참조 시간 간격 정보를 활용하는 버퍼 교체 알고리즘

고 정 국[†] · 김 길 용[††]

요 약

대규모 저장 공간과 실시간 특성이 요구되는 연속 매체 저장 시스템에서 입출력 장치는 성능 개선이 필요하다. 본 논문에서는 입출력 성능을 개선하기 위해 디스크 입출력 회수를 감소시키는 버퍼 공유 기법을 활용하였다. 또한, 연속 매체 데이터에 대한 요구를 예측하기 위해 연속 매체 스트림들의 참조 계획을 이용하여 데이터 공유를 증진시켰다. 본 논문에서는 동일 데이터를 요청하는 후속 사용자들이 버퍼를 효율적으로 공유케 하는 버퍼 공유 기법을 제안하였다. 제안된 알고리즘은 데이터 블록들에 대한 참조 시간 간격 정보를 활용하여 버퍼들을 관리한다. 제안된 알고리즘의 유효성을 검증하기 위해 시뮬레이션을 수행하였으며, 기존 버퍼 교체 알고리즘들에 비해 성능 개선 효과가 확인되었다.

A Buffer Replacement Algorithm utilizing Reference Interval Information

Jeong-Gook Koh[†] · Gil-Yong Kim[††]

ABSTRACT

To support large storage capacity and real-time characteristics of continuous media storage systems, we need to improve the performance of disk I/O subsystems. To improve the performance, we exploited buffer sharing scheme that reduces the number of disk I/Os. We utilized the advance knowledge of continuous media streams that is used to anticipate data demands, and so promoting the sharing of blocks in buffers. In this paper, we proposed a buffer replacement algorithm that enables subsequent users requesting the same data to share buffer efficiently. The proposed algorithm manages buffers by utilizing reference interval information of blocks. In order to verify validity of the proposed algorithm, we accomplished simulation experiments and showed the results of performance improvements compared to traditional buffer replacement algorithms.

1. 서 론

최근 저장 장치와 통신 기술의 빠른 발전으로 고속 통신망을 통해 사용자들에게 오디오나 비디오 같은 연속 매체(continuous media : CM) 데이터들을 제공할 수 있게 되었다. 사용자들에게 CM 데이터를 원활하게 제공하기 위해 저장 시스템은 다음과 같은 요구사항들을 충족시켜야 한다. 첫째, 비디오나 오디오 데이터는 전송 지연에 민감하다. 따라서, 스트림의 재생이 시작되면 저장 시스템은 스트림의 연속 재생을 보장할 수 있도록 충분한 시스템 자원을 할당해야 한다. 둘째, 비디오나 오디오 데이터는 디스크 검색과 데이터 전송을 위해 상당한 양의 시스템 자원들(예, 저장 공간, 대역폭 등)을 필요로 한다. 그러나, 기억장치와 통신 기술의 빠른 발전에 비해 입출력 장치의 느린 기

※본 연구는 1996년도 한국과학재단 핵심전문연구 과제의 연구비 지원에 의한 결과임(과제번호:961-0902-013-2).

† 준 회 원:부산대학교 컴퓨터공학과
†† 정 회 원:부산대학교 컴퓨터공학과

논문접수:1997년 9월 18일, 심사완료:1997년 11월 3일

술 발전은 시스템 성능 향상에 병목이 되고 있다. 연속 매체 저장 시스템에서는 다수의 저가 디스크를 사용하여 저장 공간과 데이터 전달 성능을 향상시키는 디스크 배열이 자주 사용되고 있다[2]. CM 데이터를 재생하기 위해서는 우선 주기억 장치에 적재해야 하는데 CM 데이터는 데이터의 크기가 크므로 입출력 요구에 대해 매번 디스크에 접근하면서 시스템 성능이 크게 저하된다. 따라서, 읽어온 CM 데이터들 버퍼에 유지하면서 최대한 많은 사용자들의 데이터 요청을 버퍼내의 데이터를 통해 서비스한다면 디스크 입출력이 감소되어 시스템의 성능이 향상될 수 있을 것이다. 일반적으로 연속 매체 연산들은 데이터에 순차적으로 접근하기 때문에 CM 데이터에 대한 참조 시점을 미리 예측할 수 있다. 따라서 일정 속도로 데이터에 순차 접근하는 연속 매체 연산에서는 버퍼내의 데이터가 다른 사용자에 의해 재사용될 시점을 예측할 수 있다.

본 논문에서는 연속 매체 저장 시스템에서 버퍼 공간을 효율적으로 관리함으로써 입출력 성능을 향상시키는 버퍼 교체 알고리즘을 제안하다. 제안된 기법은 이전에 참조되었던 데이터를 후행 사용자들이 재사용할 수 있도록 가능한 한 오랫동안 버퍼 캐쉬에 유지함으로써 데이터 요청에 대한 응답 시간도 단축시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 버퍼 교체 알고리즘들을 연속 매체 응용에 적용할 때의 문제점을 지적하고, 버퍼 공유 기법에 관련된 기존의 연구들을 살펴보았다. 3장에서는 연속 매체 저장 시스템 모델을 제시하고 버퍼 공유 기법의 효율성을 기술하였으며, 연속 매체 응용을 위한 버퍼 교체 알고리즘을 제안하였다. 4장에서는 시뮬레이션을 통해 제안된 알고리즘의 유용성을 평가하였고, 마지막으로 결론 및 향후 연구 계획을 5장에서 기술하였다.

2. 연구 배경 및 관련 연구

2.1 연구 배경

제한된 버퍼 공간을 관리하는 기존의 버퍼 교체 알고리즘들로는 OPT(Optimal), FIFO(First-In, First-Out), LRU(Least-recently-used), MRU(Most-recently-used), MFU(Most-frequently-used) 등이 있다.

OPT는 디스크 입출력을 최소화할 수 있으나 운영체제가 미래의 참조 순서를 미리 알고 있어야 하므로 실제적인 적용은 어렵다. 따라서, 기존의 저장 시스템에는 OPT 알고리즘에 근접한 성능을 제공하는 버퍼 교체 알고리즘들이 사용되고 있다. 그러나, 이러한 알고리즘들은 과거의 참조 기록에 근거하여 버퍼를 관리하므로 데이터 접근 시점이 예측가능한 연속 매체 응용에는 부적합하다.

LRU는 알고리즘의 복잡도가 낮고 구현이 간단하여 기존 버퍼 관리자는 대부분 LRU를 사용한다. 그러나, 최종 참조 정보만을 이용하기 때문에 상대적으로 자주 참조되는 데이터와 자주 참조되지 않는 데이터를 구별하지 못하는 단점이 있다.

최종 참조 정보만을 이용하는 LRU의 성능을 개선하기 위해 LRU-K는 버퍼에 대한 참조 간격을 이용하여 교체될 버퍼를 선정한다[1]. K는 버퍼 참조 시점간의 시간 간격 정보를 구하는데 사용되는 버퍼의 참조 수를 나타낸다. LRU-K는 데이터에 대한 참조 지역성을 잘 반영하며 다중 사용자 환경에서도 우수한 성능을 제공한다. 그러나, 과거의 참조 정보만으로 버퍼 교체 대상을 결정하므로 일반 응용에서는 LRU보다 우수하지만 연속 매체 응용에서는 큰 이득을 기대할 수 없다.

따라서, LRU나 LRU-K 등의 기존 알고리즘들은 시간대별로 가변적이고 특정 자료에 편중된 접근 형태를 보이는 응용 분야의 저장 시스템에 적용하기에는 적합하지 않다.

2.2 관련 연구

연속 매체 저장 시스템은 실시간으로 데이터에 접근하는 사용자들에게 속도를 보장해야 한다. 이를 위해 저장 시스템은 승인 제어(admission control) 알고리즘과 자원 예약(resource reservation) 알고리즘을 적용하고 있으며, 자원 예약 대상은 버퍼 공간과 디스크 대역폭이다. 초기의 연구들은 사용자마다 개별 버퍼 공간을 할당하는데 중점을 두었고, 데이터의 공유는 고려하지 않았다. 그러나, 최근에는 연속 매체 저장 시스템에서 사용자들간에 데이터를 공유할 수 있는 방안에 대한 연구가 많이 진행되고 있다[6]. 다음은 버퍼 공유 기법을 적용한 버퍼 관리 체제들에 대한 다수의 연구들 중 대표적인 것들이다.

Kamath[6]는 멀티미디어 데이터베이스 시스템에서 최대 서비스가능 사용자 수를 늘리는데 연속 매체 공유 기법이 유용함을 설명하였다. CM 데이터 공유를 위해 "캐쉉" 개념을 도입하였으며, 동일 데이터에 대한 다수 사용자들의 요청을 그룹화하여 처리하기 위해 일괄처리(batching) 기법도 함께 사용하였다. 사용된 버퍼는 대개 자유 풀(free pool)에 반환되지만, 버퍼 공유가 가능하면 사용된 버퍼를 변환하지 않고 특정 연산(예, *fix/unfix*)을 통해 유지하여 버퍼내의 데이터를 공유할 수 있도록 한다.

Özden[7]은 멀티미디어 저장 시스템을 위한 버퍼 교체 알고리즘을 제안하였다. 각 서비스 사이클이 시작될 때 이전 사이클들에서 소모된 버퍼들은 동일 CM 파일을 참조하는 사용자들의 오프셋 차이(distance)에 따라 반환된다. 버퍼들의 반환과 할당은 클라이언트마다 할당된 MRU 리스트의 앞에서 발생한다.

FFU 알고리즘[9]에서는 데이터 스트림에 대한 데이터 요청 단위를 "세그먼트"로 정의하였으며 연속 매체 캐쉉 개념을 소개하였다. FFU 알고리즘에서는 세그먼트들이 병행 트랜잭션의 세그먼트 참조 순서에 따라 유지된다. 임의의 트랜잭션이 참조한 세그먼트는 재참조에 대비하여 별도의 큐에 저장된다. 선행 트랜잭션에 의해 참조된 세그먼트는 연속된 후행 트랜잭션이 재참조하는 경우에만 버퍼에 캐쉉되며, 새로운 세그먼트를 저장할 자유 버퍼공간이 부족할 때는 트랜잭션이 가장 마지막으로 참조할 세그먼트중의 하나를 선정하여 제공한다.

본 논문에서는 기존의 버퍼 공유 기법을 바탕으로 연속 매체 저장 시스템에서 버퍼를 효율적으로 관리할 수 있는 버퍼 교체 알고리즘을 제안한다. 공유가 가능한 시점부터 선행 사용자가 사용한 버퍼를 특정 연산을 이용하여 유지하는 [6]과 달리 제안된 알고리즘에서는 동일 CM 파일에 접근하는 다수 사용자들의 재참조 시간 간격 정보를 이용하여 버퍼를 버퍼 캐쉬에 유지한다. 또한, 매 라운드마다 사용된 버퍼를 반환하고 버퍼의 반환과 할당이 클라이언트마다 할당된 MRU 리스트의 앞에서 발생하는 [7]에 비해 제안된 알고리즘은 재참조 가능성이 가장 낮은 버퍼만을 반환하고 나머지는 버퍼 캐쉬에 유지한다. 트랜잭션별로 별도의 큐를 유지하면서 세그먼트를 관리하는 FFU 알고리즘[9]과 달리 제안된 기법에서는 트랜잭션 개념을 사용하지 않고, 재참조될 버퍼는 버퍼 캐쉬에 유지하며 재참조 가능성이 가장 낮은 반환된 버퍼라도 버퍼 공유를 촉진시키기 위해 자유 리스트의 뒤에 추가한다.

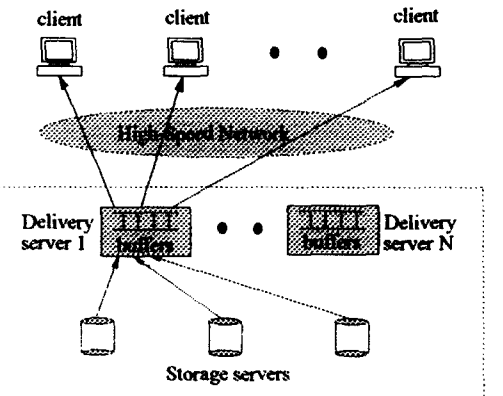
3. 참조 시간 간격 정보를 활용하는 버퍼 교체 알고리즘

본 장에서는 연속 매체 저장 시스템 모델을 제시하고 연속 매체 환경에서 버퍼 공유기법의 효율성을 설명한 후, 참조 시간 간격 정보를 활용하는 버퍼 교체 알고리즘을 제안한다.

3.1 시스템 모델

연속 매체 저장 시스템은 비디오나 오디오 데이터를 디스크에 저장해 놓고 사용자들의 검색 요청에 따라 해당 데이터를 디스크에서 버퍼 공간으로 읽어 들인 후 사용자에게 제공한다. 비디오나 오디오 데이터는 주기적인 특성을 가지므로 라운드(round)라고 하는 일정 시간 간격을 단위로 반복적으로 동작한다[3, 4]. 연속 매체 저장 시스템은 CM 데이터의 연속 재생을 보장하기 위해 매 라운드마다 스트림 고유의 재생 속도에 따라 일정한 수의 비디오 프레임이나 오디오 샘플을 제공한다.

본 논문에서는 (그림 1)과 같은 "클라이언트-서버"

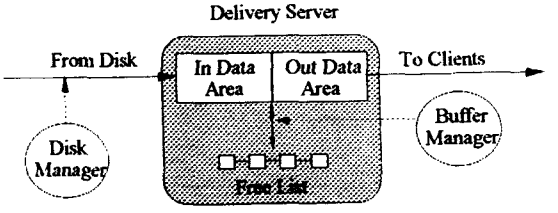


(그림 1) 연속 매체 저장 시스템 모델

(Fig. 1) The structure of a continuous media storage system

구조를 갖는 연속 매체 저장 시스템을 고려한다. 서버는 역할에 따라 저장 서버(storage server)와 전달 서버(delivery server)로 구분된다. 저장 서버는 CM 파일을 저장하며 전달 서버의 요청에 따라 주기적으로 데이터 블록을 전달 서버에 전송한다. 전달 서버는 사용자의 요청에 따라 저장 서버로부터 전송된 CM 데이터를 버퍼에 저장한 후 CM 데이터를 사용자들에게 제공한다. 통신망은 데이터 전달 지연과 전송 대역폭 측면에서 요구되는 QoS(Quality-of-Service)를 보장한다고 가정한다. 다수의 독립된 전달 서버들은 사용자들의 요구를 병행 처리하며, 승인 제어와 자원 예약 알고리즘을 사용하여 CM 파일에 대한 실시간 접근을 보장한다. 한편, 사용자는 중지(pause)와 재개(resume) 등 상호 작용적인 기능을 사용하여 전달 서버의 데이터 전달을 제어한다.

(그림 2)는 전달 서버의 버퍼 관리 구조를 보여준다. 전달 서버의 버퍼 공간은 입출력을 위해 할당된 버퍼와 자유 리스트(global pool)로 구성되며, 버퍼의 크기는 비디오 데이터 1초 분량으로 가정한다. 버퍼 관리자는 자유 리스트로부터 버퍼를 할당한다. 할당된 버퍼들은 전달 서버에 전송된 데이터를 수신하는 입력 데이터 영역(In Data Area)과 사용자에게 데이터를 전달하는 출력 데이터 영역(Out Data Area)으로 구분되며, 버퍼 캐쉬를 형성한다.



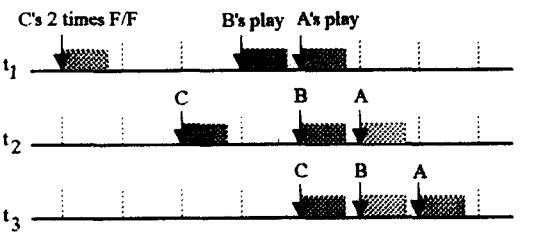
(그림 2) 전달 서버의 버퍼 관리 구조

(Fig. 2) The buffer management structure of a delivery server

3.2 버퍼 공유 기법의 효율성

본 절에서는 참조 계획을 이용하는 버퍼 공유 기법의 효율성에 대해 기술한다. 데이터를 순차적으로 접근하는 연속 매체 연산에서는 데이터에 대한 참조 시점 예측이 가능하므로 디스크로부터 읽어온 데이터를 공유하면 소요되는 버퍼량과 디스크 대역폭이 감소하여 저장 시스템이 지원할 수 있는 최대 사용자 수도 증가된다. 예를 들어, 단위 버퍼의 크기가 32 KB이며 비트율(bit rate)이 1.5 Mbps인 비디오 파일을 저장하고 있는 연속 매체 저장 시스템에 사용자로부터 특정 CM 파일에 대한 정상 속도의 재생 요청이 도착할 때, 해당 파일의 10번째 데이터 블록에 대한 오프셋(offset)과 데이터 블록의 참조 시간을 다음과 같이 예측할 수 있다. 파일 오프셋의 초기치는 0이므로 10번째 블록의 파일 오프셋은 32 KB×9=288 KB이며, 비트율을 고려할 때 10 번째 블록의 참조 시간은 재생 연산이 시작된 후 288 KB/1.5 Mbps=1.536초가 된다.

(그림 3)은 동일 CM 파일에 대해 정상 속도의 재생 연산을 수행하는 사용자 *A*, *B*와 고속 재생(F/F) 연산을 2배속으로 수행하는 사용자 *C*가 존재할 때 CM 파일에 대한 단위 시간별 접근 형태를 보여준다. t_1시점에서 사용자 *A*가 참조한 데이터는 t_2시점에서 사용자 *B*에 의해 재사용되고, t_3시점에서는 사용자 *C*에 의해 재참조된다. 따라서, 다수 사용자들이 동일 CM 파일에 병행 접근할 때 선행(leading) 사용자에 의해 사용된 데이터는 후행(lagging) 사용자들 중 요구 도착 시간 간격과 재생 위치(offset)가 최근접한 후행 사용자에 의해 재참조될 수 있으므로 효율적인 데이터 활용이 가능하다.



(그림 3) CM 파일에 대한 단위 시간별 접근 형태

(Fig. 3) Illustration of data access at each time instant in the shared case

예 1)에서는 연속 매체 저장 시스템이 지원할 수 있는 최대 사용자 수를 계산하여 버퍼 공유 기법이 효율적임을 보인다.

예 1) 연속 매체 데이터의 압축 기법으로 비트율이 4

Mbps인 MPEG-2를 사용하는 시스템에서 버퍼 공간 크기는 100 MB이며 디스크의 데이터 전달률은 10 MB/sec라고 가정한다. 단위 버퍼의 크기를 비디오 데이터 1초 분량으로 가정할 때, 버퍼 공간에 적재할 수 있는 데이터량은 $100\ MB/(0.5\ MB/sec) = 200\ sec$로서 1초 분량의 데이터 블록을 200개까지 저장할 수 있다. 또한, 디스크의 데이터 전달률만을 고려하면 최대 서비스가능 세션 수는 $(10\ MB/sec)/(0.5\ MB/sec) = 20$이다. 버퍼 공간의 크기가 최대 200개의 세션에게 각각 1초 분량의 비디오 데이터를 제공할 수 있지만 디스크 대역폭에 의해 세션 수는 최대 20개로 한정된다. 한편, 버퍼 공유로 얻게 되는 잉영 버퍼 공간 및 디스크 대역폭은 CM 데이터의 선반입이나 새로운 사용자의 서비스 요청에 적용될 수 있다. 20개의 CM 파일이 각각 $C_1, C_2, ..., C_{20}$개의 병행 세션들에 의해 공유될 때 최대 서비스가능 세션 수는 $C_1 + C_2 + ... + C_{20} = \sum_{i=1}^{20} C_i$가 된다. 그러나, $\sum_{i=1}^{20} C_i$는 버퍼를 공유하는 세션들의 요구가 시스템에 동시 도착할 때 가능하며, 세션들의 요구 도착 간격이 너무 크면 버퍼 공유가 불가능하여 최대 서비스가능 세션 수는 20개가 된다. 따라서, 세션들의 요구 도착 간격과 도착 시간을 고려한 서비스가능 세션 수의 범위는 다음과 같다.

$$20 \le \textit{the number of sessions supported} \le \sum_{i=1}^{20} C_i$$

3.3 참조 시간 간격 정보를 활용하는 버퍼 교체 알고리즘

본 절에서는 버퍼 공유 기법을 이용하며 동일 CM 파일을 재생하는 사용자들의 참조순서에 따라 선행 사용자에 의해 사용된 데이터들이 최대한 많은 후행 사용자들에 의해 재사용되도록 하며, 버퍼 관리에 "참조 시간 간격(*interval*)" 정보를 활용하는 버퍼 교체 알고리즘을 제안한다. 또한, 동일 CM 파일에 대한 병행 사용자가 여러 명일 때, 선행 사용자의 오프셋을 기준으로 최근접 오프셋을 갖는 후행 사용자와의 오프셋 차이를 "선행 사용자의 참조 시간 간격"이라는 용어로 정의한다. (그림 2)의 t_1시점에서 사용자 A의 참조 시간 간격($interval_A$)은 최근접 후행 사용자가 B이므로 1 단위 버퍼이며, 사용자 B는 사용자 C에 대해 3 단위 버퍼이다. t_2시점에서 사용자 B의 참조 시간 간격($interval_B$)은 2 단위 버퍼이며, 후행 사용자가 존재하지 않는 사용자 C는 최대값인 ∞가 주어진다.

새로운 사용자의 요청에 대해 연속 매체 저장 시스템은 승인 제어 알고리즘을 사용하여 새로운 요청의 승인 여부를 결정하며, 승인된 새로운 요청은 다음 라운드부터 서비스를 받게 된다. 따라서, CM 파일에 대한 사용자의 재생 속도를 고려하면 CM 파일에 대한 사용자들의 참조 계획을 단위 시간별로 예측할 수 있다. 한편, 제안된 알고리즘에서는 사용자들의 파일 오프셋을 이용하여 참조 시간 간격을 계산한 후, 오름차순으로 정렬하여 (그림 4)와 같은 구조를 갖는 사용자 테이블(C_table)을 구축한다. 한편, 사용자들간의 참조 시간 간격을 변화시킬 수 있는 사건 - 새로운 사용자의 도착이나 종료, 재생 중지와 재개 - 이 발생하지 않는 한 참조 시간 간격을 계산하더라도 동일한 C_table을 얻게 되므로, 본 알고리즘에서는 사건이 발생할 때만 참조 시간 간격을 재계산하여 C_table에 반영한다.

```
struct client {
    unsigned  Cid;        // Client ID
    unsigned  Fid;        // CM File ID
    long   Offset;        // File Offset
    long   Interval;      // Reference Interval
} C-table;
```

(그림 4) 사용자 테이블에 대한 자료 구조
(Fig. 4) Data structure for C_table

버퍼 반환 시에는 C_table의 참조 시간 간격 정보를 이용하여 직전 라운드에서 사용된 버퍼들 중 일부를 자유 리스트에 반환하고 나머지는 버퍼 캐쉬에 유지한다. 버퍼들의 반환 위치는 재참조 가능성에 따라 결정된다. 즉, 자유 리스트에 반환되는 버퍼들은 재참조 가능성이 가장 적은 버퍼들로서, 참조 시간 간격이 최대값인 사용자에 의해 참조된 데이터를 담고 있다. 버퍼의 할당은 자유 리스트내의 자유 버퍼 존재 여부에 따라 자유 버퍼가 존재하면 자유 버퍼를, 자유 버퍼가 없으면 재참조 가능성이 가장 적은 버퍼부터 우선적으로 할당한다.

참조 시간 간격 정보를 이용한 버퍼 관리 방법은 다음과 같다.

1) 직전 라운드에서 사용된 버퍼들 중 참조 시간 간격이 최대값인 사용자에 의해 참조된 버퍼는 자유 리스트에 반환되고 나머지는 버퍼 캐쉬에 유지된다.
2) 새로운 라운드에서 사용될 데이터 블록을 결정한다.
3) 해당 데이터 블록의 버퍼 캐쉬상 존재 여부를 검사한다.
4) 버퍼 캐쉬에 해당 블록이 존재하지 않으면, 자유 리스트에서 존재 여부를 검사한다.
   4.1) 자유 리스트에 존재하면 해당 데이터 블록을 버퍼 캐쉬로 이동시킨다.
   4.2) 자유 리스트에도 없으면 자유 버퍼의 가용 여부를 검사한다.
      4.2.1) 자유 버퍼가 존재하면 자유 버퍼를 할당한 후 디스크 입출력을 요청한다.
      4.2.2) 자유 버퍼가 없으면 재참조 가능성이 가장 낮은 버퍼를 할당한 후 디스크 입출력을 요청한다.

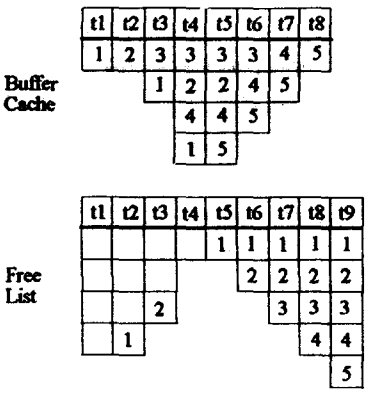
예 2) CM 파일을 정상 속도로 재생하는 사용자 *A*, *B*, *C*는 아래의 참조 계획을 갖는다. 총사용가능 버퍼의 수가 5 개이며 제안된 알고리즘을 적용할 때, (그림 5)는 단위 시간별 사용자 테이블의 변화와 매 라운드별 버퍼 공간의 상태 변화를 보여준다.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A	1	2	3	4	5			
B			1	2	3	4	5	
C				1	2	3	4	5

(a) 참조 계획

	Cid	Offset	Interval
t_0			
t_1	*A*	1	∞
t_2	*A*	2	∞
t_3	*A*	3	2
	B	1	∞
t_4	*B*	2	1
	A	4	2
	C	1	∞
t_5	*B*	3	1
	A	5	2
	C	2	∞
t_6	*B*	4	1
	C	3	∞
t_7	*B*	5	1
	C	4	∞
t_8	*C*	5	5
t_9			

(b) 단위 시간별 사용자 테이블의 변화



(C) 버퍼 공간의 상태 변화

(그림 5) 참조 계획에 따른 사용자 테이블과 버퍼 공간의 상태 변화

(Fig. 5) The change of C_table and buffer spaces according to future reference sequence

단위 시간별로 C_table은 다음과 같이 변화한다. 재생 연산이 시작되기 전인 t_0시점에서 C_table은 비어 있으며, t_1시점에서는 최초 사용자 *A*가 C_table에 반영된다. t_1/t_2시점에서는 사용자 *A*에 의해 참조된 데이터를 요구할 후행 사용자가 존재하지 않기 때문에 참조 시간 간격은 최대값(∞)이 되며 버퍼는 자유 리스트에 반환된다. t_3시점에 새로운 사용자 *B*가 추가되므로 참조 시간 간격을 재계산하고 결과에 따라 서비스 순서를 결정한다. 이 때 *A*가 사용한 데이터는 재참조 가능성이 있으나 *B*에 의해 사용된 데이터는 재참조 가능성이 없기 때문에 *A*의 데이터는 버퍼 캐쉬에, *B*의 데이터는 자유 리스트에 반환한다. t_5시점에서 사용자 *A*의 재생 연산이 종료되므로 *A*에 대한 정보를 C_table에서 제거한다. t_8시점에서 마지막 사용자 *C*의 재생 연산이 종료되므로 t_9시점에는 C_table이 비게 된다.

제안된 알고리즘의 유용성을 평가하기 위해 2명의 사용자가 동일 CM 파일을 재생 속도 r로 재생할 때 알고리즘들의 성능을 비교한다. 사용되는 기호는 〈표 1〉과 같으며, 성능 비교 기준으로 "디스크 입출력 회수"를 사용한다. 버퍼 공간의 최초 상태는 어떠한 데이터도 적재되어 있지않은 초기 상태(cold start)이므로 단위 사용자가 CM 파일을 재생하기 위해 요구되는 디스크 입출력 회수(IO)는 CM 파일의 길이인 $length_{CM}$이 된다.

〈표 1〉 기호 정의
〈Table 1〉 Symbols used in this paper

기 호	설 명
$length_{CM}$	CM 파일의 길이(데이터 블록 수)
$interval$	사용자간의 참조 시간 간격 (데이터 블록 수)
T	서비스 라운드의 최대 길이 (=1초)
r	재생 속도
d	버퍼 크기 (=T x r)
n_B	총 버퍼 수

정리 1: 버퍼 공간의 최초 상태가 초기 상태이며 두 사용자가 동일 CM 파일에 접근할 때, LRU 알고리즘 사용에 따른 디스크 입출력 회수(IO)는 다음과 같다.

1. $IO = length_{CM}$이면 if and only if $n_B \geq length_{CM}$.
2. $n_B < length_{CM}$일 때
   - ① $IO = length_{CM}$ if $interval < n_B/2$.
   - ② $IO = 2x(length_{CM} + interval - n_B)$ if $n_B/2 \leq interval < n_B$. (1)
   - ③ $IO = 2x\ length_{CM}$ if $interval > length_{CM}$. (2)

정리 2: 동일한 상황에서 제안된 알고리즘 이용시 디스크 입출력 회수는 다음과 같다.

1. $IO = length_{CM}$이면 if and only if $n_B \geq length_{CM}$.
2. $n_B < length_{CM}$일 때
   - ① $IO = length_{CM}$ if $interval \leq n_B/2$.
   - ② $IO = length_{CM}$ if $n_B/2 < interval < n_B$. (3)
   - ③ $IO = 2x\ length_{CM} - n_B$ if $interval \geq length_{CM}$. (4)

상황 2.2와 2.3에 대한 두 알고리즘간 디스크 입출력 회수의 차는 다음과 같다.

$$\text{수식 (1) - 수식 (3)}: IO = 2x(length_{CM} + interval - n_B) - length_{CM} = length_{CM} + 2x(interval - n_B) \quad (5)$$

$$\text{수식 (2) - 수식 (4)}: IO = 2x\ length_{CM} - (2x\ length_{CM} - n_B) = n_B \quad (6)$$

수식 (5)와 (6)으로부터 알고리즘간 디스크 입출력 회수의 최대 차($IO_{difference}$)는 $IO_{difference} = max\{(length_{CM} + 2x(interval - n_B)), n_B\}$이다.

이상의 결과는 제안된 알고리즘이 LRU 알고리즘에 비해 성능이 우수함을 보여준다.

4. 성능 평가

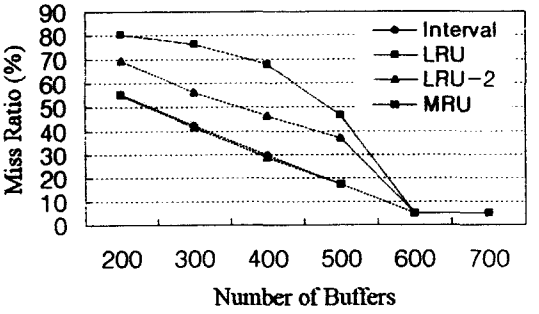
본 장에서는 제안된 알고리즘의 효율성을 평가하기 위해 일련의 시뮬레이션을 통해 성능을 평가하였으며, 제안된 버퍼 교체 알고리즘(Interval)과의 성능 비교를 위해 기존 응용에 사용되는 LRU, LRU-2, 그리고 MRU 알고리즘을 선정하였다. 알고리즘의 성능 비교에 사용될 성능 척도(performance metric)로는 총 입출력 요구 수와 실제 입출력 수의 비인 "미스율(*miss ratio*)"을 사용하였다. 시뮬레이션 프로그램은 *Delphi-Sim* 시뮬레이션 패키지를 이용하여 구현하였다.

사용자들의 평균 요구 도착 간격 시간(mean inter-arrival time)은 지수 분포를 따르며 최대 지원가능 사용자 수는 20명으로, 단위 버퍼의 크기는 비디오 데이터 1초 분량으로 시뮬레이션 매개변수(parameter)를 설정하였다. 또한, 길이가 600초(=10분)인 10개의 CM 파일들이 사용자들의 접근 반도에 따라 정렬되어 있을 때 사용자들의 접근 유형은 *Zipf* 분포를 갖는다고 가정하였다. Zipf 분포를 갖는 10개의 CM 파일 중에서 i 번째 CM 파일이 참조될 확률 $f_i = \frac{C}{i^{(1-\theta)}}$ $(1 \leq i \leq 10)$ (where θ: the parameter for the Zipf distribution, C: the normalization factor)이다. 실제로 VOD와 같은 연속 매체 응용에서 사용자들이 접근 유형은 $\theta = 0.271$ 이하의 Zipf 분포를 갖는 것으로 알려져 있다[8].

4.1 버퍼 수의 변화에 따른 성능 변화

(그림 6)과 (그림 7)은 가용 버퍼 수의 변화가 버퍼

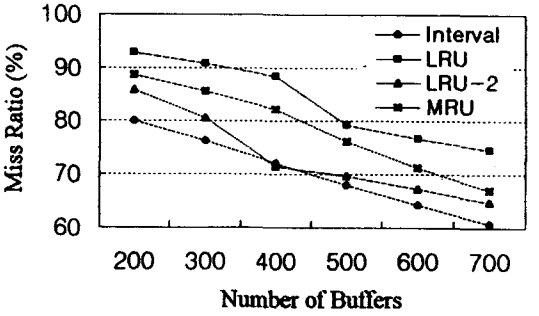
교체 알고리즘의 성능에 미치는 영향을 나타내고 있다. 각 사용자들의 평균 요구 도착 시간 간격은 CM 파일 길이의 절반인 300초이며, 가용 버퍼 수의 범위는 최소 200개에서 최대 700개이다.



(그림 6) 동일 CM 파일에 접근하는 경우

(Fig. 6) The change in miss ratios while the number of buffers is changed. All users access the same CM file

(그림 6)에서 보듯이 모든 사용자들이 동일 CM 파일에 접근하면 버퍼 공유 가능성은 증가하여 미스율이 급격히 낮아진다. 또한, 버퍼 공유 가능성이 높아지므로 (그림 7)에 비해 전반적으로 미스율이 낮다. 버퍼 공간의 크기가 CM 파일의 길이보다 커지면 파일 전체를 보관할 충분한 버퍼 공간이 확보되어 최초 사용자가 참조한 데이터를 버퍼 캐쉬에 유지할 수 있게 된다. 이 후 버퍼 캐쉬를 이용하여 후행 사용자들의 데이터 요청을 서비스하면 디스크 입출력 회수도 줄어들게 된다.



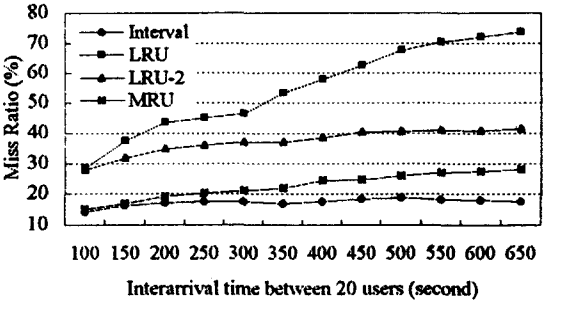
(그림 7) 10개의 CM 파일을 임의 선택하는 경우

(Fig. 7) The change in miss ratios while the number of buffers is changed. Users access 10 CM files

다수 사용자들이 CM 파일들을 임의 선택할 때는 참조된 데이터가 버퍼 캐쉬내에 유지될 가능성이 적어지므로, 버퍼 수가 작을 때에는 상당한 디스크 입출력이 유발된다. 그러나, 연속 매체 응용에서 사용자들의 자료 접근 유형은 일부 자료에 편중되어 있으므로 (그림 6)과 (그림 7)에서 보듯이 버퍼 공유 기법을 채택한 알고리즘이 우수한 성능을 발휘하고 있다. 특히, VOD 응용과 같은 연속 매체 연산의 특성상 이전 라운드에서 사용된 버퍼가 현재 라운드에서 재사용될 가능성이 희박하므로 MRU 알고리즘은 제안된 알고리즘에 근접한 성능을 나타내고 있다. 제안된 알고리즘을 기준으로 (그림 7)의 시뮬레이션 결과를 비교하면 LRU 알고리즘은 최고 29.4% 내외, LRU-2 알고리즘은 22.96% 내외이며, MRU 알고리즘은 4.01% 내외의 미스율 차이를 보였다.

4.2 평균 요구 도착 시간 간격의 변화에 따른 성능 변화

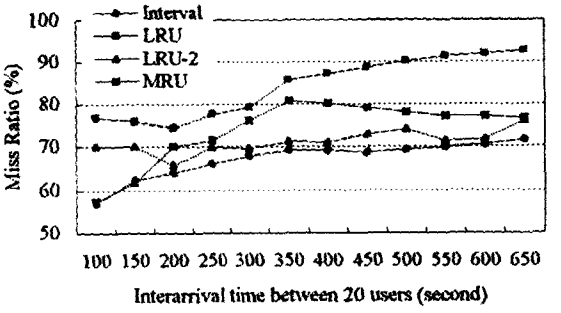
버퍼 공간의 크기를 CM 파일의 길이보다 작은 500개로 설정하였을 때, 사용자들의 평균 요구 도착 시간 간격의 변화가 여러 알고리즘들의 성능에 미치는 영향을 기술하였다. 평균 요구 도착 시간 간격의 범위는 100초에서 650초까지 변화한다. (그림 8)은 20명의 사용자가 모두 동일 CM 파일에 접근할 때 버퍼 캐쉬 미스에 미치는 영향을 보여주며, (그림 9)는 사용자들이 10개의 CM 파일을 임의로 선택할 때 알고리즘들의 성능 변화를 나타내고 있다.



(그림 8) 동일 CM 파일에 접근하는 경우

(Fig. 8) The change in miss ratios while interarrival time of users is changed. All users access the same CM file

(그림 8)에서 보듯이 평균 요구 도착 시간 간격이 300초 이하일 때는 모든 알고리즘들의 미스율이 50% 이하이며, 시간 간격의 변화에 따른 LRU-2, MRU, 그리고 제안된 알고리즘의 전반적인 성능 변화도 크지 않다. 그러나, LRU는 시간 간격의 변화에 따라 미스율이 선형으로 증가하고 있다. 특히, 선행 사용자가 참조한 데이터를 후행 사용자가 재참조할 가능성이 매우 높기 때문에 제안된 알고리즘은 시간 간격의 변화에 관계없이 20% 이내의 미스율을 나타내고 있다. 시뮬레이션 결과로부터 제안된 알고리즘은 LRU 알고리즘과 비교할 때 최대 56.24%, LRU-2 알고리즘과는 23.84%, 그리고 MRU 알고리즘과는 10.5%의 성능 차이를 보였다.



(그림 9) 10개의 CM 파일을 임의로 선택하는 경우
(Fig. 9) The change in miss ratios while interarrival time of users is changed. Users access 10 CM files

여러 사용자들에 의해 다수의 CM 파일이 임의로 선택될 때에는 다양한 데이터들이 요구되기 때문에 참조된 데이터들이 버퍼 캐쉬에 지속적으로 유지될 가능성이 희박해진다. 즉, 새로운 데이터들이 빈번하게 버퍼 캐쉬에 적재되고 참조된 데이터들이 반환되면서 알고리즘에 따라 다양한 성능 차이를 나타난다. 제안된 알고리즘의 미스율을 기준으로 (그림 9)의 시뮬레이션 결과를 비교하면 LRU와는 21.1%, LRU-2는 13.08%, 그리고 MRU와는 11.59%의 최대 성능 차이를 보인다.

시뮬레이션을 통한 성능 평가 결과 제안된 알고리즘이 모든 경우에서 가장 성능이 우수함을 확인하였다. 또한, 모든 사용자들이 동일 CM 파일에 접근할 때는 MRU 알고리즘의 성능, 10개의 CM 파일을 임의로 선택할 때에는 LRU-2의 성능이 제안된 알고리즘과 유사하였다. 이러한 결과는 다중 사용자 환경에서 다양한 데이터가 참조될 때는 참조 지역성을 반영할 수 있는 LRU-2 알고리즘이 MRU 알고리즘보다 성능이 우수함을 보여준다. LRU 알고리즘은 자주 참조되는 데이터와 그렇지 못한 데이터를 구별하지 못하기 때문에 모든 상황에서 가장 좋지 못한 성능을 나타내었다.

이상과 같이 제안된 알고리즘이 연속 매체 응용에 적용된다면 데이터의 효율적인 활용을 통해 요구 버퍼량과 디스크 대역폭의 감소 효과 외에도, 후행 사용자의 데이터 요청에 대한 응답 시간 단축 및 입출력 성능 개선 효과도 얻을 수 있다.

5. 결 론

대규모 저장 공간과 실시간 특성이 요구되는 연속 매체 저장 시스템에서 입출력 장치는 성능 개선이 필요하다. 본 논문에서는 한번 사용된 데이터라도 후행 사용자들에 의해 재사용될 수 있도록 버퍼 캐쉬에 유지함으로써 저장 시스템의 성능을 향상시키는 버퍼 교체 알고리즘을 제안하였다. 제안된 알고리즘은 동일한 연속 매체 파일에 접근하는 사용자들의 참조 시간 간격 정보를 이용하여 버퍼를 관리하며, 재생 연산 외에도 빨리 감기(F/F)나 되감기(Rewind)와 같은 상호 작용적인 연속 매체 연산에 대해서도 적용이 가능하다.

시뮬레이션 결과로부터 기존 알고리즘들에 비해 제안된 알고리즘의 성능 개선 효과를 확인할 수 있었으며, LRU에 비해서는 56.24%, LRU-2에 대해서는 23.85%, 그리고 MRU에 비해서는 11.59%의 최대 성능 향상을 보였다. 이러한 결과는 연속 매체 응용에서 사용자들의 참조 시간 간격 정보를 이용하여 버퍼 공유를 가능케 하는 제안된 알고리즘이 기존의 버퍼 교체 알고리즘들에 비해 효율적임을 보여주고 있다.

참 고 문 헌

[1] E. J. O'Neil, P. E. O'Neil and G. W. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," *Proc. of the ACM*

SIGMOD, pp. 297-306, 1993.

[2] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Array of Inexpensive Disks (RAID)," *Proc. of the ACM SIGMOD*, pp. 109-116, 1988.

[3] Rangan, P.V., H. M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communication Magazine*, Vol. 30, pp. 56-65, July 1992.

[4] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers," *Proc. of the ACM Multimedia 94*, pp. 33-40, Oct 1994.

[5] J. L. Peterson and A. Silberschatz, Operation System Concepts, 2nd Ed., Addison-Wesley Pub. Co., pp. 217-228, 1985.

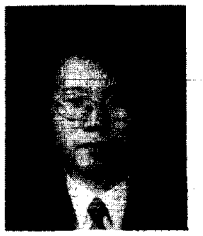
[6] Mohan Kamath, Krithi Ramaritham and Don Towsley, "Continuous Media Sharing in Multimedia Database Systems," *Proc. of 4th Int'l Conference on Database Systems for Advanced Applications(DASFAA '95)*, 1995.

[7] Banu Özden, Rajeev Rastogi and Avi Silberschatz, "Buffer Replacement Algorithms for Multimedia Storage Systems," *Proc. of the Int'l Conference on Multimedia Computing and Systems*, pp. 172-180, June 1996.

[8] A. Dan, D. M. Dias, and P. S. Yu, "Buffer Analysis for a Data Sharing Environment with Skewed Data Access," IEEE Trans. Knowledge and Data Engineering, Vol. 6, pp. 331-337, 1995.

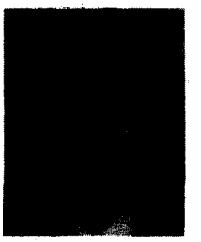
[9] 권택근, 이석호, "연속 매체를 위한 FFU 버퍼 재배치 알고리즘," *한국정보과학회 논문지(B)*, Vol. 22, No. 10, Oct 1995.

[10] 고정국, 김길용, "연속 매체 데이터를 위한 버퍼 교체 알고리즘 설계," *한국정보처리학회 춘계 학술발표논문집*, Vol. 4, No. 1, pp. 277-280, Apr 1997.



고 정 국

1992년 부산대학교 컴퓨터공학과 졸업(학사)
1994년 부산대학교 대학원 컴퓨터공학과(공학석사)
1994년~현재 부산대학교 대학원 컴퓨터공학과 박사과정
관심분야: 분산 시스템, 멀티미디어 시스템, 실시간 시스템



김 길 용

1981년 서울대학교 수학교육과 졸업(학사)
1983년 서울대학교 대학원 컴퓨터공학과(공학석사)
1988년 서울대학교 대학원 컴퓨터공학과(공학박사)
1983년~1986년 금성반도체(주) 연구원
1994년~1995년 Univ. of Southern California(USC) 객원교수
1988년~현재 부산대학교 컴퓨터공학과 정교수
관심분야: 분산 시스템, 멀티미디어 시스템, 실시간 시스템