효율적인 대용량 콘텐츠 전송을 위한 HTTP 기반의 CDN

손 세일*, 나 호진**, 김 홍준***

*(주)시디네트웍스 seiilson@paran.com
**단국대학교 정보컴퓨터학부 hojin@korea.com
***진주산업대학교 컴퓨터공학부 thinkthe@jinju.ac.kr

요 약

HTTP 기반의 CDN이 성공적으로 사용자들에게 서비스를 제공하지만, 대용량 콘텐츠 전송하기에는 문제가 있다. 본 논문에서는 클라이언트나 서버의 수정 없이 기존의 HTTP CDN을 효율적인 제어를 통해 대용량 콘텐츠를 전송하는 방법에 대해 논의한다. 제안된 방법은 전송을 위해 필요한 자원의 소모를 줄이고, 기존 CDN의 신뢰성을 향상시킨다.

키워드 : HTTP CDN, 콘텐츠 전송

1. 서 론

최근 CDN(Content Distribution Networks)의 이용이 증가하는 이유는 콘텐츠 제공자가 일정한 품질의 서비스를 많은 수의 동시 사용자들에게 제공하면서, 네트워크와 서버에 대한 투자와 운영에 대한 부담을 줄일 수 있기 때문이다. CDN은 클라이언트와 인접한 에지 서버들(edge servers)을 이용하여 원본 콘텐츠를 소유한 근원 서버(origin web server)의 부하를 줄이면서, 사용자에게 빠른 응답을 제공한다[1]. 씨디네트웍스[2], Akamai[3] 등이 제공하는 일반적인 CDN은 요청 리타이렉터(redirector)와 웹 프락시 서버(web proxy)로 구성된다. 리다이렉터는 일종의 DNS 서버로서 근원 서버의 부하, 네트워크 지연 등을 고려하여, 클라이언트의 요청을 처리할 서버를 선택하여, 요청을 중계한다. 프락시는 주로 이미지, 비디오 클립과 같이 html 파일과 비교하여 용량이 크고, 정적인 객체를 캐싱하며, 근원 서버를 대신하여 사용자 요청에 응답한다.

CDN은 작은 크기를 갖는 많은 수의 웹 객체에 대한 사용자의 평균 응답 속도를 개선하기 위해 최적화 되어 있기 때문에, 영화, 소프트웨어와 같은 수백MB 이상의 대용량 콘텐츠를 서비스하기에는 비효율적이다. CDN의 중요한 구성 요소인 프락시는 일반적으로 100KB 이하의 콘텐츠를 메모리에 캐싱하며, 수백MB 이상의 콘텐츠는 캐싱하지 않는다. 따라서 대용량 콘텐츠에 대한 요청은 근원 서버에게 직접 전달되고, 처리된다. 근원 서버는 요청된 콘텐츠에 대한 서비스를 제공하기 위해 디스크를 접근하게 되며, 이것은 응답 시간을 증가시키고, CDN의 전반적인 효율적을 저하시킨다.

본 논문에서는 대용량 콘텐츠에 대한 요청이 근원 서버에게 주는 영향을 살펴보고, 이를 개선하기 위한 HTTP 기반의 CDN에 대해 논의한다.

2. 캐시

대용량 콘텐츠의 전송은 서버의 중요한 자원인 메인 메모리를 낭비시킨다. 디스크로부터 읽혀진 콘텐츠를 캐싱하기 위해, 메인 메모리에서는 수많은 페이지-아웃과 페이지-인이 일어난다. 그 동안 CPU, 네트워크 링크와 같은 다른 자원들은 유효 상태에 있게되고, 다른 요청들의 대기 시간이 증가하면서, 서버의 처리량은 점차 낮아진다. CDN에서 대용량 콘텐츠의 전송 효율을 결정하는 중요한 두 가지 요소는 프록시의 캐시 히트율과 근원 서버의 버퍼 캐시히트율이다.

프록시는 클라이언트와 서버 사이에 존재하는

하나의 연결을 클라이언트와 프록시, 프록시와 서버 사이의 두 개 연결로 나눈다. 프록시를 사용하면 다수의 동일한 콘텐츠에 대한 요청들 중 하나 이하만의 요청을 근원 서버에 전달하고 나머지 요청들은 프록시에 캐싱된 콘텐츠를 이용하기 때문에, 근원 서버의 부하를 감소시키고, 응답 시간을 개선한다. 일반적으로 프록시는 캐싱하는 콘텐츠 전부를 메모리에 배치시키기 때문에 대용량 콘텐츠를 캐싱하기 위해서는 이것에 대한 충분한 캐시 히트율이 보장되어야 한다. 대용량 콘텐츠의 캐싱은 많은 수의 다른 콘텐츠들이 캐싱될 수 있는 기회를 박탈하기 때문에 이들을 캐싱할 때 얻을 수 있는 효과를 보상할 수 있어야 한다. 따라서 대부분의 프록시는 수백 메가 바이트 이상의 대용량 콘텐츠는 캐싱하지 않고, 근원 서버로부터 전송되는 콘텐츠를 중계만 한다. 근원 서버는 클라이언트의 요청을 처리하기 위해서 먼저 버퍼 캐시를 확인하고, 캐시-미스가 발생하면, 디스크로부터 콘텐츠를 읽는 동시에 이를 버퍼 캐시에 저장한다. 버퍼 캐시는 서버의 메인 메모리 공간이며, 다른 프로세스들과 공유되며, 프록시와 달리 콘텐츠 일부분만도 캐싱할 수 있다.

3. 대용량 콘텐츠를 서비스를 위한 CDN

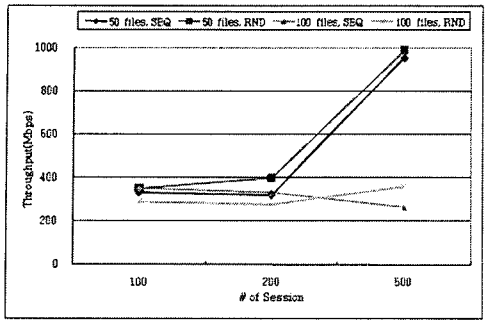
본 논문에서 대용량 콘텐츠를 효율적으로 전송하기 위한 방법은 하나의 콘텐츠를 가상적으로 작은 크기를 갖는 다수의 콘텐츠로 처리하는 것이다. 이것은 병렬 다운로드와는 다르다. 병렬 다운로드는 다운로드 시간을 줄이기 위해 콘텐츠를 다수의 서버로부터 전송받으며, 전송 단위는 콘텐츠의 분할된 조각이다[4]. 이 조각들은 클라이언트, 프록시, 서버 모두에서 하나의 콘텐츠로 인식된다.

하지만 제안된 방법은 이조각들이 모두 독립적인 콘텐츠로인식되고, 전송된다. 이 방법은 프록시가 대용량 콘텐츠를 프록시가 캐싱할 수 있게 만들고, 동일한 콘텐츠를 복제하고 있는 서버들의 버퍼 캐시 공간을 효과적으로 이용할 수 있게 한다. 이 같은 처리를 위해서는 콘텐츠와 분할된 조각들을 일관적으로 접근할수 있도록적절한 식별 방법이 필요하다. 그리고 클라이언트와 서버에서 콘텐츠를 자동적으로 분할하고, 재조합할 수 있도록 에이전트가 필요하다. 에이전트는 요청된 콘텐츠에 대한 하나의 요청을 다수의 요청으로 변환하는 작업도 수행해야 한다. 이 요청들은 HTTP를 통해 이루어져야만 프록시에 의해 해석되고, 안전하게 처리될 수 있다.

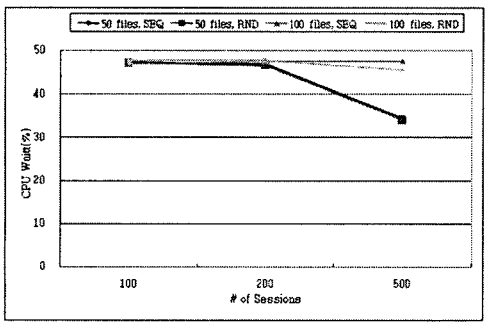
4. 실험

본 논문에서 제안된 방법을 따르는 시스템을 구현하고, 이를 평가하기 위해 다음과 같은 환경에서 실험을 진행하였다. 콘텐츠를 제공하는1대의 서버와 5개의 클라이언트를두었다. 서버와 클라이언트들은 2GB의 주기억장치를 가지며, 1Gbps 네트워크를 통해 연결되어 있다. 이들은 Linux 2.6을 운영체제로 사용하고 있으며, RAID는 이용하지 않는다. 실험은 기능 태스트와 부하 테스트 2가지를 수행하였다. 기능 테스트에서는 별도의 컴퓨터에 프록시 프로그램인Squid v2.6를 실행하고 클라이언트들의 요청과 서버의 응답이 Squid를 통해 이루어지도록 하였다. 그리고 파일을 크기를 변경해 가면서, 응답 결과와 응답 시간을 측정하였다. 본 실험의 기능 테스트 결과에 따르면 Squid v2.6은 2GB 미만의 파일들은 정상적으로 처리하지만, 2GB 이상의 파일의 전송은 하지 못했으며, 이를 처리하기 위해서는 별도의 패치가 필요하다.

부하 테스트에서 5개의 클라이언트들은 지정된 동시 세션 수를유지하기 위해 일정한 수의 200MB 크기의 파일 전송 요청을 서버에게 제기한다. 예를 들어, 서버가 처리하는 동시 세션 수를 100개라 하면, 각 클라이언트들은 항상 25개의 요청을 유지하며, 다운로드가 완료되면 새로운 요청을 제기한다. 파일은 256KB의 조각들로 분할하여, 이들을 순차적으로 다운로드(SEQ)하는 경우와 랜덤하게 다운로드(RND)하는 경우로 나누어 실험하면서, 서버의 출력 트래픽 양과 서버의 CPU 대기율을 측정하였다.



(a) 서버의 처리량(Mbps)



(b) 서버의 CPU 대기율

그림 1. 부하 테스트 결과

(그림 1)의 (a)는 서버의 출력 트래픽 양을 보여주고 있다. 이 그림에서 파일의 수가 50개인 경우는 요청의 수가 증가할수록 서버로부터 출력되는 트래픽의 양이 비례하여 증가하는것을 알 수 있다. 이것은 콘텐츠 서버의 2GB의 주기억장치 공간에서 운영체제 및 각종 프로세스들이 실행되고 남은 공간 1GB가 디스크 접근을 위한 버퍼 캐시로 이용되는데 200MB 크기의 파일이 50개인 경우는 대부분 캐시될 수있다. 이 경우 버퍼 캐시 히트율이 높기 때문에 요청의 수에 비례하여 서버의 처리량이 증가한다. 하지만 접근하는 파일의 수가 100개로 증가하면, 버퍼 캐시 히트율이 떨어지기 때문에 디스크 접근 시간이 증가하면서 서버의 처리량이 감소한다. (그림 1)의 (b)는 디스크 접근이 많아지면서 CPU 대기율이 증가하는 것을 보여준다.

(그림 1)을 보면 파일의 수가 100인 경우, 파일을 구성하는 조각들을 랜덤 접근하는 것이 순차 접근하는 것보다 처리량이 높은 것을 알 수 있다. 이것은 Linux가 버퍼 캐시를 관리하기 위해 LRU를 사용하기 때문에 먼저 캐싱 된 조각들부터 페이지-아웃되면서, 동일한 파일에 대한 이후 요청들의 버퍼 캐시 히트율이 떨어지기 때문이다. 하지만 랜덤 접근의 경우는 임의 조각부터 페이지-아웃되기 때문에 상대적으로 버퍼 캐시 히트율이 높다.

5. 결론

본 논문에서는 대용량 콘텐츠를 효율적으로 전송하기 위해 HTTP 기반의 CDN에 대해 논의하였다. 프록시를 이용하기 위해서는 대용량 콘텐츠를 분할 전송해야 하며, HTTP 프로토콜을 사용해야 된다. 제안된 방법을 성능을 평가하기 위해 실험을 수행했으며, 서비스되는 파일의 전체 크기와 버퍼 캐시 공간 간의 관계, 요청의 수와 서버의 처리량, 디스크 접근과 CPU 대기율 등에 대해 논의하였다. 실험 결과에 따르면, 서비스하는 전체 파일들의 크기만큼 서버가 버퍼 캐시 공간을 가지고 있으면, 요청의 수에 비례하여 서버의 처리량은 증가하였다. 버퍼 캐시 공간이 부족하면 디스크 접근이 많아지면서 서버의 처리량은 저하되었지만, 각 조각들을 랜덤하게 다운로드 하는 것이 순차적으로 다운로드하는 것보다 우수하였다.

앞으로 각 조각들을 병렬 다운로드 하는 경우, 클라이언트들의 다운로드 성능과 다수의 서버들이 존재할 때, 이들의 버퍼 캐시 공간을 효율적으로 이용할 수 있는 부하 분산 방법에 대한 연구들이 진행되어야 한다.

참고문헌

[1] 최승락, 양철웅, 이중식, "CDN의 핵심 구성 기술들과 경향", 정보과학회지, 20권 9호, pp. 5~11, 2002. 9.

[2] http://www.cdnetworks.co.kr

[3] http://www.akamai.com

[4] L. Cherkasova and J. Lee. "FastReplica: Efficient large file distribution within content delivery networks", In Proc. 4th USITS, Seattle, WA, Mach 2003.