웹 캐싱을 위한 발전된 적응적 리스 알고리즘

서 진 호[0], 홍 영 식
동국대학교 컴퓨터공학과
{seojin, hongys}@dgu.ac.kr

Advanced Adaptive Leases for Web Caching

Seo jin ho[0], Young Sik Hong
Dept. of Computer Engineering, Dongguk Univ.

요 약

기하급수적인 인터넷 사용자의 증가는 네트워크 인프라의 발전속도를 능가하고 있다. 홈페이지를 관리하는 관리자가 네트워크의 대역폭을 늘린다거나 고성능의 서버를 도입하는 등 성능을 향상시키려는 노력을 하지 않는다면 사용자들이 웹에 접속하여 서비스를 받는데 걸리는 시간은 점차 늘어나게 된다. 최근의 사용자들은 정보를 얻기 위해 기다리기보다는 다른 홈페이지로 이동하므로 기업의 홈페이지도 보기 좋고 화려한 것에서 빠르게 접속할 수 있고 간결한 것으로 바뀌고 있다. 또한 더 적은 서버 자원과 네트워크 자원을 사용하는 알고리즘을 도입하여 사용자들을 만족시키려는 노력을 계속하고 있다. 본 논문은 강한 일관성을 유지하면서 서버의 자원을 적게 사용할 수 있는 발전된 적응적 리스 알고리즘을 제안하고 실험을 통하여 성능을 측정한다.

1. 서론

인터넷 사용자들이 늘어남에 따라 서버에 접속하기가 점차 어려워 졌으며 접속이 되었다 하더라도 느린 속도와 잦은 끊김은 사용자들의 요구를 막는 장애가 되었다. 이를 해결하기 위해 많은 연구가 이루어졌는데 그 중 하나가 프록시(Proxy) 서버에서 캐쉬를 활용하는 것이었다. 캐쉬를 활용하는 방식에는 TTL(Time To Live)방식, 폴링(Polling)방식, 무효화(Invalidation)방식[1]이 있다.

TTL방식에서는 유효기간이 끝나기 전에 객체가 변경되더라도 프록시가 캐쉬 속의 객체를 서비스하여 잘못된 정보를 줄 수 있으므로 약한 일관성을 제공한다. 그런데 최근의 웹 상황은 빨리 변화하고 있고 주식정보, 온라인 경매 등 잘못된 정보를 제공하면 사용자에게 손해를 줄 수 있게 되었다. 때문에 사용자에게 항상 최신의 정보를 제공할 수 있는 방식이 필요하게 되었다. 폴링방식에서는 사용자가 객체를 요청하면, 프록시는 서버에게 객체가 변경되었는지 문의하여 항상 최신의 정보를 제공한다 폴링방식은 이와 같이 강한 일관성을 제공하지만 많은 메시지가 필요하여 네트워크의 자원을 많이 사용하는 단점이 있다. 무효화 방식은 강한 일관성을 제공하면서 적은 메시지 수를 유지하는 장점이 있는 반면 프록시의 리스트를 서버에서 유지하여 서버의 자원을 사용하게 되는 문제가 있다. 리스트는 어느 프록시가 객체를 참조했는지를 알기 위해 IP번호를 기록해 두는 것이다. 이러한 무효화 방식을 개선하여 서버의 상황에 맞게 리스를 두어 리스트의 크기와 메시지를 조절하는 리스방식[1]과 적응적 리스방식[3]이 있다. 리스[4,5]란 객체의 유효기간을 정한 것으로 프록시가 리스기간이 지나지 않은 객체에 대한 서비스요청을 서버에 요청하지 않고 프록시 내의 캐쉬에서 객체를 서비스하는 기간을 말한다. 또한 서버는 각 객체를 리스기간 내의 리스트만 유지하므로 적은 수의 리스트만을 유지해도 된다.

본 논문은 적응적 리스방식에서 객체의 유효기간을 객체의 변경시간의 평균으로 결정하고 각 프록시가 객체를 요구할 때 유효기간에서 현재시간을 빼서 리스기간으로 결정해서 보내준다.

2. 기존연구

TTL은 잘못된 정보가 서비스 될 수 있고, 폴링방식은 메시지의 수가 너무 많아 현재의 인터넷 이용 상황에 적합하지 않게 되었다. 그래서, 강한 일관성을 제공하면서 메시지가 적게 발생하는 무효화 방식이 연구되었다. 무효화 방식은 서버가 객체를 서비스해간 프록시의 리스트를 유지하다가 객체가 변경이 되면 프록시에게 메시지를 보내어 객체가 변경되었음을 알리고 리스트를 삭제하는 것이다. 그러므로 프록시는 항상 최신의 객체를 캐쉬에 유지하고 사용자에게 서비스 할 수 있는 것이다. 그러나 많은 프록시가 객체를 요구하고 객체가 변경이 되지 않게 되면 리스트의 크기가 커지게 되어 대용량의 저장장치가 필요하게 되었다. 많은 회원을 서비스를 하는 곳에서는 적용하기가 어려운 단점이 있다. 이러한 무효화 방식이 서버의 저장능력을 많이 요구하므로 객체의 나이(age)를 리스기간으로 주어 메시지의 양과 리스트의 크기를 조절하였다[1]. 무효화 방식에 리스기간을 정하는 기준을 세 가지로 연구[3]하였는데, 그 첫 번째는 웹의 객체는 자주 변경되는 것과 그렇지 않은 것으로 나뉘어 진다는 점에 착안하여 객체의 나이로 리스기간을 정하는 것이다. 두 번째는 객체와 지역적으로 편향된 접근이 일어나는 점에 착안하여 자주 서비스되는 객체에 긴 리스기간을 정하는 것이다.

세 번째는 서버에서 관리할 수 있는 리스트의 크기에 따라서 리스기간을 조절함으로써 리스트의 크기를 조절하는 방식이다.

이 방식은 객체의 변경시기에 네트워크에 과부하를 주게 된다. 또한 리스기간의 변화가 크게되면 리스트와 메시지의 변화도 크게 된다. 네트워크나 서버의 저장장치는 최대 부하를 수용할 수 있을 정도가 필요하게 되므로 많은 자원을 필요로 하게 된다.

3. 발전된 적응적 리스

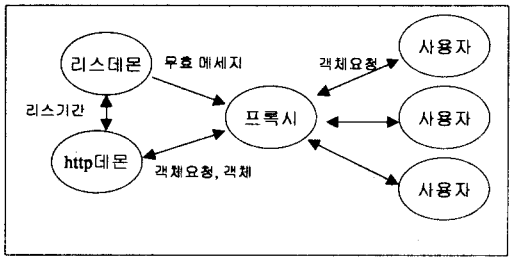

그림 1.전체 구성도

나이 기반(age based) 적응적 리스에서는 리스기간을 줄 때 각 프록시가 객체를 요청할 때마다 리스기간을 현재시간과 객체의 최종 변경시간의 차를 현재시간에 더해서 결정한다. 그러므로 각 프록시마다 동일한 객체의 리스기간이 객체를 요청한 시간만큼 다르게 된다. 객체가 변경되고 얼마 되지 않은 후에 프록시가 객체를 요청하면 짧은 리스기간이 주어지게 된다. 프록시가 다시 그 객체를 요청하면 곧바로 IMS(If Modified Since-리스기간이 지난 후 객체가 변경되었는지 물어보는)메시지를 보내야하는 오버로드가 발생하게 된다. 서버에서는 바로 전에 삭제했던 리스트를 다시 삽입해야 하는 부하를 가져오게 된다. 또한 리스기간의 길이가 한번의 객체 변경에 의해 결정되므로 객체의 변경시간간격이 변화하면 리스기간과 메시지의 양도 변화한다.

제안한 발전된 적응적 리스에서는 이전의 객체 변경추이를 반영한 평균을 구하여 유효기간을 정하게 된다. 각 프록시가 객체를 요청하면 이 유효기간에서 현재시간을 빼서 프록시에게 전달하는 것이다.

즉, 객체가 변동되었을 때 객체의 유효기간을 다음과 같이 계산한다.

유효기간 = 현재시간 + ((현재시간 - 최종변경시간) + (평균 수정시간 * 평균상수)) / (평균상수 + 1)

각 프록시가 객체를 요청하면 다음과 같이 리스기간을 계산해서 할당해준다.

리스기간= 유효기간 - 현재시간

그러면 각 프록시 캐쉬 내의 동일한 객체는 모두 같은 시간에 리스기간이 종료된다. 따라서 객체가 바뀐 직후 프록시가 객체를 요청해도 짧은 리스기간이 할당되어 금방 리스가 종료되는 상황을 막을 수 있다. 또한 리스기간이 한번의 객체 변경시간을 기준으로 한 것이 아니므로 객체 변경시간 간격의 변동이 심할 때 메시지의 수가 급변하여 병목현상을 일으키는 것을 막을 수 있다. 또한 객체 변동시간이 짧은 것에서 긴 것으로 바뀌면 리스기간이 곧 종료하므로 IMS메시지의 수가 급증하게 되는데 평균을 반영한 리스기간을 적용하면 더 긴 리스기간이 할당되므로 IMS메시지의 수가 줄어들게 된다. 객체가 변경되었을 때 적응적 리스방식에서는 리스기간이 지난 리스트가 모두 지워지지 않는다. 객체 변경 후에 프록시가 요청한 객체는 시간이 지날수록 더 긴 리스기간을 할당받게 된다. 그러므로 객체가 변경될 때 동일한 객체의 리스트중 일부가 남아 있게 된다. 그러나, 제안한 모델은 각 객체가 동일한 리스기간을 가지고 있으며 객체가 변경될 때 리스기간이 지난 리스트는 적응적 리스방식에서 남아 있던 것도 모두 삭제가 된다.

결과적으로 서버가 유지하는 리스트가 적응적 리스에서 유지하는 리스트보다 적게 되며 객체변경 시점에 네트워크에 과부하가 걸리는 것을 줄일 수 있는 것이다. 프로토콜은 다음과 같다.

처음 객체를 읽었을 때 프록시가 서버에게 객체요청(GET)메시지를 보내게 된다. 이어서 객체를 읽었을 때 리스기간이 유효한 동안은 프록시가 캐시에 있는 객체를 서비스하고 유효하지 않으면 서버에 IMS메시지를 보내 변경 여부를 문의해서 서비스한다. 리스기간의 종료 시에는 서버가 리스트에서 리스기간이 지난 프록시의 목록을 삭제하게 된다. 리스기간 중 객체가 변경되면 서버는 프록시에게 해당 객체의 무효 메시지를 보내고 리스트에서 삭제한다.

4. 성능분석

실험은 10개의 프록시를 두었고 서버에는 100개의 객체를 가지고 있는 것으로 하였다. 사용자의 객체에 대한 요청은 각 객체를 랜덤하게 하였다. 자주 바뀌는 객체와 그렇지 않은 객체의 비율은 [1]을 참조하여 실험을 하였다.

성능평가는 메시지의 양과 리스트의 크기가 최대가 되는 객체의 변경시점에서의 메시지의 양과 리스트의 크기를 비교하였다.

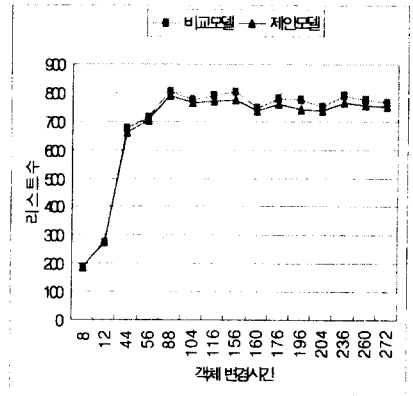

그림 2. 리스트

그림 2는 제안모델이 리스기간을 평균으로 정해서 비교

모델보다 변동폭이 좁게 나타난다. 또한 리스기간이 동시에 종료하므로 객체 변경시점에 비교모델처럼 리스트가 남아있지 않게 된다. 그림 3 의 무효 메시지 역시 같은 모습을 보여준다.

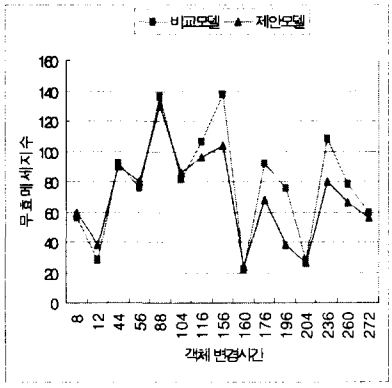

그림 3. 무효 메시지

그림 4의 비교모델에서는 객체가 변경된 후 곧 프록시가 객체를 요구하게 되면 IMS메시지를 보내게 되므로 많은 IMS메시지가 발생하게된다.

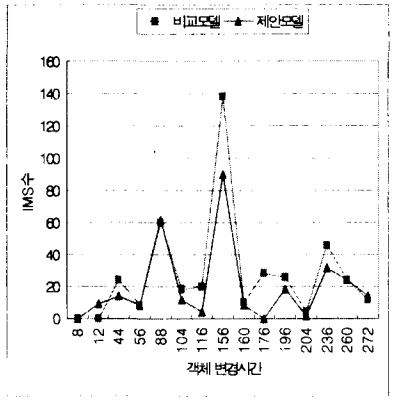

그림 4. IMS 메시지

그림 5는 각각의 최대값을 비교하고 있다.

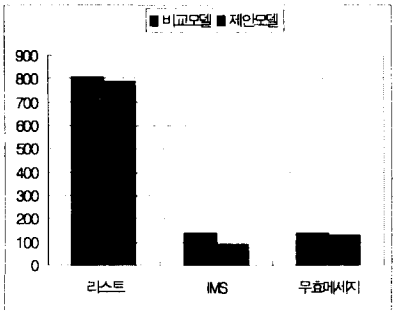

그림 5. 최대값

위의 결과에서 보듯이 리스트의 크기와 메시지의 수가 비교모델 보다 적게 나오고 변동폭도 좁게 나타나고 있다. 그림 5의 최대 리스트의 크기와 IMS, 무효 메시지의 양도 비교모델보다 적게 나오기 때문에 관리자는 더 적은 저장장치와 네트워크 자원을 가지고 서버를 구축할 수 있다. 사용자의 입장에서도 좀더 균등한 서비스를 받을 수 있게 된다.

5. 결론

네트워크의 대역폭을 늘리거나 고성능의 서버를 도입하기 어려울 경우 사용자에게 빠른 응답시간을 제공하기 위해 서버 관리자는 많은 시도를 하게된다. 사용자 쪽에서는 프록시 캐시를 두어 속도향상을 시도하게 되는데 서버 쪽에서 이를 지원해주는 적응적 리스라는 연구가 있었다. 본 논문에서는 적응적 리스에서 리스기간을 결정하는 과정을 개선하여 서버와 네트워크 자원의 부담을 더는 것을 확인할 수 있었다.

향후, 인근의 웹 캐쉬를 묶어 구성된 여러 단계의 웹 캐쉬에 향상된 적응적 리스를 적용하고자 한다.

6.참고문헌

[1] Pei Cao and Chengjie Liu "Maintaining Strong Cache Consistency in the World Wide Web", Seventeenth International Conference on Distributed Computing Systems, pages 12-21, May 1997

[2] Anawat Chankhunthod, Michael F. Schwartz "A Hierarchical Internet Object Cache", WWW CACHING WORKSHOP 1998

[3] Venkata Duvvuri, Prashant Shenoy and Renu Tewari "Adaptive Leases : A Strong Consistency Mechanism for the World Wide Web", IEEE INFOCOM' 2000, March 2000.

[4] Cary G. Gray and David R. Cheriton "Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency", the Twelfth ACM symposium on Operating System Principles, page 202-210, 1989

[5] Jian Yin, Lorenzo Alvisi, Michael Dahlin, and Calvin Lin "Volume Leases for Consistency in Large-Scale Systems", IEEE Transactions on Knowledge and Data Engineering, January 1999