論文
6-3-1

멀티미디어 서버에서 효율적인 선반입 서비스를 위한 디스크 파티션 스케쥴링 (An Efficient Pre-Fetching Service for Multi-media Server based on Disc Partition Scheduling)

최성욱(Sung-Wook Choi)[1)]

요약

멀티미디어 스트림은 일반적으로 용량이 크고, 서로 다른 미디어간의 동기화가 필요하며, 실시간으로 재생되어져야 한다는 특징이 있다. 그러므로 VOD 서버에 관계된 연구는, 궁극적으로 디스크 대역폭이나 버퍼의 크기 등 서버의 주어진 자원 한계 아래에서 얼마만큼 사용자의 수를 최대화 하느냐에 주된 관심이 되고 있다. 본 논문에서는 효율적인 멀티미디어 서비스를 위하여 서버의 자원을 동적으로 모니터하고 관리하여 효율적으로 서비스를 할 수 있는 선반입 관리정책을 제안한다. 시뮬레이션 해본결과 전통적인 방식보다 버퍼의 활용과 서비스 처리시간에서 약 20% 정도 향상된 성능을 보였는데, 이는 서비스 사용자의 수를 증가 시키는 문제와 밀접한 관련이 있다.

Abstract

Intensive studies have been made in the area of VOD server. Multimedia files in the VOD sever are characterized with the large volume of data, the requirements of synchronization and real-time playback of streams. The basic goal of the study is to find an efficient mechanism to allow maximum number of users under the limited resources such as Buffer size and disk bandwidth. we propose a efficient pre_fetching policy for multimedia services with dynamic monitoring and management of VOD sever resources. Simulation results show that the rate of buffer usage and service time of proposed scheme are about 28% performance improved than that of traditional methods. This implies that our method can allow much more users for given resources.

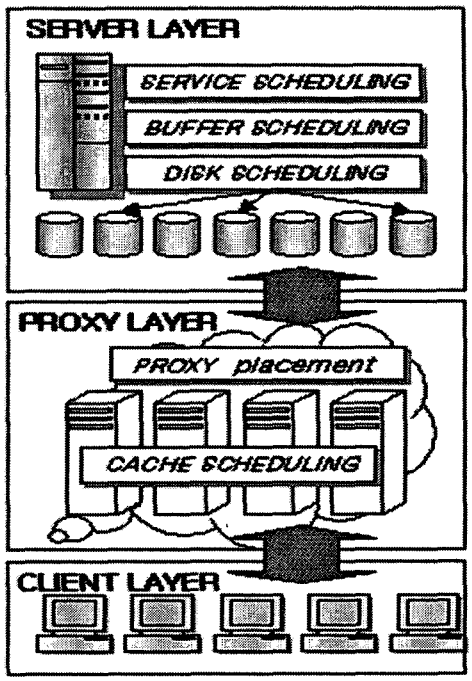
논문접수 : 2005. 5. 19.
심사완료 : 2005. 7. 9.

1) 정회원 : 시립인천 전문대학 컴퓨터 정보과 교수

1. 서론

1.1 VOD 서비스

텍스트, 오디오, 정지 화상에서 동영상이나 비디오 정보까지 아날로그 데이터를 디지털화(digitalization) 하는 기술과 수기가(GIGA)비트의 전송률을 가지는 통신망이 발달함에 따라 다양한 멀티미디어 형식의 정보를 공유할 수 있게 되어 우리의 일상생활에 많은 영향을 주고 있다. 일반적으로 VOD 서비스는 비디오나 오디오등 대용량 연속 데이터를 실시간으로 많은 사용자에게 제공해야하므로 서버나 네트워크의 과부하로 서비스의 지연 및 품질의 저하가 발생하기 쉽다. 그림1은 서버 계층, 플록시 계층, 사용자계층의 세 단계로 구성된 일반적인 VOD 시스템 구조이다. 서버계층에서의 스케줄링은 우선, 멀티캐스팅을 위하여 사용자들을 그룹화하기 위한 서비스 스케줄링[4,5]과 버퍼 공유 스케줄링[6,7], 디스크 스케줄링[8,9]으로 나눌 수 있다. 그 아래, 플록시 계층에서의 플록시 배치 전략은 복수개의 플록시를 사용하여 효과적으로 계층화[17]하는 방식과 네트워크 분산 형태의 협력 플록시[16]가 있다. 그 외에 클러스터링 서버 방식[19]과 오버레이 형태의 연합 서버[18]도 제안되고 있다.



[그림1] VOD시스템 구조

[Fig.1] *Architecture of VOD system*

그러나 이러한 캐싱이나 버퍼공유 정책은 액세스 시간이 상대적으로 긴 디스크의 접근횟수를 줄일 수가 있기 때문에, 신속한 서비스를 할 수 있지만, 대규모 디스크 저장장치에 비하여 제한적인 용량 때문에 서비스 변동에 따른 자원의 효과적인 관리 등 요구 패턴의 변화에 적응적으로 대처하기위한 정책이 가미되어야 한다. 본 논문에서는 스트림 블록 배치나 공유 버퍼 유닛, 선반입 캐시 등 서비스 자원을 모니터하여 효율적으로 사용자 서비스할 수 있는 캐시 관리 정책을 제안한다. 이를 위하여 2장에서는 멀티미디어 버퍼 스케줄링과 멀티미디어 저장 시스템의 스케줄링에 관한 관련 연구를 살펴보고 3장에서는 서버의 자원들을 효율적으로 관리하고 사용자의 서비스 처리율을 높이기 위한 선반입 공유 버퍼 스케줄링 정책을 제안한다. 4장에서는 이를 시뮬레이션해 보고 기존 방식과 비교 분석한다. 끝으로 5장에서는

결론과 함께 향후 연구 방향에 대하여 간략히 기술한다.

2. 관련연구

2.1 멀티미디어 버퍼 스케줄링

일반적으로 인터넷 문서를 메모리 캐싱 하기 위한 버퍼 재배치 기법들은 해당 문서의 작성 일자나 크기, 조회빈도, 문서끼리의 연관성 등을 이용하는데, 대표적인 기법으로는 LRU, LFU, SIZE, LRU-SIZE, LRU-MIN, LRFU이 있다[3]. 그러나 이러한 버퍼관리 정책들은 VOD 시스템에서 다루는 비디오 스트림 등의 연속 미디어 데이터를 관리하기에는 적합하지 않다. 비디오 스트림의 특징은 우선 대용량의 연속 데이터이다. 그러므로 자료를 압축하여 사용하는 방식만으로는 VOD 서버가 다수의 사용자를 수용하여, 연속적으로 서비스하기에는 여전히 문제가 된다. 연속 미디어를 위한 서버나 플록시의 버퍼 스케줄링은 비디오 스트림의 연속성에 착안하여 제안되었다. 비디오의 시작부분을 메모리에 저장 하는 Prefix Caching[10]과 비디오의 뒷부분을 저장하는 Suffix Caching[11], 그리고 현재 서비스 중인 비디오 세그먼트에 누락된 일부분의 비디오 프레임을 외부에서 다시 조달 받아 세그먼트를 완성하는 Patching방식[12,13]이 있다. 그중 Patching 기법은 동일 비디오를 요구하는 2개의 서비스 요청 사이에 시간 간격이 존재하면 나중에 도착한 서비스를 현재 서비스 중인 사용자 그룹에 포함 시켜서 스트림을 공유할 수 있도록 하고, 초반에 못 받은 스트림 부분만 서버가 추가로 전송하는 방식이다. 이 방식은 버퍼를 공유하는 사용자가 많을수록 서비스 자원을 효율적으로 관리할 수 있으며, 서비스 초기의 지연시간을 어느 정도 단축할 수가 있지만, 클라이언트 셋톱박스 내에 일정량의 메모리를 필요로 한다. Prefix 캐싱[10]은 비디오 스트림의 초기 일부분을 플록시에 저장해 둠으로써 새로운 사용자의 서비스 요청에 대하여 시간 지연 없이 즉시 플록시로부터의 서비스가 가능하고, 그 동안에 서버는 새로운 사용자에 대한 효율적인 스케줄링 정책을 수립할 수 있다는 장점이 있다. 이 방식도 어떤 비디오 스트림을 어떻게 효율적으로 플록시에 캐싱 할 것인가가 중요한 이슈라 할 수가 있다. 인터벌 캐싱(IC)[14]은 동일한 비디오에 대하여 서비스 요청이 시간차를 두고 발생할 경우 먼저 서비스한 스트림을 계속 버퍼에 남겨두어 뒤따라오는 서비스에서 이를 활용하는 방식이며, 연속 스트림의 버퍼 공유를 위한 전통적인 방식이라 할 수가 있다. 이 방식은 두 개의 서비스의 시간차만큼의 스트림을 버퍼에 캐시 하여야함으로 버퍼의 점유도가 커질 수가 있다. 때문에 Generalized IC기법[15] 등과 같이, 인터벌 캐싱을 토대로 버퍼의 활용률을 개선한 연구가 제안되었다. Pre-Fetching 기법[19,20,21]은 디스크나 버퍼, 네트워크 등 현재 서버의 자원에 여유가 잇을 경우, 현재 서비스 중인 비디오 스트림의 다음 부분을 서버나 플록시의 메모리에 미리 가져와 놓는 방식으로, VOD 서비스 환경에서 디스크 시스템이나 네트워크의 혼잡제어(Congestion Control)및 클라이언트 측의 재생 능력에 효과적으로 대응하기 위하여 제안되었으며, 이 방식은 무엇보다도 효율적인 디스크 스케줄링과 밀접한 관계가 있다.

2.1 멀티미디어 저장 시스템

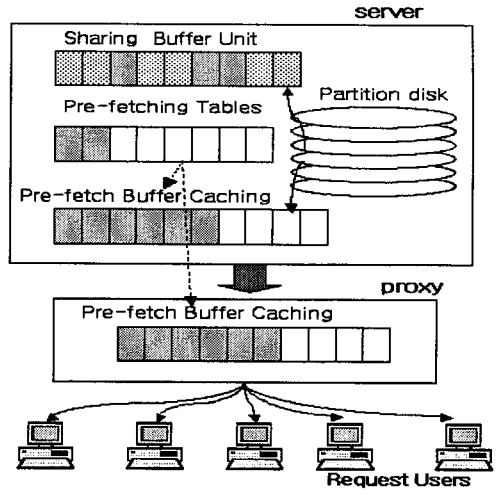
종래의 파일시스템 처리에서 사용되고 있는 방법인 회전하는 디스크의 표면을 헤드가 임의적으로 움직이면서 요구된 데이터를 검색하는 방식으로는 현재의 멀티미디어 데이터의 광범위한 특성을 수용할 수 없다 [10,11]. 멀티미디어 실시간 처리를 위한 대표적인 방식으로는 마감시간 우선 주사 방식(SCAN-EDF)[12], 고전적인 디스크 스케줄링 방법인 스캔과 실시간 보증을 위한 마감 시간 우선 방식을 조화한 방법으로 동일한 유사 마감 시간이 많을 때 접근

시간이 빠른 반면 시스템 환경 능력에 따라 스캔 시간의 부가로 인한 처리 능력에 차이가 난다. 집단 일소(Group Sweeping) 방식 [13]은 디스크 암의 동작을 최소화하기 위하여 각 스트림을 N개의 고정된 순서로 된 몇 개의 집단으로 분류하여 처리하는 방식으로 순환 시간(Cycle-Time)과 스트림의 자료 율에 따라 적절히 대응하는 버퍼가 있어 연속 처리를 가능하게 하나, 스트림을 동기화 할 경우 관련 스트림의 근접 배치에 어려운 점이 있다. 연속 미디어를 위한 디스크 스케줄링 방식으로는 디스크 분할(Disk Partition) 기법을 들 수 있는데, 디스크를 몇 개로 분할하여 종 방향 우선으로 주사시키는 방식이며, 횡(Track)방향은 다수의 다른 스트림을 배치할 수 있기 때문에 서로 다른 복수개의 스트림을 검색하기가 용이하다. 이 방식은 복수 스트림의 동시 전송을 필요로 하는 VOD 서버용 디스크 스케줄링에 적합한데, 주기가 동일한 미디어 스트림일 경우의 스케줄링 정책과[14], 다양한 주기 및 다양한 추출 단위를 위한 스케줄링 정책[15]이 제안되었다.

3. 디스크 파티션 선반입 모델

3.1 개요

스트림 미디어는 다양한 미디어와의 조합성, 시작과 끝의 명확성, 긴 서비스 시간성 등의 특징이 있기 때문에, 이를 서비스하는 VOD 시스템의 캐시나 디스크 스케줄링은 일반적인 파일 시스템이나 인터넷 웹 페이지 서비스를 위한 스케줄링과 차이가 있다. 특히 비디오 스트림의 경우는 앞뒤가 명확히 지정되어 있기 때문에 서버자원의 효율적인 활용을 위해서 선반입 스케줄링이 필요하다.



[그림2] 선반입 구조
[Fig.2] Pre-Fetching Structure

그림2는 본 논문에서 제안한 파티션 디스크를 활용한 선반입 스케줄링의 구조이다. 본 논문에서는 효율적인 선반입 스케줄링을 위하여 논리적으로 구분된 공유 버퍼 유닛과, 선반입 버퍼, 선반입 테이블을 하여 사용한다. 공유 버퍼 유닛은 현재 서비스하는 비디오 프레임과 사용한 프레임을 저장하여 뒤의 사용자들이 이를 재 사용하기위한 장소이지만, 본 논문에서는 디스크 스케줄링과 연계하여 스케줄링 함으로써 공유 버퍼의 점유를 낮춘다. 공유 버퍼에서 절약된 자원은 선반입 버퍼 캐싱에 활용되며, 여기에는 공유 버퍼 유닛과는 다르게 앞으로 사용하게 될 비디오 프레임을 저장한다. 물론 선반입 버퍼 캐시도 다른 사용자들에게 공유된다. 서버와 플룩시간의 네트워크 부하를 경감하기위하여 고 비트율을 가지는 비디오 스트림이나 상위 QOS 사용자들을 위하여 일부 플룩시 캐시를 활용할 수가 있다. 선반입 캐싱은 디스크나 네트워크의 사용을 균등하게 활용하기에 보다 유리하다. 한편 선반입 테이블은 버퍼공유에 해당 스트림 블록을 찾지 못했을 경우, 선반입 버퍼에 존재하는지를 체크하기 위한 정보를

가지고 있다. 디스크 파티션의 특징은 단일 디스크 표면에 n개의 원형 부분들 (*1, 2, 3, ... n*)인 파티션으로 나누고 각 파티션은 여러 개의 트랙들로 구성되며, 각 트랙은 고정된 블록으로 나누어진다. 그림 3에서는 4개 파티션을 갖는 디스크 파티션 방식을 보여주고 있다. 각 스트림은 일련의 연속된 블록으로 표현되며, 스트림의 연속된 블록은 각각 다른 파티션에 기록되기 때문에 헤드가 특정 파티션 내의 스트림의 시작 지점을 만나면 동일 방향으로 (디스크 안쪽에서 바깥쪽으로 혹은 그 반대 방향으로) 디스크 표면을 읽어 간다.

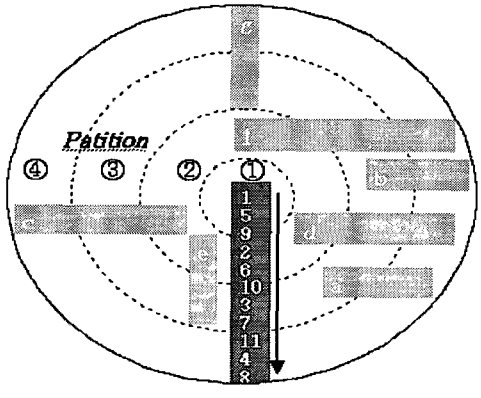


그림 3 디스크 파티션 방식의 구조
Fig 3 Layout of disk partition

예를 들어 어느 비디오 K 의 스트림 블록을 1에서 12까지를 그림 3에서 표시된 것처럼 저장하였다면, 파티션 1에는 프레임 1,5,9가 파티션 2에는 2,6,10 파티션 3에는 3,7,11 파티션 4에는 4,8,12가 저장되어진다. 그 외 각 파티션에는 비디오A 뿐만 아니라, 동기화를 위한 복합미디어 및 타 비디오 스트림(A, B, C, D, E, F, G 등)들도 비디오 K와 동일한 방식으로 저장된다. 비디오 K의 스트림 블록1에서 4까지를 읽기 위해서 헤드는 디스크 안쪽에서부터 바깥쪽으로 움직이며 파티션 1, 2, 3, 4, 순으로 읽어 간다. 만일 파티션 3에 위치한 스트림 블록 3을 읽을 차례에서 비디오 스트림 a, b, c, e, f, g의 파티션 3에 속해있는 또 다른 스트림 블록에 대한 읽기 요청이 있다면, 디스크의 회전에 의하여 짧은 시간 동안 한꺼번에 읽어 들일 수 있다. 상호 동기화가 필요한 복합 미디어를 서비스 할 경우에 실시간 디스크 스케줄링 기법간의 평균 검색 시간을 은 다음과 같이 나타낼 수가 있다. 우선 마감 시간 우선 주사(SCAN-EDF) 방식이나, 집단 일소(GSS) 방식은 임의 블록주사 (RANDOM BLOCK SCAN) 방식과 동일한 검색 패턴을 나타내므로 식(1)로 표시할 수가 있으며, 디스크 파티션 방식은 식(2)와 같다.

$$\overline{disk_{ser}} = B_n\{S(B_l/T_r + R_x + SE) + 2SE\} \quad (1)$$

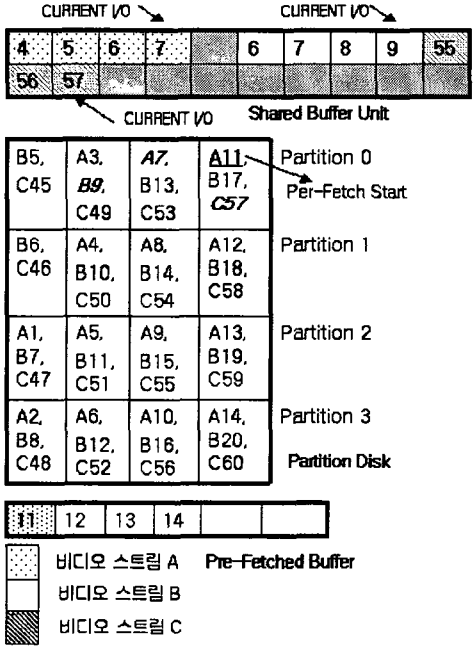
$$\overline{disk_{ser}} = B_n\{S(B_l/T_r + R_x + SE) + SE/P\} \quad (2)$$

여기서 c는 이동 실린더(cylinder)개수 이고, SE는 평균 디스크 조사시간, a는 헤드 안정 계수, b는 가속 안정 계수이다. B_n은 블록 수, S는 스트림 수, B_l은 블록 크기, T_r은 최대 전송률, R_x은 디스크 최대 회선 속도, P는 파티션 개수이다. 디스크 조사시간(SE)에서 마감시간 우선 방식 및 집단 일소 방식은 단거리 탐색이나 장거리 탐색이며, 디스크 분할 기법은 동기화 스트림이 동일 트랙 상에 존재함으로 조사시간이 거의 없거나, 최단거리 탐색으로 행할 수 있다. 또한 평균 조사시간은 장거리 탐색일 경우 $SE = a + bc$, 단거리 탐색일 경우 $SE = a + b\sqrt{c}$이며, 최단거리 탐색일 경우 약 2ms가 소요된다.

3.2 선반입 버퍼공유 스케줄링

본 논문에서 제안한 선반입 버퍼 공유 스케줄링은 버퍼 공유 스케줄링과 선반입 스케줄링으로 나누어지는데, 버퍼 공유 스케줄링은 기존 방식(IC)에 비하여 처리시간 및 버퍼 사용을 줄이고, 선반입 스케줄링은 선반입 처리시간을 단축시키고 서버의 부하를 개선하기 위하여 제안되었다. 우선 그림 4의 공유버퍼 유닛

에는 현재 서비스 중인 비디오 A의 스트림 블록 4, 5, 6, 7과 비디오 B의 스트림 블록 6, 7, 8, 9 그리고 비디오 C의 스트림 블록 55, 56, 57이 저장되어 있으며, 현재 비디오 A의 7번, 비디오 B의 9번, 비디오 C의 57번 스트림 블록이 각각 서비스 중이다. 그리고 4개의 파티션으로 구분되어진 디스크에는 파티션 0에서 3까지 세 가지(A, B, C)의 비디오 스트림 블록들이 종 방향 순서대로 나누어 저장되어 있는 모습을 나타내고 있다. 선반입 버퍼에는 현재 비디오 A의 11째 스트림 블록이 저장되는데, 나머지 11, 12, 13 스트림 블록은 앞으로 저장되어질 블록을 미리 표시한 것이다.



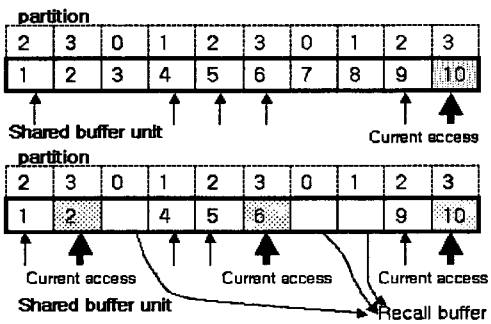
[그림4] 버퍼 공유 및 선반입 스케줄링
[Fig.4] Buffer Sharingand Pre-Fetching Scheduling

3.2.1 버퍼공유(Buffer Sharing)

공유 버퍼 유닛은 비디오 스트림과 같이 연속 매체를 서비스할 때 하나의 사용자에게 서비스한 버퍼의 내용을 이후에 계속되는 또 다른 사용자에게 재사용할 수 있게 하기 위하여, 한 번의 디스크 읽기에 의해서 앞서 서비스하고 있는 사용자의 비디오 스트림을 서비스 후에도 버퍼에 계속 남겨 두어 뒤따라오는 사용자가 이를 공유하여 사용 할 수 있게 하는 방식이다. 그러므로 버퍼의 사용 비용을 무시한다면, 일반적으로 버퍼 유닛의 사이즈는 클수록 서비스에 유리하다. 본 논문에서 제안한 파티션 디스크를 이용한 공유버퍼 정책은 보다 적은 수의 버퍼를 사용하여 비디오 스트림을 서비스 할 수 있다. 예를 들어 그림 4의 디스크 파티션에서 보는 바와 같이, 비디오 C에서 스트림 C45, C49, C53, C57은 한 번의 스캔으로 함께 읽어드릴 수가 있기 때문에 특별한 경우가 아니라면, C45에서 C57 까지 모두 공유버퍼에 저장 할 필요는 없다. 현재 비디오 A7, B9, C57은 함께 읽혀서 공유버퍼 유닛에 저장되어지고 비디오 A의 4, 5, 6. 7과 비디오 B의 6, 7, 8, 9 비디오 C 의 55, 56, 57은 선두 사용자 그룹들이 사용한 후에 버퍼에 남겨두면 뒤에 오는 사용자 그룹이 있으면 이를 재사용함으로써 신속한 서비스가 가능하고 디스크 액세스를 감소시킬 수가 있다. 그림5에는 공유 버퍼 스케줄링을 나타내었다. 2개의 표 중에서 위에 것은 인터벌 윈도의 크기가 10인 인터벌 케싱에서의 버퍼 공유 유닛이다. 현재 비디오 스트림 블록 10을 읽어 서비스하고 있다면, 나머지 비디오 스트림 블록 9, 6, 5, 4, 1은 다른 사용자들이 공유하고 있는 상태이다. 아래의 표는 본 논문에서 제안한 버퍼 공유 스케줄링이다. 비디오 스트림 블록 2, 6, 10은 순서적으로는 떨어져 있지만, 모두 동일한 파티션3에 속해 있으므로 한꺼번에 읽어 올수가 있기 때문에 총 7개의 버퍼가 사용되고 나머지 3개는 회수할 수가 있다. 만약 i번째 공유 버퍼에 포함되어 있는 스트림 블록에 대한 서비스 요청을 Si 라 하면, 버퍼 Bi 에 의해서 절약될 수 있는 I/O 량인 E[Ni]은 (3)의 식으로 표현 될

수 있다.

$$E[Ni] = \sum Si - 1 \quad (3)$$



[그림5] 공유버퍼스케쥴링

[Fig.5] shared_buffer_unit scheduling

3.2.2 선반입(Pre-Fetching)

선반입 스케쥴링은 서버가 사용자 서비스 요구를 감시하여 디스크 검색에 여유가 있을 때, 비디오 스트림 블록을 미리 읽어오는 방식이다. 그러나 통상 이러한 선반입 방식은 디스크의 부하를 증가시키기 때문에 스케쥴링의 적용이 제한적이 될 수가 있다. 본 논문에서 제안한 선반입 스케쥴링은 현재 검색 중인 동일한 파티션에 위치한 스트림 블록을 가져오기 때문에 추가되는 부하 정도가 적고 서비스 요청의 변동이 심한 경우에도 비교적 안정적으로 적용할 수가 있다. 그림 5에서 지금 디스크 헤드가 파티션 0 위치를 스캔 중이므로 선반입 스트림 블록 A11은 A7, B9, C55와 함께 스캔되어진다. 그리고 A12는 A8, B10, C56을 스캔하기 위하여 헤드가 파티션 1에 위치할 때 함께 읽어올 수가 있다. 또한 일반적인 VOD 서버는 멀티 캐스팅 서비스나 서버 자원의 효율적인 사용을 위하여, 서비스 개시 전에 사용자 요구를 비디오 별로 분류하고, 일정 시간 대기시켜, 동일한 비디오 사용자를 그룹화한다. 그림 6은 비디오 서비스 개시 초기에 선반입 스케쥴링을 나타낸 것이다. 현재 헤드가 파티션 1에서 C58을 검색하고 있을 때, 비디오 A의 서비스 개시가 요청되었고, 그 사용자는 이후 또 다른 요청 자가 있을 것을 대비하여, 서버의 사용자 스케쥴러에 의하여 일정시간 요청 큐에서 대기하게 된다. 종래의 스케쥴링에서는 그 대기 시간이 지난 후에 디스크를 스캔하여 비디오 A1을 읽어오지만, 제안한 선반입 캐싱에서는, 현재 서비스 중인 비디오 C의 다음 스트림 블록 C59를 읽어오기 위하여 헤드가 파티션 2를 스캔하는 동안에 미리 A1를 읽어올 수가 있으며, 나머지 A2, A3 등 도 마찬가지로 비디오C의 스트림 블록들과 함께 순차적으로 읽어올 수가 있다. 그리고 선반입 캐싱된 블록은 대기 시간이 지난 후 즉시 서비스를 개시 할 수가 있다. 선반입 버퍼 캐시의 크기는 현재 버퍼의 사용량에 따라 달라 질 수가 있지만, 선반입을 위한 최소한의 프레임인 pf_{MinCnt}는 필요한 프레임을 검색하여 지정된 위치로 저장하기까지의 시간인 $pf_{MinTime}$과 프레임 당 서비스 처리시간 fr_{pro}에 의하여 계산될 수가 있다. 여기서 $\overline{dsck_{ser}}$는 평균 디스크 검색 시간, $\overline{cpu_{pro}}$는 평균 cpu 처리시간, i/o_{cnt}는 서비스 요구 I/O의 수라 할 때 선반입 캐싱을 위한 최소한의 시간은 식(4)에 그 프레임 수는 식(5)와 같이 나타 낼 수 있다.

$$pf_{MinTime} = i/o_{cnt}(\overline{dsck_{ser}} + \overline{cpu_{pro}}) \quad (4)$$

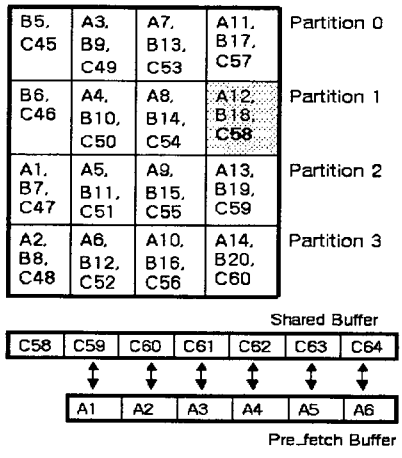
$$pf_{MinCnt} = (i/o_{cnt}(\overline{dsck_{ser}} + \overline{cpu_{pro}})) / fr_{pro} \quad (5)$$

또한 높은 QOS의 비디오 스트림을 서비스할 경우에는 서버의 공유 버퍼를 사용하더라도 수시로 변화하는 디스크나 서버에서 클라이언트에 이르는 네트워크의 상태에 따라 QOS에 영향을 줄 수가 있다. 이를 위하여 높은 QOS가

필요한 비디오 스트림을 요구하는 사용자를 그룹화하고 선반입한 스트림 블록을 플록시에 캐싱하여 이를 그룹화된 사용자가 공유하게 함으로써 네트워크의 부하를 분산시킬 수 있다. 플록시에 전달되는 최소한의 선반입 프레임 수 fpf_{MinCnt}는 필요한 프레임을 검색하여 프록시의 캐시로 저장하기까지의 시간인 $fpf_{MinTime}$과 프레임 당 서비스 처리시간 fr_{pro}에 의하여 계산될 수가 있다. 여기서 $\overline{nt_{dly}}$는 네트워크 평균 지연 시간이라 할 때 플록시 캐싱을 위한 최소한의 시간은 식(6)에 그 프레임 수는 식(7)과 같이 나타 낼 수 있다.

$$pf_{MinTime} = i/o_{cnt}(\overline{dsck_{ser}} + \overline{cpu_{pro}} + \overline{nt_{dly}}) \quad (6)$$

$$pf_{MinCnt} = (i/o_{cnt}(\overline{dsck_{ser}} + \overline{cpu_{pro}} + \overline{nt_{dly}})) / fr_{pro} \quad (7)$$



[그림6] 선반입 스케줄링
[Fig.6]pre_fetching scheduling

4. 시뮬레이션 및 분석

본 논문에서 제안한 디스크 파티션을 활용한 선반입 버퍼 관리 기법은 주요관심은 서버의 효율적인 자원의 활용면에서 버퍼 사용율과 서비스 처리시간의 단축에 이은 서비스 사용자를 증대 시키는데 있다. 시뮬레이션 자료 및 기본 파라메타의 내용은 표1과 표2에 각각 나타내었으며, 기타 케이스별 파라메타는 시뮬레이션 결과 분석에서 언급하였다. 시뮬레이션에 사용할 자료의 비디오 요구패턴은 Zif Distribution을 기본으로 하였고, 서비스 도착 율은 λ는 1sec당 한 개의 평균 도착 율을 갖는 Poisson Distribution으로 하였다. 초당 처리 프레임의 수 Fs는 30으로 하였다. 또한 Zif Distribution에 의한 인기도 분포는 10~15% 으로 그림 7은100개의 비디오에 대한 요구 패턴을 나타낸다. 그러나 비인기 비디오의 버퍼공유나 선반입 캐싱은 별 효과가 없으므로, 실제로 시뮬레이션 할 때에는 상위 10개의 인기 비디오를 중심으로 하였다. 본 논문에서 선반입 버퍼공유 정책의 성능을 분석하기 위하여, 기존 SCAN-EDF 방식의 인터벌 캐싱(IC)과 비교하여 보았다. 그림8은 이 두 정책간의 서비스 시간당 버퍼 사용률을 비교해본 것이다. 버퍼의 단위는 1Giga Byte이며, 한계 버퍼 량은 임의로 4Giga Byte로 정하였다. 이 값은 비디오 한 개의 크기를 4Giga Byte라 하였을 때, 시뮬레이션 총 비디오 크기의 10%정도 되는 값이다. 그러므로 총 비디오 량의 10%가 버퍼링될 수 있다. 그래프 G1은 본 논문의 버퍼 공유 캐싱 기법이며, G2는 인터벌 캐싱 정책이다. 그림8에서 초기에 G1의 버퍼 점유가 다소 증가하는 것은 선반입 스케줄링 때문이며, 서비스 해 갈수록 파티션 디스크와 연동 스케줄링을 함으로써 G2에 비하여 버퍼 점유가 개선되어짐을 알 수가 있다. 한편 G2는 서비스 시간 57분대에서 한계버퍼에 도달하는 것을 볼 수가 있다. 그림9에서 버퍼의 히트건수는 서비스를 진행해가는 동안에 버퍼의 공유도가 높아짐으로 점점 증가하는 양상으로 표시되고 있다. 평균 히트수는 G1이 159 G2가 133로 나타나며, G1이 기존 방

식보다 약 20% 정도 버퍼를 효과적으로 활용함을 알 수가 있다. 그림10과 표4에는 평균 서비스 처리시간을 나타내었다. 이 서비스 처리시간은 서비스 초기에 그룹화를 위한 사용자 대기시간은 포함시키지 않았다. 평균 서비스 처리시간은 G1이 24(ms), G2가 29(ms)이고 최대 서비스 시간은 G1이 292(ms) G2가 345(ms)이다. 최소 서비스 시간이 둘 다 0으로 나타난 것은 서비스 스트림 블록을 공유 버퍼에서 직접 읽어오는 경우이다.

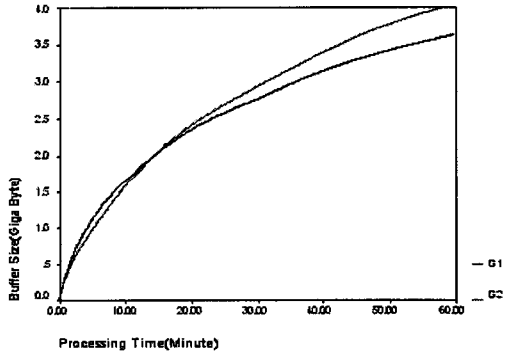
[표 1] 디스크 파라메타 테이블
[Table.1] Disc parameters table

파라메타	값
Sector size	512 Byte
Cylinders	1,962
Tracks per cylinder	19
Data sector per track	72
Number of zones	1
Track skew	8 sectors
Revolution speed	4,002 RPM
Controller reads	2.2 ms
Overhead writes	2.2 ms
sort (ms)	$3.24 + \frac{0.400}{\sqrt{c}}$
Seek time long (ms)	8.00 + 0.008c
boundary	c = 383
maximum transfer rate	10M BPS

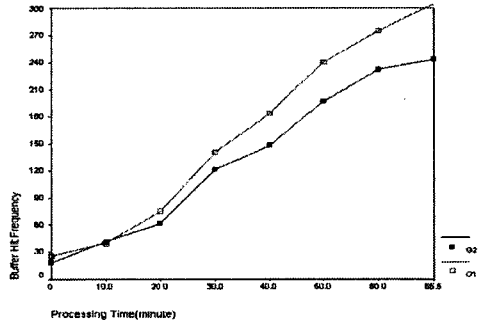
[표 2] 서비스 파라메타 테이블
[Table.2] Service parameter table

구분	내용	값
비디오 요구패턴	Zif Distribution	
λ(포아송 분포)	서비스 도착율	1/sec
V_no	비디오 수	10개
V_play_time	비디오 상영 시간	4000sec
Batch interval	배칭 간격	60sec
Service count	서비스 요구 수	2000

[그림7]비디오 요구 패턴
[Fig. 7] Request Video Pattern



[그림 8] 버퍼 이용도
[Fig. 8] Buffer Utilization



[그림 9] 버퍼 히트 빈도
[Fig. 9] Buffer Hit Frequency

[표 3] 버퍼 히트 빈도
[Table.3] Buffer Hit Frequency

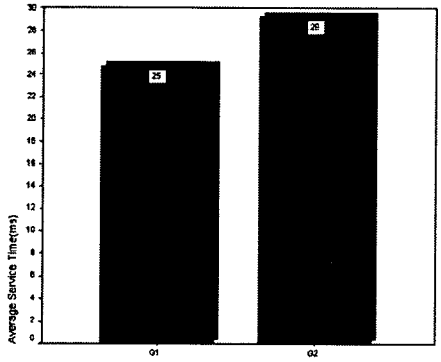
Statistics

		G2	G1
N	Valid	2000	2000
	Missing	0	0
Mean		133.1535	159.7430
Std. Deviation		79.2575	97.6788
Variance		6281.7488	9540.7493
Minimum		.00	.00
Maximum		292.00	345.00

[표 4] 평균 서비스 시간
[Table.4] Average Service Time

Statistics

		G1	G2
N	Valid	2000	2000
	Missing	0	0
Mean		24.7432	29.2582
Std. Deviation		14.2061	15.0765
Minimum		.00	.00
Maximum		48.98	54.98



[그림 10] 평균 서비스 시간
[Fig. 10] Average Service Time

5. 결론

Near-VOD에서 서버 자원의 한계를 극복하고 사용자 서비스의 수용능력을 증대시키기 위한 방식은 사용자의 일괄 수용 서비스(Batched admission service)와 버퍼 공유 정책이 있다. 배칭 방식은 미리 정의된 시간 간격 동안에 모아진 사용자의 동일한 비디오의 요구를 한번의 I/O 수행으로 처리하는 방식이며, 버퍼 공유는 한번 사용된 정보를 일정기간 메모리에 남겨두어 다음 사용자가 사용할 수 있게 하여 그만큼 디스크의 접근 횟수를 줄일 수가 있다. 특히 버퍼 공유는 최근 메모리 가격의 하락에 의하여 자주 제안되고 있는 방식이다. 이 방식은 파일과 데이터베이스의 효과적인 정보 처리를 위하여 응용되었으며, 인터넷 웹상의 페이지 객체를 서비스하기 위한 방식으로 서버나 플록시 등에 활용되고 있다. 그러나 이 정책들은 대용량 연속 비디오 스트림을 서비스하기에는 무리가 있다. 그리고 비디오 스트림을 위한 최근 몇몇의 버퍼 공유 정책도 버퍼의 한계에 효과적으로 대처하기에 어려운 점이 있다. 본 논문에서 제안한 선반입 버퍼 공유 정책은 파티션 디스크를 활용하여 현재 서비스와 공유 버퍼의 상태를 파악하여 효과적으로 스트림을 공유하여 서비스해 줌으로써, 버퍼의 점유와 실제 사용도(히트율) 면에서 약 20% 정도의 효과를 보여 주었다. 플록시의 성능이 높아지고 있는 지금, 이 방식은 비디오 서버 뿐 만아니라, 플록시의 캐싱 스케줄링에서도 활용할 수가 있을 것이다.

참고 문헌

[1] Yee-Hsiang Chang, David Cdggins, "An Open-Systems Approach to Video on Demand". IEEE Communications Magazine, pp 68-80, 1994

[2] Ohanian TA, "Digital nonlinear editing: new approaches to editing film and video. Focal Press, Boston, Mass, 1993

[3] Martin Arlitt, Rich Friedrich, and Tai Jin Hewlett-packard Laboratories, 1501 page Mill Road, 1998 [4] Leana Golubchik, John C.S. Lui, "Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers". Multimedia Systems, pp 140-15, 1996

[5] Asit Dan, Dinker Sitaram, "Dynamic batching policies for an on-demand video server". Multimedia Systems, pp 112-121, 1996

[6] Raymomd T. Ng, Jinhai Yang, "An analy sis for buffer sharing and pre-patching techn iques for multimedia systems", Multimedia Systems, pp 4: 55-69, 1996

[7] Sreenivas Gollapudi, Aidong Zhang, "Buff er model and management in distributed mul timedia systems", Multimedia Systems, pp 20 6-218, 1998

[8] Bruno, J., Brustoloni, J., Gabber, E,. zden, B., Silberschatz, A., "Disk Scheduling with Quality of Service Guarantees, " Proc. of the Int' IEEE Conference on Multimedia Computing
and Systems, June 1999.

[9] M. Kallahalla and P. J. Varman. Optimal Read-Once Parallel Disk Scheduling. In Proc. of Seventh Workshop on I/O in Parallel and Distributed Systems, To appear. 25 1999.

[10]A. L. N. Reddy, J. C. Wyllie, "I/O Issue in a Multimedia System," IEEE COMPUTE R, pp.69-74, March 1994.

[11]C. Ruemmer and J. Wilkes, "An Introduc tion to Disk Drive Modeling," IEEE Compute r, Vol.27. No. 3. pp.17-28, March 1994.

[12]A. L. N. Redduy and J. wyllie, "Disk Sch eduling in a Multimedia I/O System, " Proc. 1st ACM Int'I Conf. on Mutimedia, ACM Pr ess, New York, pp.225-233, 1993.

[13]M. -S. Chen, D.D. Kandlur, and P.S. Yu, "Optimization of the Group Sweeping Schedu ling (GSS) with Heterogeneous Multimedia S treams." Proc. 1st ACM Int'I Conf. on Multi media, ACM Press, New York, pp.235-241,1 993

[14]P. Bocheck, H. Meadows, S. Chang, "Dis k Partition Technique for Reducing Meltimed ia Access Delay", Preceedings of the IASTE D/ISMM International Conference, pp.27-30, 1994.

[15] s.u choi, s.k park, "Storing Technique of Multiple Streams on Disk with a Fixed Sin gle Zone". ITC-C,SCC, 1996.

[16] S.Acharya and B.Smith, Middleman: "A video caching proxy server" In Proc. of The 10th International Workshop on Network and Operating System Support for Digital Audio and Video IEEE NOSSDAV, 2000.

[17] Duc.A.Tran, Kien.A.Hua, and Simon Sheu, "A new caching architecture for efficient Video-on-Demand services on the internet", Proceedings of IEEE on Applications and the Internet, pp.172-181, January 2003

[18] J.Jannotti, D.Gifford, K.Johnson, .Kaasho ek, and Jr.J.O'Toole, "Overcast: Reliable mul ticasting with an overlay network", in Proc. of 4st Symposium on Operating Systems and Implementation(OSDI), pp=197-212, October 2000

[19]R. Rejaie, D. Estrin, and M. Handley, "Q uality Adaptation for Congestion Controlled Video Playback over the Internet," in To app ear in Proc. of ACM SIGCOMM '99, Cambri dge, Sept. 1999

[20] Reza Rejaie, Haobo Yu, Mark Handley, and Deborah Estrin. Multimedia proxy cachin g mechanism for quality adaptive streaming applications in the internet. Technical Report 99-XXX, USC-CS, 1999.

[21] M. Deshpande and G. Karypis: "Selectiv e Markov Models for Predicting Web-Page Accesses ", Proceedings SIAM Int. Conferenc e on Data Mining (SDM'2001), Apr. 2001.

1983년 광운 대학교 전자계산과(이학사)
1987년 경희대학교 대학원 전자공학과(공학 석사)
2001년 아주대학교 대학원 컴퓨터공학과(공학 박사)
1992년~현재 시립 인천 전문대학 컴퓨터 정보과 교수
관심 분야: 멀티미디어 및 VOD 시스템 응용
소프트웨어 시스템 디자인
이동 컴퓨팅 시스템 등