2Q-CFP: 방송에 기초한 정보 시스템을 위한 클라이언트 캐쉬 관리 기법

(2Q-CFP: A Client Cache Management Scheme for Broadcast-based Information Systems)

권 혁 민 †

(HyeokMin Kwon)

요 약 방송에 기초한 데이타 전달 방식은 방대한 규모의 클라이언트에게 데이타를 전파하기 위한 효과적인 기술로서 큰 관심을 끌고 있다. 방송에 기초한 정보 시스템(broadcast-based information system: BBIS)의 주된 동기는 자신이 지원하는 클라이언트의 수는 시스템의 성능에 전혀 영향을 미치지 않고 임의로 증가될 수 있다는 것이다. BBIS의 성능은 클라이언트 캐싱 전략과 데이타 방송 스케쥴링 기법에 크게 영향을 받는다. 본 논문은 전자의 문제를 취급하여 BBIS에 적합한 2Q-CFP로 명명된 새로운 클라이언트 캐쉬 관리 기법을 제안한다. 그리고 모의 실험 모델을 통하여 2Q-CFP 기법의 성능을 평가한다. 성능 평가 결과에 의하면 2Q-CFP 기법은 GRAY, LRU, 그리고 CF 기법보다 평균 응답시간에 있어서 더 우수한 성능을 보인다.

키워드 : 캐쉬 관리, 버퍼 관리, 데이타 방송

Abstract *Broadcast-based data delivery has attracted a lot of attention as an efficient way of* disseminating data to very large client populations. The main motivation of broadcast-based information systems (BBISs) is that the number of clients that they serve can grow arbitrarily large without any effect on their performance. The performance of BBISs depends mainly on client caching strategies and on data broadcast scheduling mechanisms. This paper addresses the former issue and proposes a new client cache management scheme, named 2Q-CFP, that is suitable to BBISs. This paper also evaluates the performance of 2Q-CFP on the basis of a simulation model. The performance results indicate that 2Q-CFP scheme shows superior performances over GRAY, LRU and CF in the *average response time.*

Key words : Cache Management, Buffer Management, Data Broadcast

1. 서 론

최근 들어, 인터넷의 폭발적인 인기와 함께 전 세계에서는 온라인 데이타 서비스에 대한 요구가 급격하게 증가하고 있고, 이에 따라 정보제공(information-feed) 응용들이 큰 관심을 끌고 있는 실정이다. 정보 제공 응용의 예로는 날씨, 주식, 뉴스, 교통 등의 정보 서비스 응용들이 있는데, 이런 종류의 응용들은 정보를 생성하여 관리하고 제공하는 단일 또는 몇몇의 서버와 정보를 검색하고 이를 이용하는 다수의 클라이언트로 구성되는 정보소비(information-consumption) 모델의 형태를 띠고 있다[1].

방송(broadcast)에 기초한 데이타 전달 기법은 정보소비 모델 형태의 정보제공 응용에서 정보 전달을 위한 주요 기술로 인정을 받고 있다[2-4]. 방송 기술을 채택한 데이타 방송 시스템에서 방송 서버는 다수의 클라이언트에게 특정 데이타들을 지속적으로 전파하고 각 클라이언트는 자신이 원하는 데이타가 방송 채널에 나타나면 이를 검색한다. 데이타 방송 시스템의 성능은 그 시스템이 지원해야 하는 클라이언트의 수에 영향을 받지 않기 때문에 시스템의 확장성(scalability)이 매우 우수하다. 따라서 방송 시스템은 웹이나 정보제공 응용과 같이 방대한 규모의 클라이언트를 지원해야 하는 응용 분야에 매우 적합하다. 뿐만 아니라, 방송 시스템은 클라이언트에

· 본 연구는 한국과학재단 목적기초연구(R05-2002-000-00901-0) 지원으로 수행되었음

† 정 회 원 : 세명대학교 소프트웨어학과 교수
hmkwon@venus.semyung.ac.kr

논문접수 : 2003년 5월 9일
심사완료 : 2003년 8월 29일

서 서버로의 통신 대역폭이 제한적이어서 클라이언트가 자신의 요청을 서버로 전달하는데 비용이 많이 들거나 또는 불가능할 가능성이 많은 이동 컴퓨팅 환경(mobile computing environment)에도 매우 적합하다.

데이타 방송 시스템의 주요 성능평가 기준은 평균 응답시간(response time)인데, 이는 클라이언트가 특정 데이타의 액세스를 요청한 시간부터 실제 그 데이타를 액세스하는데 까지 걸린 평균 시간을 의미한다. 방송 시스템에서 응답시간의 성능은 다음 두 가지의 핵심 기술에 의해 크게 영향을 받는다[1,2,5-9]: (1) 데이타 방송 스케쥴링(data broadcast scheduling) 기법과 (2) 클라이언트 캐쉬 관리 기법. 데이타 방송 스케쥴링 기법은 서버측에 적용되는 기술로서 방송할 데이타를 선정하고, 이들의 방송 순서 및 방송 빈도(broadcast frequency)를 결정하는 문제를 다룬다. 방송되는 데이타의 효용성은 클라이언트의 액세스 형태를 방송 스케쥴에 얼마나 잘 반영하는가에 달려 있다. [2, 5]의 연구에서는 각 데이타의 액세스 확률을 고려하여 방송 빈도를 결정하기 위하여, 방송 채널을 다중 방송 디스크(Multiple Broadcast Disks)로 모델링했다. [1, 7, 10]의 연구에서는 클라이언트에서 서버로의 상향링크(uplink)가 존재하는 환경을 위한 스케쥴링 기법을 연구했다. 이 연구들은 클라이언트에서 명시적으로 요청되는 정보를 바탕으로 방송할 데이타를 동적으로 변화시키는 스케쥴링 기법을 제안했다.

클라이언트 캐쉬 관리 기법은 클라이언트의 지역 메모리에 시스템이 허용하는 한도 내에서 특정 데이타들을 캐싱하여 방송 채널에서의 대기시간을 줄이기 위한 기술이다. 클라이언트가 특정 데이타들을 캐싱하게 되면, 방송 채널에서의 클라이언트 액세스 패턴에 변화가 발생한다. 또한 클라이언트에서 캐쉬미스가 발생할 경우에, 방송 스케쥴은 해당 데이타를 검색하는데 걸리는 시간을 결정한다. 이와 같이 클라이언트 캐쉬 관리 기법과 방송 스케쥴링 기법은 매우 밀접하게 연관되어 있기 때문에 각 기법은 다른 기법의 특성을 충분히 고려하여 설계되어야 한다. 클라이언트 캐쉬가 존재하지 않거나, 클라이언트의 캐쉬 용량이 작을 경우에는 방송 시스템의 성능은 스케쥴링 기법에 의해 좌우된다. 그러나 캐쉬 용량이 어느 정도 이상이 되면, 방송 시스템의 성능은 스케쥴링 기법보다는 캐쉬 관리 기법에 더 큰 영향을 받는다고 알려져 있다[8,9]. 스케쥴링 관한 연구, 캐쉬 관리 관한 연구, 그리고 캐쉬 관리 및 스케쥴링 기법이 조합된 연구는 모두 방송 시스템의 성능 향상을 위한 매우 흥미있는 연구이지만, 본 논문에서는 캐쉬 관리 기법에 한하여 연구를 진행한다. 본 논문에서는 모든 데이타들이 일정한 순서로 주기적으로 방송 채널을 통하여 전파되는 평형 방송 디스크(flat broadcast disk) 환경에서[2,5] 클라이언트 캐쉬 관리 기법을 개발한다.

클라이언트 캐쉬 관리 기법은 요청-응답(request-response) 형태의 데이타 전달 방식을 채택한 클라이언트-서버 시스템의 성능 향상을 위하여 소개된 기술이다[11,12]. LRU[13], LRU-k[14], 그리고 2Q[15] 기법과 같이 기존의 요청-응답 형태의 시스템을 위한 캐쉬 관리 기법들이 제안되었지만, 이들은 특정 데이타들이 지속적으로 전파되는 방송환경의 특수성을 고려하지 않았기 때문에 이들을 그대로 데이타 방송 시스템에 적용시키기에는 무리가 있다. 기존의 요청-응답 시스템과 비교하여, 방송 시스템에서는 다음과 같은 사항을 고려하여 캐쉬 관리 기법을 개발해야 한다. 첫째, 요청-응답 시스템에서는 각 데이타에 대해 캐쉬미스(cache-miss)를 처리하기 위한 비용이 일정하므로, 기존의 연구는 대부분 캐쉬히트율을 높이는 방향으로만 진행되어 왔다. 그러나 방송 시스템에서는 방송 스케쥴에 따라 캐쉬미스를 처리하는 비용이 각 데이타마다 상이하다는 특징이 있다. 그러므로 방송 시스템에서는 캐쉬히트율 뿐만 아니라 캐쉬미스시의 지연시간도 함께 고려하여 캐쉬를 관리해야 한다. 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 방송 시스템에서는 방송 채널을 통하여 데이타들이 주기적으로 전파되므로 특별히 서버 및 네트워크 자원에 별도의 부담을 지우지 않고도 원하는 데이타를 프리페치할 수 있다. 그러므로 방송 시스템에서는 프리페치 기법을 적극적으로 적용하는 것이 바람직하다.

본 논문은 이와 같은 관점에서 방송 시스템에 적합한 새로운 캐쉬 관리 기법 2Q-CFP(two queue with closest-first and prefetch)를 개발한다. 방송 환경을 위한 캐쉬 관리 기법의 성능은 캐쉬히트율과 캐쉬미스시의 평균 지연시간과 밀접한 관련이 있다. 2Q-CFP 기법은 캐쉬히트율을 높이기 위하여 데이타베이스내의 페이지들을 hot, warm, 그리고 cold 페이지로 동적으로 구분하여 관리한다. 2Q-CFP 기법은 액세스 확률이 높을 것으로 예상되는 hot 페이지들을 클라이언트 버퍼에 모두 캐싱한다. 그리고 중간 정도의 액세스 확률을 보이는 warm 페이지들은 그 일부만을 클라이언트 버퍼에 캐싱하는데, 방송 지연시간이 긴 페이지들이 우선권을 가지며 캐싱된다. 데이타 방송이 진행되더라도, 방송 지연시간이 긴 warm 페이지들이 캐싱되도록 프리페치 기법이 2Q-CFP에 통합되어 있다. 캐쉬 관리 기법은 다수의 페이지들이 한정된 캐쉬 슬롯을 효율적으로 사용하

도록 하기 위하여 적절한 희생자 선정 정책을 필요로 한다. 본 논문은 기존의 희생자(victim) 개념을 확장하여 슬롯 희생자(slot victim)와 엔트리 희생자(entry victim)의 개념을 소개하고 이 개념에 바탕을 둔 새로운 희생자 선정 정책을 소개한다. 2Q-CFP 기법은 데이타베이스 내의 페이지를 동적으로 구분하여 관리하고 세분화된 희생자 선정 정책을 사용하기 때문에 높은 캐쉬 히트율을 보일 수 있다. 그리고 방송 지연시간이 긴 warm 페이지들이 캐싱되도록 프리페치 기법이 적용되어 있기 때문에 캐쉬미스시의 평균 지연시간을 줄이는 것이 가능하다.

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

2. 관련 연구

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

2.1 방송 시스템 모델 및 가정

방송 데이타를 스케줄링할 때, 데이타의 액세스 확률이 높을수록 빈번하게 방송하면 전체 시스템 성능은 향상될 것이다. 클라이언트 캐쉬가 존재하지 않을 경우에 각 데이타는 자신의 액세스 확률의 제곱근 값에 비례하게 방송하는 것이 가장 이상적이라고 알려져 있다[16, 17]. [2, 5]의 연구는 액세스 확률을 고려하여 데이타들을 주기적으로 전파하기 위하여 방송 채널을 저장 용량과 방송빈도가 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 데이타들의 액세스 확률을 미리 파악할 수 없다면 모든 데이타들을 하나의 디스크에 저장하게 되는데, 이 경우에 모든 데이타들의 방송 빈도는 동일하다. 이를 평형 방송 디스크(flat broadcast disk) 모델이라 한다. 평형 방송 디스크 모델에서는 그림 1과 같이 데이타베이스의 모든 페이지들이 일정한 순서로 주기적으로 방송된다.

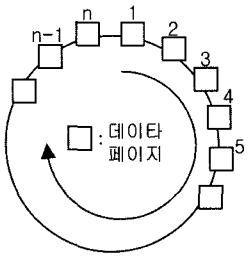

그림 1 평형 방송 디스크 모델

본 논문은 클라이언트의 액세스 패턴을 미리 파악하기가 어렵거나 불가능하다고 가정하고 평형 방송 디스크 모델하에서 클라이언트 캐쉬 관리 기법을 개발한다. 본 논문은 데이타 방송의 단위와 클라이언트 액세스의 단위가 길이가 일정한 데이타 페이지라고 가정한다. 클라이언트가 데이타를 캐싱하면, 서버의 데이타와 클라이언트의 데이타 사이에 일관성 문제가 야기될 수 있다. 본 논문은 방송 데이타는 읽기 전용 데이타로서 변경이 발생하지 않는다고 가정한다. 따라서 클라이언트 캐쉬에 저장된 데이타의 무효화(invalidation)가 발생하지 않는다. 그렇지만 서버에서 변경된 데이타에 대하여 적절한 방식으로 변경 정보를 방송하면, 이를 본 논문에서 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[11,12]. 캐쉬 관리 기법 중 LRU(least recently used)는[13] 구현이 간단하여 가장 많이 사용되는 기법이다. 이 기법은 캐쉬에 저장된 페이지들의 우선순위를 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 무작위로 액세스되는 페이지들이 많거나 데이타베이스 내의 어떤 영역을 임의로 스캔할 경우에는 캐쉬히트율이 낮은 단점이 있다. 이 단점을 해결하기 위해 제안된 LRU-k 기법은[14] 각 페이지들의 마지막 k 개의 액세스 시간을 고려하여 페이지의 우선 순위를 결정한다. 이를 위해 이 기법은 각 페이지들의 마지막 k개의 액세스 시간을 유지 관리해야 한다. LRU-k 기법은 높은 캐쉬히트율을 보이는 장점이 있지만, 큰 실행 부담을 보인다.

[15]의 연구는 LRU 기법과 유사한 실행 부담을 보이면서 LRU-2에 필적하는 캐쉬히트율을 보이는 2Q(two

queue) 기법을 제안했다. 이 기법은 자주 액세스되는 페이지와 그렇지 않은 페이지를 구분해서 관리하기 위하여 두개의 큐, A_m과 A_1 큐를 운영한다. 2Q 기법에서 어떤 새로운 페이지가 액세스되면 A_1 큐에 삽입되고, A_1 큐에 있는 페이지가 다시 액세스되면 A_m 큐로 옮겨진다. A_1 큐는 FIFO 형태로 관리되며, A_m 큐는 LRU 형태로 관리된다. 만일 새로운 페이지를 캐싱하기 위해 희생자가 필요할 경우에는, A_1 큐의 맨 뒤에 있는 페이지나 A_m 큐의 맨 뒤에 있는 페이지를 희생자로 선택한다. 2Q 기법은 무작위로 액세스되는 페이지를 일찍 캐쉬에서 제거할 수 있기 때문에 우수한 캐쉬히트율을 보인다.

LRU, LRU-k, 그리고 2Q 기법은 기존의 요청-응답 시스템을 위하여 개발된 기법으로서 캐쉬미스를 처리하기 위한 비용이 각 페이지마다 상이한 방송 환경의 특수성을 고려하지 않았다. 방송 환경을 위하여 캐쉬미스시의 처리 비용을 고려한 캐쉬 관리 기법에는 CF (closest-first), GRAY, PIX 및 LIX 기법들이 있다. CF 기법은[8,9] 캐싱되어 있는 페이지 중에서 다음에 가장 먼저 방송 채널에 나타나는 페이지를 희생자로 선정하는 기법이다. CF 기법은 과거의 액세스 형태를 고려하지 않고 단순히 방송 순서만을 고려하여 희생자를 선택하기 때문에 캐쉬히트율이 낮은 단점이 있다.

[8, 9]의 연구는 CF 기법과 마킹 알고리즘(marking algorithm)으로 알려진 1-bit LRU 기법을 통합하고, 여기에 프리페치 기법을 적용하여 GRAY 기법을 개발했다. 이 기법은 데이타베이스의 각 페이지들이 흑색, 회색 또는 흰색의 상태 중에서 어떤 상태에 있는지에 대한 정보를 유지 관리한다. 흑색 페이지는 버퍼에 캐싱된 페이지를 의미하며, 흰색 페이지는 캐싱되지 않은 페이지를 의미한다. 회색 페이지는 프리페치 대상이 되는 페이지이며 버퍼에 캐싱되어 있을 수도 있다. GRAY 기법은 희생자가 필요하면, 캐싱된 회색 페이지 중에서 가장 먼저 방송될 페이지를 희생자로 선정하여 캐쉬에서 제거한다. GRAY 기법에서 어떤 페이지가 액세스되면 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 끝에서 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 간주된다. 그러므로 방송 채널에 회색 페이지가 나타나고, 마침 그 페이지가 캐싱되어 있지 않으면 프리페치하여 버퍼에 캐싱한다. GRAY 기법에서 일단 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 되어 캐쉬 슬롯을 낭비할 수도 있다는 단점이 있다.

[2, 5]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_i 의 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_i 의 값이 가장 작은 페이지를 희생자로 선정한다. 이 기법은 방송 디스크의 수가 많을수록, PIX 기법과 유사해 지며, 방송 디스크의 수가 적어질수록 LRU 기법과 유사해진다. 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: 2Q-CFP

이 장에서는 본 논문이 새로이 제시하는 2Q-CFP (two queue with closest-first and prefetch) 기법에 대하여 기술한다. 2Q-CFP 기법은 클라이언트의 캐쉬 버퍼를 효율적으로 사용하기 위하여 데이타베이스내의 페이지들을 hot, warm, 그리고 cold 페이지로 구분하여 관리한다. hot 페이지는 액세스 확률이 높은 페이지들을 의미하며, cold 페이지는 액세스 확률이 낮은 페이지를 의미한다. 그리고 warm 페이지는 중간 정도의 액세스 확률을 보이는 페이지를 의미한다. 이들의 구분은 미리 고정되어 있는 것이 아니라, 클라이언트가 데이타를 액세스함에 따라 다음과 같이 지속적으로 변화된다. 클라이언트가 cold 페이지를 액세스하면 그 페이지는 warm 페이지로 격상되고, warm 페이지를 다시 액세스하면 hot 페이지로 격상된다. 클라이언트가 관리하는 hot 페이지와 warm 페이지의 수는 클라이언트의 캐쉬 용량에 따라 적절하게 제한된다. 따라서 어떤 페이지가 격상되면 희생자가 필요할 수도 있다. 이 경우에 희생자로 선

정된 페이지는 아래 그룹의 페이지로 격하된다.

2Q-CFP 기법은 Q_h와 Q_w로 명명된 2개의 큐를 운영하는데, hot 페이지는 Q_h에서 관리되며 warm 페이지는 Q_w에서 관리된다. Q_h와 Q_w 큐는 전체 캐쉬 슬롯을 적절하게 나누어 사용하는데, 이들이 사용하는 캐쉬 슬롯의 비율은 클라이언트의 액세스 형태를 고려하여 조정할 수 있는 변수로서 성능 튜닝을 위해 적절하게 설정해야 한다. 방송 환경을 위한 캐쉬 관리 기법의 성능은 캐쉬히트율과 캐쉬미스시의 평균 지연시간과 밀접한 관련이 있다. Q_h의 관리 방법이 높은 캐쉬히트율의 실현을 목표로 하고 있는데 비하여, Q_w의 관리 방법은 캐쉬미스시의 평균 지연시간의 최소화에 초점을 맞추고 있다. 2Q-CFP 기법은 캐쉬히트율을 높이기 위하여 액세스 확률이 높은 hot 페이지는 항상 클라이언트 버퍼에 캐싱된다. Q_h는 LRU 형태의 체인을 유지하여 hot 페이지들의 우선 순위를 관리한다. 클라이언트가 warm 페이지를 액세스하면 그 페이지는 hot 페이지로 격상되어 Q_h의 LRU 체인의 맨 앞에 배치된다. 이때 희생자가 필요하면 LRU 체인의 맨 뒤에 있는 페이지를 선정하여 warm 페이지로 격하시킨다.

Q_h는 자신에게 할당된 캐쉬 슬롯의 수 만큼의 hot 페이지를 관리하는데 비하여, Q_w는 자신에게 할당된 캐쉬 슬롯의 수보다 더 많은 warm 페이지들을 관리한다. warm 페이지들은 그 일부만이 클라이언트 버퍼에 캐싱되는데, 방송 지연시간이 긴 페이지들이 우선권을 가진다. 데이타 방송이 진행되더라도 방송 지연시간이 가장 긴 warm 페이지들이 캐싱될 수 있도록 그림 2와 같은 원리의 프리페치 기법이 2Q-CFP에 통합되어 있다.

그림 2에서 Q_w는 2개의 캐쉬 슬롯을 사용할 수 있고, 4개의 warm 페이지를 관리한다고 가정한다. 이 그림에서 hot 페이지는 항상 고정적으로 캐싱되어 있는 것이다. 시간 t_1 시점에서 클라이언트는 warm 페이지 중에서는 방송 지연시간이 가장 긴 c와 d를 캐싱하고 있다. 시간이 경과하여 페이지 a가 방송 채널에 나타나면, 클

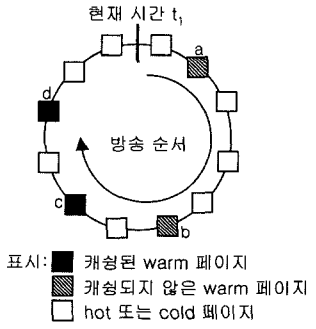

그림 2 프리페치 기법의 원리

라이언트는 캐싱된 warm 페이지 중에서 가장 먼저 방송될 c를 캐쉬에서 제거하고 대신에 a를 프리페치하여 캐싱한다. 따라서 a가 방송되고 나면 클라이언트는 그 순간에 있어 방송 지연시간이 가장 긴 warm 페이지인 a와 d를 캐싱하게 된다. 마찬가지로 b가 방송되고 난 직후에는 d 대신에 b를 캐싱한다.

이와 같은 방식으로 2Q-CFP 기법은 warm 페이지들을 교대로 캐싱한다. 2Q-CFP에서 캐싱되지 않은 warm 페이지는 프리페치 대상이 된다. 따라서 방송 채널에 프리페치 대상이 되는 페이지가 나타나면, 캐싱되어 있는 warm 페이지 중에서 다음에 가장 먼저 방송될 페이지를 희생자로 선정하여 캐쉬에서 제거하고 현재 방송되는 페이지를 프리페치하여 캐싱한다. Q_w에서 프리페치 대상이 되는 페이지를 몇 개까지 관리할 것인가는 성능 튜닝을 위한 중요한 변수이다. 본 논문에서는 이 값을 성능상의 이유로 전체 캐쉬 슬롯 수의 0.5 배로 설정한다. 전체 캐쉬 슬롯의 수를 n이라 하고, Q_w 큐가 전체 캐쉬 슬롯의 30%를 사용한다고 가정하면, Q_w에서는 $0.8 \cdot n$개의 페이지들이 관리되며 이들 중에서 방송 지연시간이 가장 긴 $0.3 \cdot n$개의 페이지들이 캐싱되고 있는 것이다.

캐쉬 관리 기법은 다수의 페이지들이 한정된 캐쉬 슬롯을 효율적으로 사용하도록 하기 위하여 적절한 희생자 선정 정책을 필요로 한다. 본 논문은 Q_w를 관리하기 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, Q_w에 새로운 엔트리를 추가하기 위하여 Q_w의 엔트리 중에서 삭제되어야 하는 희생자가 필요하다. 본 논문은 전자를 슬롯 희생자라 명명하고 후자를 엔트리 희생자라 명명한다. 엔트리 희생자로 선택된 페이지는 cold 페이지로 격하되어 Q_w에서 삭제되며, 슬롯 희생자로 선정된 페이지는 단순히 자신의 캐쉬 슬롯만을 양보한다. 만일 엔트리 희생자로 선정된 페이지가 캐싱되어 있을 경우에는 별도의 슬롯 희생자는 필요하지 않다. Q_w의 관리에서 엔트리 희생자는 FIFO 형태로 선정되며, 슬롯 희생자는 CF 형태로 선정된다. 즉, 엔트리 희생자로는 Q_w의 FIFO 체인의 맨 뒤에 있는 페이지가 선정되어 큐 엔트리에서 삭제되며, 슬롯 희생자로는 캐싱되어 있는 warm 페이지 중에서 가장 먼저 방송될 페이지가 선정되어 캐쉬 슬롯에서 제거된다.

전술한 바와 같이 Q_h는 LRU 형태로 관리되며, Q_w는 FIFO와 CF 방식이 조합된 형태로 관리된다. 2Q-CFP 기법에서 클라이언트가 어떤 페이지를 액세스할 경우에 큐를 관리하기 위한 자세한 과정이 알고리즘 1에 제시

되어 있다. 이 알고리즘은 클라이언트가 정상 상태에 도달하여 Q_h와 Q_w가 모두 최대의 큐 엔트리를 관리하고 있다고 가정하고 있다. 클라이언트가 hot 페이지를 액세스하면 그 페이지를 위한 큐 엔트리는 Q_h의 LRU 체인의 맨 앞으로 이동된다. 클라이언트가 캐싱된 warm 페이지를 액세스하면, 그 페이지의 큐 엔트리는 Q_h의 맨 앞으로 이동되어 관리된다. 그리고 Q_h의 맨 뒤에 있는 엔트리는 warm 페이지로 격하되어 Q_w의 FIFO 체인의 맨 앞으로 이동된다. 클라이언트가 캐싱되지 않은 warm 페이지를 액세스하기 위해서는 그 페이지가 방송 채널에 나타날 때까지 대기해야 한다. 그리고 Q_h의 맨 뒤에 있는 엔트리를 Q_w의 맨 앞으로 이동한다. 그리고 Q_w에서 슬롯 희생자를 선택하여 캐쉬에서 제거하고 현재 액세스하려는 페이지를 캐싱한다. 그리고 그 페이지의 큐 엔트리를 Q_w에서 Q_h의 맨 앞으로 이동한다. 클라이언트가 cold 페이지를 액세스하려면, Q_w에서 슬롯 희생자를 선정하여 캐쉬 슬롯에서 제거하고 엔트리 희생자를 선정하여 큐에서 삭제한다. 그리고 그 cold 페이지를 비워진 캐쉬 슬롯에 저장하고 그 페이지의 큐 엔트리를 Q_w의 맨 앞에 추가한다.

2Q-CFP 기법은 캐싱되지 않은 warm 페이지가 방송 채널에 나타나면 그 페이지를 프리페치하여 클라이언트 버퍼에 캐싱한다. 어떤 데이타가 프리페치될 경우에는 슬롯 희생자만이 필요하며 엔트리 희생자는 필요하지 않다. 또한, Q_w의 FIFO 체인이 조정되는 것도 아니다. 2Q-CFP 기법은 hot 페이지와 warm 페이지를 구분해서 관리하기 위하여 두 개의 큐를 운영한다는 점에서 2Q 기법과 유사한 측면이 있다. 그러나 2Q 기법은 A_1 큐를 FIFO 형태로 관리되는데 비하여, 2Q-CFP 기법은 Q_w 큐를 방송 환경의 특수성을 고려하여 CF와 FIFO가 조합된 형태로 운영된다는 점에서 다르다. 또한, 2Q-CFP 기법에는 캐쉬미스시 평균 지연시간을 줄이기 위하여 프리페치 기법이 적용되어 있다는 점에서도 다르다. 그리고 희생자 선정 정책에서 2Q 기법은 hot 페이지를 직접 cold 페이지로 격하시키는데 비하여, 2Q-CFP 기법은 hot 페이지를 warm 페이지를 거쳐서 cold 페이지로 격하시킨다.

알고리즘 1 클라이언트가 페이지 p를 액세스할 경우

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

4. 성능 평가 모델

이 장에서는 본 논문이 제안한 2Q-CFP 기법의 성능을 파악하기 위하여 성능 평가 모델을 제시한다. 비교 대상으로는 GRAY, CF, LRU 기법을 선정하였다. 성능 평가 모델은 [2, 5, 8]의 연구를 참고로 하고 있고, MCC에서 개발한 CSIM[18] 언어를 이용하여 구현하였다.

본 성능 평가 모델은 크게 방송 서버와 클라이언트로 구성된다. 방송 환경의 특성상 한 클라이언트의 성능은 다른 클라이언트들의 존재 여부에 영향을 받지 않기 때문에 본 논문은 클라이언트의 수를 하나로 고정한다. 방송 서버의 방송 스케줄러는 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 않으

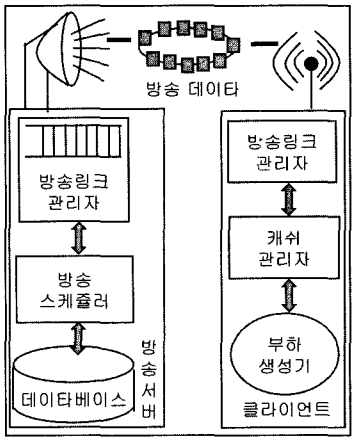

그림 3 성능 평가 모델

면 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[2, 5, 8] 마찬가지로 tick이라는 논리적 시간 단위를 사용한다. tick은 하나의 데이타 페이지를 방송하는데 걸리는 시간을 의미한다. 본 논문에서 사용된 입력변수들은 표 1에 제시되어 있는데, 이들의 설정 값은 대부분 [2, 5, 8]의 연구를 따르고 있다.

표 1 성능 평가 모델의 입력 변수

입력변수	의미	설정
ServerDbSize	데이타베이스의 크기	5000 pages
AccessRange	클라이언트가 액세스하는 페이지 범위	1000 pages
CacheSize	클라이언트 캐쉬슬롯 수	0~500 pages
Q_h_Size	Q_h가 사용하는 캐쉬슬롯 수	70% of CacheSize
Q_w_Size	Q_w가 사용하는 캐쉬슬롯 수	30% of CacheSize
ThinkTime	클라이언트가 페이지를 액세스한 후에 다음 액세스 요청까지의 시간 간격	2 ticks
RegionSize	zipf 분포에서 동일한 액세스 확률을 갖는 페이지 수	50
NoiseProb	zipf 분포를 벗어날 확률	0~100%
θ	zipf 분포의 θ 값	0.95

서버의 데이타베이스는 ServerDbSize 수의 페이지로 구성되며, 서버는 이 페이지들을 주기적으로 방송한다. 클라이언트는 이 페이지들을 모두 액세스하는 것이 아니라, 전체 데이타베이스의 일정 범위만을 액세스하는데, 이 범위를 AccessRange로 정의한다. 즉 클라이언트는 1~AccessRange 사이의 페이지를 액세스한다. 본 논문은 불균등한(non-uniform) 액세스 형태를 모델링하기 위해 많이 사용되는 zipf 분포 부하 모델[5,8,15]에서 각 기법의 성능을 비교한다. 본 논문의 zipf 부하 모델에서는 클라이언트가 액세스하는 AccessRange 내의 페이지를 RegionSize 만큼의 페이지로 나누어 다수의 영역으로 구분하는데, 각 영역은 서로 겹치지는 않는다. 따라서 표 1의 입력 변수 AccessRange와 RegionSize를 살펴보면, AccessRange 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 영역번호를 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. Q_h_Size와 Q_w_Size는 2Q-CFP 기법에서 Q_h와 Q_w가 사용할 수 있는 캐쉬 슬롯의 수를 의미한다. Q_h와 Q_w 큐가 사용할 수 있는 캐쉬 슬롯의 비율을 70%대 30%의 비율로 설정했는데, 이는 보통 이 비율로 설정하는 것이 우수한 성능을 보이기 때문이다. 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, AccessRange 내의 페이지 중에서 임의의 페이지가 선정된다. zipf 부하에서 특별한 언급이 없는 한, Noise는 0으로 설정되어 있으며 θ 값은 0.95로 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 하나의 페이지를 액세스하는데 걸리는 평균 시간을 의미한다. 실험은 복사 방법(replication approach)을[19] 사용하였는데, 실험 시작시의 초기 편향(initial bias)을 제거하기 위하여 초기 4000개의 페이지 액세스의 결과는 무시하였다. 이 절에서 제시된 결과 값은 5개의 서로 다른 임의 수를 사용하여 실시된 실험 결과의 평균값으로, 각 실험은 50000 개의 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 페이지를 액세스하는데 걸리는 평균 응답시간의 결과가 그림 4에 제시되어 있다. 그리고 각 캐쉬 용량에서 2Q-CFP 기법의 평균 응답시간을 1로 간주할 때, 각 기법의 상대적인 평균 응답시간의 결과가 그림 5에 제시되어 있다. 이는 각 기법간의 상대적인 성능 결과를 좀 더 명확히 파악하기 위함이다. 클라이언트에 캐쉬가 존재하지 않으면, 모든 기법들은 2500 tick 정도의 응답시간을 보인다. 이는 전체 데이타베이스가 5000개의 페

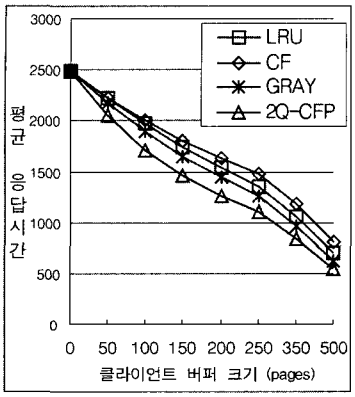

그림 4 평균 응답시간(ticks)

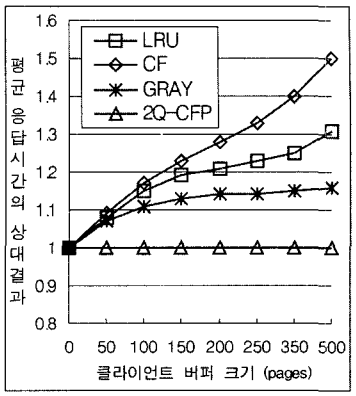

그림 5 응답시간의 상대결과

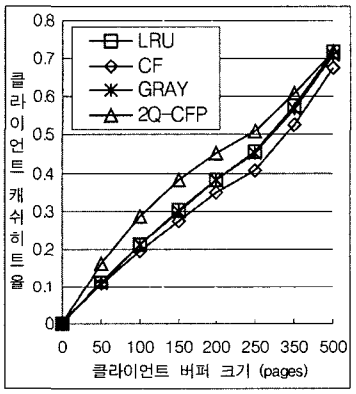

그림 6 클라이언트 캐쉬히트율

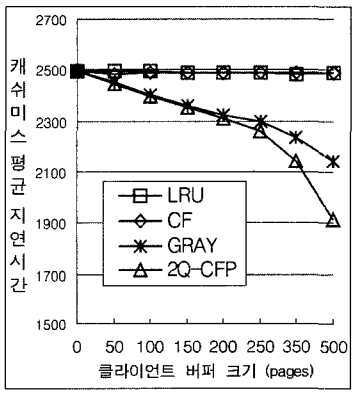

그림 7 캐쉬미스시의 평균 지연시간(ticks)

이지로 구성되고 이들이 주기적으로 방송되므로, 클라이언트가 한 페이지를 액세스하기 위해서는 평균적으로 2500 tick을 기다려야 하기 때문이다. 캐쉬 용량이 늘어남에 따라 각 기법은 더 높은 캐쉬히트율을 보일 수 있기 때문에 평균 응답시간은 점점 줄어들게 된다. 그리고 각 기법간에 존재하는 캐쉬 관리 정책의 차이로 인하여 평균 응답시간의 성능 차이가 발생하기 시작하는데, 2Q-CFP, GRAY, LRU 그리고 CF 기법순으로 우수한 성능을 발휘한다. 그림 5에서 보는 것과 같이 캐쉬 용량이 250 페이지가 되면, 2Q-CFP 기법은 GRAY, LRU, CF 기법에 비해, 대략 14%, 23%, 33% 정도의 성능 향상을 가져온다.

평균 응답시간에서 각 기법간의 성능 차이는 대부분 그림 6, 7에 제시된 캐쉬히트율과 캐쉬미스로 인한 평균 지연시간의 차이에서 비롯된다. 클라이언트가 어떤 페이지를 액세스할 때 캐쉬미스가 발생하면, 클라이언트는 그 페이지가 방송될 때까지 기다려야 한다. 캐쉬미스시의 평균 지연시간이란 이 대기시간을 의미한다. 2Q-CFP 기법은 다른 기법에 비해 훨씬 우수한 캐쉬히트율을 보이며, 캐쉬미스시의 평균 지연시간에 있어서도 우수한 성능을 보인다. 따라서 2Q-CFP 기법은 다른 기법에 비해 매우 우수한 응답시간의 성능을 보일 수 있는 것이다.

CF 기법은 과거의 액세스 형태를 완전히 무시하고 방송 순서만을 고려하여 희생자를 선정한다. 이는 캐쉬히트율 측면에서는 무작위로 희생자를 선정하는 것과 동일하기 때문에, CF 기법은 다른 기법에 비해 낮은 캐쉬히트율을 보인다. LRU 기법은 각 페이지의 최종 액세스 순서를 고려하여 희생자를 선정하기 때문에 CF 기법보다는 높은 캐쉬히트율을 보인다. 그러나, LRU 기법에서 일단 한번 액세스된 페이지는 LRU 체인의 맨 앞에 배치되어 맨 뒤에 올 때까지는 희생자로 선정되지 않는다. 이는 임의로 우연히 액세스되는 페이지들도 오랫동안 캐쉬 슬롯을 차지하고 있어야 한다는 것을 의미한다. GRAY 기법은 LRU 기법과 거의 유사한 단점이 있다. GRAY 기법에서 일단 한번 액세스된 페이지는

무조건 흑색 페이지가 되고, 흑색 페이지는 자주 액세스되는 페이지로 간주되어 현 주기가 끝날 때까지는 희생자로 선정되지 않는다. 그리고 새로운 주기가 시작될 때, 흑색 페이지는 회색으로 상태가 변하여 지속해서 프리페치의 대상이 된다. 따라서 임의로 액세스된 페이지들도 두 주기에 걸쳐서 캐쉬 슬롯을 차지할 수 있다는 단점이 있다. GRAY 기법은 LRU보다 약간 낮기는 하지만 거의 유사한 캐쉬히트율을 보인다.

2Q-CFP 기법은 데이타베이스의 페이지들을 hot, warm, 그리고 cold 페이지로 동적으로 구분하여 관리하기 때문에 다른 기법보다는 자주 액세스되는 페이지와 그렇지 않은 페이지를 좀 더 확실하게 구분할 수 있다. 2Q-CFP 기법에서 우연히 한번 액세스된 페이지들은 hot 페이지로 격상되지 못하고 warm 페이지까지 격상되었다가 다시 cold 페이지로 격하된다. 그리고 클라이언트가 cold 페이지를 액세스하면, 2Q-CFP 기법은 액세스 확률이 그다지 높지 않은 warm 페이지 중에서 희생자를 선정한다. 이와 같이 가끔씩 액세스되는 페이지들은 Q_w에서만 관리되다가 희생자로 선택되므로 비교적 빨리 캐쉬에서 제거될 수 있다. 그러므로 2Q-CFP는 다른 기법보다 훨씬 우수한 캐쉬히트율을 보인다.

그림 6에서 캐쉬 용량이 200 페이지 이상이 되면 각 기법간의 캐쉬히트율의 차이가 감소하기 시작하여, 캐쉬 용량이 500 페이지가 되면 CF 기법을 제외한 다른 기법들은 거의 유사한 캐쉬히트율을 보인다. 그럼에도 불구하고 평균 응답시간에 있어서 각 기법간에 성능 차이가 나는 이유는 그림 7의 캐쉬미스시의 평균 지연시간과 밀접한 관련이 있다. 그림 7을 살펴보면, 캐쉬미스시의 평균 지연시간에 있어서 프리페치 기법을 적용한 2Q-CFP와 GRAY 기법은 다른 기법에 비해 훨씬 우수한 성능을 발휘한다. 직관적으로는 CF 기법이 캐쉬미스시의 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.

2Q-CFP와 GRAY 기법은 유사한 프리페치 기법을 적용하고 있지만, 프리페치 대상이 되는 페이지의 관리에 있어서 차이가 있다. 2Q-CFP 기법에서는 Q_w에서 관리되는 warm 페이지들이 프리페치 대상이 되며, GRAY에서는 이전 주기에 액세스된 회색 페이지들이 프리페치의 대상이 된다. 이를 프리페치 대상이 되는 페이지 수의 관점에서 비교해 보면, 2Q-CFP 기법에서는 항상 *0.8 x CacheSize* 개의 페이지들이 Q_w에 할당된 *0.3 x 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 무렵에는 프리페치 대상이 되는 회색 페이지들이 대부분 이미 캐싱되어 있고, 주기의 끝 무렵에는 회색 페이지들이 사용할 수 있는 캐쉬 슬롯의 수가 극히 제한되기 때문에 2Q-CFP에 비해서 프리페치의 효과가 적은 편이다. 또한 프리페치 대상이 되는 페이지들의 교체에는 다음과 같은 차이점이 있다. GRAY 기법은 주기의 끝에서 모든 회색 페이지들을 흰색 페이지로 변경하여 프리페치 대상에서 한꺼번에 제외하며, 모든 흑색 페이지를 회색 페이지로 변경하여 프리페치 대상에 편입시킨다. 반면, 2Q-CFP 기법에서는 클라이언트가 데이타를 액세스함에 따라 프리페치 대상이 될 수 있는 페이지들이 지속적으로 변경된다. 따라서 프리페치 대상이 되는 페이지의 교체에 있어서도 2Q-CFP 기법은 GRAY보다 바람직하다고 볼 수 있다. 이와 같은 이유로 인하여 2Q-CFP 기법은 GRAY보다 캐쉬미스시의 평균 지연시간에 있어서 우수한 성능을 보인다.

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, 9에 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, 2Q-CFP 기법은 평균 응답시간에 있어서 다른 기법에 비하여 항상 우수한 성능을 보인다. 이는 그림 9와 10에 제시되어 있는 캐쉬히트율과 캐쉬미스시의 평균 지연시간과 밀접한 연관이 있다.

Noise 확률이 증가하면 클라이언트는 점점 더 무작위로 데이타를 액세스하기 때문에 각 기법의 캐쉬히트율은 급격하게 감소한다. 그리고 Noise 확률이 증가함에 따라 각 기법간에 존재하는 캐쉬 관리 정책의 차이가 캐쉬히트율에 미치는 영향력이 줄어들게 된다. 따라서 각 기법간의 캐쉬히트율의 차이도 감소한다. 극단적으로

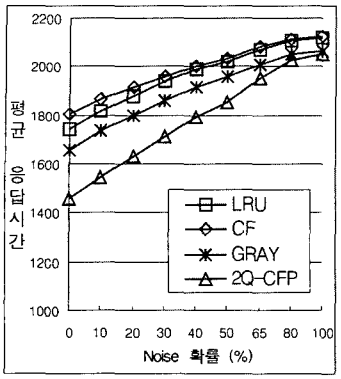

그림 8 평균 응답시간(ticks)

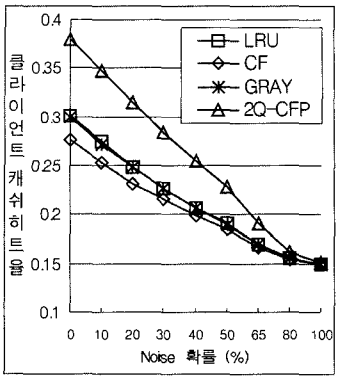

그림 9 클라이언트 캐쉬히트율

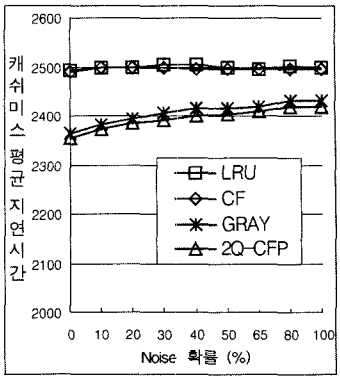

그림 10 캐쉬미스시의 평균 지연시간

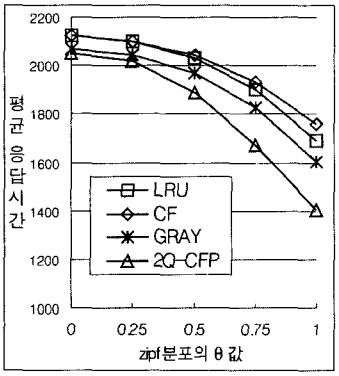

그림 11 평균 응답시간(ticks)

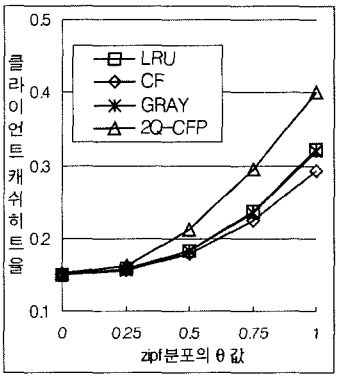

그림 12 클라이언트 캐쉬히트율

Noise 확률이 100%가 되면, 클라이언트는 Access-Range내의 모든 페이지를 균등하게 액세스하게 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 2Q-CFP와 GRAY 기법은 다른 기법에 비하여 우수한 평균 응답시간의 성능을 보인다. 이는 그림 10에 제시된 캐쉬미스시의 평균 지연시간의 차이에서 비롯된다. Noise 확률이 증가하면 캐쉬미스시의 평균 지연시간도 증가한다. 이는 Noise 확률의 증가에 따라 프리페치 대상에 속하지 않는 페이지들의 액세스 확률도 증가하여 전체적으로 캐쉬미스로 인한 지연시간을 증가시키기 때문이다.

5.3 θ 값의 변화에 따른 성능

본 실험은 zipf 부하에서 θ 값을 변경시키면서 각 기법의 성능 추이를 살펴보았다. 본 실험에서 클라이언트의 캐쉬 용량은 150개의 페이지를 캐싱할 수 있는 것으로 설정하였다. 클라이언트의 평균 응답시간과 캐쉬히트율이 각각 그림 11, 12에 제시되어 있다.

zipf 분포에서 클라이언트 액세스의 불균형의 정도는 0.0에서 1.0까지 변화되는 θ 값에 의하여 결정되는데, 이 값이 1.0에 가까울수록 불균등 정도가 큰 특성을 가진다. 만일 θ 값이 0이 되면, 각 영역의 액세스 확률이 동일하게 되어 클라이언트가 액세스하는 페이지들의 분포는 균등(uniform) 분포를 이루게 된다. 따라서 이 경우에 각 기법은 거의 동일한 캐쉬히트율을 보인다. 그러나 θ 값이 증가함에 따라 각 영역들의 액세스 확률의 차이가 점점 커지게 된다. 즉, 자주 액세스되는 페이지와 그렇지 않은 페이지들이 점점 더 뚜렷하게 구분된다. 따라서 θ 값이 증가함에 따라 각 기법의 캐쉬히트율은 급격하게 증가하게 되는데, 그 증가 속도에 있어서 2Q-CFP 기법은 다른 기법에 비해서 훨씬 빠르다. 이는 2Q-CFP는 데이타베이스의 페이지들을 hot, warm, 그리고 cold 페이지로 동적으로 구분하여 관리하기 때문에 다른 기법에 비해 자주 액세스되는 페이지와 그렇지 않은 페이지를 좀 더 확실하게 구분할 수 있고, 이를 희생자 선정 정책에 반영하고 있기 때문이다. 따라서 이 실험에서도 2Q-CFP 기법은 다른 기법에 비하여 우수한 성능을 보

인다. 특히, 데이타 액세스의 불균형의 정도가 심해질수록 2Q-CFP는 다른 기법에 비해 더 뚜렷한 성능 향상을 가져온다.

6. 결 론

방송 방식에 기초한 데이타 전달 기법은 정보소비 모델 형태의 정보제공 응용 및 이동 컴퓨팅 환경의 응용에서 정보 전달을 위한 주요 기술로 인정을 받고 있다. 본 논문에서는 이와 같은 방송 기술을 채택한 데이타 방송 시스템의 성능을 향상시키기 위하여 클라이언트 측에 적용 가능한 새로운 캐쉬 관리 기법인 2Q-CFP(two queue with closest-first and prefetch) 기법을 제안했다. 그리고 모의 실험을 통하여 2Q-CFP 기법의 성능과 GRAY, LRU 및 CF 기법의 성능을 비교하였다. 2Q-CFP 기법은 데이타베이스 내의 페이지를 hot, warm, 그리고 cold 페이지로 동적으로 구분하여 관리하고 세분화된 희생자 선정 정책을 사용하기 때문에 다른 기법에 비하여 높은 캐쉬히트율을 보인다. 그리고 2Q-CFP 기법은 슬롯 희생자를 선정할 때 방송 채널에 가장 먼저 나타날 warm 페이지를 희생자로 선정하는 정책을 취하고 있고, 데이타 방송이 진행되더라도 방송 지연시간이 긴 warm 페이지들이 캐쉬되도록 프리페치 기법이 적용되어 있다. 따라서 2Q-CFP 기법은 캐쉬 미스로 인한 평균 지연 시간에서도 다른 기법에 비하여 우수한 성능을 보인다. 이와 같은 이유의 조합으로 인하여 2Q-CFP 기법은 방송 시스템의 주요 성능평가 기준이 되는 평균 응답시간에 있어서 매우 우수한 성능을 보인다.

본 논문의 미래 연구 과제로서 방송 스케줄링 기법과 클라이언트 캐쉬 관리 기법이 조합된 연구를 진행해 나갈 것이다. 그리고 클라이언트에서 서버로의 상향링크(uplink)가 존재하는 환경에서 클라이언트의 요청을 바탕으로 방송할 데이타를 동적으로 변화시키는 방송 스케줄링 기법에 대한 연구를 진행할 예정이다.

참 고 문 헌

[1] K. Stathatos, "Air-Caching: Adaptive Hybrid Data Delivery," Ph.D. Thesis, Maryland University, 1999.

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

[3] M. Franklin and S. Zdonik, "A Framework for Scalable Dissemination-Based Systems," In the International Conference on Object-Oriented Prog ramming Languages, Systems, and Applications (OOPSLA 97), pp. 94-105, Atlanta, GA, October 1997.

[4] M. Franklin and S. Zdonik, "Data in Your Face: Push Technology in Perspective," Proceeding of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 516-519, Seattle, WA, June 1998.

[5] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric communications environments," Proceeding of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 199-210, San Jose, CA, May 1995.

[6] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast," Proceeding of the 1997 ACM SIGMOD International Conference on Management of Data, pp. 183-194, Tucson, Arizona, May 1997.

[7] D. Aksoy and M. Franklin, "Scheduling for Large Scale On-Demand Data Broadcasting," Proceeding of IEEE INFOCOM, pp. 651-659, San Francisco, CA, March 1998.

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

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

[10] K. Stathatos, N. Roussopoulos, and J. S. Baras, "Adaptive Data Broadcast in Hybrid Networks," Proceeding of the 23th International Conference on Very Large Data Bases, pp. 326-335, Athens, Greece, Aug. 1997

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

[12] 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 Distributed File System," ACM Transactions on Computer Systems, Vol. 6, No. 1, pp. 51-89, Feb. 1988.

[13] A. Tanenbaum, 'Modern Operating Systems,' Prentice Hall, 1992.

[14] E. J. ONeil, P. E. ONeil, and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," Proceeding of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 297-306, Washington, DC, May 1993.

[15] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," Proceeding of the 20th Inter-

national Conference on Very Large Data Bases, pp. 439-450, Santiago, Chile, Sep. 1994.
[16] C. J. Su, L. Tassiulas, V. Tsotras, "Broadcast scheduling for information distribution," Wireless Networks, Vol. 5, pp. 137-147, 1999.
[17] M. H. Ammar, "Response Time in a Teletext System: An Individual User's Perspective," IEEE Transactions on Communication COM-35, 11, pp. 1159-1170, Nov. 1987.
[18] H. Schwetman, CSIM Users' Guide for Use with CSIM Revision 16, Microelectronics and Computer Technology Corporation, 1992.
[19] A. M. Law and W. D. Kelton, Simulation Modeling & Analysis, McGraw-Hill, 1991.

권 혁 민

1984년 서울대학교 제어계측공학과 학사 1994년 한국과학기술원 정보및통신공학과 석사. 1998년 한국과학기술원 정보및통신공학과 박사. 1984년~1991년 대우전자 중앙연구소 컴퓨터개발부 선임연구원. 1999년~현재 세명대학교 소프트웨어학과 조교수. 관심분야는 트랜잭션 처리, 분산/병렬 데이타베이스, 이동 컴퓨팅