웹 서버 클러스터에서 내용 기반으로한 부하 분산 기법

명 원 식† · 장 태 무††

요 약

인터넷의 급속한 성장과 더불어 인기 있는 웹 사이트는 우수한 성능의 단일 서버 또는 미러(mirror) 사이트에만 의존 할 수 없을 정도로 인터넷 사용이 급증하였다. 이러한 인터넷 사용과 사용자의 급증은 과중한 전송량과 시스템의 부하문제를 야기 시켰으며 이를 해결하기 위한 여러 방안으로 클러스터 시스템이 연구되어지고 있다. 특히 클러스터의 모든 컴퓨터 성능을 최대한 발휘하기 위해서는 클라이언트의 요구를 적절히 분산시키는 스케줄링 방법이 중요하다. 본 논문에서는 웹 서버 클러스터에서 서비스 지연을 감소시키기 위해 클러스터내의 웹 서버들이 각각 서로 다른 내용을 가지는 기법을 제안한다. 이는 부하 분배기의 알고리즘을 구현하는데 있어 오버헤드를 줄이고 응답시간을 감소화 하는데 있다. 또한 부하 분배기로부터 요청을 받은 웹 서버는 부하 분배기를 거치지 않고 바로 클라이언트에게 응답을 보낸다. 성능 측정을 통해 기존의 다른 방식보다 제한한 기법이 보다 우수함을 보이고, 응답시간에서도 기존의 RR(Round Robin)과 LC(Least Connection) 방식보다 제안한 웹 서버 클러스터 기법이 각각 16%, 14% 짧다는 것을 보인다.

Content_based Load Balancing Technique In Web Server Cluster

Won-Shig Myung† · Tea-Mu Jang††

ABSTRACT

With the rapid growth of the Internet, popular Web sites are visited so frequently that these cannot be constructed with a single server or mirror site of high performance. The rapid increase of Internet uses and users raised the problems of overweighted transmission traffic and difficult load balancing. To solve these, various schemes of server clustering have been surveyed. Especially, in order to fully utilize the performance of computer systems in a cluster, a good scheduling method that distributes user requests evenly to servers in required. In this paper, we propose a new method for reducing the service latency. In our method, each Web server in the cluster has different content. This helps to reduce the complexity of load balancing algorithm and the service latency. The Web server that received a request from the load balancer responds to the client directly without passing through the load balancer. Simulation studies show that our method performs better than other traditional methods. In terms of response time, our method shows shorter latency than RR (Round Robin) and LC (Least Connection) by about 16%, 14% respectively.

키워드 : Load Balancing, Content, Round Robin, Least Connection

1. 서 론

World Wide Web(이하 웹)은 저렴한 가격과 다양하고 흥미 있는 정보를 쉽게 간편하게 찾아볼 수 있다는 이점으로 웹 사용자는 하루가 다르게 증가되고 있다. 웹 사용자의 증가와 함께 웹을 통해 전달되는 데이터 즉, 그림, 멀티미디어 데이터, 웹 문서 등의 크기 또한 빠르게 증가되고 있다[1, 2]. 또한, 최근에는 인터넷 트래픽(Internet traffic)의 많은 부분이 웹 트래픽(Web traffic)으로 구성되고 있으며 이러한 웹의 급격한 보급과 각종 멀티미디어 데이터를 포함하는 웹 서비스의 팽창은 네트워크의 병목현상(Bottleneck)을 점점 더 가중시키고 있다. 따라서 트래픽의 증가로 인해 웹 서버의 성능(Performance)과 가용성(Availability)을 높이는 방안이 더욱 중요시되고 있다.

시스템의 성능을 향상시키는 방안으로서 과거에는 높은 성능의 프로세서(Processor)를 장착하거나 슈퍼컴퓨터와 같이 여러 개의 CPU를 채택한 병렬 처리(Parallel Processing) 기법[3]을 사용하였다. 그러나, 단일 서버로의 성능향상은 물리적인 확장의 한계로 인하여 네트워크(Network) 트래픽의 증가추세를 따라오지 못하고, 성능대비 투자비용이 기하급수적으로 증가되어서 효율성의 매우 떨어졌다.

근래에 들어서는 서버의 수용 한계를 넘어선 트래픽의 증가로 인해 시스템의 성능을 향상시키기 위해 확장성과 가격 대 성능비가 우수한 웹 클러스터(Web Cluster)기법에 대하여 연구되고 있으며 최근에는 웹 서비스와 연계하여 많은 연구가 진행되고 있다[7, 8]

현재 인기 있는 웹 사이트에서는 매일 수백만의 사용자 요구를 수용하기 위해 다수의 서버를 사용하고 있으며 하

† 준 회 원 : 동국대학교 대학원 컴퓨터공학과
†† 정 회 원 : 동국대학교 컴퓨터공학과 교수
논문접수 : 2003년 7월 19일, 심사완료 : 2003년 11월 27일

나의 가상 URL을 이용하여 사용자에게 투명성을 제공하는 방식을 일반적으로 채택하고 있다[9]. 이러한 클러스터링을 이용한 웹 서버 구축 시 중요하게 고려해야 할 사항은 각 웹 서버 노드에 적절한 부하 분산이 이루어지도록 하는 것이며 웹 서버 노드에 결함이 발생할 경우에 대비하여 적절한 서비스 시스템 이주(Migration)가 이루어지게 해야 한다. 이러한 알고리즘은 시스템 성능에 가능한 한 영향을 적게 주어야 하며 결함에 따른 이주 과정에 있어서도 신속히 이루어져야 한다. 하지만 실시간 상태에서 이러한 이주를 한다는 것은 많은 오버헤드가 발생하게 된다. 따라서 부하 분배기(load balancer)는 중재 알고리즘을 통해 사용자의 요청이 발생할 때마다 각 웹 서버 노드들에게 최적의 부하 분배를 하며 특정 웹 서버 노드에 결함이 발생했을 경우에도 나머지 웹 서버 노드들에게 적절한 분배가 이루어지도록 하는 역할을 한다. 그러므로 부하 분배기의 서비스 정책은 시스템 전체 성능에 중요한 영향을 미치게 된다.

웹 서버 클러스터 시스템은 부하를 클러스터 내의 다른 서버로 분산시키는 조정자 역할을 하는 부하 분배기와 실제 서비스를 수행하고 요청에 대해 응답하는 서버로 구별된다. 웹 서버 클러스터 시스템에서 부하 분배기는 서비스 요청이 있을 경우, 특정한 알고리즘을 바탕으로 적절한 웹 서버를 선택하여 그 요청을 처리하게 한다.

본 논문에서는 고가용도 및 확장성을 제공하는 웹 서버 클러스터에서 내용 기반 부하 분산 기법을 제안하고 응답시간을 최소화할 수 있도록 하였다. 즉 웹 사이트에서 제공되어지는 컨텐트들에 대해 단일 웹 서버로 서비스하는 것이 아니고 각기 다른 컨텐트들을 서로 다른 웹 서버로 분리 관리하여 부하 분배기의 알고리즘을 계산하는데 있어 오버헤드를 줄이고 응답시간을 최소화하는데 목적을 두었다.

논문의 구성으로 2장에서는 관련 연구를 알아보고 3장에서는 본 논문에서 제안한 내용 기반으로 한 웹 서버 클러스터 기법에 대해서 알아본다. 또한 4장에서는 시뮬레이션 결과를 설명하고 마지막 5장에서 결론을 맺는다.

2. 관련 연구

클러스터 기법이란 병렬 처리 기법의 한계를 뛰어 넘어 시스템의 성능을 향상시키기 위한 방법으로 네트워크에 접속된 여러 개의 서버를 하나로 연결하는 방법을 말하며, 웹 서비스 기반하의 클러스터 기법을 웹 서버 클러스터라 한다. 한편, 웹 클러스터를 구현하는 방법으로 구현 아키텍처(Architecture)에 따라 브로드캐스팅(Broadcasting) 방식[3], 라운드 로빈 DNS(Round Robin DNS) 방식[4], 직접 라우팅(Direct Routing) 방식[5], 디스패처(Dispatcher) 방식[6] 등이 있다.

본 장에서는 구현 아키텍처와 부하 분산기에서 작업 할당하는 스케줄링에 대하여 기술한다.

2.1 브로드캐스팅 방식

이 방식은 물리적으로 네트워크에 도달한 트래픽을 모든 서버에 전달 시키고 그 중 특정한 한 서버만이 응답을 하는 방식으로, 이더넷(Ethernet)의 브로드캐스팅 특성을 이용한 방식이다. 장점으로는 병목 현상을 발생시키는 단일 지점이 없게 된다. 단점으로는 서버 운영체제의 이더넷 어댑터(Adapter)를 수정하여야 하기 때문에 GNU/Linux와 같이 소스코드가 공개되고 수정이 허용되는 운영체제 이외에는 적용될 수가 없고, 네트워크상의 모든 패킷 (Packet)이 모든 서버에서 수신되어야 하기에 불필요한 트래픽이 증가하게 된다.

2.2 라운드 로빈 DNS 방식

이 방식은 사용자가 해당 서비스의 도메인 주소(Domain Address)를 DNS 서버를 통해 IP 주소로 변환하는 과정에서 변환을 요청할 때마다 여러 서버의 다른 IP를 알려 주어 부하를 분산하는 방식이다. 장점으로는 서버의 운영체제에 비종속적이며 적은 기회 비용으로 손쉽게 구축이 가능하다. 단점으로는 구현 특성상 매번 연결을 위해 서버의 IP를 질의하기 때문에 이에 따른 연결 지연(Delay)이 존재하고, DNS 서버가 웹 클러스터 서버의 장애여부를 파악하지 못하므로 서버 장애에 따른 페일 세이프(fail-safe) 기능이 구현되지 못한다.

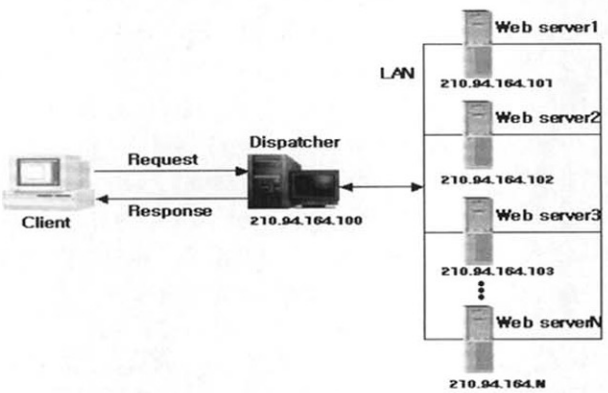
2.3 직접 라우팅 방식

직접 라우팅 방식은 요청을 중계해 주는 서버가 존재하여 클라이언트의 요청(Request)을 서비스 서버에 연결하여 주는 방식이다. 클라이언트의 최초 요청은 중계 서버를 거쳐 서버에 연결이 되지만 요청에 대한 서버의 응답은 클라이언트와 직접 통신하는 특성을 갖는다. 장점으로는 중계 서버의 트래픽 집중화를 막을 수가 있으며, 중계 서버의 다양한 트래픽 분배 정책에 따라 트래픽을 어느 정도 효율적으로 분배할 수 있게 된다. 단점으로는 브로드캐스팅 방식과 같은 이유로 서버의 운영체제에 제약이 있으며, 요청에 대한 응답은 직접 클라이언트와 이루어지기 때문에 서버의 트래픽 처리량을 정확하게 계산할 수가 없게 된다.

2.4 디스패처 방식

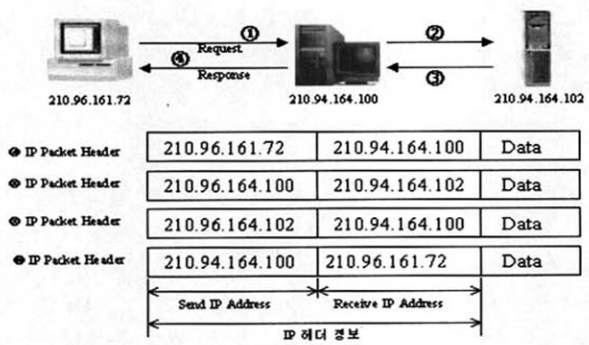
디스패처 방식은 직접 라우팅 방식과 구조적으로 유사하게 웹 클러스터 앞 단에 중계 서버가 존재한다. 그러나 직접 라우팅 방식과는 다르게 모든 클라이언트의 요청과 서버의 응답은 중계 서버를 통한다는 특성을 갖는다. 장점으로는 모든 요청과 응답을 중계 서버가 감시하기 때문에 서버의 상태 파악이 수월하고, 서버의 이더넷 어댑터를 수정할 필요가 없으므로 운영체제에 종속적이지 않고 자유롭다. 단점으로는 중계 서버가 모든 트래픽을 처리하여야 하므로 단일 병목 지점으로 작용할 수가 있게 된다.

현재의 웹 클러스터 구조는 클라이언트의 요청을 적절한 웹 서버로 전달해주는 하나의 디스패처와 동일한 컨텐트를 가진 여러 대의 서버 노드로 구성되어 있다. 이 방식은 클라이언트들로부터 서비스 요청을 디스패처에서 특정한 웹 서버에 집중되지 않도록 적절한 스케줄링방식으로 전체 서버에게 전달하는 디스패처 방식으로 이루어지고 있다. 위 설명은 (그림 1)과 같다.



(그림 1) 디스패처 방식 구조

이러한 디스패처 방식은 모든 클라이언트의 요청을 제어할 수 있고 디스패처의 처리 작업을 통해서 웹 서버에게 전달된다. 또한 디스패처로 도착하는 모든 패킷들을 관리하고 변경하는 작업들은 많은 처리 비용이 들고 클라이언트의 요청이 많아지면 병목 현상을 일으킬 수 있다. 나아가 문자 정보 같은 작은 데이터만 요구하는 패킷을 받는 경우는 별 문제가 되지는 않지만 요즘과 같이 기본적인 문자뿐만 아니라 동영상 파일, 음악 파일, 이미지 파일과 같은 많은 양의 멀티미디어(MPEG, MP3)를 요구하는 경우엔 웹 서버간 부하가 균등하더라도 요청된 데이터의 크기가 크면 특정 웹 서버의 부하는 증가되고 전체 웹 서버의 부하는 불균등 하게 된다. 따라서 패킷을 처리하는데 많은 시간이 걸리거나 심각할 경우 더 이상 서비스를 할 수 없게 될 수도 있다. (그림 2)는 디스패처 방식의 IP 패킷 헤더 정보를 Rewriting하는 과정을 보여주고 있다.



(그림 2) IP Packet의 Rewriting

웹 서버 클러스터링의 부하 분배기는 클러스터 내에 있는 특정 웹 서버에 부하를 집중시키지 않기 위해 적절한 스케줄링 알고리즘을 통해 부하를 분산시키는 역할을 하며 웹 서버 클러스터가 하나의 가상서버(IP-SVA)로 보이게 하는 역할을 한다. 즉 서비스 요구 패킷이 부하 분배기에 들어오면 스케줄링 알고리즘에 의해 실제 웹 서버가 선택되고 선택된 웹 서버로 그 패킷이 전달된다. 이 때 사용되는 스케줄링 알고리즘에 따라 작업 분배가 이루어지므로 분배 알고리즘은 매우 빠르게 계산되어져야 한다. 따라서 작업 분배 스케줄링 알고리즘은 웹 서버 클러스터의 성능에 매우 중요하다.

2.5 NAT(Network Address Translation) 방식

이 방식은 인터넷상의 공인된 IP 주소로 들어오는 클라이언트의 프로토콜 헤더를 수정하여 클러스터 내부의 다른 웹 서버들로 연결시키는 방식으로 클라이언트 입장에서는 단일한 웹 서버에 직접 연결된 듯 보이는 방법이다. 이러한 NAT의 특성을 이용하여 서로 다른 서버 IP 주소들을 구축된 시스템에 하나의 IP 주소상의 가상서비스로 구축할 수 있다.

클라이언트가 클러스터에게로 구성되어 있는 가상의 서버에 서비스를 요청하면 가상 IP 주소로 향하는 요구패킷은 부하 분산기에 도달하고, 부하 분산기는 패킷의 목적지 주소와 포트번호를 검사한 후 스케줄링 알고리즘에 따라 실제 서버를 선택한다. 부하 부산기는 구성된 접속의 내용을 해쉬 테이블에 저장하고, 실제 웹 서버는 부하 분산기로부터 전달된 클라이언트의 요청작업을 실행한 후 결과 패킷을 다시 부하 분산기에 전달한다. 부하 분산기는 해쉬 테이블에 저장된 접속 내용이 들어온다면 그 패킷의 발생지 주소와 포트를 가상 IP 주소의 것으로 변경하여 클라이언트에게로 전송한다.

이러한 NAT에 의한 가상서버는 TCP/IP를 지원하는 어떠한 운영체제에서도 운용될 수 있고 또한 단지 부하 분산기만 실제 IP 주소를 필요로 하고 다른 실제 웹 서버들은 자체적인 주소체제를 사용할 수 있다는 장점을 지니고 있다. 그러나 실제 웹 서버의 수가 일정한 개수 이상 늘어날 경우 실제 웹 서버에 작업 분산을 수행하는 부하 분산기가 전체 시스템의 병목구간이 될 수 있는 문제점을 가지고 있다.

2.6 부하 분배 스케줄링 알고리즘

부하 분배 스케줄링에 이용되는 방법 중 라운드 로빈(Round Robin) 방식은 클러스터로 구성된 모든 웹 서버들이 동등하게 한번씩 돌아가며 요구에 대하여 서비스하는 방식이다. 이는 각 웹 서버의 서비스 처리 용량과 관계 없이 서비스되므로 부하 불균형이 발생할 수 있는 단점이 있다. 이러한 문제점을 고려하여 라운드 로빈 스케줄링과 유사하나 클러스터에 속한 웹 서버들의 처리 용량에 따라 가중치를 둔 뒤 연결하는 가중기반 라운드 로빈 스케줄링 방식이 있

다. 또한 연결된 사용자 요구 수가 제일 적은 곳을 우선적으로 클라이언트 요구를 전달하는 최소 연결(Least Connection) 스케줄링 방식이 있고, 각각의 실행 웹 서버에 성능 가중치를 부여하고 동적으로 작업 접속상황을 조사하는 방식으로 실제 웹 서버를 선택하는 가중치 기반 최소 연결(Weighted Least Connection) 스케줄링 방식이 있다.

이러한 부하 분산 기법들을 이용할 경우 하나의 IP 주소로 서비스가 가능하며 웹 서버의 결합 발생 시 나머지 웹 서버들에 부하를 분산시킴으로써 결함을 허용할 수 있는 장점이 있다. 부하 분배기에서 일어날 수 있는 병목 현상은 IP 터널링이나 직접 라우팅을 통해 서비스를 하는 웹 서버와 클라이언트가 직접 통신함으로써 해결될 수 있다[16].

3. 내용 기반으로 한 웹 서버 클러스터 시스템 설계

일반적으로 웹 서버 클러스터 시스템은 하나 또는 그 이상의 부하 분산기와 클라이언트의 요청에 대한 실제 웹 서비스를 해주는 각각의 웹 서버들로 구성되어 있고, 부하 분산기 관리 하에 구성되어 있는 웹 서버들의 내용들은 복사되어 제공된다. 이러한 시스템으로 구성된 웹 서버는 클라이언트의 요청이 많아질수록 병목현상이 발생할 확률이 높아져 전체적인 시스템의 성능이 떨어질 수 있다. 또한 데이터 베이스 변경 등과 같은 갱신으로 인해 전체 컨텐트의 일괄성이 깨지는 문제가 발생할 수 있다. 이러한 문제점을 좀 더 효율적으로 해결하고자 내용 기반으로 한 웹 서버 클러스터 시스템을 제안한다.

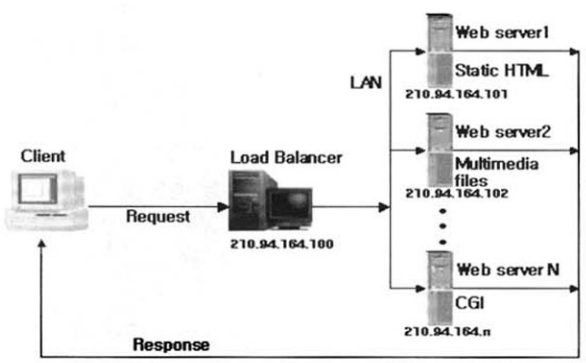
본 논문에서 제안하는 웹 서버 클러스터 기법은 웹 사이트의 컨텐트들을 인덱스별로 나누어 각각 다른 웹 서버로 구성되어지고 클라이언트의 요청을 부하 분산기에서 인덱스 테이블을 통해 해당 웹 서버에게로 분배하는 방식이다. 또한 이 시스템 구조는 기존의 직접 라우팅 방식으로 구성되어져 있다. 따라서 서비스를 하는 실제 웹 서버에서 직접 클라이언트에게로 서비스에 대한 응답을 전달하는 방식이다.

제안하는 이 시스템 구조는 각 웹 서버에 대한 효과적인 부하를 분산하는데 있어서 컨텐트들을 인덱스 테이블로 관리하고 이에 해당하는 요청을 인덱스 테이블에 따라 바로 해당 웹 서버로 요청을 전달하기 때문에 부하 분산기의 트래픽에 대한 오버헤드를 감소시키고, 웹 서버들은 각각 다른 컨텐트를 서비스하기 때문에 클라이언트의 요청에 대한 서비스 응답 시간을 최소화할 수 있다.

3.1 웹 서버 클러스터 시스템 구조

기존의 디스패처를 가진 시스템에서는 부하를 균등하게 하는 데는 성능이 좋을 수도 있지만, 요즘의 웹 환경과 같은 기본적인 문자 뿐만 아니라 동영상 파일, 음악 파일, 이미지 파일 등과 같은 많은 양의 멀티미디어를 요구하거나 전자상거래의 CGI를 제공하는 경우엔 웹 서버간 부하가 균등하더라도 요청된 데이터의 크기에 따라 특정한 웹 서버의 부하는 증가되고 이에 따른 병목현상이 발생하게 된다. 따라서 패킷들을 처리하는데 많은 시간이 걸리거나 심각할 경우 더 이상 서비스를 할 수 없게 될 수도 있다. 따라서 본 논문에서는 이러한 문제점을 개선하고자 웹 사이트의 다양한 컨텐트들을 서로 다른 웹 서버로 분리하여 서비스하는 시스템을 제안한다. 제안한 웹 서버 클러스터 시스템 설계는 (그림 3)과 같다.

(그림 3)은 하나의 부하 분산기와 부하 분산기가 관리하는 웹 서버들은 각각 다른 컨텐트의 종류에 맞게 구성되어 있다. 즉 video/audio, CGI, html/text 등 컨텐트의 종류에 맞게 웹 서버들을 특성화 시킨 것이다. 이 구조는 동적 웹 페이지의 생성이나 데이터베이스의 추가, 변경, 삭제 등의 갱신 시에도 해당 웹 서버에서만 존재하므로 관리가 용이하다. 또한 데이터 증가에 따른 저장공간 추가와 각 웹 서버의 용량, 처리속도의 변동에도 효과적으로 대처할 수 있다.

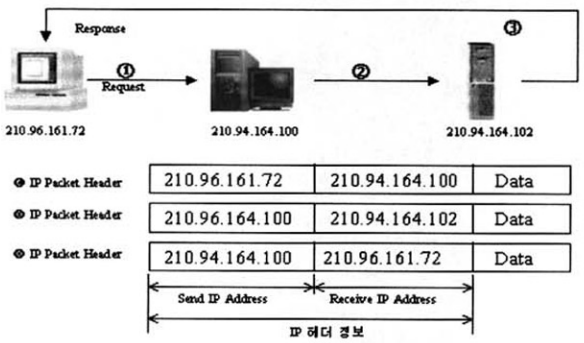


(그림 3) 웹 서버 클러스터 시스템 설계

제안한 웹 서버 클러스터 시스템 구성은 들어오는 모든 클라이언트의 요청에 대해 각각의 컨텐트를 가지고 있는 웹 서버(실행 서버)에게로 전달하는 하나의 부하 분산기와 클라이언트 요청에 대한 서비스 응답을 하는 N개의 웹 서버를 가지고 있다. 각 웹 서버에는 IP 주소가 있는데, 클라이언트 입장에서는 부하 분산기를 서버로 인식하여 모든 클라이언트는 부하 분산기의 단일 IP 주소(그림 3)에서는 210.94.164.100)만으로 접근 하며, 웹 서버 자체는 감추어지게 된다. 제안한 웹 서버 클러스터 시스템은 앞 절에서 기술한 것처럼 직접 라우팅 방식으로 운영된다. 부하 분산기는 클라이언트 요청을 인덱스 테이블에 따라 해당 웹 서버로 요청을 전달되며, 전달은IP 패킷 단위로 처리를 한다. 클라이언트에서 송신된 IP 패킷을 해당 웹 서버로 전달하는데 이때 클라이언트 IP 패킷 헤더 정보의 목적지 주소를 실제 웹 서버의 IP 주소로 바꾸는 작업을 통해 실제 웹 서버에게 요청을 전달한다. 요청을 전달 받은 웹 서버는 클라이언트에게 응답할 때 자신의 주소를 부하 분산기 서버의 주소로 변환하여 응답을 한다. 이 IP 패킷 헤더 정보 수정

(Rewriting) 과정을 도식화하면 (그림 4)과 같다.

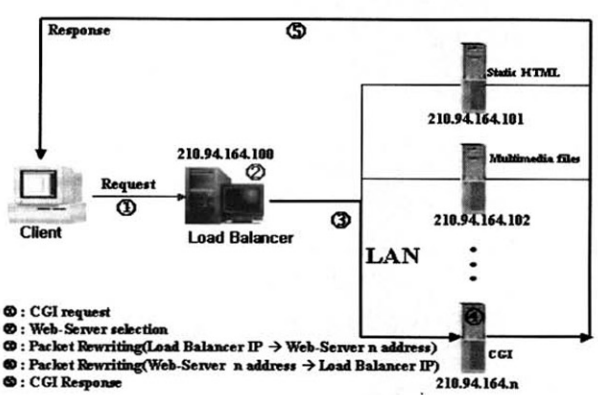
한편, N개의 웹 서버는 각각 다른 컨텐트의 형태를 가지고 있다. (그림 3)에서 보는 것처럼 웹 서버1(Web server 1)은 정적 컨텐트(Static HTML : HTML, Document, Image, etc.)를 가지고 있고, 웹 서버2(Web server 2)는 멀티미디어 컨텐트(Multimedia files : video, audio, etc.)를 가지고 있다. 웹 서버 N(Web server N)번째는 동적 컨텐트(CGI : 전자상거래, ASP, JSP, etc.)를 가지고 있다. 따라서 부하 분산기 서버에서 전달된 클라이언트 요청을 해당 웹 서버에서 응답하게 되며 클러스터로 구성된 웹 서버들은 LAN으로 구성되어져 있다.



(그림 4) 제안한 시스템 IP Packet의 Rewriting

3.2 동작 원리

제안한 내용 기반의 웹 서버 클러스터의 동작은 (그림 5)와 같다. 클라이언트가 부하 분산기 서버 이름이 포함된 URL로 DNS에게 주소를 요청하고 DNS는 부하 분산기 서버의 주소를 응답한다. 요청 받은 부하 분산기 서버는 클라이언트의 요청을 인덱스 테이블을 통해 컨텐트를 가지고 있는 웹 서버를 선택한 후 요청을 해당 웹 서버에게로 전달하게 된다. 요청을 처리한 웹 서버는 자신의 주소를 숨기고 부하 분산기 서버의 주소로 다시 변환한 후 클라이언트에게 응답을 하고 클라이언트는 실제 웹 서버의 주소를 모른 채 부하 분산기 서버에게로 서비스 요청을 계속하게 된다.



(그림 5) 내용 기반으로한 웹 서버 클러스터 시스템 동작 원리

실질적으로 직접 라우팅 방식의 단점은 웹 서버에서 직접 클라이언트에게로 응답하기 때문에 부하 분산기 서버에서는 각 웹 서버에서의 트래픽을 정확하게 파악할 수 없다. 따라서 본 논문에서는 각 웹 서버의 트래픽에 대한 병목현상이 발생하지 않는다는 조건을 두고 실험한다.

〈표 1〉 부하 분산기 서버 내의 인덱스 테이블 요소들

Content	Web server address
Static HTML(HTML, text, document 등)	210.94.164.101
Multimedia files(gif, jpg, swf 등)	210.94.164.102
:	:
CGI((게시판, 방명록, DB관련 등)	210.94.164.n

<표 1>은 클라이언트 요청이 부하 분산기에 도착했을 때 요청에 서비스를 하기 위하여 해당 웹 서버를 선택해야 하는데, 이때 부하 분산기 서버 내의 인덱스 테이블 요소들을 가지고 해당 웹 서버를 선택하도록 함을 보여주고 있다. 예를 들면, 클라이언트 요청이 게시판을 접속하고자 한다면 부하 분산기 서버는 자기가 관리하는 인덱스 테이블에 따라 210.94.164.n의 IP 주소를 갖는 웹 서버에게로 요청을 전달하게 된다.

4. 실험 및 고찰

본 논문에서 제안한 웹 서버 클러스터 기법을 객관적으로 평가하기 위하여 1대의 부하 분산기와 3대의 웹 서버로 구성하였으며, 시스템 사양은 Pentium Ⅲ 1.GHz, 256 RAM, HDD 30GB로 구성하였다. 실험에 사용된 장비는 <표 2>과 같다. 또한, WOW Linux 7.0 운영체제에서 UC Berkley 대학의 네트워크 시뮬레이터인 ns-2.1b8a를 사용하여 시뮬레이션 하였다. 성능 평가를 위해 사용한 Data Set은 본 대학의 웹 서버 로그 파일을 표본 추출하여 Data Set으로 이용하였고, 실험 결과 비교는 앞에서 언급한 RR스케줄링 방식과 LC 스케줄링 방식, 그리고 제안한 내용기반으로 한 웹 서버 클러스터 구조 기법에 대해 실험하였다.

〈표 2〉 실험에 사용된 장비

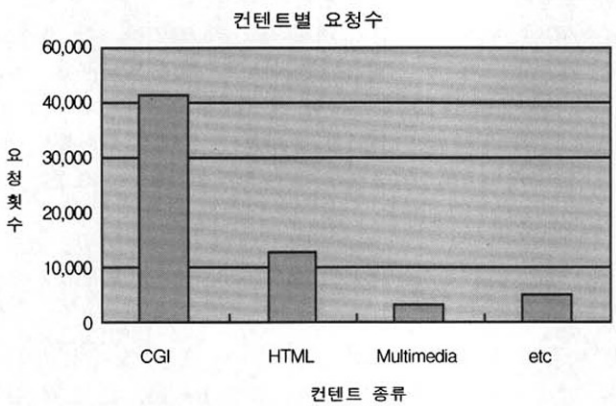
장비 이름	CPU	메모리(memory)	디스크(disk)
Load Balancer	PentiumⅢ 1GHz	256MB	30GB
Web Server1	PentiumⅢ 1GHz	256MB	30GB
Web Server2	PentiumⅢ 1GHz	256MB	30GB
Web Server3	PentiumⅢ 1GHz	256MB	30GB

4.1 로그 파일 분석

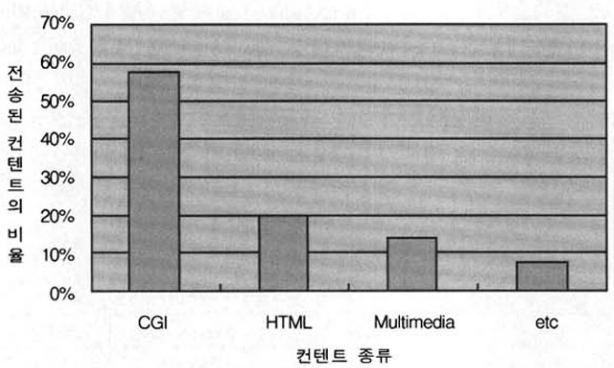
본 논문에서 제안한 내용 기반의 웹 서버 클러스터 시스템을 실험하기 위해 본 대학의 웹 서버 로그 파일을 분석하였다. 웹 로그 파일의 기간은 4일간이다. 총 요청 수는 64,600여 개로 에러부분을 제외한 컨텐트만을 추출하였다.

컨텐트의 분포를 이해하기 위해 실제 서비스된 웹 사이트를 구성하는 컨텐트의 종류를 조사하였다. 이를 통해 웹 서버에 대한 사용자의 접근 패턴을 알 수 있다. (그림 6)은 컨텐트 종류별로 요청 패턴을 분석한 것이고, (그림 7)은 컨텐트 종류별로 총 전송된 크기 비율로 요청 패턴을 분석하였다.

(그림 6)을 보면 알 수 있듯이 CGI가 가장 많은 41,344개이고, Multimedia는 3,230개로 요청이 가장 적다. (그림 7)은 컨텐트별 총 전송된 비율로 분석하여 보면 멀티미디어 데이터의 경우 요청 수는 미약하나 전송 비율은 높음을 알 수 있다. 이것은 작은 컨텐트들을 많이 요청함으로써 네트워크 트래픽이 증가할 수 있지만, 크기가 큰 컨텐트들은 몇 번만 요청하면 통신량이 많이 증가함을 의미한다.



(그림 6) 컨텐트별 요청 회수



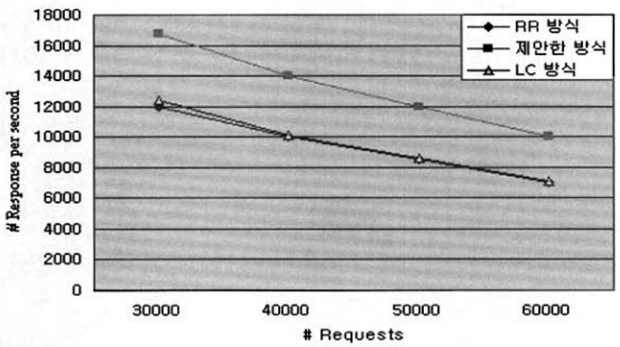
(그림 7) 컨텐트별 총 전송량 비율

지금까지의 웹 로그 부하 분석이 다른 웹 서버의 작업 부하와 정확하게 일치한다고 단정할 수는 없다. 다른 인터넷 서버를 이용하는 사용자의 경향이나 웹 서버의 역할에 따라 다르기 때문이다.

4.2 실험 결과 및 평가

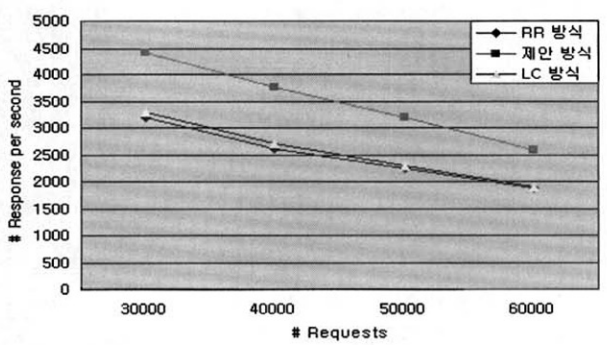
(그림 6)과 (그림 7)의 패턴 통계에 의해서 임의로3만개, 4만개, 5만개, 6만개의 입력 데이터로 구성하여 실험을 하였다.

기존의 RR 스케줄링 방식과 LC 스케줄링 방식, 그리고 제안한 시스템 구조에 대해 응답시간과 요청 수에 대한 초당 응답 수를 비교 실험하였다. 시뮬레이션 결과로 Static HTML, CGI, Multimedia files 각각에 대한 초당 응답 수를 살펴 보았다.



(그림 8) Static HTML의 실험 결과

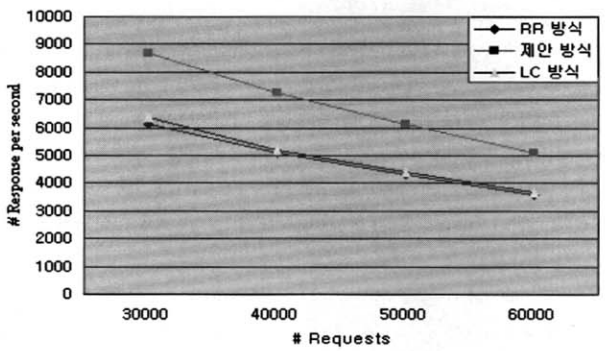
(그림 8)은 Static HTML 컨텐트를 가지고 요청 수에 대한 초당 응답 수를 실험한 결과이다. 처음의 30,000개의 요청 수에 대해 RR방식은 12,000개의 초당 응답 수를 보였고, LC 방식은 12,400개의 초당 응답 수를 보였다. 이에 반해 제안한 방식은 16,780개의 초당 응답 수를 보임으로써 기존의 방식 보다 더 나은 결과를 볼 수 있다. LC 방식의 경우 부하 부산기 서버가 각 웹 서버에 대한 현재 연결 수를 분석해야 하는 오버 헤드가 추가 되어 그 성능이 떨어지게 된다.



(그림 9) CGI의 실험 결과

(그림 9)은 CGI 컨텐트를 가지고 요청 수에 대한 초당 응답 수를 실험한 결과이다. 30,000개의 요청 수에 따라 RR 방식은 3,200개, LC 방식은 3,290개 그리고 제안한 방식은 4,413개의 초당 응답 수를 보였다. CGI 컨텐트는 Static HTML 컨텐트 비해 아주 적은 응답 수를 볼 수 있는데, 이는 CGI 컨텐트가 더 많은 실행 시간을 실행하기 때문이

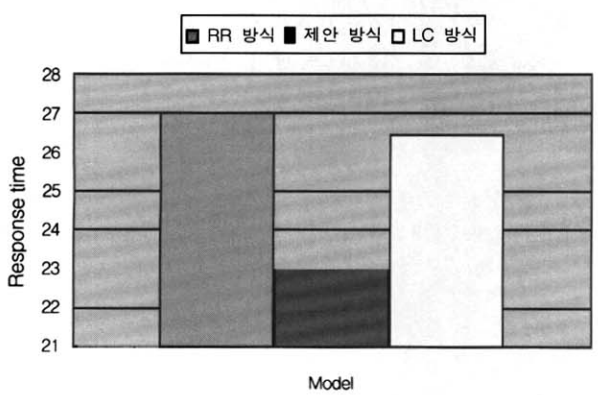
다. Static HTML 컨텐트는 실질적으로 문서(text, html 등)를 보여주지만, CGI 컨텐트는 DB 관련 등을 처리하기 때문에 많은 프로세스가 요구된다.



(그림 10) Multimedia files의 실험 결과

(그림 8), (그림 9), (그림 10)에서 보는 거와 같이 기존의 RR 스케줄링 방식과 LC스케줄링 방식은 거의 비슷하게 나타났고, 본 논문에서 제안한 내용 기반으로 한 웹 서버 클러스터 기법이 약 39%의 성능 향상이 되는 것을 알 수 있었다. 전체적으로 제안한 웹 서버 클러스터 기법이 기존의 RR 스케줄링 방식과 LC 스케줄링 방식 보다 성능면에서 월등히 향상되었음을 알 수 있었는데, 이러한 이유는 오래 걸리는 클라이언트 요청으로 인해 짧은 요청이 기다려야 하는 기존의 방식보다 내용으로 한 웹 서버 클러스터 기법이 해결 했기 때문이다.

다른 평가 단위로써 응답시간을 측정하였다. 이를 위해 실험 시간 동안 클라이언트 요청에 대한 응답시간의 평균을 구현하였고 결과는 (그림 11)과 같다.



(그림 11) 지수 분포 모델

(그림 11)과 같이 응답시간에서도 지수 분포 모델로 실험했을 때 기존의 방식(RR)은 평균 2.7초, LC 방식은 2.65초에 반해 제안한 웹 서버 클러스터 기법은 2.3초로 각각 16%와 14% 정도로 더 짧다는 것을 알 수 있었다.

5. 결 론

웹 서버(실행 서버)간의 서로 다른 컨텐트를 가지고 있는 즉, 내용 기반으로 한 웹 서버 클러스터 기법을 소개하였다. 웹 서버 클러스터는 저 비용으로 사용자에게 신뢰성 있고 안정된 서비스를 제공해준다는 점에서 이미 많은 연구가 진행되고 있고 특히 클라이언트의 요청을 정확하고 빠르게 응답해주는 방안이 중요시되고 있다. 본 논문에서는 웹 서버(실행 서버)마다 서로 다른 컨텐트를 갖는 내용 기반으로 한 웹 서버 클러스터를 설계하여 그 성능이 실험을 통해 제안한 기법이 기존 방식보다 약 39%의 성능 향상이 되는 것을 알 수 있었다. 또한 응답시간에서도 지수 분포 모델로 실험했을 때 기존의 RR방식은 평균 2.7초에 반해 제안한 웹 서버 클러스터 기법은 2.3초로 16%정도로 더 짧다는 것을 보였다. 제안한 웹 서버 클러스터를 이용한 웹 클러스터 기법은 기존의 솔루션에 비해 다음과 같은 장점이 있다.

첫 번째, 제안한 웹 서버 클러스터 구조는 각각의 웹 서버가 서로 다른 컨텐트를 가지고 있으므로 부하 분산기에 대한 오버헤드가 적어 전체 웹 서버 클러스터 성능을 효율적으로 운용할 수 있다. 두 번째, 사용자의 요청에 대한 응답이 부하 분산기를 거치지 않고 해당 웹 서버에서 바로 사용자에게 전달되기 때문에 응답 시간이 감소 된다. 세 번째, 각 웹 서버들은 서로 다른 컨텐트를 가지고 있어 짧은 응답을 요구하는 사용자는 빠르게 응답을 받게 된다.

향후 서버 규모를 증가할 때 더 큰 성능의 향상을 얻도록 개선되어야 하며, 부하 분산기 서버가 다운되었을 때 서비스가 마비될 수 있다는 점이다. 따라서 앞으로의 연구 과제로는 주 부하 분산기 서버가 다운되었을 때 전환 가능한 보조 부하 분산기 서버를 구현하여 신뢰성을 더하고, 또한 특정한 웹 서버에서 과부하가 발생하였을 때 해당 서버의 이주를 통해 요청을 분산하는 연구도 진행되어야 할 것으로 생각된다.

참 고 문 헌

[1] Tim Bray, "Measuring the Web," In Proceedings of the Fifth International World Wide Web Conference, Paris, France, pp.993-1005, May, 1996.

[2] Allison Woodruff and Paul M. Aoki and Eric Brewer and Paul Gauthier and Lawrence A. Rowe, "An Investigation of Documents from the WWW," In Proceedings of the Fifth International WWW Conference, Paris, France, pp. 963-979, May, 1996.

[3] Kangasharju and J. Ross and K. W., "A clustering structure for reliable multicasting," Computer Communications and Networks, 1999. Proceedings, Eight International Conference, pp.378-383, 1999.

[4] Kangasharju, J. and Ross, K. W., "A replicated architecture for the Domain Name System," INFOCOM 2000. Nineteenth

Annual Joint Conference of the IEEE computer and Communications Societies, Proceedings, IEEE, Vol.2, pp.660-669, 2000.

[5] Dongeun Kim and Cheol Ho Park and Deayeon Park, "Request rate adaptive dispatching architecture for scalable Internet server," Cluster Computing, 2000. Proceedings. IEEE International Conference, pp.289-296, 2000.

[6] Canal, R. and Parcerisa, J. M. Conzalez and A., "Dynamic cluster assignment mechanisms," High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium, pp.133-142, 1999.

[7] Beowulf Project, http://www.beowulf.org.

[8] A. Wong and T. Dillon, "Load balancing to Improve Dependability and Performance for Program Objects in Distributed Real-time Cooperation over the Internet," The 3rd IEEE International Symposium on Object-Oriented Real-time Distributed Computing, Mar., 2000.

[9] V. Carellini and M. Cloajanni and P. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-server Systems," Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, pp. 528-535, May, 1999.

[10] M. Colajanni, P. S Yu, V. Cardellini, "Dynamic bad balancing in geographically distributed heterogeneous Web-servers," Proc. of 18th IEEE Int'l,. Conf. on Distributed Computing Systems (ICDCS '98), Amsterdam, The Netherlands, pp.295-302, May, 1998.

[11] V. Cardellini and M. Colajanni, P. Yu, "Geographic Load Balancing for Scalable Distributed Web Systems," Modeling, Analysis and Simulation of Computer and Telecommunication System, 2000, Proceedings 8th International Symposium on, 2000.

[12] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable Web server clustering technologies," IEEE network, pp. 38-45, May, 2000.

[13] Vivek S. Pai, Mohit Aron, Gaurav Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, Erich Nahum, "Locality-Aware Request Distribution in Cluster-based Network Services," ACM 8th, ASPLOS Oct., 1998.

[14] Chu-Sing Yang and Mon-Yen Luo, "Efficient Support for Content-Based Routing in Web Server Cluster," Proceedings of the 2th USENIX Symposium in Internet Technologies and Systems, Boulder, Colorado, USA, October, 1999.

[15] Morika, M., Kurosawa, K., Miura, S., Nakamikawa, T., Ishikawa, S., "Design and evaluation of the high performance multi-processor server," Computer Design : VLSI in Computers and Processors, 1994. ICCD '94. Proceedings., IEEE International Conference, pp.66-69, 1994.

[16] Linux Virtual Server Project, http://www.linuxvirtualserver.org.

[17] Wensong Zhang and Shiyao Jin and Quanyuan Wu, "Creating Linux Virtual servers," Ottawa Linux symposium 2000, 2000.

[18] Joseph Mack and Wensong Zhang, "The Linux Virtual Server HOWTO," http://www.linuxvirtualserver.org/Joseph.Mack/LVS-HOWTO-991205.gz, 1999.

명 원 식

e-mail : wsmyung@dgu.edu

1996년 안양대학교 전자계산학과(학사)

1998년 안양대학교 대학원 전산정보학과 (공학석사)

2000년~현재 동국대학교 컴퓨터공학과 대학원(박사과정)

관심분야 : 클러스터 컴퓨팅, 부하 분산, 온라인 게임, 병렬컴퓨터구조

장 태 무

e-mail : jtm@dgu.edu

1977년 서울대학교 전자공학과(학사)

1979년 KAIST 전산학과(석사)

1998년 미국 University of South West Louisiana 교환교수

1996년 서울대학교 컴퓨터공학과(박사)

1981년~현재 동국대학교 컴퓨터공학과 교수

관심분야 : 병렬컴퓨터구조, 분산처리, Embedded System, Grid