정보학연구 제5권 제4호 2002년 12월(pp.9-21)
The Journal of Information Technology, Vol.5, No.4

데이타 방송 환경을 위한 클라이언트 캐쉬 관리 기법의 성능 평가

권혁민*

요 약

데이타 방송 환경에서 서버는 방송 채널을 통하여 데이타베이스내의 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 2Q-CF로 명명된 새로운 클라이언트 캐쉬 관리 기법을 제안하고, 모의 실험 모델을 통하여 2Q-CF 기법의 성능을 평가한다. 성능 평가 결과에 의하면 2Q-CF 기법은 LRU-CF와 CF보다 평균 응답시간에 있어서 더 우수한 성능을 보인다.

Ⅰ. 서론

최근 들어 인터넷의 폭발적인 인기와 함께 전 세계는 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 검색하고 이를 이용하는 다수의 클라이언트로 구성되는 정보소비(information-consumption) 모델의 형태를 띠고 있다[15].

서버에서 클라이언트로 정보를 전달하는 방법에는 기본적으로 요청-응답(request-response) 방식과 데이타 방송(data broadcast) 방식이 있다[3, 6, 7]. 요청-응답 방식에서는 클라이언트가 서버에게 원하는 데이타를 요청하면 서버가 그 데이타를 전송하여 응답하는 방식이다. 반면, 데이타 방송 기법은 클라이언트에서 명시적인 데이타 요청이 없어도 서버가 클라이언트에서 필요로 하는 데이타를 예상하여 전송하는 방식이다. 요청-응답 방식에서는 다수의 클라이언트로부터 일부 인기있는 데이타에 대한 중복 요청 및 서버의 중복 응답으로 인하여 서버 및 네트워크 자원이 심하게 낭비되는 경향이 있다. 이는 서비스를 제공하는 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. 따라서 시스템의 확장성(scalability)에 심각한 문제를 야기할 수 있다.

데이타 방송 기법에서 방송 서버는 다수의 클라이언트에게 특정 데이타들을 지속적으로 전파하고 각 클라이언트는 자신이 원하는 데이타가

* 세명대학교 소프트웨어학과 조교수

방송채널에 나타나면 이를 검색한다. 데이타 방송 기법은 클라이언트의 수가 증가하더라도 시스템의 성능에 영향을 받지 않기 때문에 확장성 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 적합한 기술이다. 그러나 데이타 방송 기법에서 클라이언트가 특정 데이타를 검색하기 위해서는 해당 데이타가 방송 채널에 나타나기를 기다려야 한다. 데이타 방송 기법에서 발생하는 이런 필연적인 클라이언트의 대기시간은 클라이언트의 응답시간에 큰 영향을 미치기 때문에, 이 대기시간을 줄이기 위한 많은 연구들이 수행되었다[1-4, 10, 11, 15].

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

요청-응답 방식에서는 모든 데이타에 대해 캐쉬 미스로 인해 발생되는 지연시간이 거의 일정하다. 그러므로 캐쉬 관리 기법에 대한 연구가 캐쉬 히트율을 높이는 방향으로만 진행되어 왔다. 그러나 데이타 방송 기법에서는 데이타가 언제 방송 채널에 나타나는가에 따라 캐쉬 미스를 처리하기 위한 비용이 각 데이타마다 상당한 차이를 보이는 특징이 있다. 본 논문에서는 이 점을 고려하여 데이타 방송 기법에 적합한 새로운 캐쉬 관리 기법인 2Q-CF(two queue with closest-first) 기법을 제안한다. 2Q-CF 기법은 자주 액세스되는 페이지를 관리하는 Q_h 큐와 가끔씩 액세스되는 페이지를 관리하는 Q_w 큐를 운영한다. 새로운 페이지를 캐쉥할 필요가 있어 희생자가 필요할 경우에는 Q_w에서 관리되는 페이지중 방송 채널에서 가장 먼저 나타날 페이지를 희생자로 선정한다. 따라서 2Q-CF 기법은 캐쉬 미스로 인하여 야기되는 평균 지연시간을 줄이는 것이 가능하다. 뿐만 아니라, 2Q-CF 기법은 자주 액세스되는 페이지와 그렇지 않은 페이지를 동적으로 구분하여 관리하기 때문에 캐쉬 히트율이 높은 장점이 있다. 이와 같은 이유로 인해 2Q-CF 기법은 데이타를 액세스하는데 소요되는 평균 응답 시간의 성능을 상당히 향상시킬 수 있다.

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

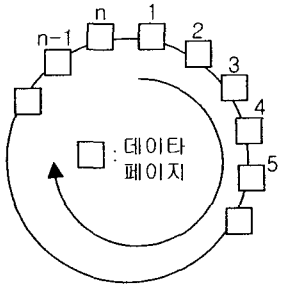
II. 관련 연구

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

2.1 데이타 방송 시스템 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 시스템 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 디스크에서는 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 방송빈도를 결정한다. 만일 데이타들의 액세스 확률을 미리 파악할 수 없는 경우에는 모든 데이타들을 하나의 디스크에 저장할 수밖에 없는데, 이 경우 모든 데이타들의 방송 빈도수는 동일하다. 이를 평형 방송 디스크(flat broadcast disk) 모델이라 한다.

본 논문에서는 클라이언트의 액세스 패턴을 미리 파악하기가 어렵거나 불가능하다고 가정하고 평형 방송 디스크 모델하에서 클라이언트 캐쉬 관리 기법을 개발한다. 평형 방송 디스크 모델에서는 그림 1과 같이 데이타베이스의 모든 페이지들을 일정한 순서로 주기적으로 방송 채널을 통하여 전파한다. 본 논문에서는 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 논문에서는 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 쉽게 본 논문에서 제안하는 캐쉬 관리 기법에 반영할 수 있다.



(그림 1) 평형 방송 디스크 모델

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) 기법은[16] 구현이 간단하여 가장 많이 사용되는 기법이다. LRU 기법은 캐쉬에 저장된 페이지들의 우선순위를 관리하기 위하여 LRU 체인을 유지한다. 어떤 페이지가 액세스되면 그 페이지는 체인의

맨 앞에 배치되고, 희생자가 필요할 때는 체인의 맨 뒤에 있는 페이지를 희생자로 선택하여 캐쉬에서 제거한다. 비록 LRU 기법이 과거의 액세스 패턴을 반영하여 희생자를 선택하기는 하지만, 이 기법은 각 페이지의 최종 액세스 순서만을 고려하기 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 캐쉬 히트율이 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 캐쉬 히트율을 보이는 장점이 있지만, 구현하는데 있어 높은 실행 부담을 보이는 단점이 있다.

[9]의 연구는 LRU 기법과 유사한 실행 부담을 가지면서도, LRU-2에 필적하는 캐쉬 히트율을 보일 수 있는 2Q(two queue) 기법을 제안했다. 2Q 기법은 자주 액세스되는 페이지와 그렇지 않은 페이지를 구분하여 관리하기 위하여 두 개의 큐, A_m과 A_1 큐를 운영한다. 2Q 기법의 기본적인 알고리즘은 다음과 같다. 어떤 새로운 페이지가 액세스되면 A_1 큐에 삽입되고, A_1 큐에 있는 페이지가 다시 액세스되면 A_m 큐로 옮겨진다. A_1 큐는 FIFO 형태로 관리되고, 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 기법에서는 우연히 한번 액세스된 페이지는 보통 A_1 큐에서 관리되다가 희생자로 선택되기 때문에 LRU 기법보다 우수한 캐쉬 히트율을 보일 수 있다.

LRU, LRU-k, 그리고 2Q 기법은 과거의 액세스를 바탕으로, 추후 가장 액세스되지 않을 것으로 예상되는 페이지를 희생자로 선택한다. 그러나 이들은 캐쉬 미스로 인하여 야기되는 지연 시간이 각 페이지마다 상이한 방송 환경의 특수성이 고려되지 않은 기법들이다. 방송 환경의 특수성을 고려한 기법에는 CF(closest first), GRAY, PIX 및 LIX 기법들이 있다. CF 기법은 [10, 11] 희생자가 필요할 경우에 캐싱되어 있는 페이지 중에서 다음에 가장 먼저 방송 채널에 나타나는 페이지를 희생자로 선정한다. 따라서 CF 기법은 캐쉬 미스로 인한 평균 지연 시간을 줄일 수 있다. 그러나 이 기법은 과거의 액세스 형태를 고려하지 않고 희생자를 선택하기 때문에 자주 액세스되는 페이지나 그렇지 않은 페이지나 희생자로 선택될 확률이 동일하다. 그러므로 이 기법은 캐쉬 히트율이 낮은 단점이 있다.

[10]의 연구는 CF(closest-first) 기법과 마킹 알고리즘(marking algorithm)으로 알려진 1-bit LRU 기법을 통합하여 1-bit LRU-CF(이후 LRU-CF라 칭함) 기법을 제안했다. 시스템의 초기에는 캐쉬내의 모든 페이지들이 마킹되어 있지 않다. 클라이언트가 어떤 페이지를 액세스하면 그 페이지는 버퍼에 캐싱되며 마킹된다. LRU-CF 기법에서 희생자가 필요하면, 마킹되지 않은 페이지중 가장 먼저 방송 채널에 나타날 데이타를 희생자로 선정하여 캐쉬에서 제거한다. 언젠가는 캐쉬내의 모든 페이지들이 마킹되어, 더 이상은 희생자 선정이 가능하지 않게 된다. 이 상태가 되면, LRU-CF 알고리즘의 한 주기가 끝나게 된다. 주기의 끝에서 모든 마킹

된 페이지는 마킹되지 않은 상태로 변경되며 새로운 주기가 시작된다. LRU-CF 기법은 방송 스케쥴을 고려하여 희생자를 선정하기 때문에 캐쉬 미스시의 평균 지연시간을 줄일 수 있다. 그러나 일단 한번 마킹된 페이지는 현 주기가 끝날 때까지는 희생자로 선정되지 않는다. 따라서 임의로 우연히 액세스된 페이지들도 현 주기의 끝까지 오랫동안 버퍼에 캐싱되어 있어야 하며, 다음 주기에서도 어느 정도의 기간동안 캐쉬 슬롯을 차지하는 단점이 있다. [10, 11]의 연구에서는 LRU-CF 기법에 프리페치 기법을 적용하여 GRAY 기법을 제안했다.

[1, 3]의 연구는 다중 방송 디스크 환경에서 캐쉬 미스의 처리 비용에 근거한 캐쉬 관리 기법인 PIX와 LIX를 개발했다. 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. PIX 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 희생자가 필요할 때마다 각 LRU 체인의 끝에 있는 페이지중 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.

III. 새로운 클라이언트 캐쉬 관리 기법: 2Q-CF

이 절에서는 본 논문에서 새로이 제시되는 2Q-CF(two queue with closest-first) 기법을 기술한다. 2Q-CF 기법은 빈번하게 액세스되는 페이지들을 관리하는 Q_h 큐와 가끔씩 액세스되는 페이지들을 관리하는 Q_w 큐를 운영한다. 본 논문은 Q_h에서 관리되는 페이지를 hot 페이지라고 명명하고, Q_w에서 관리되는 페이지를 warm 페이지라 명명한다. 그리고 최근에 한번도 액세스되지 않아 Q_h와 Q_w에서 관리되고 있지 않은 페이지를 cold 페이지라 명명한다. 클라이언트가 cold 페이지를 액세스하면 그 페이지는 warm 페이지로 격상되어 Q_w에서 관리된다. 그리고 클라이언트가 warm 페이지를 다시 액세스하면 그 페이지는 hot 페이지로 격상되어 Q_h에서 관리된다. Q_h와 Q_w 큐는 전체 캐쉬 슬롯을 적절하게 나누어 사용하는데, Q_h와 Q_w 큐가 사용할 수 있는 캐쉬 슬롯의 비율은 클라이언트의 액세스 형태를 고려하여 조정할 수 있는 변수로서 성능 튜닝을 위해 적절하게 설정해야 한다.

본 논문에서는 클라이언트에 캐싱된 페이지의 무효화가 발생하지 않는다. 따라서 시스템이 초기 전이(transient) 상태를 벗어나게 되면, Q_h와 Q_w에는 항상 자신들이 관리할 수 있는 최대의 큐 엔트리들이 존재하고 빈 캐쉬 슬롯은 존재하지 않게 된다. 본 논문에서는 알고리즘 기술의 간단성을 위해 시스템이 정상 상태에 도달했다

고 가정하고 알고리즘을 기술한다. 2Q-CF 기법에서 Q_h는 LRU 기법과 유사하게 관리된다. 따라서 Q_h에 새로운 페이지가 삽입될 경우에는 Q_h의 LRU 체인상 맨 뒤에 있는 페이지가 희생자로 선정된다. Q_h에서 희생자로 선택된 페이지는 warm 페이지가 되어 Q_w에서 관리된다. Q_h에 어떤 페이지가 삽입된다는 것은 그 페이지는 warm 페이지로서 이미 캐싱되어 있던 페이지가 다시 캐쉬 히트가 발생하여 hot 페이지로 격상되는 것을 의미하는 것으로 새로운 페이지가 캐싱되는 것은 아니다. 따라서 Q_h에서 희생자로 선택된 페이지는 단순히 warm 페이지로 격하되는 것이지 캐쉬 슬롯에서 제거될 필요는 없다. Q_w 큐는 CF(closest-first) 형태로 운영된다. 만일 클라이언트가 cold 페이지를 액세스하면, 해당 페이지를 새로이 캐싱하기 위하여 희생자가 필요하다. 이 경우에 Q_w에서 관리되는 페이지중 방송 채널에서 가장 먼저 방송될 페이지가 희생자로 선정되어 버퍼에서 제거된다.

2Q-CF 기법에서 큐 관리를 위한 알고리즘은 다음과 같다. 클라이언트가 hot 페이지를 다시 액세스하면 그 페이지는 Q_h의 LRU 체인의 맨 앞으로 이동한다. 클라이언트가 warm 페이지를 액세스하면, 그 페이지의 큐 엔트리는 Q_h의 맨 앞으로 이동되어 관리된다. 그리고 Q_h의 맨 뒤에 있는 엔트리는 Q_w로 이동되어 관리된다. 클라이언트가 cold 페이지 cp_l을 액세스하면, Q_w 큐가 관리하고 있는 warm 페이지중 방송 채널에서 가장 먼저 방송될 페이지를 희생자로 선정하여 캐쉬에서 제거하고 cp_l을 캐쉬에 저장한다. cp_l은 cold 페이지에서 warm 페이지로 격상되어 Q_w에서 관리된다. 2Q-CF 기법은 hot 페이지와 warm 페이지를 구분하여 관리하기 위하여 두 개의 큐를 운영한다는 점에서 2Q 기법과 유사한 측면이 있다. 그러나 2Q 기법에서는 A_1 큐가 FIFO 형태로 관리되는데 비하여 2Q-CF 기법에서는 Q_w 큐가 방송 환경의 특수성을 고려하여 CF 형태로 운영된다는 점에서 다르다. 그리고 희생자 선정 정책에 있어서, 2Q 기법에서는 hot 페이지가 직접 cold 페이지로 격하될 수 있는데 비해 2Q-CF 기법에서는 hot 페이지는 warm 페이지를 거쳐서 cold 페이지로 격하된다는 점에서 다르다. 2Q-CF 기법에서 특정 페이지 p를 액세스할 때의 자세한 과정은 알고리즘 1에 있다.

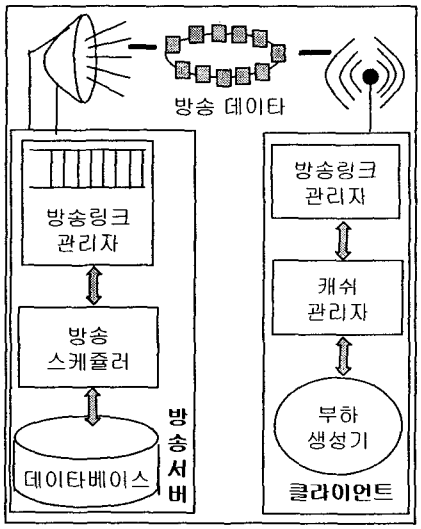
알고리즘 1: 클라이언트가 페이지 p를 액세스할 경우

```
if p is hot page in Qh then
        move p's entry on the front of Qh
else if p is warm page in Qw then
        move p's entry from Qw to the front of Qh
        move the tail of Qh to Qw
else //cold page request
        wait for p's arrival on broadcast channel
        find the closest-first warm page, wpi
        free the cache slot used by wpi
        delete the entry of wpi on Qw
        put p in the freed cache slot
        insert p's entry into Qw
end if
```

IV. 성능 평가 모델

이 절에서는 본 논문에서 제안된 캐쉬 관리

기법의 성능을 평가하기 성능 평가 모델을 제시한다. 비교 대상으로는 CF, LRU-CF 기법을 선정하였다. 성능 평가 모델은 [1, 3, 10]의 연구를 참고로 하고 있고, MCC에서 개발한 CSIM[14] 언어를 이용하여 구현하였다.



(그림 2) 성능 평가 모델

본 성능 평가 모델은 크게 방송 서버와 클라이언트로 구성된다. 방송 환경의 특성상 한 클라이언트의 성능은 다른 클라이언트들의 존재 여부에 영향을 받지 않기 때문에 본 논문에서는 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 이를 액세스한다. 해당 페이지가 자신의 지역 캐쉬에 있으면 그 데이타를 액세스하고 그렇지 않으면 방송 채널에 그 데이타가 방송될 때까지 대기했다가 액세스한다. 부하 생성기는 요구한 페이지를 액세스할 때까지는 새로운 페이지의 액세스를 요청하지 않는다. 클라이언트의 캐쉬 관리자는 성능 평가 대상이 되는 각 기법에 맞게 클라이언트의 캐쉬 관리를 책임진다.

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 1에 제시되어 있는데, 이 변수들의 설정값은 대부분 [1, 3, 10]의 연구를 따르고 있다.

<표 1> 성능 평가 모델의 입력 변수

입력변수	의미	설정
Server DbSize	데이타베이스의 크기	5000 pages
Cache Size	클라이언트 캐쉬 크기	0~500 pages
Qh_Size	Qh가 사용하는 캐쉬 슬롯의 수	70 % of CacheSize
Qw_Size	Qw가 사용하는 캐쉬 슬롯의 수	30 % of CacheSize
ThinkTime	클라이언트가 페이지를 액세스한 후에 다음 액세스 요청까지의 시간 간격	2 ticks
Access Range	클라이언트가 액세스하는 페이지 범위	1000 pages
RegionSize	zipf 분포에서 동일한 액세스 확률을 갖는 페이지의 수	50
NoiseProb	zipf 분포를 벗어날 확률	0 ~ 50 %
θ	zipf 분포의 θ 값	0.95

서버의 데이타베이스는 ServerDbSize 수의 페이지로 구성되며, 서버는 이 페이지들을 주기적으로 방송한다. 클라이언트는 이 페이지들을 모두 액세스하는 하는 것이 아니라, 전체 데이타베이스의 일정 범위만을 액세스하는데, 이 일정 범위를 AccessRange로 정의된다. 즉 클라이

언트는 1~AccessRange 사이의 페이지만을 액세스한다. 본 논문에서는 불균등(non-uniform)한 액세스 형태를 모델링하기 위하여 많이 사용되는 zipf 분포 부하 모델[1, 9, 10]에서 각 기법의 성능을 비교 평가한다. 본 논문의 zipf 부하 모델은 클라이언트가 액세스하는 AccessRange의 페이지들을 RegionSize 수 만큼의 페이지로 나누어 다수의 영역(region)으로 구분하는데, 각 영역은 서로 겹쳐지지는 않는다. 따라서 표 1의 입력 변수 AccessRange와 RegionSize를 살펴보면, 클라이언트의 AccessRange가 20개의 영역으로 구분됨을 알 수 있다. 각 영역은 1부터 시작하여 20까지의 일련번호를 갖는 영역 번호로서 구분하는데, 이 영역 번호에 zipf 분포가 적용된다. 즉, 각 영역의 액세스 확률은 $1/r^{\theta}$(r은 영역번호를 의미)에 비례한다. zipf 분포에서 클라이언트 액세스의 불균형의 정도는 0.0에서 1.0까지 변화시킬 수 있는 θ 값에 의하여 결정되는데, 이 값이 1.0에 가까울수록 불균등 정도가 큰 특성을 가진다. 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-CF 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, 그 이유는 보통 이 비율로 설정하는 것이 우수한 성능을 보이기 때문이다. ThinkTime은 클라이언트가 데이타를 액세스하고 나서 다음 페이지의 액세스를 요청하기까지의 대기 시간을 의미하는데, 클라이언트의 데이타 처리시간을 위한 것이다. NoiseProb는 클라이언트의 액세스 변화를 모델링하기 위한 것으로 클라이언트의 액세스가 zipf 부하를 따르지 않고 임의의 페이지를 액세스할 확률을 의미한다. 클라이언트의 각 액세스마다 Noise 발생 여부가 조사된다. 만일 Noise가 발생하면, 해당 액세스는 zipf 부하에 따라 페이지를 선정하지 않고, AccessRange내의 페이지중 임의의 페이지가 선정된다. zipf 부하에서 특별한 언급이 없는 한, Noise는 0으로 설정한다.

V. 성능 결과 및 분석

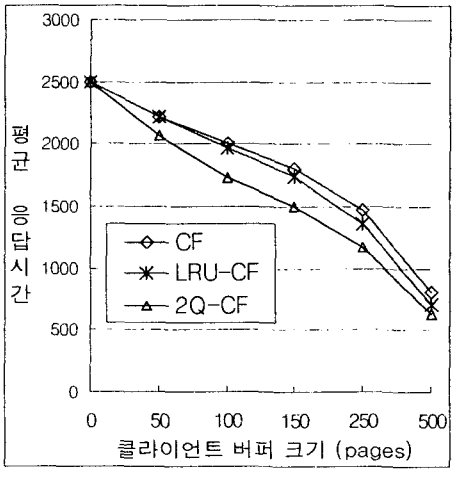
이 절에서는 4장의 성능 평가 모델을 바탕으로 성능 결과를 제시하고 분석한다. 본 논문의 주요 성능 평가 지수는 평균 응답시간으로서 특정 페이지를 액세스하는데 걸리는 평균 시간을 의미한다. 평균 응답 시간의 결과를 분석하기 위하여 보조 지수로서 캐쉬 히트율이 조사되었다. 실험은 복사 방법(replication approach)을 [12] 사용하였는데, 실험 시작시의 초기 편향(initial bias)을 제거하기 위하여 초기 4000개의 페이지 액세스의 결과는 무시하였다. 이 절에서 제시된 결과 값은 5개의 서로 다른 임의 수를 사용하여 실시된 모의 실험 결과의 평균값으로, 각 실험은 초기 시작시의 4000 개의 액세스를 제외하고 50000 개의 페이지를 액세스할 때까지 실시하였다.

5.1 실험 1: 캐쉬 크기의 변화에 따른 성능

본 실험은 클라이언트의 캐쉬 크기에 변화를

주면서 실시하였다. 클라이언트의 평균 응답시간과 캐쉬 히트율이 각각 그림 3, 4에 제시되어 있다. 클라이언트의 캐쉬가 존재하지 않을 경우에는 모든 기법들은 2500 tick 정도의 응답시간을 보인다. 이는 전체 데이타베이스의 크기가 5000 페이지이기 때문에, 클라이언트가 한 페이지를 액세스하기 위해서는 평균적으로 2500 tick을 기다려야 하기 때문이다. 캐쉬의 용량이 늘어남에 따라 각 기법은 더 높은 캐쉬 히트율을 보일 수 있기 때문에 점점 평균 응답시간은 줄어들게 된다. 또한 각 기법간에 평균 응답시간의 차이가 발생하기 시작하는데, 2Q-CF 기법은 다른 기법에 비해 우수한 성능을 발휘한다. 그림 3에서 보는 것과 같이 캐쉬 용량이 250 페이지인 경우, 2Q-CF 기법은 LRU-CF와 CF 기법에 비해, 각각 14%, 21% 정도의 성능 향상을 가져온다.

평균 응답시간에서 각 기법간의 성능 차이는 대부분 캐쉬 히트율의 차이에서 비롯된다. 2Q-CF 기법은 다른 기법에 비해 훨씬 우수한 캐쉬 히트율을 보이기 때문에, 평균 응답시간에



(그림 3) 평균 응답 시간 (ticks)

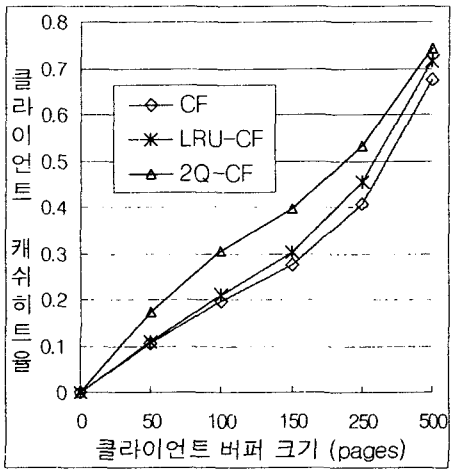


그림 4 클라이언트 캐쉬 히트율

있어서도 우수한 성능을 보이는 것이다. CF 기법은 과거의 액세스 형태를 무시하고 방송 순서만을 고려하여 희생자를 선택하기 때문에 액세스 확률이 높은 페이지들도 다른 페이지와 동일한 확률로 희생자로 선택된다. 따라서 CF 기법은 다른 기법에 비해 낮은 캐쉬 히트율을 보인다. LRU-CF 기법은 마킹(marking) 방법을 이용하여 자주 액세스되는 hot 페이지와 가끔 액세스되는 warm 페이지를 구분하고, warm 페이지중에서 희생자를 선정한다. 그렇지만 마킹이라는 방법을 통하여 hot 페이지와 warm 페이지를 확실하게 구분할 수 있는 것은 아니다. LRU-CF 기법에서 클라이언트가 어떤 페이지를 액세스하면 그 페이지는 마킹되고, 마킹된 페이지는 hot 페이지로 취급되어 희생자로 선정되지 않는다. 모든 캐쉬된 페이지들이 마킹되면 현 주기가 끝나고 마킹이 해제되는데, 이때가 되어서야 희생자로 선정될 수 있다. 이는 임의로 우연히 액세스된 페이지들도 현 주기의 끝까지 오랫동안 버퍼에 캐쉬되어 있어야 한다는 것을 의미한다. 또한 다음 주기에서도 어느 정도의 기

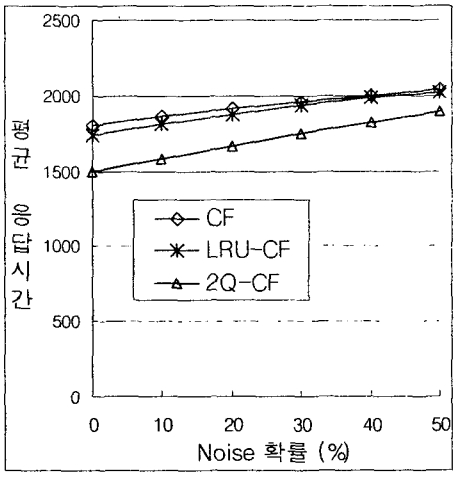
간동안 캐쉬 슬롯을 차지하고 있어야 한다. 이와 같은 이유로 인하여, LRU-CF 기법은 CF 기법보다 높은 캐쉬 히트율을 보이지만, 그 차이가 그다지 크지 않은 것이다.

반면 2Q-CF 기법은 두 개의 큐를 운영함으로 인해, LRU-CF 기법보다는 hot 페이지와 warm 페이지를 좀 더 확실하게 구분할 수 있다. 2Q-CF 기법에서는 어떤 새로운 페이지가 캐싱되면 그 페이지는 warm 페이지가 되어 Q_w에서 관리되고, warm 페이지가 다시 액세스 되면 hot 페이지가 되어 Q_h에서 관리된다. 새로운 페이지를 캐싱할 필요가 있어 희생자가 필요할 경우에는 warm 페이지중에서 선정한다. 2Q-CF 기법에서는 우연히 한번 액세스된 페이지는 Q_w에서 관리되다가 희생자로 선택되므로 LRU-CF 기법에 비해서 비교적 빨리 캐쉬에서 제거될 수 있다. 된다. 따라서 2Q-CF 기법은 LRU-CF 기법보다 우수한 캐쉬 히트율을 보일 수 있는 것이다.

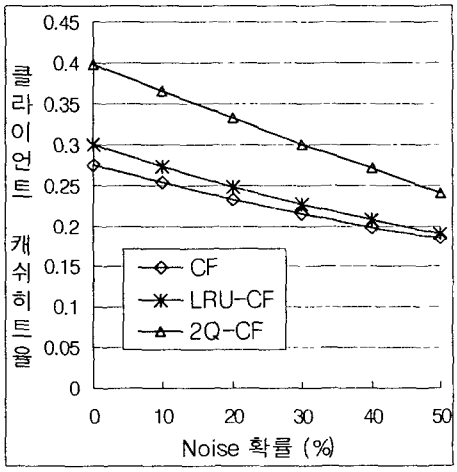
세 기법은 희생자 선정시 모두 동일한 정책을 사용한다. 즉, 희생자가 필요할 경우, 방송 채널에서 가장 먼저 방송되는 페이지를 희생자로 선정한다. 이들의 희생자 선정 정책에 있어 차이점은 희생자의 후보가 될 수 있는 페이지들이 다르다는 것이다. CF 기법에서는 모든 캐싱된 페이지들이 희생자 후보가 되고, 2Q-CF 기법에서는 Q_w에서 관리되는 페이지들이 희생자 후보가 된다. 그리고 LRU-CF 기법에서는 마킹이 되지 않은 페이지들이 희생자 후보가 된다. 이 차이점을 페이지의 수로 나타내기 위하여 전체 캐쉬 용량이 n개의 페이지를 캐싱할 수 있다고 가정해 보자. 그렇다면, CF 기법에서는 n개의 페이지들이 희생자 후보가 될 수 있고, 2Q-CF 기법에서는 0.3n 개의 페이지들이 희생자 후보가 될 수 있다. 그리고 LRU-CF 기법에서는 주기의 시작시에는 n개의 페이지가 희생자 후보가 될 수 있다. 그러나 마킹된 페이지의 수가 늘어남에 따라, 희생자 후보의 수가 점점 줄어들다가 최종적으로는 한 개의 페이지만이 희생자 후보가 될 수도 있다. 비록 희생자 후보의 수가 각 기법마다 서로 다르지만, 이들은 기본적으로 동일한 희생자 선정 정책을 취하고 있기 때문에 그 차이가 성능에 미치는 영향은 그다지 크지 않다.

5.2 Noise 확률 변화에 따른 성능

본 실험은 zipf 부하에서 Noise 확률, 즉 클라이언트의 액세스가 zipf 부하를 따르지 않을 확률을 변화시키면서 실시하였다. 그리고 본 실험에서 클라이언트의 캐쉬 용량은 150 개의 페이지를 캐싱할 수 있는 것으로 설정하였다. 클라이언트의 평균 응답시간과 캐쉬 히트율이 각각 그림 5, 6에 제시되어 있다. Noise 확률이 증가하면 클라이언트는 점점 더 무작위로 데이타를 액세스하게 된다. 따라서 각 기법의 캐쉬 히트율은 감소하게 되고 평균 응답시간은 증가하게 된다. 그리고 각 기법의 상대적인 성능 차이는 점점 들어들게 된다. 이는 Noise 확률이 증가하여 클라이언트가 무작위로 데이타를 액세스할 확률이 증가함에 따라, 각 캐쉬 관리 기법간의 차이가 캐쉬 히트율에 미치는 영향이 줄어들기 때문이다. 만일 Noise 확률이 100%가 될 경우에는 각 기법의 캐쉬 히트율이 동일하여 각 기법은 거의 유사한 성능을 보이게 된다.



(그림 5) 평균 응답 시간 (ticks)



(그림 6) 클라이언트 캐쉬 히트율

VI. 결론

본 논문에서는 특정 데이타들이 방송 채널을 통하여 지속적으로 전파되는 방송 환경의 특수성을 고려하여 데이타 방송 기법에 적합한 새로운 캐쉬 관리 기법인 2Q-CF(two queue with closest-first) 기법을 제안했다. 그리고 모의 실험을 통하여 2Q-CF 기법의 성능과 LRU-CF 및 CF 기법의 성능을 비교하였다. 2Q-CF 기법은 자주 액세스되는 페이지와 그렇지 않은 페이지들을 구분하여 관리하고, 자주 액세스되지 않는 페이지중에서 희생자를 선정하기 때문에 캐쉬 히트율이 높은 장점이 있다. 그리고 희생자 선정시 방송 채널에서 가장 먼저 방송될 페이지를 희생자로 선정하는 정책을 취하고 있기 때문에 캐쉬 미스로 인한 대기시간을 줄일 수 있는 장점이 있다. 이와 같은 이유로 해서 2Q-CF 기법은 다른 기법에 비하여 훨씬 우수한 성능을 보인다.

클라이언트 캐쉬 관리 기법에서 클라이언트가 자신이 액세스할 것으로 예상되는 데이타들을 미리 자신의 버퍼에 캐싱해 두는 데이타 프리페치 기술은 클라이언트의 응답시간을 줄이기 위한 매우 효과적인 기술중의 하나이다. 본 논문의 미래 연구 과제로서 2Q-CF 기법에 프리페치 기술을 통합시키는 연구를 진행할 예정이다. 그리고 다중 방송 디스크 환경에 적합한 캐쉬 관리 기법에 관한 연구를 진행할 예정이다.

참고문헌

[1] 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, California, May 1995.

[2] 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.

[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," Proceeding 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 International Conference Object Oriented Programming Language Systems (OOPSLA 97), Atlanta, GA, October 1997.

[7] 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, 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 Distributed File System, *ACM Transactions on Computer Systems*, 6(1), pp. 51-89, Feb. 1988.

[9] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," proceeding of the 20th International Conference on Very Large Data Bases, pp. 439-450, Santiago, Chile, September 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, January 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," Proceeding of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 297-306, May 1993.

[14] H. Schwetman, CSIM Users 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, inc., 1992.

Performance Evaluation of Client Cache Management Scheme For Data Broadcasting Environments

Hyeok-Min, 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 2Q-CF, to reduce this waiting time and evaluate its performance on the basis of a simulation model. The performance results indicate that 2Q-CF scheme shows superior performances over LRU-CF and CF in the average response time.

* Department of Software, Semyung University