LRU-CFP : 데이타 방송 환경을 위한 클라이언트 캐쉬 관리 기법

권 혁 민†

요 약

데이타 방송 환경에서 서버는 방송 채널을 통하여 데이타베이스 내의 데이타들을 주기적으로 방송한다. 그리고 각 클라이언트가 어떤 데이타를 액세스하기 위해서는 방송 채널을 감시하여 해당 데이타가 방송되기를 기다려야 한다. 클라이언트 데이타 캐싱은 클라이언트가 액세스하려는 데이타가 방송되기를 기다리는 시간을 줄이기 위한 매우 효과적인 기술이다. 본 논문에서는 이 대기 시간을 줄이기 위하여 LRU-CFP로 명명된 새로운 클라이언트 캐쉬 관리 기법을 제안하고, 모의 실험을 통하여 새로이 제시된 기법의 성능을 평가한다. 성능 평가 결과에 의하면 LRU-CFP 기법은 LRU, GRAY, 그리고 CF 기법보다 평균 응답시간에 있어서 더 우수한 성능을 보인다.

LRU-CFP : Client Cache Management Scheme For Data Broadcasting Environments

HyeokMin Kwon†

ABSTRACT

In data broadcasting environments, the server periodically broadcasts data items in the database through the broadcast channel. When each client wants to access any data item, it should monitor the broadcast channel and wait for the desired item to arrive. Client data caching is a very effective technique for reducing the time spent waiting for the desired item to be broadcasted. This paper proposes a new client cache management scheme, named LRU-CFP, to reduce this waiting time and evaluates its performance on the basis of a simulation model. The performance results indicate that LRU-CFP scheme shows superior performances over LRU, GRAY and CF in the average response time.

키워드 : 캐쉬 관리(Cache Management), 버퍼 관리(Buffer Management), 데이타 방송(Data Broadcast)

1. 서 론

최근 들어 인터넷의 폭발적인 인기와 함께 전 세계에서는 온라인 데이타 서비스에 대한 요구가 기하급수적으로 증가하고 있고, 이에 따라 정보제공(information-feed) 응용들이 큰 관심을 끌고 있는 실정이다. 이런 종류의 응용들은 정보를 생성하여 관리하고 제공하는 단일 또는 몇몇의 서버와 정보를 검색하여 이를 이용하는 다수의 클라이언트로 구성되는 정보소비(information-consumption) 모델의 형태를 띠고 있다[15].

서버에서 클라이언트로 정보가 전달되는 방식에는 요청-응답(request-response) 기법과 데이타 방송(data broadcast) 기법이 있다[3, 6, 7]. 요청-응답 형태의 시스템에서 클라이언트는 자신이 액세스하려는 데이터를 서버에게 요청하고, 서버는 이에 대한 응답으로 해당 데이터를 클라이언트로 전송한다. 반면, 데이타 방송 형태의 시스템에서는 클라이언트로부터 명시적인 데이터의 전송 요청이 없어도 서버가 클라이언트에서 필요로 하는 데이타를 예상하여 전송한다. 요청-응답 시스템은 다수의 클라이언트로부터 일부 인기있는 데이타에 대한 중복 요청 및 서버의 중복 응답으로 인하여 서버 및 네트워크 자원을 심하게 낭비하는 경향이 있다. 이는 서버의 처리 능력이나 통신 대역폭이 제한적인 상황에서 시스템 자원의 과부하 상태를 야기하여, 서비스 응답시간이 심하게 지연되거나 서비스 자체가 불가능 상태에 빠질 수도 있다. 특히 정보제공 응용에서는 정보 데이타가 공공성을 띠는 경우가 많기 때문에 방대한 규모의 클라이언트가 동시에 접속을 시도하게 되어 이 문제가 더욱 심각해 진다.

데이타 방송 시스템에서 방송 서버는 다수의 클라이언트에게 특정 데이타들을 지속적으로 전파하고 각 클라이언트는 자신이 원하는 데이타가 방송채널에 나타나면 이를 검

※ 본 연구는 한국과학재단 목적기초연구(R05-2002-000-00901-0) 지원으로 수행되었음.
† 정 회 원 : 세명대학교 소프트웨어학과 교수
논문접수 : 2003년 2월 17일, 심사완료 : 2003년 6월 30일

색한다. 데이타 방송 시스템은 클라이언트의 수가 증가하더라도 시스템의 성능이 영향을 받지 않기 때문에 시스템의 확장성(scalability) 측면에서 매우 우수하다. 따라서 웹이나 정보제공 응용과 같이 방대한 규모의 클라이언트를 지원해야 하는 응용 분야에 매우 적합하다. 뿐만 아니라, 데이타 방송 기법은 클라이언트에서 서버로의 통신 대역폭이 제한적이어서 클라이언트가 자신의 요청을 서버로 전달하는데 비용이 많이 들거나 또는 불가능할 가능성이 많은 이동 컴퓨팅 환경(mobile computing environment)에도 매우 적합한 기술이다. 그러나 데이타 방송 시스템에서 클라이언트가 특정 데이타를 검색하기 위해서는 해당 데이타가 방송 채널에 나타나기를 기다려야 한다. 이 대기시간은 클라이언트의 성능에 큰 영향을 미치기 때문에, 이를 줄이기 위한 많은 연구들이 수행되었다[1-4, 10, 11, 15].

클라이언트 시스템에 적절한 캐쉬 관리 기법을 도입하면, 클라이언트는 자신의 버퍼에 검색하려는 데이타가 캐싱되어 있을 경우에는 방송 채널에서의 데이타 검색을 위한 대기시간을 피할 수 있다. 따라서 클라이언트가 데이타를 액세스하는데 소요되는 응답 시간의 성능을 상당히 향상시킬 수 있다. LRU[16], LRU-k[13], 그리고 2Q[9] 기법과 같이 요청-응답 시스템에 적용 가능한 캐쉬 관리 기법들이 제안되었지만, 이들은 특정 데이타들이 지속적으로 다수의 클라이언트에 동시에 전파되는 방송환경의 특수성을 고려하지 않았기 때문에 이들을 그대로 데이타 방송 기법을 채택한 정보 시스템에 적용시키기에는 무리가 있다.

데이타 방송 시스템에서는 기존의 요청-응답 시스템과 비교하여 다음과 같은 사항을 고려하여 캐쉬 관리 기법을 개발해야 한다. 첫째, 요청-응답 시스템에서는 모든 데이타에 대해 캐쉬미스(cache-miss)로 인해 발생되는 지연시간이 거의 일정하다. 그러므로 기존의 캐쉬 관리 기법에 대한 연구는 캐쉬히트율을 높이는 방향으로만 진행되어 왔다. 그러나 데이타 방송 시스템에서는 데이타가 언제 방송 채널에 나타나는가에 따라 캐쉬미스로 인하여 야기되는 지연시간이 각 데이타마다 상이하다는 특징이 있다. 둘째, 요청-응답 시스템에서는 클라이언트가 액세스할 것으로 예상되는 데이타를 서버에 요청하여 미리 캐싱해 두는 프리페치(prefetch) 기법을 적용하기가 쉽지 않다. 왜냐하면 그 예상 적중률이 낮을 경우에는 서버 및 네트워크 자원의 불필요한 낭비를 초래할 위험성이 있기 때문이다. 그러나 데이타 방송 시스템에서는 데이타들이 방송 채널을 통하여 주기적으로 방송되므로 특별히 서버 및 네트워크 자원에 별도의 부담을 지우지 않고도 원하는 데이타를 프리페치할 수 있다.

본 논문에서는 이와 같은 방송 환경의 특수성을 고려하여 데이타 방송 시스템에 적합한 새로운 캐쉬 관리 기법 LRU-CFP(least recently used with closest-first and prefetch)를 개발한다. LRU-CFP 기법의 기본 철학은 캐쉬히트율을 약간 희생하더라도 캐쉬미스로 인하여 야기되는 지연시간을 최소화하여 데이타를 액세스하는데 소요되는 평균 응답 시간의 성능을 향상시키자는 것이다. 이를 위해 LRU-CFP는 클라이언트의 캐쉬 용량이 허용하는 것보다 더 많은 수의 페이지들을 관리한다. 만일 클라이언트가 n개의 페이지를 캐싱할 수 있다면, LRU-CFP 기법은 가장 최근에 액세스한 $x \cdot n$(x는 1보다 큰 실수) 개의 페이지들을 관리한다. LRU-CFP 기법은 이 $x \cdot n$ 개의 페이지들을 액세스 확률이 높은 페이지로 간주하여 이들 중에서 방송될 때까지의 지연시간이 긴 페이지들을 우선하여 캐싱한다. 그리고 LRU-CFP에는 데이타의 방송이 진행되더라도 방송 지연시간이 긴 페이지들이 캐싱되도록 프리페치 기법이 통합되어 있다. 따라서 LRU-CFP 기법은 비교적 액세스 확률이 높으면서도 방송 지연시간이 긴 페이지들을 캐싱하게 된다. 그러므로 LRU-CFP 기법은 클라이언트가 특정 데이타를 액세스하는데 걸리는 평균 응답시간의 성능을 상당히 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 데이타 방송 모델과 기존의 연구들을 살펴보고, 3장에서는 본 논문이 새로이 제안하는 클라이언트 캐쉬 관리 기법을 설명한다. 그리고 4장에서 성능평가 모델을 기술하고 5장에서는 성능 평가 결과를 분석하고, 마지막으로 6장에서 결론을 맺는다.

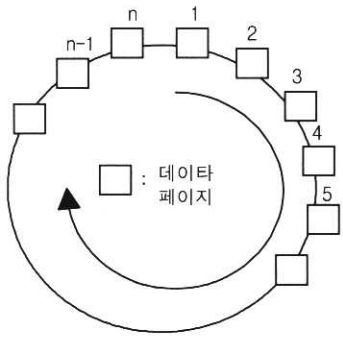
2. 관련 연구

이 장에서는 우선 본 논문이 기초하고 있는 데이타 방송 모델을 기술한다. 그리고 본 논문의 성능평가에서 사용된 기법들을 중심으로 기존에 제안된 캐쉬 관리 기법들을 살펴본다.

2.1 데이타 방송 모델

데이타 방송 시스템에서 방송 스케쥴링(broadcast scheduling) 기법은 방송할 데이타의 선정 및 방송 순서를 결정하는 문제를 다룬다. 방송 데이타를 스케쥴링할 때 데이타의 액세스 확률이 높을수록 빈번하게 방송하면, 전체 시스템의 성능은 향상될 것이다. [1, 3]의 연구는 데이타들을 주기적으로 전파하기 위하여 방송 채널을 저장 용량과 방송 빈도(broadcast frequency)가 다른 다수의 디스크로 구성하는 다중 방송 디스크(Multiple Broadcast Disks)로 모델링했다. 다중 방송 디스크 모델에서는 각 데이타의 액세스 확률을 파악하여 비슷한 확률을 가진 데이타들을 그룹화하여 다수의 그룹으로 나누고 각 그룹을 각각 다른 디스크에 저장한다. 그리고 디스크에 저장된 데이타들의 액세스 확률을 고려하여, 각 디스크의 상대적인 방송 빈도를 결정한다. 만일 데이타의 액세스 확률을 미리 파악할 수 없는 경우에는 모든 데이타들을 하나의 디스크에 저장할 수밖에 없는데,

이 경우에 모든 데이타의 방송 빈도는 동일하다. 이를 평형 방송 디스크(flat broadcast disk) 모델이라 한다. 평형 방송 디스크 모델에서는 (그림 1)과 같이 데이타베이스 내의 페이지들이 일정한 순서로 주기적으로 방송 채널을 통하여 전파된다.



(그림 1) 평형 방송 디스크 모델

본 논문은 클라이언트의 액세스 패턴을 미리 파악하기가 어렵거나 불가능하다고 가정하고 평형 방송 디스크 모델하에서 클라이언트 캐쉬 관리 기법을 개발한다. 그리고 본 논문은 [1, 2, 10, 11]의 연구와 같이 데이타 방송 및 캐쉬 관리의 단위와 클라이언트 액세스의 단위를 길이가 일정한 데이타 페이지라고 가정한다. 방송되는 데이타가 변경되는 환경에서 클라이언트가 데이타를 캐싱하면, 서버의 데이타와 클라이언트의 데이타 사이에 일관성 문제가 야기될 수 있다. 이런 환경에서 클라이언트 캐쉬의 일관성을 포함한 캐쉬 관리 기법에 대한 연구는 매우 흥미 있는 연구이지만 이는 본 논문의 범위를 벗어난다. 본 논문은 [1, 2, 10, 11]의 연구와 마찬가지로 방송 데이타는 읽기 전용 데이타로서 변경이 발생하지 않는다고 가정한다. 그렇지만 방송되는 데이타가 변경되는 환경에서 서버가 적절한 방식으로 변경된 데이타에 대한 정보를 방송하면, 이를 본 논문이 제안하는 캐쉬 관리 기법에 쉽게 반영할 수 있다.

2.2 클라이언트 캐쉬 관리 기법

클라이언트 데이타 캐싱은 요청-응답 방식을 채택한 클라이언트-서버 시스템의 성능 향상을 위해 소개되었다[5, 8]. 클라이언트 캐쉬 관리 기법 중 LRU(least recently used)는 [16] 구현이 간단하여 가장 많이 사용되는 기법이다. 이 기법은 캐쉬에 저장된 페이지들의 우선 순위를 관리하기 위하여 보통 LRU 체인을 유지한다. 어떤 페이지가 액세스되면 그 페이지는 체인의 맨 앞에 배치되고, 희생자가 필요할 때는 체인의 맨 뒤에 있는 페이지를 희생자로 선택하여 캐쉬에서 제거한다. 이 기법은 각 페이지의 최종 액세스 순서만을 고려하여 희생자를 선정하기 때문에 일단 한번 액세스된 페이지는 체인의 맨 뒤로 올 때까지 오랫동안 버퍼에 남아 있게 된다. 그러므로 LRU 기법은 임의로 우연히 액세스되는 페이지들이 많거나 데이타베이스 내의 어떤 영역을 임의로 스캔할 경우에는 캐쉬히트율이 낮은 단점이 있다.

[13]의 연구는 이 단점을 해결하기 위해 LRU-k 기법을 제안했다. LRU-k 기법은 어떤 페이지를 한번 액세스하였다고 해서 그 페이지에 우선권을 주는 것이 아니라, 각 페이지들의 마지막 k개의 액세스 시간을 고려하여 페이지의 우선 순위를 결정한다. 이를 위해 이 기법은 각 페이지들의 마지막 k개의 액세스 시간을 유지 관리한다. LRU-k 기법은 높은 캐쉬히트율을 보이는 장점이 있지만, 구현하는데 있어 높은 실행 부담을 보이는 단점이 있다. [9]의 연구는 LRU 기법과 유사한 실행 부담을 가지면서도, LRU-2에 필적하는 캐쉬히트율을 보일 수 있는 2Q(two queue) 기법을 제안했다. 2Q 기법은 두 개의 큐를 유지함에 의해 우연히 한번 액세스된 페이지들을 일찍 캐쉬에서 제거할 수 있다.

LRU, LRU-k, 그리고 2Q 기법은 과거의 액세스를 바탕으로, 이후에 가장 액세스되지 않을 것으로 예상되는 페이지를 희생자로 선정한다. 그러나 이들은 캐쉬미스로 인하여 야기되는 지연시간이 각 페이지마다 상이한 방송 환경의 특수성을 고려하지 않았다. 방송 환경의 특수성을 고려한 기법에는 CF(closest first), GRAY, PIX 및 LIX 기법들이 있다. CF 기법은[10, 11] 캐싱되어 있는 페이지 중에서 다음에 가장 먼저 방송될 페이지를 희생자로 선정하는 기법이다. CF 기법은 과거의 액세스 형태를 고려하지 않고 단순히 방송 순서만을 고려하여 희생자를 선택하기 때문에 자주 액세스되는 페이지나 그렇지 않은 페이지나 희생자로 선택될 확률이 동일하다.

[10, 11]의 연구는 CF(closest-first) 기법과 마킹 알고리즘(marking algorithm)으로 알려진 1-bit LRU 기법을 통합하고, 여기에 프리페치 기법을 적용하여 GRAY 기법을 개발했다. GRAY 기법은 데이타베이스의 각 페이지들이 흰색, 회색 또는 흑색의 상태 중에서 어떤 상태에 있는지에 대한 정보를 유지 관리한다. 흰색 페이지는 캐싱되어 있지 않은 페이지를 의미하며, 흑색 페이지는 버퍼에 캐싱되어 있는 페이지를 의미한다. 회색 페이지는 프리페치 대상이 되는 페이지이며 버퍼에 캐싱되어 있을 수도 있다. 흑색 페이지는 이미 캐싱되어 있기 때문에, 그리고 흰색 페이지는 자주 액세스되지 않기 때문에 프리페치 대상이 되지 않는다. GRAY 기법은 희생자가 필요할 경우에는 캐싱되어 있는 회색 페이지 중에서 가장 먼저 방송 채널에 나타나는 페이지를 희생자로 선정하여 버퍼에서 제거한다.

GRAY 기법에서 클라이언트가 어떤 페이지를 액세스하면 그 페이지는 현 상태에 관계없이 무조건 흑색의 상태가 되며 버퍼에 캐싱된다. 언젠가는 버퍼에 캐싱된 페이지는 모두 흑색의 상태가 되어, 더 이상 캐싱된 회색 페이지가 존재하지 않아 희생자 선정이 불가능하게 된다. 이 상태가 되면, GRAY 기법의 한 주기가 끝나게 된다. 주기의 끝에

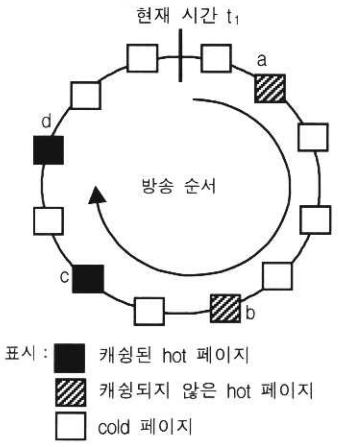
서 모든 흑색 페이지는 회색 페이지로 변경되며, 회색 페이지는 흰색 페이지로 변경되고 새로운 주기가 시작된다. 회색 페이지는 이전 주기에서 액세스된 페이지이므로 현 주기에서도 액세스될 확률이 높다고 볼 수 있다. 그러므로 방송 채널에 회색 페이지가 나타나고, 마침 그 페이지가 캐싱되어 있지 않으면 프리페치하여 버퍼에 캐싱한다. GRAY 기법에서 일단 한번 흑색으로 상태가 변경된 페이지는 현 주기가 끝날 때까지는 희생자로 선정되지 않는다. 따라서 임의로 우연히 액세스된 페이지들도 현 주기의 끝까지 오랫동안 버퍼에 캐싱되어 있어야 하며, 다음 주기에서는 프리페치 대상이 된다.

[1, 3]의 연구는 다중 방송 디스크 모델에서 캐쉬미스의 처리 비용에 근거한 PIX와 LIX 기법을 제안했다. PIX 기법은 각 페이지의 액세스 확률(p_i)과 그 페이지의 방송빈도(x_i)를 유지하여, p_i/x_i의 값이 가장 작은 페이지를 희생자로 선택한다. 이 기법은 클라이언트가 각 페이지의 액세스 확률을 미리 파악하고 있어야 구현이 가능하다는 단점이 있다. 그리고 희생자를 선정하기 위해, 캐싱되어 있는 모든 페이지의 p_i/x_i 값을 비교해야 하는 부담이 있다. 이 단점을 해결하기 위하여 제안된 LIX 기법은 클라이언트가 어떤 페이지를 액세스할 때마다 그 페이지의 추정(estimated) 액세스 확률(ep_i)을 계산한다. 그리고 캐싱되어 있는 페이지를 방송 디스크별로 구분하여 별도의 다수의 LRU 체인으로 구성하여 관리한다. 그리고 각 LRU 체인의 끝에 있는 페이지들 중에서 ep_i/x_i의 값이 가장 작은 페이지를 희생자로 선정한다. 이 기법은 방송 디스크의 수가 많을수록, PIX 기법과 유사해 지며, 방송 디스크의 수가 적어질수록 LRU 기법과 유사해지는 특징을 보인다. 만일 서버에서 클라이언트들의 액세스 정보를 취합하는 것이 불가능하여 방송 디스크의 수가 하나가 될 경우에는, LIX 기법은 LRU와 거의 동일한 기법이 된다.

3. 새로운 클라이언트 캐쉬 관리 기법 : LRU-CFP

이 장에서는 본 논문이 새로이 제시하는 LRU-CFP(least recently used with closest-first and prefetch) 기법에 대하여 기술한다. 캐쉬 관리 기법은 다수의 페이지들이 한정된 캐쉬 슬롯을 교대로 사용하도록 하기 위하여 적절한 희생자 선정 정책을 필요로 한다. 캐쉬 관리 기법 중에서 가장 많이 사용되는 LRU는 보통 LRU 형태의 큐(queue)를 유지하여 캐싱된 페이지들의 우선 순위를 관리한다. 희생자가 필요할 경우에는 LRU 큐의 맨 뒤에 있는 페이지를 희생자로 선정한다. LRU-CFP 기법도 LRU와 마찬가지로 LRU 큐를 유지하여 페이지들의 우선 순위를 관리한다. 본 논문은 알고리즘을 간편하게 설명하기 위하여 LRU 큐에서 관리되고 있는 페이지를 hot 페이지라 명명하고, 최근에 액세스되지 않아 큐에서 관리되고 있지 않은 페이지는 cold 페이지라 명명한다.

만일 클라이언트에 n개의 캐쉬 슬롯이 있다면, LRU 기법은 가장 최근에 액세스한 n개의 페이지를 큐에서 관리한다. 물론 이 n개의 페이지는 모두 클라이언트 버퍼에 캐싱되어 있다. 반면, LRU-CFP 기법은 가장 최근에 액세스한 $x \cdot n$(x는 1보다 큰 실수) 개의 페이지를 LRU 큐에서 관리한다. 이들 중에서 그 페이지가 방송될 때까지의 지연 시간이 가장 긴 n개의 페이지만이 캐싱되어 있고, 나머지 $(x-1) \cdot n$개의 페이지는 프리페치의 대상이 된다. 데이타가 방송됨에 따라 캐싱되어 있는 페이지와 프리페치 대상이 되는 페이지는 지속적으로 변경된다. 이에 대한 이해를 돕기 위해 (그림 2)를 살펴보자.



(그림 2) LRU-CFP 기법의 캐쉬 관리

(그림 2)는 클라이언트가 2개의 캐쉬 슬롯을 가지고 있고 4개의 hot 페이지를 관리하고 있다고 가정한다. 시간 t_1 시점에서 클라이언트는 방송 지연시간이 가장 긴 hot 페이지인 c와 d를 캐싱하고 있다. 시간이 경과하여 페이지 a가 방송 채널에 나타나면, 클라이언트는 캐싱된 페이지 중에서 가장 먼저 방송될 c를 캐쉬에서 제거하고 대신에 a를 프리페치하여 캐싱한다. 따라서 a가 방송되고 나면 클라이언트는 그 순간에 있어 방송 지연시간이 가장 긴 a와 d를 캐싱하게 된다. 마찬가지로 b가 방송되고 난 직후에는 d 대신에 b를 캐싱한다.

이와 같은 방식으로 LRU-CFP 기법은 항상 가장 최근에 액세스한 $x \cdot n$개의 hot 페이지 중에서 방송 지연시간이 가장 긴 n개의 페이지를 캐싱한다. LRU-CFP에서 x의 값은 hot 페이지의 수를 결정하게 되는데, 이 값을 크게 설정할수록 캐쉬히트율은 점점 낮아지고, 캐쉬미스시의 평균 지연시간은 짧아진다. 이와 같이 x의 값은 캐쉬히트율과 평균 지연시간사이에 절충(trade-off) 현상을 가져오기 때문에, 성능 튜닝을 위하여 적절하게 설정해야 한다. LRU-CFP 기법은 LRU에 비하여 x배의 큐 엔트리들을 관리한다. LRU-CFP

에서 증가된 만큼의 큐 엔트리에서는 페이지 식별자 정도만이 관리되지, 실제 페이지를 캐싱하고 있는 것이 아니므로 공간 부담이 LRU 기법에 비해 그다지 큰 것은 아니다.

본 논문은 LRU-CFP 기법의 희생자 선정 정책을 위하여 두 가지 종류의 희생자를 소개한다. 만일 클라이언트가 cold 페이지를 액세스한다고 가정해 보자. 그렇다면 그 페이지를 새로이 캐싱하기 위해 캐쉬 슬롯에서 제거되어야 하는 희생자가 필요하며, 큐의 엔트리 중에서 삭제되어야 하는 희생자가 필요하다. 본 논문은 전자를 슬롯 희생자라 명명하고 후자를 엔트리 희생자라 명명한다. 엔트리 희생자로 선택된 페이지는 cold 페이지로 격하되어 큐의 엔트리에서 삭제되며, 슬롯 희생자로 선정된 페이지는 단순히 자신의 캐쉬 슬롯만을 양보한다. 만일 엔트리 희생자로 선정된 페이지가 캐싱되어 있을 경우에는 별도의 슬롯 희생자는 필요하지 않다. LRU-CFP 기법에서 엔트리 희생자는 LRU 형태로 선정되고, 슬롯 희생자는 CF 형태로 선정된다. 즉, 엔트리 희생자로는 LRU 큐의 맨 뒤에 있는 페이지가 선정되어 큐 엔트리에서 삭제되며, 슬롯 희생자로는 캐싱되어 있는 페이지 중에서 가장 먼저 방송될 페이지가 선정되어 캐쉬 슬롯에서 제거된다.

LRU-CFP 기법은 LRU 큐를 다음과 같이 관리한다. 클라이언트가 이미 캐싱되어 있는 hot 페이지를 액세스하면, 그 페이지를 위한 큐 엔트리가 LRU 큐의 맨 앞으로 이동된다. 클라이언트가 캐싱되지 않은 hot 페이지를 액세스하기 위해서는 그 페이지가 방송 채널에 나타날 때까지 대기해야 한다. 그리고 슬롯 희생자를 선정하여 캐쉬에서 제거하고 대신 현재 액세스하려는 페이지를 캐싱한다. 그리고 그 페이지를 위한 큐 엔트리를 LRU 큐의 맨 앞으로 이동한다. 이 경우에는 별도의 엔트리 희생자는 필요하지 않다. 클라이언트가 cold 페이지를 액세스하려면, 슬롯 희생자를 선정하여 캐쉬에서 제거하고 엔트리 희생자를 선정하여 LRU 큐에서 삭제한다. 그리고 그 cold 페이지를 비워진 캐쉬 슬롯에 저장하고 그 페이지를 위한 큐 엔트리를 LRU 큐의 맨 앞에 추가한다. 클라이언트가 특정 페이지 p를 액세스할 경우에 LRU 큐를 관리하기 위한 자세한 과정이 (알고리즘 1)에 있다. 본 논문은 읽기 전용 데이타를 가정하고 있기 때문에 시스템이 초기 전이상태(initial transient state)를 벗어나 정상 상태에 도달하면 클라이언트의 캐쉬에는 빈 공간이 존재하지 않는다. 본 논문은 빈 캐쉬 슬롯이 존재하지 않는다고 가정하고 알고리즘을 기술한다.

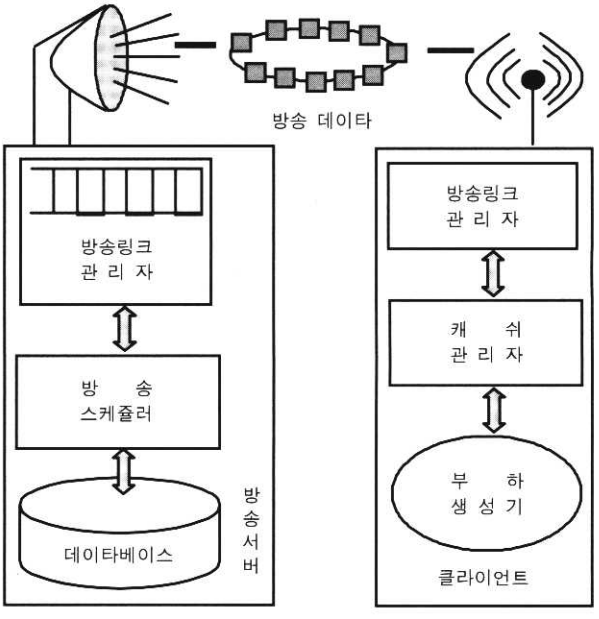
LRU-CFP 기법에서 캐싱되지 않은 hot 페이지는 프리페치의 대상이 된다. 따라서 방송 채널에 프리페치 대상이 되는 페이지가 나타나면, 캐싱되어 있는 페이지 중에서 다음에 가장 먼저 방송 채널에 나타날 페이지를 슬롯 희생자로 선정하여 캐쉬에서 제거하고 현재 방송되는 페이지를 캐쉬에 저장한다. 어떤 페이지가 프리페치될 경우에는 엔트리 희생자는 필요하지 않으며, LRU 큐의 LRU 체인이 조정되는 것도 아니다.

```
if p is a cached hot page then
    move p's entry to the head of LRU queue
else if p is a non-cached hot page then
    wait for p's arrival on broadcast channel
    find the closest-first cached page, cpi
    free the cache slot used by cpi
    put p into the freed cache slot
    move p's entry to the head of LRU queue
else  /* cold page access */
    wait for p's arrival on broadcast channel
    /* entry victim : tail entry of LRU queue */
    if the page of the entry victim is cached then
        free the cache slot used by the entry victim
    else
        find the closest-first cached page, cpi
        free the cache slot used by cpi
    end if
    put p into the freed cache slot
    delete the entry victim from LRU queue
    insert p's entry into the head of LRU queue
end if
```

(알고리즘 1) 클라이언트가 페이지 p를 액세스

4. 성능 평가 모델

이 절에서는 본 논문에서 제안된 캐쉬 관리 기법의 성능을 파악하기 위하여 성능 평가 모델을 제시한다. 성능의 비교 대상으로는 GRAY, CF, LRU 기법을 선정하였다. 성능 평가 모델은 [1, 3, 10]의 연구를 참고로 하고 있고, MCC에서 개발한 CSIM[14] 언어를 이용하여 구현하였다.



(그림 3) 성능 평가 모델

본 성능 평가 모델은 크게 방송 서버와 클라이언트로 구

성된다. 방송 환경의 특성상 한 클라이언트의 성능은 다른 클라이언트들의 존재 여부에 영향을 받지 않기 때문에 본 논문은 클라이언트의 수를 하나로 고정한다. 방송 서버의 방송 스케쥴러는 일정한 순서로 데이타베이스의 각 페이지들을 주기적으로 방송한다. 클라이언트의 응용 프로그램을 모델링한 부하 생성기(workload generator)는 각 실험의 부하 환경에 맞추어 액세스할 페이지를 선정하여 액세스한다. 해당 페이지가 자신의 지역 캐쉬에 있으면 그 데이타를 액세스하고 그렇지 않으면 방송 채널에 그 데이타가 나타날 때까지 대기했다가 액세스한다. 부하 생성기는 현재 요구한 페이지를 액세스하고 난 후에 새로운 페이지의 액세스를 요청할 수 있다. 클라이언트의 캐쉬 관리자는 성능 평가 대상이 되는 각 기법에 맞게 클라이언트의 캐쉬를 관리한다.

본 성능 평가에서 사용하는 시간 단위는 다른 연구들과 [1, 3, 10] 마찬가지로 tick이라는 논리적 시간 단위를 사용한다. tick은 하나의 데이타 페이지를 방송하는데 걸리는 시간을 의미한다. 본 논문에서 사용된 입력변수들은 <표 1>에 제시되어 있는데, 이 변수들의 설정 값은 대부분 [1, 3, 10]의 연구를 따르고 있다.

〈표 1〉 성능 평가 모델의 입력 변수

입 력 변 수	의 미	설 정
SerDbSize	데이타베이스의 크기	5000pages
CacheSize	클라이언트 캐쉬슬롯의 수	0~500pages
ThinkTime	클라이언트가 페이지를 액세스한 후에 다음 액세스 요청까지의 시간 간격	2ticks
AccRange	클라이언트가 액세스하는 페이지 범위	1000pages
RegionSize	zipf 분포에서 동일한 액세스 확률을 갖는 페이지의 수	50
NoiseProb	zipf 분포를 벗어날 확률	0~100%
θ	zipf 분포의 θ값	0.95
x	hot 페이지 수를 결정하는 변수	1.5

서버의 데이타베이스는 SerDbSize 수의 페이지로 구성되며, 서버는 이 페이지들을 주기적으로 방송한다. 클라이언트는 이 페이지들을 모두 액세스하는 하는 것이 아니라, 전체 데이타베이스의 일정 범위만을 액세스하는데, 이 일정 범위를 AccRange로 정의된다. 즉 클라이언트는 1~AccRange 사이의 페이지만을 액세스한다. 본 논문은 불균등한(non-uniform) 액세스 형태를 모델링하기 위해 많이 사용되는 zipf 분포 부하 모델[1, 9, 10]에서 각 기법의 성능을 비교한다. 본 논문의 zipf 부하 모델에서는 클라이언트가 액세스하는 AccRange내의 페이지를 RegionSize 만큼의 페이지로 나누어 다수의 영역으로 구분하는데, 각 영역은 서로 겹치지는 않는다. 따라서 <표 1>의 AccRange와 RegionSize를 살펴보면, AccRange 내의 페이지들이 20개의 영역으로 구분됨을 알 수 있다. 각 영역은 1부터 시작하여 20까지의 일련번호를 갖는 영역 번호로서 구분하는데, 이 영역에 zipf 분포가 적용된다. 즉, 각 영역의 액세스 확률은 $1/r^{\theta}$ (r은 영역 번호를 의미)에 비례한다. zipf 부하 모델에서 클라이언트가 어떤 페이지를 액세스할 것인가를 결정하기 위해서는 우선 zipf 분포에 맞게 액세스할 영역을 먼저 선정한다. 그리고 나서, 그 영역에서 어떤 페이지를 액세스할 것인가는 균등(uniform) 분포에 따라 선정한다.

CacheSize는 클라이언트가 캐싱할 수 있는 페이지의 수, 즉 캐쉬 슬롯의 수를 의미한다. x는 LRU-CFP 기법의 LRU 큐에서 관리하는 hot 페이지의 수를 결정하는 변수로서 본 실험에서는 이 값을 1.5로 설정했다. ThinkTime은 클라이언트가 데이타를 액세스하고 나서 다음 액세스를 요청하기까지의 시간을 의미하는데, 클라이언트가 데이타를 처리하는데 필요한 시간을 의미한다. NoiseProb는 클라이언트의 액세스 변화를 모델링하기 위한 것으로 클라이언트의 액세스가 zipf 부하를 따르지 않을 확률을 의미한다. 클라이언트의 각 액세스마다 Noise 발생 여부가 조사된다. 만일 Noise가 발생하면, 해당 액세스는 zipf 부하에 따라 페이지를 선정하지 않고, AccRange내의 페이지 중에서 임의의 페이지가 선정된다. zipf 부하에서 특별한 언급이 없는 한, Noise는 0으로 설정되어 있다.

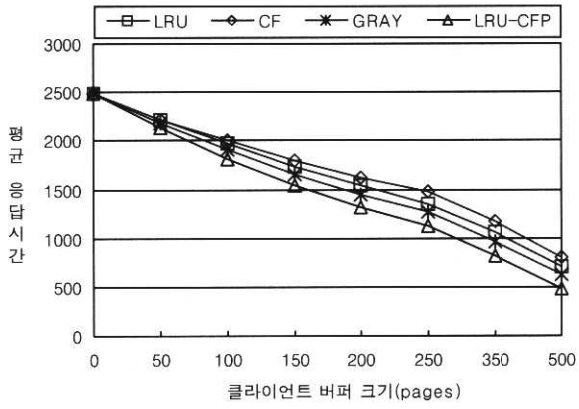
5. 성능 결과 및 분석

이 장에서는 4장의 성능 평가 모델을 바탕으로 각 기법의 성능을 평가하여 그 결과를 제시하고 분석한다. 본 논문의 주요 성능 평가 지수는 평균 응답시간으로서 특정 페이지를 액세스하는데 걸리는 평균 시간을 의미한다. 실험은 복사 방법(replication approach)을[12] 사용하였는데, 실험 시작 시의 초기 편향(initial bias)을 제거하기 위하여 초기 4,000개의 페이지 액세스의 결과는 무시하였다. 이 장에서 제시된 결과 값은 5개의 서로 다른 임의 수를 사용하여 실시된 실험 결과의 평균값으로, 각 실험은 50,000개의 페이지를 액세스할 때까지 실시하였다.

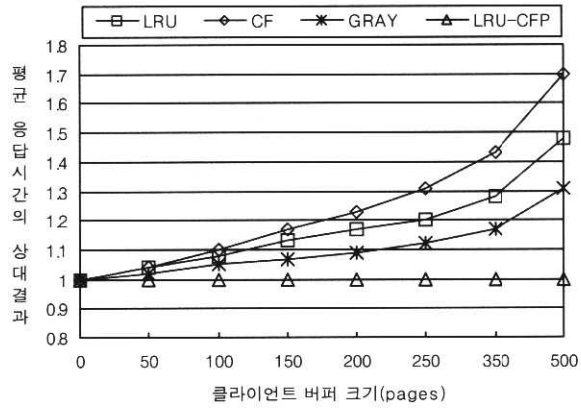
5.1 실험 1 : 캐쉬 용량의 변화에 따른 성능

본 실험에서는 클라이언트의 캐쉬 용량을 변화시키면서 각 기법의 성능추이를 살펴보았다. 클라이언트가 하나의 데이타 페이지를 액세스하는데 걸리는 평균 응답시간의 결과가 (그림 4)에 제시되어 있다. 그리고 각 캐쉬 용량에서 LRU-CFP 기법의 평균 응답시간을 1로 간주할 때, 각 기법의 상대적인 평균 응답시간의 결과가 (그림 5)에 제시되어 있다. 이는 각 기법간의 상대적인 성능 결과를 좀 더 명확히 파악하기 위함이다. 클라이언트에 캐쉬가 존재하지 않으면, 모든 기법들은 2500 tick 정도의 응답시간을 보인다. 이

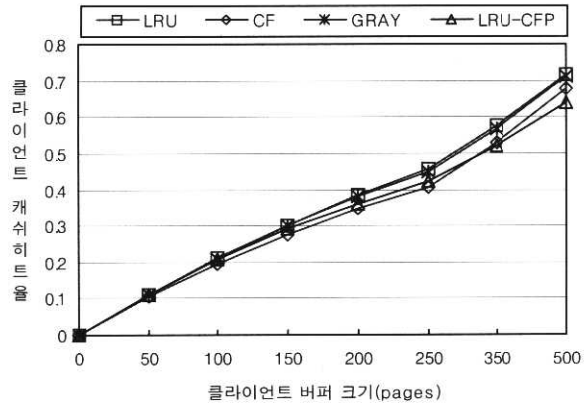
는 전체 데이타베이스가 5000개의 페이지로 구성되고 이들이 주기적으로 방송되므로, 클라이언트가 한 페이지를 액세스하기 위해서는 평균적으로 2500tick을 기다려야 하기 때문이다. 캐쉬의 용량이 늘어남에 따라 각 기법은 더 높은 캐쉬히트율을 보일 수 있기 때문에 점점 평균 응답시간은 줄어들게 된다. 그리고 각 기법간에 존재하는 캐쉬 관리 정책의 차이로 인하여 평균 응답시간의 성능 차이가 발생하기 시작하는데, LRU-CFP, GRAY, LRU 그리고 CF 기법순으로 우수한 성능을 발휘한다. (그림 5)에서 보는 것과 같이 캐쉬 용량이 250페이지 이상이 되면, LRU-CF 기법은 GRAY, LRU, CF 기법에 비해, 대략 10%, 20%, 30% 이상의 성능 향상을 가져온다.



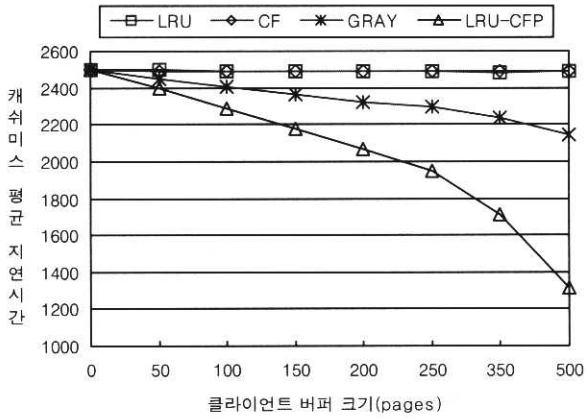
(그림 4) 평균 응답시간(ticks)



(그림 5) 평균 응답시간의 상대결과



(그림 6) 클라이언트 캐쉬히트율



(그림 7) 캐쉬미스시의 평균 지연시간(ticks)

평균 응답시간의 성능 결과를 분석하기 위하여 (그림 6)과 (그림 7)에 캐쉬히트율과 캐쉬미스시의 평균 지연시간이 제시되어 있다. 클라이언트가 어떤 페이지를 액세스할 때 캐쉬미스가 발생하면, 클라이언트는 그 페이지가 방송될 때까지 기다려야 한다. 캐쉬미스시의 평균 지연시간이란 이 대기시간을 의미한다. LRU 기법은 다른 기법에 비해 가장 높은 캐쉬히트율을 보인다. GRAY 기법은 1-Bit LRU 기법에 기초하고 있기 때문에 LRU 기법보다 약간 낮기는 하지만 거의 유사한 캐쉬히트율을 보인다. LRU 기법은 액세스 확률이 높을 것으로 예상되는 가장 최근에 액세스한 Cache Size 개의 페이지를 캐싱한다. 그리고 희생자가 필요하면 가장 오랫동안 액세스되지 않은 페이지를 선정하여 캐쉬에서 제거한다. CF 기법도 CacheSize 개의 페이지를 캐싱하기는 하지만, 과거의 액세스 형태는 완전히 무시하고 방송 순서만을 고려하여 희생자를 선정한다. 이는 캐쉬히트율 측면에서는 무작위로 희생자를 선정하는 것과 동일하다. 따라서 CF는 LRU 기법에 비해 낮은 캐쉬히트율을 보인다.

LRU-CFP 기법은 가장 최근에 액세스한 $x \cdot CacheSize$ 개의 hot 페이지를 관리한다. 이 중에서 액세스 확률과는 무관하게 방송 채널에 가장 늦게 나타날 *CacheSize* 개의 페이지를 캐싱한다. 이는 캐쉬히트율 측면에서는 $x \cdot CacheSize$ 개의 페이지 중 무작위로 *CacheSize* 개의 페이지를 캐싱하는 것과 마찬가지이다. 그러므로 LRU-CFP 기법은 항상 가장 최근에 액세스한 *CacheSize* 개의 페이지만을 캐싱하는 LRU에 비하여 낮은 캐쉬히트율을 보이는 것이다. 비록 CF 기법이 캐쉬히트율 측면에서 무작위로 희생자를 선정하지만, 일단 어떤 캐싱된, 즉 액세스 확률이 높을 것으로 예상되는 페이지는 희생자로 선정되기 전에는 자신의 캐쉬 슬롯을 양보하지는 않는다. 그리고 불행히도 희생자로 선정되어 캐쉬에서 제거됐던 액세스 확률이 높은 페이지는

다시 액세스되어 캐싱될 것이다. 따라서 CF 기법도 비교적 액세스 확률이 높은 페이지들을 캐싱하고 있을 것이다. LRU-CFP 기법은 엔트리 희생자가 필요하면, 가장 액세스 확률이 낮을 것으로 예상되는 페이지를 선정하여 큐에서 삭제한다. 그렇지만 LRU-CFP에서는 프리페치 매커니즘에 의해 hot 페이지들은 액세스 확률에 관계없이 서로 교대로 캐쉬 슬롯을 차지하게 된다. LRU-CFP의 희생자 선정 정책은 CF 기법에 비해 캐쉬히트율 측면에서 매우 바람직하다. 그렇지만, LRU-CFP의 프리페치 매커니즘은 캐쉬히트율에는 부정적인 영향을 미친다. 이와 같은 이유로 인하여 LRU-CFP와 CF 기법은 유사한 캐쉬히트율을 보인다.

LRU-CFP 기법이 상대적으로 낮은 캐쉬히트율을 보이지만, 평균 응답시간에 있어서는 매우 우수한 성능을 보인다. 그 이유는 (그림 7)의 캐쉬미스시의 평균 지연시간과 밀접한 관련이 있다. 클라이언트가 어떤 페이지를 액세스하는데 걸리는 평균 응답시간은 식 (1)과 같이 계산될 수 있다. 캐싱되어 있는 페이지를 액세스할 경우에 응답시간은 0이다.

$$\text{평균 응답시간} = (1 - \text{캐쉬히트율}) \times (\text{캐쉬미스시의 평균 지연시간}) \quad (1)$$

캐쉬 용량이 350페이지일 경우에 LRU, GRAY, LRU-CFP 기법은 각각 58%, 57%, 52%의 캐쉬히트율을 보이며, 각각 2486, 2236, 1713tick 정도의 평균 지연시간을 보인다. 이 값과 식 (1)을 이용하여 각 기법의 평균 응답시간을 구하면, LRU, GRAY, LRU-CFP 기법은 각각 1044, 961, 822 tick 정도의 성능을 보인다. 이 계산 결과는 (그림 4)의 응답시간의 결과와 거의 동일하다. (그림 7)을 살펴보면, 캐쉬미스시의 평균 지연시간에 있어서 프리페치 기법을 적용한 LRU-CFP와 GRAY는 다른 기법에 비해 매우 우수한 성능을 발휘한다. 직관적으로 CF 기법이 평균 지연시간에 있어서 우수한 성능을 보일 것 같지만 그렇지 않다. 그 이유는 희생자로 선정된 페이지를 그 페이지가 방송되기 이전에 다시 액세스한다면 지연시간이 가장 작겠지만, 방송이 되고 난 후에 액세스한다면 매우 긴 지연시간을 필요로 하기 때문이다. CF 기법은 방송 순서와 관계없이 희생자를 선정하는 LRU 기법과 거의 유사한 지연시간을 보일 뿐이다.

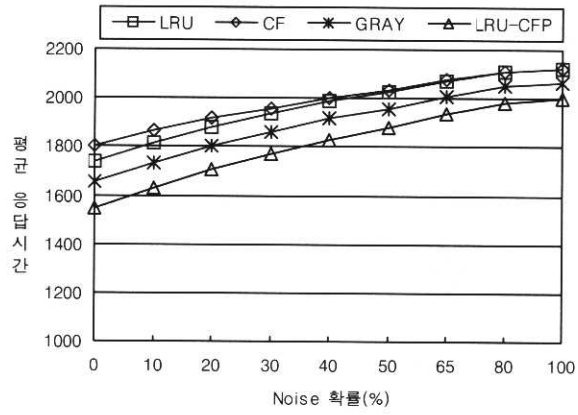
LRU-CFP와 GRAY 기법은 유사한 프리페치 기법을 적용하고 있지만, 프리페치 대상이 되는 페이지의 관리에 있어서 차이가 있다. LRU-CFP 기법에서는 큐에서 관리되는 hot 페이지들이 프리페치 대상이 될 수 있고, GRAY에서는 이전 주기에 액세스된 회색 페이지들이 프리페치의 대상이 될 수 있다. 이를 프리페치 대상이 되는 페이지 수의 관점에서 비교해 보면, LRU-CFP 기법에서는 항상 $x \cdot CacheSize$ 개의 페이지들이 $CacheSize$ 개의 캐쉬 슬롯을 사용하기 위하여 경쟁하게 된다. GRAY 기법에서는 주기의 시작시에는 프리페치 대상이 되는 모든 회색 페이지들은 캐싱되어 있고 이들이 $CacheSize$ 개의 캐쉬 슬롯을 사용한다. 흑색 페이지가 증가함에 따라 회색 페이지가 사용할 수 있는 캐쉬 슬롯의 수는 점점 감소하게 되고 결국 주기의 끝 무렵에 도달하면 회색 페이지들은 1개의 캐쉬 슬롯을 차지하기 위해서 경쟁하게 된다. 이와 같이 GRAY 기법에서는 주기의 시작 무렵에서는 프리페치의 대상이 될 수 있는 대부분의 회색 페이지들이 이미 캐싱되어 있고, 주기의 끝 무렵에서는 회색 페이지들이 사용할 수 있는 캐쉬 슬롯의 수가 극히 제한되기 때문에 LRU-CFP에 비하여 프리페치의 효과가 적다. 또한 프리페치 대상이 되는 페이지들의 교체에는 다음과 같은 차이점이 있다. GRAY 기법은 주기의 끝에서 모든 회색 페이지들을 흰색 페이지로 변경하여 프리페치 대상에서 한꺼번에 제외하고, 모든 흑색 페이지를 회색 페이지로 변경한다. 반면, LRU-CFP에서는 cold 페이지가 액세스됨에 따라 프리페치 대상이 될 수 있는 페이지들이 지속적으로 변경된다. 이와 같은 이유로 인하여 LRU-CFP 기법은 GRAY보다 캐쉬미스시의 평균 지연시간에 있어서 우수한 성능을 보일 수 있다.

5.2 Noise 확률 변화에 따른 성능

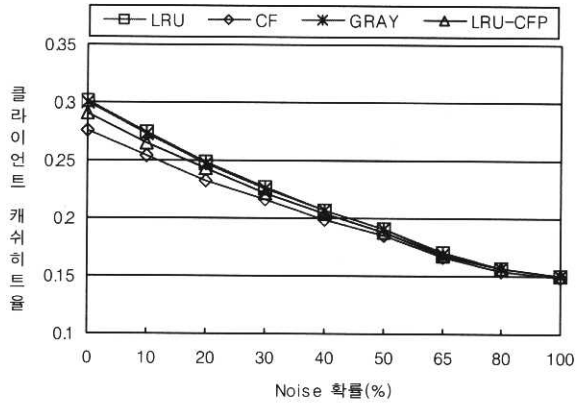
본 실험은 zipf 부하에서 Noise 확률, 즉 클라이언트의 액세스가 zipf 부하를 따르지 않을 확률을 변화시키면서 각 기법의 성능 추이를 살펴보았다. 본 실험에서 클라이언트의 캐쉬 용량은 150개의 페이지를 캐싱할 수 있는 것으로 설정하였다. 클라이언트의 평균 응답시간과 캐쉬히트율이 각각 (그림 8)과 (그림 9)에 제시되어 있다. Noise 확률이 증가하면 클라이언트는 점점 더 무작위로 데이타를 액세스하게 된다. 따라서 각 기법의 캐쉬히트율은 감소하게 되고 평균 응답시간은 증가하게 된다. Noise 확률이 증가함에 따라 각 기법간의 상대적인 성능 차이는 점점 들어들게 되지만, LRU-CFP 기법은 평균 응답시간에 있어서 다른 기법에 비하여 항상 우수한 성능을 보인다. 이는 캐쉬히트율과 캐쉬미스시의 평균 지연시간과 밀접한 연관이 있다.

(그림 9)를 살펴 보면, Noise 확률이 증가함에 따라 각 기법간의 캐쉬히트율의 차이가 감소한다는 것을 알 수 있다. 이는 Noise 확률이 증가함에 따라 각 기법간에 존재하는 캐쉬관리 정책의 차이가 캐쉬히트율에 미치는 영향력이 줄어들기 때문이다. Noise 확률이 100%가 되면, 클라이언트는 AccRange 내의 모든 페이지를 균등하게 액세스하게 되어 클라이언트의 액세스는 균등(uniform) 분포를 이루게 된다. 이 경우에는 희생자 선정시 특정 페이지에 우선권을 준다는 것이 의미가 없기 때문에 각 기법은 거의 동일한 캐쉬히트율을 보이게 된다. 그렇지만, 이 경우에도 프리페치 기법을 적용한 LRU-CFP와 GRAY 기법은 다른 기법에 비하여 우수한 평균 응답시간의 성능을 보인다. 이는 (그림 10)에 제시된 캐쉬미스시의 평균 지연시간의 차이에서 비롯된다. Noise 확률이 증가함에 따라, 캐쉬미스시의 평균 지연

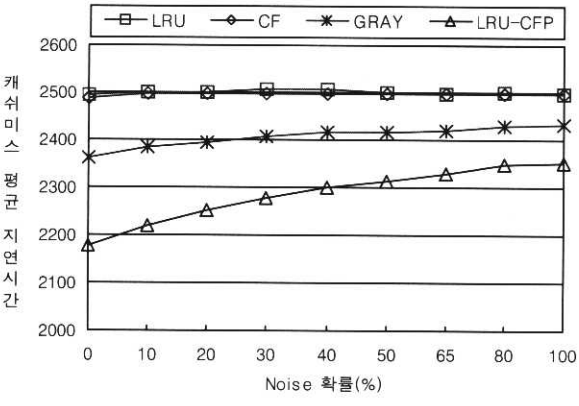
시간도 증가한다. 이는 Noise 확률의 증가에 따라 프리페치 대상에 속하지 않는 cold 페이지들의 액세스 확률도 증가하여 전체적으로 캐쉬미스로 인한 지연시간을 증가시키기 때문이다. 만일 클라이언트가 cold 페이지를 액세스하려면 평균적으로 2500 tick 정도를 대기해야 한다.



(그림 8) 평균 응답시간(ticks)



(그림 9) 클라이언트 캐쉬히트율



(그림 10) 캐쉬미스시의 평균지연시간

6. 결 론

본 논문에서는 특정 데이타들이 방송 채널을 통하여 지속적으로 전파되는 방송 환경의 특수성을 고려하여 데이타 방송 기법에 적합한 새로운 캐쉬 관리 기법인 LRU-CFP (least recently used with closest-first and prefetch) 기법을 제안했다. 그리고 모의 실험을 통하여 LRU-CFP 기법의 성능과 GRAY, LRU 및 CF 기법의 성능을 비교하였다. LRU-CFP 기법은 비교적 액세스 확률이 높은 페이지들 중에서 방송 지연시간이 긴 페이지들을 캐싱하도록 희생자 관리 정책이 세분화되어 있고 프리페치 기법이 적용되어 있다. 그러므로 LRU-CFP 기법은 적당한 수준의 캐쉬히트율을 보이면서도 캐쉬미스로 인한 지연시간을 획기적으로 줄일 수 있다. 따라서 LRU-CFP 기법은 클라이언트가 특정 데이타를 액세스하는데 걸리는 평균 응답시간의 성능이 GRAY, LRU 및 CF 기법에 비하여 훨씬 우수하다.

본 논문의 미래 연구 과제로서 다중 방송 디스크 환경에 적합한 캐쉬 관리 기법과 방송 스케쥴링 기법에 관한 연구를 진행할 예정이다.

참 고 문 헌

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, "Broadcast Disks : Data Management for Asymmetric communications environments," Proc. of ACM SIGMOD Int. Conf. on Management of Data, San Jose, California, pp.199-210, May, 1995.

[2] S. Acharya, M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcast," Proc. of ACM SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, pp.183-194, May, 1997.

[3] S. Acharya, "Broadcast Disks : Dissemination-based Data Management for Asymmetric Communication Environments," PhD thesis, Brown University, 1998.

[4] D. Aksoy and M. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting," Proc. of IEEE INFOCOM, San Francisco, CA, March, 1998.

[5] M. Franklin, M. Carey and M. Livny, "Transactional Client-Server Cache Consistency : Alternatives and Performance," ACM Trans. on Database Syst., Vol.22, No.3, pp.315-363, 1997.

[6] M. Franklin and S. Zdonik, "A Framework for Scalable Dissemination-Based Systems," In the Int. Conf. Object Oriented Programming Language Systems (OOPSLA '97), Atlanta, GA, Oct., 1997.

[7] M. Franklin and S. Zdonik, "Data in Your Face : Push Technology in Perspective," Proc. of ACM SIGMOD Int. Conf. on Management of Data, Seattle, WA, June, 1998.

[8] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nicols, M. Satyanarayanan, Robert N. Sidebotham and Michael J. West, "Scale and Performance in a Dis-

tributed File System," ACM Trans. on Computer Systems, Vol.6, No.1, pp.51-89, Feb., 1988.

[9] T. Johnson and D. Shasha, "2Q : A Low Overhead High Performance Buffer Management Replacement Algorithm," Proc. of Int. Conf. on VLDB, Santiago, Chile, pp.439-450, Sep., 1994.

[10] V. Liberatore, "Caching and Scheduling for Broadcast Disk Systems," Technical Report UMIACS-TR-98-71, University of Maryland, 1998.

[11] V. Liberatore, "Caching and Scheduling for Broadcast Disk Systems," In the Second Workshop on Algorithm Engineering and Experiments ALENEX 00, San Francisco, Jan., 2000.

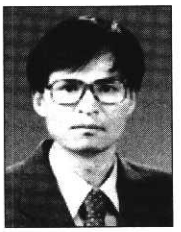
[12] A. M. Law and W. D. Kelton, "Simulation Modeling & Analysis," McGraw-Hill, 1991.

[13] E. J. ONeil, P. E. ONeil and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp.297-306, May, 1993.

[14] H. Schwetman, CSIM User's Guide for Use with CSIM Revision 16, Microelectronics and Computer Technology Corporation, 1992.

[15] K. Stathatos, "Air-Caching : Adaptive Hybrid Data Delivery," PhD Thesis, Maryland University, 1999.

[16] A. Tanenbaum, "Modern Operating Systems," Prentice Hall, 1992.



권 혁 민

e-mail : hmkwon@semyung.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1994년 한국과학기술원 정보 및 통신공학과 (공학석사)

1998년 한국과학기술원 정보 및 통신공학과 (공학박사)

1984년~1991년 대우전자 중앙연구소 컴퓨터개발부 선임연구원

1999년~현재 세명대학교 소프트웨어학과 조교수

관심분야 : 트랜잭션 처리, 분산/병렬 데이타베이스, 이동 컴퓨팅