RwO-캐싱 : 연관 웹 객체 기반의 웹 캐싱 기법 연구

RwO-Caching : A Study on Web Caching with Related Web Object

나희성(Huiseong Na)*, 고일석(Franz I.S. Ko)**

초 록

웹 사용자의 폭발적인 증가는 웹 시스템의 부하 및 네트워크 트래픽을 증가시키는 가장 큰 원인이 되고 있으며, 이는 웹 사용자의 서비스 만족도를 떨어뜨리고 있다. 이러한 현상을 극복하기 위해서 웹 콘텐트 가속화에 대한 연구가 활발히 진행되고 있다. 웹 캐싱은 클라이언트 요청을 신속하게 처리하고, 네트워크에 대한 트래픽을 줄여 전체 웹 시스템의 부하를 줄이게 하는 가속화 기법 중의 하나이다. 본 논문은 웹 처리의 단위인 객체의 특성을 기반으로 한 연관 웹 객체(Related Web Object : RwO)를 기반으로 하는 새로운 웹 캐싱 기법을 제안하였다. 또한 실험을 통해 기존의 기법과 제안기법의 성능을 비교함으로서 본 연구의 유용성을 입증하였다.

ABSTRACT

The most important reason for increasing web traffic and overloaded web servers is a dramatic growth of web users which leads to a great dissatisfaction of each individual user. For overcoming this situation, studies for an acceleration of web content have been conducted actively. We use web caching technology for reducing the load of the web system and traffic in the network. In this paper, we proposed a new web caching technology with the related web object which based on the unit of web processing and characteristics of the web object. Also we verified the availability of the proposed system with comparison and experiments.,

키워드 : 웹 캐싱, RwO 웹 캐싱, 연관객체, 프리패칭, 웹 가속화
Web Caching, RwO-Web Caching, Related Content, Prefetching, Web Acceleration

---

This work was supported by the Korea Research Foundation Grant (KRF-2005-D00685).
* 동국대학교 컴퓨터 멀티미디어 학부 석사과정
** 동국대학교 컴퓨터 멀티미디어 학부 교수
2008년 10월 02일 접수, 2008년 10월 31일 심사완료 후 2008년 11월 08일 게재확정.

1. 서 론

웹 사용자의 급격한 증가로 네트워크를 통해 전송되는 데이터의 용량과 트래픽이 갈수록 증가하고 있다. 이에 따라 인터넷 백본의 용량이 해마다 60% 정도 증가 추세이며, 실제로는 이를 초과하는 웹 서비스로 인해 백본 증가 비율 이상의 대역폭을 요구하고 있는 실정이다. 이러한 현상은 네트워크의 트래픽을 더욱 가중시키고 있다. 이러한 현상을 극복하기 위해서 웹 콘텐트 가속화에 대한 연구가 활발히 진행되고 있다.

웹의 가속화를 위한 서버측 기술로는 부하밸런싱(Load Balancing)과 분산처리(Distributed Processing) 등이 있으며, 클라이언트 측면에서 일반적으로 사용되는 기술은 웹 캐싱 기법이다.

웹 캐싱은 클라이언트 요청을 신속하게 처리하고, 네트워크에 대한 트래픽을 줄여 전체 웹 시스템의 부하를 줄이게 하는 기술이다. 일반적으로 사용되는 웹 캐싱 기법은 웹 서버와 클라이언트 사이에서 클라이언트가 요청한 웹 객체를 대신하여 처리 및 서비스한다.

이러한 웹 캐싱의 효율을 높이기 위하여 다양한 교체(Replacement) 기법들과 프리패칭(Prefetching) 기법이 사용된다. 프리패칭은 사용자의 요청에 대해 웹 캐시 공간의 적중율을 높이는 중요한 기법이다.

웹은 객체를 단위로 하는 서비스를 제공하고 있기 때문에 이 객체의 특성을 기반으로 하는 효율적인 객체 처리 기법이 필요하다.

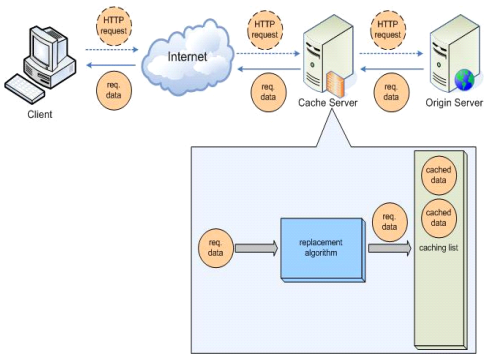
본 연구는 이러한 연구 배경을 통해 웹 처리의 단위인 객체의 특성을 기반으로 한 *연관 웹 객체(Related Web Object : RwO)*를 기반으로 하는 새로운 웹 캐싱 기법을 제안하였다. 또한 실험을 통해 기존의 기법과 제안기법의 성능을 비교함으로서 본 연구의 유용성을 입증하였다.

2. 관련 연구

상기 기술한 바와 같이 웹 객체의 효율적 이용을 위한 관심사는 웹 캐싱과 프리패칭이다. 본 장에서는 이들 두 가지 기법의 기존 연구 기법을 살펴본다.

웹 캐싱은 클라이언트 요청을 신속하게 처리하고, 네트워크에 대한 트래픽을 줄여 전체 웹 시스템의 부하를 줄이게 하는 기술이다. 일반적으로 사용되는 웹 캐싱 기법은 웹 서버와 클라이언트 사이에서 클라이언트가 요청한 웹 오브젝트를 대신하여 처리 및 서비스한다. 이 때 클라이언트의 요청에 대해 클라이언트에서 가장 가까운 캐시 서버에서 응답을 처리하게 되어 사용자에 대한 응답지연율을 감소시킬 수 있게 한다. 더 나아가 이러한 캐시 서버를 사용함으로써 네트워크의 로드를 줄이고 클라이언트에 의해 발생하는 대역폭을 감소시킨다.

<그림 1>은 일반적인 캐싱 기법을 나타낸 것이다. 일반적 웹 캐싱 기법에서 유저는 먼저 자신의 요구할 객체를 인터넷을 통해 요청하고 그 요청을 받은 캐시 서버는 자신의 저장공간을 검사하여 요청된 객체가 존재한다면 그것을 유저에게 제공하고, 존재하지 않을 경우 웹 서버에 해당 객체를 요청하여 자신에게 저장공간에 저장한 후 이것을 유저에게 제공한다.



〈그림 1〉 일반적인 웹 캐싱 기법

캐시 서버의 중요한 기능 중 하나는 클라이언트들로부터 오는 요청들에 대한 조절 기능이다. 클라이언트의 요청에 대해 자신의 메모리로부터 검사하여 있을 경우는 자체적으로 요청에 반응하지만 없을 경우에만 본 서버에 자료를 요구하여 클라이언트에 제공하거나 자신의 메모리로 복사한다[9]. 이러한 방법을 사용함으로 인해 본 서버에 과중되는 부담을 줄이고 부하 균등을 이루려고 한다. 이러한 메커니즘을 사용하여 지연율을 감소시켜 웹을 더욱더 응답적으로 만든다. 또한 트래픽의 조정을 통해 대역폭을 효과적으로 유지하여 좀 더 손쉬운 관리를 하게 된다[13].

이에 사용되는 캐싱 알고리즘은 한정된 저장 영역을 가지고 있으므로 캐시의 저장 영역의 관리가 필요하며, 이를 위해 LRU, LFU, LRU-min, GD-SIZE등과 같은 교체 알고리즘(Replacement Algorithm)이 필요하다.

이러한 캐싱 기법들을 사용함으로 얻을 수 있는 캐시 적중률을 높이기 위해 프리패칭(Prefetchong) 기법이 사용된다.

〈표 1〉 수행위치에 따른 프리패칭

구 분	특 징
Server-initiated prefetching	웹 서버에서 프리패칭이 수행
Client-initiated prefetching	클라이언트에서 프리패칭을 수행
Proxy-initiated prefetching	프락시 서버에서 사용자들의 웹 문서 접근 패턴을 바탕으로 프리패칭

프리패칭은 <표 1>과 같이 세가지 위치에서 수행될 수 있다.

- 웹 서버에서 수행될 경우 : 어떤 객체가 요청될 경우 그 다음에 따라올 요청을 예측하고 예측된 객체를 요청한 곳에 푸싱(Pushing)한다[2, 10, 11].
- 클라이언트에서 프리패칭할 경우 : 과거의 접근 패턴을 모니터할 수 있는 사용자 에이전트(Agent)를 바탕으로 프리패칭할 웹 문서를 초기화 할 수 있고 프리패칭 가능하다[4, 11, 14].
- 프락시 서버에서 프리패칭할 경우 : 클라이언트로부터 웹 객체에 대한 요청을 받은 후 바로 다음에 요청할 웹 객체를 예측하여 프리패칭을 수행하거나, 일괄적으로 프리패칭할 웹 문서들을 선정하여 일정한 시간대에 수행한다[5, 8, 12].

〈표 2〉 프리패칭될 객체의 선정

구 분	특 징
Statistical prefetching	클라이언트들에 대한 최근의 로그를 바탕
Deterministic prefetching	관리자나 사용자가 프리패칭될 웹 객체를 하나하나 지정

<표 2>는 프리패칭될 객체의 선정 방법에 대한 연구를 분류한 것이다. 프리패칭될 객체를 결정할 경우 어떤 객체를 대상으로 할 것인지는 중요한 관건이 된다. 이 경우 통계적 방법(Statistical prefetching)[3]과 결정적 방법(Deterministic prefetching)[11]이 있는데 통계적 방법은 클라이언트의 로그를 바탕으로 하여 통계를 낸 후 그 결과를 프리패칭하기 때문에 결정적 방법에 비해 다수의 로드가 발생 한다. 결정적 방법은 인간이 직접 프리패칭될 객체를 지정하는 방법이므로 웹 전체에 시행하기에는 무리가 있다.

하지만, 어떤 방법으로 프리패칭을 하더라도 불필요한 객체를 프리패칭할 수 있다. 그 결과로 프리패칭을 사용하지 않는 시스템보다 더 많은 요청을 만들어내어 본 서버에 추가 요청을 만들어 낸다.

추가 요청은 프리패칭의 두 가지 비용적 문제를 야기한다. 하나는 추가적인 로드의 발생이고 두 번째는 네트워크의 대역폭 여분이 감소이다. 이때 비용의 양을 정하는 것이 중요한 이유는 프리패칭을 사용하는 클라이언트나 사용하지 않는 클라이언트 혹은 둘 다에서 더 나쁜 결과를 나타낼 수 있기 때문이다. 일부 연구는 총 초과 요청이나 총 초과 바이트와 같은 정확한 측정을 통해 비용을 계산하고 폭발적 요청이 프리패칭에 미치는 영향과 프리패칭에 의한 여분의 대역폭 감소로 인해 네트워크의 일부분이 병목현상을 일으키는지 연구하고 있다.

또한 프리패칭은 추가적인 위험요소를 가지고 있다. 이것은 프리패칭에 대해 정보를 제공하지 않는 경우 미래 행동에 대한 예측을 하기가 힘들고, 잘못된 예측은 프리패칭에 따른 추가 비용을 증가시키기 때문이다. 따라서 프리패칭을 성공적으로 활용하기 위한 방안과 이에 대한 이점에 대해 충분히 검토를 하여야 한다.

Kroeger et al.는 공유 프락시 캐시에서 프리패칭에 의한 지연율 감소의 한계에 대해 연구하였다[7]. 이 연구는 약 24,600,000 요청 횟수, 1Mbps의 대역폭, 무제한적 캐시 사용에 대해 시뮬레이션을 통한 연구결과이다. 연구결과 End-to-end 지연율의 대부분을 프락시 서버와 웹 서버 사이에서 보다 프락시 서버와 클라이언트가 사용하고(77%~88%) 프락시 서버에 프리패칭이 사용될 경우 지연율을 60% 아래로 낮출 수 있다는 것이다. 이런 제한된 지연율 감소를 가져오는 가장 중요한 이유는 캐시가 불가능한 객체의 수가 많다는 것이다. 캐시가 불가능한 객체란 캐시가 되지 않거나 이미 프리패칭되어 있는 객체를 말한다.

3. 연관 웹 객체 기반의 웹 캐싱 기법

본 장에서는 연관 웹 객체 기반의 캐싱 기법에 대해 다루고 있다.

3.1 RwO에서의 객체

본 연구를 통해 제안한 연관 웹 객체 기반의 웹 캐싱 기법은 요청 객체에 대해 총괄적으로 연계된 객체를 사용한다. 따라서 이 연구결과는 기존의 어떠한 프리패칭 알고리즘에 적용할 수 있으며, 또한 기존의 캐싱 알고

리즘에도 활용이 가능하다. 다음의 정의 1은 연관객체에 대한 정의이다.

- 정의 1 : 연관 객체
  연관 객체란 요청된 웹 객체의 Hyper-link에 연결된 객체들 중에 의미를 가진 객체들
- 정의 2 : 복합 객체
  복합 객체란 요청된 객체와 연관 객체를 캡슐화시켜 하나로 묶은 객체의 집합

따라서 연관객체는 요청된 웹 객체와 이에 연결된 텍스트 객체, 이미지 객체 및 동영상 객체와 같은 모든 웹의 객체 중에 의미를 가진 것을 말하는 것이다.

이러한 특성은 캐시 메모리 안에서 널리 사용되는 웹 객체의 액티브 집합을 포함하는 Working Set Concept[6]에서 비롯되며, 사용성이 떨어지면 해당 객체를 무시하게 된다. 요청을 받은 객체의 사용성은 그것과 연계된 객체에 직접적으로 연결되어 있다. 다시 말하면 그 객체가 더 이상 필요하지 않게 된다면 그것과 연관된 객체 역시 필요하지 않다는 뜻이다. 만약 연관 객체가 요청된다면 관련 객체 자체와 함께 다른 캡슐 객체로 간주한다. 가장 중요한 점은 프리패칭된 객체가 유저에 의해 요청되는가 되지 않는가 하는 것이다. 요청되지 않는다면 그들은 그들의 연관 객체와 함께 복합 객체로 만들어지지 않고 기억장치에서 사라질 것이다. 요청된다면 또 다른 연관 객체와 함께 복합 객체로서 캡슐화되고 캡슐화된 복합 객체는 프리패칭 될 것이다. 복합 객체를 만드는 동안 캐싱 알고리즘은 복합 객체의 기본 식별자로서 복합 객체의 헤드를 사용한다.

- 정의 3 : 복합 객체의 헤드
  복합 객체의 헤드는 요청받은 객체의 헤드이며 복합 객체의 기본 식별자

연관 웹 객체 기반의 웹 캐싱 기법에서는 기존 기법이 사용하는 객체의 헤드 부분을 대신하여 복합 객체의 헤드부분을 식별한다.

RwO의 기본 개념을 정리하면 다음과 같다.

- 캐싱 알고리즘이 캐시할 객체와 그와 연계된 객체를 캐시 할 때 그것을 하나의 캡슐로 만든다.
- 요청된 객체는 연관된 객체와 함께 캡슐화되고 하나의 객체로서 관리된다.
- 하나로 캡슐화된 객체는 더 이상 사용하지 않거나 필요 없을 경우 연관 객체와 함께 지워진다.

3.2 리스트

위와는 다른 관점에서 볼 때, 캐싱 방법은 두 개의 추상적 데이터의 분류이다. 첫 번째 추상적 관점은 데이터의 실제값을 제공하고, 두 번째 추상적 관점은 데이터 참조는 데이터 값의 지시자(Pointer)를 대신하며 필요한 정보의 이름과 크기와 같은 데이터를 포함한다. 이러한 접근 방식은 많은 복합 객체에 대해 같은 데이터를 공유할 수 있도록 한다. 이런 접근은 웹 페이지에서 같은 이미지와 다른 임베디드 객체를 통해 나타난다. 이 경우 임베디드 객체를 한 번만 캐시하고 재사용시 그 데이터의 값을 나타내는 지시자를

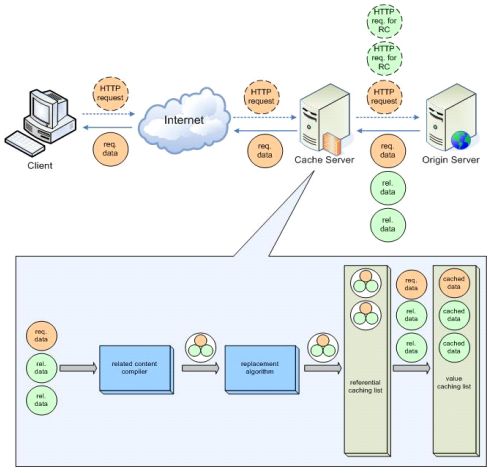
사용하여 캐시 서버가 본 서버에 같은 객체에 대해 반복적으로 캐싱하는 것을 방지한다. 이러한 방법의 결과로 서버 간의 로드를 줄인다. 본 논문에서 제안한 캐싱 기법에서 복합 객체의 데이터 값을 오로지 지시자를 통해 조회한다. 복합 객체가 컴파일 되는 동안 객체 생성자는 첫 번째 검사를 한다. 만약 복합 객체가 캐시되었다면 본 서버에 해당 객체를 요청한다. 요청된 객체가 캐시 리스트에 캐시되었다면 그와 일치하는 지시자가 생성되고 컴파일된 복합 객체의 내부로 지시자를 보낸다. 그리하여 복합 객체의 합성은 임베디드 데이터의 값을 가르키는 지시자가 된다. 복합 객체는 참고 리스트와 같은 다른 리스트에 저장되고 사용되는 모든 교체 알고리즘은 참고 리스트에 있는 복합 객체에 적용된다. 만약 복합 객체가 참고 리스트에서 제거된다면 캐시 리스트에서도 동시에 제거된다. 다시 말하면 복합 객체의 지시자를 조회하는 모든 데이터는 다른 복합 객체에서 참고되지 않을 경우 캐시 리스트에서 제거한다. 캐시된 객체 지시자의 숫자를 추척하는 목적은 캐시 리스트는 각각의 캐시된 데이터에 대해 특별한 카운터를 가져야하고 자신이 지시자가 조회될 때 카운터의 값을 증가시킨다. 그리고 복합 객체가 포인터를 가진 상태에서 참고 리스트에서 제거될 경우 그 값을 감소시킨다. 카운터의 값이 0이라면 지시자가 가르키는 데이터의 값은 캐시 리스트에서 제거된 것이다.

- 정의 4 : 캐시 리스트
  캐시 리스트는 캐시된 데이터의 값을 저장하는 리스트

RwO는 교체 알고리즘의 조작으로부터 캐시 리스트를 제외하며 캐시 리스트는 자기가 지닌 요소에 대해 고유함을 지닐 책임을 가진다.

- 정의 5 : 참고 리스트
  참고 리스트는 복합 객체의 레코드에서 객체가 연결된 타입을 나타내는 리스트

RwO에서는 먼저 객체를 연결하는 캐시 리스트에는 지시자로 구성된 객체가 있고 교체 알고리즘은 참고 리스트에 적용이 된다. 그래서 이것은 직접적으로 캐시 리스트에 접속이 되며 캐시 리스트와 동가화되어 업데이트 된다.



〈그림 2〉 연관 웹 객체 기반의 캐싱 기법

3.3 RwO

<그림 2>는 본 연구에서 제안한 RwO를 나타낸 것이다. RwO는 기본적으로는 일반적인 웹 캐싱 기법들의 개념을 같이 사용하

지만 캐시 서버의 관리 면에서는 큰 차이를 보인다.

유저의 요청을 처리함에 있어 먼저 캐시 리스트에서 요청된 객체를 검색한 후, 다음으로 요구 데이터와 일치하는 복합 객체가 발견된다면 참조 리스트로 교체 알고리즘이 적용된다. 만약 캐시 리스트에서 객체가 발견되지만 참고 리스트는 복합객체가 없다고 확인된다면 지금 요청된 객체는 미리 요청된 객체의 연관 객체라 말할 수 있다. 이 때 요청되는 객체는 연관 객체와 함께 복합 객체로 생성되면서 복합 객체의 헤더부분에 그 내용을 표시하게 된다.

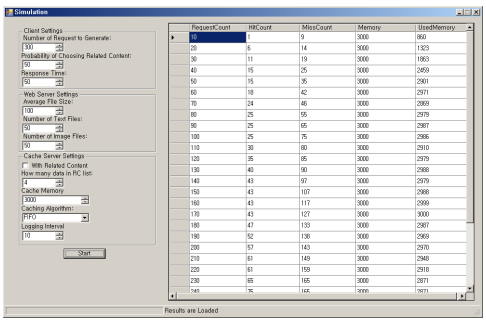
새로 캐시서버에 들어온 복합 객체는 이를 위한 참조 리스트를 만드는 동안 캐시 서버는 이의 저장을 위한 사용 가능한 저장공간을 메모리를 확인한다. 이때 사용 가능한 메모리가 없다면 참조 리스트부터 교체 알고리즘에 따라 캡슐화된 복합 객체를 삭제하고 필요한 만큼의 메모리 용량이 확보될 때까지 삭제작업을 계속한다. 새로운 복합 객체를 위한 공간이 확보되면 이것은 캐시 메모리에 위치하게 된다.

만약 캐싱된 객체를 삭제한다면 제안된 방법은 삭제되는 객체와 연관되면서 다른 객체와 연결되지 않는 데이터를 삭제한다. 만약 삭제할 객체와 연관된 데이터가 있을 경우 그 데이터는 삭제 대상에서 제외된다. 근본적 교체 알고리즘의 원리는 기존의 기법을 사용할 수도 있다.

복합 객체의 요소는 다른 복합 객체에 의해 참조될 수 있다. 복합 객체의 위치는 캐시 리스트를 통해 참조할 수 있고, 그것을 참조할 실제 저장 공간은 캐시 서버의 저장 공간이 된다. 동일한 캐시서버에 저장된 복합 객체의 각 객체는 다른 복합 객체의 단일 객체가 될 수 있으며, 이를 통해 복합 객체의 중복성을 줄일 수 있다.

4. 실험 및 결과

<그림 3>은 실험에 사용된 시뮬레이터의 실행화면을 나타낸 것이다. 본 시뮬레이터는 PC 환경에서 닷넷(.NET) 플랫폼을 기반으로 개발되었다.



〈그림 3〉 시뮬레이션 결과 화면

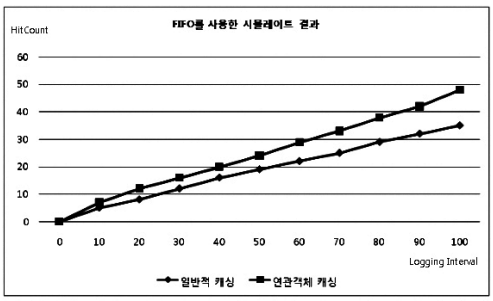
〈표 3〉 실험 인자(Factor)

Factor	
Number of Request to Generate	요청의 숫자
Hit Count	캐시 서버의 적중률
Logging Interval	실험 결과 측정을 위한 타임 스탬프의 단위

<표 3>은 실험에 사용된 주요한 인자들에 대해 나타낸 것이다. Number of Request to Generate은 객체의 요청의 총 횟수이며, 이에 따라 Logging Interval에서 입력한 타임

스탬프에 따라 시뮬레이션 된 값이 출력된다. 예를 들면, Number of Request to Generate의 값이 100이고 Logging Interval값이 10일 경우, 10시간 단위마다 실행 결과가 나타나게 된다. Hit Count는 캐시 서버의 적중률을 뜻한다.

상기한 바와 같이 제안한 RwO 기법은 기존에 개발된 다양한 알고리즘에 적용할 수 있다. 본 실험에서는 가장 일반적인 기법인 FIFO, LRU 및 LRU-min에 대해, 각각 일반적인 방식으로 실험한 경우와 제안 기법을 적용한 경우에 대한 실험을 통해 본 연구의 효용성을 보인다.

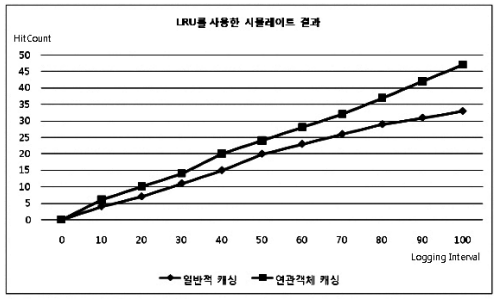


〈그림 4〉 FIFO를 사용한 실험결과

<그림 4>는 클라이언트로부터의 100회 요청과 Logging Interval 10에 대해 캐시 서버에서 FIFO 알고리즘을 사용하였고, 각 Logging Interval당 Hit Count를 측정한 실험 결과이다. 또한 실험 결과는 1000회의 실험을 반복하여 통계를 낸 수치이다.

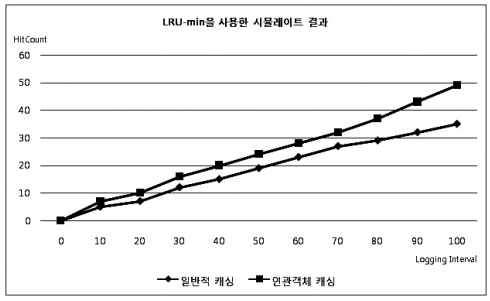
실험결과 기존의 FIFO 기법에 대해 제안한 RwO 기법을 적용한 결과 성능의 향상을 확인할 수 있다.

<그림 5>는 앞서의 실험인 FIFO 알고리즘을 적용한 결과에서 사용된 입력에 대해, LRU알고리즘으로 바꾸어 실험한 결과이다.



〈그림 5〉 LRU를 사용한 실험결과

이 실험에서도 기존의 일반적인 LRU 기법의 실험결과 보다, RwO를 적용한 기법이 더욱 효과적임을 알 수 있다.



〈그림 6〉 LRU-min을 사용한 실험결과

<그림 6>은 상기의 두 실험과 같은 입력값에 대해 LRU-min 알고리즘을 적용한 실험이다. 이 실험 또한 앞의 두 실험과 같이 성능의 향상을 알 수 있으며, 그 결과는 요청의 수가 많을수록 증가됨을 알 수 있다.

이와 같은 세 가지 실험결과 제안 기법인 RwO 기법은 기존의 알고리즘에 부가적으로 적용함으로서 그 성능을 향상시킬 수 있다는 것이다. 또한 실험 그 자체만으로 볼 때는 요청이 횟수가 작을 경우 성능의 큰 차이를 보

여주지는 않으나 요청의 횟수가 늘어남에 따라 Hit Count의 이득이 높아짐을 알 수 있다. 또한 이것은 요청이 많아질수록 연관 객체를 사용하는 일이 자주 발생한다는 것을 나타낸다.

<표 4>를 통해 상기 실험에서 이득률(Gain Ratio)을 살펴보도록 한다.

〈표 4〉 이득률(Gain Ratio)

	FIFO	LRU	LRU-min
10	1.4	1.5	1.4
20	1.37	1.4	1.42
30	1.33	1.27	1.33
40	1.23	1.33	1.33
50	1.19	1.2	1.26
60	1.26	1.21	1.21
70	1.28	1.23	1.18
80	1.24	1.27	1.27
90	1.3	1.35	1.34
100	1.41	1.42	1.39

앞의 분석 결과 횟수 측면에서의 이득율은 요청의 횟수가 많아질수록 높아졌으나, 실제 평균 이득율 면에서는 큰 차이가 없음을 알 수 있다. 이를 통해 제안 기법이 요청이 많은 경우와 그렇지 않은 경우 모두에 대해서 성능 향상을 기대할 수 있음을 알 수 있다.

5. 결 론

웹 사용의 증대는 사용자의 요청에 대한 효율적인 서비스에 대한 중요성을 증가시키고 있으며 웹 캐싱은 이러한 사용자의 요구를 좀 더 빨리 처리하고자 하는 기술적인 접근이다.

본 연구는 웹을 기반으로 하는 서비스 시스템에서 사용자 요청의 가속화를 위한 연구이다. 본 연구에서는 웹 처리의 단위인 객체의 특성을 기반으로 한 연관 웹 객체(Related Web Object : RwO)를 기반으로 하는 새로운 웹 캐싱 기법을 제안하였고 또한 실험을 통해 기존의 기법과 제안기법의 성능을 비교함으로서 본 연구의 유용성을 입증하였다.

본 연구에 이어 현재 연관 웹 객체에 대한 유용성의 범위에 대한 연구와 본 연구를 통해 개발된 시뮬레이터의 기능을 확장하는 연구가 이루어지고 있다.

참 고 문 헌

[1] Abdullaev, S. and Ko, I. S., “An Object-oriented Design of a Simulator for Caching with Related Content using LRU, Advances in Information Sciences and Services,” AICIT Series, S. Korea, Vol. 2, November 2007, pp. 96-103.

[2] Azer Bestavros, “Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Tome in Distributed Information System,” In International Conference on Data Engineering, pp. 180-189, New Orleans, LO, February 1996.

[3] David Barnes and Neil G. Smith,

"An Analysis of World-Wide Web Proxy Cache Performance and its Application to the Modelling and Simulation of Network Traffic," In Proceedings of the Fourth International Conference on Telecommunication Systems Modeling and Analysis, March 1996, http://www.cs.ukc.ac.uk/people/staff/djb/pubs.html.

[4] James Griffioen and Randy Appleton, "Reducing File System Latency using a Predictive Approach," Proceedings of the 1994 Summer USENIX Technical Conference, Boston, Massachusetts, USA, 1994, http://usenix.org/publications/library/proceedings/bos94/griffioen.html.

[5] Ken-ichi Chinen and Suguru Yanaguchi, "An Interactive Prefetching Proxy Server for Improvement of WWW Latency," INET'97, 1997, http://www.isco.org/INET97/procceding/A1/A1_3.HTM.

[6] Ko, I. S., "ACASH : An Adaptive Web Caching method based on the Heterogeneity of Web Object and Reference Characteristics," Information Sciences(ISSN : 0020-0255), ELSEVIER SCIENCE INC., Vol. 176, No. 12, June 2006, pp. 1695-1711.

[7] Kroeger, T. M., Long, D. D. E., and Mogul, J. C., "Exploring the bounds of Web latency reduction from caching and prefetching," In Proc. of the USENIX Symposium on Internet Technologies and Systems, 1997, pp. 13-22.

[8] Li Fan, Quinn Jacobson, Pei Cao, and Wei Lin, "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance," In Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems(SIGMETRICS '99), Atlanta, GA, May 1999, http://www.cs.wisc.edu/~cao/.

[9] Nottingham, M., "Concepts of Web Caching," 2007, http://www.mnot.net/cache_ docs/.

[10] Tomas M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching," In Proceedings of USENIX Symposium on Internet Technology and Systems, December 1997, http://www.usenix.org/publications/library/proceedings/usits97/kroeger.html.

[11] V. Padmanabhan and J. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," Computer Communication Review, Vol. 26, No. 3, July 1996, pp. 22-36.

[12] Wcol Group, "WWW Collector the prefetching proxy server for WWW," 1997, http://shika.aistnara.ac.jp/products/wcol/wcol.html.

[13] Wessels, D., "Web Caching : Making the Most of Your Internet Connection," 2007, http://www.web-cache.com.

[14] Z. Wang and J. Crowcroft, "Prefetching in World Wide Web," IEEE Globecom 96, http://www.cs.ucl.ac.uk/staff/zwang/papers/prefetch.ps.z.

저 자 소 개

나희성 (E-mail : hsna@dongguk.ac.kr)
2006년 충북대학교 컴퓨터 (학사)
현재 동국대학교 전산학부 석사과정

고일석 (E-mail : isko@dongguk.edu)
1996년 경북대학교 컴퓨터공학 (석사)
2000년 USIU MBA 졸업
2006년 연세대학교 박사
현재 동국대학교 교수