에 지원할 수 있는 사용자 수의 증대와 초기 대기시간의 감소는 비디오 저장 서버가 상용 서비스에 사용되기 위해서 갖추어야 할 필수적인 요소이다. 본 논문에서는 기존의 스트라이핑 방법에 비하여 각 디스크에서 읽어오는 데이터의 크기가 2배가 될 수 있는 데이터 배치 방법을 제안한다. 이러한 방법은 디스크의 회전 대기시간이 차지하는 비율이 작아지도록 하므로 디스크를 효율적으로 활용하게 되어 사용자 수가 크게 증대되는 성능의 향상을 보인다. 또한 초기 대기시간이 일정 시간을 넘지 않도록 하는 디스크 스케쥴링 방법을 제안하였고, 허용 가능한 초기 대기시간 개념을 도입하여 저장 서버의 순차적인 설계 방법을 제안한다. 구현 사례와 기존의 방법과의 비교를 통하여 제안한 방법의 우수한 성능 향상이 있음을 보인다.

Design of a Video Storage Server that Maximizes Concurrent Streams and Minimizes Initial Latency

Pyeong-Soo Mah[†] · Chang-Sik Cho[††] · Yun-Sook Jin[††] · Gyu-Sang Shin[†]

ABSTRACT

One of the most important functionality that commercial video storage servers should provide is to maximize the number of concurrent streams and to minimize the initial latency of new requests. In this paper, we propose a data placement scheme whose disk read unit size can be twice larger than that of conventional striping methods. The proposed scheme can significantly increase the number of concurrent streams, since the ratio of rotational latency time is decreased and the disks are effectively utilized. The disk scheduling scheme we propose guarantees constant initial latency time. We also propose a procedural design method for a storage server by introducing the concept of allowed initial latency. The comparison with previous research shows that the proposed scheme provides better performance.

1. 서 론

비디오 데이터는 매우 큰 저장 공간과 높은 전송 대역폭을 필요로 하고 연속적으로 재생되어야 하는 특성을 가지고 있기 때문에, 비디오를 포함한 멀티미디어 콘텐츠를 대규모의 사용자에게 제공하는 멀티미디어 서비스의 구현은 매우 어려운 문제이며, 네트워크와 디스크가 이러한 서비스를 제공하는데 병목 지점이 되고 있다[4]. 최근에는 ATM(Asynchronous Transfer Mode)이나 ADSL(Asymmetric Digital Subscribe Line)에 기반한 네트워크가 개인에게 수 Mbps 정도의 통신 대역폭을 제공함에 따라 비디오 데이터를 실시간으로 전송 받을 수 있는 네트워크 환경이 갖추어 지고 있다. 그러나 디스크에 저장된 비디오 데이터를 일정한

† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소 책임연구원
†† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소 연구원
논문접수 : 1999년 7월 24일, 심사완료 : 1999년 9월 13일

주기로 읽어서 네트워크를 통하여 클라이언트 시스템에 송신하는 비디오 저장 서버가 VOD(Video On Demand)와 같은 상업적인 멀티미디어 서비스에 사용될 수 있을 정도의 경제성을 갖추려면 아직도 해결하여야 할 많은 문제가 남아있는 상태이다.

이러한 문제 중에서 비디오 저장 서버가 동시에 지원할 수 있는 사용자의 수가 경제성을 갖추는데 가장 중요한 요소이므로, 기존의 많은 연구에서는 디스크를 효과적으로 제어하여 보다 많은 사용자를 지원하는데 역점을 두었다[3, 7, 9]. 이러한 기존의 연구에서는 시스템 설계시에 일정한 주기(period)를 설정하고, 설정된 주기별로 여러 비디오 스트림의 디스크 read 요구를 일괄적으로 처리하여 디스크의 seek time을 줄이고, 디스크에서 읽어오는 디스크 read unit의 크기를 최대로 하여 디스크 액세스의 오버헤드 비율이 적어지도록 하였다. 이러한 연구에서는 대규모의 사용자를 지원함을 목표로 하였으며, 비디오 데이터가 매우 큰 저장 공간과 전송 대역폭을 필요로 하므로 다중 디스크(disk array)의 사용은 필수적이다. 그러나 다중 디스크를 사용하면서 디스크의 read unit을 크게 하면 라운드의 주기가 커지게 되므로, 새로운 사용자의 디스크 read 요구가 도착하였을 때부터 해당 비디오가 디스크에서 처음으로 읽어질 때까지의 초기 대기시간(initial latency)이 너무 길어지게 된다는 단점이 있다.

이러한 초기 대기시간은 사용자가 비디오를 처음으로 선택하였을 때는 물론이고, fast forward 또는 fast rewind와 같은 VCR 제어를 선택한 경우나 임의의 위치로 점프하여 재생하는 경우에도 필요하게 된다. 기존의 비디오 저장 서버는 주로 영화를 재생하는데 활용되었으므로 십여 초의 초기 대기시간도 별 문제가 되지 않았으나, 비교적 짧은 시간 동안 재생되는 비디오나 사용자의 인터랙션이 빈번한 경우에는 짧은 초기 대기시간도 동시에 지원할 수 있는 사용자의 수의 최대화와 함께 시스템에서 지원하여야 할 필수적인 요구사항이다.

초기 대기시간을 감소하기 위해서 설정된 주기별로 여러 사용자의 디스크 read 요구를 일괄적으로 처리하는 방식을 택하지 않고 사용자의 디스크 read 요구를 개별적으로 처리하는 방법을 채택한 시스템[1, 2]도 있으나, 이러한 방법은 비디오 저장 서버가 동시에 지원할 수 있는 사용자의 수가 현저하게 감소하여 비디오 저장 서버의 경제성이 작아진다는 단점이 있다. 일부 시스템[11]에서는 개별 사용자 별로 디스크 read 요청을 하는 주기만 있고 시스템 전체적으로 여러 사용자가 공유하는 주기는 없는 경우도 있는데, 이러한 방법 역시 초기 대기시간의 감소효과는 있으나 디스크를 first-come first-served 방식으로 처리하는 결과를 초래하여 동시에 지원 가능한 사용자의 수가 현저하게 감소하게 된다는 단점이 있다.

기존의 연구에서는 보다 많은 사용자를 지원하거나 [3, 4, 7, 9] 초기 대기시간을 감소하는 방법[1, 2]을 별개의 연구 목표로 하였으나, 이 두 가지는 저장 서버의 경제성과 활용성에 반드시 필요한 기능이다. 본 논문에서는 비디오 저장 서버에서 지원할 수 있는 사용자 수를 최대로 하면서 동시에 초기 대기시간을 적절한 시간 이내로 할 수 있는 방법을 제시한다.

2. 제안하는 Scheme

비디오 데이터는 매우 큰 저장 공간과 높은 전송 대역폭을 필요로 하므로 비디오 저장 서버가 여러 개의 디스크를 사용하는 것은 필수적이다. 본 논문에서는 하나의 프로세싱 노드와 이의 제어를 받는 소수의 디스크로 구성되는 소규모의 비디오 저장 서버를 대상으로 하고 있으며, 이러한 소규모의 저장 서버는 클러스터링되어 대규모의 저장 서버를 구성할 때에도 사용될 수 있다. 본 논문에서는 디스크의 수가 d개라고 가정하고 설명한다. 기존의 연구에서는 라운드(round) 마다 read unit 크기의 비디오를 d개의 디스크 전체에서 읽어오는 방법을 택하고 있다[7, 12]. 이러한 기존의 스트라이핑(striping) 방법은 각 디스크 간의 load balancing에는 효과가 있으나, 한 디스크에서 읽어오는 데이터의 크기가 작게 되어 디스크를 효율적으로 사용하지 못하게 되고, 결과적으로 다수의 디스크를 사용하였음에도 불구하고 동시에 지원할 수 있는 사용자의 수가 그리 크지 않다는 단점이 있었다.

이러한 문제를 해결하기 위해 본 논문에서는 기존 스트라이핑 방식에 비하여 각 디스크에서 읽어오는 데이터의 크기가 2배가 될 수 있도록 하는 비디오 데이터의 배치 방법을 제안하고, 이러한 경우 각 디스크 간의 부하가 불균형될 수 있는 문제는 인기 비디오를 중복으로 저장하여 해결하는 방법을 제안한다. 또한 초기 대기시간이 사용자가 허용하는 일정 시간이 넘지 않도록 하는 디스크 스케줄링 방법도 제안한다.

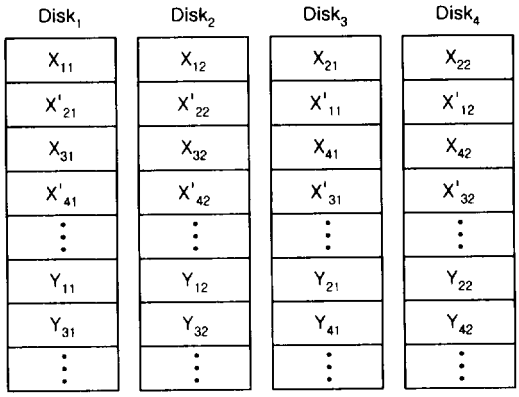
2.1 데이터 배치 및 Retrieval 방법

디스크에서 읽어오는 read unit이 클수록 디스크 액세스의 오버헤드 비율이 작아지게 되어 디스크를 효율적으로 활용할 수 있기 때문에, 본 논문에서는 $d/2$ 개의 디스크에 한 라운드에서 읽어야 할 read unit 만큼의 데이터를 저장하는 방식을 제안한다. 이러한 방식은 라운드 마다 d 개의 디스크 전체에서 데이터를 읽어오는 기존 스트라이핑 방식에 비하여 각 디스크에서 읽어오는 데이터의 크기가 2배가 되므로, 디스크 액세스 시간 중에서 디스크 헤드가 읽고자 하는 데이터의 시작부분 위에 도달할 때까지 기다리는 회전 대기시간(rotational latency)의 비율을 감소하여 결과적으로 보다 많은 수의 사용자를 지원할 수 있게 한다. 그러나 이러한 방법은 각 디스크 간의 부하가 불균형되는 문제가 발생할 수 있으므로 일부 비디오를 중복으로 저장하여 load balancing하는 방법을 제안한다.

영화나 뮤직 비디오를 대상으로 하는 VOD 시스템에서는 일주일에 수 백번씩 선택되는 인기 비디오(hot movie)는 물론이고, 일주일에 한 두번 이하가 선택되는 비인기 비디오(cold movie)도 저장되어 있다. 본 논문에서는 이러한 인기도 편차가 존재함을 활용하여 저장 서버에 저장되는 비디오 중 선택 빈도가 높으리라고 예상되는 상위 10% 정도의 비디오를 인기 비디오로 분류하여 중복 저장하는 방법을 제안한다.

(그림 1)은 4개의 디스크로 구성된 저장 서버에 인기 비디오 X와 비인기 비디오 Y를 저장하는 방법을 예시한다. 디스크가 4개인 경우이므로 라운드 마다 읽어오는 read unit은 2개의 디스크에 분할하여 저장하였으며, 인기 비디오 X의 시작 위치와 복사본인 X′의 시작 위치가 다르도록 중복 저장하였다. 하나의 디스크 내에서 X_i, X'_j, Y_k의 상대적 위치에 대한 제한은 없으며, 단지 각각의 데이터가 연속적인(contiguous) 공간에 저장되어 있기만 하면 된다. 인기 비디오가 시간이 지남에 따라 비인기 비디오로 분류되면 중복 저장된 2개의 파일 중 하나를 삭제하면 된다.

(그림 1)의 X_i와 Y_i는 각 비디오가 디스크에서 읽어지기 시작된 후 i번째 라운드에서 읽어야 할 데이터를 나타낸다. 예를 들어 비디오 Y가 선택된 경우에는 (Y_{11}과 Y_{12})가 첫번째 라운드에 읽어오는 read unit이 된다. 저장 서버가 한 라운드 동안에 읽어야 할 read unit의 적절한 크기는 3절에서 설명되며, 각 디스크에 저장되는 Y_{ij}는 read unit의 $1/(d/2)$ 크기이다. 사용자가 인기 비디오 X를 선택하면, 좌측과 우측의 디스크 부하에 따라 (X_{11}과 X_{12}) 또는 (X'_{11}과 X'_{12})를 첫번째 라운드에 읽어오게 된다. 그러나 비인기 비디오 Y를 선택한 경우에는 각 디스크의 load에 관계없이 시작 부분이 저장된 디스크에서부터 순차적으로 읽어오게 된다. 만약 모든 비디오를 이러한 방식으로 읽어오면 일정한 순간에 (그림 1)의 좌측 디스크인 디스크 1과 2를 액세스하는 사용자 수와 우측 디스크인 디스크 3과 4를 액세스하는 사용자 수가 크게 다른 경우가 발생할 수 있다. 이와 같은 hot spot 문제가 발생하면 저장 서버 전체에서 지원되는 사용자 수가 작아지게 되므로, 좌측 디스크들과 우측 디스크들을 액세스하는 사용자 수가 균형이 되도록 하는 방법이 필요하다.



(그림 1) 비디오 데이터를 배치하는 방법

이를 위해 본 논문에서는 인기 비디오의 경우에는 좌측 디스크군과 우측 디스크군을 액세스하고 있는 사용자 수를 비교하여 사용자 수가 적은 쪽에서 시작되도록 스케쥴링하는 방법으로 디스크를 사용하는 사용자의 수가 균형을 이루도록 하였다. 이러한 방법은 인기 비디오를 선택한 사용자 수가 현재 사용자 수의 50% 이상이면 항상 완전하게 load balancing이 이루어지게 되며, 인기 비디오를 선택한 사용자 수가 현재 사용자 수의 50% 이하인 경우에도 좌우 디스크군의 사용자 수의 차이보다 크기만 하면 load balancing이 이루어지는 효과가 있다.

저장 서버에서 동시에 지원 가능한 최대 사용자 수가 N인 경우, 새로운 사용자의 디스크 read 요구가 시작될 방향의 디스크들을 액세스하고 있는 사용자 수

가 $N/2$ 미만이면 새로운 사용자는 디스크를 액세스할 수 있도록 허용된다. 그러므로 본 논문의 방법은 비인기 비디오가 집중적으로 선택되고 이들이 한쪽으로 편중된 예외의 경우에도 사용될 수 있으나, 이러한 경우에는 동시에 지원할 수 있는 사용자의 수가 다소 감소하게 된다. 기존의 스트라이핑 방법과의 사용자 수 비교는 5절에서 설명된다.

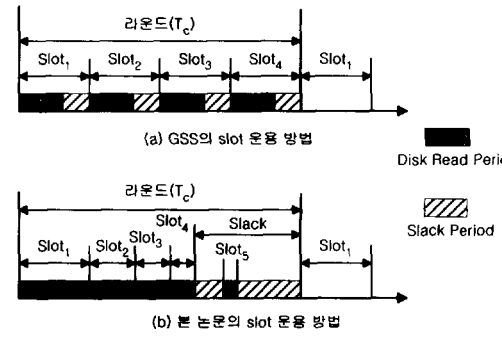
2.2 디스크 스케쥴링 방법

비디오 저장 서버가 경제성을 갖기 위해서는 지원할 수 있는 사용자 수를 최대화하여야 하기 때문에 디스크 read unit은 가능한 범위 내에서 최대로 정하여야 한다. 그러나 이러한 경우 라운드 시간이 커지게 되어, 새로운 비디오가 선택되었을 때부터 시작 부분의 데이터를 디스크에서 처음으로 읽어올 때까지의 초기 대기시간(initial latency)이 커지게 된다는 문제가 생긴다. 이러한 문제를 해결하기 위해 본 논문에서는 라운드를 여러 개의 slot으로 나누고, 각 slot에서는 디스크 read 요구를 SCAN 방식으로 처리하는 방식을 택하였다.

이처럼 라운드를 여러 개의 slot으로 나누는 방법은 Philip S. Yu 등이 제안한 Grouped Sweeping Scheme(GSS)[12]에서도 사용된 방법이며, GSS에서는 라운드를 일정한 크기의 slot으로 나누고 각 slot에 디스크 read 요구를 균등하게 분배하는 방식으로 시스템에서 필요로 하는 메모리를 최소화하려고 하였다. GSS에서는 각 slot에 디스크 read 요구를 균등하게 분배하는 것이 필수적이므로 (그림 2)의 (a)에 예시된 바와 같이 각 slot에 할당된 디스크 read가 처리된 후 다음 slot이 시작될 때까지 대기하였다가 다음 slot에 할당된 디스크 read 요구를 처리하는 방식으로 스케쥴링된다. 이 방법에서는 라운드에서 처리할 모든 디스크 read는 라운드 시작 전에 각 slot에 균등하게 할당하며 라운드 시작 이후에 요구된 새로운 사용자의 디스크 read 요구는 다음 라운드에서 처리되기 때문에, 최악의 경우에 새로운 사용자는 라운드 시간의 2배 동안 대기한 후에 비디오의 시작 부분을 읽어오게 되는 문제가 발생한다.

이러한 문제를 해결하기 위해 본 논문에서는 각 slot이 시작될 때 새로 시작한 비디오 스트림을 우선적으로 처리하는 방식으로 스케쥴링하여 새로운 사용자를 가장 빠른 slot에서 처리되도록 하였다. 즉, 각 slot의 시작 시점에 새로운 사용자가 있는지를 검사하여 새로운 사용자들의 디스크 read 요구를 포함하여 각 slot에서 처리할 수 있는 최대수의 디스크 read 요구를 처리하는 방식으로 스케쥴링한다. 이와 같은 스케쥴링 방법에 의하여 특정 slot이 시작된 직후에 도착한 새로운 비디오 스트림도 다음 slot에서는 반드시 처리되므로, 본 논문의 방법에서는 새로운 사용자는 최악의 경우에도 slot 시간의 2배 이내에 디스크에서 데이터를 읽어오게 된다.

또한 하나의 slot이 완료되면 다음 slot을 즉시 시작하도록 하고 디스크에서 읽어온 데이터는 즉시 전송하여 네트워크의 혼잡에 대비할 수 있도록 하였다. (그림 2)의 (a)에 예시한 바와 같이 GSS에서는 각 slot의 후반부 마다 작은 slack이 발생하는데 비하여, 본 논문에서는 (그림 2)의 (b)에 예시한 바와 같이 라운드의 후반부에 하나의 커다란 slack이 발생하게 된다. 이러한 slack은 마지막 slot이 수행된 후에 도착한 새로운 사용자들의 디스크 read 요구를 처리하기 위한 extra slot에 사용될 수 있으므로, 본 논문의 방법은 새로운 사용자가 가장 빠른 시간에 디스크를 액세스할 수 있도록 한다. (그림 2)의 (b)에 나타난 Slot5는 다음 라운드가 시작될 때까지의 여유 시간인 Slack time 동안에 도착한 새로운 사용자의 디스크 읽기 요구를 처리하는 extra slot의 예이다.



(그림 2) Slot의 운용 방법 비교

저장 서버에서 동시에 지원할 수 있는 사용자의 수가 N이고 한 라운드 내의 slot 수가 g인 경우, 각 slot에서는 최대 N/g개의 디스크 read 요구를 처리하게 되며 이러한 경우 slot의 최대 처리 시간은 라운드 시간(T_c)의 $1/g$이다.

3. 변수 설정 방법

본 논문에서 제안하는 방법을 적용한 비디오 저장 서버를 구현하기 위해서는 복잡하게 관련되어 있는 여러 가지 변수들의 값을 적절하게 결정하는 과정이 필수적이다. 즉, 라운드 시간, slot의 수, 디스크 read unit의 크기, 비디오 저장 서버의 메모리 크기, 최대 지원 가능한 사용자 수를 결정하는 방법이 필요하며 이러한 변수들(variables)과 그 기호는 <표 1>과 같다.

〈표 1〉 값을 결정해야 할 변수들

변 수 내 용	변 수 기 호
라운드 시간	T_c
라운드 내의 slot 수	g
디스크 read unit의 크기	*readUnitSize*
전체 메모리 크기	B
최대 사용자 수	N

비디오 저장 서버에서 사용할 디스크를 선택하면 해당 디스크의 특성에 따라 디스크의 최소 data rate, seek time, rotation time, minimum startup time 등의 값이 결정되며, 디스크의 수(d)는 PCI bus, SCSI card, disk의 data bandwidth가 균형이 되도록 적절한 수가 선택된다. 디스크의 수(d)를 결정하는 과정은 4절에서 설명된다. 허용 가능한 초기 대기시간($t_{latency}$)과 비디오의 재생 속도(V_p)의 값도 각각 시스템 설계자의 판단과 선택된 비디오의 타입에 의하여 그 값이 결정되므로 상수로 처리한다. 이러한 상수들(constants)의 내용 및 그 기호는 <표 2>와 같다.

〈표 2〉 상수들

상 수 내 용	상수기호
디스크의 수	d
디스크의 최소 data rate	*dataRate*
디스크의 최대 seek time	t_{seek}
디스크의 rotation time	t_{rot}
디스크의 minimum startup time	t_{min}
허용 가능한 초기 대기시간	$t_{latency}$
비디오의 재생 속도	V_p

최근에 생산되는 디스크는 안쪽 트랙과 바깥쪽 트랙의 크기가 차이가 나므로[8], <표 2>의 상수 trackSize는 크기가 가장 작은 안쪽 트랙의 크기로 설정한다. <표 2>의 다른 상수들도 worst case의 값으로 설정하여 최대수의 사용자가 동시에 지원되는 상황에서도 저장 서버 시스템이 안정적으로 운용되도록 한다.

본 논문에서는 초기 대기시간의 최소화와 함께 지원 가능한 사용자 수의 최대화를 목표로 하고 있으므로 디스크 read unit을 최대한 크게 하였다. 이러한 경우 라운드 시간(T_c)이 허용 가능한 초기 대기시간($t_{latency}$)보다 훨씬 클 수 있으므로 각 라운드를 g개의 slot으로 분할하여 각 비디오의 실제 초기 대기시간이 허용 가능한 초기 대기시간($t_{latency}$)보다 작도록 하였다. 이러한 경우 각 라운드 마다 g번의 디스크 seek가 발생하게 된다. 또한 좌측과 우측의 디스크군에서는 각각 최대 N/2 개의 디스크 read 요구가 처리될 수 있다.

본 논문에서는 read unit을 $d/2$ 개의 디스크에 저장하므로, 각 디스크에서 *readUnitSize*/($d/2$) 크기의 데이터를 읽어오게 되며 *readUnitSize*/(($d/2$)*dataRate)의 시간이 소요된다.

디스크에서 read unit 크기의 데이터를 읽어오기 위해서는 이처럼 실제로 데이터를 읽는 시간(data transfer time) 외에도 디스크 헤드가 데이터의 시작 부분 위에 도달할 때 까지 기다려야 하므로, 한 스트림의 read unit을 읽어오기 위해 최악의 경우 각 디스크는 1회전의 회전 대기시간(rotational latency)을 필요로 한다. 그러므로 본 논문의 방법에서 라운드 시간(T_c)은 다음과 같은 식으로 표현된다.

$$T_c = \frac{N}{2} t_{\min} + g\, t_{seek} + \frac{N}{2}\left(\frac{readUnitSize}{d/2\ dataRate} + t_{rot}\right) \quad (1)$$

또한 비디오가 중단없이 연속적으로 재생되기 위해서는 read unit의 데이터가 클라이언트 시스템에서 재생되는 시간(*readUnitSize*/V_p)이 디스크 read 주기(T_c)보다 커야 하므로 $readUnitSize / V_p \geq T_c$ 이며, 본 논문의 방법은 *readUnitSize*의 데이터를 $d/2$ 개의 디스크에 저장하므로 *readUnitSize*는 $V_p\ T_c$ 보다 크거나 같으면서 $d/2$의 배수여야 한다. 또한 *readUnitSize*가 필요 이상으로 크면 메모리가 낭비되므로 *readUnitSize*는 다음과 같이 정한다.

$$readUnitSize = \left\lceil \frac{V_p T_c}{\frac{d}{2}} \right\rceil \frac{d}{2} \quad (2)$$

본 논문에서 제시한 방법이 사용될 경우에 최대 초

기 대기시간은 slot이 수행되는 시간(T_c/g)의 2배이며, 이는 허용 가능한 초기 대기시간($t_{latency}$)보다 작아야 하므로, 다음과 같은 식을 도출할 수 있다.

$$2\frac{T_c}{g} \le t_{latency} \tag{3}$$

가장 경제성이 있는 메모리 크기와 라운드 시간을 결정하기 위해 위의 식을 바탕으로 slot의 수(g)가 1인 경우부터 순차적으로 계산한다.

먼저 식 (3)으로부터 최대 크기의 T_c를 다음 식을 사용하여 계산한다.

$$T_c = \frac{g\ t_{latency}}{2} \tag{4}$$

식 (4)에서 구한 T_c의 값을 식(2)에 대입하여 *readUnitSize*의 값을 구한다. 즉,

$$readUnitSize = \left\lceil \frac{V_p\ g\ t_{latency}}{d} \right\rceil \frac{d}{2} \tag{5}$$

지원 가능한 최대 사용자 수는 식 (1)에서 N을 구하는 식을 도출하여 계산한다. 즉,

$$N = \frac{2(T_c - g\ t_{seek})}{t_{\min} + \left(\frac{2}{d}\frac{readUnitSize}{dataRate} + t_{rot}\right)} \tag{6}$$

여기에서 T_c와 *readUnitSize*는 식 (4)와 식 (5)를 사용하며, N은 사용자의 수이므로 식 (6)의 계산값의 소수점 이하를 절삭한 정수로 정한다.

비디오 저장 서버에서는 read unit 크기의 데이터를 읽어오는 과정과 이를 네트워크를 통하여 전송하는 과정이 동시에 수행되므로 각 사용자 별로 2개의 read unit를 저장하는 버퍼(buffer)가 필요하게 되며, 저장 서버 전체적으로는 (2 * read unit의 크기 * 사용자 수) 만큼의 버퍼가 필요하게 된다. 위의 단계에서 g가 1인 경우의 *readUnitSize*와 N이 정해졌으므로, 저장 서버의 메모리 크기를 다음 식을 사용하여 계산한다.

$$B = 2\ readUnitSize\ N \tag{7}$$

다음 단계에서는 B 크기의 메모리를 사용하여 N명의 사용자를 지원하는 비디오 저장 서버가 경제적인지를 판단한다. 더 많은 메모리를 사용하여 더 많은 사용자를 지원하는 시스템을 개발하고 싶으면 g를 1 증가하고 위의 단계를 반복한다. 계산이 반복됨에 따라 *readUnitSize*는 일정하게 증가하는데 비하여 N은 logarithm 곡선 형태로 증가하므로, 몇 차례의 반복적인 계산으로 가장 경제적인 규모의 메모리를 B만큼 사용하여 얻을 수 있는 최대의 사용자 수 N을 계산할 수 있다.

4. 구현 사례

본 절에서는 제안한 방법을 PC를 사용하는 중규모의 저장 서버의 구현에 적용한 과정을 기술한다. PCI bus의 data bandwidth는 132 MB/s이며, 구현에 사용한 AHA-3950U2B Ultra2 SCSI adapter는 80MB/s의 data transfer rate를 지원하고 있으므로, 2개의 SCSI adapter를 사용하였다. 본 연구에서 사용한 IBM Ultrastar 9ES 디스크[6]의 sustained data rate가 8.4 ~ 13.4 MB/s이며 전체 디스크의 최대 data rate가 PCI bus의 data bandwidth를 초과할 수 없으므로, 각 SCSI adapter에 4개의 디스크를 연결하는 형태로 총 8개의 디스크를 비디오 데이터의 저장에 사용하였다. 서버의 OS는 Linux를 사용하였으며, 저장 서버 프로그램과 함께 별도의 디스크에 저장되어 있다. 비디오 데이터는 매우 큰 read unit을 필요로 하므로 Linux Ext2 File System을 멀티미디어 데이터를 다룰 수 있도록 수정하여 사용하였다.

<표 3>은 본 연구에 사용된 MPEG-1 비디오와 디스크의 특성을 포함한 상수들의 실제 값이다. IBM Ultrastar 9ES Disk 1개의 저장 용량은 9.1 GB이며, 8개의 디스크에는 MPEG-1으로 압축된 4분 분량의 music video 1600여 곡을 저장할 수 있다. 가장 안쪽 트랙의 디스크 data rate인 8.4 MB/s를 최소 data rate로 사용하였으며, 허용 가능한 초기 대기시간은 1.0초로 설정하였다.

〈표 3〉 구현에 사용된 상수들

상 수 기 호	실 제 값
d	8
dataRate	8.4 MB/s
t_{seek}	15 ms
t_{rot}	8.33 ms
t_{min}	2 ms
$t_{latency}$	1.0 s
V_p	187.5 KB/s

이와 같은 상수의 값과 제 3절에서 기술한 방법을 사용하여 slot의 수, 라운드 시간, 스트림 1개의 read unit 크기, 최대 지원 가능한 사용자 수, 비디오 저장 서버의 메모리 크기를 결정하는 과정이 <표 4>에 나타나 있다.

〈표 4〉 저장 서버 설계과정의 예

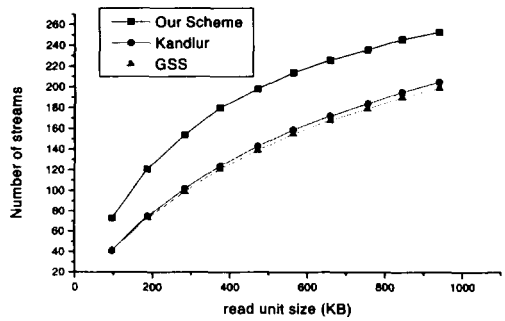
g	T_c (초)	*ReadUnitSize* (KB)	N	B (MB)
1	0.5	96	73	15
2	1.0	188	121	46
3	1.5	284	154	88
4	2.0	376	180	136
5	2.5	472	198	187
6	3.0	564	214	242
7	**3.5**	**660**	**226**	**299**
8	4.0	756	236	356
9	4.5	844	246	416
10	5.0	940	253	476

본 연구에서는 구현에 사용한 서버 시스템이 128 MB 크기의 메모리를 3개까지 사용할 수 있는 하드웨어 특성이 있었으므로, 사용자 수를 최대화하기 위해 필요한 버퍼의 크기가 299 MB이고 지원 가능한 최대 사용자 수가 226명인 경우를 택하였다. 이 경우 라운드 시간은 3.5초 이고, slot의 개수는 7개이며, read unit의 크기는 660 KB이다.

5. 성능 분석

본 절에서는 제안한 방법의 성능을 기존의 방법들과 비교하여 분석한다. 동일한 조건에서 비교되도록 하기 위해 본 절의 비교에서는 각 방법이 4절에서 설명한 시스템 구성과 8개의 multi-zone disks를 사용하였다고 가정한다. (그림 3)은 다중 디스크(Disk Array)를 이용한 대표적인 기존의 방법인 [7, 12]와 제안한 방법의 최대 지원 가능한 사용자 수를 비교하였다.

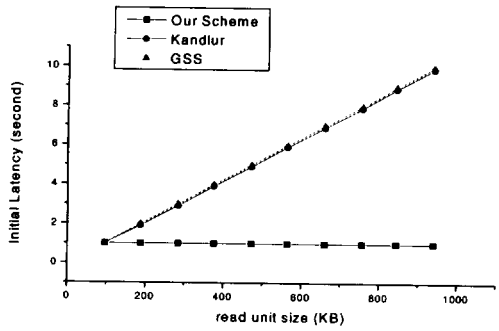
Kandlur의 방법[7]에서는 각 라운드 마다 d 개의 디스크에서 데이터를 읽어오며 각 디스크는 SCAN 방식으로 처리되므로, 라운드 시간 $T_c = N\ t_{min} + t_{seek} + N\ (readUnitSize/(d*dataRate) + t_{rot})$이다. GSS[12]에서도 각 라운드 마다 d 개의 디스크에서 데이터를 읽어오지만, 각 라운드는 동일한 크기의 slot으로 나누어지며 각 slot은 SCAN 방식으로 처리되므로, 라운드 시간 $T_c = N\ t_{min} + g\ t_{seek} + N\ (readUnitSize/(d*dataRate) + t_{rot})$이다. (그림 3)은 이러한 식들로부터 최대 지원 가능한 사용자 수 N을 도출하여 비교한 그래프이다.



(그림 3) 최대 지원 가능한 사용자 수 비교

본 연구의 구현에서 사용한 3.5초의 라운드 시간과 660 KB의 read unit이 선택된 경우 Kandlur의 방법은 172명의 사용자를 지원 할 수 있고, GSS에서는 168명의 사용자를 지원 할 수 있다. 이러한 수치는 제안한 방법이 이들 방법보다 각각 31%와 35% 더 많은 사용자를 지원할 수 있음을 나타낸다. 이러한 성능의 향상은 각 라운드 마다 $d/2$ 개의 디스크에서 데이터를 읽어올 수 있는 데이터 배치 방법으로 디스크의 효율이 높아졌기 때문이며, 인기 비디오의 중복 저장으로 소비된 10%의 디스크 저장 공간의 비용보다 월등한 소득이라고 할 수 있다.

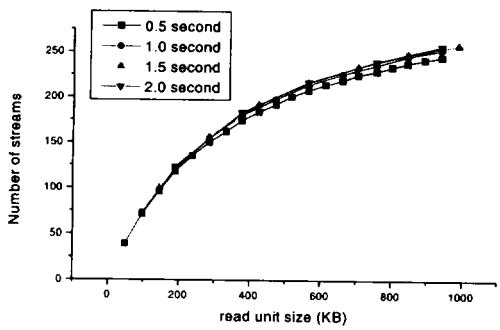
[10]에서는 수 백개의 디스크가 사용될 경우 디스크 전체에서 데이터를 읽어올 수는 없으므로 디스크들을 몇 개의 그룹으로 나누어 그룹 내에서만 스트라이핑하는 방법을 제안하고, 몇 개의 그룹으로 나누는 경우에 부하의 불균형이 가장 적은지를 분석하였다. 그러나 그룹 간의 부하가 불균형될 수 있는 문제는 해결하지 못하였으며, 그룹 내의 스트라이핑 방법은 Kandlur의 방법과 동일하다. 이처럼 대규모의 디스크가 사용된 경우, 몇 개의 그룹으로 나누고 이들 그룹 내에서만 스트라이핑하는 방법은 Tiger Shark 시스템[5]에서도 사용되었다.



(그림 4) 최대 초기 대기시간의 비교

(그림 4)에서는 read unit의 증가에 따른 최대 초기 대기시간을 비교하였다. Kandlur의 방법과 GSS의 초기 대기시간은 라운드 시간의 2배이며 read unit의 증가에 따라 라운드 시간이 증가하므로, read unit이 커지면 초기 대기시간도 커지게 된다. 반면에 본 논문의 방법은 라운드 시간이 를 경우에는 여러 개의 slot으로 나누어 수행되며, 새로운 사용자의 디스크 read 요구는 우선적으로 가까운 slot에서 처리되므로 초기 대기시간은 항상 일정하다.

제안한 방법과 대부분의 저장 서버[5, 7, 9]에서는 각 스트림 당 read unit의 2배에 해당하는 버퍼를 필요로 하며, GSS는 각 스트림 당 read unit의 $(1+1/g)$배에 해당하는 버퍼를 필요로 한다. GSS는 메모리 감소 측면에서는 우수한 방법이지만, 각 slot에서 사용하는 버퍼가 공유되기 때문에 일부 스트림에서 부분적인 전송 지연이 있는 경우에도 공유된 버퍼의 overflow가 발생할 위험이 있으므로 이러한 경우에 대비하기 위해 구현이 복잡해진다는 단점이 있으며, 초기 대기시간이 길다는 단점이 있다.



(그림 5) 허용 가능한 초기 대기시간별 사용자 수 비교

(그림 5)는 각기 다른 허용 가능한 초기 대기시간($t_{latency}$)의 경우 제안한 방법이 몇 명의 사용자를 지원할 수 있는지를 나타내고 있다. 허용 가능한 초기 대기시간을 크게 잡으면 동일한 라운드 시간의 경우 slot의 수가 작게 되므로, 디스크의 seek time으로 소비하는 시간이 작아지게 되어 보다 많은 사용자를 지원할 수 있게 된다. 그러나 (그림 5)에서 알 수 있듯이 허용 가능한 초기 대기시간($t_{latency}$)의 크기에 따른 사용자 수의 차이는 별로 크지 않았다. 이는 방대한 크기의 데이터를 처리하는 비디오 저장 서버에서는 디스크의 seek time 보다 rotation time이 동시에 지원할 수 있는 사용자 수의 결정에 중요한 역할을 하기 때문이다.

6. 결 론

동시에 지원할 수 있는 사용자 수의 증대와 초기 대기시간의 감소는 비디오 저장 서버가 상용 서비스에 사용되기 위해서 갖추어야 할 필수적인 요소이다. 사용자 수의 증대를 위해 디스크에 비디오 데이터를 스트라이핑하는 기존의 방법은 초기 대기시간이 커지게 되는 단점이 있었으며, 초기 대기시간을 감소하기 위해 디스크 read를 first-come first-served 방식으로 처리하는 방법은 사용자 수가 크게 감소되는 단점이 있었다. 본 논문에서는 디스크의 회전 대기시간이 차지하는 비율이 작아지도록 하는 데이터 배치 방법과 초기 대기시간이 일정 시간을 넘지 않도록 하는 디스크 스케줄링 방법을 사용하여, 사용자수를 최대화하면서 동시에 초기 대기시간을 최소화하는 방법을 제안하였다.

제안한 방법은 기존의 스트라이핑 방법에 비하여 각 디스크에서 읽어오는 데이터의 크기가 2배이므로 디스크를 효율적으로 활용하게 되어 사용자 수가 크게 증대되었다. 성능 비교에 나타난 바와 같이 방대한 크기의 데이터를 처리하는 비디오 저장 서버에서는 디스크의 seek time 보다 rotation time이 라운드 시간의 대부분을 차지하므로, read unit의 크기를 크게 할 수 있는 방법이 동시에 지원할 수 있는 사용자의 수의 최대화에 결정적인 역할을 함을 확인하였다.

기존의 방법은 read unit의 크기에 비례하여 초기 대기시간이 커지지만, 본 논문의 방법에서는 항상 일정한 시간 이내에 새로운 사용자의 디스크 read 요구가 처리된다. 이러한 성능의 향상은 대부분의 기존 연구와 마찬가지로 각 스트림 당 read unit의 2배에 해당

하는 더블 버퍼를 사용하되, 새로 시작한 비디오 스트림을 우선 처리하는 스케줄링 방법으로 얻을 수 있었다.

본 연구는 메모리의 가격이 매우 낮아진 현재의 상황을 반영하여 서버는 물론이고 클라이언트 시스템에도 충분한 크기의 메모리가 있다는 가정에서 수행되었으나, 향후에는 메모리가 충분한 PC와 메모리 크기가 매우 작은 PDA와 같이 클라이언트 시스템을 모두 지원하여야 하는 저장 서버의 설계와 구현을 수행할 계획이다.

참 고 문 헌

[1] E. Chang and H. Garcia-Molina, "Effective Memory Use in a Media Server," 23rd VLDB Conference, pp.496-505, Aug. 1997.

[2] E. Chang and H. Garcia-Molina, "BubbleUp : Low Latency Fast-Scan for Media Servers," Proc. of the 5th ACM International Conference on Multimedia, pp.87-98, Nov. 1997.

[3] D.J. Gemmel and J. Han, "Delay-Sensitive Multimedia on Disks," IEEE Multimedia, Vol.1, No.3, pp.56-67, Fall 1994.

[4] D.J. Gemmel, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe, "Multimedia Storage Servers : A Tutorial," IEEE Computer, Vol.28, Vo. 5, pp.40-49, May 1995.

[5] R.L. Haskin, "Tiger Shark A Scalable File System for Multimedia," IBM Journal of Research and Development, Vol.42, No.2, pp.185-198, 1998.

[6] IBM, "Hard Disk Drives," http://ssdweb01.storage.ibm.com/hardsoft/diskdrdl/ultra/9esdata.htm#3

[7] D.D. Kandlur, M.S. Chen, and Z.Y. Shae, "Design of a Multimedia Storage Server," IS&T/SPIE Int. Symposium on Electronic Imaging: Science and Technology, San Jose, pp.164-178, Feb. 1994.

[8] R.V. Meter, "Observing the Effect of Multi-Zone Disks," Proc. of the Usenix 1997 Technical Conference, Jan. 1997.

[9] A.L.N. Reddy and J.C. Wyllie, "I/O Issues in a Multimedia System," IEEE Computer, Vol.27, No.3, pp.69-74, Mar. 1994.

[10] P. Shenoy and H.M. Vin. "Efficient Striping Techniques for Multimedia File Servers," Proc. of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), pp.25-36, May 1997.

[11] Y.S. Son et al., "CrownFS : A Clustered Continuous Media server," Proc. of the International Conference on Parallel and Distributed Systems, Dec. 1997.

[12] P.S. Yu, M.S. Chen, and D.D. Kandlur, "Grouped Sweeping Scheduling for DASD-based Multimedia Storage Management," Multimedia Systems, Vol.1, No.1, pp.99-109, Jan. 1993.

마 평 수

e-mail : pmah@etri.re.kr

1985년 서울대학교 식물병리학과 졸업(학사)

1992년 City University of New York, USA 전산학과(석사)

1995년 Wright State University, USA 전산학과(박사)

1985년~1989년 시스템공학연구소 연구원

1989년~1990년 (주)태양금속 정보산업연구실 대리

1996년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 책임연구원

관심분야 : 멀티미디어 저장서버, 멀티미디어 검색, 웹 기술 등

조 창 식

e-mail : cscho@etri.re.kr

1993년 경북대학교 전자계산학과 (학사)

1995년 경북대학교 전자계산학과 (석사)

1995년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 연구원

관심분야 : 멀티미디어 저장서버, 객체지향 기술, 웹 기술 등

진 윤 숙

e-mail : ysjin@etri.re.kr

1993년 경북대학교 전자계산학과 졸업 (학사)

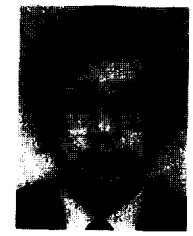
1996년 경북대학교 컴퓨터과학과 졸업 (석사)

1993년 2월~1993년 12월 삼성데이타시스템 근무

1995년 12월 시스템공학연구소 입소

1998년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 연구원

관심분야 : 멀티미디어 기술, 객체지향 기술, 웹 기술 등



신 규 상

e-mail : gsshin@etri.re.kr

1981년 성균관대학교 통계학과 (학사)

1983년 서울대학교 계산통계학과 (석사)

1983년~1996년 시스템공학연구소 연구원/선임연구원

1997년~1998년 시스템공학연구소 책임연구원

1998년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 책임연구원

관심분야 : 소프트웨어공학, 객체지향 기술, 멀티미디어 기술 등