웹 캐시 서버에서의 ACL 개선방안

강 선 애[0] * 김 기 창 * 심 종 익 **
* 인하대학교 전자계산공학과
sun_love@super.inha.ac.kr kchang@inha.ac.kr
** SOVIC 기술 연구소
jishim@hanseo.ac.kr

ACL Improvement in Web Cache Server

Sun-Ae Kang[0] * Ki-Chang Kim * Jong-Ik Shim **
* Dept. of Computer Science & Engineering, Inha University
** SOVIC Technical Research Institute

요 약

인터넷 사용자의 증가로 인한 서버의 과부하 및 네트워크 응답시간 지연의 대안으로 캐시 서버의 사용이 보편화되고 있다. 그러나, 최근 유해사이트의 급격한 증가로 캐시 서버의 부가 기능인 ACL(Access Control List)또한 중요한 기능의 하나가 되면서 대량의 유해사이트 차단 목록이 캐시 서버의 병목현상을 가중시키고 있다. 본 논문에서는 캐시 서버의 대명사격인 Squid의 ACL방식과 대량의 black list를 적용했을 때의 ACL의 regular expression방식의 문제점을 살펴보고, 대량의 black list에 적합한 hash기반 ACL 처리 방식을 제안한다. hash를 이용한 ACL방식은 빠른 검색을 제공하여 캐시 서버의 응답시간에 큰 영향을 주지않기 때문에 캐시 서버가 정상적인 서비스를 할 수 있도록 해준다.

1. 서 론

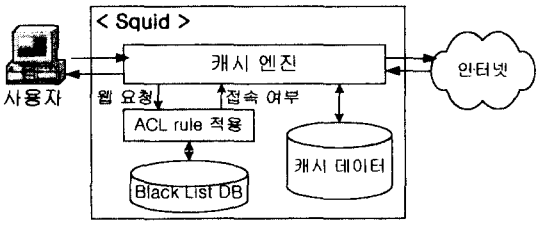
인터넷 사용자의 증가로 인하여 네트워크 응답 시간이 지연됨에 따라 캐시 서버의 필요성이 높아지고 있다. 많은 기업과 학교에서는 캐시 서버를 설치하여 인터넷 접근속도를 빠르게 할뿐만 아니라, 효율적인 교육시간과 업무시간을 위하여 캐시 서버의 부가기능인 ACL(Access Control List)을 이용한다. 원래, ACL은 허락된 사용자만이 캐시 서버를 접근 가능하도록 하는 기능이었으나, 기능이 더 확장되어 사용자의 접근시간 제어, 유해 사이트 차단 등의 기능이 추가되었다. 최근 유해 사이트의 폭발적인 증가로 유해정보차단에 대한 관심은 날로 증가하고 있다. 인터넷 유해정보차단 기술에는 차단 목록 기반의 선별 기술(Black List Filtering), 허용 목록 기반의 선별 기술(White List Filtering), 내용등급 기반의 선별 기술(Neutral Label Filtering)등이 있으나, 현재 가장 널리 사용되는 기술은 Black List Filtering이다[1]. Black List Filtering이 캐시 서버에 적용되고, 유해 사이트가 급격하게 늘어남에 따라 유해 사이트 차단 목록(이하, black list)이 증가하여 ACL이 캐시 서버의 병목현상을 가중시키고 있다[2].

본 논문의 구성은 2절에서 공개 소스인 Squid에서의 ACL 방식을 알아보고, 대량의 black list를 ACL에 적용할 때의 문제점을 살펴 보도록 하겠다. 3절에서는 hash방식을 기반으로 한 ACL을 제안하며, 실험을 통하여 hash기반 ACL을 Squid에 적용하였을 때 효율적인 유해정보차단을 제공함과 hash기반 ACL이 캐시 서버의 성능에 영향을 미치지 않는다는 것을 응답시간 측정으로 확인할 것이다.

2. 관련 연구

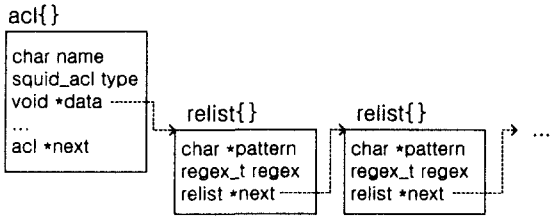
2.1 Squid의 ACL 동작방식

Squid는 오픈 소스 형태로 개발되고 있는 캐시 서버로서 상용제품에도 많은 영향을 주고 있다. 최근 캐싱에 대한 관심이 높아짐에 따라 캐시 서버 자체의 성능을 높이는 연구는 활발히 진행되고 있으나, 부가적으로 제공되는 기능으로 인한 성능저하 문제는 쉽게 간과되고 있는 실정이다. [그림 1]은 Squid의 서비스 방식을 보여주고 있다. Squid에 클라이언트의 웹 요청이 도착하면 우선 Squid에 설정되어있는 ACL rule을 이용하여 클라이언트의 요청을 분석하게 된다. Squid는 ACL rule 적용 루틴에서 적합한 사용자인지, 사용자가 요청한 URL이 black list에 포함되어있는지 검사하여 인터넷에 연결여부를 확인한 후 캐시 서비스를 제공하게 된다.



[그림 1] Squid의 서비스 방식

즉, ACL은 캐시 서버를 사용하기 위한 첫 관문이라 할 수 있다. 클라이언트의 서비스 요청 폭주와 대량의 black list는 ACL의 많은 처리량을 요구하게 되고 자연히 ACL은 캐시 서버에서 병목현상을 유발하는 주요한 원인이 된다. ACL에 black list를 적용하면 [그림 2]와 같은 acl{} 구조체가 생기고, acl{}구조체 안의 data에 실질적인 URL이 linear linked list로 저장된다. 이러한 linear linked list 구조를 갖게 된 근본적인 이유는 ACL에서 유해 사이트를 검색할 때 regular expression의 방법을 사용하기 때문이다. regular expression은 패턴인식을 하여, 주어진 단어가 포함되기만 하면 선택이 되는 방식을 말한다.[3, 4] 예를 들어, sex라는 단어가 data에 링크되어 있는 리스트중의 하나였다면, URL중에 sex라는 단어가 포함되어있을 경우 유해사이트라고 판단하고 사이트에 대한 접근이 허락되지 않는다. 즉, 유해 사이트가 아님에도 불구하고 URL에 sex라는 단어가 포함되었다는 이유만으로 접근이 되지 않는 경우가 발생할 수 있다는 것이다. 그러므로, 대부분의 유해 사이트 차단 소프트웨어에서는 특정 단어를 이용한 유해 사이트 차단보다는 black list를 이용하는 방법을 주로 사용하고 있다.



[그림 2] 기존 ACL 구조체

2.2 기존 ACL방식의 문제점

regular expression방식은 특정 확장자를 가지는 파일에 대한 접근을 막을 경우 유용하다. 그렇지만, 대량의 유해 사이트목록을 [그림 2]와 같은 구조로 사용을 한다면, 10만개의 black list를 적용했을 때, 최대 10만번의 비교를 수행하여야 한다. 즉, 많은 시간을 ACL에서 소비하게 됨으로써 사용자의 요청 처리가 늦어지게 되고, 캐시 서버의 응답시간도 현저히 늘어나게 된다. 대량 black list에 대해서는 단지 black list에 해당되는 URL만을 차단하면 되므로, 패턴 인식의 방법보다는 최대한 빠른 방법으로 해당 URL이 black list에 해당되는지 판단하는 방법이 효율적이다. 따라서, 본 논문에서는 hash방법을 이용한 ACL방식을 제안하여 캐시 서버에 부하를 주지않는 ACL을 만들고자 한다.

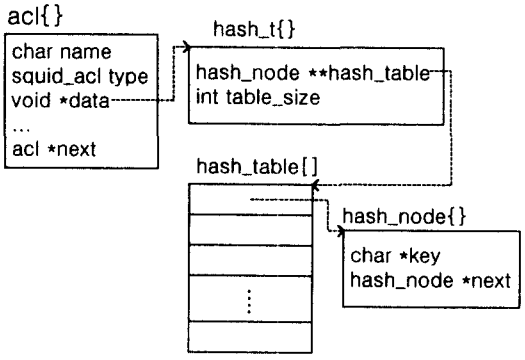
3. 대량 black list를 위한 hash기반 ACL

3.1 MD5를 이용한 key생성과 black list DB구축

hash기반의 ACL에 black list가 적용이 되기 전, 모든 black list의 URL을 MD5로 암호화하여 각 url에 대한 key를 생성한다. 이 MD5값은 클라이언트가 요청한 URL이 black list에 존재하는지 여부를 판단하고자 hash table에 접근할 때에 인덱싱 값을 구하기 위한 것이며, black list를 암호화 하여 black list DB를 구축 함으로써 유해사이트의 외부 유출을 완전히 방지할 수 있는 이점이 있다. MD5로 변경된 URL들은 파일로 저장되며, Squid가 시동될 때 파일로부터 읽어 Memory로 올린 후 ACL에 사용된다.

3.2 hash 기반 ACL을 위한 구조체 설계

제안하고자 하는 hash기반의 ACL의 구조는 [그림 3]과 같다.



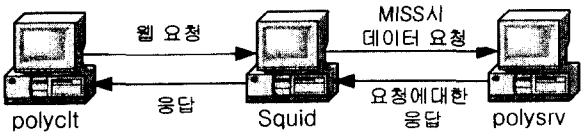
[그림 3] hash기반 ACL 구조체

hash_t{} 구조체는 hash_size와 함께 hash_table[]의 포인트 값을 가지고 있다. hash_size는 black list의 크기를 갖게 되며, Squid가 구동시에 black list를 파일로 읽어올 때 설정된다. hash_table[]은 hash_size만큼 배열로 생성된다. hash_table[] 배열은 MD5로 만들어진 key값을 이용하여 인덱싱 되는데, hash_node{} 구조체에 대한 포인터를 가지고 있다. hash_table[]에 대한 인덱스 값은 perfect hash를 이용하였는데, perfect hash는 충돌없이 한번에 원하는 값을 찾아주어 효율성을 높여준다[5, 6].

hash를 이용하여 검색하는 방법을 간단히 살펴보면, 우선 현재 요청이 들어온 클라이언트에 적용되어야 하는 acl{} 구조체를 찾는다. 그리고, 클라이언트로부터 요청된 URL을 MD5 key로 변형 후 perfect hash를 이용하여 인덱스 값을 계산하고 hash_table로 접근을 한다. 해당 위치에 링크되어 있는 hash_node를 검색하여 URL의 차단 여부를 결정한다.

3.3 성능 평가

Polygraph를 이용하여 ACL을 사용하지 않을 때와 기존 ACL을 사용할 때, hash기반의 ACL을 사용할 때의 캐시 서버의 응답 시간을 비교하여 성능을 비교하고자 한다. Polygraph는 서버와 클라이언트의 상호 작용을 실제환경과 비슷하게 시뮬레이션한 프로그램으

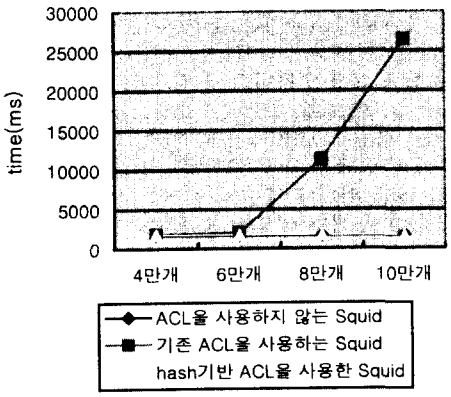


[그림 4] 실험환경 구성도

로 캐시 벤치 마킹에 사용되고 있다[7, 8]. 실험 환경은 [그림 4]와 같이 Polygraph의 클라이언트(polyclt)와 서버(polysrv) 사이에 Squid를 위치 시키고, polyclt가 Squid에 요청을 보내면, Squid는 HIT시에는 캐싱되어있는 object를 polyclt에 제공하고, MISS시에는 polysrv로부터 object를 가져와 polyclt에게 제공하게 된다. 이때, polyclt쪽에서 요청을 보내서 응답을 받을 때까지의 시간을 측정하게 된다. Squid는 2.3.STABLE3를 사용 하였고, Polygraph는 2.5.4버전을 사용하였다. 같은 실험 환경에서 ACL을 사용하지 않은 Squid의 응답시간, 기존 ACL을 사용한 Squid의 응답시간, hash기반 ACL을 사용한 Squid의 응답시간을 여러 번 실행하여 평균 응답시간을 구하였다. 이 실험에서 사용된 black list의 개수는 4만개, 6만개, 8만개, 10만개로 구분하여 실험하였으며, worst case에 대한 경우로 black list에 존재하지 않는 URL을 요청하여 응답시간을 측정하였다.

3.4 실험 결과

[그림 5]에서 볼 수 있듯이 ACL을 사용하지 않는 Squid와 hash기반 ACL을 사용하는 Squid는 응답 속도 차이가 거의 나지 않는 것을 볼 수 있다. 기존 ACL은 black list가 4만개, 6만개일 때까지는 많은 성능차이를 보지 않지만, black list가 8만개일 때부터 응답시간이 현저히 느려짐을 알 수 있다. black list가 10만개일 때, 기존 ACL을 사용하는 squid는 ACL을 사용하지 않을 때보다 15배가량의 응답속도 차이를 보이고 있는 반면에, hash기반 ACL을 사용한 Squid는 ACL을 사용하지 않는 Squid와 응답시간이 거의 차이가 없는 것을 확인할 수 있다.



[그림 5] 실험 결과 그래프

기존 ACL은 regular expression을 이용한 linear search를 하므로 worst case의 경우 black list개수만큼 search를 요하게 되고, hash기반 ACL은 perfect hash를 이용하여 한번에 원하는 key를 찾아내므로 worst case의 경우에도 O(1)의 속도로 빠르게 찾아낼 수 있기 때문이다. 그러므로, hash기반의 ACL을 사용할 경우 캐시 서버의 응답시간에 큰 영향을 주지않고 효율적으로 대량의 black list를 적용할 수 있다.

4. 결 론

인터넷 사용이 보편화 되어감에 따라 캐시 서버의 중요 성은 더욱 높아질 것이며, 효율적인 네트워크 이용을 위한 연구 또한 계속 될 것이다. 인터넷이 발전함에 따라 인터넷을 통한 정보 공유라는 긍정적인 면도 있지만, 유해 사이트를 통한 청소년 문제 또한 사회적 문제가 됨에 따라 유해사이트 차단에 대한 관심도 높아지고 있다. 날로 늘어만 가는 black list를 효율적으로 처리할 수 있는 기술로 본 논문에서는 perfect hash를 이용한 방법을 제안하였다. 네트워크 트래픽이 많은 회사나 학교에서는 본 논문에서 제안된 hash기반의 ACL을 이용하여 캐시 서버의 성능에 영향을 주지않고 Black List Filtering 을 적용하여 유해 정보를 차단할 수 있다. 또한, 기존 ACL인 regular expression을 이용하여 오디오 파일이나 동영상 파일과 같이 큰 대역폭을 요구하는 파일에 대한 접근을 제한 함으로써 대역폭 조절을 하여 트래픽이 많은 시간에 효율적인 네트워크 사용을 유도할 수 있다.

참고 문헌

[1] 유해정보 차단 기술, http://antix.icec.or.kr/mintro.htm
[2] Duane Wessels, " Web Caching," O' relly, pp. 55~56, 2001
[3] Steve Mansour, http://sitescooper.org/tao_regexps.html, 1999
[4] friedl and jeffrey E. F., " Mastering Regular Expressions, January 1997
[5] Fredman, M.L. and Szemeredi, " Storing a sparse table with O(1) worst case access time" , Journal of the ACM 31, pp.538~544, 1984
[6] Czech and Z.J., Quasi-perfect hashing, The Computer Journal 41, pp.416~421, 1998
[7] Polygraph, http://polygraph.ircache.net
[8] Banga, Gaurav and Peter Druschel, " Measuring the Capacity of a Web Server" , USENIX Symposium on Internet Technologies and Systems, December 1997
[9] Squid Web Proxy cache, http://www.squid-cache.org/