LMF로 수정된 웹 프락시 캐쉬용 LRU 페이지 교체 정책

(A Modified LRU Page Replacement Policy with LMF for Web Proxy Cache)

이 용 임 † 김 주 균 ††
(Yong Im Lee) (Ju Gyun Kim)

요 약 웹 사용자들의 QoS를 위해 채택되는 프락시 캐쉬의 관리 정책은 크게 페이지 교체와 일관성 유지 정책으로 볼 수 있다. 현재까지 이 두 가지의 연구는 서로간의 관련 유무를 떠나 독립적으로 진행되어 왔다. 본 논문에서는 LRU 페이지 교체 방식에 일관성 유지 정책에서 사용하는 최종 수정 인수의 특성을 도입함으로써 두 정책간의 상호 보완으로 프락시 캐쉬의 효율적 운영이 가능함을 보였다. 프락시 캐쉬의 관리를 위해 다양한 정책이 논의되고 있는 시점에서 이러한 시도는 캐쉬의 히트율 증가와 서버로의 전송 부하 감소 등과 같은 성능 향상을 위한 의미 있는 방법이 될 수 있을 것이다.

키워드 : 웹 프락시, 페이지 교체, 일관성, 최종 수정 인수

Abstract Management policies of Web Proxy Cache, for the QoS of Web users, are mainly focused on the page replacement and the data consistency policy. But the two subjects have been studied independently to each other regardless of its possibility of cooperation. In this paper, we introduce the performance improvement obtained by adapting the characteristic of LMF used in data consistency policy to LRU, thus taking the better performance synergy as a result of complementary cooperation. Various policies for the management of Web Proxy Cache are in progress, this study can be a way of performance guide to increase cache hit ratio and reduce the transmission overhead of Web Server.

Key words : Web Proxy, LRU, Consistency, LMF

1. 서 론

최근 네트워크 사용자가 증가함에 따라 웹 서버의 객체를 복사한 후 클라이언트에게 대신 서비스하는 웹 프락시(Web proxy) 캐쉬의 중요성이 커지고 있다. 웹 프락시 캐쉬는 클라이언트에게는 빠른 응답을, 서버에게는 부하 분배(load sharing)의 효과를, 그리고 네트워크 전체적인 측면으로는 트래픽(Traffic)의 감소와 네트워크 대역폭 증가를 가져올 수 있으며 이러한 긍정적인 측면 때문에 최근 웹 프락시 캐쉬에 대한 연구가 활발히 진행되어지고 있다[1-7].

웹 프락시 캐쉬에 관련된 연구의 두 가지 큰 맥락이 페이지 교체 정책(page replacement policy)과 데이타 일관성 유지 정책(*data consistency policy*)임은 잘 알려져 있다. 특히 데이타 일관성 유지 정책은 웹 프락시 캐쉬가 웹 서버를 대행하여 객체를 클라이언트에게 서비스한다는 측면에서 봤을 때 웹 프락시 캐쉬의 신뢰도에 관련된 상당히 중요한 문제라고 할 수 있다.

이러한 두 주제의 연구는 각자 개별적으로는 많이 연구되어지고 있으나 이 두 가지 정책간에 존재 가능한 상호 보완적 측면을 발견하고 이를 활용하기 위한 시도는 거의 발견되지 않고 있다[8]. 본 논문에서는 페이지 교체가 일어날 때 무효(invalid)하거나 곧 무효화될 가능성이 있는 객체들을 일차적인 교체 대상으로 선정토록 하여 추후 발생할 일관성 검사의 부하를 줄이고 웹

· 본 연구는 숙명여자대학교 2003년도 교내연구비 지원에 의해 수행되었음
† 비 회 원 : ㈜쌍용정보통신
yommi@sicc.co.kr
†† 정 회 원 : 숙명여자대학교 정보과학부 교수
jgkim@sookmyung.ac.kr
논문접수 : 2002년 8월 20일
심사완료 : 2003년 5월 9일

프락시 캐쉬의 신뢰성을 높이는 방안을 연구하였다.

일관성의 검사는 현재 가장 많이 상용화되어 사용되어지고 있는 정책인 Adaptive TTL(Time To Live)을 사용하되 Adaptive TTL에서 핵심으로 사용되는 최종 수정 인수(Last modified Factor, LMF)의 특성을 LRU(Least Recently Used) 페이지 교체 알고리즘에 삽입시켜 교체 대상의 선정 시에 일차적으로 영향을 미칠 수 있도록 하였다. LRU 역시 현재 검증된 페이지 교체 정책 중 가장 바이트 히트율(byte hit ratio) - 히트된 객체의 크기로 측정되며 유효 히트의 경우 객체의 크기만큼 전송의 부담이 없고 따라서 바이트 히트율이 높다 - 이 높다고 알려져 있으므로[1,7,9] 가장 보편적으로 사용되는 이 두 가지 정책을 활용, 수정하여 운영해봄으로써 기존의 Adaptive TTL과 LRU를 순수하게 사용했을 경우와 비교해 성능향상의 가능성과 함께 그 정도를 확인하는 것이 논문의 요체이다.

2장에서는 기존의 페이지 교체 정책과 일관성 유지 정책과 관련하여 Adaptive TTL과 LMF에 대해 알아보고 3장에서는 LMF가 추가되어 수정된 LRU를 제안하고 분석한다. 4장에서는 제안된 정책과 기존의 정책간 비교 시뮬레이션으로 얻어진 결과 값의 분석을 통해 이러한 시도를 함으로써 얻을 수 있는 성능 향상의 가능성을 제시하고 5장에서는 결론과 향후 연구 방향에 대해 논한다.

2. Adaptive TTL과 LMF

일관성 유지 정책은 크게 약한 일관성 유지 정책(Weak consistency policy)과 강한 일관성 유지 정책(Strong consistency policy)으로 나눌 수 있는데, 약한 일관성 유지 정책은 무효화된 객체가 클라이언트에게 서비스될 확률이 강한 일관성 유지 정책보다 높으나 웹 상에서 구현하기 편하고 웹 프락시 캐쉬의 부하도 작고 일관성 검사를 위해 웹 서버로 보내는 요청이 적어서 트래픽이 적은 이유로 일반적으로 많이 사용되고 있는 방법이며 이 정책에 속하는 것들로서 TTL과 Adaptive TTL, Client Polling 등이 있다[5,10].

TTL은 객체마다 유효를 인정하는 시간 값(*ttl* 값)을 주고 이 값이 만기가 될 때까지 해당 객체는 유효하다고 간주하는 방법이다. 객체가 요청될 때마다 이 값을 검사하며 값이 만기가 되면 객체를 가지고 있는 웹 서버에 일관성 검사를 요청하고 유효하면 *ttl*값을 재조정하고 객체를 요청한 클라이언트에게 서비스하며, 무효화되었다고 판단되어지면 캐쉬에서 객체를 삭제하고 웹서버에서 새로 캐싱해 온다[11]. TTL값은 각 객체의 URL에 포함되어 있는 Expire 필드에서 읽어오는데[7] 이 기법의 단점은 적절한 *ttl*값을 결정하는 것이 어렵다는 것이다. 만약 값이 너무 작다면 웹 서버에 일관성 검사를 너무 자주 요청하게 되어 웹 서버의 부하를 증가시키며 값이 너무 크다면 클라이언트가 무효화된 객체를 서비스 받을 확률이 높아진다.

이러한 문제를 해결하기 위해 *ttl*값을 비율적으로 구하여 적용하는 Adaptive TTL이 고안되었다[10]. Adaptive TTL은 고정된 ttl값으로 객체의 유,무효성을 검사하지 않고 LMF라는 비율적인 값으로 객체가 유효한지 여부를 검사한다. LMF는 객체가 웹 서버에서 최종 수정되었을 때로부터 현재까지의 경과 시간 즉, 객체의 나이(age)로부터 객체가 캐싱 되어진 이후의 시간까지의 비율로 정의할 수 있다. 예를 들어, 서버에서 최종 수정된 지 열흘이 지났고 캐싱 후 이틀이 지난 후 참조 요청이 있을 경우 이 객체의 현재 LMF 값은 0.2가 된다. 객체가 요청될 때마다 웹 프락시 캐쉬는 해당 객체의 그 시점에서의 LMF 값을 계산한 후 웹 프락시 캐쉬에서 정한 기준 값(Threshold Value)과 비교하여 기준 값 아래이면 아직 그 객체는 유효한 것으로 판정한 후 클라이언트에게 서비스하고, 기준 값을 초과하면 웹 서버에 객체의 일관성 검사를 요청한다. 위의 예에서 기준 값을 0.1로 정했을 경우 LMF 값이 0.2인 객체는 서버로 일관성 검사를 요청해야하며 만약 그 동안 서버에서 수정이 안된 경우라면 유효하다는 판정을 캐쉬에 알려줌과 동시에 이 객체의 나이는 12로 조정되며, 그렇지 않을 경우 객체의 재 적재와 함께 나이가 변경된다. 기준 값은 웹 프락시 캐쉬마다 다양할 수 있으며 객체마다 다르게 정할 수도 있는데 직관적으로 특정 객체가 과거 얼마동안 변하지 않았으니 향 후 기준 값을 곱한 기간동안은 유효할 것이라는 예측에 근거하고 있다[7, 12]. Adaptive TTL은 유효성 오류의 일정부분을 담보로 빠른 응답시간과 대역폭(bandwidth)의 향상 등의 장점을 가지고 있어 현재 상용 웹 프락시 캐쉬에서 주로 사용되고 있는 일관성 유지 정책이다[7]. Client Polling은 웹 프락시 캐쉬가 주기적으로 웹 서버에 캐싱된 객체들의 일관성 검사를 요청하는 것이며[11], 그외 대표적인 강한 일관성 유지 정책으로 Invalidation과 Polling every time이 있는데, Invalidation은 객체가 수정되면 웹 서버가 그 객체를 캐싱하고 있는 웹 프락시 서버에 수정되었음을 알려주는 방식이며 polling every time은 요청이 들어올 때마다 요청을 받은 객체의 일관성 검사를 웹 서버로 요청하도록 한다. 따라서 사용자가 무효화된 객체를 받을 가능성이 거의 없지만 웹 프락시 캐쉬의 부하가 지나치

게 크고 웹 서버로의 트래픽이 많아서 실제로 상용 제품에는 거의 사용되어지지 않고 있다[7,10].

페이지 교체 정책중 대표적인 LRU는 구현하기 간단하고 매우 좋은 성능을 가지고 있어 현재 거의 모든 웹 프락시 캐쉬에서 사용하고 있는 알고리즘이다[1,7,9]. 이 방식은 잘 알려진 것처럼 참조된 지 가장 오래된 객체를 페이지 교체 대상으로 선정하여 삭제하며 이러한 페이지 교체 작업은 캐쉬되어야 할 새로운 객체의 크기만큼 캐쉬내 여유공간이 생길 때까지 반복하여 실행한다. LRU에 객체의 크기를 고려한 LRU-Threshold나 LFU (Least Frequently Used), 이외에도 SIZE, HYBRID, GDS(Greedy Dual Size)등과 같이 객체의 크기를 기준으로 하는 여러 정책들이 알려져 있는데 참고문헌에 정리해 두었다[1,7].

3. LMF를 적용한 페이지 교체 알고리즘

기존의 상용 웹 프락시 캐쉬에서는 페이지 교체 정책과 일관성 유지 정책이 개별적으로 각자 필요할 때마다 수행되어진다. 그러나 객체의 유·무효성을 검사하기 위해 사용되어지는 일관성 유지 정책의 인수인 LMF를 가져다가 기존에 사용하고 있는 페이지 교체 대상 선정 정책에 포함시켜 함께 적용하면 페이지 교체 작업 이후 일어나는 일관성 검사의 부하를 줄이고 웹 프락시 캐쉬의 신뢰성과 함께 유효한 객체들을 보다 많이 캐쉬할 수 있게 하는 부가적인 효과를 예상할 수 있으므로 이 장에서는 그러한 방법론에 대해 제안하였다. LMF의 사용 이유는 이미 위에서 언급한대로 약한 일관성 유지 정책이 거의 모든 캐쉬 제품에서 채택되어지고 있고 그 중 LMF를 사용한 Adaptive TTL기법이 구현이 편리하고 가장 성능이 높은 것으로 평가되고 있기 때문이다 [7,10].

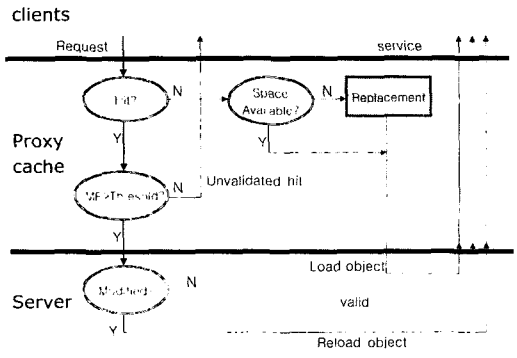

그림 1 요청에서 응답까지의 흐름도

본 논문에서 LMF를 결합하여 사용하고자 하는 페이지 교체 정책은 LRU이며 LRU의 페이지 교체 대상 선정 척도인 타임스탬프 값 즉, 최종 참조 후 경과시간(Time-since-reference)과 그 객체의 LMF 값을 페이지 교체 시 함께 고려하여 페이지 교체 대상을 정하게 될 것이다. 그림 1은 클라이언트와 캐쉬 그리고 서버간의 요청(request)부터 최종 응답까지를 절차와 함께 보여주며 화살표에 표시된 설명들은 본문에서 다룰 것이다.

3.1 페이지 교체 대상 선정

순수한 LRU는 참조 된지 가장 오래된 객체, 즉 객체의 타임스탬프 값이 가장 낮은 객체를 페이지 교체 대상으로 선정하여 삭제한다. 본 논문에서는 LRU를 기본으로 하고 있으나 페이지 교체 대상 선정의 기준으로 객체의 타임스탬프 값뿐 아니라 그 객체의 LMF도 고려하게 되는데 그 이유는 교체시기에 무효화되었거나 곧 무효화될 가능성이 있는 객체를 우선적인 페이지 교체 대상으로 선정함이 목적이기 때문이다.

페이지 교체 시 제일 먼저 LMF의 값을 검사하여 그 시점에서 무효화된 객체가 우선적으로 페이지 교체 대상으로 선정되어 삭제되어지며, 그것으로 새로운 객체를 캐쉬할 만한 충분한 공간이 확보되어지지 않았을 경우 가장 오래 전에 참조된 객체와 두 번째로 오래 전에 참조된 객체와의 타임스탬프 값의 비와 LMF의 비를 조사한 후 타임스탬프 값의 비보다 LMF의 값의 비가 더 높을 때 확률적으로 더 무효화될 가능성이 높은 객체라 판단하여 LMF 값이 더 큰 객체를 삭제한다. 만약 이러한 경우가 발생하지 않는다면 타임스탬프 값이 작은 객체들끼리 즉, 오래전에 참조된 객체들끼리의 타임스탬프 값의 비와 LMF의 값의 비를 타임스탬프 값의 오름차순으로 차례대로 검사하여 타임스탬프 값의 비보다 LMF의 값의 비가 더 큰 경우가 발생할 경우까지 진행시킨다. 결과적으로 두 객체의 LMF의 값의 비가 타임스탬프 값의 비보다 더 클 경우 타임스탬프 값이 좀 더 낮더라도 LMF의 값이 높기 때문에 먼저 삭제될 경우가 발생할 수 있다. 다시 말해 순수한 LRU를 적용했을 때 교체 대상으로 선정될 정도보다 LMF를 적용했을 때 선정될 정도가 더 큰 객체가 먼저 교체되도록 조정하는 것이다.

본 논문에서 제안하는 방법에서 페이지 교체 대상이 되는 객체의 조건은 다음 차례와 같다. 객체가 다음 세 가지 조건을 한꺼번에 만족해야 페이지 교체 대상으로 선정되는 것이 아니라 조건 1에 해당하는 객체가 없을 경우 조건 2를 적용하고 이때도 해당하는 객체가 없다면 순수한 LRU인 조건 3이 적용되어지게 된다.

조건 1: LMF의 값이 기준 값을 초과해 무효화되었다고 판정되는 객체

조건 2: 타임스탬프 값이 작은 순서대로 두 객체의 타임스탬프 값의 비(ratio)보다 LMF의 비가 더 크다면 그 중 LMF의 값이 더 큰 객체

조건 3: 가장 오래 전에 참조되어 타임스탬프 값이 가장 작은 객체(순수한 LRU)

조건 2와 3은 기존의 LRU의 페이지 교체 대상 선정 기준인 타임스탬프 값을 고려하고 있다. 조건 3의 경우는 기존의 LRU가 페이지 교체 대상을 선정할 때와 똑같은 조건이다. 조건 2는 두 가지의 경우에 따라 다음과 같이 적용되어진다.

조건 2-1 : 같은 타임스탬프 값을 가졌다면 LMF값이 큰 객체를 삭제

조건 2-2 : 다른 타임스탬프 값을 가졌다면 LMF값의 비를 비교하여 LMF의 비가 더 클 경우 둘 중 LMF 값이 더 큰 객체를 삭제

조건 2-1은 과거 똑같은 시간에 참조되었다면 둘 중 빠른 시간 안에 무효화될 객체를 먼저 삭제한다는 의미이며, 조건 2-2는 객체가 각기 다른 시간에 참조되었고 한 객체가 다른 객체보다 좀 더 오래 전에 참조되었더라도 참조된 시간의 차가 무효화될 때까지의 시간차 보다 작다면 좀 더 가까운 시간에 참조되었더라도 확률적으로 보다 빨리 무효화될 객체를 페이지 교체 대상으로 선정한다는 것이다.

3.2 페이지 교체 단계

본 논문에서 제안하는 페이지 교체 단계는 위에서 언급한 조건들을 정리한 표 1과 같이 총 3단계이다. 페이지 교체 시 이 3단계가 모두 실행되어지는 것은 아니며, 새로운 객체가 캐쉉되어질 수 있도록 충분한 여유공간이 확보되면 중단되어진다. 즉 1단계를 실행하고 난 이후 새로운 객체가 캐쉉될 공간이 확보되어지면 바로 새로운 객체를 캐쉉하고 2단계와 3단계는 실행되어지지 않고 페이지 교체 작업은 완료된다.

1단계는 페이지 교체 시에 캐쉉 되어있는 모든 객체의 현재 LMF 값을 계산한 후 무효화되었다고 판정되어지는 객체, 즉 프락시에서 정한 기준 값을 초과하는 객체들을 모두 삭제한다. 1단계를 실행하는 도중 새로운 객체가 캐쉉될 만한 충분한 공간이 확보되어도 1단계는 중지되어지지 않고 계속 진행되어진다. 그 이유는 1단계는 페이지 교체뿐 아니라 일관성 검사의 성격도 띄고 있기 때문이다. 페이지 교체의 시기에 일관성 검사가 병행되어 실행된다는 점에서 이후 발생할 일관성 검사의 부하를 상당 부분 줄여줄 수 있음을 알 수 있다.

2단계는 단 한번만 실행되어진다. 객체를 타임스탬프 값의 오름차순으로 정렬한 후 순서대로 객체의 타임스탬프 값의 비와 LMF 값의 비를 비교한다. 이 과정은 타임스탬프 값의 비보다 LMF 값의 비가 더 크다는 조건을 만족하는 객체가 있을 때까지 진행된다. 조건을 만족하는 객체를 만나면 해당 객체를 삭제하고 만약 2단계를 실행하는 도중에 조건을 만족하는 객체가 하나도 없거나, 아직 새로운 객체를 캐쉉할 여유공간이 확보되지 않은 채 2단계가 끝나게 되면 3단계로 넘어가게 된다.

3단계에서는 기존의 LRU와 같이 타임스탬프 값이 가장 작은 객체부터 순서대로 삭제되어지는데 새로운 객체가 캐쉉 될만한 여유공간이 확보될 때까지만 삭제가 진행되어진다. 2단계에서 이미 타임스탬프 값 순서대로 객체를 정렬해놓았으므로 정렬에 따른 부하는 따르지 않는다. 이상의 페이지 교체 알고리즘은 부록에 수록되어 있다.

표 1 페이지 교체 단계

1단계 : LMF를 재계산하여 무효화되었다고 판정되는 객체들을 전부 삭제
2단계 : 타임스탬프 값이 작은 객체의 순으로 타임스탬프 값의 비와 LMF 값의 비를 비교한 후 LMF의 값의 비가 더 큰 객체가 발견되면 삭제
3단계 : 타임스탬프 값이 가장 작은 객체 삭제

4. 실험 및 성능향상 평가

본 장에서는 앞에서 제안한 LMF를 적용한 LRU를 사용했을 때 어느 정도의 성능 향상이 발생하는지 알아보기 위해 다양한 조건으로 실험을 하고 그 결과를 분석하였다. 페이지 교체 정책은 LRU를 사용하고 일관성 유지 정책으로는 Adaptive TTL을 사용하는 기존의 웹 프락시 캐쉬와, Adaptive TTL과 본 논문에서 제안한 LMF를 적용한 LRU를 사용한 웹 프락시 캐쉬를 비교하여 성능의 향상 정도를 보였다. 성능 비교의 척도로는 웹 프락시 캐쉬에서 웹 서버로 나가는 전체 요청 횟수, 그리고 유효 히트(unvalidated hit)율이다. 유효 히트란 캐쉉된 객체가 유효하여 서버로의 일관성 검사 요청 없이 즉시 클라이언트(Client)로 응답할 수 있는 히트를 의미하며 그래프에서 표시된 히트율 또한 클라이언트로부터 캐쉬로의 총 요청 중 유효 히트가 발생한 비율을 의미한다. 참고로 실험의 목적 상 사용된 객체는 동일 크기로 하였다.

4.1 시간에 따른 성능 평가

이 절에서는 시간의 흐름에 따라 제안한 정책이 얼마만큼의 성능향상을 보이는지 실험하였다.

시간은 6시간 단위로 24시간까지 실험하였으며 클라이언트로부터의 총 요청 수는 DEC에서 1996년 8월 29일부터 9월 22일까지 모은 로그에서 산출해낸 하루의 평균 요청 수를 사용하였다[13]. 실험에 사용된 웹 프락시 캐쉬의 크기는 캐싱의 대상이 되는 전체 데이타의 크기에 대해 상대적인 비율로 2%에서 50%로 각각 나누어 실험하되[13,14,16] 시간에 따른 성능 평가 실험에서는 기존의 연구들에서 가장 현실성 있는 캐쉬 히트(Cache hit) 율이 나오는 크기인 20%로 고정하여 실험하였다.

캐싱의 대상이 되는 객체들의 나이는 30초에서 2분까지 무작위 분포한 값이 전체의 15%, 13분에서 17분까지 무작위 분포한 값이 전체의 30%, 50분에서 70분까지 무작위 분포한 값이 전체의 20%, 23시간에서 25시간까지 무작위 분포한 값이 전체의 15%, 그리고 나머지 20%는 최대 1달까지로 무작위로 분포하여 실제 환경을 최대한 반영할 수 있도록 다양하게 선정하였다[15]. 캐쉬의 크기나 객체의 나이에 관한 값들은 기존 유사연구에서의 자료와 가능한 한 비슷한 환경으로 맞추기 위해 선정되었다. 또한 클라이언트로부터의 객체 요청에 지역성(locality)를 주기 위해 캐싱의 대상이 되는 전체 데이타를 1:2:4의 비율로 개수를 나누었고 1/7의 비율을 가진 군에 전체 요청의 50%가, 2/7의 비율을 가진 군에 전체 요청의 20%가, 그리고 나머지에 전체 요청의 30%가 분포되도록 하였다.

실험이 시작될 초기 상태의 캐쉬는 100% 캐싱된 객체로 차 있어 여유 공간이 전혀 없는 상태로 가정하였는데 이유는 웹 프락시 캐쉬가 실제로 운영될 때 거의 모든 공간을 다 활용하여 객체를 캐싱하고 있다는 점을 살리기 위함이다.

4.1.1 웹 서버로의 총 요청

먼저 시간의 흐름에 따라 웹 프락시 캐쉬에서 웹 서버로 나가는 총 요청을 측정하였다. 성능 평가의 결과 값은 클라이언트로부터의 총 요청 수에 대한 웹 서버로의 요청 수의 비율로 나타내었다. 즉, 웹 서버로의 총 요청 - (웹 서버로의 총 요청수/클라이언트로부터의 총 요청수)*100을 의미한다.

제안한 정책은 페이지 교체가 일어날 때마다 무효화된 객체들과 확률적으로 곧 무효화될 객체들을 미리 삭제함으로서 한번의 페이지 교체가 일어난 이후로는 웹 프락시 캐쉬가 웹 서버로 일관성 검사를 요청하는 횟수가 줄게 된다. 따라서 기존의 LRU와 Adaptive TTL을

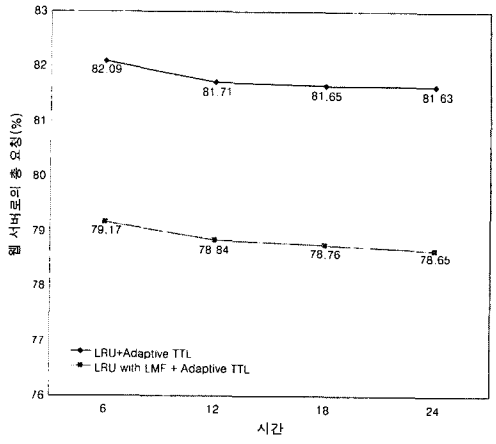

그림 2 웹 서버로의 총 요청(%)

사용하는 경우에 비해 웹 프락시 캐쉬의 웹 서버로의 요청이 그만큼 줄어들게 되므로 그림 2와 같이 약 4% 정도의 성능향상을 보이고 있다. 구체적인 예로, 현재 비교의 대상인 기존 정책 하에서 클라이언트로부터 요청이 있을 경우 이미 LMF값이 기준값을 초과하는 객체가 캐쉬에 있다면 실제 요청 시 캐쉬에서는 히트하였으나 유효하지 않으므로 서버로 일관성을 위한 검사 요청이 있게 되며 서버에서의 수정여부에 따라 아직 유효하다는 신호 또는 객체의 재 적재를 가져와 결과적으로 트래픽을 증가, 대역폭의 감소를 유발하게 될 것이다. 같은 경우 제안된 정책 하에서는 직전의 교체정책에서 이런 객체가 이미 삭제될 확률이 매우 높고 - 100%라고 하지 못하는 이유는 직전 교체 정책의 가동 시점과 현 시점의 시간 차이 때문임. - 따라서 유효하지 못할 객체들이 차지하는 캐쉬공간을 새롭게 참조될 객체들이 대신할 수 있는 가능성을 높일 수 있으므로 바로 다음의 요청에 대해서는 LMF값 때문에 일관성 검사를 해야하는 확률은 거의 0에 가까울 것이라는 점을 예측할 수 있다. 다시 말해 교체 작업이 가동된 직 후 캐쉬에 남아있는 객체들은 그 시점에서 전부 유효한 것이므로 일관성의 검사를 위해 트래픽을 발생시키지 않을 것이라는 점이다.

4.1.2 시간대 별 유효 히트율

다음으로 시간의 흐름에 따라 웹 프락시 캐쉬의 유효 히트율을 측정하였다. 그림 3에서 알 수 있다시피 제안한 정책이 기존의 LRU보다 유효 히트율이 높은 이유는 위에서의 예와 마찬가지로 페이지 교체가 일어날 때마다 그 시점에서 무효화된 객체들을 미리 삭제해버리기 때문에 한 번의 페이지 교체가 일어난 직후 웹 프락

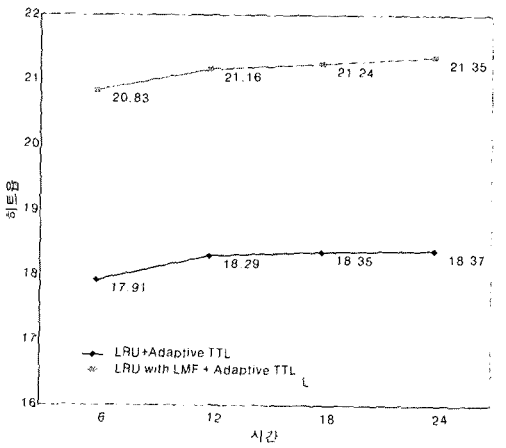

그림 3 유효 히트율

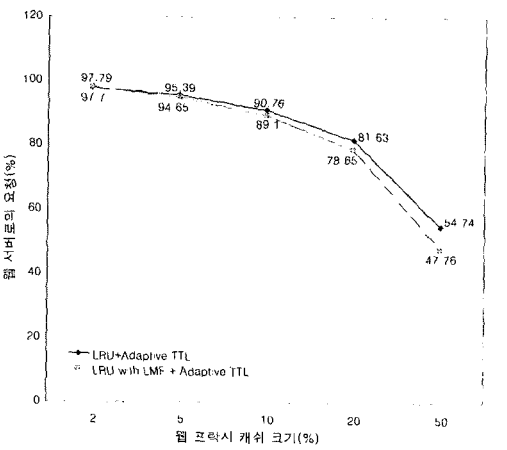

그림 4 웹 프락시 캐쉬 크기에 따른 웹 서버로의 요청(%)

시 캐쉬내에 캐싱되어져 있는 객체들은 모두 유효한 객체들이기 때문이다. 따라서 페이지 교체 직후에 일어나는 히트들은 유효 히트일 확률이 매우 높게 나타난다.

4.2 웹 프락시 캐쉬의 크기에 따른 성능평가

이 절에서는 웹 프락시 캐쉬의 크기에 따라 제안한 정책의 성능평가를 하였다. 웹 프락시 캐쉬의 크기는 전체 실험 데이타 크기에 대해 상대적인 비율로 지정하였다. 실험에 사용된 웹 프락시 캐쉬의 크기는 전체 실험 데이타 크기의 2%, 5%, 10%, 20%, 50%로 두었다 [13,14,16].

4.2.1 웹 서버로의 총 요청

여기서는 웹 프락시 캐쉬의 크기에 따라 캐쉬에서 웹 서버로 나가는 총 요청의 비를 실험하였다. 그림 4에서 알 수 있듯이 캐쉬의 크기가 클수록 본 논문에서 제안한 정책의 성능이 더욱 향상되고 있다. 이것은 캐쉬의 크기가 클수록 객체를 많이 캐싱할 수 있으며 비례하여 캐쉬 히트율이 높아지기 때문이다. 제안한 정책은 일단 많은 객체가 캐싱 되어진 이후 페이지 교체가 빈번하게 일어날수록 성능 향상 폭이 커지기 때문에 많은 히트가 일어날수록 유리하다. 참고로 캐쉬의 크기가 50%라는 것은 캐싱의 대상이 되는 전체 객체의 반을 수용할 수 있는 크기를 의미한다.

4.2.2 크기별 유효 히트율

다음으로 웹 프락시 캐쉬 크기에 따른 캐쉬의 유효 히트율을 실험하였다. 그림 5에서 웹 프락시 캐쉬 크기가 클수록 제안한 정책과 LRU와 Adaptive TTL을 사용하는 비교 대상 정책과의 유효 히트율의 차가 커짐을 알 수 있다. 이는 웹 프락시 캐쉬 크기가 커짐으로 인해 보다 많은 객체를 캐싱할 수 있는 상황에서 클라이

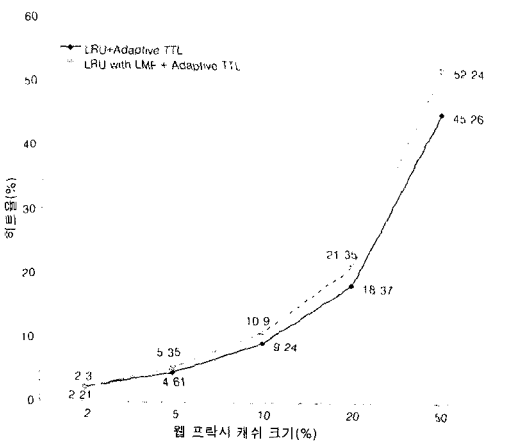

그림 5 웹 프락시 캐쉬 크기에 따른 유효 히트율(%)

언트로부터의 전체 요청에서 캐싱되어진 객체를 찾는 요청이 차지하는 비율 즉, 캐쉬 히트율(Cache hit ratio)이 커지게 되고 이 경우 제안한 정책은 교체 정책의 실행 결과 상대적으로 많은 유효한 객체들을 캐싱하고 있으므로 유효 히트율이 더 커지는 것으로 판단된다.

2단계 페이지 교체 작업 중에 타임스탬프값의 오름차순으로 객체를 정렬하고 순서대로 객체의 타임스탬프값의 비와 LMF의 값의 비를 비교하는 작업을 하기 때문에 이 비교 작업에 의한 부하가 있을 수 있다. 그러나 이것은 1단계에서 교체대상을 발견하지 못했을 경우에 한정되며 순수한 LRU에서도 타임스탬프값의 비교를 위해 정렬이나 검색의 부하가 요구된다는 점에서 허용 가능한 오버헤드로 판단된다. 더구나 이것은 웹 프락시 캐쉬 자체의 계산 작업(Computation-bound job)이므로

웹 환경에서 QoS는 트래픽에 의해 결정됨을 감안하면 제안된 정책을 통해 얻을 수 있는 네트워크 트래픽의 절약과 그로 인한 클라이언트로의 응답시간 단축 등을 비교해 볼 때 충분히 보상 가능한 부하가 될 것이다.

성능 평가의 결과 제안된 정책의 효과가 기존 정책과의 비교에서 우위를 보이고 있으며 이 차이는 결국 제안된 정책이 교체 정책을 통하여 최대한 캐쉬에 유효한 객체를 적재토록 유도하고 있는 점에 의해 발생된다고 판단된다. 또한 그 과정에서 LMF가 기준값을 초과하는 모든 객체를 삭제함으로서 향 후 적재될 객체를 위한 공간을 확보하고 이것은 그만큼의 교체 정책 가동 횟수도 줄일 수 있는 가능성을 높인다는 점도 성능 우위의 요인으로 작용할 것이다. 물론 제안된 정책에서 응답 시간과 트래픽의 감소가 교체 정책의 실행 시 좀 더 많은 시간을 요구함으로서 가능하다는 것이며 이에 대한 타협(Trade-off)은 이미 언급하였다.

살펴보면 LRU와 Adaptive TTL에서 사용되는 LMF가 모두 객체의 선택 기준에 결정적 요인으로 작용하는 것이 객체의 수정 간격과 참조 빈도에 대한 과거의 자료를 바탕으로 가까운 미래를 예측하는 시간의 문제임을 알 수 있고 이것으로부터 두 개의 정책이 상호 보완적 요소가 있을 가능성을 예측하여 LMF를 LRU에 포함시켜 그 만큼의 성능 개선을 확인하는 것이 이 논문의 핵심이라 할 수 있다.

5. 결론 및 향후 연구 과제

본 논문에서는 웹 프락시 캐쉬의 페이지 교체시 일관성 검사의 척도인 LMF를 함께 고려함으로서 웹 프락시 캐쉬의 히트율을 높이고 일관성 검사의 부하를 줄이는 방안을 연구하였다. 일관성 검사의 부하 면에서는 상당한 성능의 향상을 보였으나 무효화된 객체들을 페이지 교체 1단계에서 많이 삭제함에 따라 빈 공간이 늘어남으로서 전체적으로 웹 서버로 나가는 요청의 수는 평균 3%의 향상을 보였다.

히트율은 성능 비교의 결과 그림 2와 같이 시간에 따라 기존의 정책을 적용할 경우 17%에서 18% 정도로 나타났으나 본 논문에서 제안한 정책을 사용하면 대략 20%에서 21%를 보임으로서 히트 면에서도 향상되었음을 알 수 있다. 제안한 정책을 사용하면 웹 프락시 캐쉬는 매 시점마다 유효한 객체들을 상대적으로 많이 캐싱하고 있고 따라서 클라이언트는 웹 서버로의 일관성 검사의 지연시간 없이 바로 요청한 객체를 서비스 받을 가능성이 높다.

이번 연구에서는 객체의 개별적인 크기를 고려하지 않고 동일 크기의 객체로 가정하고 실험하였다. 페이지 교체는 새로운 객체를 캐싱할 만한 충분한 여유공간이 획득될 때까지 계속되어지는 것이므로 객체의 크기를 다양하게 주어 보다 더 정확한 성능 평가를 할 수 있을 것이다. 그리고 다양한 페이지 교체 정책과 일관성 검사 정책의 요소들을 결합해 봄으로서 좀더 구체적인 성능의 평가 척도들을 발견할 수 있을 것이다.

웹의 객체들에 대한 요청의 지역성 부여는 본 논문에서 어느 정도 시도하였으나 이러한 요청들은 시간과 장소에 따라 상당히 다양하게 변할 수 있다. 따라서 이러한 지역성을 하나의 경향으로 적용하기는 어려운 것이 사실이나 특정 시간 대 특정 장소에 따른 요청의 지역성 경향을 적용하여 성능 평가를 시도해 볼 수도 있을 것이다.

페이지 교체 정책과 일관성 유지 정책에서 사용되어지고 있는 정책들을 결합하는 시도는 본 논문에서 그 가능성을 입증하였으므로 여러 정책들의 결합과 상호보완을 통해 보다 나은 성능 향상이 나올 수도 있음을 기대하며 이는 향후 연구과제로 남긴다.

참 고 문 헌

[1] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams, and E.A. Fox, "Caching Proxies: Limitations and Potentials," In Proceedings of the Fourth International WWW Conference, 1995.

[2] Hyokyung Bahn, Yong Hyeon Shin, and Kern Koh, "A Cooperative Caching Policy for Web Caches," 6th Asia Pacific Conference on Communications (APCC2000), Seoul, Korea, pp.1058 1062, 2000.

[3] Greg Barish, Katia Obraczka, "World Wide Web Caching: Trends and Techniques," IEEE Communications Magazine, May 2000.

[4] Pei Cao, "Web Caching: Scalability, Functionality and Performance Issues," Tech. Report, University of Wisconsin, April 1999.

[5] 김주용 "프락시 캐쉬 일관성 유지기법 및 성능분석", Tech. Report, 서울대학교, 2000.

[6] Ari Luotonen, CERN, Kevin Altis, INTEL, "World Wide Web Proxies," April, 1994.

[7] Duane Wessels, "Web Caching," O'REILLY, 2000.

[8] James Gwertzman, Margo Seltzer, "World Wide Web Cache Consistency," Proc. of the USENIX 1996 Annual Tech. Conf., January 1996.

[9] Luigi Rizzo, Lorenzo Vicisano, "Replacement policies for a proxy cache," IEEE/ACM Transactions on Networking, VOL. 8, NO 2, April 2000.

[10] Pei Cao, Chengjie Liu, "Maintaining Strong Cache Consistency in the World Wide Web," Proceedings of ICDCS'97, pp.12 21, May 1997.

[11] Jooyong Kim, Hyokyung Bahn, and Kern Koh.

"Exploiting Metadata of Absent Objects for Proxy Cache Consistency," IEICE Transactions on Communications, Vol.E84 B, No.5, pp.1406-1412, May 2001.

[12] Mark Nottingham, "Optimizing Object Freshness Controls in Web Caches," WCW99.

[13] Pei Cao, Sandy Irani, "Cost-Aware WWW Proxy Caching Algorithms," Proceedings of the USENIX Symposium on Internet Technology and Systems, pp. 193-206, 1997.

[14] Hyokyung Bahn, Hyunsook Lee, Sam H. Noh, Sang Lyul Min, and Kern Koh, "Replica-Aware Caching for Web Proxies," Computer Communications, Elsevier Science, Vol.25, No.3, pp.183-188, Feb. 2002.

[15] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, Jeffrey Mogul, "Rate of Change and other Metrics : a Live Study of the World Wide Web," USENIX Symposium on Internetworking Technologies and Systems, December 1997.

[16] Hyokyung Bahn, Sam H. Noh, Sang Lyul Min, and Kern Koh, "Efficient Replacement of Non-uniform Objects in Web Caches," to appear in IEEE Computer, Vol.35, No.6, June 2001.

부 록

```
LMF를 적용한 LRU의 페이지 교체 단계
try caching of new object;
if(free cache size < the size of new object) {        /* 1단계 시작*/
    for(all cached objects) {
        set TS[i] = TSOb(i);        /*캐싱되어 있는 객체의 타임
                                      스탬프 값을 리스트 TS에
                                      입력시킴*/
        AGEOb(i) = current system time - cachedtimeOb(i);
        /*캐싱되어 있는 객체의 캐쉬내 Age 계산*/
        LMFOb(i) = AGEOb(i)/LMA(GEOb(i);
        /*캐싱되어 있는 객체의 LMF 재계산*/
        /*LMA(GEOb(i) = cachedtimeOb(i) - LastMdTOb(i);
    즉 객체가 캐싱된 시간 - 객체가 서버에서 마지막으로 수정된 시간*/
        if(LMFOb(i) > Threshold value) {
            drop Ob(i); /*재계산된 LMF값이 시스템에서 정
                          한 기준 값보다 낮을 경우 해당객체
                          를 무효화된 객체로 판단하여 삭제*/
        }
    }

    if(free cache size < the size of new object) {
        /* 새로운 객체를 수용할 충분한 공간이 확보되었는지 검사*/
        /* 그렇지 못한 경우 2단계 시작*/
        for (all cached object) {
        for (all items of TS) {
            sort increasing order;
            }
        for(all items of TS) {
        for(i = 0;i <= cache size;i++) {
        if(TSOb(a) = TS(i)) {
        if(TSOb(b) = TS(i+1) {
        if(TSOb(a)/TSOb(b) <= LMFOb(a)/LMFOb(b)) {
    /*가장 오래전에 참조된 객체부터 비교하기 시작하여 타임스탬
      프 값의 비가 LMF값의 비보다 작은 경우 생길 때 LMF 값
      이 큰 객체 삭제*/
            drop Ob(b);
        }
        }
        }
        }
        }
        }
        }
    if (free cache space < size of new object) {
    /*새로운 객체를 캐싱할 만한 충분한 공간이 확보되었는지 검사*/
    /*3단계 시작*/
    for (all cached objects) {
        search the lowest timestamp value in cached objects;
        drop object that has the lowest timestamp;
        /*가장 낮은 타임스탬프 값을 가지고 있는 객체를 삭제*/
    if (free cache space >= size of new object) {
        break;
        /*새로운 객체를 캐싱할만한 충분한 공간이 확보된 때까
          지 계속 수행*/
        }
    }
}
}
```

이 용 임

2000년 2월 숙명여자대학교 전산학과 졸 2002년 8월 숙명여자대학교 대학원 컴퓨터과학 전공 석사 졸. 2002년 10월~현재 ㈜쌍용정보통신 근무

김 주 균

1985년 서울대학교 계산통계학과 졸 1985년~1986 DEC Korea 근무. 1988년 서울대학교 계산통계학과 계산학 석사 1992년 서울대학교 계산통계학과 계산학 박사. 1992년~현재 숙명여자대학교 정보과학부 컴퓨터과학전공 교수. 관심분야는 운영체제, 성능평가, DOS, MOS