주문형 비디오 서버에서의 개선된 그룹핑과 버퍼 공유 기법

(Improved Grouping and Buffer Sharing Method in VOD Server)

정 홍 기 † 박 승 규 ††
(Hong-ki Jung)(Seung-kyu Park)

요 약 주문형 비디오 (Video On Demand) 서버의 구현은 현재까지도 많은 부분에 한계를 가지고 있으며, 이를 해결하기 위한 연구들이 활발히 진행되고 있다. 주문형 비디오 서비스의 한계는 대부분이 자원(디스크, 버퍼)의 제한 때문에 발생하고 있으며, 디스크어레이(Disk Array)나, 메모리 뱅크(Memory Bank)와 같은 특수한 하드웨어를 사용하여 해결하고 있다. 또한, 이와 아울러 그룹핑이나 버퍼 공유기법과 같은 소프트웨어적인 방법도 제안되었다. 본 논문에서는 주문형 비디오 서버의 한계 중 디스크대역폭과 버퍼 공간의 한계점에 의한 문제를 해결하기 위하여 개선된 그룹핑 방법과 버퍼 공유 및 관리 방법을 제안한다. 제안하는 그룹핑 방법은 비디오의 인기도에 기반을 둔 방법을 사용하며 버퍼 공유방법은 버퍼의 크기 및 디스크의 대역폭 가용량에 따라 적응적으로 병합, 분할하는 방법을 사용한다. 제안하는 병합, 분할 방법은 피기백킹(piggy-backing) 개념을 도입하였으며 시뮬레이션을 통해 제안한 방법의 우수성을 보였다.

Abstract A number of research activities have been recently made in the area of VOD servers. Due to the large volume of multimedia data in nature, and increasing large number of concurrent requests for services, I/O and network bandwidth are still limited and expensive resources. This paper proposes improved grouping and buffer sharing mechanism to maximize the performance of server's resources. The grouping method determine the time interval for grouping users requests based on popularity of videos. The buffer sharing method has two operations, piggy-merging and piggy-splitting based on piggybacking concept. The simulation result shows that the proposed method performs good performance.

1. 서 론

최근, 네트웍과 저장장치, I/O 장치의 발달로 네트웍을 통해 개인이 가정에서 또는 개인 단말기로 멀티미디어 데이타를 사용할 수 있게 되었다. 일반적으로 멀티미디어 데이타는 부피가 크고, 실시간으로 처리되어야 하며, 서로 다른 미디어간의 동기화와 한 미디어 안에서의 동기화를 필요로 하는 등의 특징을 가지고 있다. 이와 같은 특징은 멀티미디어 데이타를 네트웍을 통해 다수의 사용자에게 제공하는 경우, 최적화 해야할 많은 문제점을 발생시킨다. 특히, 주문형 비디오(Video On Demand) 서버를 구현하는데 있어서, 현재까지도 많은 부분에서 한계에 부딪치고 있으며, 이를 해결하기 위한 연구들이 활발히 진행되고 있다. 주문형 비디오 서버를 구현하기 위해, 하드웨어적으로 디스크어레이(Disk Array)가 폭넓게 사용되고 있으며, 디스크 상에 파일 할당과 파일 복제 방법이 중요한 논점으로 대두되었다. [1,2]

이와 병행하여 또 다른 측면에서는 사용자 그룹핑 방법과 버퍼 공유 방법에 관한 연구가 진행 되어왔다. 그룹핑은 사용자의 비디오에 대한 요구를 일정 기간동안

· 이 논문은 정보통신연구 진흥원에서 출연한 '99 대학기초 사업의 지원을 받아 연구되었습니다.
† 학생회원 : 아주대학교 정보통신전문대학원
keeper@madang.ajou.ac.kr
†† 종신회원 : 아주대학교 정보통신전문대학원 교수
sparky@madang.ajou.ac.kr
논문접수 : 1999년 11월 12일
심사완료 : 2000년 10월 21일

모아서 마치 하나의 사용자처럼 처리하는 방법을 말한다.[3,4] 이러한 방법으로 처리해 주는 서버를 Near-VOD 서버라고 한다. 그룹핑 기법에서 중요한 요소는 사용자가 기다려야 하는 시간, 즉 그룹핑을 하기 위한 시간을 결정하는 것이다. 가장 단순한 방법으로는 고정된 시간을 가지고 그룹핑을 하는 방법이 있으며 좀 더 진보된 방법으로는 비디오의 인기도를 기반으로 한 방법을 사용하여 사용자가 기다려야 하는 시간을 결정하거나, 사용자가 요구를 취소할 확률을 사용하여 시간을 결정하는 방법이 있다.[3]

버퍼 공유 기법이라는 것은 어떤 사용자가 요구한 비디오 스트림이 다른 사용자의 요구에 의하여 미리 디스크로부터 읽어 와서 버퍼 상에 존재할 때, 디스크로부터 스트림을 읽어오지 않고 메모리내의 스트림을 공유하는 방법을 말한다.[4,5] 일반적으로 주문형 비디오 서버에서 병목이 발생하는 부분은 디스크 대역폭이므로, 만약 버퍼 공간이 충분하다면 버퍼 공유기법을 사용하는 것이 명백하게 좋은 효과를 보인다. 버퍼 공유 기법에서 논점은 버퍼 공유에 필요한 메모리의 크기를 결정하는 방법과 버퍼를 공유한 그룹간의 병합과 분할 방법이다. 기존의 병합과 분할 방법의 문제점은 병합을 하는 경우 추가의 버퍼공간이 필요하고, 분할을 하는 경우 추가의 디스크 대역폭이 요구된다는 점이다. 따라서 세션의 병합과 분할 시에 여분의 자원이 없다면 더 이상 병합과 분할을 진행하지 못하게 된다. [6]

본 논문에서는 개선된 사용자 그룹핑 및 비디오 스트림의 버퍼 공유 방법을 제안한다. 제안하는 그룹핑 방법은 사용자가 기다려야 하는 시간을 비디오의 인기도에 기반 하여 결정할 것이다. 제안하는 버퍼 공유 방법에서는 세션의 크기를 비디오의 인기도에 따라 유동적으로 변할 수 있도록 할 것이며, 자원의 가용량 변화가 불가능한 경우에는 피기백킹(piggy-backing) 개념을 도입한 병합과 분할방법을 적용함으로써 기존 문제점의 개선을 시도할 것이다.

2. 주문형 비디오 서버의 기본 구조

본 논문에서 제안하는 기본적인 주문형 비디오 서버의 구조는 <그림 1>과 같다. 그림에서 보듯이, 저장 장치는 보조 저장 장치와 디스크어레이로 구성되어 있다. 보조 저장장치는 데이타를 읽어오는 시간은 느리지만 가격이 저렴하다는 장점이 있으며, 디스크어레이는 데이타를 읽어오는 시간은 빠르지만 가격이 비싸다는 단점이 있다. 따라서, 인기도가 높은 비디오는 디스크어레이에, 인기도가 낮은 비디오는 보조 저장 장치에 보관한다. 기억장치로부터 읽어온 스트림은 버퍼 공간으로 이동되고, 제안하는 버퍼 공유방법을 통해 관리가 되며, 서비스 시간에 스트림을 전송할 경우에는 ATM 네트워크 망을 통하여 동시에 해당 그룹에게 스트림을 전송한다.

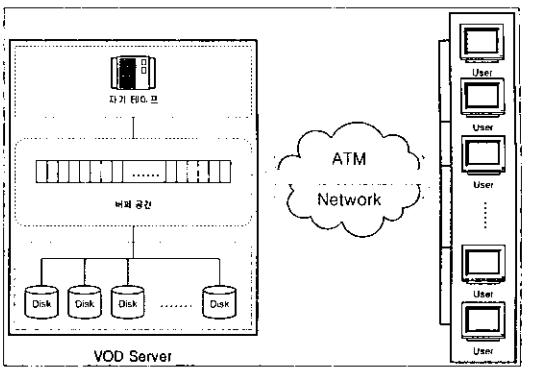

그림 1 주문형 비디오 서버의 기본 구조

<그림 2>는 사용자가 요구한 비디오 스트림의 가상적인 흐름을 나타내고 있다. 본 논문에서는 가상적인 계층을 크게 그룹계층, 세션계층, 그리고 디스크 계층으로 나눈다. 디스크 계층은 물리적인 디스크가 존재하는 곳이며, 각각의 영화는 디스크에 분할(stripe)된 방식으로 저장되어 있다고 가정한다. 세션계층은 물리적으로는 버퍼공간을 의미하며 가상적인 의미에서는 버퍼를 공유하는 그룹을 의미한다. 따라서 하나의 세션은 하나의 디스크 대역폭을 사용하게 되고, 같은 종류의 비디오 데이타를 가지고 있다. 마지막으로 그룹 계층은 그룹이 존재하는 계층으로 각각의 그룹들은 일정한 시간 간격동안 같은 비디오 스트림에 대한 요구가 들어 왔을 때, 마치 하나의 사용자와 같이 합쳐지고, 각각의 그룹 안의 사용자들은 동시에 서비스가 된다. 또한 요구된 스트림은 그룹 안의 사용자들에게 ATM망을 통해 멀티 캐스팅 된다.

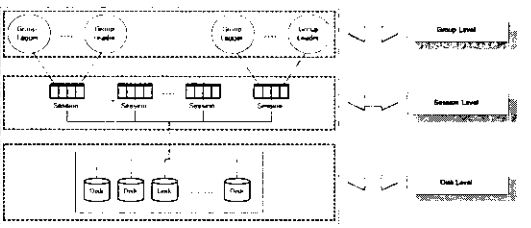

그림 2 스트림의 흐름에 대한 계층구조

3. 그룹핑 기법

그룹핑의 다른 이름은 '배칭'(batching) 이라는 이름

으로 널리 통용된다.[5] 여러 사용자의 요구를 그룹화 시키기 위해서는 배칭윈도우[1]의 크기를 결정하는 것이 중요하다. 배칭 윈도우의 크기에 따라서 그룹핑이 되는 사용자의 수와 사용자가 비디오를 요구한 시점부터 기다리는 시간에 비례한 취소 확률(reneging probability) 사이에 트레이드 오프(trade off)가 존재하기 때문이다. 본 논문에서는 효율적인 배칭윈도우의 크기를 결정하는 방법을 제안한다.

3.1 배칭 윈도우의 크기 결정 방법

배칭윈도우의 크기는 두 가지 측면에서 고려할 수 있다. 첫 번째는 단위적인 측면으로써, 요구들의 개수를 단위로 배칭윈도우의 크기를 결정하는 방법과 시간을 단위로 배칭윈도우의 크기를 결정하는 방법이 있다. 일반적으로 후자가 더 좋은 효율을 보이고 있다.[7] 두 번째는 유동성의 측면으로써, 고정된 크기를 사용하는 방법과 가변적인 크기를 사용하는 방법이 있다. 본 논문에서는 시간을 단위로 하는 방법과 가변적인 크기를 사용하는 방법을 적용하여 배칭윈도우의 크기를 결정한다.

시간을 단위로 크기를 결정하기 위해, 본 논문에서는 비디오의 인기도에 따른 배칭윈도우 크기 결정방법을 제안한다. 비디오의 인기도는 Zipf 분포를 따르고 있으며, 분포 함수는 다음과 같다.

$$f(i) = c/i^{(1-\theta)} \text{ where } C = 1/\sum_{i=1}^{I} \frac{1}{i^{(1-\theta)}} \quad (1)$$

수식 (1)은 일반적인 Zipf 분포함수를 수정한 수식으로, θ=0.271로 적용하여 인기도에 따른 접근 확률의 편중도를 높이도록 만들었다.[3]

일반적으로 비디오의 인기도가 높을수록 사용자가 기다려야 하는 시간이 짧아지는 것이 좋은 효율을 나타내므로, 본 논문에서는 배칭윈도우를 각각의 비디오에 할당하고, 크기는 해당하는 비디오의 인기도에 따라 결정하는 방법을 사용한다. 또한, 인기도가 매우 낮은 비디오는 그룹핑의 의미가 없게 되므로 어느 정도의 인기도를 가진 비디오까지 그룹핑을 할 것인지를 결정하는 것 또한 중요하다. 본 논문에서는 그룹핑 되는 비디오를 결정하는 방법으로 상위 10%의 인기도를 가진 비디오까지를 그룹핑을 하는 방법을 사용한다.

$$t_{wait_i} = t_{wait_{max}} \times \frac{\sum_j f(j) - f(i)}{\sum_j f(j)} \text{ where}$$

$$j \text{ 그룹핑 되는 모든 비디오} \quad (2)$$

수식 (2)는 i-번째 비디오의 배칭윈도우 크기를 결정하는 방법을 나타낸다. $t_{wait_{max}}$는 통계적인 방법으로 일반 사용자가 최대한 기다릴 수 있는 시간으로 미리 결정되어 있다고 가정한다. j는 그룹핑 되는 비디오이며, 이는 상위 10% 인기도를 가진 비디오로 결정하였다. 수식 (2)에서는 각 비디오의 인기도에 근거하여 전체 그룹핑 되는 비디오 중에서 i-번째 비디오의 인기도가 차지하는 비율에 맞춰서 기다리는 시간을 결정하였다. 따라서 인기도가 높을수록 기다리는 시간이 짧아 지며, 반대로 인기도가 높을수록 기다리는 시간이 길어진다. 그러나 최대 기다리는 시간은 $t_{wait_{max}}$값을 초과하지 못한다.

3.2 그룹핑 기법

배칭윈도우의 크기가 정해지면 그룹핑을 하는 방법은 간단하게 해결된다. <그림 3>에서 볼 수 있듯이, i-번째 비디오에 대한 요구가 들어온 시점부터 t_{wait_i} 시간 내에 들어오는 i-번째 비디오에 대한 요구들은 하나의 그룹으로 만들어지고, t_{wait_i} 시간이 지나면, 하나의 그룹으로서 서비스를 받게된다.

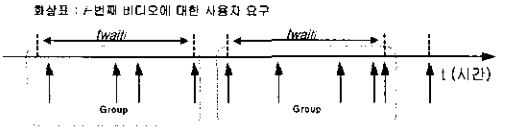

그림 3 i-번째 비디오에 대한 사용자 요구의 그룹화

4. 버퍼 공유 기법

버퍼 공유 기법은 같은 비디오에 대한 요구들(그룹들) 중, 시간 차이가 작은 요구들에 대하여 버퍼를 공유하는 것을 의미한다. 예를 들어 두 개의 그룹 G_1, G_2가 t_{gap}의 시간 차이로 i-번째 비디오의 k번째 스트림을 요구한다고 가정했을 때, t_{gap}의 값이 작은 차이라면 G_1이 디스크로부터 읽어온 스트림을 G_2가 사용하게 된다. 따라서 G_1입장에서는 스트림을 공유하는 그룹을 위해 자신이 읽어온 스트림을 메모리에 남아있도록 고정(pin)시킨다. 고정된 스트림은 G_2가 읽은 시점에서 삭제된다. 따라서 버퍼공유 기법은 디스크의 대역폭 사용을 줄일 수 있으나, 추가 버퍼 공간이 필요하다는 단점을 가진다. 본 논문에서는 이와 같은 단점을 완화시킬 수 있는 버퍼 공유기법을 제안한다.

4.1 세션의 크기와 세션에 포함될 조건

세션의 크기를 정하는 방법에는 세션의 크기를 고정시키는 방법과, 세션의 크기가 가변적으로 변하는 방법이 있다. 전자는, 서버를 구현하고, 버퍼를 관리를 쉽게 할 수 있으나, 시간에 따라 변화하는 사용자의 수에 적

1) 사용자의 요구를 그룹화 하는 조건을 의미한다.

응적으로 대처할 수 없으며, 버퍼공간의 낭비도 초래 할 것이다. 따라서 본 논문에서는 후자의 방법을 사용하여 버퍼공간의 낭비를 줄이고 네트웍의 변화에 적응적으로 대처할 수 있는 버퍼 공유기법을 제안한다. 본 논문의 3장에서, 그룹핑을 할 때 i-번째 비디오에 대하여 배칭 윈도우의 크기를 t_{wait_i}라고 했다. 또한, t_{wait_i}의 값은 비디오의 인기도에 의존하여 값이 정해지게 되며, 비디오의 인기도가 높을수록 사용자가 기다려야 하는 시간이 작아지는 것이 효율적이라고 언급했다. 버퍼 공유에서도 유사한 상황이 발생한다. 인기도가 높은 비디오에 대한 요구는 실제적으로 서버 내에서 많은 자원을 사용하게 되므로, 버퍼 공유를 통해서 자원 사용을 최소화시킨다면 서버에서의 과부하를 줄일 수 있게 된다. 사용자의 요구가 단위시간당 λ개로 포아송 분포로 들어온다고 가정을 하였을 때, 수식 (3)은 단위시간당 i-번째 비디오를 요구하는 개수를 나타낸다. 수식에서 $f(i)$는 3장에서 구한 Zipf 분포 함수이다.

$$\lambda_i = \lambda f(i) \tag{3}$$

<그림 4>는 세션에 그룹이 들어오는 간격을 표현하고 있다. <그림 4>에서 $1/\lambda_i$는 i-번째 비디오 요구들 사이의 평균 시간간격을 의미하므로, i-번째 비디오를 요구한 그룹이 세션에 들어오는 평균 시간간격은 수식 (4)와 같이 구해진다.

$$t_{inter-arrival_i} = t_{wait_i} + \frac{1}{\lambda_i} \tag{4}$$

수식 (4)에서도 알 수 있듯이, 그룹들이 세션에 들어오는 평균 시간 간격은 λ_i와 t_{wait_i}값에 의하여 결정이 되고, 이 두 개의 값은 Zipf 분포 함수인 $f(i)$와 밀접한 관계를 가진다.

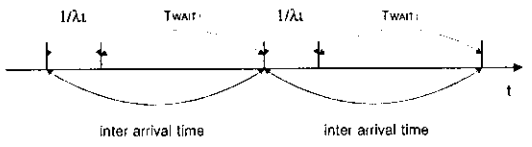

그림 4 Session에 들어오는 그룹의 inter-arrival time

수식 (5)는 새로운 그룹이 기존의 세션에 포함될 수 있는 시간 차이를 나타낸다. C값은 버퍼의 용량과 관련된 상수이며, $N_{session}$은 그룹핑을 하는 비디오의 수를 의미한다. 만약 현재 서비스가 진행중인 세션과 새로운 그룹간의 시간 차이가 t_{join}내에 있다면 새로운 그룹은 세션에 포함된다.

$$t_{join} = \frac{C}{N_{session}} \times \sum_{i=1}^{N_{session}} t_{inter-arrival_i} \tag{5}$$

수식 (4)와 수식 (5)에서 알 수 있듯이, 비디오의 인기도가 높을수록 세션에 포함될 확률이 높아지며, 세션의 크기도 상대적으로 커진다. 따라서 세션의 크기를 결정하는 방법도 비디오의 인기도에 기반을 하였다. 수식 (6)은 새로운 그룹이 기존의 세션에 포함되어 질 수 있는 프레임의 수를 t_{join}을 사용하여 계산한다. 여기서 R_{frame}은 초당 보여지는 프레임의 수를 나타내며 본 논문에서는 30 frame/s로 가정한다.

$$N_{SessionJoin} = \lceil t_{join} \times R_{frame} \rceil \tag{6}$$

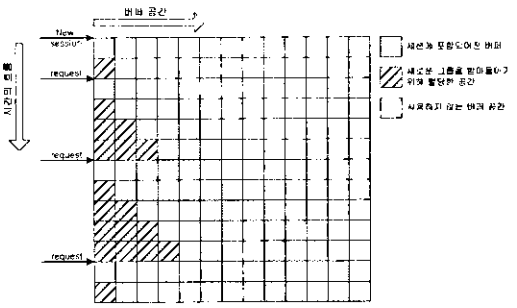

그림 5 그룹이 세션에 포함되는 과정

<그림 5>는 그룹이 세션으로 포함되는 과정을 보여준다. 여기서는 세션에 포함될 수 있는 프레임 수가 4일 때의 과정이다. 마지막 요구가 들어 왔을 때, 이미 프레임 차이가 4이상으로 되었기 때문에 기존의 세션은 다음 그룹과의 버퍼 공유를 위해 할당한 프레임을 해제시키고, 하나의 세션으로 독립한다. 따라서 다음에 들어온 그룹은 새로운 세션을 만들게 된다. 그러나, 이러한 방법으로 버퍼공간을 할당한다면, 인기 있는 비디오가 (그룹핑을 하는 비디오) 모든 버퍼 공간을 사용하는 경우가 발생한다. 따라서 인기 없는 비디오에 대한 요구는 버퍼공간의 부족으로 수용제어에서 거절 될 확률이 높아진다. 이것은 형평성에 위반이 되므로, 본 논문에서는 이 문제점을 보안하기 위해서 버퍼공간을 크게 두 개의 영역으로 분할하였다. 수식 (7)과 수식 (8)은 각각 버퍼 공유를 하지 않는 비디오와 버퍼 공유를 하는 비디오를 위해서 할당되는 버퍼의 크기를 계산하는 식이다.

$$B_{nonsession} = C_1 \times \frac{D_{nonsession}}{R_{frame}} \tag{7}$$

$$B_{session} = B - B_{nonsession} \tag{8}$$

수식 (8)에서 $D_{nonsession}$은 그룹핑을 하지 않는 사용자를 위해 할당된 디스크 대역폭을, B는 서버의 전체 버퍼

의 크기를 의미하며, C_1 은 서버의 버퍼의 크기에 관련하여 변하는 상수이다.

4.2 세션의 병합과 분할

4.2.1 세션의 병합

세션의 병합이란 두 개의 서로 다른 세션이 같은 비디오 스트림을 서비스하고 있고, 세션간의 서비스하는 프래임의 차이가 작을 때, 두 세션을 하나의 세션으로 만드는 것을 의미한다. 세션의 병합은 현재 디스크 대역폭 사용률이 높을 때 일어난다. 하나의 세션은 하나의 디스크 대역폭을 할당받고 있으므로 두 개의 세션을 합치게 되면 사용 가능한 디스크 대역폭이 늘어나기 때문이다. 그러나 기존의 세션병합의 문제점은 추가적으로 버퍼공간을 요구한다는 점이다. 따라서 만약 요구하는 버퍼공간이 남아있지 않다면, 세션병합은 진행될 수 없다. 본 논문에서는 이러한 문제점을 완화시킬 수 있는 방법을 제안한다. 제안하는 세션 병합 방법은 피기백킹의 개념을 이용하였다. 그러나 피기백킹 방법의 문제점은 QoS(Quality of Service)를 떨어뜨린다는 단점이 있다. 그러므로 본 논문에서는 현재 자원(버퍼)의 사용률이 버퍼 공간의 한계인 $B_{boundary_m}$보다 작은 경우에는 기존의 세션 병합방법을 사용하고, 큰 경우에는 제안하는 세션 병합방법을 사용한다.

① 현재 버퍼 사용률이 $B_{boundary_m}$ 보다 작은 경우

이와 같은 경우에는 버퍼 공간의 여유가 있기 때문에, 기존의 세션 병합 방법을 사용한다. 수식 (9)는 i-번째 비디오 대한 j-번째 세션과 k-번째 세션간의 병합을 위한 프래임의 수를 의미하며, 수식 (10)은 병합을 위한 버퍼 공간을 의미한다. 수식 (10)에서의 S_{frame}은 프래임의 크기를 의미한다.

$$N_{SessionMerge_{i,j,k}} = R_{SessionLagger_{i,j}} - R_{SessionLeader_{i,k}} - 1 \quad (9)$$

$$B_{merge_{i,j,k}} = N_{SessionMerge_{i,j,k}} \times S_{frame} \quad (10)$$

예를 들어 두 개의 세션이 존재하고 각각의 세션은 57~66, 46~52의 프래임을 읽고 있다고 가정하자. 수식 (9)에 의하여 두 세션의 병합을 위해서는 4개의 프래임이 필요하다. 따라서 두 개의 세션을 병합하기 위해 필요한 시간은 수식 (11)과 같이 계산된다.

$$t_{merge} = \frac{N_{SessionMerge_{i,j,k}}}{R_{frame}} \quad (11)$$

② 버퍼 사용률이 $B_{boundary_m}$ 보다 큰 경우

만약 두 개의 세션을 병합하려고 하는 시점에서 충분한 버퍼공간의 여유가 없다면 기존의 방법으로 병합하는 것은 불가능하다. 따라서 본 논문에서는 이 문제를 해결하기 위해서 현재 버퍼 사용률의 수준이 $B_{boundary_m}$ 보다 크게 되면 피기백킹 개념을 도입한 병합방법을 사용하는 것을 제안한다. 피기백킹 방법이란 동일한 비디오의 서로 다른 위치를 보고 있는 두 사용자의 재생률을 변화시켜 결국에는 같은 위치를 서비스 받도록 하는 방법이다.

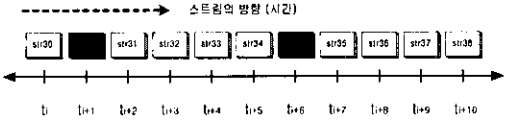

그림 6 스트림 사이에 쓰레기 스트림을 넣었을 경우

<그림 6>는 프래임을 삽입하여 비디오의 재생률이 사용자가 지각할 수 없는 범위 내로 느리게 하는 예를 보여준다. (실제로 서버에서 제공되는 전송률은 변화가 없다.) 만약 비디오를 사용자에게 초당 30 프래임의 비율로 보여주고 있고, 다섯 개의 프래임마다 한 개의 쓰레기 프래임을 삽입한다면 사용자가 실제 보는 재생효과는 초당 프래임 수가 30×5/6로 작아진다. 즉 1/6의 느린 동작 효과를 나타낸다. <그림 7>과 <그림 8>은 피기백킹 방법을 세션에 적용했을 때의 모습을 나타낸다. <그림 7>에서와 같이 앞서가는 세션이 재생률을 느리게 함으로써 두 개의 세션을 병합하게 된다면, d_{m-new}만큼의 시간이 소요되며, <그림 8>에서와 같이 뒤에서 오는 세션이 재생률을 빠르게 함으로써 두 개의 세션을 병합하게 된다면 d_m만큼의 시간이 소요된다. 따라서 본 논문에서는 <그림 8>에서 보여지는 방법으로 두 세션간의 병합작업을 한다. 재생률을 빠르게 하기 위해서 프래임을 삭제하는 방법을 사용한다.

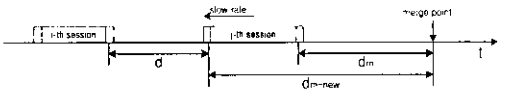

그림 7 세션간의 피기백킹 방법을 사용한 병합 - a

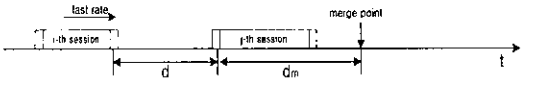

그림 8 세션간의 피기백킹 방법을 사용한 병합 - b

두 개의 세션을 피기백킹 방법을 사용하여 병합하는데 소요되는 시간은 수식 (12)와 같이 계산된다. 수식에서 $N_{SessionMerge_{i,j,k}}$ 는 k 번째 비디오에 대하여 i 번째

세션과 j 번째 세션의 프레임 차이를 나타내고, $R_{frame-skip}$ 은 빠른 재생 효과를 주기 위해서 건너뛰는 프레임 비율을 나타낸다.

$$t_{piggy-merge} = \frac{N_{SessionMerge_{i,j}}}{R_{frame-skip}} \quad (12)$$

본 논문에서 제안하는 방법을 사용하여 세션 병합을 할 경우, $t_{piggy-merge}$ 만큼의 시간이 경과한 후에 병합이 이루어진다. 기존의 병합방법 보다는 시간이 길어지게 되지만, 추가적으로 버퍼를 요구하지 않는다는 장점이 있다. 반면에 프레임을 삭제하기 때문에 QoS의 하락이 발생지만, 사람의 인지 불가 범위인 5-8% 사이에서의 범위에서 이루어지고, 지속적인 병합이 아니라 일시적인 병합이므로 기존의 피기백킹 방법에 비하여 심각하게 고려할 문제점은 아니다.

5.2.2 세션의 분할

세션의 분할이란, 동일한 비디오에 대한 요구 중에서 버퍼를 공유하는 그룹을 두 개의 그룹으로 나누는 것을 의미한다. 세션의 분할은 현재 버퍼공간이 부족할 경우 발생한다. 기존의 세션분할 방법은 매우 간단하게 세션을 분할시켜 버퍼공간을 확보할 수는 있지만, 추가적으로 디스크 대역폭을 요구한다는 단점이 있다. 따라서 만약 요구한 디스크 대역폭이 남아 있지 않다면, 세션 분할은 진행 될 수 없다. 본 논문에서는 세션 병합과 마찬가지로, 피기백킹 개념을 이용한 분할 방법을 제안한다. 제안하는 세션 분할 방법도 어느 정도의 QoS 하락이 발생하기 때문에, 만약 현재 사용중인 디스크 대역폭이 디스크 대역폭의 한계인 $D_{boundary}$ 보다 작다면 기존의 분할 방법을 사용하고, 크다면 제안하는 방법을 사용하여 QoS의 하락을 최대한 방지하는 방법을 사용한다.

① 디스크 사용률이 $D_{boundary}$ 보다 작을 경우.

디스크 대역폭의 여유가 있기 때문에 기존의 방식을 사용한다. 따라서 선택된 세션 안에 있는 그룹 중 가장 큰 프레임 차이를 가지고 있는 두 개의 그룹을 찾아내 세션 분할을 적용한다. <그림 9>는 일반적인 세션 분할 방법을 나타낸다. 그림과 같이 세션을 분할하게 되면 사용 가능한 버퍼공간이 $N_{\max-gap} \times S_{frame}$ 만큼 증가하는 반면에, 추가적인 디스크 대역폭을 사용한다는 단점이 있다.

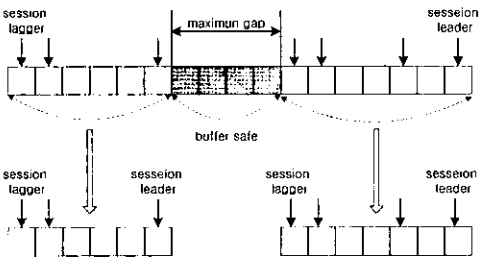

그림 9 일반적인 세션의 분할 방법

② 디스크 사용률이 $D_{boundary}$ 보다 클 경우

기존의 분할 방법은 쉽고 빠르게 세션을 분할 하지만, 추가적으로 디스크 대역폭을 요구하기 때문에 만약 디스크 대역폭이 충분히 여유가 없다면 연쇄적인 세션 병합 방법을 초래하거나 분할 작업을 진행하지 못한다. 본 논문에서는 추가의 대역폭을 요구하지 않고 세션 분할의 효과를 낼 수 있는 방법을 제안한다. 제안하는 방법은 선택된 세션 안에 가장 큰 프레임 차이를 가지는 두 개의 그룹을 찾아 뒤에서 오는 그룹의 재생률을 빠르게 함으로써 앞서가는 그룹과 합하는 방법을 사용한다. 두 개의 그룹이 병합이 되면, 기존의 세션 분할과 같은 결과가 발생한다. 그러나 제안하는 방법은 다소 복잡하며 세션 분할을 위해서는 일정한 시간이 필요하다는 단점이 있다. 수식 (13)은 세션 분할을 하는데 필요한 시간이다.

$$t_{piggy-split} = \frac{N_{\max-gap}}{R_{frame-skip}} \quad (13)$$

<그림 10>은 총 세션의 크기가 9이고, $N_{\max-gap}$이 4인 세션에서의 분할 작업을 시간의 흐름에 따라 보여준다. 여기서 N_p는 프레임을 삭제하는 사이클을 의미한다. 세션을 분할하기 위해 선택된 그룹 중에서 뒤에서 오는 그룹이 읽는 프레임을 살펴보면 {3,5,6,7,9,10,11,13,14,15,17,18,19,21}의 순서로 읽어오는 것을 볼 수 있다. 마지막 시간대에서 두 개의 그룹은 같은 21번째 프레임을

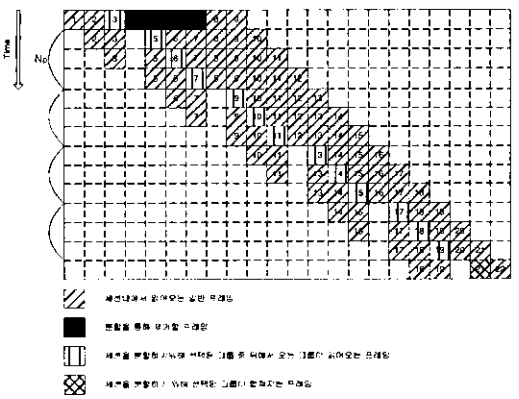

그림 10 피기백킹 방법을 사용한 세션 분할의 시간적 단계

읽게 되고 이 시점에서 그룹사이의 병합작업은 끝나게 된다. 두 개의 그룹간의 병합은 기존의 세션 분할 방법과 유사한 결과를 초래한다. 제안하는 세션 분할 방법은 세션 분할을 위해 시간이 필요하다는 단점은 있으나, 추가적으로 디스크 대역폭을 요구하지 않고 버퍼 공간의 사용을 줄일 수 있다는 장점이 있다.

5. 시뮬레이션

5.1 시뮬레이션 요소

시뮬레이션의 요소로서 사용자의 요구가 들어오는 시간 간격은 포아송 분포를 사용했으며, 포아송 분포의 λ 값은 0.1/s 로 설정하였다. 비디오를 선택하는 확률 분포는 Zipf 분포 ($\theta=0.271$)를 사용하였다. 시뮬레이션 언어로는 C++를 사용했으며 사용자의 요구는 2000번으로 설정하였다. 그 밖의 요소들은 <표 1>에서 보여진다. 그리고 버퍼나 디스크 사용률이 85% 이상이 되었을 때 제안한 알고리즘을 사용하였고, 자원 사용률이 75% 이하가 될 때까지 적용하였다. 두 가지 사용률의 한계는 임의로 결정하였으며 향후과제로 가장 효과적인 자원사용률의 한계를 결정할 계획이다.

표 1 시뮬레이션 요소

요 소	설 명	값
λ	Arrival Rate	0.1/sec
N_{video}	비디오의 개수	100개
L_{video}	비디오의 길이	60분 (1시간)
R_{frame}	초당 프레임의 수	30 frame/s
S_{frame}	프레임의 크기	0.0083MB
$N_{frame-skip}$	퍼기백킹 시 당 삭제될 프레임 수	5 개/s
B	버퍼의 크기	1000M (1GB)
B_{disk}	디스크의 대역폭	40MB/s
$D_{session}$	세션을 지원하기 위한 디스크 대역폭	20MB/s
$D_{nonsession}$	그룹으로 만들어지지 않는 요구를 위한 대역폭	20MB/s
N_{user}	사용자의 수	2000명
$t_{wait_{max}}$	사용자가 최대로 기다릴 수 있는 시간	100초

5.2 결과 및 분석

본 시뮬레이션은 버퍼공간의 사용률과 디스크 대역폭의 사용률에 초점을 맞추었다. 그 이유는 버퍼 공간의 사용률이나 디스크 대역폭의 사용률이 높다는 의미는 새로운 요구에 대한 수용이 어렵기 때문이다. <표 2>는 그룹화를 하기 위해서 사용자가 기다려야 하는 시간을 나타낸다. 최대 기다리는 시간은 각각의 비디오에 대하여 본 논문에서 제안한 수식 (2)를 사용하여 계산된 값이고, 실제 기다린 시간은 시뮬레이션을 통해 구해진 사용자가 실제로 기다린 평균시간을 의미한다. 순위 3부터 10까지, 그룹핑을 위해서 기다리는 시간의 차이가 적은 이유는 Zipf 분포에 의해서 구해진 인기도가 비슷하기 때문이며 따라서 실제로 기다린 시간도 차이가 작다. Near VOD의 특성상 사용자가 요구한 후 일정시간 동안 기다리지만, 본 논문에서는 기다리는 시간을 휴리스틱한 방법으로 정해진 최대 기다리는 시간에 근거하여 계산한다. 또한 실제 기다린 평균시간이 9번째 인기도를 가지고 있는 비디오가 8번째 인기도를 가지고 있는 비디오 보다 적게 나타나는 이유는 비슷한 인기도를 가졌기 때문에 입력 패턴에 따라 영향을 받았기 때문이다.

표 2 사용자가 기다린 시간

순위	최대 기다리는 시간	실제 기다린 시간	순위	최대 기다리는 시간	실제 기다린 시간
1	74	55.339623	6	92	82.574468
2	84	68.307692	7	93	82.622222
3	88	81.585714	8	94	86.200000
4	90	83.129870	9	94	82.604167
5	91	80.868852	10	95	84.214286

<그림 11>과 <그림 12>는 그룹화를 하지 않는 요구에 대한 자원 사용률을 보여준다. 그림에서 버퍼나 디스크의 사용률은 비디오의 길이(6000초)에 의존적으로 변화하는 것을 볼 수 있다. 따라서 사용자의 요구를 받아들일 수 있는 구간은 현재 서비스를 하고 있는 비디오가 끝나는 시점에서만 가능하다는 것을 알 수 있다.

<그림 13>과 <그림 14>는 일반적인 병합과 분할 방법을 사용하는 경우의 자원 사용률을 보여준다. 그룹핑

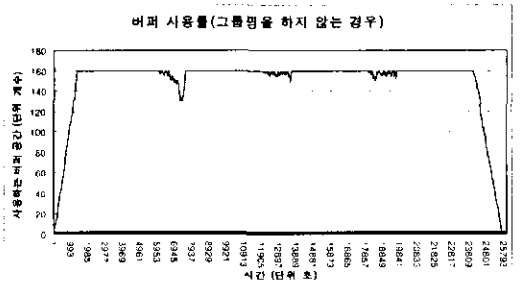

그림 11 버퍼 사용률(그룹핑을 하지 않는 경우)

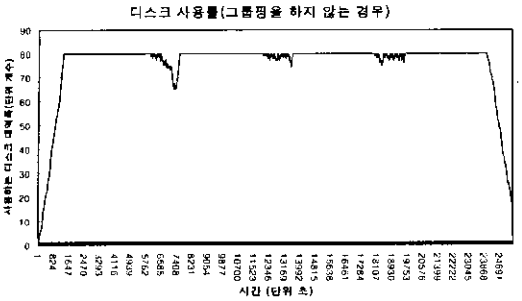

그림 12 디스크사용률(그룹핑을 하지 않는 경우)

을 하지 않는 방법에 비해서는 자원 사용률의 변화가 전체 구간에서 생기는 것을 볼 수 있다. 그러나 이 방법은 어느 정도의 디스크 대역폭이나 버퍼 공간이 남아 있을 경우에만 변화가 심하게 생기며, 그렇지 않은 경우에는 세션 병합과 분할을 진행 할 수 없게 되므로 변화가 점점 없어지다가 기존에 서비스를 받았던 비디오가 끝나는 시점에서만 새로운 요구를 수용할 수 있게 된다.

<그림 15>와 <그림 16>은 제안한 버퍼 관리 방법과

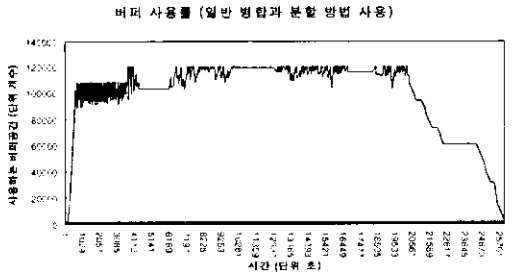

그림 13 버퍼 사용률 (일반적인 병합과 분할 방법을 사용한 경우)

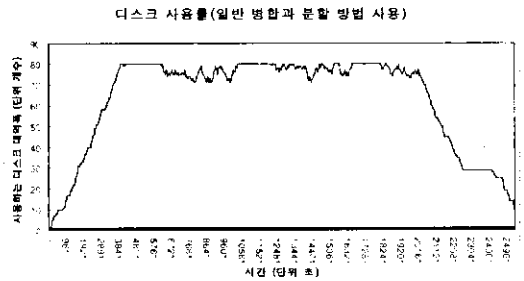

그림 14 디스크사용률(일반적인 병합과 분할 방법을 사용한 경우)

기존의 방법을 병행하는 사용했을 경우의 자원 사용률을 보여준다. 기존의 버퍼 관리 방법만 사용한 경우에 비하여 자원 사용률이 높은 경우에도 피기백킹 방법을 이용한 병합과 분할 방법을 이용하여 병합과 분할을 진행 할 수 있으므로 자원 사용률의 변화가 계속적으로 나타나는 것을 볼 수 있다. 그러나 그래프의 12000초부터 20000초 정도에서 자원 사용률이 높은 상태로 남아 있는 이유는 제안한 방법을 사용한 병합과 분할 방법은 시간을 필요로 하기 때문에, 그 시간 동안에 지속적으로 요구가 도착하게 되면 서버의 자원 사용률은 높은 상태로 남게 된다. 그러나 기존의 방법보다는 현재 서버의 자원 사용률을 적응적으로 떨어뜨려 동시에 많은 사용자를 서비스 할 수 있다는 점에서 좋은 효과를 보여준다. 실제적으로 시간대 별로 취소된 요구의 개수를 살펴보면, 기존의 병합과 분할 방법을 사용한 것 보다 제안한 방법이 요구의 취소개수가 절반 이상 낮다는 것을 볼 수 있다.

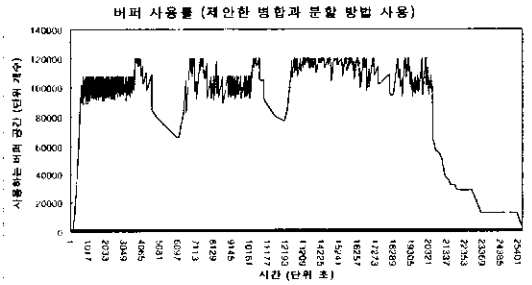

그림 15 버퍼 사용률 (제안한 병합과 분할 방법을 사용한 경우)

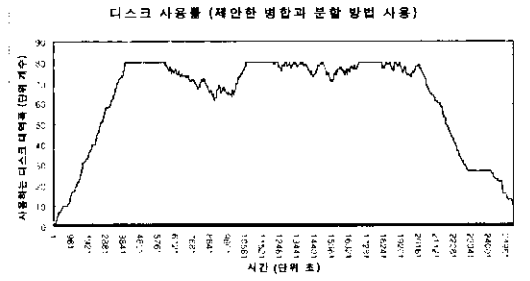

그림 16 디스크 사용률(제안한 병합과 분할 방법을 사용한 경우)

7. 결 론

본 논문에서는 사용자의 비디오에 대한 요구를 그룹

핑을 할 때, 비디오의 인기도에 기반을 두어 사용자가 기다려야 하는 시간을 결정하였다. 이렇게 결정된 시간은 버퍼공유를 할 때 세션의 크기를 결정하는데 중요한 파라미터로 사용되었으며, 이는 세션의 크기가 비디오 인기도에 의해 결정되었음을 알 수 있다. 또한 본 논문에서는 시간에 따라 변화하는 사용자의 요구와 서버 자원의 사용률에 적응적으로 대처할 수 있는 세션간의 병합과 분할 방법을 제안하였다. 이 방법은 기존에 있던 버퍼 관리 방법의 단점을 보안하여 추가의 자원을 사용하지 않고 병합과 분할을 진행할 수 있으며, 따라서 동시에 보다 많은 사용자의 요구를 서비스 할 수 있는 장점을 가지고 있다.

버퍼와 디스크 자원이 동시에 포화상태인 경우 수행되는 피기백킹 방법은 QoS를 떨어뜨린다는 단점을 가지고 있으나, 사람이 인지할 수 없는 5%이내의 가감 상태의 비디오 재현 속도 조절로 큰 문제를 유발하지 않고, 효율적으로 자원을 활용할 수 있도록 한다.

실제 환경과 유사한 파라미터로 시뮬레이션을 한 결과 병합, 분할 방법을 사용하지 않는 경우에는 곧 포화상태가 되어 더 이상의 사용자 요구를 수용할 수 없었으며, 기존에 제안되었던 병합, 분할 방법을 적용했을 경우에는 자원 사용률이 높을 때, 유사한 현상이 발행하여 사용자 요구를 수용할 수 없었다.

본 논문에서 제안한 적응적인 자원 공유 방식의 경우 버퍼 및 디스크 사용률에 상당부분의 여유 자원을 지속적으로 확보할 수 있어, 기존의 방법보다 더욱 많은 사용자 요구를 수용할 수 있었다.

참 고 문 헌

[1] Gregory R. Ganger, Bruce L. Worthington, Robert Y. How, and Yale N. Patt, "Disk Arrays, High-Performance, High-Reliability Storage Subsystems," IEEE Computer magazine, pp.30-36, 1994

[2] Kimberly Keeton, Randy H. Katz, "Evaluating video layout strategies for a high-performance storage server," Multimedia systems, vol.3, pp.43-52, 1996

[3] Asit Dan, Dinkar sitaram, Perwez Shahabuddin "Dynamic batching policies for an on-demand video server," Multimedia systems, vol.4, pp.112-121, 1996

[4] Hadas Shachnai, Philip S. Yu, " Exploring wait tolerance in effective batching for video-on-demand scheduling," Multimedia systems, vol. 6, pp.382-394, 1998

[5] Mohan Kamath, Krithi Ramamritham, Don Towsley, "Continuous media sharing in Multimedia Database Systems," Proceedings of the Fourth International Conference on Database Systems for Advanced Applications (DASFAA'95), April 10-13, 1995

[6] Wen-Jin Tsai, Shu-Yin Lee, "Dynamic Buffer Management for Near Video-On-Demand systems," Multimedia Tools and Applications 6, pp.61-83, 1998

[7] Leana Golubchik, John C.S. Lui, Richard R. Muntz, "Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers," Multimedia systems, vol. 4, pp.140-155,1996

정 홍 기

1996년 아주대학교 정보 및 컴퓨터 공학과 졸업. 1999년 아주대학교 정보 및 컴퓨터 공학 졸업 (공학 석사). 1999년 ~ 현재 아주대학교 정보통신 전문대학 박사 과정 중. 관심 분야는 VOD, 멀티미디어, Mobile computing.

박 승 규

1974년 서울대학교 공과대학 응용수학 졸업(공학사). 1976년 한국과학원(KAIST) 전산학 졸업(석사). 1982년 Institut National Polytechique de Grenoble 전산학 졸업(박사). 1976년 ~ 1977년 한국과학기술연구소(KIST) 연구원. 1977년 ~ 1978년 한국전자기술연구소(KIET) (현 ETRI) 연구원. 1978년 ~ 1982년 프랑스 그레노블 IMAG 연구원/학생. 1982년 ~ 1984년 한국 전자기술연구소(KIET) 실장/선임연구원. 1984년 ~ 1985년 미국 IBM 왓슨연구소 연구원. 1985년 ~ 1992년 한국전자통신연구소(ETRI) 연구위원 / 책임연구원. 1992년 ~ 현재 아주대학교 정보통신 전문대학원 교수. 관심분야는 다중 및 분산처리 컴퓨터구조, 멀티미디어처리 컴퓨터 시스템구조, 실시간 컴퓨터 시스템 및 성능 평가, 이동컴퓨팅 시스템