다중 Virtual Hosting Server의 QoS 향상 기법에 관한 연구

A Scheme to Improve QoS in a Multi-Virtual-Hosting Server

류 상 우, 고 성 준, 이 상 문, 김 학 배, 박 진 배, 장 휘
(Sangwoo Ryou, Soungjun Ko, Sangmoon Lee, Hagbae Kim, Jinbae Park, and Whie Chang)

Abstract : Virtual hosting is a typical service to connect each directory of site and domain name. If traffic amounts may increase at one site present in the server, then it affects traffic amounts of other sites as well (including the sites which have few requests). To overcome this problem, we suggest a simple feedback-control concept for the system by periodically monitoring the traffic and properly actuating traffic dispersions by investigating the log file. Specifically, large files are to be served in a backup server (to reduce the workload of the main server) by changing their own URL's in html format. In other words, it automatically redistributes the workload by using the URL. Furthermore, we also use the redirecting method by just adding html tags to html header. This method efficiently handles the workload and maintains the capability of the server effectively to the varying workload.

Keywords : virtual hosting, feedback control, workload monitoring, traffic dispersions, QoS

I. 서론

하드웨어 서버 한 대에 한 사이트만을 운영할 경우, 용량이 얼마 되지 않은 홈페이지를 운영하기 위해서 고성능 서버를 운영하는 것은 비효율적인 방식이다. 큰 부하에 대한 일반서버의 용량제약을 해결하고 서버 자원을 효율적으로 활용하기 위해 솔루션으로 등장한 것이 호스팅 서비스(hosting service)이다. 호스팅이란 단순하게 정의하자면, 서버에 공간을 할당하는 것을 말한다. 각 호스트들은 HDD의 특정 디렉토리에 소스를 저장한다. 웹서버는 각 호스트에 해당하는 URL 요청이 들어왔을 때 각각에 매칭(matching)되는 공간으로 연결시켜 주는 것이다[1]. 하드용량에 따라 다르겠지만, 호스팅 서비스는 단일 서버에 수십개에서 수백개의 사이트를 동시에 수용할 수 있게 해주어, 자원을 효율적으로 이용하게 한다. 또한, 방식에 따라 차이가 있지만, 단지 IP 하나만 사용하여 많은 수의 사이트를 운영할 수 있는 이점도 있다[2]. 반면에, 모든 사이트가 한 서버의 CPU와 각 리소스(resource)를 동시에 공유하기 때문에, 한 사이트에 예상보다 큰 과부하가 걸릴 경우, 그 사이트가 리소스를 거의 독점하게 되고, 서버에 상주하는 나머지 사이트들은 부족한 리소스를 나눠 사용하게 되는 현상이 발생하여 속도의 저하를 야기시킨다. 이러한 문제로 인하여, 서버의 상태를 최상으로 유지하고 사용자에게 홈페이지 서비스를 동시에 할 수 있는 QoS(Quality of Service) 방법이 제기되었다.

이러한 서버 트래픽 문제를 해결하기 위해 Qguard[3], 컨텐츠 변환[4], 리디렉션 기법[5], 로드밸런싱 기법[6] 등이 있다. Qguard는 IP 패킷을 분석하고 승인 또는 거부할지 여부를 판단하여 필요없는 트래픽을 제한함으로써 QoS를 보장한다. Qguard는 트래픽을 제한하자는 관점에서 제안한 방식이다. 또한 들어오는 각 패킷에 웨이트(weight)를 주어 우선순위를 부여하는 방법을 사용함으로써, 속도를 보장받고자 하는 특정 사용자의 QoS를 보장한다. 즉 등급에 따라 차별된 서비스를 제공하는 것이다.

트래픽의 상태와 이에 따라 미리 정해진 등급의 컨텐츠를 다르게 제공하는 컨텐츠 변환 방법은 트래픽 상태가 양호할 때에는 정상적인 상태의 컨텐츠를 제공하고, 트래픽이 높아질 때마다 반비례하여 낮은 등급의, 사이즈가 작은 컨텐츠를 제공하여 QoS를 보장하게 된다. 이것은 대부분 트래픽을 일으키는 것이 사이즈가 큰 이진파일에 의한 것이라는 사실에 착안하여 트래픽을 일으키는 컨텐츠의 등급을 여러 가지로 제공하고자 함이다.

리디렉션 기법은 사용자 요구(리퀘스트)를 다른 서버로 돌려주는 기법으로 서버의 능력이 한계치에 도달했을 때, 더 이상 서버에 요청이 들어오지 못하도록 하는 것을 말한다. 대표적인 방법으로 RRDNS(Round Robin Domain Name System) 방법이 있다. 도메인 서버에 한 개의 도메인과 복수의 IP를 매핑시켜 놓고 도메인에 리퀘스트가 들어올 때마다 할당된 IP에 공평히 분배하여 한 IP에 집중하지 못하도록 하는 방법이다.

로드밸런싱 기법은 부하를 두 개 이상의 서버에 나누어 부하를 분산하는 방법이다. 서버의 앞단에 리퀘스트를 전담하는 로드밸런서를 통하여, 들어오는 리퀘스트를 리얼서버에 라운드로빈 등의 스케줄링 정책(policy)에 따라 나누어 부하를 분산케 함으로써, 여러 서버가 마치 한 대의 대형 서버처럼 동작하게 하는 방법이다.

본 연구에서는 멀티 호스트 서버와 백업 서버로 이루어진 간단한 시스템을 가지고, Qguard와 컨텐츠 변환에 대한 내용을 보완하고 리디렉션 기법과 로드밸런싱 기법을 이용하여, 서비스를 제공하는 피드백 시스템을 제안한다. 이 시스템은 트래픽을 분석하는 모니터링 부분과 구동부분(actuator)으로

접수일자 : 2001. 12. 10., 수정완료 : 2002. 1. 26.
류상우, 고성준, 이상문, 김학배, 박진배 : 연세대학교 전기전자공학과(johnryou@yonsei.ac.kr/gosun@yonsei.ac.kr/sangmoon@yonsei.ac.kr/hbkim@yonsei.ac.kr/jbpark@control.yonsei.ac.kr)
장 휘 : 씨아이사㈜(wchang@c-eisa.com)
※ 본 논문은 한국과학재단의 목적기초연구사업(R01-2001-00316)에 의해 지원되었습니다.

이루어졌는데, 로그 파일을 이용하여 시스템에 미치는 부하를 최소로 하기 위한 모니터링 방법을 제시하고, 구동부분에서는 트래픽의 정도에 따라 파일이 큰 컨텐츠의 URL을 변환하거나, 서버를 리디렉션하는 방법을 제시한다. 이러한 방법으로 트래픽이 많이 걸렸을 때에도 정상적인 서비스를 중단없이 공급할 뿐만 아니라, 과부하시에도 사용자의 요청에 대해 지연(delay)없이 응답하게 하여 QoS를 보장한다. 2장에서는 기본적인 가정과 개념에 대해 정리하고, 3장에서는 전체적인 시스템 구성과 동작방식에 대해 설명하며, 4장에서는 부하를 분산시키기 위한 알고리즘과 해킹 방지에 대해 설명한다.

II. 기본가정 및 개념

1. QoS 처리

텍스트, 그래픽, 오디오, 그리고 비디오 등의 멀티미디어 데이터를 다루는 시스템을 멀티미디어 시스템이라 하며, 이러한 시스템의 주요한 특징 중 하나는 시간 종속적(time-dependent)인 성질을 가진다는 것이다. 분산된 환경에서의 멀티미디어 시스템에 대한 관심이 증가함에 따라 QoS 개념이 대두되었다. QoS를 Open distributed processing을 위한 참조 모델에서는 하나 이상의 객체(object)들의 총체적인 행동에 대한 질적 요구사항의 집합이라 정의하였고, ISO에서는 시스템의 실제적인 행동을 모형화하는데 사용되는 것이라고 정의하였다. 일반적으로 QoS란, 응용 프로그램에 요구되어지는 기능을 지원하기 위하여 필요로 되어지는 분산 멀티미디어 시스템의 질적인 그리고 양적인 특징들의 총체라고 볼 수 있다[8].

2. 가상호스팅 서버

가상호스팅 서버는 하나의 컴퓨터에 하나의 도메인 이름을 포함하고 있는 웹 서비스를 여러 개 운영하는 컴퓨터라 할 수 있다. 현재 가상호스트로 이용되어지고 있는 방법은 두 가지인데, IP 주소를 기반으로 하는 주소기반 가상호스팅(Address-based virtual hosting) 기법과 호스트 이름을 기반으로 하는 이름기반 가상호스팅(name-based virtual hosting) 기법이 있다.

주소기반 가상호스팅 : 이것은 서버에 있는 하나 또는 그 이상의 LAN 카드에 여러 IP를 등록하고 웹서버에서 각 IP에 다른 디렉토리를 지정하는 것이다[9]. 즉, 클라이언트가 URL을 입력하여 가상 서버에 요청이 들어오면, 웹서버 설정 파일에 미리 입력된 IP와 연결된 지정 디렉토리를 찾아서 그 디렉토리의 컨텐츠를 클라이언트의 브라우저로 보내주는 것이다. 각 사이트마다 IP가 할당되어 있으므로 보다 안정적인 서비스를 할 수 있고 별도의 복잡한 DNS 서버 세팅이 필요 없어 간단하게 서버를 구현할 수 있다. IP를 독립적으로 사용함으로써 비용면에서 효율은 떨어지지만 성능면에서 효율적인 서비스 방식이다.

이름기반 가상호스팅 : 이 방식은 여러 도메인을 하나의 IP로 할당시켜 놓고 외부에서 요청이 들어왔을 때, 웹서버에서 해석하여 각 사이트를 연결시키는 방식이다. IP를 절약하면서 서버를 운영할 수 있으나, 각각의 도메인이 대표 IP를 가리키도록 DNS 서버에서 세팅을 별도로 해주어야 하는 제약이 따르게 된다. 이름기반 가상호스팅 방법은 클라이언트가 HTTP1.1 프로토콜을 지원해 주어야 하고[10], 주소기반의 가상호스트보다 간단하며, 외관상으로는 주소기반 가상호스팅 방법처럼 보인다. IP-based와 이름기반 가상호스트와의 다른 점은 가상호스트 설정에 어떠한 IP가 이름기반 호스팅 방법에 사용될 것인가를 나타내어 주는 name virtual host가 포함되어 있다는 점이다.

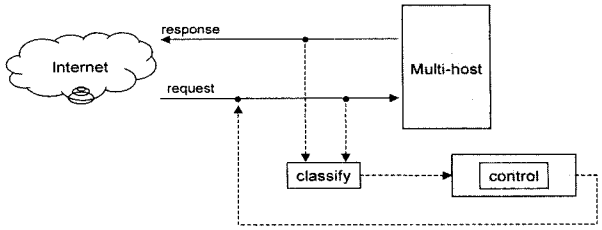


그림 1. Multi host 기반 단일 웹서버 개념.
Fig. 1. Multi-host based single web server.

웹서버의 구성 : 웹서버는 Apache로 구축하였으며, multi host를 기반으로 한 단일 웹서버로 구현하였다. 다수의 클라이언트가 네트워크를 통하여 연속적으로 웹서버에 서비스 요청을 하고 있다고 가정을 하며, 클라이언트 요청에 의한 웹 서비스는 soft deadline(hard deadline과 loose한 개념의 시간 제약성으로, 클라이언트의 서비스 요청을 정해진 시간 내에 수행하여야 하며, 그렇지 못할 경우 성능 저하 및 QoS가 감소됨)을 가지며, 클라이언트가 가지는 지연시간은 네트워크와 서버에서 소요되는 시간을 포함하게 된다.

III. 시스템 알고리즘

1. 시스템의 구조

200-300개의 사이트 중 특정 사이트에 대한 홈페이지 요청이 많아지면 서버의 부하가 증가하고, 요청 빈도가 없거나 낮은 사이트까지 서비스 속도가 떨어지게 되는 문제점이 있다. 본 시스템의 수행 목적은 우선 서버의 과부하를 방지하기 위한 것과 클라이언트의 서비스 요청에 대한 응답시간 및 성능을 보장하기 위한 것 두 가지이다. 시스템을 구현하기 위해서 서버의 부하 상태에 따라 상태를 변화시킬 수 있는 적절한 구동부분(actuator)을 설계하고 현재의 부하 상태를 신뢰성 있게 측정할 수 있는 모니터링을 구현한다.

시스템 구조 : 시스템은 논리적으로는 시스템의 상태를 감시하고 판단하는 모니터 부분과 판단된 상태에 따라 대처하는 실행기의 구조로 되어 있다[11].

모니터링 : 트래픽을 감시하는 기능을 담당한다. 패킷을 일일이 검사하여 시스템에 부하를 주는 방법 대신에, 각 사이트에 들어오는 리퀘스트에 대한 로그 파일을 통해 트래픽 상태를 감시한다. 먼저, 각 로그 파일의 크기를 감시하고 크기가 급격히 증가하면 로그 파일의 구체적인 내용을 감시한다.

구동부분(actuator) : 모니터의 상태 판단에 따르는 조치를 취한다. 과부하라고 판단이 되면, 상태에 따라 html에 포함된 큰 사이즈 파일의 URL을 백업 서버의 URL로 바꾸는 조치를 취하고, 만약, 부하가 더 증가하면 부하가 걸린 호스트 사이트의 html 헤더에 리디렉션 스크립트를 삽입하는 조치를 취한다.

2. 사이트 분산

리퀘스트가 높은 사이트의 복사본(replica)을 미리 만들어

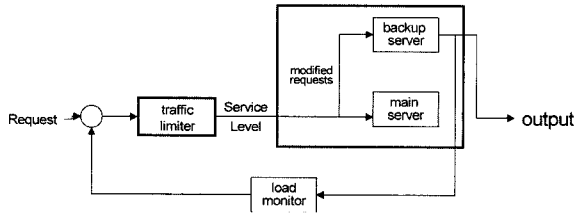


그림 2. 시스템 다이어그램.
Fig. 2. System diagram.

놓는다. 즉, standby 디렉토리를 만들어 놓고, 웹서버에서 사이트에 들어오는 트래픽을 측정하고 있다가 어느 정도 이상의 요청이 들어오면, 그 다음부터는 복제된 사이트로 요청을 분산시키는 방법이다. 주기적으로 서버의 내용을 복사하여 저장하고 있다가 사고가 발생하면 백업 서버의 내용을 주 서버로 복제한다.

주 서버의 특정 사이트에 부하가 많이 걸렸을 때, 그 사이트의 내용을 백업 서버에서 제공함으로써 주 서버의 부하를 분산시킨다. 부하가 걸렸을 때도 부하를 모니터링하고 일정 리퀘스트 아래로 떨어지면 본래대로 주 서버가 전담하게 되며, 백업 서버는 다시 standby 상태로 돌아가게 된다

이 기법은 기존의 리눅스 가상 서버에서 사용하는 가상 IP와는 다른 방법이다. 가상 IP는 부하가 걸린 서버를 완전히 이전하고 주 서버를 사용하지 않는데 반해, 제안된 방법은 주 서버에서 부하가 걸린 디렉토리만 백업서버에서 기능을 담당하는 것으로 더 효율적이라 할 수 있다.

3. 부하 측정

모니터링은 피드백 컨트롤러 개념에서 가장 중요한 부분이다. 모니터링 방법은 여러 가지가 있을 수 있다. 하지만 본 논문에 적합한 것은 모니터링에 큰 부하가 걸리지 않으면서 정확한 부하를 측정할 수 있는 명세사항(specification)이 필요하다. 이런 조건을 만족시키기 위해 제안하는 모니터링 방법은 서버의 로그 파일을 분석하여 서버의 상태를 측정하는 것이다[7].

로그(log) 파일 : 서버 내에는 반드시 로그 파일이 존재하며, 그 서버에 접속한 클라이언트의 경로(IP)와 접속 시간, 접속한 파일, 디렉토리 등이 기록되어진다. 따라서, 로그 파일을 분석하면, 서버의 사용 상태(status)를 알아낼 수 있다. 로그 파일은 두 가지가 존재한다. 서버 전체의 리퀘스트 입출력 상태를 기록하는 로그 파일과 각 호스트 사이트를 관리하는 관리자를 위해서 각 사이트의 입출력 상태만을 기록한 로그 파일 두 가지이다. 본 논문에서는 전체 로그 파일을 메인 로그, 각 호스트를 위한 로그 파일을 로컬 로그라고 구분한다.

로그 파일 분석 방법 : 첫 번째 단계로, 각 사이트의 로컬 로그 파일 사이즈를 감시하여 파일의 크기가 급격히 커지는 파일을 찾아낸다. 로그 파일 사이즈가 커지는 것은 로그 파일에 기록되는 내용이 많아지므로 그만큼 리퀘스트가 높아진다는 것을 의미한다. 따라서 찾아진 로컬 로그 파일이 존재하는 호스트 사이트는 리퀘스트가 높은 호스트로 규정한다.

다음 단계로는, 이렇게 찾아진 리퀘스트가 높은 호스팅 사이트에 있는 로그 파일의 IP 문자열을 주기적으로 점검한다. 물론 두 가지 방법을 사용할 수가 있는데, 메인 로그파일

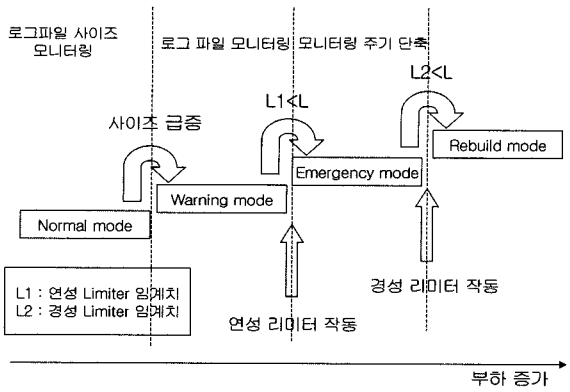


그림 3. 부하에 따른 모드 변환.
Fig. 3. Mode variation due to the traffic workload.

에서 접속하는 디렉토리를 감시하여 카운트하는 방법과 로컬 로그 파일을 개별적으로 감시하는 방법이 있다. 전자의 방법은 로그 파일 하나만을 감시하게 되므로 간편하다는 이점이 있지만, 모든 사이트들을 총망라한 메인 로그 파일은 크기가 크므로 메모리에 로딩하여 시스템에 부하를 줄 수 있다는 단점이 있다. 이에 반해, 후자는 각 디렉토리에 산재하여 있는 로컬 로그 파일들을 모두 감시하여야 하는 단점은 있으나, 필요한 로컬 로그 파일만 감시하므로 부하가 적다는 장점이 있다.

시간에 대해 미분을 하여 주기 사이에 값이 급격히 증가하면, 일단 사이트를 "주의 모드(warning mode)"로 세팅한다. 주의 모드란, 모니터링 이후에 주기간격을 줄이며, 감시해야 할 예비 후보(candidate)라는 것을 의미한다. 즉, 로그 파일의 사이즈를 감시하는 모니터링 단계에서 로그 파일의 내용과 관련있는 리퀘스트 비율을 조사하겠다는 것을 뜻한다. 어떠한 식으로 모드를 변환할 것인지에 대해서는 그림 3에 구체적으로 기술되어있다.

대역폭(bandwidth) 모니터링 : 파일 크기가 Mbyte에 해당하는 mpg 파일이나, 멀티미디어 파일같이 큰 파일은 접속 횟수가 적더라도 CPU 점유율이 높으므로 이에 대한 대안이 필요하다. 이를 위하여, 각 사이트에 존재하는 파일 중 크기가 큰 파일에 대해 감시를 별도로 한다. 각 파일에 대한 로그를 따로 만들어 로그에 접속한 시간을 기록하고 일정 이상 접속 기록이 검출되면, 그 파일에 대한 URL을 바꾸어 서비스 할 수 있도록 조치를 취한다. 하지만, 개별 파일을 모두 감시하는 방법보다 그런 파일들을 포함하는 html 문서 파일의 로그 파일 크기를 점검하여 간접적으로 감시한다. 그림 3에서 단계를 결정하는 L은 개별 사이트의 로그 파일 사이즈를 말하며 L1 및 L2는 연성과 경성 리미터 임계치를 말한다. 효율적인 로그 파일 분석을 위해 index.html, index.php, default.html 등 웹서버 설정 파일에 디폴트로 설정되어 있는 파일을 감시하여, 로그 파일 전체행을 감시해야 하는 불필요한 작업량을 줄인다.

IV. 부하 분산 알고리즘

1. 상태 판단(State decision)

정상 상태 여부의 판단을 위해, 평소 홈페이지나 특정 파

일의 특성들에 대한 데이터 중에서 히트율을 이용한다. 그 데이터의 평균(average)과 분산(dispersion)을 기록해놓고, 현재 들어오고 있는 히트율의 평균과 분산을 계산한 값과 비교를 통해, 현재 데이터를 유효범위 내에서 기각여부를 판단한다.

2. 트래픽 제어 방법(Traffic control method)

트래픽 제어는 메인 서버에 폭주하는 리퀘스트 중 일부를 백업서버 쪽으로 돌려서 트래픽을 분산하는 방법을 적용한다. 본 논문의 트래픽 컨트롤 방식을 리미터(limiter)라 정의하고, 어느 정도 이상의 트래픽이 감지되면 트래픽을 제한하여 메인 서버의 성능을 보장한다. 트래픽을 제한하는 리미터는 방식에 따라 경성 리미터와 연성 리미터로 나눈다. 기준이 되는 L1 및 L2는 평소 트래픽을 기준으로 평균 트래픽의 일정 비율을 초과했을 때로 정한다. 이를 위해 일단 특정 값을 정하고 이 데이터를 실제로 적용하면서 계속 보정해간다.

연성 리미터(soft limiter) : 연성 리미터는 부하량을 측정하여 트래픽이 발생했을 때 부하량이 많이 걸리는 사이트의 컨텐츠중 용량이 크고 CPU 점유율이 높은 파일을 메인 서버 대신 백업 서버에서 제공하는 방식이다. 즉 html같은 텍스트 위주의 파일은 메인 서버에서 공급하고, 그림 파일 같은 바이너리 파일은 다른 서버에서 제공하는 것이다.

경성 리미터(hard limiter) : 경성 리미터는 연성 리미터의 한계를 초과했을 때 서버에서 리디렉팅을 하는 방법이다. 리디렉션은 클라이언트에 의해 서버의 주소를 요청 받았을 때, 서버가 필요에 따라 리퀘스트의 방향을 바꿔주는 것을 말한다. 부하가 많이 걸렸을 때, 서버가 작동하지 않을 때, URL 주소가 바뀌었을 때 사용하게 된다. 본 논문에서는 html의 태그를 이용하여 리디렉션을 하는 방식을 제안한다. 이 방법은 기존 html 문서의 헤더에 스크립트를 삽입하기만 하면 된다. 연성 리미터의 기준치를 초과하고 부하가 많이 걸려 정상적인 서비스가 어렵다고 판단이 되면, 스크립트를 실행시켜 디렉토리 안의 html 헤더에 다음과 같은 코드를 삽입한다.

이것은 클라이언트가 호스팅 사이트에 접속을 하여, 해당 html 파일을 클라이언트의 브라우저에 로딩하게 되면, 즉시 위의 Refresh 스크립트가 작동하면서 백업 서버에 있는 해당

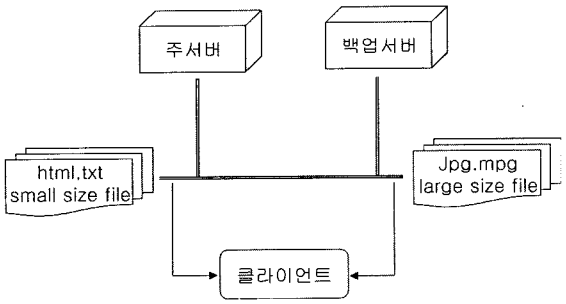


그림 4. 연성 리미터 구성.
Fig. 4. Soft limiter structure

```
<META http-equiv="Refresh"
content="0; URL=http://백업서버/index.html">
```

그림 5. 리다이렉션 html 태그.
Fig. 5. Redirection html tag.

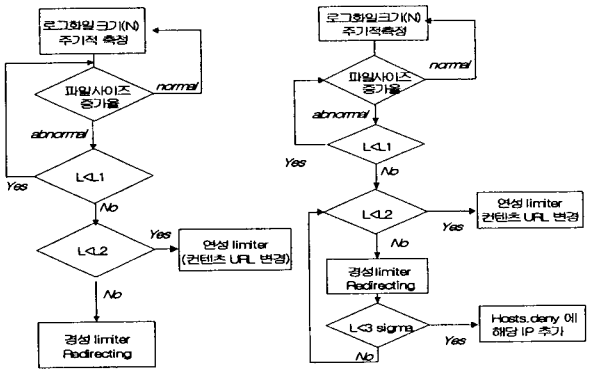


그림 6. 부하분산과 DoS 해킹방지를 위한 확장 알고리즘.
Fig. 6. Traffic dispersion and extended algorithm to defend DoS attacks.

html 파일이 클라이언트의 컴퓨터에 서비스되게 된다. 따라서, 경성 리미터가 작동한 이후부터는 메인 서버에 존재하는 디렉토리에서 제공되지 않고 백업서버에서 파일들을 제공하게 되므로 메인 서버의 트래픽이 감소하게 된다.

3. 해킹 방지를 통한 QoS 보장

로그 파일을 분석하여 들어오는 IP를 분석했을 때, 만약 특정 IP에서 시간당 집중적으로 들어오는 패턴이 발생하거나 과도한 접속이 이루어진다면, 그 IP를 차단하여 DoS(Denial of Service) 공격을 미연에 방지할 수 있다. 해킹에 의한 서비스 중단을 막고 정상적인 서비스를 제공하는 방법이며, QoS 측면에서 효율적이라고 할 수 있다. 연성 리미터는 1시그마를 기준으로 경성 리미터는 2시그마를 기준으로 상황을 판단한다. 이것은 각각 정확도가 유효 범위 5%와 1%에 해당하는 수치이다.

경성 리미터가 활성화되면 해킹 여부에 관해 감시를 시작한다. 첫째, 히트율이 평소 데이터의 3시그마를 넘는다면 이것은 확률로 1.35×10^{-3}에 해당하고, 예외로 판단할 수 있으므로 해킹으로 결정한다. 이 경우는 서버 전체의 서비스 유지를 위해 웹서버 설정 파일(httpd.conf)에서 해당하는 사이트의 설정 부분을 제외시킨다. 또, 특정 IP로부터 연속적으로 들어오는 리퀘스트는 해킹으로 판단하고, 이 특정 IP를 거부 목록에 추가해서 그 IP가 서버에 접근하는 것 자체를 막는다. 웹서버 설정 파일 /etc/hosts.deny는 서버의 접근 가부를 정하는 파일로 이 파일에 등재되어 있는 IP는 접근을 막을 수 있다. 그림 6은 기본 알고리즘과 DoS 해킹 방지를 위해 확장된 알고리즘의 순서도이다.

V. 결론

Qguard는 트래픽이 발생하면 들어오는 IP 패킷을 분석하여, 낮은 등급의 IP를 차단함으로써 서버의 상태(status)를 보장하는 기법이다. 하지만, 본 논문의 리미터는 IP를 차단하는 것이 아니라 다른 서버로 전환시킴으로써, 서버의 트래픽을 조절하게 되므로 클라이언트가 거부당하는 일은 발생하지 않는다. 시스템상에 문제가 발생할 경우, 시스템 전체를 리부팅해야 하는 Qgaurd와 달리 단순히 프로그램만 리셋해주면 된다.

컨텐츠 변환은 단일 서버로 트래픽을 분산할 수 있는 장점이 있다. 하지만, 트래픽을 보장하기 위하여 컨텐츠의 질

을 낮추어 공급해야 하지만, 본 논문에서는 두 대의 서버를 사용하여, 구조는 다소 복잡해졌지만 트래픽 분산을 시키되 정상적인 컨텐츠를 제공하는 것이 가능하다.

또한, 본 논문에서는 멀티호스트 서버의 부하분산을 위하여 백업서버를 이용한 컨텐츠 전달에 대한 방안을 제안하였다. 즉, 백업서버는 일반적으로 서버의 고장 발생시 고가용성을 위하여 운영되는 것에서 더 나아가, 정상적인 동작을 할 때에도 전체 서버 및 각 사이트의 과부하 발생에 따라 부분적으로 서버의 역할을 하게 함으로써, 부하 분산의 효과를 기대할 수 있다.

트래픽 모니터링을 위하여 각 사이트의 로그파일을 분석하는데, 이것은 간단한 스크립트로 작성할 수 있기 때문에, 서버의 추가적인 부하발생을 최소화 할 수 있다. 이를 바탕으로, 부하 발생정도에 따라 그 상태를 연성과 경성으로 구분하고, 각각을 URL redirection과 IP mapping에 의해 특정 파일 및 사이트 전체를 사용자에게 제공하는 두 가지 방법의 부하분산 방안을 제시하여 QoS를 보장한다.

참고문헌

[1] Ludmila Cherkasova, "FLEX : Load balancing and management strategy for scalable web hosting service," *Computers and Communications, 2000. Proceedings. ISCC 2000. Fifth IEEE Symposium*, pp. 8-13, 2000

[2] *http://www.apache.kr.net/documents/vhost-story.html.*

[3] J Hani, R John, Kang G. Shin, "QGuard : protecting internet service from overload," *Realtime Computing Laboratory*, University of Michigan, 2000.

[4] Tarek F. Abdelzher, "Web server QoS management by adaptive content delivery," *Quality of Service*, IWQoS '99. Seventh International Workshop, pp. 216-225, 1999.

[5] Colajanni M, Yu P. S, and Dias D. M, "Scheduling algorithms for distributed Web servers," *Proceedings of the 17th International Conference*, pp. 169-176, 1999.

[6] *Linux Virtual Server Project http://www.linux-vs.org.*

[7] R. Cooley, "Grouping web page references into transactions for mining world wide web browsing patterns," *IEEE Knowledge and Data Engineering Exchange Workshop. Proceedings*, pp. 2-9, 1997.

[8] *http:// www.qosforum.com.*

[9] *http://www.apache.kr.net/documents/name-virtual.html.*

[10] *http://www.apache.kr.net/documents/ip-virtual.html.*

[11] T. F. Abdelzaher, "Performance guarantees for web server end-systems: a control-theoretical approach," *Real-Time Technology and Applications Symposium, Proceedings. Fourth IEEE*, pp. 121-130, 1998.

류 상 우

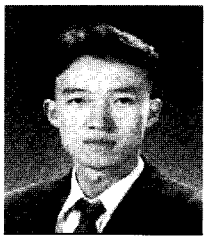
2000년 연세대 전기공학과 졸업. 현재 동 대학원 석사과정. 관심 분야는 네트워크 관리 및 고장포용 기술, 인터넷 QoS.

이 상 문

1997년 영남대 전기공학과 졸업. 1999년 연세대 대학원 전기컴퓨터공학과 졸업. 현재 동대학원 전기전자공학과 박사과정. 관심분야는 실시간 시스템, 고장포용 시스템, 인터넷 서버.

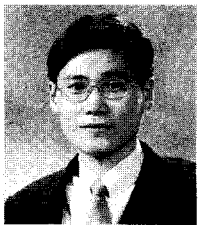
박 진 배

1977년 연세대 전기공학과 졸업. 1985년 미국 Kansas State University 전기 및 컴퓨터 공학과(공학석사). 1990년 미국 Kansas State University 전기 및 컴퓨터 공학과(공학박사). 1990년~1991년 미국 Kansas State University 전기 및 컴퓨터 공학과 조교수. 1998년~1999년 연세대 연세공학원(센터장). 1994년~현재 동대학 전기전자공학과 교수. 1999년~현재 제어·자동화·시스템공학회 이사. 관심분야는 강인제어, 비선형제어, 지능제어.



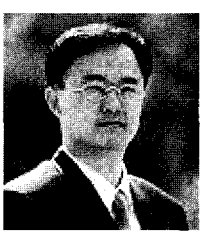
고 성 준

1999년 연세대 전기공학과 졸업. 현재 동 대학원 전기전자공학과 석사과정. 관심분야는 웹가속기, 커널 네트웍 프로그래밍, 인터넷 서버.



김 학 배

1988년 서울대 전자공학과 졸업. 1990년 미국 미시간대 대학원 전기공학과(EECS) 졸업(공학석사). 1994년 동 대학원 졸업(공학박사). 1994년 9월~1996년 8월 미국 National Research Council(NRC) Research Associate at NASA Langley Research Center. 1996년 9월~현재 연세대 전기전자공학과 부교수. 관심분야는 실시간시스템, 신뢰도 모델링 및 고장포용이론, 인터넷 인프라 기기, 홈네트워킹.



장 휘

1988년 서울대 기계설계공학과 졸업. 1989년 미국 미시간대 대학원 기계공학과 졸업(공학석사). 1993년 동 대학원 전산역학전공 졸업(공학박사). 1994년~1996년 Minnesota Supercomputer Center, Research Associate. 1996년~1997년 삼성자동차기술연구소 선임연구원. 1997년~2000년 서울산업대 조교수. 2000년 4월~현재 씨아이사(주) CEO. 관심분야는 QoS 지향 시스템, 내장형 S/W, 홈네트워킹, 고성능 웹서버.