ACASH: 웹 객체의 이질성과 참조특성 기반의 적응형 웹 캐싱 기법

(ACASH: An Adaptive Web Caching Method with Heterogeneity of Web Object and Reference Characteristics)

고 일 석 † 임 춘 성 †† 나 윤 지 †††
(Ilseok Ko) (ChunSeong Leem) (Yunji Na)

요 약 웹 객체의 저장과 처리를 위한 캐시의 사용이 증대하고 있으며, 캐시 저장영역의 효율적인 관리를 위한 많은 연구가 활발히 이루어지고 있다. 웹 캐싱 기법은 전통적인 기법과 차이가 있다. 특히 웹 캐싱의 처리 단위인 웹 객체의 이질성과, 시간에 따른 웹 객체 참조특성 변화는 기존 기법들의 성능을 감소시키는 중대한 원인이 되고 있다. 본 연구에서는 새로운 웹 캐싱 기법인 ACASH(the Adaptive Caching Algorithm with Size Heterogeneity)를 제안하였다. ACASH는 웹 객체와 캐시 영역을 이질성을 기반으로 분할 관리함으로서 객체의 이질성 편차를 줄였고, 시간의 흐름에 따른 객체 참조 특성의 변화를 적응적으로 반영하고 있다. 또한 객체의 이질성을 고려한 두 개의 실험 모델에 대해, 기존의 대체 기법들과 비교 실험을 통해 ACASH의 우수성을 확인하였다.

키워드 : 웹 객체, 웹 캐싱, 크기 이질성

Abstract The use of a cache for a storing and processing of Web object is becoming larger. Also, many studies for efficient management of storing scope on cache are performed actively. Web caching technique have many differences with traditional techniques. Particularly, a heterogeneity of Web object which is a processing unit of Web caching and a variation of Web object reference characteristic with time are the important causes to decrease performance of existing techniques. In this study, We proposed the ACASH which was new web caching technique. As ACASH divided and managed Web object and a cache scope with a heterogeneity, It can reduced a heterogeneity variation of an object. Also, it is reflecting a variation of object reference characteristics with time adaptively.

In the experiment, We verified that the performance of ACASH was improved than existing techniques on the two experiment model which considered a heterogeneity of an object.

Key words : web object, web caching, Size Heterogeneity

1. 서 론

네트워크에서 웹 트래픽이 차지하는 비율이 점차 높아지고 있기 때문에, HTTP 요구의 효율적인 처리는 네트워크 관리의 효율을 평가하는 중요한 요인이 되고 있다. 일반적으로 웹 캐시는 HTTP 요구를 효율적으로 처리할 수 있으며, 인터넷의 성능을 향상시킨다[1,2].

웹 캐시의 성능은 한정된 웹 캐시 저장 영역의 효과적인 관리에 달려있다. 이를 위해, 자주 사용되는 웹 객체를 캐시의 저장 영역에 유지하기 위한 대체 기법에 대한 연구들이 활발히 진행되고 있다[3-5]. 웹 캐시를 위한 대체 기법은 웹 객체의 특성을 반영하여야 하며[6,7], 웹 객체는 다음과 같은 특성을 가지고 있다.

- 사용자에 의해 참조되는 웹 객체의 크기 편차는 매우 크다. 따라서 웹 캐시는 인터넷 사용자가 요구하는 가변적인 객체를 효율적으로 지원하여야 한다.
- 웹 객체는 시간과 지역에 따른 참조 국지성을 가진다. 또한 연령 및 인터넷 사용의 숙달 정도와 같은 사용자 개인의 참조 특성에 따라 가변적인 객체 참조 특성을 가진다. 이러한 웹 객체의 참조 특성은 웹 객체의 이질성 편차를 높인다.
- 특정한 웹서비스의 성격에 따른 객체 참조 특성은 캐

† 정 회 원 : 충북과학대학 전자상거래과 교수
isko@ctech.ac.kr
†† 비 회 원 : 연세대학교 정보산업전공 교수
leem@yonsei.ac.kr
††† 비 회 원 : 충북대학교 컴퓨터공학과
yjna2967@korea.com
논문접수 : 2003년 6월 9일
심사완료 : 2004년 1월 30일

시의 성능에 중대한 영향을 미친다.

- 웹 캐싱에서는 크기가 큰 하나의 객체에 의해 크기가 작은 많은 객체가 저장 영역에서 제거되는 경우가 발생하며 이것은 웹 캐싱의 성능에 많은 영향을 미친다.

네트워크 기술의 발전과 디지털 있는 컨텐츠 제작 기술의 발달로 인해 웹 객체의 이질성은 점차 심해지고 있으며 이러한 객체의 이질성은 점차 웹의 효율성에 많은 영향을 미친다. 이와 같은 객체 이질성의 심화는 더욱 빈번한 웹 객체 대체를 발생시키지만, 전통적인 대체 기법에서는 이러한 특성을 충분히 반영하지 못하고 있다. 또한 참조의 국지성과 다른 참조 특성은 시간에 흐름에 따라 가변적이며 이는 기존 캐싱 기법의 성능을 떨어뜨리는 중요한 요인이 되고 있다[8,9].

본 연구에서는 웹 객체의 이질성을 적응적으로 반영한 웹 캐싱 기법인 ACASH(the Adaptive Caching Algorithm with Size Heterogeneity)를 제안하였다. ACASH는 웹 캐시의 효율을 높이기 위해 웹캐싱문제를 다음과 같은 아이디어로 접근하였다.

- 객체 이질성은 객체의 크기 편차와 밀접한 관계가 있다. 따라서 이질성에 따른 객체의 구분은 크기에 따른 객체 구분을 가능하게 한다.
- SIZE 기법과 LRUMIN 알고리즘은 크기가 큰 객체에 의해 크기가 작은 많은 객체가 저장 영역에서 제거되는 경우를 줄일 수 있다. 하지만 객체의 크기 편차가 큰 참조 특성에 대해서는 그 성능이 감소된다.
- 객체를 크기별로 나누어 관리하면 각 영역에서 객체 크기 편차는 하나로 관리되는 것보다 줄어들게 된다. 이 경우 크기 편차의 감소로 인해 하나의 큰 크기 객체의 대체를 위해 발생되는 작은 크기 객체의 수가 줄일 수 있다.
- 객체의 참조특성은 매우 가변적이다. 이에 따라 이질성의 편차 또한 가변적이다. 가변적인 참조특성을 반영하기 위해서는 적응적인 캐싱기법이 필요하다.

또한 본 연구에서는 ACASH의 성능을 평가하기 위해 객체 이질성을 반영한 두 가지 객체 사용 모델을 설정하고, 기존의 대체 기법들(LRU, LRUMIN, SIZE)과 객체적중률 및 객체크기적중률, 응답시간을 비교 실험하였다. 실험결과 기존의 대체 기법들에 비해 성능의 향상을 확인할 수 있었다.

2. 연구배경

2.1 관련연구

지금까지 많은 웹서버가 텍스트와 이미지 객체를 주로 서비스하였다. 하지만, 웹은 점차 멀티미디어 환경의 서비스로 발전하고 있으며 향후 웹 서비스의 50% 이상이 멀티미디어 서비스가 될 것이라 예측하고 있다[10].

웹 캐시에 클라이언트의 객체 요구가 발생하면, 캐시 관리자는 요구된 객체가 웹 캐시 서버의 저장 영역에 존재하는지 판단한다. 이때, 클라이언트가 요구한 객체가 웹 캐시의 저장 영역에 존재하면 cache hit가 발생한다. 그리고 존재하지 않으면 cache miss가 발생하며 이것은 캐시의 성능을 감소시키는 요인이 된다. cache hit의 경우 해당 객체를 캐시의 저장 영역에서 읽어서 클라이언트에게 전달한다. cache miss의 경우 요청된 URL로부터 객체를 전송받아 클라이언트에게 제공한다. 이 경우, 자주 사용할 새로운 객체를 위해 캐시의 저장 공간을 제공하여야 한다. 이 과정에서 교체기법은 이미 저장되어 있는 사용하지 않는 객체와 자주 사용할 새로운 객체를 교체하기 위해 필요하다. 결국, 웹 캐싱은 자주 사용되는 객체를 캐시에 저장함으로서 인터넷의 처리 능력을 향상시키며, 웹 서버의 부하와 전체 네트워크의 트래픽을 감소시킴으로서 클라이언트에게 객체의 빠른 응답을 제공한다. 웹 캐싱의 종류에는 클라이언트 캐싱, 서버 캐싱, 프락시 캐싱, 계층 캐싱, 협동 서버 캐싱이 있다[11,12].

캐싱의 공통적인 관심사는 한정된 저장 영역의 효율적인 운용이다. 사용 빈도가 높은 객체를 캐시의 저장영역에 저장해야 캐시의 성능을 높일 수 있다. 따라서 효율적인 대체 기법은 캐싱의 성능을 향상시키는 중요한 요인이다[11,13]. 현재까지 FIFO(First In First Out), LRU(Least Recently Used), LFU(Least Frequently Used), LUV, LRUMIN과 같은 여러 대체 기법들이 연구되어 왔다. LRU와 LFU는 전통적인 대체 기법을 인터넷 캐싱 분야에 적용한 기법이다. 또한 SIZE와 LRUMIN은 키에 기반을 둔 기법이다. 대표적인 기법은 다음과 같다.

- LRU는 새로운 객체가 저장 될 공간을 마련하기 위하여 가장 최근에 사용되지 않은 객체를 저장 영역에서 교체하는 기법이다. LRU는 최근에 오랫동안 사용되지 않은 객체가 미래에 참조되지 않을 가능성이 가장 높은 객체라는 특성을 이용하는 방식이며, 객체 참조의 국지성[9]을 이용한다. LFU는 참조의 빈도수를 기반으로 자주 사용되지 않은 객체를 우선적으로 교체하는 방식이다.
- SIZE는 새로운 객체가 저장 될 공간을 마련하기 위하여, 저장 영역에 저장된 객체 중에서 가장 큰 객체를 교체하는 기법이다. 인터넷 캐시는 하드웨어 캐시, 파일 시스템 캐시와 차이가 있으며, 인터넷 캐시는 객체의 크기가 가변적이고 대체의 단위가 웹 객체이다[14]. 따라서 크기가 큰 하나의 객체에 의해 크기가 작은 많은 객체가 저장 영역에서 제거되는 경우가 발생한다[3]. SIZE의 경우는 캐시 저장영역의 객체 중에

서 가장 큰 객체를 교체함으로서 이 문제점을 개선하였다.

- LRUMIN은 LRU의 변형이며 동작은 다음과 같다. 먼저, 이 기법은 새롭게 들어오는 객체의 크기를 S로 설정한다. 다음으로, S보다 큰 여유 공간이 없으면 S/2 이상의 큰 객체들 중에서 LRU로 제거한다. 이때, S/2 이상의 큰 객체가 없으면 S/4 이상의 큰 객체들 중에서 LRU로 제거한다. LRUMIN은 이러한 방식으로 새로운 객체의 저장을 위한 여유 공간이 생길 때까지 이 과정을 계속해서 반복한다.

2.2 웹 객체의 참조특성

네트워크의 발전과 멀티미디어 컨텐츠 제작 시술의 발전과 사용자의 다양한 참조 특성은 웹 객체의 이질성을 더욱 심화시키고 있다. 따라서 이러한 웹 객체의 이질성은 캐시의 성능에 많은 영향을 미치고 있다. 사용자가 참조하는 웹서비스의 성격이나 사용자의 개인 참조 특성, 사용환경 특성 및 각종 물리적 특성은 객체의 이질성에 영향을 미친다. 결국 객체의 다양한 이질성은 객체 크기 편차를 크게 하여 빈번한 캐시 대체를 발생시키는 요인이 되고 있다.

그림 1은 객체의 참조빈도비율을 나타낸 것이다. 좌변의 눈금은 참조빈도비율을 % 단위로 나타낸 것이다. 또한 그림 2는 로그분석을 통해 객체의 객체크기 누적비율을 좌변의 참조빈도비율을 % 단위로 나타낸 것이다. 실험결과의 분석 결과 크기가 10KB 미만인 객체에 대한 참조가 가장 빈번히 일어났다. 또한 그림 3은 총전송량비율을 좌변의 % 단위로 나타낸 것이다. 또한 그림 4는 총전송량누적비율을 좌변의 %단위로 나타낸 것이다. 총전송량면에서는 10KB-40KB 객체의 비율이 가장 높으며 총전송량누적비율에서도 가장 급격한 누적이 발생함을 알 수 있다.

웹 캐싱은 일반적인 캐싱과는 달리 네트워크 트래픽을 고려해야만 한다. 이러한 관점에서 보면 총전송량 비율은 중요한 의미를 가진다. 총전송량의 비율이 가장 높다는 것은 네트워크 트래픽에 가장 많은 영향을 준다는 것을 의미한다. 로그 분석을 통해 알 수 있는 웹 캐싱의 성능에 영향을 미치는 요인은 다음과 같다. 첫 번째, 큰 크기의 객체보다 작은 크기의 객체에 대한 요청 횟수가 빈번하며 크기가 작은 객체의 빈번한 대체 요구에 대한 캐시 부재는 캐시의 효율을 떨어뜨리는 요인이 된다. 또한 크기가 큰 객체에 대한 적은 횟수의 요청이 네트워크 트래픽의 급격한 증가를 발생시키며, 크기가 큰 객체의 캐시 부재에 의한 캐시 대체는 급격히 효율을 떨어뜨리는 요인이 된다.

그림 1, 2에서 볼 수 있듯이, 10k까지의 요청비율이 전체 참조의 약 85%에 해당한다. 하지만 그림 3, 4에서

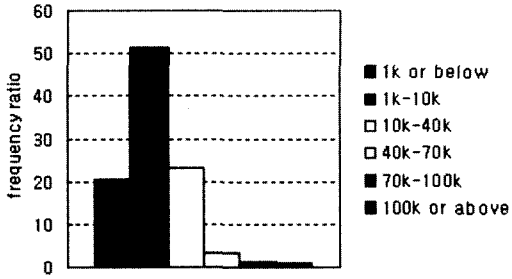

그림 1 Request frequency ratio(%)

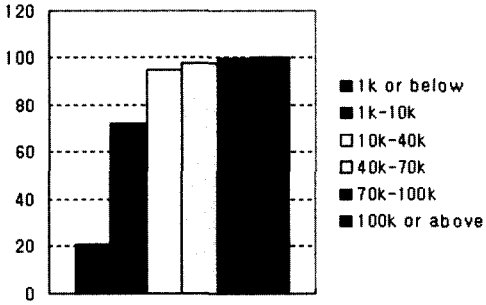

그림 2 Request frequency accumulation ratio(%)

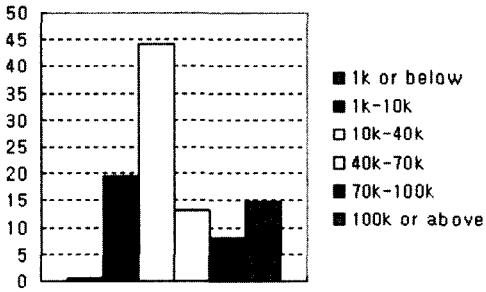

그림 3 Total transmission quantity ratio(%)

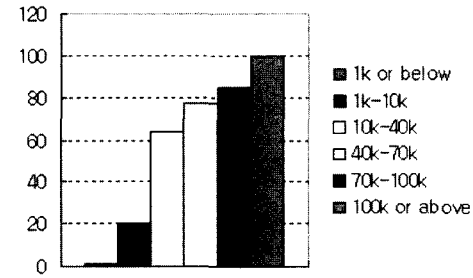

그림 4 Total transmission quantity accumulation ratio(%)

객체의 크기를 고려하면 약 21% 정도 비중을 차지하게 된다. 따라서 웹 캐시의 공간 분할에 대해 단순 크기를 비교한다면 약 2:8의 결과가 나타날 것이나, 객체의 참

조빈도수를 고려하면 객체의 분할 비율은 좀 더 다양성을 가질 수 있을 것이다. 또한 향후 분할의 기준이 되는 객체의 크기에 대해서는 네트워크의 속도향상, 캐시의 성능, 웹서비스의 성격을 고려한 추가적인 연구가 필요할 것이다.

3. ACASH

3.1 ACASH 동작

객체 이질성의 심화는 더욱 빈번한 캐시 대체를 발생하며, 객체 이질성의 심화는 객체의 크기 편차를 크게 한다. 객체의 크기에 따른 분할은 각 분할된 영역에서 객체의 크기 편차를 줄여 객체 이질성을 감소시킬수 있다. ACASH는 캐시의 저장공간을 객체의 크기에 따라 두 개의 영역으로 분할하여 관리한다. 이때 고려해야할 점은 다음과 같다,

- 웹 캐시에는 가능한 많은 웹 객체를 저장하는 것이 좋다. 이런 관점에서는 크기가 작은 웹객체를 캐시에 저장하는 것이 유리하다.
- 언급한 바와 같이, 크기가 큰 객체의 대체는 많은 수의 크기가 작은 객체를 한꺼번에 대체시킨다. 이 경우 급격한 캐시의 성능 저하를 가져온다. 크기가 큰 웹 객체의 캐시 부재는 네트워크 트래픽을 급격히 증가시켜, 시스템의 성능을 떨어뜨리게 된다. 결국 네트워크 측면에서는 크기가 큰 객체를 캐시에 저장하는 것이 유리하다.

또한 앞의 웹 객체 참조 특성 분석에서 살펴본 것과 같이 객체크기 10k를 기준으로 총전송량과 참조빈도수의 급격한 차이점을 구분할 수 있다. 따라서 ACASH에서는 LARGE는 10K 이상, SMALL은 10K 미만의 웹 객체를 구분 관리한다. 이때 각 영역의 분할 비율의 가변적인 관리를 통해 시간의 흐름에 따른 웹 객체의 참조 특성에 대해 적응성을 갖게 하고 있다. 그림 5는 ACASH를 나타낸 것이다. 클라이언트의 객체 요청이 들어오면 캐시 관리자는 객체의 크기에 따라 해당되는 분할 영역에 객체의 존재 여부를 확인한다. 이때 캐시적중(cache hit)이 발생하면 클라이언트가 요구한 객체의 서비스를 제공하게 된다. 이 때, 캐시 관리자는 이 객체가 LRUMIN 대체에서 높은 순위를 할당받게 하기 위해 이용된 시간 기록을 갱신한다.

만일 cache miss가 발생하면 관리자는 해당 URL의 인터넷 서버에 객체 서비스를 요청하여 전송받는다. 전송 받은 객체는 크기에 따라 해당 등급으로 분류되고 캐시 관리자는 해당 등급의 캐시 영역에 이 객체를 저장할 공간이 있는지를 확인한다. 저장할 공간이 있을 경우 객체를 캐시에 저장하게 되고, 없을 경우 해당 캐시 영역에 LRUMIN에 의해 여유 공간을 배정하고 객체를

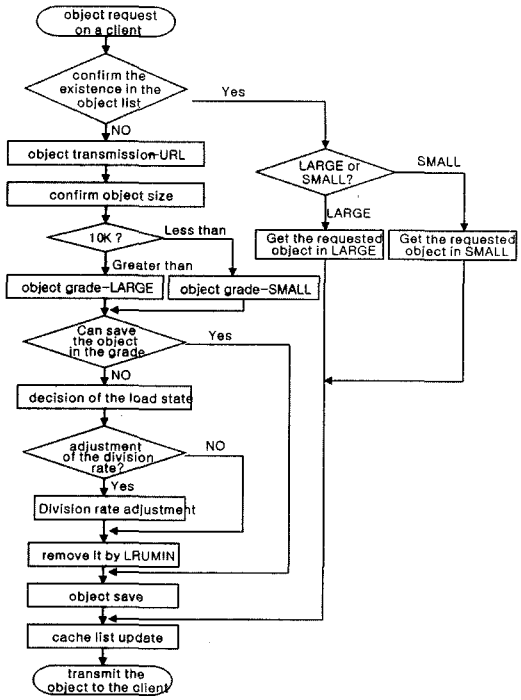

그림 5 ACASH 플로우차트

저장한다. 각 부분에 저장된 웹 객체는 같은 등급의 객체들 사이에서만 대체되며 새로 들어온 객체의 시간 기록을 저장하여 높은 순위를 할당받는다. 이때 ACASH는 객체의 저장공간을 SMALL, LARGE로 객체의 크기로 분할하여 관리하기 때문에 LRU, LRUMIN, SIZE에 비해 관리는 웹 객체의 크기 편차가 상대적으로 작다. 따라서 기존의 기법에 비해 크기가 하나의 큰 크기 객체에 의해 발생하는 작은 크기 객체의 숫자를 줄일 수 있다.

또한, ACASH는 웹 객체의 참조 특성의 시간에 따른 변화를 적응적으로 반영하고 있다. 이를 위해 그림 6과 같은 ACASH ADAPTOR를 사용한다. ACASH ADAPTOR는 분할 비율의 적응성을 갖기 위해 사용되며 동작은 다음과 같다.

1) 각 영역은 load의 상태에 따라 lightly loaded state, lightly overloaded state, overloaded state로 구분된다. 객체의 크기 편차를 반영하기 위해 LARGE 영역과 SMALL 영역의 load state의 기준 비율은 그림 6과 같이 차이를 가진다.
2) 각 영역의 load state는 처음 5:5로 분할된 각 분할 영역에 대해 객체의 참조에 따라 증가한다.
3) 시간이 흐름에 따라 각 분할의 ACASH ADAPTOR의 load state는 객체의 참조특성을 반영하여 변화한다. 이 때 한 영역의 load state가 overloaded state

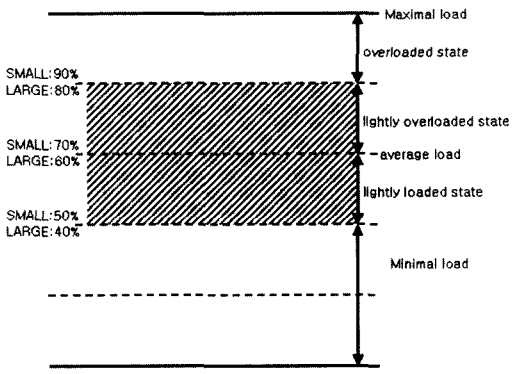

그림 6 ACASH ADAPTOR

에 도달하면 다른 영역의 load state를 확인한다.

3-1) 이때, 다른 영역의 load state가 lightly overloaded state 보다 작은 경우 overloaded state 분할에 대해 전체 캐시에서의 분할 비율을 5% 증가시킨다.

3-2) 이때, 다른 영역의 load state가 lightly overloaded state이면 분할 비율의 조정 없이 진행한다.

4) 분할 비율의 조정에서 어느 한 영역의 비율이 70%를 넘지 못한다. 즉, 최대 분할 비율은 7:3 또는 3:7이 된다.

표 1은 기존의 대체 기법인 LRU, LRUMIN, SIZE와 ACASH의 특성을 비교한 것이다. 제안한 기법은 LRU나 LRUMIN처럼 객체의 크기를 기반으로 한다는 점에서 비슷하다. 하지만 LRU는 과거의 객체 참조 정보 중에서 객체가 참조된 직전의 시각과 객체의 크기를 참조하고 있으며, 참조 빈도와 객체의 이질성을 반영하고 있지 않다. 또한 LRUMIN은 근본적으로는 LRU와 비슷하게 동작하지만 객체의 사이즈에 대한 고려에서 최소한의 객체를 대체하기 위해 작은 크기의 객체를 우선한다는 점이 다르다. 하지만 LRUMIN 역시 객체의 이질성을 반영하고 있지 않다. 하지만 제안 기법은 객체의 크기와 이에 따른 이질성을 충분히 반영하고 있으며 적응성을 갖고 있다. 또한 ACASH는 참조특성에 따른 적응성을 갖고 있다.

표 1 특성 비교

기법	참조성향	참조빈도	객체이질성	객체크기 편차	참조특성 적응성
LRU	직전의 참조 시각	고려치 않음	고려하지 않음	크다	없음
LRUMIN	직전의 참조 시각	고려치 않음	고려하지 않음	크다	없음
SIZE	직전의 참조 시각	고려치 않음	고려하지 않음	크다	없음
ACASH	직전의 참조 시각	고려	이질성 고려	작다	있음

4. 실험 및 분석

4.1 성능 평가 배경

일반적으로 대체기법의 성능을 평가하기 위해서는 먼저, 객체적중률을 측정한다. 하지만 웹 캐싱의 대체 기법 성능 평가를 위해서는 네트워크 트래픽에 영향을 미치는 요인을 포함한 실험이 필요하다. 따라서 본 연구에서는 다음과 같은 평균객체크기적중률, 응답시간을 실험하였다.

1) 응답시간

대체 기법의 성능을 평가하는 중요한 척도는 응답시간(RT)이다. 응답시간 RT는 클라이언트가 객체를 요청한 후 응답을 받을 때까지 걸리는 시간이다. 클라이언트의 객체 요구에 대한 응답시간은 다음과 같은 10가지 지연 요소를 갖는다.

① $TDT_{client\_to\_cache}$: 클라이언트에서 캐시서버간의 객체 요구에 대한 전송 지연시간

② $LSJDT_{AA}$: ACASH ADAPTOR의 load state 확인 지연시간

③ $PADT_{AA}$: ACASH ADAPTOR의 분할 조정 지연시간

④ DDT_{cache} : 캐시서버의 cache hit 또는 cache miss의 판별 지연시간

⑤ SDT_{cache} : Large 또는 small 영역에 저장된 객체의 캐시 검색 지연시간

⑥ $TDT_{cache\_to\_client}$: 캐시서버에서 클라이언트로 객체 전송 지연시간

⑦ $TDT_{cache\_to\_URL}$: 캐시서버에서 URL로 객체 요구 전송 지연시간

⑧ $TDT_{URL\_to\_cache}$: URL에서 캐시서버로 객체 전송 지연시간

⑨ RDT_{cache} : 캐시서버에서 객체 교체 지연시간

⑩ UT_{cache} : 캐시리스트 갱신 지연시간

응답시간에는 cache hit의 응답시간 RT_{ch}(Response Time of cache hit)와 cache miss의 응답시간 RT_{cm}(Response Time of cache miss)가 있다.

① cache hit의 경우

$$RTch=TDT_{client\_to\_cache}+DDT_{cache}+SDT_{cache}+TDT_{cache\_to\_client}+UT_{cache} \quad (1)$$

② cache miss의 경우 : 분할 조정이 필요 없는 경우

$$RTcm=TDT_{client\_to\_cache}+LSJDT_{AA}+DDT_{cache}+TDT_{cache\_to\_URL}+TDT_{URL\_to\_cache}+RDT_{cache}+TDT_{cache\_to\_client}+UT_{cache} \quad (2)$$

③ cache miss의 경우 : 분할 조정이 필요한 경우

$RTcm=TDT_{client\_to\_cache}+LSJDT_{AA}+PADT_{AA}+DDT_{cache}+TDT_{cache\_to\_URL}+TDT_{URL\_to\_cache}+RDT_{cache}+TDT_{cache\_to\_client}+UT_{cache}$ (3)

응답속도는 객체적중률과 밀접한 관계를 가지게 된다. 식(2)와 같이 요구된 객체가 캐시에서 적중한 경우 응답시간은 5단계의 지연이 발생하지만, 비적중의 경우 ACASH ADAPTOR의 load state에 따라 식(2), 식(3)과 같이 8단계 또는 9단계의 지연이 발생한다. 따라서 ACASH에서는 ADAPTOR에 따른 load가 추가적으로 발생한다. 하지만, 상기 언급된 10가지 지연시간 중에서 $TDT_{client\_to\_cache}$, $TDT_{cache\_to\_client}$, $TDT_{cache\_to\_URL}$, $TDT_{URL_to_cache}$의 4가지 지연시간이 네트워크의 물리적인 영향을 가장 많이 받으며, 웹 캐싱의 성능에 큰 영향을 주기 때문에 ADAPTOR로 DS해 발생하는 load 보다는 ACASH로 인해 cache miss를 줄일 수 있다면, 웹 캐싱의 성능 향상을 가져올 수 있게 된다.

식 (2)나 식 (3)의 cache miss의 경우 식 (1)의 cache hit보다 응답에 걸리는 지연 시간이 상당히 길어진다. 이 식을 통해 결국 웹 캐싱의 성능 향상을 위한 근본적인 방법에 대한 본 논문에서의 접근에 대해 두 가지 결론을 얻을 수 있다.

① cache hit 비율의 향상을 통한 cache miss의 감소

② 네트워크의 물리적인 영향을 받는 4가지의 지연요인의 감소 : 큰 크기의 웹객체에 대한 적중률의 증가

2) 평균객체크기적중률

웹 객체는 이질성의 편차가 매우 크다. 또한 객체의 크기 요인으로 인해 대체시 발생하는 트래픽 및 응답시간의 편차 또한 크다. 앞의 응답시간에서 다룬 것과 같이 웹 환경은 네트워크의 물리적인 환경에 영향을 받기 때문에, 트래픽에 영향을 끼치는 요인에 대해 고려해야한다. 트래픽에 영향을 미치는 다양한 요인 중에 객체 자신이 가지는 요인은 객체의 크기이다. 따라서 웹 객체의 크기 요인을 반영할 수 있어야 한다. 식 (4)의 평균객체크기적중률은 객체적중률에 객체의 크기에 대한 요인을 반영한 것이다. 평균객체크기적중률은 요청한 객체에 대한 객체적중률과 객체의 크기를 평균값으로 나타낸 것이다.

$$\frac{\sum_{j=0}^{k}(OB_{hit(j)} \times SO_{hit(j)})}{\sum_{i=1}^{n}(OB_{req(i)} \times SO_{req(i)})} \quad (4)$$

$OB_{req(i)}$: requested object

$SO_{req(i)}$: size of requested object

$OB_{hit(i)}$: hit object

$OB_{hit(i)}$: size of hit object

4.2 실험

ACASH의 성능을 평가하기 위해 표 2와 같은 두 개의 서로 다른 객체 참조 특성을 가진 사용자 실험그룹을 설정하였다. 실험모델1은 객체의 편차가 비교적 작고, 작은 크기의 객체 참조가 많은 경우를 설정하였고, 실험모델2는 이와는 다르게 객체의 편차가 비교적 작고, 큰 크기의 객체 참조가 많은 경우를 설정하였다. 이를 위해 실험모델1의 사용자 그룹은 주로 텍스트 문서가 많이 포함된 웹사이트와 비교적 작은 크기의 GIF, JPEG, AVI가 포함된 웹사이트(자료실과 게시판 위주의 전문 커뮤니티 사이트, 지역 정보사이트, 텍스트와 사진 위주의 관광 정보 사이트, 사이버 신문 사이트 등)를 참조하게 하였다. 또한 실험모델2의 동일한 사용자 그룹에 대해 크기가 큰 그래픽 객체 및 MPEG 등의 동영상 객체의 비중이 높은 웹사이트(방송국사이트, 10대 위주의 커뮤니티 사이트, 전자상거래사이트, 영화사이트, 음악프로 사이트 등)를 참조하게 하였다.

표 2 실험모델 특성

구분	참조 특성	객체 크기 편차
실험모델1	- 많은 양의 HTML 텍스트 포함 - 크기가 작은 GIF, JPEG, AVI 의 비중이 높음	상대적으로 작다
실험모델2	- 크기가 큰 그래픽 객체 및 MPEG 등의 동영상 객체의 비중이 높음	상대적으로 크다

표 3 분할의 변화

구분	영역	T1	T2	T3	T4	T5	T6	T7	T8
실험모델1	LARGE	5	5	4.5	4.5	4.5	4	4.5	4
	SMALL	5	5	5.5	5.5	5.5	6	6.5	6
실험모델2	LARGE	5	5.5	5.5	6	6.5	6.5	6.5	7
	SMALL	5	4.5	4.5	4	3.5	3.5	3.5	3

실험을 통해 이 두 가지의 실험모델에 대해 객체적중률과 객체평균적중률과 응답시간을 측정하고 기존 기법인 LRU, LRUMIN, SIZE와 비교 평가하였다.

표 3은 실험모델1과 실험모델2에 대해 시간구간별(T1,T2, …,T8)로 ACASH ADAPTOR의 load state 변화를 나타낸 것이다. 시간구간은 객체의 참조특성과 캐시용량에 따라 변하기 때문에 구간의 시간 간격은 일정치 않으며, 시간구간의 값보다는 각 영역의 분할 비율의 변화가 더 큰 의미를 갖는다. 분할의 변화에서 알 수 있듯이 실험모델2의 분할 변화가 실험모델1의 분할 변화보다 훨씬 빠르고 크다. 그것은 실험모델2가 실험모델1에 비해 참조하는 객체의 크기가 크기 때문에, 캐시를 빠른 속도로 채웠기 때문이다.

실험은 실험모델1과 실험모델2 각각에 대해 객체적중률과 객체크기적중률, 응답시간을 측정하였다. 그림 7, 8의 객체적중률(object hot ratio)은 실험에 대한 객체의 적중률을 나타낸 것이고, 객체크기적중률은 객체의 크기

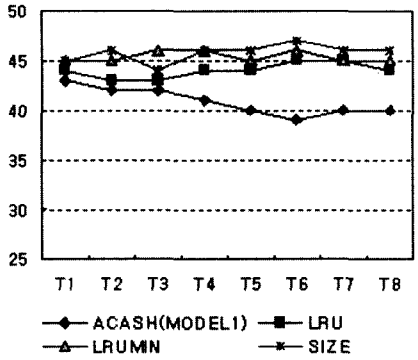

그림 7 object hit ratio with 실험모델1

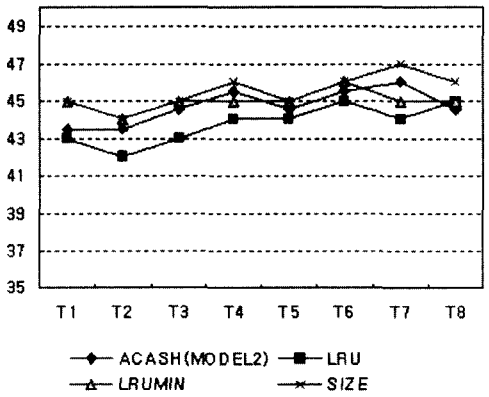

그림 8 object hit ratio with 실험모델2

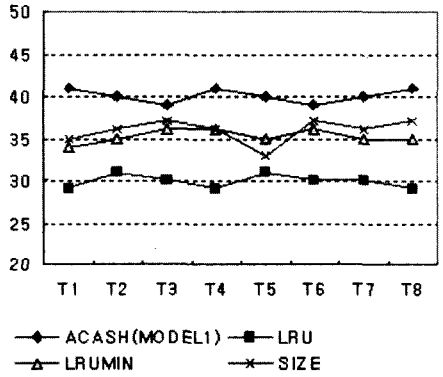

그림 9 object size hit ratio with 실험모델1

를 고려한 객체적중률의 값을 나타낸 것이다. 따라서 크기가 큰 객체의 적중률의 향상은 객체크기적중률의 향상을 가져올 수 있으며, 객체크기적중률의 향상은 네트워크를 기반으로 하는 웹 환경의 트래픽을 줄일 수 있게 한다.

그림 7의 객체적중률의 실험결과 제안 기법이 실험모델1에 대해 오히려 성능이 좋지 않음을 알 수 있었다.

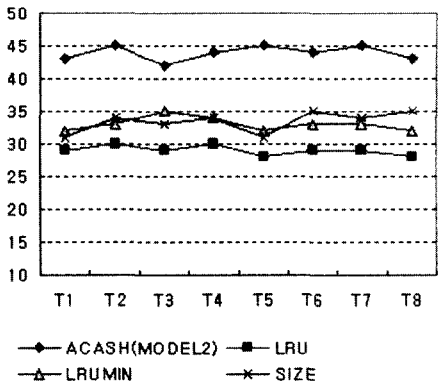

그림 10 object size hit ratio with 실험모델2

그것은 실험모델1의 객체 특성이 작은 크기 객체에 대해 빈번한 객체 요청이 일어나기 때문에, 분할 영역을 사용하지 않는 기존 기법들 보다 LARGE영역의 분할로 인한 SMALL영역의 크기 감소로 인한 것이다.

하지만 큰 크기 객체에 대한 객체적중률은 기존 기법들에 비해 증가하였기 때문에 그림 9와 같이 객체의 크기를 고려한 실험에서는 평균 15% 정도의 성능 향상을 나타내고 있다.

그림 8은 실험모델2에 대한 객체적중률의 실험결과이다. 실험모델1에 비해서는 객체적중률이 다소 증가한 것을 알 수 있다. 이것은 큰 크기 객체에 대한 요청이 많아져서 객체적중률의 평균값이 상승하였기 때문이다. 하지만 실험모델1에 대한 실험과 비슷한 이유로 객체적중률이 기존 기법들과 비슷하거나 다소 감소한 것을 알 수 있다. 큰 크기 객체의 객체적중률의 향상은 객체의 크기를 반영한 그림 10의 실험에서 기존기법들보다 평균 30% 정도의 성능 향상을 나타내고 있다. 또한 캐시 전용서버의 사용이 늘고 있고 캐시의 용량이 커지고 있어 기존기법과 ACASH의 객체적중률 차이가 점차 줄어들 것이다.

그림 11과 그림 12는 두 가지 실험 모델에 대한 응답시간을 측정한 것이다. 그림에서 MODEL1은 실험모델1에 대한 ACASH의 실험을 나타낸 것이고, MODEL2는 실험모델2에 대한 ACASH의 실험을 나타낸 것이다. 실험모델1에 대한 응답시간이 실험모델2에 비해 다소 빠르다. 이것은 실험모델2가 실험모델1에 비해 참조하는 객체의 크기가 크기 때문에 발생하는 지연시간 때문이다. 기존기법과 비교하면 큰 크기 객체에 대한 참조가 많은 실험모델2의 응답시간 향상률이 실험모델1에 비해 높은 것을 알 수 있다.

실험모델2에 대한 ACASH의 실험결과 객체적중은 기존 기법과 비슷하지만 제안기법이 약간 떨어짐, 이것은 작은 크기 객체의 적중률은 떨어지고, 큰 크기 객체

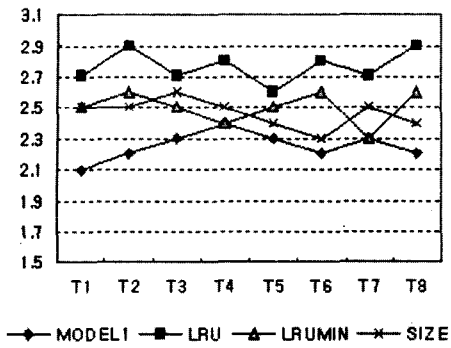

그림 11 response time with 실험모델1

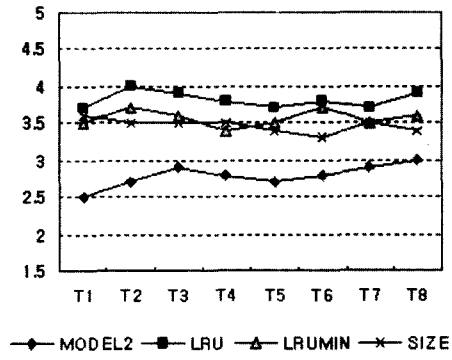

그림 12 response time with 실험모델2

의 적중률은 향상되었지만, 전체적인 숫자면에서는 약간 감소되었기 때문이다. 또한 평균객체크기적중률은 큰 성능의 향상을 알 수 있다. 큰 크기 객체의 객체적중률의 향상은 객체크기적중률과 응답시간에 많은 영향을 미친다. 실험결과 객체적중률이 다소 감소하더라도 객체크기적중률과 응답시간이 크게 향상되었음을 알 수 있다. 응답시간은 ACASH가 제안기법이 20%이상 향상되었다. 이것은 특히 트래픽의 영향이 많은 환경에서 더 효율적일 것이다. 웹 캐싱은 네트워크를 기반으로 하고 있기 때문에 응답시간의 향상은 중요한 의미를 가진다. 따라서 객체크기적중률의 향상과 이에 따른 응답시간의 향상은 ACASH의 우수성을 입증하는 것이다. 또한 실제 로그분석 결과 실험모델1과 같은 경우보다는 실험모델2와 같은 경우가 많이 발생하며, 특히 인터넷의 주된 사용자인 10대와 20대에서의 특성이 실험모델2와 비슷하였다. 따라서 실험모델2에 대한 성능의 향상은 ACASH의 효용성을 입증하는 것이다.

5. 결 론

웹 사용자들의 다양한 참조특성은 웹 객체 이질성을 더욱 심화시키고 있지만, 전통적인 대체 기법에서는 이것을 충분히 반영하지 못하고 있다. 따라서, 객체의 참조특성을 가변적으로 반영한 웹 캐싱 기법이 필요하다. 본 연구에서는 웹 객체의 참조특성과 이질성을 적응적으로 반영한 웹 캐싱 기법인 ACASH를 제안하였다. ACASH는 객체의 참조특성을 기반으로 이질성을 반영하고 있으며, 시간의 흐름에 따른 사용자의 참조특성을 적응적으로 반영할 수 있다. ACASH의 성능을 평가하기 위해 객체 이질성을 반영한 두 가지 실험 모델에 대해 기존의 대체 기법들과 객체적중률 및 객체크기적중률, 응답시간을 비교 실험하였다. 실험결과 ACASH가 기존의 대체 기법들에 비해 객체크기를 반영한 척도에서 성능의 향상을 확인할 수 있었다.

본 연구를 통해 제안한 ACASH는 키를 기반으로 객체의 참조특성을 반영한 대체 기법이다. 향후에는 키와 비용을 기반으로 한 혼합형 기법에 대한 연구를 통해 웹 객체의 이질성과 네트워크 상에서의 전송 비용을 반영한 기법의 개발이 필요하다.

참 고 문 헌

[1] G. Barish, K. Obraczka, World Wide Web Caching: Trends and Techniques. IEEE Communications, Internet Technology Series, May 2000.

[2] H. Bahn, S. Noh, S. L. Min, and K. Koh, "Efficient Replacement of Nonuniform Objects in Web Caches," IEEE Computer, Vol.35, No.6, pp.65-73, June 2002.

[3] L. Rizzo, L. Vicisano, "Replacement Polices for a Proxy Cache," IEEE/ACM Trans. Networking, vol.8, no.2, pp.158-170, 2000.

[4] S. Williams, M. Abrams, C. R. Standridge, G. Abhulla and E. A. Fox, "Removal Policies in Network Caches for World Wide Web Objects," Proc. 1996 ACM Sigcomm, pp.293-304, 1996.

[5] V. Almcida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing Reference Locality in the WWW," In Proc. of the 4th Int'l Conf. on Parallel and Distributed Information Systems, 1996.

[6] J. C. Bolot and P. Hoschka, "Performance engineering of the World-Wide Web: Application to dimensioning and cache design," Proc. of the 5th Int'l WWW Conf., 1996.

[7] M. Abrams, C. Standridge, G. Abdulla, S. Williams and E. Fox, "Caching Proxies: Limitations and Potentials," Proc. 4th Int'l World Wide Conf., 1995.

[8] N. Niclausse, Z. Liu, P. Nain, "A New and Efficient Caching Policy for the World Wide Web," Proc. Workshop on Internet Server Performance (WISP 98), pp.94-107, 1998.

[9] C. Aggarwal, J. Wolf and P. Yu, "Caching on the World Wide Web," IEEE Trans. Knowledge and Data Engineering, vol 11, no.1, pp.94-107, 1999.

[10] S. Sahu, P. Shenoy, D. Towsley, "Design Considerations for Integrated Proxy Servers," Proc. of IEEE NOSSDAV'99, pp.247-250, June, 1999.
[11] J. Wang, "A Survey of Web Caching Schemes for the Internet," ACM Computer Communication Review, 29, pp.36-46, October, 1999.
[12] E. Cohen and H. Kaplan, "Exploiting Regularities in Web Traffic Patterns for Cache Replacement," In Proceedings of the 31st Annual ACM Symposium on Theory of Computing, ACM, 1999.
[13] C. D. Murta, Virgilio A. F. Almeida, Jr. W. Meira, "Analyzing Performance of Partitioned Caches for the WWW," In Proceedings of the Third International WWW Caching Workshop, Manchester, England, June, 1998.
[14] Annie P Foong, Yu-Hen Hu, Dennis M Heisey, "Web Caching: Locality of References Revisited," IEEE International Conference on Networks (ICON'00), Singapore, September 05-08, 2000.

고 일 석

연세대 컴퓨터산업시스템공학(박사수료) 미)USIU 경영학과(MBA). 경북대 컴퓨터공학(공학석사). 경북대 컴퓨터공학(공학사). 현재 충북과학대학 전자상거래과 조교수. 관심분야는 웹기반시스템, 멀티미디어, 네트워크성능관리

임 춘 성

미)UC Berkeley 산업공학과(공학박사) 서울대학교 산업공학과(공학석사). 서울대학교 산업공학과(공학사). 1993년~1995년 미) Rutgers University 산업공학과 조교수. 1995년~현재 연세대학교 정보산업전공 부교수. 관심분야는 전자상거래, 기업정보화방법론, CBD

나 윤 지

충북대 컴퓨터공학(공학박사). 미)NYIT Communication ART 전공. 충북대 컴퓨터공학(공학석사). 경북대 생명공학(이학사). 관심분야는 멀티미디어, 게임이론, 콘텐트공학, 웹기반시스템