論文
5-9-8

# Near-VOD 서비스 환경에서 평균 오류의 최소화를 고려한 Balanced piggybacking 기술 (A Balanced Piggybacking Techniques with Minimizing Average Errors in Near-VOD Service Environment)

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 멀티 캐스팅 환경에서 서버의 자원을 동적으로 모니터하고 관리하여 효율적으로 서비스를 할 수 있는 버퍼 관리정책을 제안한다. 시뮬레이션 해본결과 전통적인 방식보다 버퍼의 활용과 QOS의 변동에서 약 23% 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 user-grouping policy for multi-casting services with dynamic monitoring and management of VOD sever resources. Simulation results show that the rate of buffer usage and QOS change of proposed scheme are about 23% performance improved than that of traditional methods. This implies that our method can allow much more users for given resources.

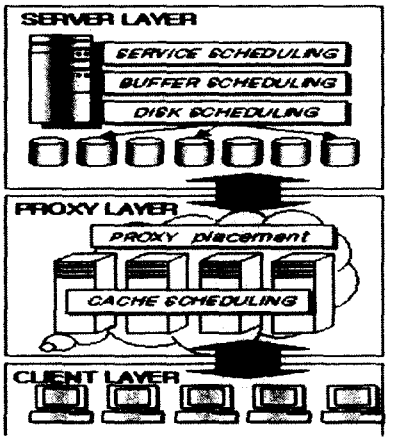
논문접수 : 2004. 10. 15.
심사완료 : 2004. 11. 3.

1) 정회원

# 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. 비디오 정보를 활용한 응용의 형태는 정보의 사용 시기와 그 저장 여부에 따라 실황 비디오 응용(live video application)과 저장 비디오 응용(stored video application)으로 구분될 수 있다[1]. 실황 비디오 응용의 대표적인 것으로는 화상 회의(live video conferencing)나 컴퓨터 협동 작업(Computer Supported Collaborati
ve Work ; CSCW)이 있으며, 이러한 응용 서비스를 활용하면 원격지의 구성원들이 대화식으로 멀티미디어 화된 정보를 실시간으로 교환함으로써, 서로의 생각을 공유하고 협의할 수 있다. 저장 비디오 응용으로는 디지털 도서관이나 주문형 비디오(Video-On-Demand ; VOD)가 있으며 사용자가 어떠한 주제에 관하여 저장된 비디오 자료를 검색함으로서 원하는 정보를 얻을 수 있다. Near-VOD 서비스는 실시간 대화형 기능이나 VCR의 Full operation (정지, 되감기, 빨리 감기 등)에 일부 제약이 있지만, true-VOD 서비스에 비하여 서버나 네트워크를 효율적으로 사용할 수 있는 방식이다. 그러나 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[8,9], 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. 플록시 캐싱 전략은 플록시 단독으로 행하여지는 캐시의 재배치 스케줄링[3]과 다수의 서버나 플록시를 활용하여 서비스를 할 경우 각 노드에 걸리는 부하를 상호 보완하여 서비스의 효율을 높이는 방안이 필요하다. 이를 위하여 버퍼나 캐시를 이용한 비디오 스트림 공유정책을 서버나 플록시에 활용하는 방안들이 제안되었다[10,11,12,13,14,15]. 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 서버의 자원을 모니터하여 사용자 서비스 중에 발생할 수 있는 서버 자원의 부족과 이에 의한 서비스의 중단 및 사용자 QOS의 변화를 최소화할 수가 있는 서버의 버퍼 관리 정책을 제안한다.




[그림1] VOD 시스템 구조

[Fig.1] *Architecture*

2장에서는 VOD 시스템의 서비스 스케줄링 및 버퍼공유 정책에 대한 관련 연구를 살펴보고 3장에서는 서버의 자원들을 효율적으로 관리하여 서비스 능력의 증가에 따른 사용자 QOS 변동 오류를 최소화 할 수 있는 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 연속 미디어 특성

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 대용량이다. 그러므로 자료를 압축하여 사용하는 방식만으로는 VOD 서버가 다수의 사용자를 수용하여, 연속적으로 서비스하기에는 여전히 문제가 된다. 그리고 데이터의 연속성 및 다양한 미디어와의 조합성, 시작과 끝의 명확성, 긴 서비스 시간성 등의 특징이 있기 때문에 비디오 스트림을 위한 서비스 스케줄링 정책도 이러한 특성을 감안해야할 필요가 있다.

### 2.2 버퍼 관리 스케줄링

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 세그먼트를 메모리에 저장하는 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, 이 방식을 토대로 버퍼의 활용률을 개선한 연구가 제안되었다.

### 2.3 사용자 서비스 스케줄링

서버의 자원을 효율적으로 사용하기위한 멀티 캐스팅(multi-casting)서비스에서는 동일한 비디오 스트림을 요구하는 사용자들을 시간적으로 동기화 시키는 정책이 필요하다. 이러한 정책을 사용자 그룹화(Grouping)라고 하는데, VOD 서비스에 활용되고 있는 배칭(batching)[5] 및 piggybacking[4]등에서 중요한 요소가 되고 있다. 배칭 정책은 동일한 비디오를 요구하는 사용자가 있을 경우, 이들의

서비스 개시 시간을 일정한 시간 간격(time interval)동안 모아서 함께 처리하는 방식이며, 이 방식은 배칭 간격에 따른 서비스 개시의 지연을 초래하는 불편함이 있다. 재생율 조정 방식은 서비스 중인 스트림의 재생률을 늘리거나 감소시켜서 현재 서비스 중인 다른 사용자 그룹과 통합하는 방식이며, piggyba
cking은 이 방식을 사용자 서비스 스케줄링에 도입한 것이라 할 수가 있다. piggybacking은 서비스 요구를 한 사용자에 대하여 일단 서비스를 개시하고, 그 이후 서비스 처리율을 조정하여 가까운 사용자 그룹에 포함시키는 방식이다. 결국 이 방식은 배칭과 유사하지만, 사용자 서비스 개시가 빠르다는 장점이 있다. 그러나 처리율의 조정에 의하여 서비스 품질의 저하를 초래할 수도 있으므로 이를 최소화하는 대책이 필요하다.

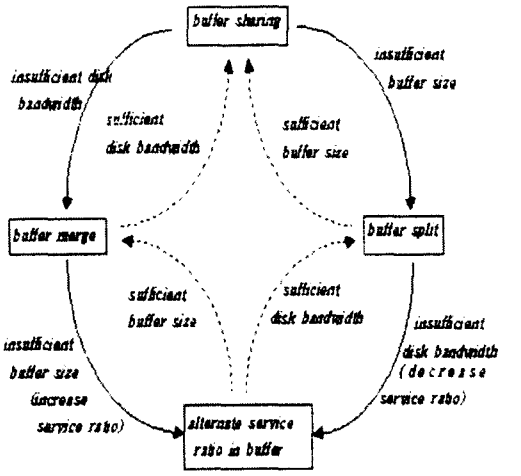
## 3. 균형적 piggybacking 기법

### 3.1 개요

서비스 재생율 조정은 비디오 재생률을 ±5% 한도 내에서 변화시키더라도 사용자는 그 변화를 잘 인식하지 못한다[2]는 것에 기초하고 있으며, 현재 진행 중인 두 개의 서비스를 하나의 그룹으로 만드는 작업은 버퍼 공유 스케줄링에서 서버나 플록시의 가용 자원이 부족할 경우 이를 해소할 수 있는 방법이라 할 수가 있다. 그러나 재생률의 변동은 사용자 QOS의 기준으로 볼 때는 일종의 오류로 취급 될 수가 있으므로 가능한 그 대상과 횟수를 줄여야할 필요가 있다. 본 논문에서 제안한 균형적 piggybacking은 종래의 piggybacking스케줄링을 버퍼 관리에 적용하여, 사용자 재 그룹화에 따른 디스크 대역폭의 급격한 변동을 방지하고, 재생률의 적용 범위나 처리시간을 최소화할 수 있는 스케줄링 정책을 제안한다. 또한 이 방식을 포함한 버퍼관리 정책은 서버의 자원 활용 상태에 따라 서로 연계되어 일관성을 갖게 함으로써 갑작스런 서버자원의 고갈로 인한 서비스의 장애를 방지하고, 효율적인 서비스를 할 수가 있다.

### 3.2 버퍼 관리 Baseball 모델

본 논문에서는 버퍼 병합, 버퍼 분할, 서비스 처리율 조정을 서로 연계하여 현재 서버의 자원 상황에 따라 자원의 관리를 효율적으로 스케줄링 할 수 있는 버퍼 관리 정책을 제안한다. 그림 2는 본 논문에서 제안한 버퍼관리 Baseball 모형을 나타낸 것이다.



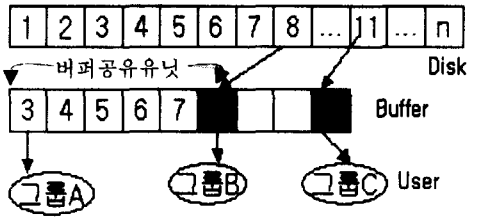

[그림 2] 버퍼 관리 *Baseball model*

[Fig.2] *Buffer sharing Baseball*

#### 3.2.1 *Buffer sharing state*

버퍼 공유 상태는 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 있는 상태이다. 버퍼 내에서 이러한 영역들이 분리되어 존재할 수가 있으며, 이러한 영역을 본 논문에서는 공유 버퍼 유닛이라고 하기로 한다. 프레임

그림3에서 보는바와 같이 그룹A는 그룹B가 사용한 후, 버퍼에 남겨둔 비디오 스트림 3을 재사용함으로써 신속한 서비스가 가능하고 디스크 액세스를 감소시킬 수가 있다. 여기서 프레임 번호 3에서 8까지가 한 개의 공유 버퍼 유닛이 되고, 그 크기는 단위 프레임 크기 * 6이 된다. 서비스를 진행하는 동안, 디스크 대역폭이나 버퍼가 부족할 경우, 각각 *Buffer merge state* 나 *Buffer split state*로 상태 전이된다.




[그림3] 버퍼 공유

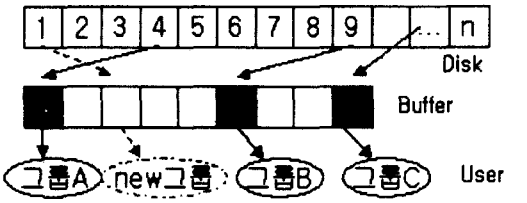
[Fig.3] *Buffer sharing*

### 3.2.2 *Buffer split state*

버퍼 분할 상태는 버퍼의 여유가 없을 경우에 버퍼 공유 유닛 단위 당 실제로 서비스 중인 프레임을 중심으로 분할하여, 프레임과 프레임 사이에 버퍼 공유를 위하여 사용한 부분을 회수한다. $i$ 번째 버퍼 공유 유닛의 프레임 수를 $Bfc\_i$ 라하고, 현재의 공유 버퍼 유닛 $i$ 에 서비스 중인 그룹수를 $Gc\_i$ 라 하면 $i$ 번째 버퍼 공유 유닛에서 회수될 수 있는 버퍼의 크기 $Bss\_i$ 은 식(1)과 같이 정의 할 수가 있다. 여기서 $Fs$ 는 단위 프레임의 크기이다. 또한 $i$ 번째 버퍼 공유 유닛에서 버퍼 분리에 걸리는 처리 시간 $Bst\_i$ 는 공유된 버퍼와의 연결을 삭제하는 시간, 즉 버퍼 스위치 시간 $Bswt\_i$ 와 분리된 그룹에 비디오 서비스를 위하여, 디스크를 엑세스하는 시간 $Dat\_i$ 이라 하면 처리 시간은 식(2)와 같이 정의 할 수 있다. 그림4에서 보는 바와 같이 그룹A와 그룹B사이의 버퍼 4단위가 회수되고 그 사이에 새로운 사용자 그룹을 서비스할 수가 있다.

$$Bss\_i = (Bfc\_i - Gc\_i) * Fs \quad (1)$$

$$Bst\_i = Bswt\_i + Dat\_i \quad (2)$$




[그림4] 버퍼 분리

[Fig.4] *Buffer split*

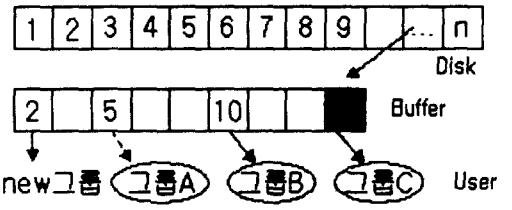
그러나 이 방식은 디스크의 읽기 횟수가 $Gc\_i - 1$ 만큼 증가하므로, 그 이를 수용할 디스크 대역폭에 여유가 없을 경우에는 서비스 처리율을 증가시키기 위하여 *alternate service ratio in buffer* 로 상태 전이되며, 버퍼의 여유 생기면 다시 *Buffer sharing state*로 상태 전이된다.

### 3.2.3 *Buffer merge state*

버퍼 병합 상태는 사용 디스크 대역폭의 여유가 없을 경우에 버퍼의 공유 도를 높여 디스크의 사용률을 감소시킨다. 먼저 현재 서비스 중인 다른 버퍼 중에서 가까운 번지에 위치한 인근 버퍼와 통합을 실시한다. 만일 $n$ 개의 공유 버퍼 유닛이 통합 되었다하면 $n-1$ 개의 디스크 읽기 동작을 절약할 수 있다. 그러나 이 상태는 동일한 비디오를 서비스하는 인근 버퍼끼리의 통합에 필요한 추가 버퍼가 있어야한다. 동일한 비디오를 서비스 받고 있는 인근의 공유 버퍼 유닛 $i$, $j$가 있고 현재 서비스 중인 프레임 번호가 $i < j$ 라고 할 때, 버퍼 통합에서 추가로 요구되는 버퍼의 크기를 $Bms\_ij$, 앞선 그룹의 마지막 프레임 번호를 $Gf\_l\_j$, 뒤따르는 그룹의 첫 번째 프레임 번호를 $Gf\_f\_i$ 라 하면 식(3)과같이 정의 할 수 있다. 그리고 버퍼 통합에 소요되는 시간을 $Bmt\_ij$ , 프레임 당 서비스 처리 시간을 $fp$ 라 하면, 병합 시간은 식(4)로 정의할 수 있다.

$Bms\_ij = ( Gf\_l\_j - Gf\_f\_i - 1 ) * fs$ (3)

$Bmt\_ij = ( Gf\_l\_j - Gf\_f\_i - 1 ) * fp$ (4)



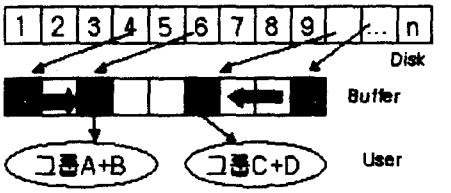

[그림5] 버퍼 통합

[Fig.5] *Buffer merging*

그림5에서는 3개의 그룹(New, A, B)이 그룹 C와 통합되는 형태를 보여주고 있다. 그리고 디스크 읽기동작은 4회에서 1회로 단축되었다. 만약 현재의 여유 버퍼의 크기가 *Bms_ij* 보다 작을 경우에는 서비스 처리율을 감소시키기 위하여 *alternate service ratio in buffer*로 상태 전이된다. 또한 이 상태에서 디스크 대역폭에 여유가 생기면 다시 *Buffer sharing state*로 상태 전이된다.

### 3.2.4 *Alternate service ratio in buffer state*

*Buffer merge state* 나 *Buffer split state*로 전이된 상태에서 버퍼 및 디스크의 여유가 없을 경우에는 버퍼 병합이나 분할 정책을 수행할 수 없기 때문에 버퍼내의 서비스 처리율을 증가시키거나 감소시켜 버퍼나 디스크의 여유 자원을 확보한다. 서비스 처리율 조정의 방식은 뒤쪽 프레임에서부터 처리율을 증가시켜, 앞쪽 프레임과 병합하는 방식인 처리율 증가 방식(Increase ratio time)과 앞쪽의 서비스 처리율을 감소시켜 뒤이어오는 프레임과 병합하는 처리율 감소 방식(Decrease ratio time)이 있다. 그리고 이 두 방식을 통합한 양방향 조정(Bi-way ratio time)이 있는데, 이 방식은 그룹 병합 속도가 바로 위의 두 방식보다 빠르기 때문에 자원의 부족이 급하게 예상될 때 사용할 수가 있다. 그림6은 그룹A, B, C, D의 현재 서비스 중인 프레임 번호가 4, 6, 11, 14라 하고, 그룹 내의 멀티태스킹 인원수가 각각 2, 10, 15, 8이라 할 때, 재생율 조정에 의한 그룹 병합의 예를 보여준다. 여기서는 재생율 조정의 시간과 그 범위를 최소화하기 위하여 그룹간의 병합은 일단 서비스 시간차가 짧은 두 개의 그룹을 우선으로 선정하고, 서비스 재생률의 조정은 멀티태스킹 인원수가 적은 그룹으로 한다. 결국 4개의 그룹은 각각 그룹B와 그룹C로 통합되며, 다음 차수에서는 두 개의 그룹이 다시 그룹C로 통합될 수가 있다.




[그림6] 서비스 재생율 변경

[Fig.6] *Alternative service ratio*

**(1) 그룹간의 시간편차가 클 경우**

버퍼 공유 유닛 내에서 시간차이가 가장 작은 두 개의 그룹을 선택한다. *G_dev* 를 그룹간의 처리시간의 표준편차, 를 기준치라 하고, *N*을 버퍼 공유 유닛내의 그룹 수, *Gst_k* 를 *k*번째 그룹의 현재 서비스 시간이라면 가장 짧은 거리 *Ts*는 식(5)과 같다.

*if G_dev >* *THEN*

$Ts = min(Gst\_j - Gst\_i)$

*where* $i=1....N-1$ , $j=2.....N$ , $i \neq j$ (5)

**(2) 그룹간의 시간차의 편차가 크지 않을 경우**

병합할 두개의 그룹을 선정하기 위하여 두개의 그룹의 크기를 더하여 가장 작은 것을 조사한다. *Ns*를 가장 작은 그룹의 크기라 하고 *N*을 버퍼 공유 유닛내의 그룹 수, *Gsn_k*를 *k*번째 그룹 내의 사용자 수라 하면 식(6)으로 작성될 수 있다.

*if Tdev* ≤ *THEN*

$Ns = min(Gsn\_j +, Gsn\_i)$

*where i=1....N-1 , j=2.....N , i ≠ j (6)*

**(3) 처리율 조정 그룹의 선택**

그룹 내 사용자의 수가 작은 그룹을 움직여 병합하는 방식을 제안한다. Gn_k를 k번째 그룹의 서비스 사용자의 수라 하고, forward 병합의 프로지저를 *forward_pro()*, backward 병합의 프로시저를 *backward_pro()*라 할 때 그룹간의 병합 방향을 결정하기 위한 식은 식(7)과 같다.

*if Gn_j < Gn_i*
*then backward_pro(Gn_i, Gn_j)*
*else forward_pro(Gn_i, Gn_j);* (7)

**(4) 처리 시간의 계산**

처리율 변경을 위한 초당 삭제 프레임 수를 *Fd* , 추가 프레임 수를 *Fa* 라하고, 초당 프레임 서비스 처리 수를 *Ef* 라 할 때 그룹 병합에 걸리는 시간은 *forward group merge*와 *backward group merge*
, *bi-way group merge*을 각각 식(8), 식(9), 식(10)로 표시 할 수 있다.

*Gmt_ij_For(Gst_j, Gst_j) = Ts * Ef / Fd (8)*

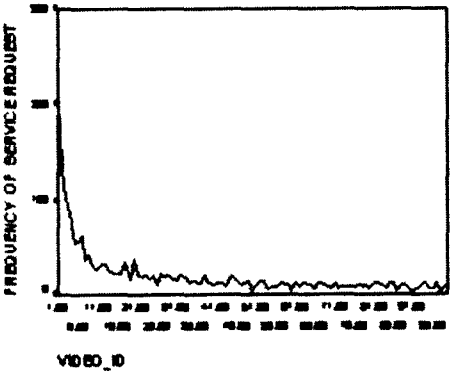
*Gmt_ij_Back(Gst_j, Gst_j) = Ts * Ef / Fa (9)*

*Gmt_ij_way(Gst_j, Gst_j) = Ts / (1 / Ed + 1 / Fa) (10)*

## 4. 시뮬레이션 및 분석

본 논문에서 제안한 균형적인 Piggybacking을 활용한 버퍼 관리 기법은 주요관심은 서버의 효율적인 자원의 활용과 서비스 품질 저하의 최소화 및 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 Distribution 을 기본으로 하였고, 서비스 도착 율은 λ는 1sec당 한 개의 평균 도착 율을 갖는 Poisson Distribution으로 하였다. 초당 처리 프레임의 수 Fs는 30으로 하였고, 프레임 추가(Fa), 삭제(Fd) 수는 QOS에 미치는 영향을 감안하여 공히 ±5%를 적용하였다. 또한 Zif Distribution에 의한 인기도 분포는 10~15% 으로 그림 7은100개의 비디오에 대한 요구 패턴을 나타낸다. 그러나 비인기 비디오의 버퍼공유는 별 효과가 없으므로, 실제로 시뮬레이션 할 때에는 상위 10개의 인기 비디오를 중심으로 하였다. 본 논문에서 제안한 균형적 piggybacking 버퍼관리 정책의 성능을 분석하기 위하여, 기존의 Adaptive piggybacking을 활용한 버퍼공유 정책(IC)과 비교 분석하여 보았다. 그림8은 이 두 정책간의 버퍼 사용률을 비교해본 것이다. 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. 버퍼 사용률은 한정된 시간동안 버퍼를 통하여 비디오 프레임을 서비스 받은 사용자 그룹의 수에 대한 공유 버퍼의 크기로 판단 할 수가 있다. 그래프 D1은 본 논문이 제안한 버퍼 관리 기법이며, D2는 인터벌 캐싱 정책이다. 기존의 인터벌 캐싱은 서비스 그룹 수에 비례적으로 버퍼를 요구하여 20개의 그룹을 서비스할 때부터 버퍼 한계치에 근접하게 되지만, 본 논문에서 제안한 버퍼관리 기법은 그 변동 정도가 크지 않고 안정적이며, 버퍼의 사용율도 25% 정도 절감하고 있다. 그만큼의 추가의 사용자 서비스가 가능하다고 할 수 있다. 그림9은 배칭 작업이 진행되는 동안, 버퍼를 액세스하여 서비스된 서비스수를 나타내었다. 버퍼의 여유가 남아 있는 서비스 초반에는 기존의 버퍼공유 정책의 히트 율이 높지만, 버퍼의 사용률이 높아질수록 제안된 버퍼관리 정책이 유리하게 나타나고 있다. 표2에서는 이를 도표로 정리하여 보았는데, 제안된 기법이 약간의 우수함만을 보이고 있다. 그 이유는 그림8에서

언급한 바와 같이 제안된 방식이 상대적으로 작은 버퍼의 점유율을 가지고 있기 때문이라고 생각된다. 그림 10에서는 사용자 QOS의 변동 현황을 나타내었다. 처리율 조정에 의한 QOS 변경 작업건수는 42, 32로 제안된 기법이 약간 높게 나타나고 있다. 그러나 중요한 점은 QOS 변경 대상 인원(519 : 772)과 그 작업 시간(743 : 1081)에서 낮게 나타나고 있는 점을 보면, 균형적 piggybacking 정책이 재생율 조정 작업 시에 QOS 변화를 감소시키는데 효과가 있다는 것을 알 수가 있다.



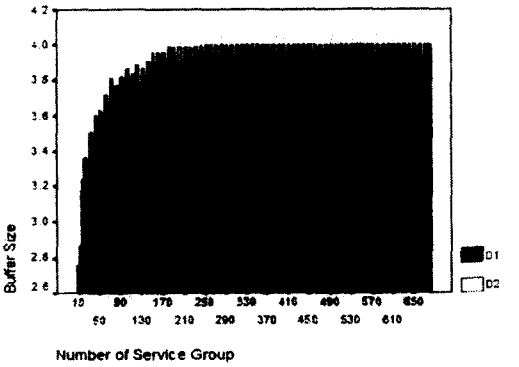

[그림 7] 비디오 요구 패턴

[Fig. 7] Request Video Pattern

[표 1] 서비스 파라메타 테이블

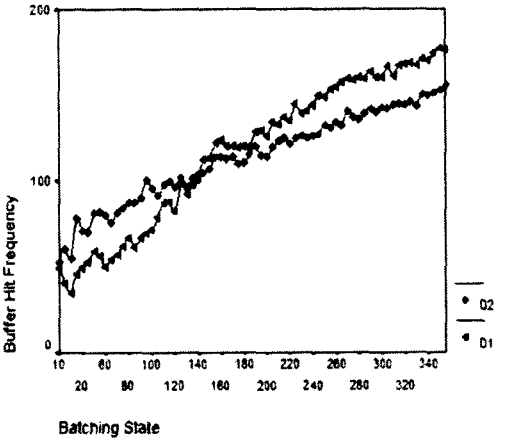
[Table.1] Service parameter table

| 구분 | 내용 | 값 |
|---|---|---|
| 비디오 요구 패턴 | Zif Distribution | |
| λ(포아송 분포) | 서비스 도착율 | 1/sec |
| V_no | 비디오 수 | 100개 |
| V_play_time | 비디오 상영 시간 | 3600 sec |
| Batch interval | 배칭 간격 | 60 sec |
| Service count | 서비스 요구 수 | 2000 |




[그림 8] 버퍼 이용도

[Fig. 8] Buffer Utilization



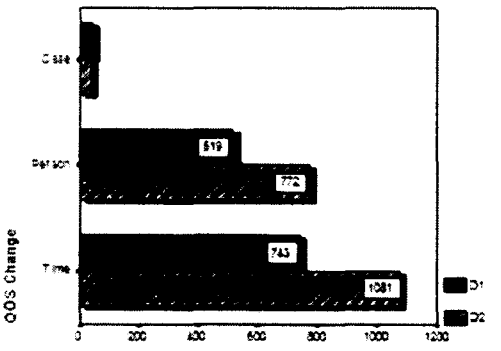

[그림 9] 버퍼 Hit 상태

[Fig. 9] Buffer Hit Status

[표2] 버퍼 Hit 빈도

[Table.2] Buffer Hit Frequency

| | | D2 | D1 |
|---|---|---|---|
| N | Valid | 2000 | 2000 |
| | Missing | 0 | 0 |
| Mean | | 113.7033 | 117.7814 |
| Variance | | 1104.1004 | 2102.8936 |
| Minimum | | .00 | .00 |
| Maximum | | 171.76 | 185.74 |




[그림 10] QOS 변동
[Fig. 10] QOS Change

## 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 균형적 piggybacking 버퍼관리 정책은 현재의 서버 가용 자원을 감안하여 스케줄링 하므로 효율적인 버퍼 관리를 할 수 있다. 그리고 버퍼 내에서 서비스 재생율 조정을 할 경우, 사용자 QOS의 변화를 최소화할 수 있는 방안을 제시하였다. 플록시의 성능이 높아지고 있는 지금, 이 방식은 비디오 서버 뿐만아니라, 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 analysis for buffer sharing and pre-patching techniques for multimedia systems", Multimedia Systems, pp 4: 55-69, 1996

[7] Sreenivas Gollapudi, Aidong Zhang, "Buffer model and management in distributed multimedia systems", Multimedia Systems, pp 206-218, 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] Subhabrata.Sen, Jennifer,Rexford, and Donald F. Towsley, "Proxy Prefix Caching for Multimedia Streams," in *Proc. of IEEE INFOCOM, pp=* "1310-1319", 1999

[11] A. Hue, " Video-on-demand broadcasting protocols: A comprehensive study" in Proc. IEEE INFOCOM, April 2001.

[12] K. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in Proc. ACM Multimedia, September 1998

[13] S. Sen, L. G o, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming,' in Proc. Inter. Workshop on Computer Communications and Networks, 1997

[14] A.Dan, D.Dias, R.Mukherjee, D.Sitaram, and R.Tewari,"Buffering and caching in large scale multimedia servers", *in Proceedings of IEEE COMPCON, pp.217-224,* March 1995

[15] A.Dan and D.Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments", *in IS&T SPIE Multimedia Computing and Networking Conference,(San Jose, CA),* January 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, .Kaashoek, and Jr.J.O'Toole, "Overcast:Reliable multicasting with an overlay network", in Proc. of 4st *Symposium on Operating Systems and Implementation(OSDI), pp=197-212,* October 2000

[19] 권춘자"동적 버퍼 분할을 이용한 클러스터 VOD 서버의 효율적 부하 분산 방법" 정보처리학회논문지, 제9권-C권, 제5호, 10, 2002