압축 기능을 가진 웹캐시 시스템 개발

박 진 원† · 김 명 균†† · 홍 윤 환†††

요 약

인터넷 사용자의 수가 폭발적으로 늘어나고 웹 컨텐츠의 양이 많아지면서 웹서버의 부하를 줄이면서 사용자에게 보다 빠르게 웹서비스를 제공하는 것이 중요한 문제가 되었다. 웹캐시 시스템은 사용자와 웹서버 사이에 위치하여 웹서버의 부하를 줄여주고 사용자의 서비스 응답시간을 줄여주기 위한 효과적인 방법으로 많이 사용되고 있다. 본 논문에서는 소스가 공개되어 있는 스퀴드 웹캐시 시스템을 이용하여 데이터 압축 기능을 가진 웹캐시 시스템을 개발하였다. 압축기능을 가진 웹캐시 시스템은 웹캐시 시스템과 사용자 사이의 데이터를 압축하여 전송함으로써 네트워크의 트래픽을 줄여주고 사용자의 웹서비스 응답시간을 줄여주는 효과가 있다. 특히 리버스 캐시 시스템의 경우에는 종단간 병목구간인 인터넷 구간의 트래픽 양을 줄여주므로 그 효과는 더욱 크게 나타난다. 성능측정 결과 원래 웹 객체의 크기에 따라 압축율은 달라지지만 2배에서 크게는 8배 정도까지의 데이터 전송량이 줄었으며 사용자의 응답시간은 평균 37% 정도의 속도 향상이 있었으며 원래 데이터의 크기가 10Kbyte 이상인 경우에서는 평균 속도 향상율이 87% 정도로 향상 정도가 더욱 크다는 것을 알 수 있었다.

Development of A Web-cache System with Compression Capability

Zin-Won Park† · Myung-Kyun Kim†† · Yoon-Hwan Hong†††

ABSTRACT

As the number of Internet users and the amount of web contents have increased very fast, reducing the load of web servers and providing web services more rapidly have been great issues. A web-cache system, which is located between the user and the web server, has been used by many web service providers as an effective way to reduce the load of web servers and the web service response time. In this paper, we have developed a web-cache system which is based on the Squid cache and has a compression capability. The web-cache system in which compression capability reduces the amount of network traffic and the web service response time by transfering the web contents in the compressed format over the network between the web-cache system and the user. The performance enhancement is greater in the reverse-cache system than in the forward-cache system because in the case of the reverse-cache system, the cache reduces the amount of traffic on the Internet which is the bottleneck in the network path between the user and the web server. The experimentation result shows that the amount of data traffic has reduced from 2 to 8 times depending on the size of the web contents. The web server response time has reduced 37% on the average and when the size of the web content is greater than 10Kbyte, the response time has reduced 87% on the average.

키워드 : 웹캐시(Web Cache), 리버스 캐시 시스템(Reverse Cache System), 웹서비스(Web-Service), 통합관리 시스템(Integrated Cache Management System)

1. 서 론

인터넷 사용자의 수가 급등하면서 대부분의 정보들이 인터넷을 통하여 검색되어지고 있다. 비단 정보뿐만 아니라 각종 서비스 역시 인터넷을 통한 웹 기반으로 이루어지고 있어 대부분의 인터넷 사용자들은 웹을 통하여 정보를 주고받고 있는 실정이다. 이렇게 웹 기반으로 이동하고 있는 정보의 양이 많아짐으로 해서 웹을 사용하는 사용자의 입장에서나 웹을 통한 정보 제공자의 입장에서 빠른 웹 서비스 이용과 안정적인 서비스 제공을 위해 웹캐시 시스템을 도입하고 있다[1, 2].

웹서비스 제공에 있어서 빠른 정보 전달은 매우 중요한 요소이다. 사용자는 웹 사이트에 접속하여 수초내에 웹페이지를 열람할 수 없다면 그 웹사이트를 떠난다는 연구결과도 나와 있다. 따라서 서비스 제공자는 보다 빠르게 정보를 전달하기 위하여 네트웍의 대역폭을 높이고 고성능의 웹서비스 시스템을 도입하고 있다. 하지만 서비스 제공자의 대역폭이 높아지고 시스템의 성능을 개선된다고 하더라도 인터넷에는 많은 데이터가 이동하고 있을 뿐만 아니라 최종 사용자의 네트워크 대역폭에는 한계를 가지고 있다. 이런 환경속에 더 빠른 정보 전달을 위하여 HTTP는 컨텐츠 압축을 허용하고 있다. 컨텐츠를 압축하면 데이터의 크기가 줄어들게 되고 그만큼 빠른 전달이 가능해진다. 따라서 웹캐시 시스템이 컨텐츠를 압축하여 사용자에게 전달한다면 웹서버에는 부하를 주지 않고 보다 빠른 웹서비스가 가능해진다.

많은 이용자를 가진 대학이나 기업, 포털 사이트 같은 동시 접속자의 수가 수만에 이르는 대형 웹사이트에서는 안정적이고 원활한 웹 서비스를 위해 웹캐시 시스템이 필수가

※ 본 연구는 한국과학재단 지정 울산대학교 네트워크 기반 자동화연구센터 및 울산대학교 교내 연구비의 지원에 의한 것입니다.
† 준 회 원 : 울산대학교 대학원 컴퓨터 · 정보통신공학부
†† 정 회 원 : 울산대학교 컴퓨터 · 정보통신공학부 교수
††† 정 회 원 : (주)닥터소프트 대표이사
논문접수 : 2003년 10월 16일, 심사완료 : 2003년 12월 26일

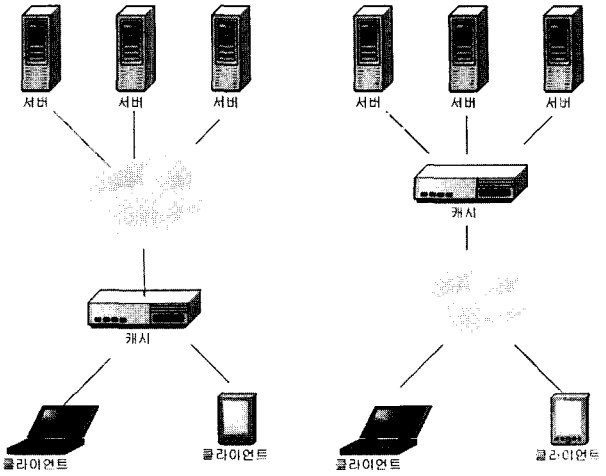
되어 가고 있다. 뿐만 아니라 캐시 시스템의 부하 분산을 위하여 여러 대의 웹캐시가 필요하게 된다. 그리고 웹캐시 시스템의 효율적인 관리와 지속적인 모니터링은 캐시 시스템 운영에 필수 요소이다[3]. 복수의 웹캐시 시스템은 관리와 모니터링을 하기 위해 각 시스템의 관리 장비를 따로 두어야 했다. 웹캐시의 수에 비례하는 웹캐시 모니터링 장비는 관리와 효율의 측면에서 많은 문제점을 가지고 있다.

본 논문에서는 웹서버의 부하를 줄이고 정보의 전달 속도를 향상시키기 위하여 컨텐츠를 압축하여 전달하는 웹캐시 시스템을 설계, 구현하였으며 복수의 웹캐시 환경에서 통합된 관리 툴을 이용하여 여러 대의 웹캐시 시스템을 관리하고 모니터링 할 수 있는 웹 기반의 통합 관리 시스템을 설계하고 구현하였다. 웹캐시 시스템은 클라이언트와 웹서버 사이에 위치하여 클라이언트에서 접근했던 웹객체들을 캐시에 저장하고 있다가 클라이언트 웹서비스 요청시 해당 객체를 로컬캐시에서 찾아보고 있을 경우 캐시의 데이터를 보내주어 웹서버의 부하를 줄여주고 사용자의 응답시간을 줄여주게 된다. 본 논문에서는 소스코드가 공개되어 있는 스퀴드(Squid) 캐시를 이용하여 압축기능을 가진 웹캐시 시스템을 개발하였다. 웹캐시 시스템에서 데이터를 압축하여 클라이언트에게 전송하게되면 캐시와 클라이언트 사이의 네트워크 트래픽을 줄여주게 되어 사용자의 응답시간을 줄이고 네트워크의 부담을 줄여 줄 수 있는 효과가 있다. 이러한 웹캐시 시스템을 리버스 캐시에 적용할 경우에는 인터넷과 사용자 네트워크 구간에서의 네트워크 트래픽을 줄여 주게 되므로 더욱 효과가 크다. 따라서 본 연구에서 개발한 압축기능을 가진 웹캐시 시스템은 일차적으로 리버스 캐시에서 사용될 목적으로 개발되었지만 경우에 따라서 포워드캐시에서도 사용될 수 있을 것이다.

본 논문의 순서로는 2장에서 웹캐시 시스템과 캐시 시스템의 구성, 그리고 복수의 웹캐시 시스템 구성 환경에 대한 배경 연구를 살펴 본 다음, 3장에서는 본 논문의 캐시 시스템의 압축 기능에 대해서 알아보고, 4장에서는 웹 기만 통합 관리 시스템의 구성 및 구현 방법을 설명한 후 5장에서는 결론을 맺고 및 향후 과제에 대해서 알아본다.

2. 배경 연구

웹캐시 시스템은 사용자와 웹서버 사이에 위치하여 사용자들이 접근했던 웹페이지들을 일정한 시간 동안 캐시에 저장해 두었다가 사용자의 요청이 있을 경우 웹서버 대신에 정보를 서비스 해 주는 역할을 한다. 현재 캐시 시스템은 종류에 따라 HTTP 뿐만 아니라 FTP, Gopher 등 다양한 정보를 캐싱 해 주고 있다. 그러나 대부분의 캐시들이 HTTP 캐싱을 위해 사용되고 있는 점을 고려하여 본 논문에서는 웹캐시 시스템을 중점으로 다룬다. 하지만 HTTP 캐시뿐만 아니라 다른 캐시 서버 역시 구성이나 작동 방식은 웹캐시와 유사하다. 웹캐시 시스템은 설치 방식과 목적에 따라 포워드 캐시(forward cache)와 리버스 캐시(reverse cache)로 나눌 수 있다.



(a) 포워드 캐시 (b) 리버스 캐시

(그림 1) 웹캐시 시스템의 구조

2.1 포워드 캐시(Forward Cache)

포워드 캐시는 (그림 1)(a)와 같이 클라이언트 측에 위치한다. 이 캐시의 역할은 클라이언트들의 요청을 받으면 자신의 저장 장치를 검색하여 해당 정보가 없을 경우 웹서버에게 요청을 보내고 웹서버가 보내온 정보를 저장한 다음 클라이언트에게 정보를 전달해 준다. 여기서 한 번 저장된 정보는 일정 시간동안 계속 캐시 서버에 남아 있게 되는데 정보가 저장되어 있는 시간은 캐시 서버의 정책으로 정하거나 웹 페이지 작성자가 명시적으로 지정할 수 있다[4].

포워드 캐시의 역할은 클라이언트에게 제공했던 정보들을 캐시 서버가 저장해 두었다가 다음 번 클라이언트가 요청을 해 오면 캐시된 정보를 서비스 해 주는 것이다. 이 방식의 캐시 서버는 불특정 다수의 웹서버 정보를 저장하고 있게 되며 특정 클라이언트들의 요청만을 받는다는 특징이 있다. 또한 정보가 인터넷 구간을 거쳐오는 동안의 지연시간을 없애줌으로써 웹 페이지의 로딩(loading) 속도가 빨라지고 클라이언트 측 백본 네트웍의 트래픽(backbone traffic)을 줄일 수 있다는 장점이 있다.

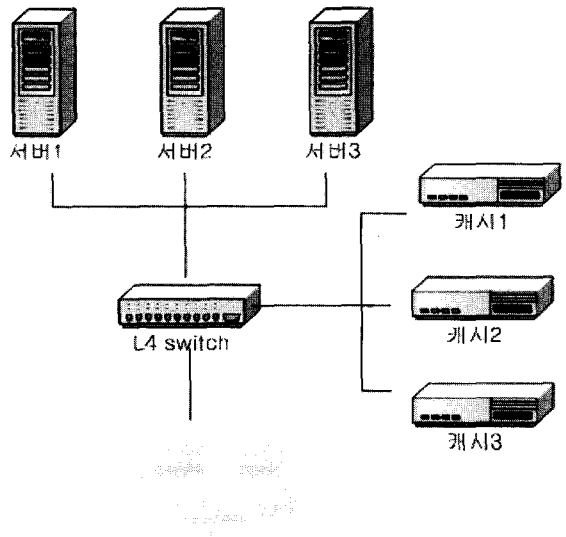
2.2 리버스 캐시(Reverse Cache)

리버스 캐시는 (그림 1)(b)와 같이 웹서버 측에 위치한다. 이 방식은 포워드 캐시와 같이 클라이언트의 요청을 캐시가 받아서 웹 서버를 대신하여 클라이언트에게 서비스한다. 저장시간은 포워드 캐시와 같은 방법으로 결정된다.

리버스 캐시는 불특정 다수의 클라이언트의 요청을 받게 되며 특정 서버의 정보만을 캐싱하는 특징이 있다. 이 방식은 수많은 클라이언트의 요청에 웹 서버가 다운되거나 서비스 속도가 느려지는 것을 막아 주고 서버의 부하를 줄이는데 목적이 있다[5]. 뿐만 아니라 클라이언트가 서버에 직

접적으로 접속하지 않기 때문에 직접적인 공격으로부터 서버를 보호하는 역할도 한다.

동시 접속자 수가 많은 웹 사이트의 경우 (그림 2)와 같이 웹서버를 여러 대 두기도 한다. 이렇게 하는 이유는 한 서버에 많은 접속자가 동시에 접속했을 경우 웹 서버가 다운되거나 혹은 서비스를 하는 속도가 매우 느려지기 때문이다. 늘어나는 사용자 수에 대처하기 위해 서버를 늘리는 것은 많은 비용을 필요로 한다. 이에 반해 캐시를 사용하게 되면 웹서버의 부하를 경감시키게 되어 비용대 효과 면에서 훨씬 우수하다. 따라서 많은 대형 웹사이트에서는 많은 수의 서버와 캐시들을 두어 부하를 분산시키고 확장성을 용이하게 하고 있다.



(그림 2) 복수의 웹캐시 시스템 구조

(그림 2)는 이와 같이 복수개의 서버와 복수개의 캐시 시스템을 통해 웹서비스를 수행하는 대형 웹사이트의 구조를 나타낸다. 웹서버와 캐시 서버 사이에는 스위치를 두어 로드 밸런싱(load balancing)을 한다. 뿐만 아니라 Layer-4 스위치를 사용할 경우 트랜스페어런트 캐시(transparent cache)로 동작하여 클라이언트에게는 캐시 서버가 존재하지 않는 것처럼 구성 할 수도 있으며 웹서버의 환경 설정에 변화를 주지 않고 캐시 시스템을 이식시킬 수 있다[6].

각 캐시 서버는 ICP(Internet Cache Protocol)를 이용하여 서로간의 정보를 교환한다[7, 8]. 예를 들어, 클라이언트의 요청을 캐시1이 받았을 때, 캐시1에 클라이언트가 요청하는 정보가 없다면 캐시1은 다른 캐시 서버들에게 ICP request 메시지를 보내게 되고 해당 정보를 가진 캐시 서버가 있을 경우 그 캐시 서버로부터 정보를 얻어와 자신의 저장소에 저장하고 클라이언트에게 정보를 제공한다. 만약 정보를 가진 캐시 서버가 없을 경우 웹 서버에게 요청을 보내고 해당 정보를 가져오게 된다.

2.3 스퀴드 캐시(Squid cache)

스퀴드 캐시는 인터넷에 공개된 캐시. 프로그램으로 유닉스 기반의 시스템에서 동작한다. 또한 오픈 소스 프로젝트로 개발되어 소스 코드가 공개되어 있어 많은 캐시 시스템이 스퀴드 캐시를 기반으로 개발되어 상품화되어 있다[9]. 스퀴드 캐시는 트랜스페어런트 캐싱, SSL 캐싱, DNS 캐싱, HTTP, FTP, WCCP, ICP, SNMP 등을 지원한다. 이렇게 다양한 프로토콜과 관리 기능을 제공함으로 해서 캐시 시스템을 개발하는 많은 개발자가 스퀴드 캐시를 참고하여 캐시 시스템을 개발하였다.

스퀴드 캐시는 설정 파일인 squid.conf 파일을 이용하여 acl(access control list) 기능을 제공하는데 이 acl을 이용하여 캐시 시스템으로의 접근 제어 및 캐시 시스템의 사용자를 제한을 할 수 있다.

접속자 수가 많은 대형 웹사이트의 경우 복수개의 웹서버와 복수개의 웹캐시 시스템을 두어 많은 접속자로부터 발생하는 부하를 분산 시킨다. 복수개의 웹캐시 환경에서 이러한 부하 분산을 위한 캐시 시스템 구축을 위해 스퀴드 캐시를 이용한 계층적 분산 캐시 시스템(hierarchical and distributed caching)을 구축할 수 있다[10, 11].

스퀴드 캐시는 웹 기반의 모니터링 툴(cachemgr.cgi)을 제공한다. 이 툴은 CGI 기반으로 동작하며 스퀴드 캐시와 소켓 통신을 이용하여 메시지를 주고받는다. cachemgr.cgi는 cache_object라는 요청 메시지를 전송하는데 cache_object 뒤에 따르는 요청 항목에 따라 스퀴드 캐시는 자신이 가지고 있는 정보를 cachemgr.cgi로 전송한다. 스퀴드 캐시가 제공하는 cachemgr.cgi는 텍스트로 된 정보만 제공하고 있으며 단일 캐시에 대한 관리 기능을 제공한다. 따라서 복수의 캐시를 보유한 대형 웹사이트의 경우에는 전체 캐시 시스템에 대한 통합관리를 위한 기능이 필요하다. 본 논문에서는 스퀴드 관리기능을 보완 확장하여 복수의 서버 및 캐시를 보유한 웹사이트에 대한 통합관리를 위한 웹캐시 통합관리 시스템을 구축하였다.

3. 압축 기능을 가진 웹캐시 시스템

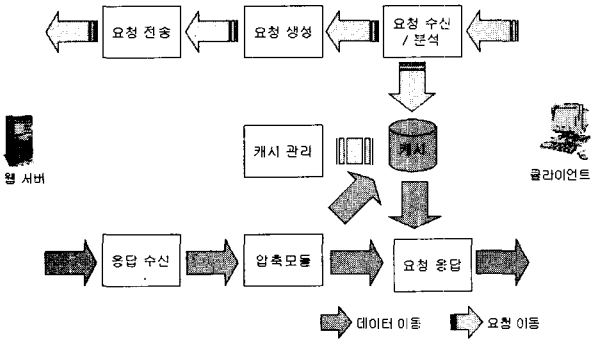
본 절에서는 구현된 압축기능을 가진 웹캐시 시스템의 웹서비스 처리 절차에 대해 기술한다.

3.1 압축 기능을 가진 웹캐시 시스템의 전체 흐름

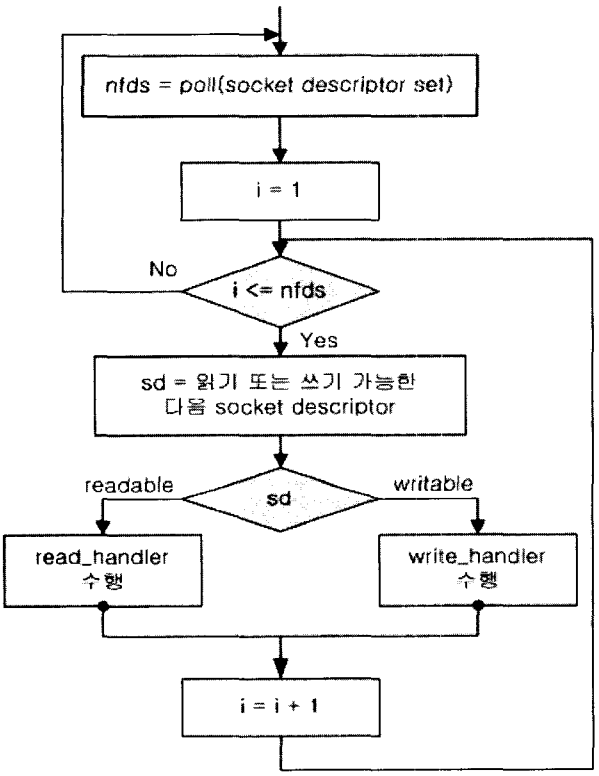
클라이언트로부터 웹서비스 요청이 왔을 때 캐시시스템에서 처리되는 개략적인 절차는 (그림 3)과 같다.

먼저 클라이언트로부터 웹서비스 요청이 도착하면 캐시 시스템은 자신의 로컬캐시 디렉토리에서 해당 객체가 존재하는지를 검사하고 해당 객체가 존재하고 유효할 경우 요청 응답 함수를 호출하여 클라이언트에 전송한다. 만약 요청된 웹객체가 로컬캐시에 존재하지 않으면 캐시시스템은 웹서버로 연결을 설정하고 클라이언트의 연결요청 메시지를 서버

에 전송하고 서버로부터의 응답을 기다린다. 이 때 IP_spoofing 옵션이 설정되어 있으면 연결요청 메시지의 송신자 주소를 클라이언트의 IP 주소로 설정하여 보내게 된다.



(그림 3) 압축기능을 가진 웹캐시 시스템 구조



(그림 4) 캐시 시스템 전체 흐름도

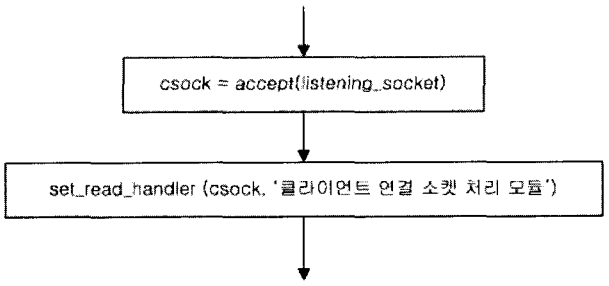
웹서버로부터 응답을 수신하면 웹캐시 시스템은 압축을 할 것인지를 검사하여 압축을 하지 않을 경우에는 요청응답 함수를 호출하여 클라이언트에 전송하고, 압축을 해야할 경우에는 서버로부터의 응답을 버퍼에 저장하고 전체 응답 메시지가 다 도착하면 압축 함수를 호출하여 압축을 수행한 다음 요청 응답 함수를 통해 클라이언트에 전송하게 된다.

3.2 서버, 클라이언트와의 연결 방법

웹캐시 시스템은 클라이언트로부터의 웹서버 요청을 받기 위한 요청 대기 소켓과 클라이언트와 메시지 송수신을 위한 클라이언트 연결 소켓, 그리고 서버와의 웹메시지 송수신을 위한 서버 연결 소켓을 통해 클라이언트와 서버 사이의 통신을 수행한다. 이러한 클라이언트와 서버 소켓들 사이의 메시지 송수신을 비동기적으로 처리하기 위해 스퀴드에서는 poll(또는 select) 함수를 사용한다. 캐시 시스템의 개략적인 전체 흐름도는 (그림 4)와 같다.

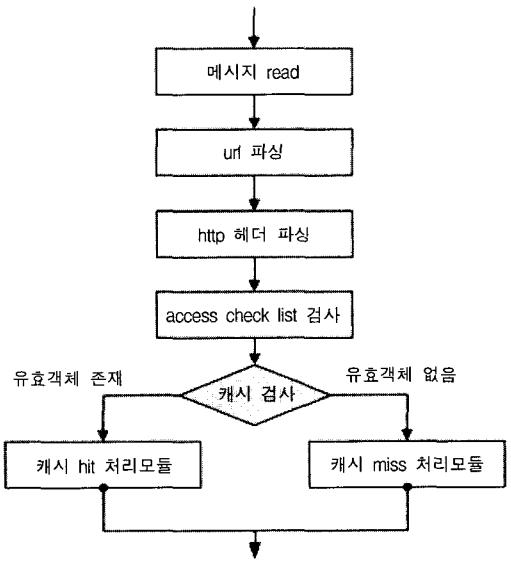
먼저 각 소켓들마다 해당 소켓이 읽기 가능 또는 쓰기 가능해질 때 수행할 함수들을 등록해 놓고 전체 소켓들 사이에 읽기 가능 또는 쓰기 가능한 소켓들을 poll(또는 select) 함수를 통해 검사한다. poll 함수에서 복귀하면 소켓 디스크립터들을 검사하여 읽기 가능하면 해당 소켓 디스크립터에 등록된 read_handler를, 쓰기 가능하면 해당 소켓 디스크립터에 등록된 write_handler 함수를 수행한다.

클라이언트의 listening 소켓이 읽기 가능할 때 수행하는 절차는 (그림 5)과 같다. 캐시는 클라이언트의 연결요청을 받아들이고 새로운 연결 소켓 csock을 생성하고 생성된 연결 소켓 csock에 대해 read_handler를 '클라이언트 연결 소켓 read_handler(그림 6)'로 설정하고 마친다.



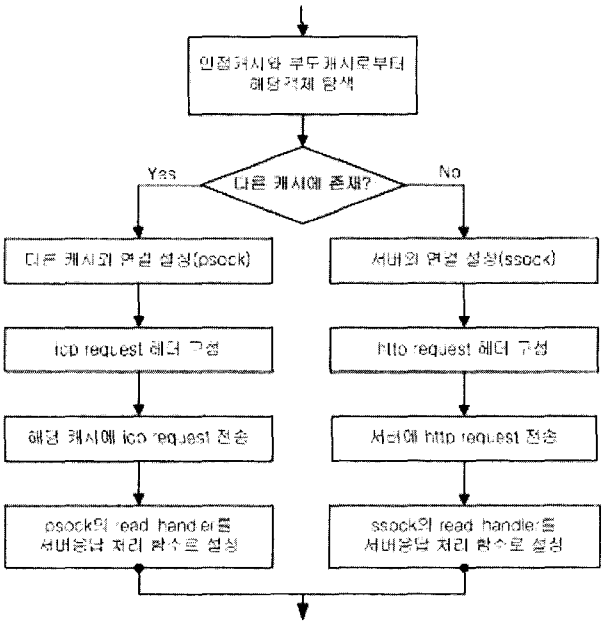
(그림 5) 클라이언트 listening 소켓 read_handler

클라이언트의 연결 소켓이 읽기 가능할 때 수행되는 read_handler 처리 절차는 (그림 6)과 같다.



(그림 6) 클라이언트 연결 소켓 read_handler

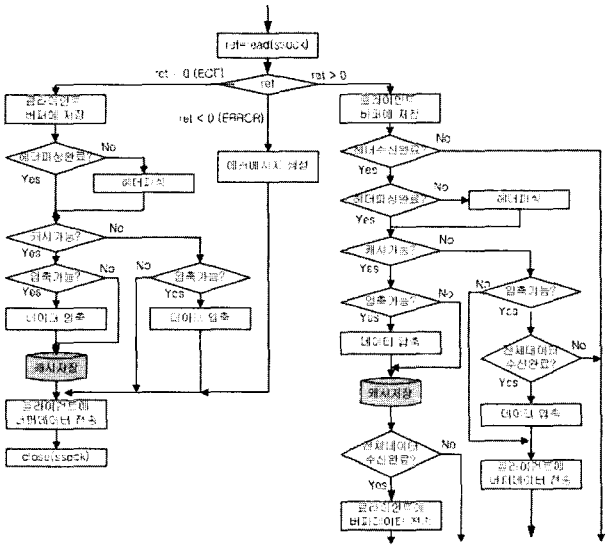
먼저 연결 소켓 csock에 대해 도착한 데이터를 읽은 다음 도착한 http 요청 메시지의 첫 라인의 url과 http 요청 헤더등을 파싱한다. http 요청 헤더의 파싱 정보와 웹캐시 설정 파일에 설정된 접근 제어 리스트를 검사하여 정당한 요청인지를 검사하여(정당한 클라이언트 또는 서버 IP, 도메인 검사, 허용된 url, 프로토콜, method, 최대 connection 수, http 또는 icp access 제어 기능 등의 검사) 정당한 요청이 아닌 경우에는 에러 처리를 한다. 다음에는 로컬 캐시 디렉토리에서 요청된 웹객체가 존재하며, 유효한지를 검사한다. 검사결과 유효한 객체가 캐시에 존재하면 캐시 hit 처리 모듈을, 존재하지 않으면 캐시 miss 처리 모듈을 수행한다. 캐시 hit 처리 모듈에서는 캐시 디렉토리에 있는 해당 웹객체를 읽어 클라이언트에 전송하게 된다. 캐시 miss 처리 모듈에서는 요청된 웹객체를 웹서버로부터 가져와서 클라이언트에 보내주게 되는데 그 절차는 (그림 7)과 같다.



(그림 7) 캐시 miss 처리모듈

3.3 캐시 miss 처리 모듈

캐시 miss 처리 모듈은 우선 캐시 설정파일에 설정이 되어있을 경우 요청된 웹객체가 인접 또는 부모 캐시에 존재하는지를 검사하게 되는데 이 때 캐시 시스템간의 메시지 전송은 ICP 프로토콜을 이용한다. 인접 캐시 또는 부모 캐시에 존재하게 되면 해당 캐시에 요청을 설정한다. 연결 설정이 끝나면 캐시 소켓(psock)이 만들어지게 되고, 캐시는 이 psock을 이용하여 해당 캐시서버에 웹객체를 요청하는 icp 메시지를 구성하여 ICP 요청메시지를 전송하게 된다. 인접캐시에 없을 경우 해당 서버에 웹객체를 요청하기 위해 서버와 연결을 설정하게 된다. 연결 설정이 끝나면 서버 소켓(ssock)이 만들어지게 되고, 캐시는 이 ssock을 이용하여 서버에 해당 웹객체를 요청하는 http 메시지를 구성하여 http 요청메시지를 전송하게 된다. 이 때 캐시 설정파일에 IP_spoofing 옵션이 on되어 있으면 transparent proxy 소켓을 통해 보내는 송신자를 원래 클라이언트의 주소로 보내게 된다. 기본적으로 IP_spoofing 옵션값은 off로 설정되어 있다. 이 후 서버로부터의 응답 메시지가 도착할 경우 처리하기 위해 ssock의 read_handler를 서버응답처리 모듈함수로 설정하고 캐시 miss 처리 모듈을 마치게 된다.



(그림 8) 서버 응답 처리 모듈

3.4 서버 응답 처리 모듈

서버 응답 처리 모듈은 캐시 miss가 발생하여 서버에 해당 웹객체를 요청하는 http 요청 메시지를 보내고 해당 서버로부터 오는 응답메시지를 처리하는 모듈이다(그림 8). 우선 서버로부터 응답 메시지가 도착하면 메시지를 읽어들이고 읽은 결과 EOF(End-Of-File)를 만나면 서버로부터 연결 종료 패킷이 도착한 경우이므로 해당 웹객체가 캐시 가능한 객체인 경우에는 캐시에 저장하게 되고 서버 소켓 ssock을 close하게 된다. 읽은 결과 에러가 발생하면 에러 메시지를 클라이언트에 전송하고 서버 소켓을 close하게 된다. 읽은 결과 정상적으로 메시지를 수신한 경우 받은 데이터를 버퍼에 저장하고 응답 헤더를 다 수신하지 않았으면 응답헤더를 다 수신할 때까지 받은 메시지 처리를 미루게 된다. 응답 헤더를 다 수신하면 응답 헤더를 파싱하고 이 파싱 결과를 이용하여 해당 웹객체의 캐시 가능여부와 압축대상인지를 검사하게 된다.

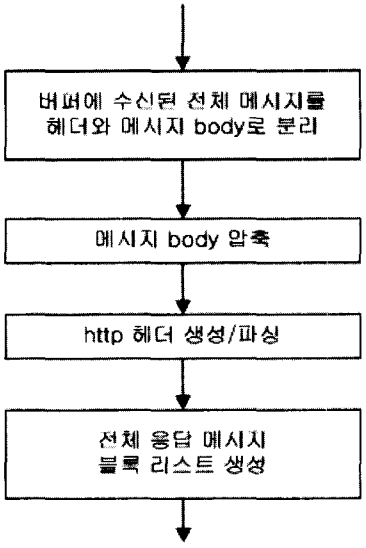
해당 웹객체가 압축대상이 아닐 경우에는 버퍼의 데이터를 바로 클라이언트에 전송하게 되고 압축 대상일 경우에는 전체 데이터를 다 수신할 때까지 클라이언트로 전송을 미루고 기다리게 된다. 압축대상이고 전체 데이터를 다 수신하면 버퍼에 저장된 전체 수신 메시지를 압축하여 전송하기 위한 압축 모듈을 수행하게 된다.

압축 모듈은 모든 컨텐츠에 대해 압축을 수행하는 것이 아니라 캐시 설정에서 명시한 컨텐츠 종류(Content-type)에

대해서만 수행하며 압축의 효율을 증대시키기 위해 압축할 원본 데이터의 최소 사이즈를 제한할 수 있게 하였다. 이렇게 하는 이유는 웹에서 사용하는 이미지(GIF, JPEG, PNG)들의 경우 이미지 포맷 자체가 압축을 수행한 것이기 때문에 추가로 압축을 하여도 크기의 변화가 거의 없으므로 오히려 비효율적으로 작용할 수 있기 때문이다. 또한 너무 작은 데이터의 경우 압축에 소요되는 시간과 전송시간을 고려하여 적절하게 제한하는 것이 더 나은 결과를 내기 때문이다.

3.5 압축 수행 방법

압축 모듈은 압축 대상 웹객체에 대해 서버로부터 수신한 웹객체를 압축한 다음 필요한 응답 헤더를 구성하여 주고 클라이언트에 전송하여 주는 기능을 수행하며 그 절차는 (그림 9)와 같다.

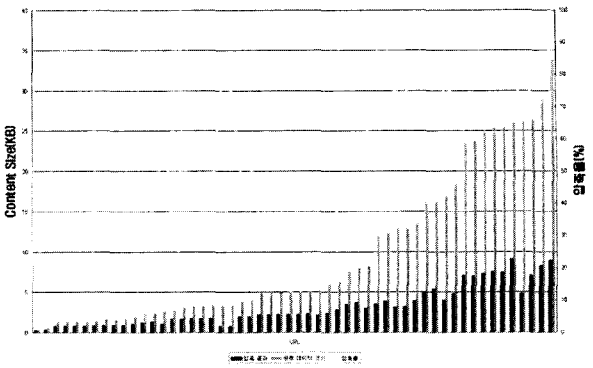


(그림 9) 압축모듈

먼저 압축모듈에서는 수신된 전체 응답 메시지를 응답 헤더 부분과 메시지 body 부분으로 분리하고 메시지 body 부분에 대해 압축 알고리즘을 통해 압축을 수행한다. 압축을 수행하면 메시지 body의 길이가 변하고 또한 압축 데이터라는 것을 클라이언트에게 알려주기 위해 서버로부터 받은 응답메시지의 헤더를 수정해 주어야 한다. 압축된 메시지 body에 새 응답헤더를 추가하고 나면 다시 한번 응답 헤더를 파싱하고 클라이언트에 전송하기 위해 전체 메시지를 블록 단위로 나누어 전송할 메시지 블록의 리스트를 수정한다(클라이언트로의 메시지 전송은 한번에 한 블록씩 수행한다). 전체 송신할 응답 메시지 블록 리스트가 생성되면 해당 클라이언트로 전송을 시작한다.

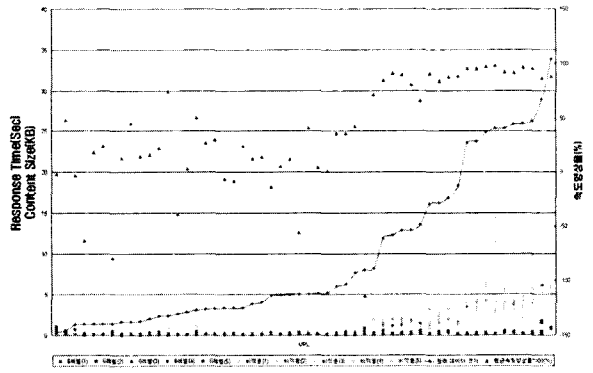
3.6 성능 테스트 결과

본 논문에서 개발한 압축 기능을 가진 웹캐시 시스템을 H사의 그룹웨어에 적용해 보았다.



(그림 10) 컨텐츠 압축에 따른 데이터 크기 변화 비교

(그림 10)에서는 H사의 그룹웨어를 사용하는데 있어서 압축 이전의 데이터 크기와 압축 이후의 데이터 크기를 비교한 결과를 보여주고 있다. 그래프에서의 옅은 색(▬)의 막대는 압축을 수행하기 이전의 데이터의 사이즈를 나타내고 짙은 색(▬)의 막대는 압축을 수행한 이후의 데이터의 사이즈를 나타낸다. 또한 실선( )은 데이터 압축에 따른 압축률($\frac{\text{압축한데이터의크기}}{\text{압축하지않은데이터의크기}} \times 100(\%)$)을 나타내고 있다. 데이터의 컨텐트 종류(Content-type)는 text/html 형식의 데이터들이다. 그래프에서 볼 수 있듯이 컨텐트 압축 기법을 통해 전송되는 데이터는 평균 56.8%로 그 크기가 줄어든 효과를 볼 수 있다.



(그림 11) 컨텐츠 압축에 따른 전송 시간 변화 비교

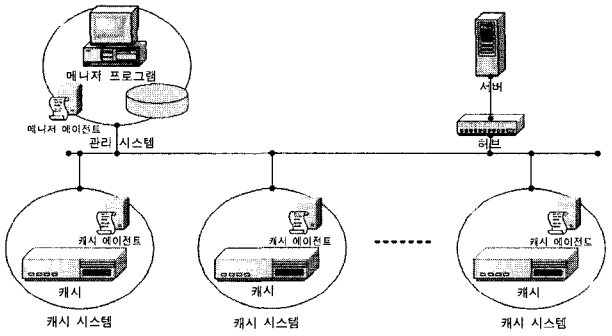
(그림 11)은 압축으로 인한 데이터 크기 변화로 인한 전송 시간의 차이를 비교한 그래프이다. 전송 시간은 테스트할 시점의 환경의 영향을 많이 받을 수 있으므로 5번 반복하여 각 결과를 함께 표시하였다. 그래프에서 나타나는 사각형의 점(■)은 압축을 한 데이터의 전송 시간을 나타내고 동그란 점(◇)은 압축을 수행하기 이전의 원래 데이터의 전송 시간을 나타내고 있다. 마름모 점을 연결한 실선(—◆—)은 원래 데이터의 크기를 나타내고 있으며 삼각형의 점(▲)은 평균 전송 시간의 향상율($\left(1-\frac{\text{압축한후의전송시간}}{\text{압축하기이전의전송시간}}\right)\times 100(\%)$)을 나타내고 있다. 그래프에서 볼 수 있듯이 5번의 테스트에서

나타난 속도의 향상율은 평균 36.8%의 시간을 단축할 수 있었다. 하지만 데이터의 크기가 10Kbyte를 넘는 경우 압축 후 전송하면 전송 시간은 평균 86.6%를 단축하는 결과를 보였다.

1GHz의 CPU를 가진 캐시 시스템에서 30KByte의 데이터를 압축하는데 걸리는 시간을 측정해본 결과 1.02ms 정도로 나타났다. 압축을 해제하는데 걸리는 시간 역시 압축을 하는데 걸리는 시간과 큰 차이가 없다. 이것은 하나의 응답시간과 비교해 볼 때 매우 경미한 시간이므로 압축에 따른 오버헤드는 무시할 수 있는 수준이라고 볼 수 있다.

4. 웹 캐시 통합 관리 시스템

동시 접속자 수가 많은 대형 웹사이트에서는 복수개의 웹서버 및 복수개의 웹캐시 시스템을 필요로 한다. 이러한 다중 웹캐시 시스템에서는 전체 웹캐시 시스템을 통합적으로 모니터링 및 제어를 하기 위한 통합 웹캐시 관리시스템이 필요하다. 본 논문에서는 복수의 캐시 시스템을 사용하는 웹 서비스 환경에서 여러 대의 캐시 시스템을 통합된 환경에서 관리하고 모니터링 할 수 있는 관리 시스템을 개발하였다. 또한 별도의 장비 없이 웹 브라우저만으로 관리가 가능하도록 웹 기반의 인터페이스를 제공한다.



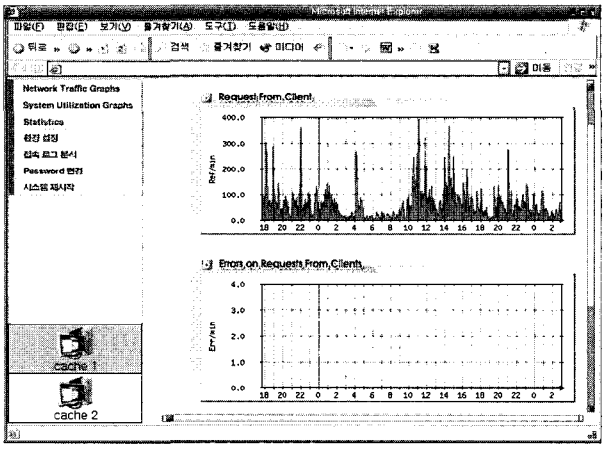
(그림 12) 통합 관리 시스템의 구성

(그림 12)는 본 논문에서 개발한 통합 관리 시스템의 구성을 보여주고 있다.

관리자는 관리 시스템에 접속하여 캐시 서버의 상태를 모니터링 하거나 설정을 변경하기도 하고 캐시 서버를 재시작 시킬 수도 있다.

관리 시스템에 존재하는 메니저 에이전트는 서버와의 직접적인 통신을 담당한다. 이 에이전트는 캐시 서버가 클라이언트로부터 요청 받은 회수, 요청에 대한 에러 수, 웹 서버로부터 가져온 데이터의 양(bytes), 웹 서버로 전송한 데이터의 양(bytes), CPU 사용률, 메모리 사용률, 시스템 페이지 폴트가 발생한 회수 정보를 캐시로부터 1분마다 수집하여 데이터 베이스에 저장한다. 또한 관리자가 원하는 실시간의 정보를 캐시에게 요청하고 캐시 서버가 보내오는 응답 메시지를 관리 프로그램으로 전달한다. 여기서 요청될 수 있는 실시간 정보로는 메모리 점유 및 이용 상태, 런 타임 정보, 파일 오픈 정보, DNS 상태, IP 캐시 상태, 현재 처리중인 요청 리스트, 클라이언트의 리스트 등이 있다. 메니저 프로그램은 메니저 에이전트에 의해 수집되어 데이터베이스에 저장된 정보와 캐시를 접근하여 전송 받은 정보를 처리하여 관리자가 이해하기 쉬운 표나 그래프 형태로 나타내주는 기능을 한다. 메니저 에이전트가 캐시 서버와 주고받는 메시지는 스퀴드 캐시가 제공하는 cachemgr.cgi이 사용하는 메시지 형식을 그대로 사용하도록 하였다. 또한 캐시 서버의 acl 설정에서 관리 시스템의 접근만을 허용하게 하여 보안의 측면도 고려하였다.

각 캐시 시스템에는 캐시 에이전트라는 에이전트를 두고 있다. 이 에이전트는 메니저 에이전트와 통신을 하면서 스퀴드 캐시의 설정 파일(squid.conf)을 읽거나 수정하고 로그 파일(access.log)을 읽어서 메니저 에이전트로 전송하는 것이다. 이렇게 해서 관리 프로그램은 웹 기반으로 스퀴드의 설정 파일을 수정할 수 있고 로그 파일을 관리 시스템에서 분석할 수 있게 된다. 관리 프로그램을 캐시 시스템에 두지 않는 이유는 관리 기능으로 인한 캐시 시스템의 부하를 최소화하기 위한 것이다.



(그림 13) 웹 기반 웹 캐시 통합 관리 시스템 구현

웹캐시 통합 관리 시스템 구현에 사용된 운영체제는 리눅스 커널 2.4이고 관리 프로그램은 PHP4, 에이전트는 C언어를 이용하여 구현했으며 데이터 베이스는 MySQL을 이용하였다. (그림 13)은 통합 웹캐시 관리 시스템의 메인 화면을 나타내는데 관리자는 좌측 하단에 있는 캐시 서버 리스트에서 설정을 변경하거나 모니터링 하고자 하는 캐시 서버를 선택할 수 있게 하였다. (그림 13)은 Cache1 시스템에 대한 분당 클라이언트 요청수와 이에 대한 에러율을 보여주고 있다.

5. 결론 및 향후 과제

본 논문에서 제시한 압축 기능을 가진 웹캐시 시스템은

웹컨텐츠를 서버에서 읽어와 압축을 수행한 다음 클라이언트에게 전달하는 매커니즘을 수행한다. 압축을 한 데이터는 그 크기가 작아지게 되므로 클라이언트에게 도달하는 시간이 줄어들게 되고 웹페이지 로딩 속도가 향상되게 된다. 압축하는 컨텐츠의 종류를 선별하는 과정을 거침으로 해서 비효율적인 압축과정을 피하고 압축율을 조절할 수 있어 압축의 효율성을 향상시킬 수 있었다. 또한 실제 환경에 캐시 시스템을 적용하여 실험을 거친 결과 텍스트로 이루어진 컨텐츠의 경우 평균 56.8%로 데이터의 크기를 줄일 수 있었으며 전송시간의 경우 86.6%로 단축시킬 수 있었다.

또한 다수의 웹캐시를 운영하는 웹서비스 환경에서 통합관리 시스템을 이용하여 복수의 웹캐시 시스템을 하나의 통합된 환경으로 관리하고 모니터링 할 수 있게 되었다. 본 논문에서 제시한 통합관리 시스템은 표 형식과 그래프화된 이미지를 제공하여 캐시 시스템의 사용률과 동작 상태를 모니터링하고 웹 환경에서 설정파일을 수정하여 캐시 시스템에 적용할 수 있도록 하여 관리의 효율을 향상시킬 수 있었다. 캐시 시스템의 관리 기능을 캐시 시스템과 독립적인 환경에 구축함으로 해서 캐시 시스템의 부하를 최소화로 줄여 성능 저하를 막고 이해하기 쉽고 관리하기 쉬운 환경을 제공할 수 있었다.

향후 계획으로는 웹캐시의 성능향상을 위한 멀티스레드 방식의 모듈구조 전환과 웹서버의 설정 및 로그 분석 기능을 추가하여 통합 웹사이트 관리 툴로 발전 시켜 나갈 것이다.

참 고 문 헌

[1] Vakali, A., "A Web-based evolutionary model for Internet data caching," Database and Expert Systems Applications, Proceedings, Tenth International Workshop, pp.650-654, Sept., 1999.

[2] Mingkuan Liu, Fei-Yue Wang, Zeng, D., Lizhi Yang, "An overview of World Wide Web caching," Systems, Man, and Cybernetics, 2001 IEEE International Conference, Vol.5, pp.3045-3050, Oct., 2001.

[3] Sahuquillo, J., Pont, A., "The filter cache : a run-time cache management approach," EUROMICRO Conference, 1999. Proceedings. 25th, Vol.1, pp.424-431, Sept., 1999.

[4] Ker Kob, Sam H. Nob, Sang Lyul Min, "Efficient Replacement of Nonuniform Of Objects in Web Cache," IEEE Computer, Vol.35, Issue 6, pp.65-73, June, 2002.

[5] Hyokyung Bahn, Kern Koh, Noh, S. H., Lyul, S. M., "Site-Based Approach in HTTP Proxy design," Proc. Int'l Conf. Parallel Processing, Workshop on Internet, IEEE Computer Soc. Press, Los Alamitos, Calif., pp.228-233, 1999.

[6] Liang, Z., Hassanein, H., Martin, P., "Transparent distributed Web caching," Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference, pp.225-233, Nov., 2001.

[7] Duane Wessels and K. Claffy., "ICP and the Squid Web Cache," Selected Areas in Communications, IEEE Journal on, Vol.16, Issue 3, pp.345-357, April, 1998.

[8] Chiang, C.-Y., Liu, M. T., Muller, M. E., "Caching neighborhood protocol : A foundation for building dynamic Web caching hierarchies with proxy servers," Parallel Processing, 1999 Proceedings, 1999 International Conference, pp.516-523, Sept., 1999.

[9] "Squid-based Products," online documents http://www.squid-cache.org/products.html.

[10] Rodriguez, P., Spanner, C., Biersack, E. W., "Analysis of Web caching architectures : hierarchical and distributed caching," Networking, IEEE/ACM Transactions, Vol.9 Issue 4, pp.404-418, Aug., 2001.

[11] Cho-Yu Chiang, Yingjie Li, Liu, M. T., Muller, M. E., "On request forwarding for dynamic Web caching hierarchies," Distributed Computing Systems, 2000 Proceedings, 20th International Conference, pp.262-269, April, 2000.

박 진 원

e-mail : zwsonic@cnlab.ulsan.ac.kr
2002년 울산대학교 공학사
2004년 울산대학교 공학석사
현재 울산대학교 컴퓨터·정보통신공학부 박사과정
관심분야 : 정보통신, 정보처리, 정보보호

김 명 균

e-mail : mkkim@mail.ulsan.ac.kr
1984년 서울대학교 공학사
1986년 한국과학기술원 공학석사
1996년 한국과학기술원 공학박사
1989년~1997년 우석대학교 교수
1997년~현재 울산대학교 컴퓨터·정보통신공학부 부교수
관심분야 : 정보통신, 정보처리, 망관리

홍 윤 환

e-mail : hong@drsoft.co.kr
1983년~1997년 현대중공업
1993년 울산대학교 공학사
2000년 (주)닥터소프트 대표이사
2003년 신소프트웨어 대상(정보통신부 장관상)수상
현 재 (주)닥터소프트 대표이사
관심분야 : 정보통신