제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

웹 상에서의 캐싱기법에 관한 연구

나종원*
*동강대학 정보통신과
e-mail:najwon@dkc.ac.kr

A Study of Web Cashing & World Wide Web

Jong-Won Na*
*Dept.of Information & Communication, Dongkang College

요 약

정보 공유를 위한 거대한 정보 시스템인 월드 와이드 웹의 폭발적인 증가는 기존 네트워크의 트래픽을 유발시키고 서버의 부하를 가져오는 가장 큰 원인이 되고 있다. 많은 요청으로 인하여 원활한 서비스를 제공하기 어렵고, 또한 만족할만한 응답시간을 보장받지 못하고 있다. 이러한 병목현상의 해결방안으로 주목받고 있는 것이 웹 캐시이다.

본 논문에서는 웹 캐싱기법의 기본적인 기술 측면과 다양한 캐싱기법들을 분석해 보고, 보다 효율적인 캐시 시스템을 구현하기 위한 캐시 시스템을 제안한다. 동일한 네트워크 내에서 일정 수준 이상의 접속 회수를 보여주는 웹 객체를 우선적으로 선반입 하는 사용자 액세스 패턴을 이용한 캐시 시스템을 구현하여 본다. 또한 동일한 네트워크 내에서 일반적인 캐싱 알고리즘을 채용한 캐시와 실험 결과를 비교하여 평가한다.

1. 서론과 연구배경

웹을 사용하는 사용자의 급증으로 인하여 웹브라우징 시에 발생하는 네트워크 상의 트래픽 증가와 사용자 요구응답시간의 지연 등의 심각한 문제를 발생시키며, 이런 문제점을 해결하기 위해 캐싱 기법에 대한 관심이 증가되고 있다. 이러한 인터넷 트래픽의 주된 원인은 웹에서의 HTTP 트래픽 비중이 상대적으로 매우 높다는 사실에 기인한다.

인터넷 트래픽의 증가에 따라 최근 활발한 연구가 진행되고 있는 분야가 캐싱 기법을 통해 네트워크 트래픽 감소, 한정된 대역폭의 보다 효율적인 이용에 관한 부분이다.

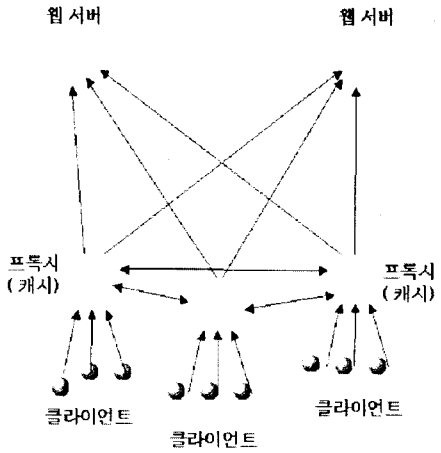
웹 상에서 캐시는 인터넷 앞에 위치하여 클라이언트가 요청한 웹 페이지를 저장하고 있다가 다른 클라이언트가 같은 페이지를 요청 할 경우 인터넷 밖의 먼 곳까지 가지 않고 대역폭이 큰 LAN에서 실제 서버 대신 사용자가 요구한 웹페이지 또는 객체들을 대신 전송해 줌으로써 클라이언트의 응답시간을 향상시키고, 대역폭을 줄여준다. 이러한 웹 캐시 시스템에서 중요한 것은 얼마나 많은 클라이언트의 요청에 대한 응답 지연을 줄여 주는가 하는 것이다.

본 논문에서는 웹 캐싱 기법에 대해 이해하고, 효율적인 사용자 접근 패턴을 이용한 캐싱기법을 제안하고 평가한다.

2. 웹캐시 시스템의 구조

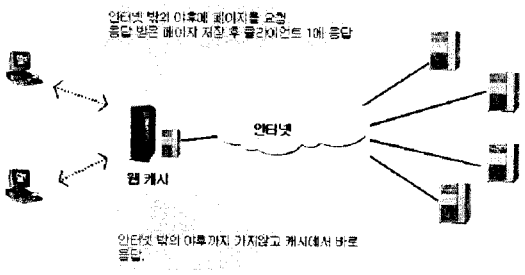
웹 캐시의 성능은 접속하는 클라이언트 커뮤니티의 크기에 의존한다. 사람들이 많이 사용할수록 문서가 이미 요청되어 캐시 안에 존재하고 있을 가능성이 더 높아진다. 캐시들은 어떤 문서의 적중확률을 높이기 위하여 서로 협동한다. 프록시란 클라이언트의 요청에 대하여 클라이언트를 대신해서 서버로 요청하고 그 결과를 다시 클라이언트에게 되돌려주는 아주 간단한 일을 하는 총체적인 것을 가리키

며 가장 포괄적이며, 방화벽이나 게이트웨이, 그리고 캐시가 프록시에 해당된다. 캐시가 방화벽이나 게이트웨이 등과의 다른 특징은 요청한 데이터를 캐시서버에 저장하고 있다가 다른 사용자가 같은 페이지를 요청하면 응답을 한다.

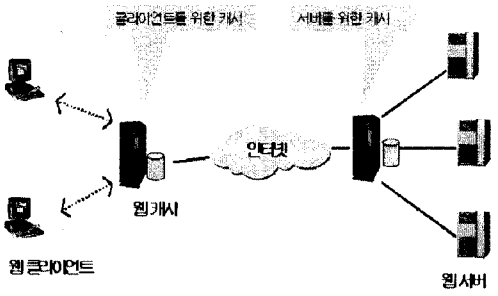


(그림1) 일반적인 WWW캐시 시스템 구조도

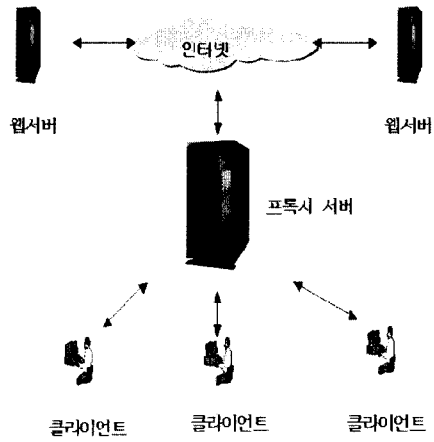
웹 캐시 시스템이란 사용자가 요구하는 웹 객체를 미리 저장하고 있다가 저장된 객체를 사용자에게 서비스 해주는 형태를 근간으로 하는 일종의 네트워크 기술이다.[1,2] 웹서비스의 성능 향상의 방법인 웹 캐시는 단순히 극소수의 사용자를 위한 시스템이 아니라 다수의 사용자를 위한 시스템이며, 재론 하자면 사용자가 많은 만큼 같은 사이트를 방문할 가능성이 높으므로 재사용 될 가능성이 높다. 웹 캐시의 가장 큰 특징은 클라이언트가 같은 웹 페이지를 요청할 경우 멀리 있는 서버까지 가지 않고 직접 응답한다는 것이다. (그림2)은 기본적인 웹 캐시 시스템을 보여주고 있다.



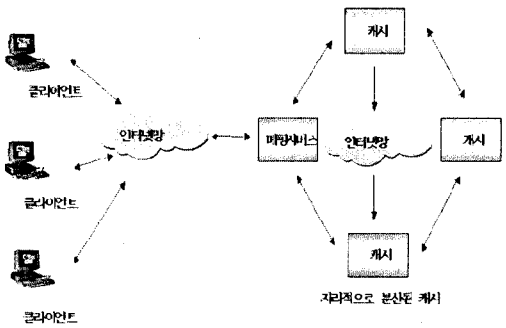
(그림2) 기본적인 웹캐시



(그림3) 가속기 형태의 캐시 구성



(그림4) 프록시 캐시



(그림5) 분산 캐시의 일반적인 구조

3. 웹 캐시 시스템의 캐싱 알고리즘 제안

웹 캐시의 기술적인 목적은 캐시 되는 위치는 네트워크의 여러 지점으로 서로 다르지만 "어떻게 한정된 인터넷 캐시 공간을 효율적으로 사용 할 것인가?" 라는 문제를 가지고 있다. 따라서 인터넷 캐시

기법의 문제는 어떻게 한정된 저장공간을 효과적으로 관리하느냐에 달려있다. 이를 위해, FIFO(First in First out). LRU(Least Recently Used), LFU(Least Frequently Used)와 같은 여러 대체 알고리즘들이 연구대상이다. 인터넷 캐시는 파일 시스템, 가상메모리 시스템과 같은 전통적인 대체 기법과는 좀 다르다. 인터넷 캐시에서는 가변 크기의 인터넷 객체를 지원해야 한다.

기존에 캐시 알고리즘은 모든 객체를 하나의 캐시에 관리하는 것이나 최근의 활발한 연구 결과로 객체들의 크기와 사용자들의 액세스 패턴에 기반 하여 저장하는 부분을 논리적으로 달리하는 새로운 캐시 기법이 생겨나고 있다. 프록시에서 문서를 캐시하여 웹 성능이 향상될지라도, 이 기술로부터 얻을 수 있는 장점은 제한되어 있다. 이전의 연구는 어떤 캐시 알고리즘에 의해 얻을 수 있는 최대 캐시 적중률은 일반적으로 50%내외에 불과하다는 것을 보였다. 즉 캐시 시스템의 사용에도 불구하고 두 개의 문서 중에 하나는 캐시에서 찾을 수 없다. 캐시 적중률을 더 높이는 한가지 방법은 미래의 문서 요청을 예상하고 국부 캐시 안에 그 문서를 미리 적재하거나 선반입하는 것이다.

프록시 캐시 효과의 핵심은 높은 적중률을 얻을 수 있는 문서 배치와 교체알고리즘이다. 최근에 적중률, 바이트 적중률, 평균지연, 총 비용과 같은 다양한 비용 척도의 최소화를 시도하는 많은 캐시 교체 알고리즘이 제안되었다.

4. 선반입 알고리즘

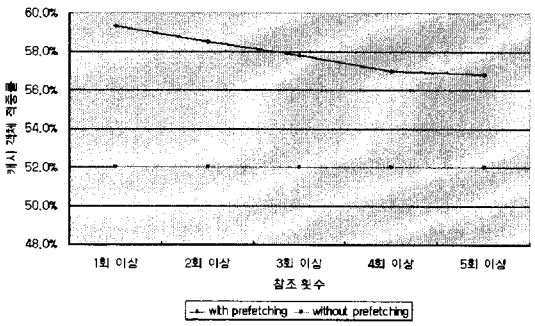
사용자의 액세스 패턴을 분석하여 미리 해당되는 웹 객체를 선반입 함으로써 원하는 웹 페이지의 성능 향상을 가져올 수 있는 시스템에서는 사용자가 A라는 HTML URL을 요청하면 액세스 패턴 데이터베이스에 있는지 여부를 검색하며, 요청한 URL 이 있으면 웹페이지 A와 연관된 웹 객체B, C를 찾는다.

웹 페이지 A에 대해서는 기존의 Squid가 객체를 처리하듯이 동작하고, 웹 객체B, C는 사용자가 요구한 것처럼 가상의 HTTP 패킷을 만들어 Adaptive Squid로 보내는 가상 클라이언트를 만든다. 제안된 사용자 접근 패턴을 이용한 캐시 서버에 적용 된 선반입 모듈의 핵심 구현 부분은 (그림 6)와 같다.

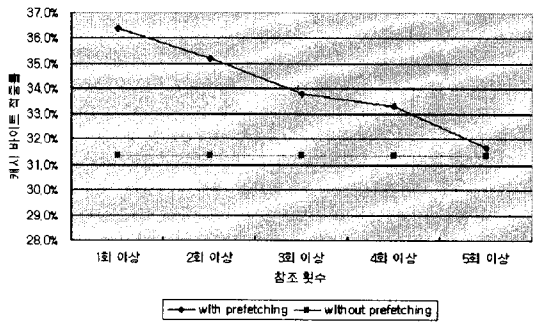
```
Void Prefetch(Request)
    ParseHttpRequest (Request, URL, Method, Headerinfo)
    ProcessClientRequest(Request) // Process HIT or MISS
   if ( Request is from real client) {
        Copy Headerinfo  for  virtual clients
        GenerateVirtualRequest (URL, Headerinfo)
    }
Void  GenerateVirtualRequest(URL, Headerinfo)
    //find if URL is in rule set
    R <- find (URL)
    if (R != NULL)     // if a rule is found for URL
      for each  URL  in the RHS of R
      {  //Compose and produce vitual request
        Write ( "GET" + URL )
        Write ( HTTP Version )
        Write ( Headerinfo)
      }
```

(그림 6) 선반입 알고리즘

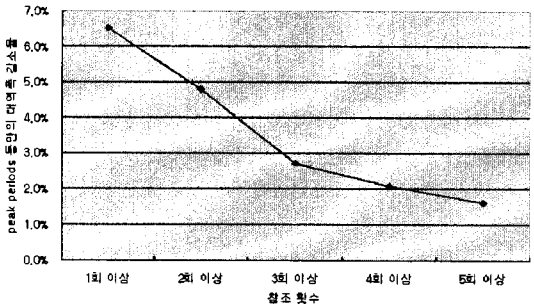
5. 성능평가



(그림7) 선반입 수행과 미수행시의 캐시객체 적중률



(그림8) 선반입 수행과 미수행시의 캐시바이트 적중률



(그림9) Peak Periods 동안의 대역폭 사용량 감소율

6. 결론 및 연구의 필요성

웹 캐시는 서버에 대한 요청을 흡수하여 전체적인 네트웍 상의 교통량을 줄이고, 사용자들에게 보다 신속한 응답시간을 제공한다. WWW의 증가에 대비한 가장 유용한 방법으로 알려진 웹 캐싱기법의 기본적인 기술 측면과 다양한 캐싱기법들을 분석해보고 웹 캐시의 저장공간에 반입될 객체의 히트율을 높이고 보다 효율적인 캐시 시스템을 구현하기 위한 캐시 시스템을 제안하였다. 대체적으로 같은 네트워크 내부의 인터넷 사용자는 특정 웹사이트를 계속적으로 재 방문 할 가능성이 크다. 제안된 캐싱 기법은 사용자들의 접속패턴 분석을 통해 요청이 예상되는 객체를 네트워크 대역폭이 여유가 있는 시점을 통하여 미리 선반입하여 요청이 폭주하는 시점이 캐시를 통해 서비스 하는 방법을 사용한다. 이는 네트워크의 휴지 시간을 통해 작동되므로 전체 네트워크의 부하를 줄일수 있고 사용자들의 웹 접근 패턴을 기반으로 하기 때문에 선반입 되어진 객체에 대해 높은 히트율을 보인다.

본 논문은 Peak periods 동안의 대역폭 사용량을 감소시키기 위한 목적으로 웹 트래픽을 대상으로 사용자의 접근 패턴을 이용하여 선반입하는 알고리즘을 제안하고, 시뮬레이션을 통해 그 유효성을 보였다. 시뮬레이션 결과는 본 논문에서 제안한 사용자 접근패턴에 의한 선반입 알고리즘으로 off-peak periods 동안의 사용되지 않는 대역폭을 사용하여 미리 예상되는 웹 객체를 읽어옴으로써 peak periods동안의 대역폭 사용량을 줄일 수 있다는 것을 보여줬다.

캐시서버에 본 논문에서 제시한 선반입 기법을 사용함으로써 낭비되는 대역폭이 거의없이 peak periods 동안의 대역폭 사용량을 줄 일 수 있었다. 따라서, 제안된 기법은 사용자 웹 액세스 지연을 최소화 시키고 사용자들이 자주 같이 액세스하는 웹 객체들을 하나의 그룹으로 하여, 그룹단위로 캐쉬에 선반입 함으로써, 캐시의 히트율을 높일 수 있었다. 궁극적으로 사용자의 웹 문서 요구에 대한 응답시간을 축소하기 위해서는 다른 요소들도 함께 개선되어야 하지만, 히트율의 증가가 이에 미치는 영향은 크다고 볼 수 있다. 웹 캐시에서 프록시 배치, 캐시 라우팅, 동적 데이터 캐시, 결함포용, 보안성 등과 같은 미해결 향상을 위한 최첨단 연구는 현재와 차세대 네트워크 문제들이 아직 있다는 것을 알 수 있었으며, 이에 대한 대처방안 중에 하나는 웹 성능에서 쉽게 확장될 수 있는 효과적이면서도 확장가능하고, 적응적이면서도 안정적인 웹 캐시 방법을 개발하고 향후 다양한 관점, 특히 실시간 네트워크 측면의 캐싱 기법에 관한 지속적인 연구가 필요하다.

[참고문헌]

[1] E.Levy -Abegnoli, A. Iyengar, J. Song , and D. Dias, "Design and Performance of Web Server Accelerator, "Proceedings of Infocom'99,1999.

[2]R.Malpani, J . Lorch, and D. Berger, "Making World Wide Web Caching Servers Cooperate, " Proceedings of the 4th International WWW Conference,Dec. 1995.

[3] V. Valloppillil and K. W. Ross, Cache Array Routing Protocol v 1.0, Internet Draft, 1997.

[4] Z. Wang, "Cachmesh : A Distributed Cache System for World Wide[3] CacheFlow Technologies, Web Caching White Paper, http://www.cacheflow.com/ , 1998.

[5] P. Cao, J. Zhang, and K. Beach, "Active Cache: Caching Dynamic Contents on the Web," Proceedings of Middleware' 98, pp. 373-388, 1998.

[6] A. Chankhunthod, P. B. Danzig, C. Neerdeaels, M. F. Schwartz, and K. J . Woorrel, "A Hierarchical Internet Object Cache," Usenix '96, Jan. 1996.

[7] Cisco, Cisco Systems Cache Engine 500 Series Q and A, http://www.cisco.com/ , 2000