효과적인 웹 캐싱 계층을 위한 설계

강만모. 유대승. 구자록
울산대학교 컴퓨터 · 정보통신공학부

Design for Effective Web Caching Hierarchy

Manmo Kang, Daesung Yoo, Jarok Koo
School of Computer Engineering · Information Technology, University of Ulsan.

요 약

기하급수적으로 증가하는 인터넷 트래픽(traffic)의 대처방안으로 캐시 일관성 유지, 캐싱 알고리즘 등 웹 캐싱에 대한 연구가 끊임없이 진행되고 있다. 본 논문에서는 인터넷 트래픽의 양을 줄이기 위해 다중의 캐시 서버를 설계하였다. 다중의 캐시 서버는 단일 캐시 서버를 사용할 때 발생하는 웹 서버의 부하를 감소시켜 네트워 트래픽의 양을 줄여준다. 또한 다중의 캐시는 라우터, 디스크, 메모리 같은 하드웨어의 비용을 절감할 수 있어 경제적인 측면에서도 효율적이다.

1. 서론

웹이 성장함에 따라, 사용 빈도가 높은 웹 서버들을 액세스하는 클라이언트들의 수는 점점 늘어나고 있으며, 클라이언트들이 웹 서버들에 접속하는데 요구되는 네트웍 대역폭(bandwidth)은 더욱 더 커지게 되었다. 클라이언트 요청을 제공하기 위하여 네트웍과 서버의 대역폭을 조절하는 것은 비용이 많이 드는 방법이다.

그 대처 방안으로 제시되고 있는 것이 캐싱이다. 캐싱은 가까운 클라이언트에서 자주 사용되는 문서의 복사본을 웹 서버로부터 효과적으로 캐시 서버에 전송(migration)할 수 있다. 웹 클라이언트들은 URL에 문서를 요청할 때 좀 더 적은 지연 시간으로 문서를 볼 수 있고, 네트웍 관리자들은 네트웍의 트래픽을 감소시킬 수 있으며, 웹 서버는 클라이언트의 과다한 요청에 따른 부하를 줄일 수 있다[7]. 네트웍의 하부구조(infrastructure)는 성장이 느린 반면, 네트웍 트래픽은 상당히 높은 지수로 성장을 계속하고 있다. 따라서 현존하는 네트웍은 증가하는 클라이언트 요청(request)에 의해 거의 포화 상태에 이르게 되었다. 미국의 온라인 같은 서비스 제공업체들은 이미 네트웍 구조가 가진 능력을 초과한 웹 사용자가 수만 명에 달한다고 설명하고 있다. 캐시 서버에 복사본(copy)을 전송하는 것은 서버 사이트가 아닌 곳에 데이터를 저장함으로써 자원(resource)의 재사용을 증가시킨다.

웹에서 캐싱은 두 가지 형태로 사용된다[6,11]

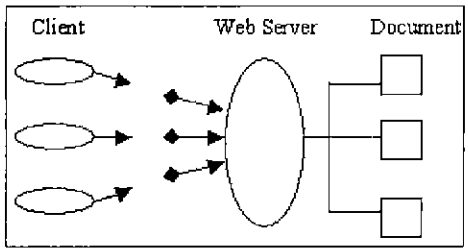
첫째는 웹 브라우저 내의 클라이언트 캐시이다 캐싱을 가진 웹 브라우저는 브라우저 윈도우에 현재 나타나는 문서뿐만 아니라 과거에 요청받은 문서도 함께 저장한다. 이러한 클라이언트 캐시들은 두 가지 유형을 가지고 있는데, 지속성(persistent)과 비지속성(non-persistent)이다. 대표적인 브라우저인 넷스케이프는 지속적인 캐시를 사용한다. 클라이언트 캐시는 웹 브라우저를 호출할 때 그 문서의 내용을 가지고 있다. 반면, 모자익 같은 비지속성 클라이언트는 사용자가 브라우저를 종료했을 때, 캐싱을 위해 사용되었던 메모리나 디스크에 할당했던 문서를 해제시킨다. 두 번째 캐싱 형태는 웹에 의해 사용되는 캐싱 프락시이다. 이 캐시는 다수의 클라이언트들로부터 다중의 서버들의 경로 상에 있는 기계에 위치하게 된다. CERN 프락시 서버, UNIX Nensa Archive 등이 대표적인 캐싱 프락시이다. 일반적으로, 캐싱 프락시는 다수의 클라이언트들에 의해 요청된 URL를 캐시한다. 캐싱 프락시는 계층적으로 사용하는 것이 가능하며, 이렇게 사용함으로써, 캐싱 프락시는 또 다른 캐싱 프락시로부터 URL를 캐시할 수 있다. 이런 경우, 캐시들은 First-Level Cache과 Upper-Level Cache로 나눌 수 있다[3,5,10].

본 논문에서는 캐시를 First-Level Cache와 Upper-Level Cache로 나누어서 설명한다. First-level에서는 단일 웹 캐시(Web Cache)를 사용하는 경우에 발생하는 문제점에 대해 알아보고 해결방안으로 다중(multiple)의 웹 캐시를 설계함으로써 그 문제점을 해결한다. Upper-Level Cache에서는 한정된 네트웍 대역폭(bandwidth)을 효과적으로 사용할 수 있는 방안을 제시한다[5]. 2장에서는 캐시 일관성에 대해 알아보고 3장에서는 First-Level과 Upper-Level에서의 다중 캐시 서버 모델을 설계한다. 끝으로 4장에는 결론 및 향후과제에 대해 언급한다.

2. 관련 연구

현재 HTTP 프로토콜은 캐시 일관성(consistency)을 유지하기 위해 두 가지의 메커니즘을 제공한다. URL 문서는 time-to-live 필드를 가지고 있다. 이것은 문서가 얼마나 오래 동안 변경되지 않은 유효한 기간을 의미한다. 즉, Time-to-live는 캐시된 복사본이 변경되었는지 체크할 때 사용한다[1,2]. 캐시 일관성을 위해 사용하는 것으로 Adaptive

TTL, Polling every time, Invalidation 등 3가지 접근 방법이 있다. Adaptive TTL 접근 방법은 문서에 적절한 time-to-live를 할당하는 방법으로 이 값이 너무 작다면 서버는 클라이언트가 보내는 if-modified-since 메시지로 인해 부하가 많이 걸릴 수 있다. 반대로 그 값이 너무 크다면 클라이언트는 효력을 상실한 문서를 가지게 될 수 있다.



[그림 1] 클라이언트의 문서 요청 모델링

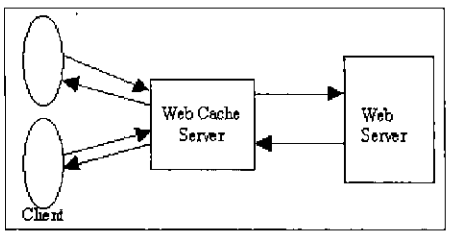
Adaptive TTL은 적절한 time-to-live 값을 할당함으로써 이러한 단점을 극복한다. polling-every time은 캐시에서 문서를 찾았을 때(hit) 매번 if-modified-since 메시지를 보낸다. Invalidation 접근 방법은 위의 두 가지 경우와 달리 서버가 문서의 변경여부를 클라이언트들에게 알려주는 방법이다. 하지만 현재의 HTTP 프로토콜은 Invalidation 메시지를 포함하지 않는다[1,2,4].

3. Cache hierarchy

3.1 First-level 웹 캐시

일반적으로 모든 클라이언트는 first-level 웹 캐시를 사용하도록 환경을 설정하는 것은 바람직한데, 넷스케이프를 사용하는 경우 Automatic Proxy Configuration으로 설정하는 것이 여기에 해당한다. First-level 캐시는 가능한 한 외부 인터넷 연결 지점 가까운 위치나 사용자(client)와 가까운 곳에 위치하는 것이 바람직하다.

3.1.1 단일 웹 캐시 서버를 사용할 경우

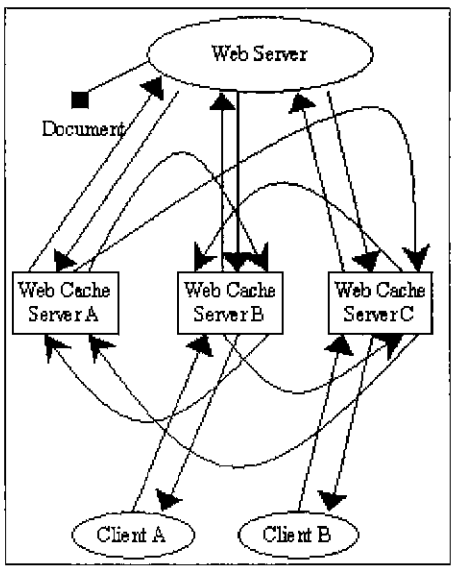


[그림 2] 단일 웹 캐시 사용

사용자는 웹 서버에 접속하기 하기 전에 웹 캐시 서버에서 원하는 문서를 먼저 찾는다 만약에 캐시 서버에 문서가 없을 경우 브라우저는 웹 서버에서 직접 문서를 가져오게 된다. 웹 캐시에 문서가 있다면 캐시 서버는 투명하게(transparently) 문서를 사용자에게 전달하게 된다. [그림2]는 사용자가 단일 웹 캐시 서버를 사용하는 것을 나타내고 있다. 여기에는 몇 가지 결점이 있다. 첫째는 캐시 서버가 기간이 지난 문서를 가지고 있는 경우다. 이러한 문서를 stale copy라고 한다. 이런 결점을 보완시키기 위해서는 Adaptive TTL을 사용하여 문서의 time-to-live를 적절하게 조정함으로써 캐시 일관성(consistency)을 유지하는 것이 필요하다[1,2]. 둘째는 캐시 서버에 사용자가 찾는 문서가 존재하지 않는 경우다. 브라우저는 직접 웹 서버에서 문서를 찾는다. 이런 경우, 사용자가 캐시 서버에서 문서를 읽어올 때 보다 웹 서버에는 부하가 많이 걸리며, 네트웍 트래픽도 증가하여 사용자는 원하는 문서를 가져오는데 좀 더 많은 시간이 걸린다 이러한 단점은 다중의 캐시 서버를 사용함으로써 극복할 수 있다.

3.1.2 다중의 웹 캐시 서버를 사용할 경우

단일 캐시 서버를 사용할 때 발생하는 웹 서버의 부하와 지연시간을 줄이기 위해 다중의 캐시 서버를 구현하면 [그림 3]과 같다.



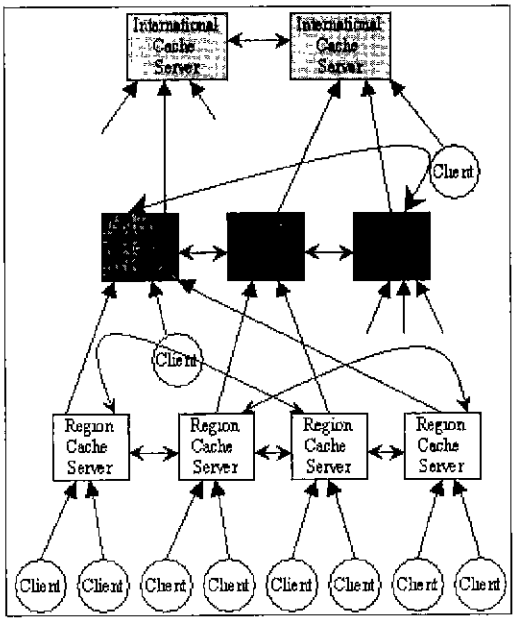
[그림 3] 다중의 웹 캐시 서버 사용

[그림 3]에서 웹 캐시 서버 A, B, C는 각각 이웃하는 캐시 서버들이다. Client A가 웹 서버에 문서를 요청하기 전에 먼저 캐시 서버 B에 찾고자 하는 문서(Document)가 있는지 체크한다. 만약 문서가 없다면, 캐시 서버 B는 Client A가 요청한 문서가 캐시 서버 A나 캐시 서버 C에 있는지 의뢰한다. 문서가 존재하지 않으면 캐시 서버 B는 Client A에게 직접 웹 서버에서 문서를 찾도록 한다. 그렇지 않고 Client A가 요청한 문서가 캐시 서버 A나 C에 존재한다면

캐시 서버 A나 C는 캐시 서버 B에 그 문서를 보낸다. 그리고 캐시 서버 B는 Client A에게 캐시 서버 A나 C에서 받은 문서를 Client A에게 보낸다.

3.2 Upper-level 웹 캐시

Upper-level 캐시 서버는 네트웍들이 서로 합쳐지는 위치에 놓여지는 것이 일반적이다. 즉, 인터넷의 교환지점에 위치하는 것이 효과적이다 이 지점은 트래픽이 많이 발생하기 때문이다. 다수의 웹 캐시는 이웃하는 캐시 서버와 서로 협력하여 인터넷 트래픽을 줄일 수 있다. 토폴러지(topology)는 upper-level 계층구조를 만들 때 고려되어야 하며, 만약 그렇지 않다면, 웹 캐싱이 없는 것보다 인터넷의 트래픽을 초래할 수도 있다. upper-level 캐시는 대륙간(inter-continental), 국제적인(international) 연결(link)로 이루어지므로 비용이 문제가 된다. 이런 종류의 연결은 비싼 비용이 들어가며, 국제적인 대역폭은 한정이 되어 있다. 토폴러지 지식(knowledge)은 서버들이 계획을 세우는데 필요하다. ISP(Internet Service Provider)는 어떤 연결점들이 업그레이드 되었는지, 어떤 네트워의 통로가 견고한지 알아야 한다.



[그림 4] Upper-Level Cache의 계층구조

[그림 4]는 이러한 네트워의 비용을 줄이고 HTTP 서버의 부하를 줄일 수 방법을 제시한다. 같은 계층의 캐시 서버는 상호 유기적으로 결합되어 있고, National Cache의 경우 클라이언트나 Region Cache에서 요청된 문서가 없을 경우 이웃하는 National Cache에 요청 받은 문서를 의뢰하게 된다. 같은 계층의 캐시 서버에 요청 받은 문서가 존재하면 그 문서를 요청한 클라이언트에게 되돌려준다.

4. 결론 및 향후과제

본 논문에서는 단일(single) 캐시를 사용할 때 발생하는 웹 서버의 부하를 줄이고 네트웍 트래픽을 감소시키기 위한 방안으로 캐시 서버를 first-level(Region Cache)과 upper-level(National Cache, International Cache)로 나누어 다중의 캐시 서버를 설계하였다. First-level의 경우, 다중의 캐시 서버는 같은 계층의 캐시 서버와 상호 연결하여 클라이언트가 요청한 문서에 응답(Response)함으로써 단일 캐시 서버를 사용할 때 발생하는 웹 서버의 부하를 감소시킬 수 있다. 또한, 각 계층의 캐시 서버는 단일 캐시 서버를 사용할 때와 같이 Adaptive TTL를 사용하여 적절한 time-to-live를 사용하여 캐시 일관성을 유지하는 것이 무엇보다 중요하다. Upper-level 경우, 네트웍이 연결되는 지점에 설치하는 것이 효과적이다. 동일한 계층의 캐시 서버는 .네트웍의 대역폭이 제한되어 있으므로 토폴러지(topology)를 고려하여 설계함으로써 네트웍의 트래픽을 줄이고 라우터, 메모리, 디스크 등 하드웨어에 드는 비용을 줄일 수 있다.

앞으로 연구해야 할 과제는 각 캐시 서버가 효력을 상실한 문서를 가능한 한 줄여서 캐시 일관성을 유지할 수 있도록 설계하여야 하며, 각 레벨의 캐시 서버는 이기종간에도 상호 연결을 보장할 수 있도록 하여야 한다

[참고문헌]

[1] Maintaining Strong Cache Consistency in the World Wide Web. Pei Cao and Chengjie Liu..

[2] World-Wide-Web Cache Consistency James Gwertzman, Microsoft Corporation.

[3] Adaptive Web Caching: Towards a New Global Caching Architecture. Lixia Zhang, Scott Michel, Sally Floyd Van Jacobson, Khoi Nguyen, Adam Rosenstein, Lawrence Berkeley National Laboratories, UCLA Computer Science Department.

[4] Seminar Paper Survey of World Wide Web Caching, University of British Columbia.

[5] NLANR Cache Hierarchy, Duane Wessels and K Claffy, UC San Diego.

[6] Cache Proxies : Limitations and Potentials, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, Edward A. Fox.

[7] Design and Implementation of a Soft Caching Proxy, Jussi Kangasharju, Young GapKwon, Antonio Ortega.

[8] Proxy Caching that estimates page load delays, Roland P Wooster and Marc Abrams, Network Research Group, Computer Science Department, Virginia Tech.

[9] Some Recommendations on Building Proxy Caching Services, Andrey Naumenko.

[10] Caching for Large Scale Systems, Robert E. McGrath

[11] A Case for Delay - Conscious Caching of Web Documents, Department of Electrical and Computer Engineering Northwestern University.