효율적인 버퍼 관리를 위한 동적 버퍼 할당 및 버퍼 교체 기법

김형진* · 나인호*

A Dynamic Buffer Allocation and Substitution Scheme for Efficient Buffer Management

Hyoung-Jin Kim* · In-Ho Ra*

요 약

클라이언트/서버 환경에서 컴퓨터의 성능을 측정하는 척도로써 주어진 작업에 대한 응답시간과 단위 시간 내의 얼마나 많은 작업들을 수행 완료할 수 있는가를 나타내는 처리율이 컴퓨터의 성능을 판단하는 지표로 사용되고 있다. 본 논문에서는 멀티미디어 서버에서 제한된 버퍼의 이용을 극대화하기 위해 상대적으로 작은 시간 간격으로 나누어 하나의 미디어 스트림을 버퍼에 할당/회수하는 윈도우 기반 버퍼 관리 기법을 제안하였다. 또한, 한번 사용된 데이터 블록에 대해 재 참조 시간을 계산하고 후속 스트림이 재사용할 수 있게 하여 멀티미디어 서버의 입출력 횟수를 줄일 수 있는 버퍼 교체 기법을 제안하였다.

ABSTRACT

Respond time and processing rate representing how many tasks can be done during an unit time in a client/sever environment are generally used for measuring the performance of computers, In this paper, we suggest a window buffer managing scheme based on a window with many of short-term sliced slots where a media stream is allocated and deallocated into them so that it can maximize the utilization ratio of the limited buffer on a multimedia server. And we also propose a buffer substitution scheme for reducing I/O times of a multimedia server by counting re-reference time point about a used block and then it can be reused by the next consecutive media stream.

키워드

Multimedia Sever, Dynamic Buffer Management

Ⅰ. 서 론

멀티미디어는 연속적인 미디어 스트림(오디오, 비디오)과 비연속적인 미디어 스트림(텍스트, 데이터, 이미지)을 포함하는 다른형의 데이터를 통합하는 것을 말한다. 이러한 스트림의 정보단위 사이에는 어떤 시간적인 관계가 존재한다. 멀티미디어 시스템은 자료를 저장하고 전송하고 나타낼 때 이러한 관계를 유지해야한다. 또한, 멀티미디어 데이터에 대한 관심이 늘어나고 이로 인한 멀티미디어 서버의 활용이 증가하는 추세에 있다.

본 논문에서는 멀티미디어 서버 환경에서 효율적인 버퍼링 시스템을 제안 하고자 한다. 먼저 기존의 정적 버퍼 할당 기법[1]의 문제점을 해결하기 위한 방안으로서 윈도우를 기반으로 한 동적 버퍼 할당(dynamic buffer allocation) 기법을 제시 한다. 또한, 한번 사용한 데이터 블록에 대해 재 참조가 가능한 버퍼 교체 기법을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 시스템 설계를 하고, Ⅲ장에서는 동적 버퍼 관리자에

* 군산대학교 전자정보공학부

접수일자 : 2005. 1. 28

대해 기술한다. Ⅳ장에서는 제안된 시스템의 성능 평가를 하고, 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 시스템 설계

윈도우 기반(window-based) 버퍼 관리 기법은 클라이언트의 서비스 지속 시간을 윈도우라는 상대적으로 작은 시간 간격으로 나누어 하나의 미디어 스트림에게 버퍼를 할당하고 회수하는 개념으로서 기존의 버퍼 관리 기법에 비해 버퍼 요구량을 최소화하면서 버퍼 이용율을 최대화할 수 있다는 장점을 제공할 수 있다. 제한된 버퍼 공간만을 이용할 수 밖에 없는 서버 환경에서 동시에 여러 클라이언트가 요구하는 서비스를 만족시킬 수 있다. 또한, 서로 다른 대역폭을 요구로 하는 다중스트림 멀티미디어 간에 동일한 크기로 분할된 버퍼 공간을 파이프라인 형태로 공유하기 위해서는 버퍼에 데이터가 할당되고 회수되는 단위가 동일해야 하기 때문에 제한된 버퍼 공간을 최대한 활용할 수 있다. 이에 서버가 서비스를 수행 할 때 각 윈도우 사이즈로 나누어 버퍼에 할당할 때 라운드 로빈(Round-Robin) 방식으로 버퍼를 공유한다[2].

따라서, 본 논문에서는 응용 프로그램에서 요구로 하는 서비스 품질과 미디어 객체의 크기 및 속도가 시간에 따라 변하더라도 이것을 탐지하여 자동적으로 실시간에 가장 적합한 윈도우 크기를 할당해 주는 방법을 제시하고자 한다.

또한, 사용자가 비디오 데이터에 대해 재생 후 고속탐색이나 역 탐색과 같은 사용자의 개입이 없는 경우 비디오 데이터는 순차적으로 접근이 이루어지기 때문에 특정 데이터 블록에 대한 참조시점을 예측할 수 있다. 그리고 하나의 비디오 데이터가 여러 사용자들에 의해 재 참조될 경우 특정 데이터 블록의 재 사용 시점도 예측할 수 있다. 따라서, 연속 매체의 재생 연산 중 요구되는 데이터의 전부 또는 일정 부분이 버퍼에 존재할 경우 디스크의 접근 횟수가 감속하여 시스템의 입출력 성능이 향상 된다. 제한된 버퍼크기를 고려할 때 비디오 데이터를 모두 버퍼에 유지하기보다는 데이터 블록의 재 참조 가능성을 판단하여 재 참조 가능성이 높은 데이터 블록을 버퍼에 유지할 수 있는 버퍼교체 기법이 필요하다. 따라서, 본 논문에서는 버퍼 공간에 대한 공유 기법과 데이터 블록의 공유 기법을 함께 적용하여 제한된 버퍼의 활용률을 높이고 멀티미디어 서버의 입출력 성능을 개선한 버퍼관리 기법을 제안하고자 한다.

본 논문의 멀티미디어 서버의 기본 구조는 그림 1과 같다.

서버는 멀티미디어 데이터를 저장하는 디스크와 디스크로부터 데이터를 읽어오는 서버 그리고 각각의 사용자에게 할당되는 버퍼로 구성 되어 있다. 또한, 멀티미디어 서버는 사용자 요청에 따라 버퍼를 할당한다.

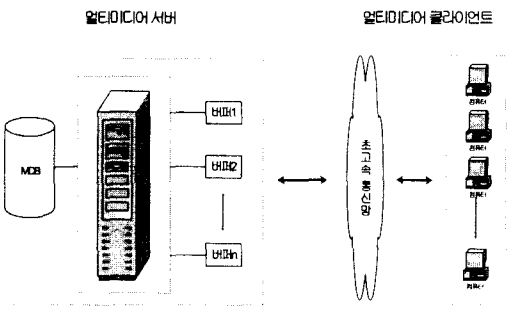


그림 4. 멀티미디어 서버 시스템 구조
Fig. 1 The structure of multimedia server system

Ⅲ. 동적 버퍼 관리자

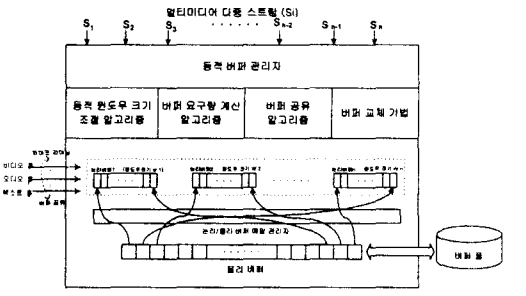


그림 5. 동적 버퍼 관리자
Fig. 2 Dynamic buffer manager

그림 2는 동시에 여러 클라이언트가 요구하는 멀티디어 스트림을 서비스하기 위해 본 논문에서 제안한 동적 버퍼 관지자의 전체적인 구조를 나타낸 것이다.

3.1 윈도우 크기 조절 기법

본 논문에서 제안한 윈도우 크기 조절 기법은 다음과 같은 절차를 거친다.

한 사이클 동안에 클라이언트가 요구한 스트림 데이터를 제공하기 위해서 동적 버퍼 관리자는 한 사이클 동안에 요구된 스트림을 저장하기 위한 버퍼 공간을 할당한다. 디스크 관리자는 디스크로부

터 데이터를 읽어들여 파이프라인 형태로 버퍼에 저장한다. 일단 디스크로부터 요구된 데이터를 해당 사이클 동안에 모두 읽어 들이면 이와 동시에 각 스트림을 요구한 클라이언트에 데이터 세그먼트 단위로 전달하여 처리하도록 한다. 파이프라인의 기본 구조는 환형 큐의 형태로 운영되므로 제일 먼저 저장된 첫 번째 스트림 데이터가 디스크에서 읽혀짐과 동시에 버퍼로 전송되어 소비되면서 사이클이 진행되고, 이때 각 스트림에 대한 데이터가 세그먼트 크기로 모두 전달된다. 이와 같은 과정에서 할당된 버퍼 공간은 어떤 시점에서 미디어 스트림의 서비스가 완료되거나 새로운 스트림의 요구가 발생할 경우, 그에 따른 버퍼 공간을 추가적으로 해제하거나 요구하게 된다. 따라서, 서비스되고 있는 미디어 스트림의 사이클 길이 단위는 고정된 크기의 윈도우 단위로 버퍼를 할당해야 한다. 본 논문에서는 윈도우 크기 단위로 버퍼를 할당/회수함으로써 버퍼 관리에 드는 오버헤드를 줄일 수 있도록 하였다.

3.2 버퍼 요구량

본 논문에서 얼마나 많은 버퍼 공간을 필요로 하는지 분석한다. 먼저, 각 읽기 주기의 끝에서 슬롯을 차지한 총 공간과 필요한 버퍼 공간의 양은 같다.

제안된 기법은 정확히 n개의 빈 슬롯이 필요하다. 그래서 제안된 기법의 버퍼 요구량은 최소 버퍼 요구량과 n개의 추가 슬롯의 합이다. 따라서 버퍼 요구량은 식(1)과 같이 주어진다.

$$\frac{n}{2}b+n*\frac{b*T}{n}=\frac{n}{2}b+\frac{n}{n-1}b=\frac{n+1}{n-1}*\frac{n}{2}b \qquad (1)$$

p : 소비율, T : 사이클의 길이, b : 버퍼 크기

$bT=(\frac{n}{n-1})b$이고 이상적인 버퍼공유는 $\frac{n}{2}b$버퍼 공간을 요구한다. 제안된 기법은 단지 $\frac{n+1}{n-1}$항이 추가되었다. 그리고 n값이 커짐에 따라 제안된 기법은 이상적인 최소값 요구량에 근접한다.

3.3 버퍼 공유 기법

버퍼 공유[3,4]은 여러 개의 미디어 스트림들이 버퍼를 공유할 경우에 버퍼 소비량을 상당히 절약할 수 있는 연구 결과를 토대로 발전되어 왔다. 본 논문에서 제안한 버퍼 공유 기법은 슬롯단위로 분할한 버퍼를 파이프라이닝 형태로 공유하는 기법으로서 병행 스트림의 개수에 따라 버퍼 요구량이 어느 정도인지를 결정하고 나서 버퍼 소비율에 따라 버퍼 공간을 시공간적으로 중첩 사용할 수 있도록 하는 파이프라이닝 기법을 도입한 것이다. 또한, 버퍼 요구량을 최소화하는 부분에 초점을 두었다.

그림 3은 4개의 병행 스트림에 대한 파이프라이닝 방식으로 버퍼를 할당하는 과정을 나타낸 것이다. 본 논문에서는 병행 스트림에 대한 버퍼 요구량과 사이클 길이를 윈도우 단위로 정의하여 윈도우 기반으로 한 버퍼 할당 알고리즘을 제안하였다.

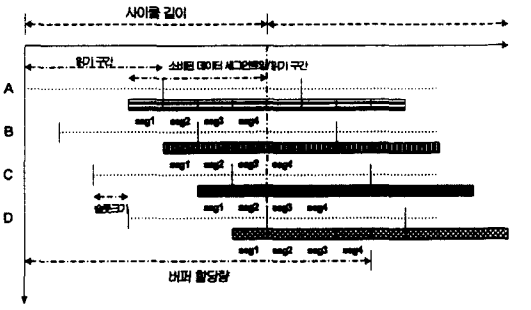


그림 6. 병행 스트림에 대한 파이프라이닝
Fig. 3 The pipelining of concurrent streams

그림 3에서 알 수 있듯이 파이프라이닝 형태로 버퍼를 슬롯단위로 나누어 할당하면 미디어 스트림의 수가 증가할수록 최적의 버퍼 공유 효율을 얻을 수 있다. 본 논문에서 제시한 파이프라이닝 방식의 버퍼 공유 알고리즘은 다음과 같다.

표 1. 파이프라이닝 알고리즘
Table. 1 pipelining algorithm

```
Input  : media stream
         cycle length
Output :
Calculate buffer requirement of each stream requested;
 Calculate total minimum buffer requirement;
 Devide memory space into slot units;
 Devide reading data of each stream into slot units;
 if(data segment <= buffer slot) {
   classify slot units by allocated group;
      for(the increment for cycle unit) {
         for(the increment for the number of reading
period) {
                  for(the increment of the slot number)
                     G(group number, slot number)
               store data segment in slot buffer;
               consume data segment in slot buffer;
         }
      }
   }
```

3.4 버퍼 교체 기법

본 논문에서 제안한 기법은 슬롯에 적재된 데이터의 재사용 가능성을 이용하기 때문에 재 사용될

슬롯은 재 사용되지 않을 슬롯보다 큰 가중치를 둔다. 또한 동일한 비디오 스트림을 참조하는 스트림들의 참조 시간 간격 정보를 이용하여 재사용 가능성의 근거로 사용한다. 즉, 같은 비디오 스트림을 접근하는 스트림들 중에서 임의의 스트림에 대해 참조 시간 간격이 기준값보다 작으면 이 임의의 스트림에 대하여 재 참조될 가능성이 높다고 판단한다. 이렇게 재 참조될 가능성이 높다고 판단된 스트림에 대해서는 최대한 오랫동안 버퍼에 유지해 디스크로부터 직접 데이터를 직접 읽어오는 것에 비해 바로 버퍼에 유지된 데이터를 사용함으로써 디스크의 접근횟수를 줄일 수 있을 뿐만 아니라 디스크 대역폭을 줄일 수 있고 또한 사용자의 평균 대기시간을 크게 줄일 수 있다. 스트림의 재 참조 가능성이 기준값과 비교했을 때 크면 재 참조될 가능성이 없다고 판단하여 버퍼를 재 할당하도록 한다.

스트림의 소비속도를 감안하면 스트림들에 대한 예상참조계획을 산출할 수 있으며 계산된 참조 시간 간격 정보는 표 1과 같은 스트림 관리테이블에 기록된다. 스트림 관리테이블에서 File_id는 스트림들이 참조하는 멀티미디어 데이터에 대한 구분을 하기 위한 파일 ID 정보를 갖는다. Reference_time은 멀티미디어 데이터에 대한 스트림의 참조시점을 나타내며 Interval 에는 동일한 멀티미디어 데이터를 참조하는 스트림들 간의 참조간격 정보를 저장한다.

표 2. 스트림 관리 테이블
Table. 2 Stream managing table

```
Static struct Stream
  {
     double File_id;       //멀티미디어 데이터 설명
     double Reference_time; //참조한 시간
     double Interval; //참조간격
  } Stream_Manager[USER_REQUEST];
```

제안된 기법에서는 스트림 관리테이블을 이용하여 버퍼를 관리하므로 스트림 관리테이블에는 서비스 중에 있는 스트림들에 대한 최신 정보가 유지되어야 한다. 따라서, 매 사이클마다 스트림들의 참조 시간 간격을 계산하여 변동사항이 발생하면 이러한 변동사항을 스트림 관리테이블에 반영해야 한다. 그러나 스트림들의 참조 시간 간격을 변화시킬 수 있는 새로운 사용자의 서비스 요청이나 스트림의 서비스 종료, 일시정지, 재시작 등의 사건이 발생하지 않으면 서비스중인 스트림들의 참조 시간 간격에는 아무런 변화도 일어나지 않으며 매 사이클마다 참조 시간 간격은 동일한 결과를 얻게된다. 따라서, 참조 시간 간격을 변화시킬 수 있는 사건(즉, 새로운 요구의 도착이나 스트리밍 서비스의 종료)이 발생할 때만 스트림들의 참조간격을 재계산하여 스트림 관리테이블에 반영해 스트림 관리테이블 유지에 수반되는 부하를 줄일 수 있다.

사용된 데이터 블록을 반환할 때에는 스트림 관리테이블의 참조 시간 간격 정보에 근거하여 이전 사이클에서 사용된 데이터 블록 중 일부만 버퍼 재할당을 위해 버퍼 풀에 반환하고 나머지는 버퍼에 유지한다. 이때, 버퍼 할당방식은 버퍼 풀 내 자유버퍼의 유무에 따라 달라진다. 즉, 자유버퍼와 반환된 버퍼들이 공존할 때는 버퍼 풀의 앞에 위치한 자유버퍼를 우선적으로 할당하며, 반환된 버퍼들만 존재할 때는 버퍼 풀의 뒤에 위치한 버퍼를 할당한다. 버퍼 풀이 비어있을 때는 버퍼에서 교체대상을 선정하여 할당한다. 교체대상으로는 버퍼 내의 데이터 블록 중 데이터 참조 시점이 현재의 서비스 사이클보다 작으면서 가장 최근의 블록을 담고 있는 버퍼를 선정한다. 이와 같이 재 참조 시간을 기반으로 한 버퍼 교체 기법의 알고리즘은 그림 4와 같다.

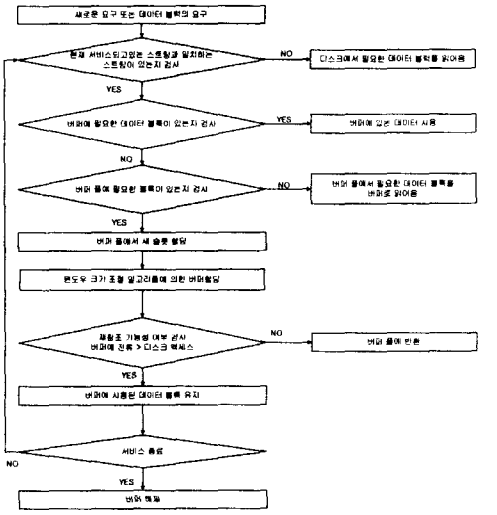


그림 7. 제안된 기법의 흐름도
Fig. 4 The flow chart of a suggested skill

Ⅳ. 성능평가

그림 5는 윈도우 크기 조절 기법을 통한 버퍼 공유 기법의 경우 이상적인 경우와 거의 유사한 버퍼 요구량을 보이고 있지만 데이터 공유의 경우 버퍼

공유의 경우 보다 다소 많은 양의 버퍼를 더 필요로 한다. 하지만, 버퍼 공유기법을 적용하지 않았을 경우보다 약 50%정도의 버퍼 절약률을 보인다.

조건 : 50명의 사용자, 100개의 비디오 스트림 중 하나의 비디오 스트림을 선택할 때 멀티미디어 서버에 대한 서비스 요청.

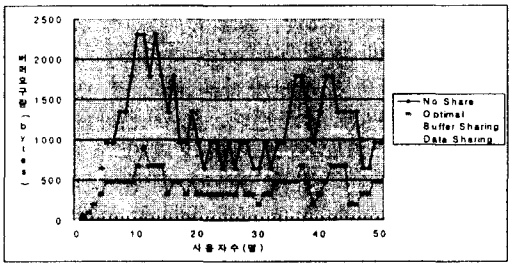


그림 8. 버퍼요구량 비교
Fig. 5 Compared to buffer demands

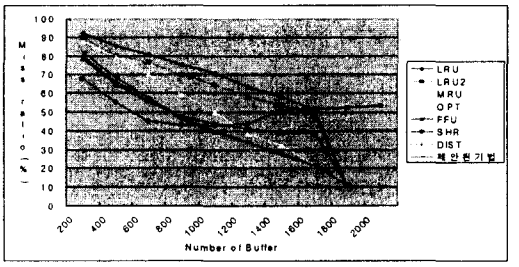


그림 9. 버퍼 공간의 크기 변화에 따른 성능 변화
Fig. 6 Performance change according to size change of buffer space

그림 6은 버퍼 공간의 크기를 변화시킬 때 기존 기법들의 성능변화와 본 논문에서 제안한 버퍼교체 기법을 비교한 것이다. 그림 6에서 알 수 있듯이 기존 버퍼교체 기법들에 비해 멀티미디어 응용을 위한 버퍼교체 기법의 성능이 우수함을 알 수 있다.

조건 : 시스템에 도착하는 평균 시간 간격은 비디오 데이터 길이의 절반인 900초로 정하고, 가용 버퍼 수의 범위는 200개 ~ 2000개로 설정.

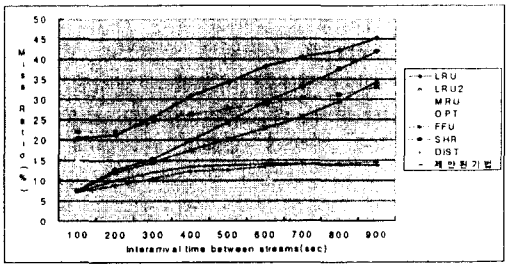


그림 10. 평균 요구 도착 시간 간격 변화에 따른 성능 변화 (1)
Fig. 7 Performance change according to a change of average demand arrival time slot (1)

그림 7은 모든 클라이언트들이 하나의 비디오 데이터만을 참조할 때, 그림 8과 그림 9는 클라이언트들이 Zipf's Law 분포에 따라 10개와 50개의 비디오 데이터들 중에서 하나를 선택하여 스트림 서비스를 요청할 때 버퍼 교체 기법의 성능 변화를 보여주고 있다. 요구 도착 시간 간격이 짧을수록 스트림들간의 참조시점 차이가 작아서 버퍼내의 데이터가 재 사용될 가능성이 높다. 그림 8에서 평균 요구 도착 시간이 50초일 때 범용기법들의 캐쉬 부재율이 11%~21%임을 감안할 때 데이터 공유를 지원하는 기법들의 캐쉬 부재율이 5%내외라는 점은 상당한 입출력 향상을 의미한다. 또한, 평균 요구 도착 시간 간격이 데이터 길이의 절반인 900초 이하에서는 모든 교체 기법들의 캐쉬 부재율이 50% 이하로 나타났다.

조건 : 버퍼 공간의 크기는 1500개, 평균 요구 도착 시간 간격의 범위(최소 50초에서 최대 900초).

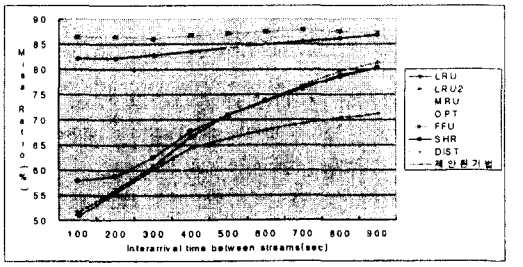


그림 11. 평균 요구 도착 시간 간격 변화에 따른 성능 변화 (2)
Fig. 8 Performance change according to a change of average demand arrival time slot (2)

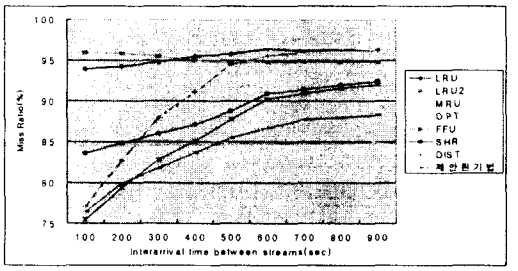


그림 12. 평균 요구 도착 시간 간격 변화에 따른 성능 변화 (3)

Fig. 9 Performance change according to a change of average demand arrival time slot (3)

VI. 결 론

본 논문에서 제안한 멀티미디어 서버에서 효율적인 버퍼 관리를 위한 동적 버퍼 할당 및 버퍼 교체 기법은 실시간에 가장 적합한 윈도우 크기를 할당해 주어 최소 버퍼를 이용하여 버퍼 활용률을 최대로 하고 버퍼 공간에 대한 공유 기법과 데이터 블록의 공유 기법을 함께 적용하여 제한된 버퍼 공간의 활용률을 높이고 멀티미디어 서버의 입출력 성능을 개선한 버퍼관리 기법을 제안했다. 또한, 시뮬레이션을 통해 멀티미디어 성능을 분석하고 기존의 버퍼 교체 기법들과 제안된 기법의 성능을 비교 분석하여 본 논문에서 제안한 기법의 우수성을 보였다.

참고문헌

[1] Chang, E. and Garcia-Molina, H., "Accounting for Memory Use, Cost, Throughput, and Latency in the Design of a Media Server", Technical Report SIDL-WP-1998-0096, Stanford University, 1998 (Available at http://www-db.stanford.edu/pub/papers/jvld98.ps).

[2] Chang, E. and Garcia-Molina, H., "Effective Memory Use in a Media Server", In proc.23rd Int'l Conf. on Very Large Data Bases, pp496-505, Athens, Greece, 1997.

[3] T. N. Raymond, C. Faloutsos and T. Sellis, "Flexible Buffer Allocation based on Marginal Gains", Proc. of ACM-SIGMOD, pp. 387-397, 1991.

[4] T. N. Raymond and J. Yang, "An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems", Multimedia Systems, Vol. 4, No. 2, pp. 55-69, 1996.

[5] S. Gollapudi and A. Zhang, "Buffer Management in Multimedia Database Systems", The third IEEE International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan, Jun. 1996.

[6] T. N. Raymond and J. Yang, "An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems", Multimedia Systems, Vol. 4, No. 2, pp. 55-69, 1996.

[7] Chang, E. and Garcia-Molina, H., "Bubble-Up:Low Latency Fast-Scan for Media Servers, In proc. 5th ACM Int'l conf. on Multimedia, pp87-98, seattle, WA, 1997.

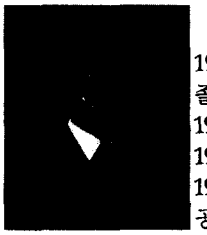
[8] Y. S. Ryu and K. Koh, "A Dynamic Buffer Management Technique for a Video-on-Demand Server", Proc. of IPSJ International Symposium on Multimedia Systems, Yokohama, Japan, Mar. 1996.

[9] To, T. P. and Hamidzaeh, B., "Dynamic Real-Time Scheduling Strategies for Interactive Continuous Media Servers", Multimedia Systems, Vol.7, No. 2, pp.91-106, 1999.

저자 소개

김형진(Hyoung-Jin Kim)

1997년 호원대학교 전자계산학과 졸업
1999년 군산대학교 정보통신공학과 석사
2004년 군산대학교 정보통신공학과 박사
2004. 9 ~ 현재 군산대학교 전자정보공학부 계약교수
※관심분야 : 멀티미디어 DBMS, 멀티미디어 시스템



나인호(In-Ho Ra)

1988년 울산대학교 전자계산학과 졸업
1991년 중앙대학교 전자계산학 석사
1995년 중앙대학교 전자계산학 박사
1995. 9 ~ 현재 군산대학교 전자정보공학부 부교수
※관심분야 : 멀티미디어 통신시스템, 초고속 통신망