커널 쓰레드 웹가속기(SCALA-AX) 개발

박 종 규[†]·민 병 조[†]·임 한 나[†]·박 장 훈[††]·장 휘[††]·김 학 배[†††]

요 약

주로 캐싱서버로 사용되는 기존의 프록시웹캐시는 단순히 웹서버의 컨텐츠를 복사해서 서비스를 제공하는 시스템이다. 이 방법은 실제로 컨텐츠 서비스를 담당하는 웹서버의 성능 향상보다는 컨텐츠 전달 중간단계의 속도 향상에 초점을 맞추고 있다. 그러나, 과다한 동시 접속자로 인하여 웹서버에 과부하가 걸렸을 경우에는 많은 효과를 보지 못하고 있다. 본 논문에서는 웹서버의 성능을 극대화시켜 클라이언트에게 보다 빠른 서비스를 제공하기 위해서 웹가속기(SCALA-AX)를 제안한다. SCALA-AX는 리눅스 커널 모듈로 구현되어, 유저레벨 웹서버 어플리케이션과 함께 작동을 한다. SCALS-AX는 HTTP 요청을 커널쓰레드를 이용하여 처리 하기 때문에 전달 속도 향상뿐만 아니라 캐싱서버 설치로 인한 추가비용도 발생하지 않는다. SCALS-AX의 성능평가 결과, SCALA-AX가 구현된 웹서버는 기존의 웹서버보다 데이터 전달면에서 5배 이상의 속도 향상을 얻었다. 즉, 웹서버의 성능이 크게 향상되었음을 볼 수 있다.

Development of a Kernel Thread Web Accelerator (SCALA-AX)

Jonggyu Park[†]·Byungjo Min[†]·Hanna Lim[†]
JangHoon Park[††]·Whi Chang[††]·Hagbae Kim[†††]

ABSTRACT

Conventional proxy web cache, which is generally used to caching server, is a content-copy based system. This method focuses on speeding up the phase delivery not improving the webserver performance. However, if immense clients attempt to connect the webserver simultaneously, the proxy web cache cannot achieve the desired result. In this paper, we propose the web accelerator called the SCALA-AX, whitch improves web server performance by accelerating the delivery contents. The SCALA-AX is built in the Linux-based kernel as a kernel module and works in combination with the conventional webserver program. The SCALA-AX speeds up the processing rate of the webserver, because it processes the requests using the kernel thread. The SCALA-AX also applies the well-developed cache algorithm to the processing, and thus it obtains the advantage of the caching server without installing additional hardware. A banchmarking test demonstrates that the SCALA-AX improves webserver performance by up to 500% for content delivery.

키워드 : 웹가속기(web accelerator), 웹캐쉬(web cache), 리눅스(linux), 커널쓰레드(kernel thread)

1. 서 론

아파치등 기존의 웹서버는 클라이언트/서버 모델과 웹의 HTTP를 사용하여 웹 페이지가 포함되어 있는 화일을 사용자들에게 제공하는 것이다. 이러한 웹서버의 성능은 일반적으로 여러 가지 요소에 의해 제약을 받는다. 다중쓰레드 구조의 속도상 오버헤드와 메모리 복사와 디스크 접근에서 일어나는 자원의 낭비는 웹서버가 효율적인 서비스를 수행하는데 큰 제약요소가 된다. 또한, 요구/응답 메커니즘으로 이루어지는 HTTP 프로토콜에서는 비교적 조그만 데이터 양을 가지는 패킷들의 연속적인 교환으로 이루어지기 때문에 결과적으로 더 많은 갯수의 TCP 세그먼트를 양산하게 될 수 있다. 그래서, 웹 성능향상에 가장 효과적인 해결방법으로 부각되고 있는 것이 웹 캐시이다. 웹 캐시는 HTTP에서 서비스 된 페이지를 캐싱한 상태에서 다시 이 페이지가 요청될 때 다시 서버에서 페이지를 가져올 필요가 없이 캐싱되어 있는 페이지를 바로 서비스하게 된다[1]. 즉, 하드디스크의 엑세스를 최소화하기위하여 한번 요청되었던 페이지는 메모리에 저장해 두어 다음번의 같은 페이지 요청시 사용된다. 이러한 웹 캐싱기법은 인터넷 트래픽의 80% 이상을 차지하는 HTTP 트래픽을 매우 효율적으로 관리하고 처리할 수 있다. 하지만 이와 같은 기존의 프락시 웹캐시(proxy web cache)[1-5]는 단순한 컨텐츠 복사중심 시스템(content-copy based system)[6]이기 때문에 캐싱서버(caching server)의 위치 선정문제[7]와 더불어 컨텐츠 플레쉬니스(content-freshness)문제[8]와 근래의 웹에서 중요한

※ 본 논문은 산업자원부 중기거점 과제 "초고속 Scalable 웹서버 개발"의 지원하에 연구되었습니다.
† 준 회 원 : 연세대학교 대학원 전기전자공학과
†† 정 회 원 : 씨아이사(C-EISA) CEO
††† 정 회 원 : 연세대학교 전기전자공학부 교수
논문접수 : 2001년 7월 31일, 심사완료 : 2002년 7월 2일

위치를 차지하고 있는 동적페이지(dynamic page)[9-12]의 처리 문제를 야기한다.

또한 기존의 웹서버 머신들이 웹 서비스를 제공하기 위해 사용하는 Apache나 IIS, Iplanet 등은 빠른 속도의 서비스보다는 여러 가지 다양한 웹 요청에 유연하게 대응하기 위한 것이므로, 정상적인 상황(순간 접속 몇 백 명 이하)에서는 속도나 서비스 측면에서 유연하게 동작할 수 있으나, 더 많은 사용자, 더 많은 시스템 자원 사용이 요구되는 경우에는 정상적인 기능을 수행하기 어렵다. 이를 해결하기 위해서 고성능의 서버를 도입하고 높은 대역폭의 망을 구축하는 방법이 있을 수 있겠지만, 이것은 많은 추가 비용을 요구한다.

그러나, 본 연구에서 구현한 웹가속기는 웹 서비스를 제공하는 웹서버와 동일한 머신에서 작동하여 웹서버의 서비스 속도를 극대화한다. 그리고 대역폭을 절감해 주고 서버의 과부하를 줄여주어 클라이언트에게는 더욱 더 신속한 응답시간을 보장해주는 역할을 한다. 웹서버와 웹가속기가 물리적으로 하나의 머신 안에 존재하므로, 따로 캐싱서버를 구입할 필요가 없어서 경제적이며, 컨텐츠 플레쉬니스[8]에 대한 문제를 걱정할 필요가 없다. 또한, SCALA-AX를 커널 레벨에서 구현하고 정적페이지에 관해서는 커널 레벨의 빠른 동작과 웹캐시기법을 통하여 성능을 향상시켰다. 그리고 웹캐시의 문제점으로 대두되고 있는 동적페이지에 관한 처리[9-12]를 커널 레벨의 빠른 처리와 기존의 웹서버(Apache)의 안정적인 동적페이지처리 능력을 결합하여 동적페이지에 관하여서도 안정적으로 속도를 향상 시켰다.

우리는 이런 아이디어들을 실제 리눅스의 커널레벨에서 구현하여 리눅스 웹서버의 성능을 비약적으로 향상시키는 커널 레벨 다중쓰레드 웹가속기를 구현하였다. 그리고 http 1.1의 연결지속(persistent connection)이나 파이프라인(pipeline)을 구현했을 뿐만 아니라 해당되는 요청만을 웹서버에게 전달하고 나머지 모든 연결지속과 파이프라인을 통해 전달된 요청을 다시 직접 처리함으로써 뚜렷한 성능 향상을 이루었다. 마지막으로, Web bench의 초당 처리 요청 수와 클라이언트 당 throughput을 이용하여 웹서버의 성능향상을 벤치마크 하였다.

2. 커널레벨 웹가속기의 구현

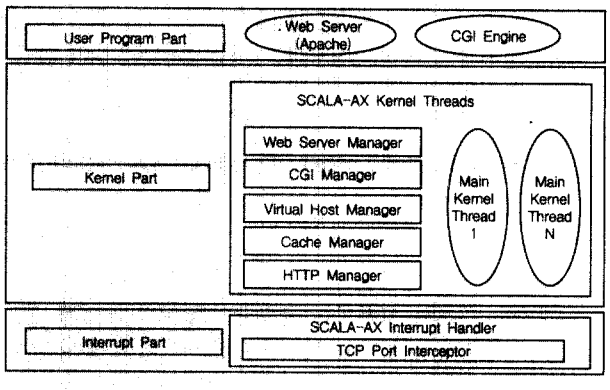
2.1 전체구조

SCALA-AX는 웹서버에 탑재되며 최신 캐시 기술을 바탕으로 커널 레벨에서 동작한다. 또한, 사용자 환경에서 구현되어 있는 http 프로토콜 스택을 커널 레벨로 내리고, 정적데이터 및 일부 동적데이터의 웹 서비스를 커널에서 수행함으로써, 일반적인 웹서버가 지니는 다중쓰레드 구조의 속도상 오버헤드와 메모리 복사와 디스크 접근에서 일어나는 자원 낭비를 줄임으로써, 서비스 속도와 효율을 획기적으로 향상시킬 수 있다. 이를 통해, 웹서버의 응답속도(response time)및 초당 처리 요청 수(requests/sec)를 획기적으로 개선할 수 있다. 또한 http1.1 프로토콜을 지원하여, 연결지속 및 파이프라인 요청(pipelining request) 등의 기능을 구현한다. 이는 기존의 웹가속기[3, 4] 기술들과 현격하게 차별화 되는 것으로, 사용자가 많고 접속이 계속 이어지는 E-commerce 사이트 등에 유용하게 적용될 수 있다. 앞서 설명된 바와 같이 중요한 특징 중의 하나는 정적페이지뿐만 아니라, 동적페이지의 서비스를 향상시켰다는 면이다.

전방 프락시 캐싱(Forward proxy caching) 및 후방 프락시 캐싱(reverse proxy caching)방식의 독립 캐싱서버(caching server) 혹은 기존의 웹가속기는 html, 이미지 파일 등의 정적페이지를 웹서버가 처리하기 전에 미리 그 앞에서 처리함으로써 전체적으로 웹서비스 속도를 향상시키고자 하는 개념이다. 그러나, 전자 상거래 사이트 등에서 보여지는 바와 같이 클라이언트의 개별적 요구에 웹서버 프로그램이 유연하게 응답하여야 하는 시스템 하에서는 정적페이지 처리 뿐 아니라 동적페이지까지 처리할 수 있는 솔루션을 요구하고 있다. 이러한 추세에 따라 본 연구에서는 커널 기반에서 동작함으로써 아파치 등의 웹서버가 처리해야 하는 동적페이지 처리까지 간섭하여 처리할 수 있도록 하였다[9-12].

2.2 커널 쓰레드 및 모듈

본 연구에서 구현된 SCALA-AX는 모듈로 구현하여 커널 패치 등의 다른 영향을 받지 않고 리눅스 -2.4.0 이상의 모든 커널에 설치 가능한 구조를 가지고 있다.



(그림 1) SCALA-AX의 전체구조

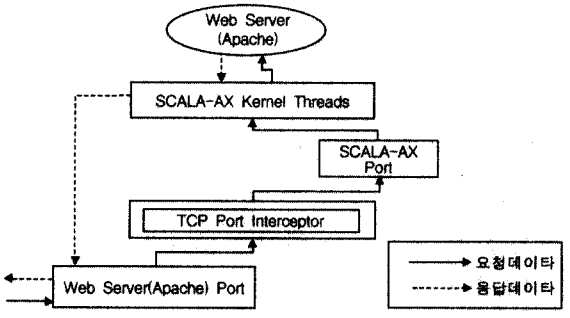
(그림 1)의 전체 구조에서 볼 수 있듯이 리눅스 플랫폼상에서 구현된 위치에 따라 크게 입력 TCP 계층의 패킷을 intercept하고 가속기의 handler가 위치하는 interrupt part와 web server manager, cgi manager, virtual host manager, cache manager, http parser가 위치하는 kernel part로 나누고 user program part와 연동되어 동작한다.

네트워크를 통해서 클라이언트의 요청이 전달되면, interrupt part의 TCP port interceptor가 이 데이터를 가로채어 먼저 SCALA-AX에게 전달되도록 하므로 아파치 등의 사

용자 프로그램의 웹서버 보다 항상 앞서 처리 할 수 있다. 커널쓰레드 형태로 동작하는 SCALA-AX는 CPU 갯수와 같은 수의 주요 쓰레드가 수행되면서 http 요청을 해석하는 http parser, 응답할 데이터를 메모리 상에서 관리하는 cache manager, virtual hosting을 지원하는 virtual host manager, user program part의 웹서버를 관리하는 web server manager, cgi를 커널에서 처리할 수 있게 관리하는 cgi manager등의 라이브러리를 이용하여 전달받은 클라이언트의 요청을 해석 및 처리한다. 클라이언트의 요청은 여러 가지 조건에 따라 SCALA-AX가 자체적으로 처리할 것인지 또는 메모리 상에 저장할 것인지, user_program part의 웹서버에게 넘겨야 하는 것인지 등을 판단하여 이에 따른 행동을 취하고 클라이언트에게 응답을 하게 된다. 이러한 과정은 모두 커널 내에서 이루어지게 되며 메모리 상에 데이터를 올려놓음으로써 얻어지는 속도에 대한 이득 이외에도, 다양한 커널의 기능을 직접 이용 및 조정함으로써 웹서버로서 성능 향상을 달성할 수 있다.

2.2.1 데이터 전달 구조

클라이언트로부터 전달된 요청은 일반적인 웹서버가 열어놓은 TCP 포트로 전달된다. SCALA-AX는 user_program part의 웹서버가 전달받을 클라이언트의 요청을 커널에서 먼저 처리하여 속도를 개선하면서도 아파치 등의 웹서버가 그 사실을 인식하지 않도록(transparent) 하는데 초점을 맞추고 있다. 따라서 (그림 2)와 같은 구조가 요구된다.



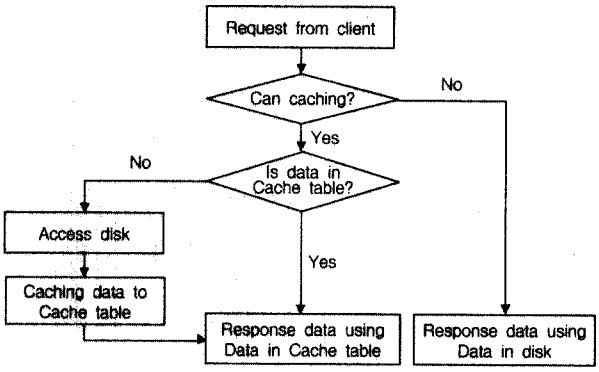
(그림 2) 데이터 전달구조

클라이언트는 일반적으로 웹서비스를 위해 사용하는 TCP의 80 포트로 요청을 보낸다. 이 요청은 interrupt part의 TCP port interceptor가 가로채어 SCALA-AX가 열어놓은 TCP 포트로 전달한다. 이 과정으로 SCALA-AX는 모든 클라이언트의 요청을 가져올 수 있게 되고, 이에 따른 적절한 해석 및 동작을 수행한 후에 웹서버가 열어놓은 TCP 포트를 통해 클라이언트에 응답을 한다. 일부의 경우 SCALA-AX가 처리하지 못할 요청은 웹서버에게 넘겨주지만 응답은 역시 SCALA-AX를 통해서 한다.

이렇게 입력부에서 간단히 TCP 정보를 해석하여 처리하는 구조는 클라이언트와 사용자프로그램부(user program part)의 웹서버 모두 설정의 변경 없이 SCALA-AX가 리눅스 커널의 모듈로서 삽입될 수 있게 하며, 클라이언트의 요청을 user part까지 전달하지 않고 kernel part에서 처리함으로써 성능 및 속도의 향상을 이룰 수 있도록 한다.

2.2.2 캐싱 구조

SCALA-AX는 클라이언트의 요청에 대한 응답 데이터를 자체적으로 메모리에 저장하여 테이블로 관리하기 때문에 상대적으로 속도가 느린 디스크 접근을 최소화하고, 네트워크를 통한 응답시에 요구되는 형태로 미리 저장함으로써 데이터의 복사 및 변경을 최소화한다.



(그림 3) 캐싱 구조

클라이언트로부터 요청이 들어오면 SCALA-AX는 응답할 데이터가 메모리에 저장되어야 할 것인지를 판단하고, 만일 저장 가능하다면 같은 데이터가 이미 저장되어 관리되고 있는지를 살펴본다. 저장되어 있는 경우라면 바로 읽어 들여 응답을 하고, 저장되어 있지 않으면 디스크에 저장되어 있는 데이터를 이 후 사용할 네트워크의 특성에 알맞은 형태로 변환하고 필요한 항목들을 미리 추가하여 메모리에 저장한 후 응답을 한다.

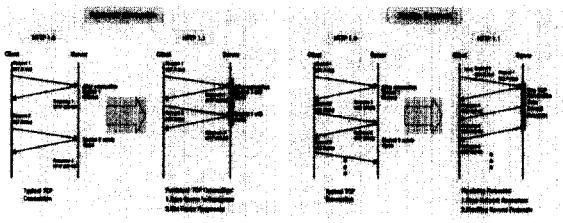
이와 같은 과정은 처음 한 번의 경우는 디스크를 접근하게 되지만 클라이언트의 요청이 진행됨에 따라 동일한 요청이 많아지고 결국은 대부분의 경우 저장된 메모리를 사용하게 되며 속도의 지속적인 향상이 이루어질 수 있다.

2.2.3 http 1.1 지원

http 1.1 프로토콜의 핵심적인 변화요소는 연결지속 및 파이프라인 요청기법이다[14]. 기존 http 1.0까지의 프로토콜은 하나의 TCP 커넥션에 하나의 http 요청을 보냄으로써 동작하는데, 클라이언트가 빈번한 요청(html, jpg, GIF등등)을 보내는 경우, 실제의 데이터 송수신보다 TCP 커넥션을 맺는데 서버 및 네트워크의 자원과 시간을 많이 낭비하여, 효율적인 네트워크 대역폭을 조정하는데 어려움이 있었다. 하지만 http 1.1 이후부터는 http 프로토콜의 헤더를 이용하여, 하나의 커넥션이 끝난 이후에도, 서버 쪽의 소켓을 종료하지 않고, 같은 클라이언트의 다음 요청을 같은 소켓으

로 처리할 수 있게 되었다.

즉, 이로 인해 실제적으로 TCP 커넥션에서의 초기화과정을 생략할 수 있고, 시스템의 자원 낭비 및 네트워크의 시간 지연요소를 줄일 수 있다. 또한 이러한 연결지속을 이용하여, 클라이언트의 요청을 pipeline을 이루게 함으로써 하나의 TCP 커넥션에 여러 개의 요청을 보내고, 서버는 이를 해석하여, 적절하게 응답해줄 수 있게 되어, 전체적으로 TCP 커넥션에 걸리는 시간과 대역폭을 절약할 수 있다.

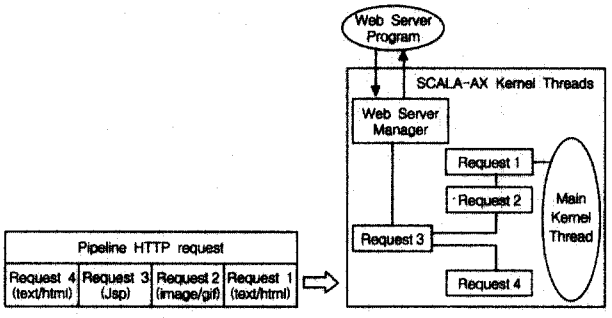


(그림 4) http 1.1

SCALA-AX는 http 1.1의 연결지속이나 파이프라인을 구현했을 뿐만 아니라 위와 같은 경우, 해당되는 요청만을 웹서버에게 전달하고 나머지 모든 연결지속 또는 pipeline을 통해 전달된 요청을 다시 직접 처리함으로써 뚜렷한 성능 향상을 이루었다.

따라서 속도 및 성능 개선을 위해서는 http 1.1의 구현은 필수적이며 또한 기존의 사용자 프로그램형태인 웹서버와의 조화로운 동작이 필요하다. 즉, SCALA-AX와 같은 웹 가속기의 경우 자신이 처리할 수 없는 요청은 발생할 수 밖에 없으며 이 경우 웹서버에게 전달하게 되는데 처리할 수 없는 요청이 연결지속과 파이프라인을 통해 전달되면 해당되는 요청뿐만 아니라 동시에 전달된 나머지 모든 요청을 웹서버에 넘겨주게 된다.

(그림 5)와 같이 pipeline으로 전달된 http 요청들은 SCALA-AX가 해석한 뒤, 내부적으로 분리되어 관리하고 처리할 수 없는 요청만을 웹서버 관리자를 통해 웹서버에 전달하고 그 송신을 감지한다. 이 과정은 전달된 순서에 따라 이루어지기 때문에 http 1.1 프로토콜에 적합한 형태로 가능한한 SCALA- AX가 많은 요청을 처리할 수 있게 하여 성능의 개선이 이루어질 수 있다.



(그림 5) http 요청 전달 순서

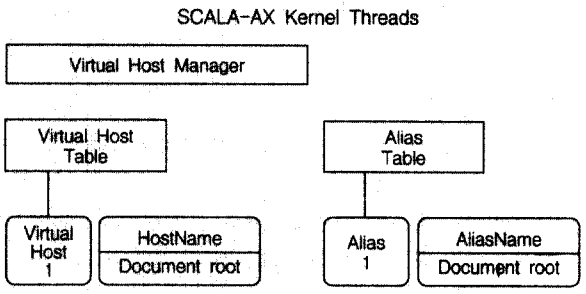
2.2.4 cgi 지원

SCALA-AX는 cgi API가 구현되어 동적인 데이터를 처리할 수 있다. 일반적인 경우 user program part의 웹서버가 클라이언트의 요청을 해석하여 사용자 환경에서 cgi 데이터를 처리할 수 있지만 이러한 과정을 모두 커널에서 수행함으로써 동적인 데이터 중 cgi의 처리 속도를 향상시킨다.

2.2.5 Virtual Hosting 지원

SCALA-AX는 같은 서버에서 서로 다른 호스트를 관리하는 virtual hosting 기능을 지원한다.

각기 다른 호스트들은 virtual host manager에서 관리하는 virtual host 표에 등록되어 해당되는 호스트 이름과 서비스 할 문서 루트를 지니게 되어 클라이언트의 요청에 대응한다. 또한 호스트 이름에 대한 확장을 고려하여 alias table에 동일한 문서의 루트를 갖는 다른 호스트를 지정할 수 있도록 하여 virtual host 표와 연결될 수 있다.



(그림 6) Virtual Hosting 지원

즉, alias 1이 virtual host 1과 연결되어 있다면 클라이언트 요청이 둘 중 어느 이름으로 도착하더라도 같은 문서 루트를 참조하도록 하였으며, 메모리에 저장하는 데이터도 한 가지로 하여 성능과 기능이 향상되었다.

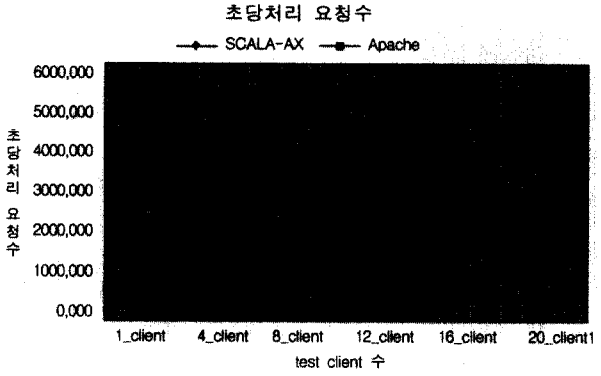
2.2.6 압축 로그 지원

SCALA-AX는 로그를 남기는데 사용되는 시간과 공간을 최소화하기 위해서, 이진형식의 로그를 지원한다. 이 방법은 로그를 W3C ascii 형식으로 하는 것보다 입출력 대역폭이나 디스크 공간을 적게 차지한다. 결국, 일반적인 ascii 파일보다 50% 작은 압축 로그를 만들게 된다.

3. 성능 비교

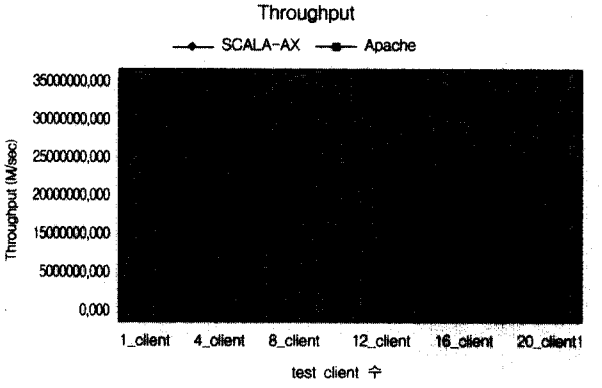
본 연구 및 시스템 개발 단계에서는 SCALA-AX의 성능을 평가하기 위해서, 800MHz의 dual PC processor를 사용하였다. 웹서버에 대한 요청은 zdnet의 web bench 3.0을 사용하였으며[13], 각각 gigabit LAN 환경에서 총 24대의 테스트 클라이언트 서버를 통해 이루어졌다. 웹서버의 성능 향상을 보여주는 기준으로 초당처리 요청수와 throughput을 측정하였다.

(그림 7)은 테스트 클라이언트당 초당 처리 요청수를 보여주고 있다. 아파치만으로 웹 서비스를 하였을 경우, 웹서버는 약 초당 1000개의 요청을 수행하였으며, SCALA-AX를 탑재하여, 웹 가속 및 캐싱 서비스를 하였을 경우, 대략 초당 5000개의 요청을 수행하였다. 아파치만으로 이루어진 웹서버에 비하여 SCALA-AX가 탑재된 웹서버는 초당처리 요청수 부분에서 약 5배 이상의 성능향상을 보임을 알 수 있다.



(그림 7) 테스트 클라이언트 당 초당처리 요청수

(그림 8)은 테스트 클라이언트당 throughput을 보여주고 있다. 아파치만으로 웹서비스를 하였을 경우, 웹서버는 대략 초당 5[Megabytes/sec]의 throughput을, SCALA-AX를 탑재하여, 웹가속 및 캐싱 서비스를 하였을 경우, 대략 초당 30[Megabytes/sec]의 결과를 낸다. 아파치만으로 이루어진 웹서버에 비하여 SCALA-AX가 탑재된 웹서버는 throughput 부분에서 약 6배 이상의 성능향상을 보임을 알 수 있다.



(그림 8) 테스트 클라이언트당 throughput

성능 시험결과로 나온 테스트 클라이언트당 초당 처리 요청수와 throughput의 그래프는 본 연구에서 구현한 SCALA-AX가 위에서 언급한 것과 같이 웹서버의 서비스 속도를 극대화하며, 대역폭을 절감해주고 서버의 과부하를 줄여서 클라이언트에게 더욱더 빠른 응답시간을 제공해주는 것을 보여 준다.

4. 결 론

본 논문에서 우리는 하드웨어나 통신선로의 업그레이드 없이 웹서버의 성능을 극대화시킬 수 있는 웹가속기(SCALA-AX)를 구현하였다.

전체적으로 웹서버시스템의 성능을 향상시키기 위하여 아래의 방법들이 사용되었다.

첫 번째, 기존의 웹서버가 수행하는 웹 서비스의 기능 중 실제 서비스의 대부분을 차지하고, 단순하고 반복적이지만 시스템 자원 및 시간을 크게 차지하는 기능을 SCALA-AX의 캐싱 기능을 통해서 수행하도록 하여 메모리 복사와 디스크 접근에서 일어나는 자원의 낭비를 줄여 웹서버의 성능을 향상시켰다.

두 번째, http 1.1프로토콜을 지원하여 연결지속 및 파이프라인 요청 기법을 구현함으로써 TCP 커넥션에서의 초기화 과정을 생략할 수 있고, 클라이언트의 요청을 pipeline을 이루게 함으로써 하나의 TCP 커넥션에 여러 개의 요청을 보내고, 서버는 이를 해석하여, 적절하게 응답해줄 수 있게 되어 시스템의 자원낭비 및 네트워크의 시간 지연요소를 줄일 수 있다.

세 번째, SCALA-AX를 리눅스 커널 패치를 통하여 커널 쓰레드 형태로 구현하여 다중작업구조의 속도 상 오버헤드를 해결하고 관련자원을 효율적으로 관리 할 수 있도록 하였다.

네 번째, 웹서버와 SCALA-AX가 물리적으로 하나의 머신 안에 존재 할 수 있도록 하여 캐싱서버를 별도로 설치함으로써 발생되는 content-freshness에 대한 문제를 해결하였다.

본 논문에서는 위의 방법들과 커널의 최적화를 통하여 사용자 프로그램 형태의 웹서버 프로그램에서 보다 5배 이상의 속도 향상을 얻었다.

향후과제로는 커널쓰레드로 구현된 SCALA-AX 프로그램을 더욱더 최적화함과 동시에 SCALA-AX의 캐시 알고리즘을 개선된 효율적인 알고리즘으로 대체함으로써 캐시의 단위시간당 접속률(히트수)을 높여 보다 효율적으로 H/W 및 S/W자원을 운용할 수 있도록 한다.

그리고, 웹서버 구성 사례별로 개발된 SCALA-AX를 적용해 봄으로써 다양한 사례에서도 능동적으로 적용될 수 있는 웹가속기로 발전시킨다.

참 고 문 헌

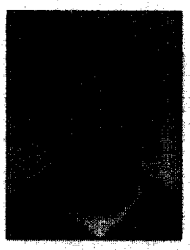
[1] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," the USENIX Symposium on Internet Technologies and Systems, December, 1997.

[2] A. Chankhunthod et al, "A Hierarchical Internet Object Cache," USENIX Technical Conference, pp.153-163, Janu-

ary, 1996.
[3] E. Levy, A. Iyengar, J. Song, and D. Dias, "Design and Performance of a Web Server Accelerator," IEEE INFOCOM '99, March, 1999.
[4] J. Song, E. Levy, A Iyengar, and D. Dias, "Design Alternatives for Scalable Web Server Accelerators," IEEE ISPASS, pp.184-192, 2000.
[5] A. Chankhunthod et al, "A hierarchical Internet Object Cache," USENIX Technical Conference, pp.153-163, January, 1996.
[6] V. Pai et al, "Locality-Aware Request Distribution in Cluster-based Network Services," ASPLOS-VIII, October, 1998.
[7] S. Gadde, J. Chase, and M. Rabinovich, "A Taste of Crispy Squid," Workshop on Internet Server Performance, 1998.
[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache : A Scalable Wide-Area Web Cache Sharing Protocol," SIGCOMM'98, 1998.
[9] J. Challenger, P. Dantzig, and A. Iyengar, "A Scalable and Highly Available System for Serving Dynamic Data at Frequently Accessed Web Sites," ACM/IEEE SC98, November, 1998.
[10] J. Challenger, A. Iyengar, and P. Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data," IEEE INFOCOM'99, March, 1999.
[11] A. Iyengar and J. Challenger, "Improving Web Server Performance by Caching Dynamic Data," USENIX Symposium on Internet Technologies and Systems, December, 1997.
[12] A. Iyengar, E. MacNair, and T. Nguyen, "An Analysis of Web Server Performance," GLOBECOM'97, November, 1997.
[13] zdnet, Inc. Webbench Benchmark Information. http://www.zdnet.com/etestinglabs/sto-ries/benchmarks/0,8829,2326243,00.html, 1998.
[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners Lee, "Hypertext transfer protocol-HTTP/1.1," IETF RGC 2068, January, 1997.

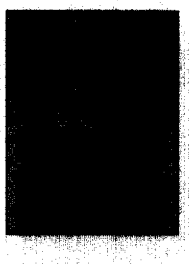
박 종 규

e-mail : baram2k@yonsei.ac.kr
1998년 원광대학교 제어계측공학과 졸업(학사)
2000년 원광대학교 대학원 제어계측공학과(공학석사)
2000년~현재 연세대학교 전기전자공학과 박사과정
관심분야 : 실시간 OS, QoS알고리즘, 웹가속기



민 병 조

e-mail : bjmin@c-eisa.com
1998년 연세대학교 전기공학과 졸업(학사)
2000년 연세대학교 대학원 전기공학과(공학석사)
2000년~현재 연세대학교 전기전자공학과 박사과정
관심분야 : 웹 서버, 웹 가속기



임 한 나

e-mail : hanna482@yonsei.ac.kr
2001년 연세대학교 기계전자공학부 전자전공 졸업
2002년 연세대학교 대학원 전기전자공학과 석사과정
관심분야 : Network in Linux, Linux kernel programming and webserver acceleration

장 휘

e-mail : wchang@c-eias.com
1988년 서울대학교 기계설계공학과 졸업(학사)
1989년 Univ. of Michigan at Ann Arbor 대학원 기계공학과(공학석사)
1993년 Univ. of Michigan at Ann Arbor 대학원 전산역학(공학박사)
1994년~1996년 Minnesota Supercomputer Center, Minneapolis Research Associate
1996년~1997년 삼성자동차기술연구소 선임연구원
1997년~2000년 서울산업대 조교수
2000년~현재 씨아이사(C-EISA) CEO
관심분야 : 웹가속기, 인터넷 보안, 홈 네트웍

김 학 배

e-mail : hbkim@yonsei.ac.kr
1988년 서울대학교 전자공학과 졸업(학사)
1990년 미국 미시간대 전기 및 컴퓨터공학과(EECS)(공학석사)
1994년 미국 미시간대 전기 및 컴퓨터공학과(EECS)(공학박사)
1994년~1996년 미국 National Research Council(NRC) Research Associate at NASA Langley Research Center
1996년~2000년 연세대학교 전기컴퓨터공학과 조교수,
2000년~현재 연세대학교 전기전자공학부 부교수
관심분야 : 실시간 및 내장형 시스템, 인터넷 웹서버 기술, 디지털 시스템 고장포용 및 신뢰도 평가 분야