2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

가상 호스팅 서버에서 QoS 향상 기법 연구

류상우*, 이상문, 민병석, 김학배
연세대학교 전기전자공학과
hbkim@yonsei.ac.kr

A Scheme Developing QoS in a Virtual Hosting Server

Sangwoo Ryou*, Sangmoon Lee, Byungseok Min, Hagbae Kim
Dept. of Electric and Electronic Engineering, Yonsei University

Abstract - 가상 멀티 호스팅 서버란 한 서버 안에 여러 도메인을 관리하면서 각각의 도메인에 해당되는 사이트를 운영할 수 있도록 하는 서버를 말한다. 보통 한 단일 서버 내에 수 십에서 수 백 개의 사이트를 운영하게 된다. 하지만 외부로부터의 트래픽이 많아지면 각 사이트가 제대로 동작하지 않을 수 있다. 본 논문에서는 각 사이트들이 독립적으로 운영되면서 컨텐츠 전달의 본 목적을 충분히 발휘하면서 최적의 성능을 유지할 수 있도록 하는 QoS를 향상 시킬 수 있는 구조를 연구한다. 또한 부하의 상태에 따라 피드백을 통한 부하 분산을 적용함으로서 QoS를 보장하고자 한다.

1. 서 론

용량이 얼마 되지 않는 홈페이지를 운영하기 위해 고성능 서버를 구입하는 것은 효율성으로 보면 낮은 효율을 뜻한다. 이 문제를 위해 솔루션으로 등장한 것이 호스팅 서비스이다. 호스팅이란 서버에 각 공간을 할당하고 URL 요청이 들어왔을 때 웹서버에서 각각의 공간으로 연결시켜주는 것을 말한다[1]. 클라이언트 입장에서는 마치 그 홈페이지가 독립된 서버에서 서비스 되는 것처럼 보이게 된다. 이 방식은 단일 서버에 하드 용량에 따라 다르지만 수 십개에서 수 백개의 사이트를 동시에 수용할 수 있게 되어 자원을 효율적으로 쓸 수 있는 장점이 있으며 방식에 따라 차이가 있지만 IP를 하나만 사용하여도 많은 수의 사이트를 운영할 수 있어서 IP를 절약할 수 있는 장점도 있다[2]. 반면 서버에 한 사이트에라도 과부하가 걸리면 서버의 나머지 사이트들이 한꺼번에 속도가 느려질 수 있는 약점이 있다.

현재 이런 서버 트래픽 문제를 해결하기 위해 나와있는 솔루션인 [3]는 IP 패킷을 분석하여 승인 또는 거부할지 여부를 판단하여 필요없는 트래픽을 제한하여 QoS를 보장한다. [4]는 멀티호스트내의 각 사이트 컨텐츠를 미리 여러 레벨로 구분해 놓은 다음 부하량에 따라 적적할 레벨의 컨텐츠를 제공하여 서버의 QoS를 보장한다. 그러나 여기서는 전체 서버의 QoS를 보장하기 위해 낮은 품질의 contents가 클라이언트에게 전달된다. QoS 보장과 컨텐츠 품질이 trade-off 관계에 있다. [5]는 리퀘스트를 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 매핑시켜놓고 도메인에 리퀘스트가 들어올 때마다 할당된 IP에 각각 공평히 분배하여 한 IP에 집중하지 못하도록 하는 방법이다. 로드밸런싱 기법[6]은 부하를 2개 이상의 서버에 나누어 부하를 분산하는 방법이다. 서버의 앞단에 리퀘스트를 전담하는 로드밸런서가 들어오는 리퀘스트를 리얼서버에 라운드로빈등의 스케쥴링 정책(policy)에 따라 나누어 부하를 분산케 함으로 여러 서버가 마치 한 개의 대형 서버처럼 동작하게 하는 방법이다.

본 논문에서는 멀티 호스트 서버와 백업 서버로

이루어진 간단한 시스템을 가지고 트래픽의 상태와 관계없이 일정한 서비스를 제공할 수 있는 피드백 시스템을 생각해본다. 즉 이 시스템은 트래픽을 분석하는 모니터링과 이 모니터 결과에 따라 변하는 실행기로 이루어진다. 리다이렉션 기법을 응용하여 트래픽 상태에 따라 두 단계로 트래픽을 구분하여 각각에 경우에 대해 QoS를 보장할 수 있는 방안을 제안한다. 트래픽이 평균치를 조금 상회할 때에는 파일이 큰 컨텐츠의 URL을 변환하고 평균치를 많이 상회할 때는 서버를 리다이렉션하는 방법을 이용할 것이다. 이런 분산 기법을 통해 트래픽이 많이 걸렸을 때에도 정상적인 서비스를 중단없이 공급하는 기법을 제안하고 이에 대해 논의하고자 한다.

2. 시스템의 구조

2.1 구조

시스템은 논리적으로는 시스템의 상태를 감시하고 판단하는 모니터 부분과 판단된 상태에 따라 대처하는 실행기의 구조로 볼 수 있다.

· 모니터 - 트래픽을 감시하는 기능을 담당한다. 트래픽은 패킷을 일일이 검사하여 시스템에 부하를 주는 방법 대신 각 사이트에 들어오는 리퀘스트를 로그 파일을 통해 트래픽 상태를 감시한다. 먼저 각 로그 파일의 크기를 먼저 감시하고 크기가 급격히 증가하면 로그 파일의 구체적인 내용을 감시한다.

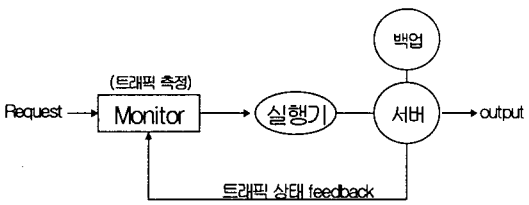

그림 1. 시스템 다이어그램

· 실행기 - 모니터의 상태 판단에 따르는 조치를 취한다. 과부하라고 판단이 되면 상태에 따라 html 에 포함된 큰 사이즈 파일의 URL을 백업 서버의 URL로 바꾸는 조치를 취하고 만약 부하가 더 증가하면 부하가 걸린 호스트 사이트의 html의 헤더에 라다이렉션 스크립트를 삽입하는 조치를 취한다.

· 주 서버 - 정상 상태(Normal state)일 때 홈페이지를 서비스하는 서버이다.

· 백업 - 과부하 발생시 html을 제공하게 되는 서버이다. 백업서버에는 본래 서버의 디렉토리 구조와 동일한 구조와 내용을 저장하고 있다.

2.2 원리

웹서버에서 각 사이트에 들어오는 트래픽을 감시하고 있다가 기준 이상의 요청이 들어오면 그 다음부터는 복제된 사이트로 요청을 전달해 주는 것이다. 백업 서버는 리소스를 다른 사이트와 공유하지 않고 부하가 걸린 사이트만을 위해 돌아가기 때문에 부하가 많이 줄어들 수 있게 된다. 이 방법은 기존의 리눅스 가상 서버[6]에서 사용하는 가상 IP와는 다르다. 가상 서버는 부하가 걸린 서버를 완전히 이전하고 주 서버를 사용하지 않는데 반해 제안된 방법은 주 서버에서 부하가 걸린 디렉토리만 백업서버에서 그 기능을 담당하는 것으로 더 효율적이라 할 수 있다.

2.3 모니터링

서버의 로그 파일을 분석하여 모니터링한다[7]. 1단계로 각 사이트의 로그 파일 사이즈를 감시한다. 이를 통해 급격히 커지는 파일을 찾아낸다. 로그 파일 사이즈가 커진다는 것은 그만큼 리퀘스트가 높아진다는 것을 의미한다. 이렇게 찾아진 로그 파일의 특정 문자열 즉 각 사이트에 해당하는 IP 문자열을 주기적으로 체크한다. 시간에 대해 미분을 하여 주기 사이에 값이 급격히 증가하면 일단 사이트를 "주의 모드"로 세팅한다. 이후에 주기(period)를 줄이며 모니터링한다.

2.3.1 대역폭(Bandwidth) 모니터링

파일 크기가 큰 파일은 접속 회수가 적더라도 CPU 점유율이 높으므로 이에 대한 대안이 필요하다. 이를 위하여 각 사이트에 상주하는 큰 파일에 대해 감시를 별도로 한다. 개별 파일을 감시하기보다는 그 파일을 포함하는 html 파일의 로그 파일 크기를 체크하여 그 특정 파일의 로그 파일이 커지면 바로 감시에 들어가는 방식을 이용하면 효율적으로 감시할 수 있다. 즉 로그 파일에서 위에서 언급한 특정 html 파일에 대해서 따로 감시하게 되면 개별 html의 로그 파일을 따로 작성할 필요가 없으므로 효율적이라고 할 수 있다.

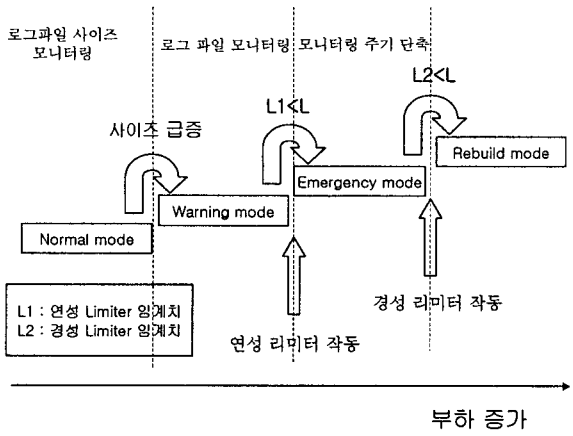

그림 2. 부하에 따른 모드 변환

각 단계를 결정하는 L은 각 사이트의 로그 파일 사이즈를 말하며 L1, L2는 연성과 경성 리미터 임계치를 말한다.

2.3.2 모니터링 개선

로그 파일 분석 단순화는 모든 파일의 입출입 기록을 추적하는 것보다 발전된 방법으로 그 사이트에 들어가기 위한 관문(portal)인 index.html, index.php. default.html등 웹서버 설정 파일에 디폴트로 설정되어 있는 파일만을 감시하는 방법이다. 각 사이트를 접속하려는 클라이언트는 특별한 경우가 아니라면 대부분 디폴트 파일을 거쳐 들어가야만 한다. 이는 로그 파일의 각 행을 모두 감시할 필요가 없어 훨씬 원활한 모니터링 방법이 된다.

3. 부하분산 알고리즘

3.1 상태 판단(Decision)

정상 상태 여부의 판단을 위해 평소 트래픽 형태를 조사한 데이터를 이용한다. 평소의 홈페이지나 특정 파일에 대한 히트율을 평균과 분산등의 1,2차 모멘트로 기록해놓는다. 이 기록에 바탕하여 현재의 히트율을 분석, 평소 히트율과 비교를 통해 유효범위내에서 기각할지 아닐 지를 판단한다. Large number theory를 적용하여 샘플이 많아지면 어느 특정한 날 트래픽이 갑자기 평소 데이터와 다른 형태를 보이지 않을 것이라는 전제를 바탕으로 한다.

3.2 트래픽 제어 방법

트래픽 컨트롤 방식을 리미터(limiter)라 정의한다. 미터에는 방식에 따라 경성리미터와 연성 리미터로 나눈다. 기준이 되는 L1, L2는 평소 트래픽을 기준으로 평균 트래픽의 일정 비율을 초과했을 때를 기준으로 잡는다. 이를 위해 일단 특정 값을 정하고 이 데이터를 실제로 적용하면서 계속 보정해간다.

3.2.1 연성 리미터

연성리미터는 부하량을 측정하여 트래픽이 발생했을 때 부하량이 많이 걸리는 사이트의 컨텐츠중 용량이 크고 CPU 점유율이 높은 파일을 백업 서버에서 클라이언트에 제공하는 방식이다. 다음과 같이 미리 각 사이트에 정해놓은 특정 파일 - 크기가 큰 jpg나 mpg 같은 파일의 URL를 치환하여 주 서버에 걸리는 부하를 백업 서버로 분산 시키는 효과를 가져 올 수 있다.

img src="./picture/mypic.jpg" → img src="**http://백업서버URL/picture/mypic.jpg**"

다음은 위의 연산을 수행하는 연성 리미터의 pseudo-code이다

```
//
soft-limiter( )
{
scanning html in overload directory
replace <img src="./picture/mypic.jpg"> with
<img src="http://백업서버URL/picture/mypic.jpg">
}
//
```

3.2.2 경성 리미터

경성리미터는 제한량이 훨씬 초과했을 때 서버에서 리다이렉팅을 하는 방법이다. 리다이렉팅에는 두 가지 방식이 있는데 DNS 서버에서 IP mapping을 바꿔주는 방법과 단순히 html 파일의 헤더에 스크립트를 추가하는 방법이 있다. 전자는 DNS 세팅을 바꿔야하는 절차가 복잡하기 때문에 여기서는 단순히 html의 헤더에 스크립트를 삽입하는 후자의 방법을 쓴다. 부하가 많이 걸려 정상적인 서비스가 어렵다고 판단이 되면 스크립트를 실행시켜 디렉토리안의 html의 헤더에 다음과 같은 코드를 삽입한다.

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

위의 연산을 수행하는 경성 리미터의 pseudo-code이다

```
//
```

```
hard-limiter( )
{
scanning html in overload directory
insert redirection script in html header
}
//
```

클라이언트의 요청으로 서버의 html이 읽혀지면 바로 백업서버의 html이 클라이언트 컴퓨터에 로딩되어 부하가 백업서버로 전환되는 효과가 생긴다.

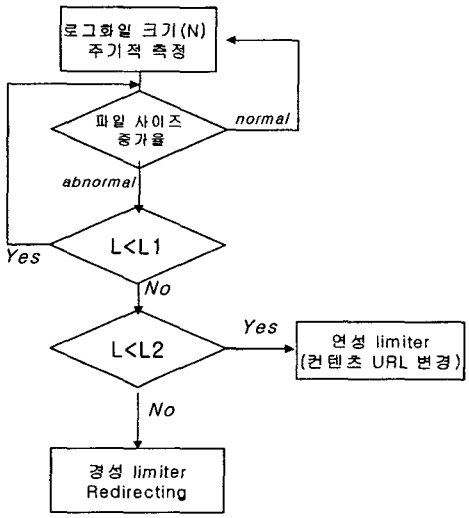

그림 3. 부하분산 알고리즘

4. 결론

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 동작시에도 전체 서버 및 각 사이트의 과부하 발생에 따라 부분적으로 서버의 역할을 하게 함으로서 부하 분산의 효과를 기대할 수 있다. 서버의 트래픽 모니터링을 위하여 각 사이트의 로그파일을 분석한다. 이것은 간단한 스크립트로 작성할 수 있으며, 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 제시하였다.

참 고 문 헌

[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] Hani J., John R., Kang G. Shin, " QGuard : Protecting Internet Service from Overload" 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.; 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.