차별화된 웹 서비스 시스템의 설계와 구현

이명섭° 김환섭 신경철 박창현
영남대학교 컴퓨터공학과
{skydream°, kcsin, hskim, park}@cse.yu.ac.kr

Design and Implementation of a Differentiated Web Service System

Myungsub Lee° Kyungchul Sin Hwansub Kim Changhyeon Park
Dept. of Computer Engineering, Yeungnam University

요 약

최근 들어, 인터넷 사용자의 폭발적인 증가로 인하여 차별화된 웹 서비스를 제공해주는 웹 응용프로그램들의 개발이 활발해지고 있다. 이에 따라 웹 서버내의 품질향상을 보장해주는 웹 QoS 기술은 전자상거래나 웹 호스팅 같은 부분에서 점점 더 중요한 문제로 대두되고 있다. 그러나, 대부분의 웹 서버들은 FIFO 방식의 최선 서비스만을 제공하고 있으며, 정보의 중요도나 정보를 제공받는 사용자의 중요도에 따라 차별화된 품질보장을 제공하지 못한다. 본 논문에서는 웹 서비스의 차별화된 품질보장을 제공하는 웹 서버 구현을 위한 두 가지 접근 방식을 제시한다. 첫째는 커널 수준 접근방법으로, 커널상에 실시간 스케줄링 프로세서를 두어 웹 서버에서 수행중인 스케줄링 프로세서와 연동시켜 커널 내부에서도 웹 서버에서 할당된 사용자 요청 우선순위를 유지하도록 한다. 둘째는 부하분산 접근방법으로, IP 수준의 가장법과 터널링 기술을 이용하여 웹 서버의 부하를 분산하여 웹 서비스의 신뢰성을 보장하고 응답속도를 개선한다.

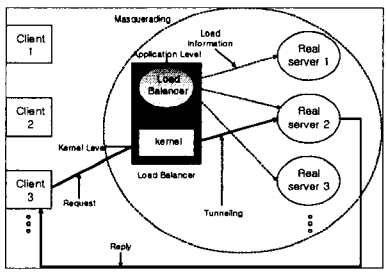
1. 서 론

World Wide Web(이하 웹)은 저렴한 가격과 다양하고 흥미 있는 정보를 쉽고 간편하게 찾아볼 수 있다는 장점으로 웹 사용자는 빠르게 증가되고 있으며 웹 사용자의 증가와 함께 웹을 통해 전달되는 데이터 즉, 웹 문서, 그림, 멀티미디어 데이터 등의 크기 또한 빠르게 증가되고 있다[1]. 웹의 급격한 보급과 각종 멀티미디어 데이터를 포함하는 웹 서비스의 팽창으로 인하여 웹 서버 내의 서비스 품질보장을 위한 웹 QoS(Quality of Service)기술은 웹 서비스[2] 부분에서 점점 더 중요한 문제로 대두되고 있다. 차별화된 웹 서비스를 제공하기 위해서는 정보의 중요도와 정보를 제공받는 사용자의 중요도에 따라 컨텐츠를 분류하는 모듈과 분류된 컨텐츠를 스케줄링할 수 있는 모듈이 웹 서버에 포함 되어 있어야 한다. 그러나 대부분의 인터넷 서버들은 이러한 컨텐츠의 종류와는 무관하게 단지 FIFO(First In First Out)방식의 스케줄링으로 최선 서비스(best effort service) 만을 제공하고 있어서, 서버의 과부하시 중요 사용자(premium user)의 서비스를 제공하지 못한다는 단점을 가지고 있다[4]. 일반적으로 가장 많이 사용되고 있는 아파치 웹 서버[3]의 경우, 웹 서버상의 요청 형태를 알고 있음에도 FIFO 방식으로 요청을 처리하고 있다. 따라서 서비스 품질을 보장하고 서비스를 특정 분류 기준에 따라 분류하여 차별화 서비스를 제공하기 위한 새로운 서버 개발은 반드시 필요하다. 웹 사용량은 폭발적으로 증가하고 있으나, 이런 수요 증가에 비해 제공되는 웹 서버의 서비스 성능 부족으로 인해 차별화 서비스를 제공하는 웹 서버를 구축 하더라도 완전한 서비스 품질을 보장하지 못한다. 이를 위해 부하 분산형 웹 서버가 중요한 해결수단으로 제시 되고 있다[5]. 부하 분산형 웹 서버는 급격히 늘어나는 서비스/접속요청에 신속한 응답을 제공하고 안정적이고 유연성 있는 서비스를 제공함으로써 서비스 품질보장을 위해 반드시 필요하다. 그러나, 기존연구에서 제시하는 부하분산 웹 서버 기술은 클라이언트의 응용프로그램 변경, 서버 과부하 처리 불능, HTTP 요청/응답 처리의 과부하, 패킷 변환 오버헤드 등의 문제점이 있다.

본 논문에서는 웹 서비스의 차별화된 품질보장을 제공하는 부하 분산형 웹 서버 구현을 위한 두 가지 접근 방식을 제시한다. 첫째, 웹 서버 상에는 클라이언트 요청 정보의 중요도에 따라 우선순위를 부여할 수 있는 스케줄링 모듈을 추가하고, OS 커널 상에는 이와 연동되는 실시간 스케줄러를 두어 웹 서버에서 부여된 우선순위가 커널 내부의 프로세서에도 그대로 유지될 수 있도록 함으로써 더 효율적인 차별화 서비스를 제공할 수 있도록 한다. 둘째, IP 가장법(masquerading)과 터널링(tunneling)기술을 이용한 분산 웹 서버를 구성하여 웹 서버의 부하를 클래스별로 분산함으로써 웹 서비스의 신뢰성과 응답속도를 개선할 수 있도록 한다.

2. 제안 시스템

본 논문에서 제안하는 차별화된 웹 서비스를 제공하는 부하 분산형 웹서버는 클라이언트의 요청에 우선순위를 제공하기 위하여 웹 서버에서 컨텐츠를 분류하고 스케줄링을 수행하며, 커널 수준의 실시간 스케줄러에게 프로세스를 맵핑하는 커널 수준 접근법을 지원하며, 웹 서비스의 신뢰성보장과 응답속도를 개선하기 위하여 IP 수준의 가장법과 터널링 기술을 이용하여 웹 서버의 부하를 분산하는 시스템이다. 본 논문에서 제시한는 차별화된 웹 서비스 시스템의 전반적인 구성은 [그림 1]에서 보이고 있으며, 자세한 구조 및 동작에 대한 설명은 아래의 각 절에서 주어진다.

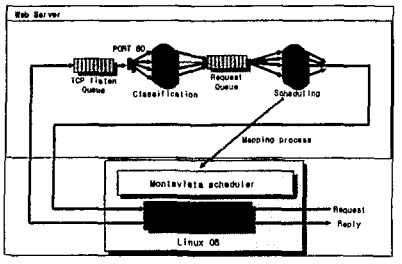


[그림 1] 시스템 전체구성도

2.1 커널 수준 접근법

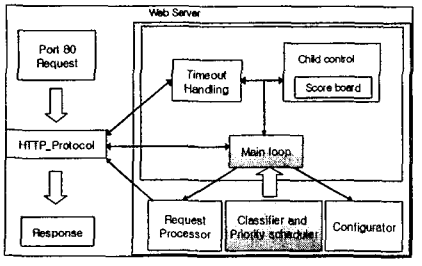
커널 수준 접근 방식은 하나의 요청에 대한 효율적인 우선순위를 보장하기 위한 것으로, 아파치 웹 서버의 스케줄러가

커널내부의 실시간 스케줄러에게 프로세스를 맵핑하는 방법으로 시스템을 구현한다. [그림 2]에서 커널 수준 접근법의 구조를, [그림 3]에서 수정된 웹 서버의 구성도를 보인다.



[그림 2] 커널 수준 접근법의 구조

[그림 2]에서, 클라이언트의 요청이 NIC (Network Interface Card)를 통해서 들어오면 웹 서버는 TCP listen 버퍼에서 80번 포트로 요청을 받아들이고, 연결 관리자에서 분류 정책(클라이언트 IP, URL, 파일이름, 디렉토리, 사용자 인증 등)에 따라 각 연결별로 분류를 한다. 분류된 요청은 우선 순위를 부여하여 각각의 요청 큐에 입력되고 DLC 스케줄링 정책에 따라 스케줄링 된다. 이때, 웹 서버에서 스케줄링을 수행하는 각 프로세서들은 커널 내의 실시간 프로세서와 1:1로 연결되어 수행한 후 클라이언트에게 응답 패킷을 전송한다.



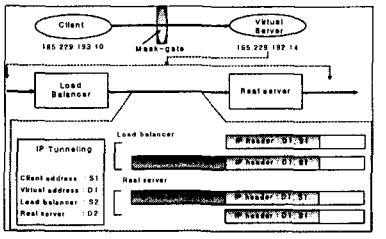
[그림 3] 수정된 웹 서버 구성도

[그림 3]은 수정된 아파치 웹서버의 구성도이며, 수정된 아파치 웹 서버를 시작하게 되면 마스터 프로세서가 생성되고 이 프로세서는 prim-level, high-level, default 클래스를 가진 자식 프로세서를 생성하여 httpd 데몬을 구성한다. prim-level 클래스와 high-level 클래스는 실시간 스케줄링을 수행하도록 하며, 기본(default) 클래스는 일반 커널 스케줄러로 수행하도록 구성한다. 실시간으로 프로세서를 실행시키기 위하여 shched_setscheduler 시스템 콜 함수를 사용하며, 이 시스템 콜 함수는 프로세스 ID, policy(FIFO, Round-Robin), 우선순위와 같은 세 개의 인자를 가진다. 아파치 웹 서버의 conf 디렉토리 내의 class_ip.conf 파일은 클래스별로 파일 이름을 저장하고 있다. 그래서 아파치 웹 서버가 실행될 때 initialize_board라는 함수가 이 파일을 읽고 ctrl_levelizer 구조체에 파일 이름을 저장한다. 이 구조체는 파일 이름별로 우선순위 정보를 가지고 있다. 부모 프로세스가 생성(fork)시킨 자식프로세스가 이 구조체에 빠르고 쉬게 접근하게 하기 위해 본 논문에서는 공유 메모리 기법을 사용한다.

2.2 부하분산 접근법

본 논문에서 제시하는 부하분산형 웹 서버는 IP 수준의 가장법과 터널링을 이용하여 패킷 전송 속도를 향상시키고 부하 제어기의 병목 현상을 제거함으로써 성능을 높이고 확장이 쉽다. 이는 1대의 부하 제어기가 클라이언트들로 부터의 서비스 요청을 각 실행 서버로 분산시켜주는 부하분산형 구조로 구성되고, 가장법을 이용하여 외부에서는 마치 단일 IP 주소를 가진 한대의 서버처럼 보이게 된다. 부하 제어기의 구조는 커널 수준과 IP 수준으로 구성되며 각각의 구조와 동작방식은 다음의 세부 절에서 기술한다.

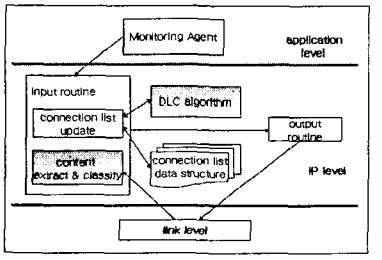
IP 가장법은 여러 컴퓨터들(분산 서버들)을 외부 네트워크(또는 인터넷)상에서 게이트웨이 역할을 하는 오직 하나의 컴퓨터(가상 서버) 뒤에 숨겨져 동작될 수 있도록 한다.



[그림 4] 커널 수준의 구조

[그림 4]에서 165.229.193.10이라는 실제 IP를 사용하여 클라이언트의 요청이 들어오면 mask-gate에서 접속정보를 변환하여 패킷을 전달함으로써 내부 네트워크 상의 컴퓨터들은 인터넷을 통할 수 있으나, 클라이언트는 내부 서버들에 대한 접속정보를 알 수 없게 된다. 터널링 기법은 외부에 공개된 실제 IP 주소를 가진 IP 패킷 헤더에 내부 각 서버의 가상 IP 주소를 새로 추가하는 캡슐화(encapsulation)과정과 그 역과정인 역 캡슐화(decapsulation)과정을 수행한다. 이를 통하여 패킷을 받은 서버들이 다시 주소를 변환하지 않고도 요청 패킷의 IP 주소를 이용하여 직접 외부 네트워크로 데이터를 전송한다.

IP 수준은 컨텐츠 추출 모듈, 분류 모듈, DLC 알고리즘으로 구성된다.



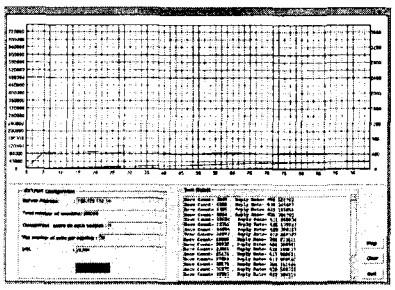
[그림 5] IP 수준의 부하제어기 구성도

[그림 5]에서 IP 수준의 부하제어기의 구성도를 보인다. [그림5]에서 클라이언트로부터의 요청이 들어오면 IP 계층의 컨텐츠 추출과 분류 모듈에서 분류와 스케줄링을 수행한다. IP 계층에서 수신된 요청 메시지를 다루기 위해서 본 논문에서는 리눅스 커널의 자료구조인 sk_buff를 이용하며, 이 자료구조의 포인터를 이용하여 HTTP 요청데이터의 패스(path)를 읽어온다.

3. 구현 및 실험

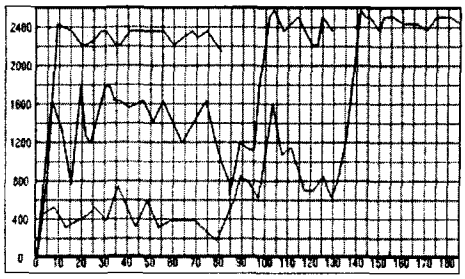
본 논문에서 제시한 차별화된 웹 서비스 시스템은 Pentium-III 800MHz, 256MB 사양의 PC와 Linux Kernel 2.4.7 운영체제 상에서 구현되었으며, 시스템의 동작을 실험하기 위해서 클라이언트 3대, 부하 제어기 1대, 서버 2대, 모니터링 서버 1대를 네트워크로 연결한 실험환경을 구성하였다. 웹 서버는 Apache Web Server 2.4.17을 개조하여 이용하고 있으며, Linux 커널에는 Montavista 실시간 스케줄러[6]를 접목하였다.

클라이언트 인터페이스는 사용자가 특정 웹 서버의 웹 페이지에 시간 단위와 전송량을 설정하여 요청을 보냄으로써 웹 서버의 수행 능력을 테스트할 수 있는 모듈이다. 각 클라이언트에는 이러한 모듈이 있으며, 이를 이용하여 각 클라이언트는 부하제어기를 통하여 실행서버에 요청을 보낸다. 본 논문에서는 클라이언트 요청에 대한 서버의 응답율을 GUI형태로 제공하기 위하여 각 클라이언트와 모니터링 에이전트부분에 클라이언트 인터페이스 모듈을 설치한다. [그림 6]에서 본 논문의 클라이언트 인터페이스 화면을 보이고 있는데, 왼쪽 하단의 창은 테스트 환경을 설정할 수 있는 부분으로서, 'Server Address' 는 요청을 보내고자 하는 가상 서버 주소를, 'Total number of seesions'은 연결하는 총 연결 개수를 나타낸다. 'Concurrent users on each session'은 한번에 동시 접속할 수 있는 접속자 수를 나타내며, 'The number of calls per session'은 한 연결 당 요청하는 세션 수를 나타낸다.



[그림 6] 클라이언트 인터페이스

본 논문에서 수행한 실험은 서버에 과부하가 발생하지 않을 경우와 서버에 과부하가 발생했을 경우 그리고 서버의 과부하 시 클래스별 요청이 중단되었을 경우로 나누어서 수행하며 서버에 과부하가 발생하고 클래스별 요청이 중단 되었을 경우를 위한 실험 결과만 보인다. 가상 IP는 165.229.192.140이고 전체 연결 개수는 50000, 동시 접속자는 30명, 한 연결 당 요청 개수는 50으로 실험한다. [그림 7]을 보면 모든 클래스의 요구가 지속적으로 발생하다가 80초의 시간이 흐른 뒤 a.html을 요구하는 클라이언트의 요구가 중단 되었을 때 b.html과 c.html을 요구한 응답율이 증가하는 것을 볼 수 있다. A 클래스의 경우 가장 우선순위가 높은 클래스, 클래스 A에 해당하는 요구가 중단되었기 때문에 b.html의 요구가 가장 우선순위가 높다. 이때 130초의 시간이 흐른 뒤 b.html을 요구한 클라이언트의 요구가 중단 되었을 때 다시 c.html의 응답율이 증가하는 것을 그래프로 볼 수 있다.



[그림 12] 클래스별 요청 중단 시 그래프

4. 결론

본 논문에서는 정보의 중요도나 정보 사용자의 중요도에 따라 서비스를 제공할 수 있는 차별화된 웹 서비스 시스템의 구현을 위한 두 가지 접근 방법, 즉, 커널 수준 접근 방법과 부하 분산 접근 방법을 제시하였다. 커널 수준 접근방법에서는 커널 내부에 실시간 스케줄링 프로세서를 접목하였으며, 부하 분산 접근방법에서는 IP 수준의 가장법과 터널링 기법을 이용한 부하제어기를 구현하였다. 본 논문의 부하제어기는 기존의 LC 알고리즘을 개선한 DLC 알고리즘을 제시하여 서비스 요청의 우선순위에 따라 차별화 된 웹 서비스를 제공할 수 있도록 하였다. 본 논문에서 구현한 부하 분산 시스템의 동작을 세 가지 경우에 대해서 실험하였으며, 이 실험에서 차별된 웹 서비스를 지원하고 있다는 것을 보였다. 본 논문의 DLC 알고리즘은 LC 알고리즘과 마찬가지로 정적으로 동작됨으로 각 실행 서버의 동적인 부하변화를 반영할 수는 없다. 따라서, 각 실행 서버의 CPU 모니터링 및 서버 상태 분석 등을 통한 서버 부하 정도를 고려한 동적인 분산 서비스 시스템에 대한 연구는 계속 진행 중이다.

참고문헌

[1] R.Fielding, J. Getys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", IETF, January 1997.

[2] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User Perceived Quality into Web Server Design", Proc. Of the 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000, pp.92-115.

[3] Apache Group, http://www.apache.org.

[4] R. Bhatti and R. Friedrich., "Web Server Support for Tiered Services.", IEEE Network, September/October 1999, pp.64-71.

[5] Wensong Zhang, "Linux Virtual Server Project",http://proxy.iinchina.net/ ~wensong/ippfvs/,May 1998.

[6] Montavista Software, http://www.monta -vista.com