실시간 멀티미디어 데이터베이스 환경을 위한 효율적인 버퍼교체 기법 설계 및 구현

신재룡[†] · 피준일[†] · 유재수[††] · 조기형[†††]

요 약

본 논문에서는 실시간 멀티미디어 데이터를 위한 효율적인 버퍼 교체 기법을 제안한다. 제안하는 기법은 실시간 특성을 고려하기 위해 다단계의 우선순위 레벨을 갖는다. 각 우선순위 레벨은 처음 참조된 데이터를 위한 콜드 셋(cold set)과 재 참조된 데이터를 위한 핫 셋(hot set)으로 구분된다. 희생 데이터 선정 작업은 버퍼 할당을 요구하는 트랜잭션의 우선순위 레벨보다 낮은 레벨만을 대상으로 콜드 셋의 최하위 레벨부터 핫 셋의 최상위 레벨까지 순차적으로 수행된다. 콜드 셋의 각 레벨에서는 가장 큰 미디어부터 교체 대상으로 선정하고, 핫 셋의 각 레벨에서는 가장 긴 참조 간격을 갖는 미디어부터 선정한다. 이로 인해 한정된 버퍼 공간에 많은 수의 인기 있는 미디어를 오랫동안 유지시킬 수 있으므로 버퍼 히트 비율이 증가되고, 많은 수의 서비스 요청을 처리할 수 있게 되어 전체적인 시스템 성능은 향상된다. 제안하는 기법에 대한 성능 평가에서는 Priority-Hints 기법을 대상으로 버퍼 히트 비율 및 트랜잭션의 마감시간 초과 비율을 비교한다. 이를 통해 기존의 기법들보다 제안하는 기법의 성능이 뛰어남을 보인다.

Design and Implementation of an Efficient Buffer Replacement Method for Real-time Multimedia Databases Environments

Jae Ryong Shin[†], Jun Il Phi[†], Jae Soo Yoo[††] and Ki Hyung Cho[†††]

ABSTRACT

In this paper, we propose an efficient buffer replacement method for the real-time multimedia data. The proposed method has multi level priority to consider the real-time characteristics. Each priority level is divided into a cold data set that is likely to be referenced for the first time and a hot data set that is likely to be re-referenced. An operation to select the victim data is sequentially executed from the cold set with the minimum priority level to the hot set with the maximum priority level. It is chosen only at the lower level than or equal to the priority of the transaction that requests a buffer allocation. In the cold set, our method selects a media that has the maximum size in the level for a target of victim first of all. And in the hot set, our method selects a medium that has the maximum interval of the reference first of all. Since it maintains many popular media in the limited buffer space, the buffer hit ratio is increased. It also manages many service requests. As a result, our method improves the overall performance of the system. We compare the proposed method with the Priority-Hints method in terms of the buffer hit ratio and the deadline missing ratio of transactions. It is shown through the performance evaluation that our method outperforms the existing methods.

Key words: 실시간, 멀티미디어, 데이터베이스, 버퍼 교체

본 연구는 한국과학재단 목적기초연구(특정기초연구 과제번호 R01-1999-00244)지원으로 수행되었음.

[†] 충북대학교 정보통신공학과 박사과정

[††] 정회원, 충북대학교 전기전자및컴퓨터공학부, 컴퓨터정보통신연구소 부교수

[†††] 충북대학교 전기전자및컴퓨터공학부, 컴퓨터정보통신연구소 교수

1. 서 론

최근 들어, 메모리 가격이 저렴해지고, 용량은 크게 증가함에 따라 대용량 멀티미디어 데이터를 위한 메모리 버퍼 공간 할당이 용이해졌다. 또한 저장 장치와 정보통신 기술의 비약적인 발전에 따라 초고속 정보통신망 및 차세대 인터넷 망 등이 구축되면서 대용량의 문서, 영상, 오디오 정보와 같은 다양한 멀티미디어 데이터를 저장하고 검색, 처리하는 서비스가 일반화되고 있다. 산업, 과학, 국방, 교육, 행정, 의료 등 사회 전반에 걸쳐 멀티미디어 컴퓨팅 기술이 폭넓게 확대되고 있으며, 그 대표적인 예로는 주문형 오디오(AOD : audio on demand), 주문형 비디오(VOD : video on demand), 원격 교육, 원격 회의, 원격 진료, 데이터 저장, 그리고 디지털 도서관(DL : digital library)과 같은 서비스가 존재한다.

멀티미디어 데이터의 경우 영상 디지털화는 연속되는 프레임을 생성하고, 오디오 디지털화는 연속되는 샘플을 생성한다. 그리고 이렇게 연속적으로 기록된 일련의 영상 프레임 또는 오디오 샘플은 스트림(stream)을 형성한다. 이 스트림은 적시에 연속적으로 표현되는 경우에 의미가 있기 때문에 각 미디어 스트림의 기록 및 재생은 반드시 실시간 속도를 유지하며 처리되어져야 한다. 게다가 원격 진료 및 원격 회의와 같은 응용의 경우, 멀티미디어 데이터의 접근을 요구하는 트랜잭션의 우선순위에 따라 서비스의 순서를 다르게 적용해야 하는 경우가 발생한다. 즉, 각 트랜잭션이 마감시간과 같은 실시간 제약사항을 갖는 경우로 만약, 트랜잭션이 제약사항을 만족하지 못하는 경우 트랜잭션 결과의 가치가 다소 떨어질 수 있고(소프트 실시간 트랜잭션), 어떤 경우에는 결과가 무의미해지고, 이로 인한 피해 및 금전적인 손해를 가져올 수도 있다(펌 실시간 트랜잭션). 뿐만 아니라 원격 진료 서비스에서는 최악의 경우, 치명적인 손실 및 큰 재앙을 야기할 수도 있다(하드 실시간 트랜잭션). 따라서 실시간 특성을 갖는 멀티미디어 서비스의 시간 제약성 만족을 위한 효율적인 버퍼 교체 기법이 필요하다. 이에 본 논문에서는 현재까지 연구되어진 다양한 버퍼 교체 기법들을 분석하여 실시간 멀티미디어 데이터베이스 시스템(MMDBS : multimedia database system)을 위한 효율적인 버퍼 교체 기법을 제안한다.

먼저 멀티미디어 데이터를 위한 기존의 버퍼 교체 기법들을 분석해 보면 다음과 같다. 첫째, 최근에 자주 이용되는 인기 있는 오디오 또는 비디오 데이터의 접근 확률은 시간에 비례하여 증가하는 특성을 갖는다. 반면에, 인기가 떨어지는 미디어 데이터는 접근 확률이 급격하게 낮아지는 특성을 갖는다. 이와 같은 특성을 멀티미디어 데이터의 접근 확률 가변 특성이라 한다. 예를 들어, AOD 또는 VOD 서비스의 경우 저장된 미디어의 인기도에 따라 접근 빈도가 좌우되므로 접근 확률 가변 특성이 반드시 고려되어져야 한다.

둘째, 멀티미디어 데이터에 대한 사용자 요구는 대부분 미디어 스트림의 처음부터 순차적으로 접근되기를 바라는 특성을 갖는다. 왜냐하면 미디어의 편집을 담당하는 사용자를 제외하면, 대부분의 일반 사용자는 오디오 또는 비디오 데이터를 처음 시작 부분부터 듣거나 보기를 원하기 때문이다. 이와 같은 멀티미디어 데이터의 특성은 실시간 데이터베이스 시스템의 실시간 특성과 결합되어져야 한다. 즉, 전체 트랜잭션의 마감시간 초과 비율을 최소화하도록 항상 높은 우선순위를 갖는 트랜잭션의 선행 처리가 보장되어야 한다. 결국, 실시간 멀티미디어 데이터베이스 시스템을 위한 버퍼 교체 기법은 실시간 특성과 멀티미디어 데이터의 특성 모두를 동시에 고려해 주어야 한다.

이와 같은 멀티미디어 데이터의 특성 및 실시간 요구사항을 만족시키기 위해 제안하는 버퍼 교체 기법에서는 미디어 데이터를 접근하는 트랜잭션의 우선순위에 따라 버퍼를 n개의 우선순위 레벨로 구분하여 관리한다. 그리고 각 우선순위 레벨은 처음 참조되는 미디어 스트림을 위한 콜드 셋 영역과 재 참조된 미디어 스트림을 위한 핫 셋 영역으로 구분되어 전체적으로 총 2n개의 리스트를 갖는다. 콜드 셋 영역(1 ~ n 리스트)에 위치한 각 미디어는 미디어 스트림의 시작과 끝 부분의 참조 정보(timestamp)를 갖는다. 그 이유는 버퍼 교체 시, 동일 리스트에 위치한 미디어 데이터 중에서 크기(미디어의 시작 부분과 끝 부분의 참조 간격)가 가장 큰 미디어를 먼저 교체시키기 위함이다. 이로 인해 교체되는 미디어 개수를 최소화시킬 수 있고, 또한 보다 많은 수의 새로운 미디어를 버퍼에 유지시킬 수 있다. 그러나 핫 셋 영역에서의 교체 기준은 다르게 적용하고 있다. 왜냐하면

수많은 멀티미디어 데이터 중에서 최소한 한 번 이상의 재 참조가 이루어진 미디어이므로 단순히 미디어의 크기보다는 재 참조 간격을 기준으로 교체 대상을 선정하는 것이 더 효율적일 것이기 때문이다. 따라서 핫 셋 영역(n+1 ~ 2n 리스트)에 위치한 각 미디어는 가장 최근의 참조 정보와 바로 그 이전 참조 정보를 유지하도록 하였다. 핫 셋 영역에서 버퍼 교체 시, 동일 리스트에 위치한 미디어 데이터 중에서 재 참조 간격이 가장 큰 미디어 데이터를 우선적으로 교체시킨다. 이로 인해 다른 미디어 데이터에 비해 참조 빈도가 높은 미디어 데이터를 더 오랫동안 버퍼에 유지시킬 수 있게 한다. 그리고 교체 대상 선정 시 요청 트랜잭션의 우선순위보다 높은 우선순위 레벨은 대상에서 제외시킴으로써 우선순위가 높은 트랜잭션이 사용하는 또는 사용했던 미디어를 보호한다. 결국 우선순위가 높고 재 참조 확률이 높은 미디어 데이터를 오랫동안 버퍼에 유지시킴으로써, 버퍼 히트 비율을 향상시키고 마감시간 초과 비율을 감소시키는 효과를 가져온다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 버퍼 교체 기법들에 대해 살펴보고, 3장에서는 실시간 특성을 고려한 새로운 버퍼 교체 기법에 대해 자세하게 기술한다. 그리고 4장에서는 성능평가를 통해 제안하는 버퍼 교체 기법의 우수성을 입증하고, 마지막 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

기존 버퍼 교체 기법들은 일반 기법, 혼합 기법, 멀티미디어 응용 기법, 실시간 응용 기법으로 분류해 볼 수 있다. 먼저 일반 버퍼 교체 기법으로 FIFO (first-in first-out), LRU(least recently used), LFU(least frequency used), MRU(most recently used), LRU-k 기법 등이 있다[1~3]. 둘째, 혼합 버퍼 교체 기법은 일반 버퍼 교체 기법들을 상호 보완적으로 결합한 것으로 ABRN(adaptive buffer replacement using neural networks), UBM(unified buffer management) 기법 등이 연구되어졌다[4, 5]. 셋째, 멀티미디어 특성을 고려한 기법으로는 인터벌 캐싱(interval caching), BASIC, DISTANCE 등이 있으며[6~12], 마지막으로 실시간 트랜잭션의 우선순위를 고려한 기법으로는 Priority-LRU, Priority-Hints 등이 연구되어졌다[13, 14]. 위에서 언급한 여러 버퍼 교체 기법에 대한 특성 및 분석 내용을 정리하면 (표 1)과 같다.

표 1. 주요 버퍼 교체 기법에 대한 비교 분석

구 분	교체 방법	장·단점
LRU	가장 오랫동안 사용되지 않은 페이지 교체	구현 간단, 비편중 분포를 갖는 응용에 유용
LFU	과거 접근 기록을 바탕으로 접근 빈도가 가장 낮은 페이지 교체	편중된 접근 분포를 갖는 응용에 유용, 캐시 오염 문제를 발생
MRU	가장 최근에 접근된 페이지 교체	멀티미디어 서비스 응용에 유용
LRU-k	최근 k개의 참조 정보를 바탕으로 참조 밀도를 유추하여 참조 간격이 가장 큰 페이지 교체	편중된 데이터 분포에서 LRU가 갖는 단점을 LFU의 특성으로 보완
ABRN	신경망을 이용하여 편중 데이터와 비편중 데이터를 구분하여 각각 LFU, LRU 기법 적용	비편중/편중 분포를 갖는 일반 응용에 적합
UBM	검출기를 통해 순차, 순환, 기타 참조 그룹으로 분할하여 각각 MRU, 가장 긴 주기 페이지 교체, 그리고 LRU, LFU, LRU-k 중에 한 가지 기법 적용	비편중/편중 분포를 갖는 일반 응용에 적합
Interval Caching	선행 스트림과 후행 스트림간의 간격이 가장 긴 스트림 교체	멀티미디어 서비스 응용에 유용
Priority -LRU	우선순위 레벨별로 구분하여 각 레벨에 LRU를 적용한 결과 셋 중에서 가장 오래된 페이지부터 교체	실시간 응용에 유용
Priority -Hints	질의 분석을 통해 일반, 선호 페이지 그룹으로 구분하여 각각 전역 LRU, 레벨별로 지역 MRU 적용	실시간 응용에 유용 Priority-LRU보다 우수

위에서 언급한 내용 중에서 인터벌 캐싱 방법과 실시간 특성을 고려한 방법들에 대해 보다 자세하게 살펴보면 다음과 같다.

인터벌 캐싱은 AOD, VOD 서비스처럼 하나의 미디어 스트림이 어떤 시간 간격을 두고 연속해서 읽혀질 수 있는 특성을 이용한 방법이다. 즉, 선행 서비스

요구를 위해 버퍼에 읽어 들인 미디어 스트림을 버퍼에 그대로 유지시켜 후행 요구를 위해 사용될 수 있도록 하는 것이다. 이 방법에서는 연속 스트림간의 간격(interval)이 가장 큰 스트림을 우선적으로 교체한다. 왜냐하면 한정된 버퍼 공간에 보다 많은 수의 연속 스트림을 유지시킴으로써 버퍼 히트 비율을 높일 수 있기 때문이다. 인터벌 캐싱 방법의 장점을 요약하면 다음과 같다. 첫째, 후행 스트림은 디스크 입출력 없이 버퍼상의 선행 스트림을 이용하여 빠르게 서비스 될 수 있다. 둘째, 디스크 입출력이 감소됨으로써 다른 여러 서비스 요청을 추가적으로 처리할 수 있다. 따라서 전체적인 서비스 품질 및 시스템 효율을 향상시키는 효과를 갖는다.

실시간 데이터베이스에서는 높은 우선순위를 갖는 트랜잭션의 선행처리가 보장되어야 한다. 따라서 실시간 버퍼 교체 기법의 경우, 트랜잭션의 우선순위에 따라 사용하는 데이터를 차등을 두어 관리하고 교체해야 할 필요가 있다. 이와 같은 실시간 특성을 반영하는 버퍼 교체 기법으로는 Priority-LRU와 Priority-Hints 기법이 있다. Priority-Hints 기법에서 수행한 성능 평가 결과 Priority-LRU 보다 더 좋은 성능을 보이고 있음을 알 수 있다. 그런데 아직까지 Priority-Hints 기법을 능가하는 실시간 버퍼 교체 기법이 제안되어진 바가 없으므로 이에 대한 연구는 중요한 의미를 갖는다고 볼 수 있다. Priority-LRU와 Priority-Hints 기법에 대해 자세하게 살펴보면 다음과 같다.

먼저, Priority-LRU 기법은 N개의 우선순위 레벨별로 LRU 큐를 별도로 관리한다. 그리고 버퍼 교체가 필요한 경우 할당을 요청하는 트랜잭션의 우선순위보다 낮은 레벨에서만 교체 대상을 찾는다. 버퍼 교체를 위한 처리 절차는 다음과 같다. 먼저 버퍼 교체를 야기한 트랜잭션보다 높은 우선순위 레벨은 교체 대상에서 제외시킨다. 그리고 나머지 우선순위 레벨에서 각각 LRU 기법을 적용하여 가장 오래된 페이지를 하나씩 선정한다. 이렇게 선정된 페이지들은 경계값(threshold)을 기준으로 오래된 페이지와 최근 페이지로 구분된다. 교체 대상은 오래된 페이지들 중에서 가장 우선순위가 낮은 레벨에 속한 페이지가 된다. 모두 최근 페이지인 경우에는 그 중에서 가장 우선순위가 낮은 페이지가 교체 대상으로 선정된다. 만약, 아래 (그림 1)의 예와 같이 중간(Mid) 우선순위

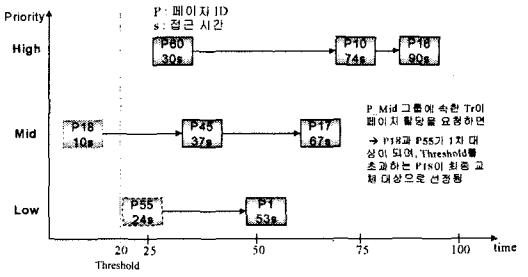


그림 1. Priority-LRU 기법의 페이지 교체 정책

를 갖는 오래된 페이지와 낮은(Low) 우선순위를 갖는 최근 페이지가 교체 대상 후보인 경우에는 전자를 희생페이지로 결정한다. 왜냐하면 우선순위가 높더라도 경계값 이전에 참조된 페이지는 오래된 것이며, 상대적으로 가치가 떨어지기 때문이다. 따라서 응용분야에 따라 적절하게 경계값을 설정하는 것이 중요한 문제라 할 수 있다.

다음으로 Priority-Hints 기법에서는 질의 분석을 통해 순차 참조되는 페이지를 관리하는 일반 페이지 그룹(normal pages group)과 순환 참조되는 페이지를 관리하는 선호 페이지 그룹(favored pages group)을 갖는다. 일반 페이지 그룹의 페이지들은 재 참조되지 않으므로 해제(unfix) 시 자유 리스트(free list)에 추가되어 통합 관리된다. 반면에 선호 페이지 그룹의 페이지들은 재 참조되어야 하므로 해당 우선순위 리스트에서 별도로 관리된다. 교체 대상을 선정하는 방법은 다음과 같다. 먼저 자유 리스트에서 우선순위를 무시한 전역(global) LRU 기법이 적용된다. 자유 리스트의 모든 페이지들은 더 이상 참조되지 않을 것이므로 우선순위를 고려하지 않는 것이다. 만약 자유 리스트에서 필요한 버퍼 공간을 모두 할당받지 못할 경우에는 계속해서 선호 페이지 그룹을 대상으로 교체 대상 선정 작업을 수행한다. 이때 할당을 요청한 트랜잭션의 우선순위보다 더 높은 우선순위 리스트는 교체 대상에서 제외된다. 희생 페이지는 가장 낮은 우선순위 레벨에서 지역(local) MRU 기법에 따라 선정된다. 만약 자신의 우선순위 보다 낮은 우선순위 리스트에서 희생 페이지를 찾지 못하는 경우에는 자신의 우선순위와 동일한 리스트까지 탐색을 수행하여 가장 최근에 사용된 페이지를 교체 대상으로 선정한다. (그림 2)의 예와 같이 중간 우선순위를 갖는 트랜잭션이 선호 페이지 그룹에 대한 페이지 교체를 야기하는 경우 우선순위가 가장 낮은 해제

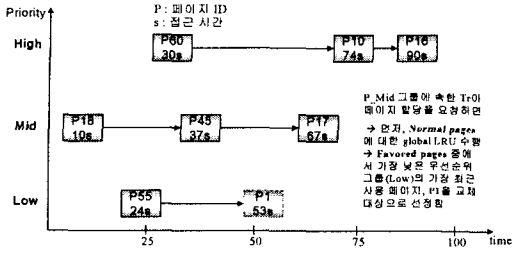


그림 2. Priority-Hints 기법의 페이지 교체 정책

리스트({P1, P55})를 대상으로 MRU를 적용하여 페이지 "P1"을 선정하게 된다.

3. 제안하는 실시간 버퍼 교체 기법

본 장에서는 실시간 멀티미디어 데이터베이스 시스템을 위한 효율적인 버퍼 교체 기법을 제시하고자 한다. 제안하는 기법은 미디어 스트림에서 가장 최근에 참조된 부분을 희생시키는 MRU 기법의 특징을 포함하고 있으며, 선행 스트림의 버퍼 캐싱을 통해 후행 스트림의 디스크 입출력을 방지하는 인터벌 캐싱 기법을 이용한다. 또한 특정 미디어의 접근 분포가 시시각각 변화되는 특성을 고려하기 위해 각 미디어마다 최근 2개의 참조 정보를 유지하도록 하여 이 참조 간격이 가장 큰 미디어를 교체 대상이 되게 한다. 물론 처음 참조된 콜드 셋의 미디어는 하나의 참조 정보만을 갖는다. 그러나 이 경우에는 미디어의 시작 부분과 끝 부분에 대한 참조 정보를 유지하도록 하여 가장 큰 미디어를 식별할 수 있도록 한다. 이를 통해 가장 큰 크기를 갖는 미디어를 우선적으로 교체시킴으로써 제한된 버퍼 공간 내에 보다 많은 수의 미디어를 유지시킬 수 있게 된다. 게다가 제안하는 기법은 실시간 시스템의 가장 중요한 특징이라 할 수 있는 우선순위를 고려하기 위해 여러 우선순위 레벨별로 미디어 데이터를 별도로 관리한다. 따라서 높은 우선순위를 갖는 트랜잭션에 의해 사용된 미디어를 더 오랫동안 버퍼에 유지시킬 수 있게 한다. 제안하는 실시간 버퍼 교체 기법의 특징을 정리하면 다음과 같다.

- 실시간 특성을 고려하기 위해 여러 단계의 우선순위 레벨로 구분한다. 그리고 각 레벨별로 처음 참조되는 미디어를 위한 콜드 셋 영역과 재 참조된 미디어를 위한 핫 셋 영역으로 이분한다.
- 교체 대상 선정 시, 버퍼 교체를 요구하는 트랜잭션의 우선순위보다 높은 우선순위 레벨은 교체 대상에서 제외된다. 그리고 나머지 우선순위 레벨 중에서 가장 낮은 우선순위 레벨의 콜드 셋 영역에서부터 교체 대상을 찾는다.
- 콜드 셋 영역에서는 가장 큰 미디어부터, 그리고 핫 셋 영역에서는 가장 긴 참조 간격을 갖는 미디어부터 교체한다.
- 미디어는 처음 부분부터 순차적으로 접근되는 특성을 갖기 때문에 교체 대상으로 선정된 특정 미디어는 내부적으로 가장 최근에 접근된 페이지부터 교체한다.

3.1 버퍼 구조

버퍼 영역은 (그림 3)과 같이 n개의 우선순위 레벨을 두는 경우 총 2n개의 리스트로 분할된다. 각 우선순위 레벨별로 처음 참조되는 미디어는 콜드 셋, 그리고 재 참조된 미디어는 핫 셋에서 관리한다. 즉, 버퍼에 존재하지 않는 미디어를 접근하는 경우에는 트랜잭션의 우선순위에 따라 해당 우선순위 레벨의 콜드 셋에 미디어를 추가한다. 그리고 해당 레벨의 우선순위보다 m 만큼 높은 트랜잭션에 의해서 동일한 미디어가 재 참조되는 경우, 해당 미디어는 현재 리스트(k)에서 k+(n+m) 리스트로 이동되어 관리된다. 예를 들어 설명하면 다음과 같다.

ⓐ 먼저, (그림 3)의 n-3 리스트(k=n-3)에 있는 미디어가 가장 높은 우선순위 레벨(P_n)을 갖는 트랜잭션에 의해 재 참조되는 경우(m=3), 새롭게 이동될 리스트 번호는 (n-3)+(n+3)이므로 2n 리스트가 된다. 즉, 우선순위는 3 레벨만큼 높아지며, 재 참조에 의해 콜드 셋의 미디어는 핫 셋으로 이동(n만큼 이동)되어진 것이다. 여기에서 n은 우선순위 레벨 수이다.

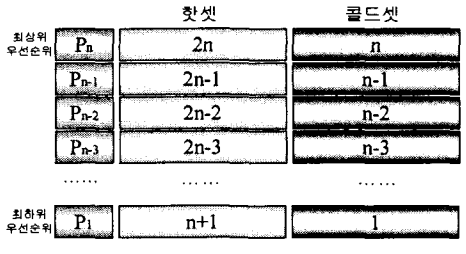


그림 3. 제안하는 버퍼 구조

ⓑ 만약 동일 우선순위 또는 하위 우선순위 트랜잭션에 의해 재 참조되는 경우에는 동일 레벨의 핫 셋으로 이동된다. 즉, 위의 예의 경우 n 만큼을 더한 n-3+(n) = 2n-3 리스트로 이동된다.

ⓒ 핫 셋 미디어의 재 참조 시에는 단지 높은 우선순위 트랜잭션과의 우선순위 차이 값만큼을 더하여 새로운 리스트를 결정한다. 즉, (2n-3) 리스트에 존재하는 미디어를 가장 높은 우선순위 레벨(P_n)을 갖는 트랜잭션이 재 참조하는 경우, 우선순위 차이 값(P_n - P_{n-3})이 3이므로 (2n-3)+3 = 2n 리스트로 이동된다.

ⓓ 만약 동일 또는 하위 우선순위를 갖는 트랜잭션에 의해 재 참조되는 경우(우선순위 차이 값이 0보다 작거나 같은 경우)에는 미디어의 이동이 발생되지 않는다. 왜냐하면 미디어의 우선순위 결정은 해당 미디어를 접근한 최상위 트랜잭션의 우선순위에 의해 결정되기 때문이다.

위와 같은 버퍼 구조에서 리스트 번호는 콜드 셋의 최하위 레벨이 가장 작은 값(1)을 가지며, 핫 셋의 최상위 레벨이 가장 큰 값(2n)을 갖는다. 여기에서 리스트 번호가 큰 값일수록 우선순위가 높은 트랜잭션에 의해 참조된 것이며, 재 참조가 이루어진 미디어이다. 따라서 희생 미디어를 선정하는 기준은 재 참조가 되지 않은 콜드 셋의 최하위 레벨에서부터 시작되어 재 참조가 이루어진 핫 셋의 최상위 레벨까지 순차적으로 이루어지는 것이 가장 효율적이다. 제안하는 기법에서 희생 미디어를 선정하기 위한 방법을 정리하면 다음과 같다.

ⓐ 희생 미디어를 탐색하는 작업은 리스트 번호가 가장 작은 리스트 1부터 n까지 오름차순으로 순차적으로 진행한다. 그리고 대부분의 사용자들은 미디어의 중간 또는 끝 부분보다는 처음 시작되는 부분을 요구하기 때문에 미디어의 일부분만을 교체하여 버퍼 할당 요청을 충족시킬 수 있는 경우에는 해당 미디어에서 가장 최근에 사용된 부분부터 교체시킨다. 이렇게 함으로써 해당 미디어 시작 부분이 버퍼에 남게 되고, 추가적인 서비스 요청에 재 사용할 수 있다는 장점을 갖는다.

ⓑ 콜드 셋(리스트 1 ~ n)에서 희생 미디어를 선정하는 경우에는 가장 큰 크기를 갖는 미디어를 우선적으로 선정한다. 이를 통해, 한정된 버퍼 공간에 보다 많은 수의 미디어를 유지시킬 수 있으므로 버퍼 히트 비율을 높이는 효과를 얻게 된다.

ⓒ 핫 셋(리스트 n+1 ~ 2n)에서 희생 미디어를 선정하는 경우에는 가장 긴 참조 간격을 갖는 미디어를 우선적으로 선정한다. 이를 통해 재 참조가 빈번하게 발생하여 참조 간격이 작은 미디어는 오랫동안 버퍼에 유지되고, 반대로 참조 간격이 큰 미디어는 우선적으로 교체시킴으로써 버퍼 영역을 효율적으로 사용하게 된다.

이와 같은 미디어 선정 방법을 위해 버퍼에 저장되는 각 미디어마다 두 개의 참조 정보(r_1, r_2)를 갖는다. 먼저, 콜드 셋에 추가된 미디어는 미디어의 시작 부분과 끝 부분에 대한 참조 정보를 각각 r_1, r_2 에 유지한다. 그리고 재 참조되어 핫 셋에 추가된 미디어는 이전 참조 정보와 현재 참조 정보를 각각 r_1, r_2 에 유지한다. 여기에서 이전 참조 정보는 해당 미디어가 콜드 셋에 존재할 때 유지하고 있던 참조 정보 r_1 이므로 재 참조되어 핫 셋으로 이동되는 경우, 현재의 참조 정보만을 r_2 에 갱신시키고 r_1 은 그대로 유지시키면 된다.

3.2 허용 제어

새로운 트랜잭션(T_{new})이 생성되면 먼저 충분한 버퍼 공간이 있는지를 검사하게 된다. 버퍼 공간이 충분한 경우에는 버퍼를 할당받고 처리를 시작하게 된다. 그러나 버퍼 공간이 부족한 경우에는 현재 수행중인 트랜잭션 중에서 새 트랜잭션 보다 우선순위가 낮은 하위 트랜잭션들을 오름차순으로 중지시키고 반환 받은 버퍼 공간을 할당받는다. 만약 모든 하위 트랜잭션들을 중지시켜서 새 트랜잭션이 필요로 하는 버퍼 공간을 할당받을 수 없는 경우에는 시스템 밖에서 대기한다. 이와 같은 허용 제어 방법을 사용함으로써 중요하거나 긴급한 트랜잭션의 선행 처리를 가능하게 하여 전체적인 마감시간 초과 비율을 감소시킨다. 허용 제어에 대한 자세한 처리 과정은 다음과 같다.

ⓐ 현재 수행중인 트랜잭션 중에서 T_{new} 보다 낮은 하위 트랜잭션이 존재하는 경우, 이들의 우

선순위를 오름차순으로 정렬(P_{min}, P_{min+1}, P_{min+2}, ... , P_{new-1})한다. 여기에서 P_{min}은 수행중인 하위 트랜잭션 중에서 가장 낮은 우선순위를 갖는 트랜잭션이고, P_{new-1}은 가장 높은 (T_{new} 보다는 우선순위가 낮은) 우선순위를 갖는 트랜잭션이다. 단, 모든 하위 트랜잭션들이 사용하고 있는 버퍼 공간의 합이 T_{new}의 버퍼 요구량 보다 적은 경우에는 처리를 중단하고 T_{new}는 시스템 밖에서 대기하게 된다.

ⓑ 정렬된 결과를 바탕으로 최하위 트랜잭션(P_{min})부터 필요한 버퍼 공간을 확보할 수 있는 트랜잭션까지 오름차순으로 일괄 선택한다. 그리고 선택된 모든 트랜잭션들을 일괄적으로 중지시킨 후 반환 받은 버퍼 공간을 한꺼번에 할당받고 수행에 들어간다.

3.3 버퍼 할당 및 교체

어떤 미디어를 요청하는 트랜잭션은 먼저 해당 미디어가 버퍼 풀(buffer pool)에 존재하는지를 검사하게 된다. 해당 미디어가 버퍼 풀에 존재하는 경우(버퍼 히트인 경우)에는 요청 트랜잭션의 우선순위를 바탕으로 미디어의 리스트 위치를 결정하고, 재 참조 정보를 갱신시킨 후 사용하게 된다. 만약 버퍼 풀에 해당 미디어가 존재하지 않는 경우(버퍼 미스인 경우)에는 디스크로부터 읽어올 페이지를 저장하기 위한 버퍼 공간을 미리 할당받은 후 디스크 입출력을 수행하게 된다. 이때 버퍼 공간이 부족하면 버퍼 할당이 불가능하다. 따라서 버퍼 교체 정책에 따라 희생 미디어를 선정하여 교체하는 작업이 선행되어져야 한다. 이 작업을 통해 필요한 만큼의 버퍼 공간을 할당받게 되면 비로소 디스크 입출력이 시작된다.

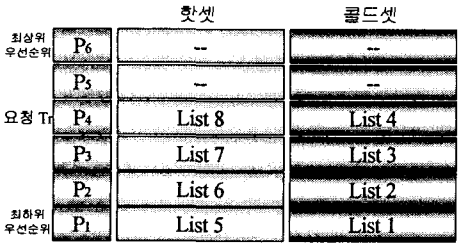


그림 4. 버퍼 교체 정책

버퍼 교체 대상을 찾는 작업은 제안된 버퍼 구조상의 리스트 순서에 따라 오름차순으로 진행된다. 이때 버퍼 교체를 야기한 트랜잭션의 우선순위를 기준으로 상위 레벨은 교체 대상에서 제외시킨다. 예를 들어 (그림 4)와 같이 우선순위 레벨이 P_4인 트랜잭션이 버퍼 교체를 요청하는 경우, 상위 레벨인 P_5와 P_6은 제외시키고 나머지 레벨을 대상으로 콜드 셋의 최하위 레벨부터 핫 셋의 동등 레벨까지 오름차순으로 교체 대상 검색 작업을 수행한다. (그림 4)를 중심으로 제안하는 버퍼 교체 정책을 설명하면 다음과 같다.

ⓐ 버퍼 교체를 요청하는 트랜잭션보다 높은 우선순위를 갖는 레벨(P_5, P_6)은 교체 대상에서 제외시키고, 나머지 우선순위 레벨(P_1 ~ P_4)의 콜드 셋과 핫 셋의 리스트(List 1 ~ 8)만을 대상으로 한다.

ⓑ 교체 대상을 선정하기 위한 검색 작업은 콜드 셋의 최하위 리스트(List 1)부터 실시한다. 만약 충분한 버퍼 공간을 확보하지 못하는 경우, 상위 리스트들을 순차적으로 검색해 나간다.

ⓒ 콜드 셋 영역(List 1, 2, 3, 4)에서 희생 미디어를 찾는 경우, 각 미디어의 시작 페이지와 끝 페이지에 대한 참조 정보를 바탕으로 이 간격(r_2 - r_1)이 가장 큰 미디어를 교체 대상으로 선정한다. 이 미디어를 교체시킴으로써 충분한 버퍼 공간이 확보되는 경우에는 선정된 미디어의 끝 부분(미디어에서 가장 최근에 참조된 페이지)부터 교체시키고 시작부분을 버퍼에 그대로 남겨둔다. 만약 선정된 미디어 전체를 교체하고도 요청된 버퍼 크기를 확보하지 못하는 경우에는 동일 리스트에서 두 번째로 참조 간격이 큰 미디어를 찾아서 앞에서와 마찬가지로 미디어의 끝 부분부터 교체시킨다. 현재 리스트의 모든 미디어를 교체시킨 후에도 버퍼 공간이 부족한 경우에는 'ⓑ'에서 언급한 바와 같이 상위 리스트로 이동하여 버퍼 교체 작업을 계속 수행한다.

ⓓ 콜드 셋 영역에서 충분한 버퍼 공간을 확보하지 못한 경우에는 핫 셋 영역(List 5, 6, 7, 8)에서 희생 미디어를 찾게 된다. 핫 셋 영역에서는 재 참조 간격이 가장 큰 미디어부터 교체한다. 그러나 콜드 셋 영역에서와 마찬가지로 두 개의 참조 정보 간격(r_2 - r_1)이 가장 큰 미디어를

교체하면 된다. 그리고 나머지 처리 방법 또한 콜드 셋에서와 마찬가지로 적용된다.

3.4 Priority-Hints 기법과의 비교분석

마지막으로, 본 논문에서 제안하는 기법과 Priority- Hints 기법에 대한 비교 분석 결과를 요약 정리하면 (표 2)와 같다.

4. 성능 평가

4.1 시뮬레이션 모델

관련 연구 부분에서 언급한 바와 같이 기존의 버

표 2. 제안하는 기법과 Priority-Hints 기법 비교

구 분	Priority-Hints 기법	제안하는 기법
전처리	루핑 연산과 스캐닝 연산을 구별하기 위해 질의 분석이 선행되어져야 한다.	질의 분석과 같은 전처리 과정이 필요 없다.
허용 제어 방법	자신보다 낮은 우선순위를 갖는 트랜잭션들을 대상으로 자신이 필요로 하는 버퍼 용량을 확보할 때까지 낮은 우선순위를 갖는 트랜잭션들을 중지시키는 작업을 수행한다.	Priority-Hints 기법의 경우, 새롭게 진입하는 트랜잭션보다 낮은 우선순위를 갖는 모든 트랜잭션들을 중지시킨 후에도 필요한 버퍼 용량을 확보하지 못한다면, 중지된 낮은 우선순위 트랜잭션들은 모두 불필요하게 중지된 결과가 된다. 따라서 제안하는 기법에서는 각 우선순위 리스트별로 버퍼 사용량을 유지하도록 하고 있다. 이를 바탕으로 새로운 트랜잭션의 허용 여부를 결정함으로써 낮은 우선순위 트랜잭션들이 불필요하게 중지되는 것을 방지한다.
버퍼 구조	전처리 과정(질의 분석)을 통해 얻어진 결과에 따라 한 번만 참조되는 페이지와 두 번 이상 참조되는 페이지를 구분하여 두 개의 공간에서 별도로 관리한다. 1. normal pages: 스캐닝 연산을 위한 공간 2. favored pages: 루핑 연산을 위한 공간	질의 분석과 같은 복잡한 전처리 과정을 필요로 하지 않는다. 단지 버퍼 미스/히트 여부에 따라 구분하여 관리하면 된다. 즉 처음 참조되는 데이터는 콜드 셋 영역에서 관리하고, 콜드 셋 또는 핫 셋 데이터가 재 참조되는 경우 핫 셋 영역의 해당 우선순위 레벨에서 관리한다. 1. 콜드 셋: 처음 참조된 데이터를 위한 공간 2. 핫 셋: 재 참조된 데이터를 위한 공간
우선순위 고려 여부	favored pages에 대해서만 우선순위 레벨별로 관리한다. 버퍼 교체 시, normal pages에 대해서는 우선순위와 상관없이 전역 LRU 기법을 적용하고, favored pages에 대해서는 요청 트랜잭션의 우선순위보다 낮은 레벨들만을 대상으로 한다.	콜드 셋과 핫 셋 모두 우선순위 레벨별로 관리한다. 버퍼 교체 시, 요청 트랜잭션의 우선순위보다 낮은 레벨들만을 대상으로 한다. 이 때, 핫 셋 영역보다는 한 번만 참조된 데이터를 관리하는 콜드 셋 영역이 우선적으로 교체 대상이 된다.
교체 대상 선정 방법	1. normal pages를 대상으로 전역 LRU 기법을 적용한다. 따라서 스캐닝 연산에 자주 이용되는 페이지는 버퍼 미스가 자주 발생될 수밖에 없다. 2. favored pages를 대상으로 요청 트랜잭션의 우선순위보다 낮은 레벨을 대상으로 희생 페이지를 선정한다. 이 때, 반복해서 재 사용되는 멀티미디어 데이터의 특성을 반영하기 위해 MRU 기법을 사용하므로 가장 낮은 우선순위 레벨에서 가장 최근에 참조된 페이지부터 교체한다.	1. 콜드 셋 영역에서 요청 트랜잭션의 우선순위보다 낮은 레벨들만을 대상으로 최하위 레벨부터 교체 대상 선정 작업을 시작한다. 이 때, 가장 큰 크기를 갖는 미디어 데이터부터 내부적으로 MRU 기법을 적용하여 교체한다. 따라서 제한된 공간에 보다 많은 수의 미디어 데이터를 유지할 수 있다. 2. 핫 셋 영역에서 요청 트랜잭션의 우선순위보다 낮은 레벨들만을 대상으로 최하위 레벨부터 교체 대상을 선정한다. 이 때, 재 참조 간격이 가장 큰 미디어 데이터부터 내부적으로 MRU 기법을 적용하여 교체한다. 따라서 참조 빈도가 높은 미디어 데이터를 오랫동안 버퍼에 유지시킬 수 있다.

퍼 교체 기법들은 크게 4가지 부류로 나누어 볼 수 있다. 첫째, LRU, LFU, MRU, LRU-k[1~3]와 같은 일반 기법, 둘째, ABRN, UBM[4, 5]과 같은 혼합 기법, 셋째, 인터벌 캐싱, BASIC, DISTANCE[6~12]와 같은 멀티미디어 응용 기법, 그리고 마지막으로 Priority-LRU, Priority-Hints[13, 14]와 같은 실시간 응용 기법이다. 본 논문에서 제안하는 기법은 실시간 응용 기법의 하나이므로 그 우수성을 입증하기 위해 Priority-LRU 그리고 Priority-Hints와 성능평가가 이루어져야 한다. 그러나 Priority-Hints 기법이 Priority-LRU 기법 보다 우수하다는 것이 [14]에 의해 입증되었기 때문에 Priority-Hints 기법을 대상으로 성능 평가를 실시하였다.

성능 평가에 사용된 시뮬레이션 모델은 [14]에서 적용한 모델을 기반으로 하였다. 실험에 사용된 컴퓨터는 인텔 펜티엄 III 프로세서, 1GHz, 256MB이며, OS는 윈도우즈 2000, 컴파일러는 비주얼 C++ 6을 사용하였다. 시뮬레이션에 사용된 데이터베이스에는 1000개의 페이지를 두고, 각 트랜잭션은 1~40개의 페이지를 접근하도록 하였다. 디스크 접근 시간과 메모리 접근 시간은 [15]에서 제시한 클럭 사이클을 바탕으로 다음과 같이 설정하였다. 디스크 접근과 메모리 접근을 위해 각각 6M 클럭 사이클, 40 클럭 사이클이 필요하다. 실험에 사용된 프로세서가 1GHz이므로 초(sec)로 환산하여 각각 6 ms와 40 ns를 얻을 수 있었다. 전체적인 페이지 접근 성향은 평균 16을 갖는 정규분포가 되도록 하였다. 그리고 트랜잭션의 마감 시간은 (식 1)과 같으며, SlackFactor는 2~7로 설정하여 각 트랜잭션마다 실행 시간의 1~6배의 여유시간을 갖도록 하였다. 그리고 기타 성능 평가에 사용된 파라미터는 (표 3)과 같다.

표 3. 시스템 파라미터

파라미터	내용	설정값
DBSize	데이터베이스 크기	1000 pages
AccessDisk	Disk 접근 시간	6 ms
AccessMem	Memory 접근 시간	40 ns
NumBuffers	버퍼 수	100
NumTrans	트랜잭션 수	5000
TransSize	트랜잭션당 page 수	1~40
PriorityLevel	우선순위 레벨	50% P_{Low}, 30% P_{Mid}, 20% P_{High}
Deadline	트랜잭션의 마감시한	도착시간+실행시간* SlackFactor
TransMix	트랜잭션 타입별 분포	50% looping, 50% scanning
ArrivalRate	트랜잭션 도착율	30 trans/sec

마감시간=도착시간+실행시간*SlackFactor .....(식 1)

시뮬레이션 모델의 전체적인 구조는 (그림 5)와 같다. 총 5,000개의 트랜잭션을 수행시켜 보았으며, 25%:75%, 50%:50%, 75%:25% 비율로 루핑(looping) 연산 대 스캐닝(scanning) 연산의 비율을 다르게 적용하여 실험하였다. 그리고 전체 트랜잭션을 대상으로 우선순위는 3개 레벨로 구분하여 P_{Low}, P_{Mid}, P_{High} 레벨에 대해 각각 50%, 30%, 20%의 비율로 할당되도록 하였다. [14] 기법의 경우 4개의 리스트를 유지하도록 하여 하나는 일반 페이지 그룹에 대한 전역 LRU 기법을 위해 사용하였으며, 나머지 3개 리스트는 선호 페이지 그룹에 대해 3개의 우선순위 레벨별로 각각 사용하였다. 그리고 제안하는 기법의 경우에는 6개의 리스트를 유지하도록 하여 콜드 셋과 핫 셋에서 우선순위 레벨별로 하나씩 사용하였다. 버퍼 교체 시 [14] 기법에서는 먼저 일반 페이지 그룹을 대상으로 전역 LRU 기법을 적용하였으며 버퍼 할당 용량이 부족한 경우 선호 페이지 그룹의 가장 낮은 우선순위 레벨(P_{Low})부터 버퍼 할당을 요청하는 트랜잭션의 우선순위 레벨까지 필요한 버퍼 용량을 할당받을 때까지 순차적으로 버퍼 교체 작업을 진행하도록 하였다. 반면에 제안하는 기법에서는 항상 콜드 셋의 최하위 레벨부터 버퍼 교체 대상 선정 작업을

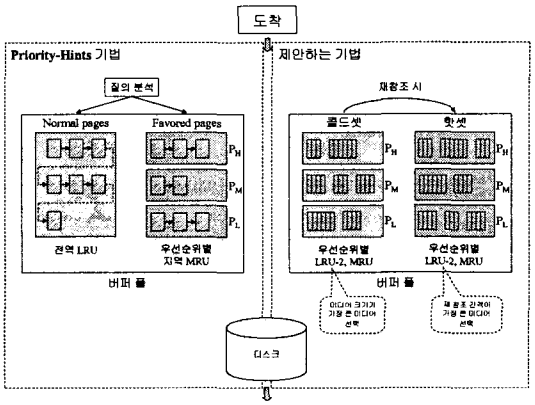


그림 5. 시뮬레이션 모델

시작하여 핫 셋의 해당 우선순위 레벨까지 순차적으로 진행하도록 하였다. 각 리스트별로 희생 데이터를 선정하는 기준은 항상 최대 참조 간격을 갖는 것을 우선으로 한다. 즉, 콜드 셋에서는 가장 큰 크기를 갖는 미디어 데이터를 우선적으로 교체한다. 그리고 핫 셋에서는 재 참조 간격이 가장 큰 미디어 데이터를 우선적으로 교체한다. 이를 위해 LRU-2 기법과 같이 데이터별로 최근 2개의 참조 정보를 유지하도록 하였다. 그리고 희생 데이터로 선택된 미디어는 MRU를 적용하여 최근 사용 부분부터 필요한 양만큼 교체된 후 나머지 부분은 그대로 버퍼에 유지하도록 하였다.

4.2 시뮬레이션 결과

[14] 기법과 제안하는 기법의 성능 비교 실험은 크게 2가지로 실시하였다. 하나는 트랜잭션 수행 개수 증가에 따른 버퍼 히트 비율을 비교하는 것이고, 다른 하나는 트랜잭션 수행 개수 증가에 따른 마감시간 초과 비율을 비교하는 것이다. 각 실험에서는 루핑 연산 대 스캐닝 연산의 비율을 각각 25%:75%, 50%:50%, 그리고 75%:25%로 구분하여 비교 실험하였다.

먼저, 첫 번째 실험에서는 트랜잭션 수행 개수를 증가시켜 가면서 버퍼 히트 비율의 변화를 그래프로 나타내 보았다. (그림 6) ~ (그림 8)은 루핑 연산 대 스캐닝 연산 비율을 각각 25%:75%, 50%:50%, 그리고 75%:25%로 설정하여 시뮬레이션을 수행한 결과 그래프이다. 세 가지 결과 모두에서 제안하는 기법은 연산 비율에 상관없이 거의 일정한 버퍼 히트 비율을 가짐을 알 수 있었다. 또한 [14] 기법과 비교해 볼 때, 루핑 연산이 적은 경우에는 매우 큰 차이를 보이

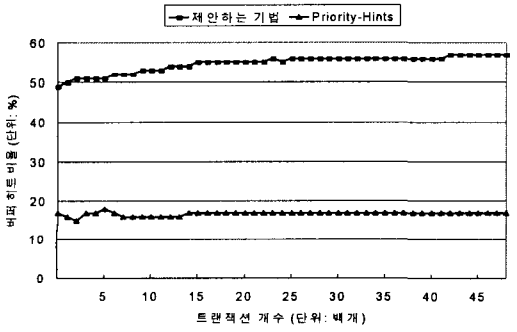


그림 6. 루핑 연산이 25%인 경우 버퍼 히트 비율의 변화

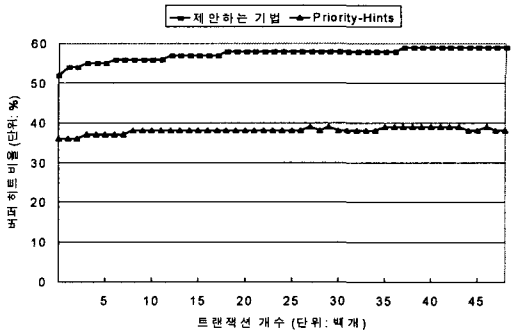


그림 7. 루핑 연산이 50%인 경우 버퍼 히트 비율의 변화

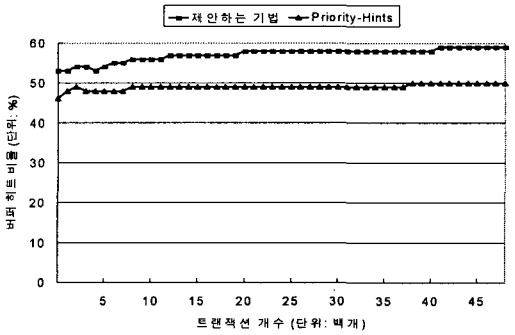


그림 8. 루핑 연산이 75%인 경우 버퍼 히트 비율의 변화

고 있으며, 루핑 연산이 많은 경우에도 적지 않은 차이를 보임을 알 수 있었다.

제안하는 기법의 경우, 연산 종류와 상관없이 한 번만 사용되는 데이터는 콜드 셋에서 우선순위 레벨별로 관리하여 버퍼 교체 시 우선적으로 교체 대상에 포함시키고, 이와 달리 재 사용되는 데이터는 핫 셋에서 우선순위 레벨별로 관리하여 버퍼 교체 시 교체 대상에서 멀어지도록 함으로써 재사용 율을 높이고 있기 때문이다. 반면, [14] 기법은 루핑 연산에 사용되는 데이터만을 선호 페이지 그룹에서 우선순위 레벨별로 관리하고, 나머지는 모두 우선순위와 상관없이 일반 페이지 그룹에서 관리하여 전역 LRU 기법을 적용하기 때문에 스캐닝 연산에 여러 번 재 사용되는 데이터라 하더라도 빈번하게 교체가 발생되는 단점을 갖는다. 게다가 이와 같은 스캐닝 연산 데이터는 우선순위를 고려하지 않으므로 마감시간 초과 비율을 증가시키는 요인으로 작용함을 알 수 있었다. 성능 평가 결과 평균 버퍼 히트 비율의 차이를 구해 보면 (그림 6)의 경우 38.1%, (그림 7)의 경우 19.4%, 그리고 (그림 8)의 경우 8.2%로 스캐닝 연산 비율에 따라 버퍼 히트 비율이 크게 차이가 남을 알 수 있었

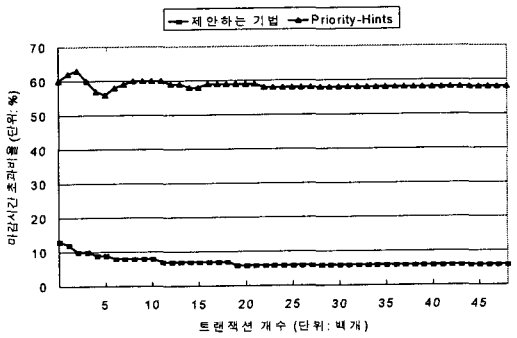


그림 9. 루핑 연산이 25%인 경우 마감시간 초과 비율의 변화

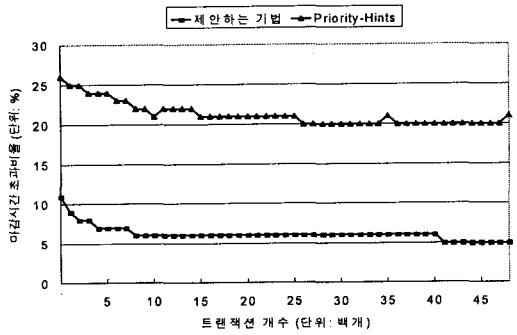


그림 10. 루핑 연산이 50%인 경우 마감시간 초과 비율의 변화

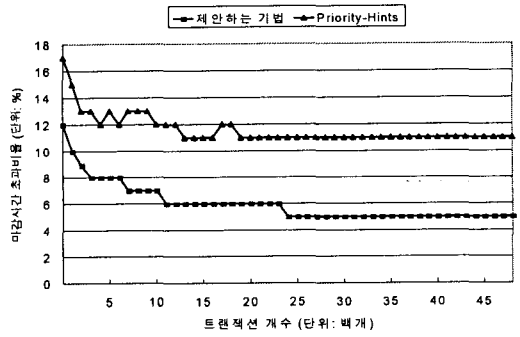


그림 11. 루핑 연산이 75%인 경우 마감시간 초과 비율의 변화

다. 즉, [14] 기법의 경우 스캐닝 연산의 비율 증가에 따라 버퍼 히트 비율이 크게 낮아짐을 알 수 있었다.

두 번째 실험에서는 트랜잭션 수행 개수를 증가시켜 가면서 트랜잭션의 마감시간 초과 비율의 변화를 그래프로 나타내 보았다. (그림 9) ~ (그림 11)는 루핑 연산 대 스캐닝 연산 비율을 각각 25%:75%, 50%:50%, 그리고 75%:25%로 설정하여 시뮬레이션을 수행한 결과 그래프이다. 첫 번째 실험 결과와 마찬가지로 세 가지 결과 모두에서 제안하는 기법이 [14] 기법보다 트랜잭션 마감시간 초과 비율이 낮음을 알 수 있었다. 성능 평가 결과 평균 마감시간 초과 비율의 차이를 구해보면 (그림 9)의 경우 51.7%, (그림 10)의 경우 15.0%, 그리고 (그림 11)의 경우 5.6%로 스캐닝 연산의 비율에 따라 트랜잭션 마감시간 초과 비율이 크게 차이가 남을 알 수 있었다.

이 실험 결과를 분석해 보면 다음과 같다. 먼저, 비교 대상인 [14] 기법에서는 루핑 연산의 비율이 낮은 경우, 우선순위 레벨별로 관리되는 데이터가 적어지고 대부분의 데이터는 우선순위와 상관없이 전역 LRU 기법에 의해 관리된다. 이로 인해, 실시간 버퍼 교체 기법에서 가져야할 우선순위를 적절하게 고려하지 못하므로 트랜잭션의 마감시간 초과 비율을 낮추지 못하는 단점을 갖는다. 반면에 루핑 연산의 비율이 높은 경우에는 대부분의 데이터가 우선순위 레벨별로 관리되며 버퍼 교체 대상에서 멀어진다. 왜냐하면 일반 페이지 그룹에 속한 스캐닝 데이터부터 버퍼 교체 대상이 되고 선호 페이지 그룹의 데이터는 2차적으로 교체 대상이 되기 때문이다. 따라서 루핑 연산의 비율이 높은 경우, 트랜잭션의 우선순위와 같은 실시간 특성이 적절하게 반영되고 있으며 재사용 데이터는 버퍼 교체 대상에서 가능한 멀어지기 때문에 트랜잭션의 마감시간 초과 비율이 낮아짐을 알 수 있다.

위의 두 실험 결과를 종합하여 비교 대상 기법과 제안하는 기법과의 버퍼 히트비율 차이 값 그리고 마감시간 초과비율 차이 값을 정리해 보면 (표 4)와 같다. (표 4)에서와 같이 Priority-Hints 기법은 루핑 연산 비율에 따라 성능이 크게 좌우된다. 왜냐하면

표 4. 시뮬레이션 결과 분석표

루핑 연산 대 스캐닝 연산 비율	버퍼히트 비 율 (%)	마감시간 초과비율 (%)	설 명
25 : 75	38.1	51.7	스캐닝 연산 비율이 큰 경우의 성능 차이
50 : 50	19.4	15.0	스캐닝 연산 비율과 루핑 연산 비율이 같은 경우의 성능 차이
75 : 25	8.2	5.6	루핑 연산 비율이 큰 경우의 성능 차이

트랜잭션 대부분이 루핑 연산인 경우에는 반복적으로 재 사용되는 데이터 비율이 높아지고 이 데이터들이 모두 선호 페이지 영역에서 관리되므로 일반 페이지 영역보다 더 늦게 교체 대상이 되어 버퍼 히트 비율 차이가 8.2% 정도이다. 또한 마감시간 초과 비율도 제안하는 기법에 비해 5.6% 만큼 높을 뿐이다. 그러나 루핑 연산 비율이 작은 경우에는 대부분의 데이터가 일반 페이지 영역에서 통합 관리되고, 우선순위와 무관하게 전역 LRU 기법에 의해 교체되기 때문에 버퍼 히트비율이 제안하는 기법과 큰 차이(38.1%)를 보임을 알 수 있다. 또한 우선순위에 따라 레벨별로 관리되는 페이지가 적기 때문에 마감시간 초과비율도 제안하는 기법에 비해 51.7%만큼 높은 결과를 보인다. 따라서 제안하는 기법은 평균적으로 15~20% 정도 더 좋은 효율을 보임을 알 수 있다. 이것은 제안하는 기법이 항상 우선순위를 고려한 버퍼 관리 정책을 사용하고 재 참조 정보를 유지하여 인기 있는 미디어 데이터를 오랫동안 버퍼에 유지하도록 하기 때문이다. 위의 실험 결과를 바탕으로 제안하는 기법과 비교 대상 기법을 비교 분석해 보면 다음과 같이 정리해볼 수 있다.

첫째, Priority-Hints 기법은 루핑 연산과 스캐닝 연산을 구별하기 위해 질의 분석이 반드시 선행되어져야 한다. 그러나 제안하는 기법은 단순히 재 참조되는 데이터를 구별하고, 참조 간격을 유지하면 된다. 둘째, 새롭게 시스템에 도착한 트랜잭션의 허용 제어 시, Priority-Hints 기법은 자신보다 낮은 우선순위를 갖는 트랜잭션들을 대상으로 자신이 필요로 하는 버퍼 용량을 확보할 때까지 반복해서 트랜잭션 중지 작업을 수행한다. 만약, 수행중인 트랜잭션 중에서 자신보다 낮은 우선순위를 갖는 모든 트랜잭션들을 중지시킨 후에도 필요한 버퍼 용량을 확보하지 못한다면 불필요한 중지를 수행한 것이라 할 수 있다. 이와 달리 제안하는 기법에서는 각 우선순위 레벨별로 버퍼 사용량을 유지하도록 하여 낮은 우선순위를 갖는 트랜잭션들의 불필요한 중지를 방지하고 있다. 셋째, Priority-Hints 기법은 루핑 연산에 사용되는 데이터만을 우선순위 레벨별로 관리한다. 그러나 제안하는 기법에서는 핫 셋과 콜드 셋 영역 모두에서 우선순위에 따라 레벨별로 관리한다. 또한 미디어의 크기 정보와 재 참조 간격 정보를 유지한다. 따라서 스캐닝 연산에 사용되는 데이터라 하더라도 재 참조 간격이 작은 인기 있는 미디어 데이터인 경우 버퍼에 오랫동안 유지되도록 한다.

5. 결 론

본 논문에서는 실시간 멀티미디어 서비스를 효율적으로 지원하기 위한 버퍼 교체 기법을 제안하였다. 이를 위해 다단계의 우선순위 레벨을 두고, 각 우선순위 레벨은 처음 참조되는 미디어 데이터를 위한 콜드 셋과 재 참조된 미디어 데이터를 위한 핫 셋 영역으로 구분된다. 각 미디어 데이터마다 두 개의 참조 정보를 유지하여 콜드 셋 영역에 위치한 각 미디어 데이터의 크기와 핫 셋 영역에 위치한 미디어 데이터의 재 참조 간격을 저장, 관리한다. 버퍼 교체 시, 버퍼 할당을 필요로 하는 트랜잭션의 우선순위에 따라 상위 우선순위 레벨은 교체 대상 선정에서 제외시키고, 나머지 우선순위 레벨 중에서 콜드 셋의 최하위 우선순위 리스트부터 핫 셋의 최상위 우선순위 리스트 순으로 교체 대상에 대한 검색을 수행한다. 그리고 각 리스트에서는 참조 간격이 가장 큰 미디어 데이터부터 교체시킴으로써 한정된 버퍼 공간에 보다 많은 수의 미디어 데이터를 유지시킬 수 있고, 재 참조된 핫 셋 영역의 미디어 데이터는 우선순위와 상관없이 콜드 셋 영역의 미디어 데이터보다 더 오랫동안 버퍼에 유지되므로 가장 우선순위가 높고, 가장 참조 빈도가 높은 인기 있는 미디어 데이터를 가장 오랫동안 버퍼에 유지시킬 수 있다. 이를 통해 동일한 미디어 데이터를 참조하는 여러 후행 스트림에 대한 불필요한 디스크 입출력을 방지하여 보다 빠른 서비스와 보다 많은 수의 서비스를 가능하게 한다. 향후 연구 계획은 보다 다양한 환경에서 제안된 기법을 비교 평가하여 그 우수성을 입증하는 것이다.

참 고 문 헌

[1] Wolfgang Effelsberg and Theo Harder, "Principles of Database Buffer Management," TODS 9(4), pp. 560-595, 1984.

[2] Hong-Tai Chou and David J. DeWitt, "An Evaluation of Buffer Management Strategies for Relational Database Systems," 11th International Conference on VLDB, August

21-23, 1985, Stockholm, Sweden, Proceedings. pp. 127-141. 1985.

[3] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," In ACM SIGMOD Conference, Washington DC, pp. 297-306, May 1993.

[4] 정광철, 박웅규, "ABRN: 주문형 멀티미디어 데이터베이스 서비스 시스템을 위한 버퍼 교체 알고리즘", 한국정보처리학회 논문지, 제3권 제7호, pp. 1669-1679, 1996.

[5] J. M. Kim, J. Choi, S. H. Noh, J. Kim, S. L. Min, Y. Cho, and C. S. Kim, "A Low-Overhead High-Performance Unified Buffer Management Scheme that Exploits Sequential and Looping References," In Proceedings of the 4th Symposium on Operating Systems Design and Implementation, Oct. 2000.

[6] Asit Dan and Dinkar Sitaram, "Buffer management policy for an on-demand video server," Technical Report RC 19347, IBM T.J Watson Research Center, 1994.

[7] Asit Dan, Daniel M, and R. Mukherjee, et. al, "Buffering and Caching in Large-Scale Video Servers," Proceedings of IEEE Data Engineering Conferences, pp. 217-224, 1995.

[8] Asit Danm and Dinker Sitatam, "Multimedia caching strategies for heterogeneous application and server environments," Technical Report RC 20670, IBM T.J Watson Research Center, 1996.

[9] Banu Ozden, Rajeev Rastogi, and Avi Silberschatz, "Buffer replacement algorithms for multimedia storage systems," In proceeding of Third IEEE International Conference on Multimedia Computing and Systems, pp. 172-180, Hiroshima, Japan, 1996.

[10] R. Alonso, D. Barbara, H., and Garcia-Molina, "Data Caching Issues in an Information Retrieval System," ACM Trans. Database Systems, Vol. 15, page(s) 359-384, 1990.

[11] D. Thie'baut, H. S. Stone, and J. L. Wolf, "Improving Disk Cache Hit-Ratios Through Cache Partitioning," IEEE Trans. Computers, Vol. 41, pp. 665-676, 1992.

[12] F. Moser, A. Kraib, and W. Klas, "L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS," Proceedings of the 21th VLDB Conference, Zurich, Switzerland, pp. 275-286, 1995.

[13] Michael J. Carey, Rajiv Jauhari, and Miron Livny, "Priority in DBMS Resource Scheduling," Proceedings of the Fifteenth International Conference on VLDB, August 22-25, 1989, Amsterdam, The Netherlands, pp. 397-410, 1989.

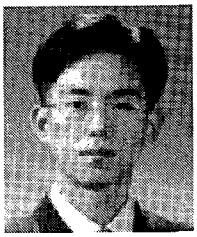
[14] Rajiv Jauhari, Michael J. Carey, and Miron Livny, "Priority-Hints: An Algorithm for Priority-Based Buffer Management," 16th International Conference on VLDB August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings, pp. 708-721, 1990.

[15] Kihong Kim, Sang K. Cha, and Keunjoo Kwon, "Optimizing Multidimensional Index Trees for Main Memory Access," ACM SIGMOD 2001 May 21-24, Santa Barbara, California USA, pp. 139-150, 2001.

신 재 룡

1996년 2월 충북대학교 정보통신공학과(공학사)
1998년 8월 충북대학교 정보통신공학과(공학석사)
2002년 8월 충북대학교 정보통신공학과(공학박사)

관심분야 : 데이터베이스 시스템, 실시간 시스템, 멀티미디어 데이터베이스 등



피 준 일

1999년 2월 충북대학교 컴퓨터공학과(공학사)
2001년 2월 충북대학교 정보통신공학과(공학석사)
2001년 3월~현재 충북대학교 정보통신공학과 박사과정 재학 중

관심분야 : 고차원 색인 구조, 데이터 베이스 시스템, 메모리 상주형 데이터 베이스 시스템, 저장 시스템, 실시간 시스템 등

유 재 수

1989년 2월 전북대학교 컴퓨터공학과(공학사)
1991년 2월 한국과학기술원 전산학과(공학석사)
1995년 2월 한국과학기술원 전산학과(공학박사)
1995년 3월~1996년 8월 목포대학교 전산통계학과 전임강사
1996년 9월~현재 충북대학교 전기전자및컴퓨터공학부, 컴퓨터정보통신연구소 부교수

관심분야 : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스, 분산 객체 컴퓨팅 등

조 기 형

1966년 2월 인하대학교 전기공학과(공학사)
1984년 8월 청주대학교 산업공학과(공학석사)
1992년 2월 경희대학교 전자공학과(공학박사)
1981년~1988년 충주공업전문대학 조교수
1988년~현재 충북대학교 전기전자및컴퓨터공학부, 컴퓨터정보통신연구소 교수

관심분야 : 데이터베이스, 소프트웨어 시스템 설계 및 구현, 컴퓨터 네트워크 설계 등