리눅스 커널 쓰레드에 기반한 SSL 웹 암호화 가속기 개발

남 의 석[†] · 민 병 조[††] · 김 학 배[†††]

요 약

인터넷의 상업적인 사용이 크게 증가되고 관련된 기업, 기관, 그리고 개인들에게 웹 트랜잭션에 관한 보안성 확보가 중대한 문제가 되고 있다. 이러한 인터넷 트랜잭션상에서의 보안성을 확보하기 위한 여러 가지 네트워크 프로토콜 중에서 SSL이 인터넷 웹 상의 네트워크 보안에 관련된 가장 널리 사용되는 프로토콜이다. 하지만, SSL을 이용한 웹 서버의 경우 사용자들이 SSL을 사용해서 서버에 접속할 때의 서버부하가 크게 증가되어 전반적인 웹 서비스 속도를 크게 저하시키게 되는 문제점이 있다. 일반적인 웹 서비스에 해당되는 HTTP 프로토콜의 경우는 서버와 클라이언트가 두번의 메시지 핸드쉐이킹으로 요청과 응답과정을 수행하지만, SSL의 경우는 사용자 인증 등의 협상(negotiation)과정과 메시지 암호화/복호화에 관련해 최소 세번에서 여섯 번까지의 요청과 응답 핸드쉐이킹을 하게 된다. 이로 인한 네트워크 메시지 및 핸들링 관련 오버헤드가 존재하며 서버의 자체에는 주고받는 모든 데이터에 대한 복호화, 인증, 무결성 검사 등의 과정에 대한 오버헤드가 존재한다. 따라서, 웹의 상업적인 이용을 위해 필수적인 SSL의 원활한 사용을 위해서는 SSL에 관한 서버 처리량을 높여줄 수 있는 방법이 필요하다. 웹 서버단에서 SSL의 처리량을 높이려는 기존의 연구는 크게 두 가지로 구분된다. SSL에 관한 연산작용을 전담하는 co-processor를 설치하는 방법과 독립형 SSL 어플라이언스 기술을 사용하는 것이다. 하지만 두 가지 방법 모두 관리비용 측면에서 효율적이지 못하며, 또한 향후 SSL 프로토콜의 변화, 주요 암호화 알고리즘의 추가/변경/업데이트 및 웹 서버 프로그램의 변화에 유연하게 대처할 수 없고, 하드웨어 어플라이언스 자체가 전체 웹 서비스 속도의 병목(bottle neck)이 될 수 있으며, 시스템 장애시 단일 장애지점(single point of failure)이 될 수 있고, 또한, SSL 어플라이언스 이외의 백엔드 웹 서버들에게 클라이언트의 인증 정보 등 SSL 연산에 대한 정보가 공개되지 못해, 서비스가 투명하지(transparent) 못하다는 단점들이 있다. 본 논문에서는 부가적인 하드웨어나 장비의 도움 없이 SSL 웹 서버의 처리 속도 향상을 위하여 SSL 가속 모듈을 사용자 레벨이 아닌 리눅스 커널 레벨에서 구현하는 기법을 제안한다. 즉, 스위칭 시간 및 프로세싱 자원의 불필요한 낭비를 줄이기 위해 커널 쓰레드를 사용한다. 또한, 각 사용자 쓰레드에 메모리 할당으로 인한 불필요한 메모리 사용을 줄이기 위하여 모든 응답들은 단일 메모리 단위로 관리되어진다. 제안된 기법의 성능평가를 위하여 ZDNet의 WebBench 4.1을 이용하여 기존의 사용자 영역에서 SSL 처리 방식과 제안된 기법의 처리방식을 비교한 결과 2배 정도의 처리속도가 증가함을 확인할 수 있었다. 이를 통해, 제안된 기법은 전용 하드웨어의 추가 또는 교체, 전체 웹 서비스 구조를 변경시킬 필요가 없는 방식으로 성능과 효과를 확인할 수 있었다.

Development of a Linux Kernel-Thread Secure Socket Layer (SSL) WEB Accelerator

Euiseok Nahm[†] · Byungjo Min[††] · Hagbae Kim[†††]

ABSTRACT

We propose a kernel-level Secure Socket Layer (SSL) accelerator which significantly increases the processing rate of the SSL operation on a webserver. Because most of conventional SSL's applied user threads for processing http requests, the CPU overhead, where the slow processing time are unavoidable. Our SSL accelerator, embedded in the kernel level, utilizes kernel threads as same number of CPUs. Thus, the webserver can allocate total CPU resources to handle the requests, which not only reduces switching time incurred from using system calls but also minimizes the send/receive operation on the network layer. To validate the effectiveness of our scheme, we evaluate its performance in comparison with the conventional Apache with/without our SSL accelerator. Consequently, the SSL accelerator improves the webserver performance by up to 200%.

키워드 : SSL 가속기(SSL Accelerator), 웹 서버(Web Server), 커널 쓰레드(Kernel Thread), 아파치(Apache)

1. 서 론

인터넷의 상업적인 사용이 크게 증가되고 관련된 기업, 기관, 그리고 개인들에게 웹 트랜잭션에 관한 보안성 확보가 중대한 문제가 되고 있다. 이러한 인터넷 트랜잭션상에서의 보안성을 확보하기 위해, IPSec(IP Security), SET(Secure Electonic Transaction), SSL(Secure Sockets Layer), TLS(Transport Layer Protocol) 등 여러 가지 네트워크 프로토콜들이 사용되고 있다. 이러한 보안 관련 프로토콜 중, IPSec

† 정 회 원 : 극동대학교 정보통신학부 교수
†† 준 회 원 : 연세대학교 대학원 전기전자공학과
††† 정 회 원 : 연세대학교 전기전자공학과 교수
논문접수 : 2003년 5월 6일, 심사완료 : 2003년 8월 14일

은 IP 레이어에서 데이터를 암호화/복호화해서 보안성을 강화하는 프로토콜이다. 그러나 모든 라우터, 스위치 등의 네트워크 장비가 모두 IPSec을 지원하지는 않기 때문에 IPSec의 모든 기능이 구현되기 힘들다. SET의 경우, 프로토콜 자체가 신용카드라는 특정 응용에 맞게 프로토콜 디자인과 구현이 이루어졌기 때문에 다른 응용에 적용에는 많은 한계가 있다. 반면, SSL 프로토콜은 웹 브라우져 회사였던 넷스케이프에 의해서 만들어졌으며 인터넷 익스플로어 및 넷스케이프 등의 주요 웹 브라우져가 SSL 및 TLS에 관한 프로토콜 스택을 거의 완벽하게 지원하여 사용자가 특별한 프로그램을 다운로드받지 않아도 넷스케이프와 익스플로러가 자동으로 보안 채널을 연결해 준다. 따라서, SSL 및 TLS는 주요 웹사이트의 인증, 신용카드 및 주요 로그인 패스워드 등 주요 데이터의 암호화, 데이터의 무결성(integrity) 보증 등 주요 전세계 전자상거래 웹 트랜잭션의 근간으로 인터넷 웹상의 네트워크 보안에 관련된 가장 널리 사용되는 프로토콜이다.

하지만, SSL을 이용한 웹 서버의 경우 사용자들이 SSL을 사용해서 서버에 접속할 때의 서버부하가 크게 증가되어 전반적인 웹 서비스 속도를 크게 저하시키게 되는 문제점이 있다. 일반적인 웹 서비스에 해당되는 HTTP 프로토콜의 경우는 서버와 클라이언트가 두 번의 메시지 핸드쉐이킹으로 요청과 응답과정을 수행하지만, SSL의 경우는 사용자 인증 등의 협상(negotiation) 과정과 메시지 암호화/복호화에 관련해 최소 세 번에서 여섯 번까지의 요청과 응답 핸드쉐이킹을 하게 된다[1]. 이로 인한 네트워크 메시지 및 핸들링 관련 오버헤드가 존재하며 서버의 자체에는 주고받는 모든 데이터에 대한 복호화, 인증, 무결성 검사 등의 과정에 대한 오버헤드가 존재한다. 또한, SSL의 암호화에는 기밀성과 무결성을 높이기 위해 RSA(Rivest-Shamir-Adleman)와 같은 공개키 암호시스템(PKI : Public Key Infrastructure)에 기반한 현대 암호화 알고리즘이 사용되는데, 이는 매우 큰 수(2^{56}, 2^{128}, 2^{1024})에 관한 나머지 연산(modular) 및 소인수(prime number) 연산이 필요로 하며, 이러한 연산은 평문(plain text)을 주고받는 HTTP에 비교하여 서버 CPU에 매우 큰 부하(serious burden)가 된다[2]. 따라서, 웹의 상업적인 이용을 위해 필수적인 SSL의 원활한 사용을 위해서는 SSL에 관한 서버 처리량을 높여줄 수 있는 방법이 필요하다.

웹 서버단에서 SSL의 처리량을 높이려는 기존의 연구는 크게 두 가지로 구분된다. 첫째는 SSL에 관한 연산작용을 전담하는 보조 프로세서(co-processor)를 설치하는 방법이다[3]. 초기에 제조업체들은 서버에 전용의 보조 프로세서를 장착하여 암호화 프로세싱에 대한 서버의 부담을 줄임으로써 SSL 문제를 해결하고자 하였다. 즉, SSL에 관련된 프로세싱을 아식(ASIC)화된 PCI 카드로 전담시켜, 실제 SSL 암호화/복호화시 전체 서버의 성능을 크게 저하시키는 CPU 오버헤드를 없애는 것이다. 이러한 하드웨어적 접근방식은 웹 서버에 추가의 프로세싱 능력을 제공하여 안전한 데이터를 유지하기 위한 복잡한 작업을 수행할 수 있게 한다. 또한 이러한 SSL 전담 하드웨어의 추가로 웹 서버의 응답시간을 빠르게 하고 서버가 초당 처리할 수 있는 트랜잭션 수를 늘리는 것을 도와준다. 하지만 서버 팜(farm)과 같은 경우에는 각 웹 서버마다 설치되어야 하기 때문에 관리비용 및 방법에서 효율적이지 못하며, 또한 향후 SSL 프로토콜의 변화, 주요 암호화 알고리즘의 추가/변경/업데이트 및 웹 서버 프로그램의 변화에 유연하게 대처할 수 없다는 단점을 지니고 있다. 두 번째는 독립형 SSL 어플라이언스 기술을 사용하는 것이다. 이는 네트워크 구성도에서 모든 웹 서버의 상단에 위치한다. 이러한 방식은 많은 계산을 필요로 하는 작업을 독립된 SSL 전용장비로 고립시킴으로써 각 웹 서버에서 SSL에 관한 연산을 하지 않게 하여, 웹 서버의 부담을 줄여주고, 웹 트래픽 전송 속도를 높인다. 하드웨어 방식과의 차이점은 PCI 카드와 같은 한정된 메모리/CPU를 가진 구현 방식에 비해, 메모리나 스카시 하드와 같은 자원의 이용이 자유롭다는 점이다. 하지만, SSL 어플라이언스 자체가 전체 웹 서비스 속도의 병목(bottle neck)이 될 수 있으며, 시스템 장애시 단일 장애지점(single point of failure)이 된다. 또한, SSL 어플라이언스 이외의 백엔드 웹 서버들에게 클라이언트의 인증 정보 등 SSL 연산에 대한 정보가 공개되지 못해, 서비스가 투명하지(transparent) 못하다.

본 논문에서는 부가적인 하드웨어나 장비의 도움 없이 SSL 웹 서버의 처리 속도 향상을 위하여 SSL 가속 모듈을 사용자 레벨이 아닌 리눅스 커널 레벨에서 구현하는 기법을 제안한다. 즉, 스위칭 시간 및 프로세싱 자원의 불필요한 낭비를 줄이기 위해 커널 쓰레드를 사용한다. 또한, 각 사용자 쓰레드에 메모리 할당으로 인한 불필요한 메모리 사용을 줄이기 위하여 모든 응답들은 단일 메모리 단위로 관리되어진다.

제안된 기법의 성능평가를 위하여 ZDNet의 WebBench 4.1을 이용하여 기존의 사용자 영역에서 SSL 처리 방식과 제안된 기법의 처리방식을 비교한 결과 2배 정도의 처리속도가 증가함을 확인할 수 있었다[5]. 이를 통해, 제안된 기법은 전용 하드웨어의 추가 또는 교체, 전체 웹 서비스 구조를 변경시킬 필요가 없는 방식으로 성능과 효과를 확인할 수 있었다.

2. 기존의 웹 서버의 SSL 처리

본 절에서는 리눅스 커널레벨에서의 소프트웨어 구현방식에 대해 설명하고, 사용자 레벨에서 구현된 기존 SSL 처

리방식을 커널 레벨로 구현하였을 경우의 차이점과 성능이득에 대해 설명한다.

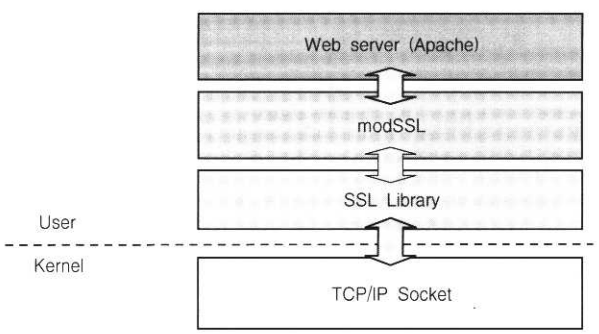
2.1 리눅스 커널의 특징

리눅스(linux)는 1991년 이후 새로이 등장한 운영체제로서 오픈 소스이다. 즉 여러 개발자들의 자발적인 참여로 이루어진 운영체제의 핵심인 커널 개발 속도가 매우 빨라 새로운 메모리, 파일 시스템, 네트워크 서비스 등에 관한 신기술에 매우 유연하다. 이러한 장점들에 힘입어, 서버 플랫폼에서 크게 각광받아 현재 전세계 웹 서버 플랫폼의 약 45% 정도를 점유하고 있으며, 앞으로도 서버 플랫폼에서의 점유율이 높아질 것으로 예상된다. 또한 커널 코드가 완전히 공개되어 있어, 커널 코드가 공개되지 않거나, 공개되더라도 비싼 라이센스비가 포함되는 기타 운영체제에 비해, 커널 레벨에서의 어플리케이션을 자유롭게 만들어 사용할 수 있고, 이러한 커널 어플리케이션이 런타임(run time)시에 동적으로 관리되는 장점이 있다.

리눅스는 멀티태스크와 멀티유저, 멀티플랫폼을 지원하며 커널의 각 기능이 완전히 통합되어 있는 monolithic 커널이다. 또한, 실시간 시스템을 지원하는 커널과 달리 비선점형(non-preemtive) 커널로서 하나의 커널 쓰레드가 CPU를 점유하고 있을 경우, 그 쓰레드가 CPU 스케줄링을 포기하지 않는 한, 다른 쓰레드로 CPU가 할당되지 않는다는 특징을 지닌다.

2.2 기존 웹 서버의 사용자 레벨의 SSL 처리 방식

(그림 1)은 일반적인 웹 서버의 SSL 처리 방식을 나타낸다. 대부분의 웹 서비스에서의 SSL 처리 구조는 실제 웹 요청을 처리하는 아파치(Apache) 웹 서버 프로그램과 SSL 요청처리 아파치 모듈 프로그램인(modSSL), 그리고 실제 SSL에 관한 암호화/복호화 알고리즘 라이브러리인(OpenSSL)로 이루어진다[5, 7].



(그림 1) 기존 웹 서버의 SSL 처리 구조

이러한 일반적 HTTPS 서버는 SSL을 지원하기 위해서 사용자 레벨에서 TCP/IP 소켓을 열고 그 위에서 SSL을 라이브러리를 로딩하여, 그 위에 다시 HTTP를 제공하는 서버를 올리는 방법으로 구현된다. 이러한 구성을 취할 경우 일반적인 HTTP보다 훨씬 복잡한 데이터 전송을 하는 HTTPS는 커널 스위칭이 많아지기 때문에 CPU 오버헤드가 존재한다. 또한 사용자 레벨에서는 OS의 멀티테스킹으로 SSL의 복잡한 알고리즘 연산 수행도중 스케줄링을 당할 수 있기 때문에 캐쉬의 효용성이 감소되며 다른 프로세스에 의해 우선순위를 빼앗길 수 있다.

또한, 기존의 SSL 처리 방식의 경우, 브라우져를 통한 사용자들의 요청이 서버에 도달하면, 각 요청에 대해 처리 쓰레드를 실행하여, 각 쓰레드가 하나의 요청을 전담한다. 이러한 처리구조에서는 동시 접속자 수가 많아지게 되면, 쓰레드 수의 증가로 이에 대한 서버의 메모리 할당이 증가되고, 각 쓰레드마다 할당되는 CPU 점유시간은 감소하게 된다. 결과적으로, 서버의 메모리 및 CPU 시간과 같은 자원을 많이 사용하게 되어, 결과적으로 서버의 요청 처리 능력이 감소하고, 사용자의 응답시간은 늦어지게 된다[6]. 이러한 기존 방식에서 사용되는 쓰레드는 리눅스 커널 쓰레드와는 달리, 사용자 영역에서 런칭되게 되어, CPU에 대한 독점적, 비선점적 사용이 불가능하다. 그러므로 같은 아파치 프로그램에서 만들어진 쓰레드들이 CPU에 대한 같은 수준의 우선순위를 같게 되어 CPU에 의해 모두 같은 시간을 할당받게 되고 많은 쓰레드가 동작중인 경우 이러한 CPU 스위칭 시간이 증가하여, 처리속도에 커다란 지연요소로 작용한다. 또한 SSL 세션 기능을 사용할 경우, HTTPS 프로토콜의 경우, 접속 과정에서 최소한 두 번 송수신 데이터를 주고 받는다. 일반적인 HTTPS 서버는 사용자 레벨에서 최소 두 번의 수신(receive) 시스템 콜과 두 번의 송신(send) 시스템 콜을 수행해야만 접속이 이루어지기 때문에 이로 인한, 커널 모드 스위칭 오버헤드가 존재한다[2].

3. 제안된 커넬레벨 SSL 처리 방식

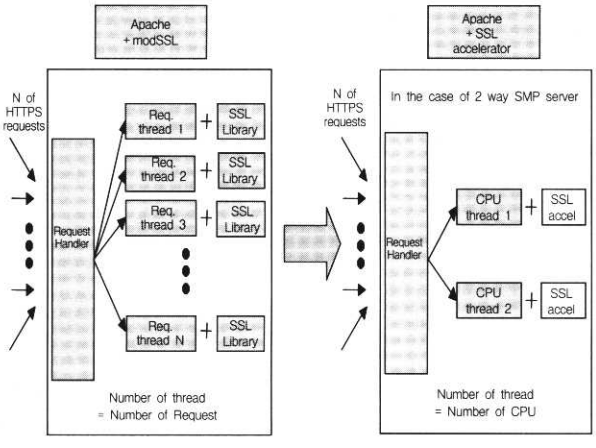
기존 SSL 처리구조의 단점을 극복하기 위해서 본 논문에서는 SSL 처리 구조를 리눅스 커널 레벨에서 설계하여 구현하였으며, 설계상의 핵심 고려사항 및 주요 성능개선 요인은 다음과 같다.

3.1 SSL 요청 처리구조의 변화

사용자 영역 멀티 쓰레드 처리 구조를 커널 영역 단일 쓰레드 구조로 변경하고 매우 중요한 코드 블록을 non-preemptive하게 하여 CPU를 아주 짧은 시간동안 독점하여 요청을 처리한다. 또한, 각 요청마다 새로운 쓰레드를 구동시키는 구조 대신 요청을 받아들이는 커널 쓰레드를 하나씩으로 처리하도록 한다. 즉, 각 요청들이 들어오면 이들을 하나의 큐에 모으고 이 큐에서 SSL 처리 커널 쓰레드가 각 요청들을 처리하는 구조로 한다. 단일 쓰레드에 하나의 큐를

사용해서 멀티 쓰레드보다 처리 속도가 느려 보일 수 있지만 CPU를 독점할 수 있는 커널 영역의 쓰레드로 구현되어 있다는 점과 짧은 시간에 동작되는 매우 중요한 처리 코드를 블럭화하여 이러한 단위 블록들이 비선점형으로 실행되게 함으로써 처리 속도를 극대화할 수 있다[8].

(그림 2)는 두 방식의 SSL 요청 처리 구조의 차이를 표현한 것이다. SSL 프로토콜 요청 핸들러는 한 개 이상의 CPU를 가진 경우 최적의 성능을 발휘할 수 있도록 각 CPU당 하나의 커널 쓰레드를 할당한다. 클라이언트가 처음 접속되면 그 접속을 받은 CPU가 각 커널 쓰레드를 할당하고, SSL 접속에 관련한 모든 것을 담당한다. 결과적으로 SSL 요청 핸들러의 경우 SSL 처리를 담당하는 커널 쓰레드의 개수가 CPU 개수 만큼이기 때문에 많은 요청이 쇄도해도 너무 많은 프로세스 동작으로 인한 CPU 및 메모리 오버헤드가 없다. 즉, SSL 요청 방식의 변화로 인해, 전체 시스템의 속도 저하를 막을 수 있으며, 이로 인한 성능 이득을 얻는다[9].



(그림 2) 제안된 SSL의 처리 구조

3.2 사용자 영역 스위칭의 감소

시스템 콜이란 사용자 영역에서 커널 영역으로 어떤 동작을 수행해 주기를 요청하면 이를 커널에서 수행하고 다시 사용자 영역으로 응답해주는 운영체제의 구현방식을 의미한다. 이러한 시스템 콜의 경우, 기본적인 연산은 단순히 어떤 값을 변경하거나 그 값을 리턴하는 구조로 되어 있지만 읽기/쓰기 등과 같은 파일/네트워크 연산에 관련된 시스템콜의 경우, 커널과 사용자 각 영역에서의 메모리 접근 방식의 차이로 인해 2~3회의 버퍼 복사 연산이 더 발생하여 커널 레벨에서 직접 읽기/쓰기 연산을 하였을 때보다, 메모리 오버헤드, 처리시간 지연 등이 초래된다. 또한, SSL 세션 기능을 사용할 경우, 접속과정에서 최소한 두 번 송·수신 데이터를 주고 받기 때문에 사용자 레벨에서 최소 두 번의 받기(Receive) 시스템 콜과 두 번의 보내기(Send) 시스템 콜을 수행해야 한다. 따라서, 기존의 SSL 처리구조에서는 파일, 메모리, 시간 관련 연산 등에 대한 시스템 콜을 사용하였지만, 커널로 구현하였을 경우에는 시스템 콜로 구현하는 것이 아니라 직접 커널 함수를 사용하므로, 시스템 콜을 사용하지 않고 커널 모드와 사용자 영역 모드와의 스위칭이 필요 없다.

3.3 네트워크 연산의 커널화

사용자와 서버와의 주고받는 데이터의 보내기(send)/받기(receive) 연산을 직접 커널에서 수행하여 사용자 영역에서 하는 경우에 비해 호출되는 함수 콜이 적어지게 되고, 사용자 영역과 커널 영역사이로의 버퍼 카피 연산이 줄어들게 되어, 이로 인한 이득을 얻을 수 있다.

제안된 SSL 처리구조는 SSL을 전담하는 커널 쓰레드를 CPU 별로 두고, 각 쓰레드가 SSL 커넥션의 암호화/복호화를 하나의 큐(queue)에서 처리하도록 하며, 사용자 레벨과 달리 커널 쓰레드 자신이 CPU를 풀어주지 않는 한, 다른 프로세스에게 선점되지 않는다. 이 특징을 이용하여, SSL 커넥션의 처리가 가장 효율적인 스케줄링을 하도록 스케줄 시점이 SSL 메시지교환의 상태에 따라 결정되어 있다, 이로 인한 CPU 스케줄링 시간의 이득을 얻는다. 또한 불필요한 메모리 카피나 커널영역과 사용자 영역 스위칭에 대한 오버헤드가 없고 네트워크 연산을 모두 커널영역에서 수행하도록 함으로 처리속도를 증가시킨다.

4. SSL 커널 가속 모듈 S/W 구조

본 절에서는 위에서 언급한 SSL 요청처리 방식을 이용하여 구현된 SSL 커널 가속 모듈의 구성과 SSL 암호화/복호화 알고리즘을 담당하는 커널 레벨 SSL 라이브러리의 구현방식에 대해 설명한다.

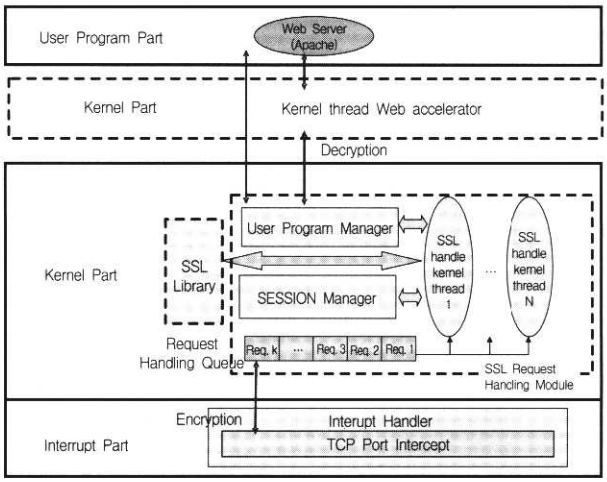
4.1 SSL 처리 핸들러 구조

SSL 커널 가속 모듈은 사용자 인증, 암호화/복호화, 메시지 무결성 등을 지원하는 SSL 라이브러리 모듈과 클라이언트와의 통신, 웹 서버와의 연동, 쓰레드 관리 등의 일을 하는 SSL 요청 처리 모듈로 구성된다.

SSL 모듈은 보안 알고리즘을 수행하기 위한 코드들이며 오픈 소스인 OpenSSL을 기반으로 구성된다[5]. 이 모듈은 OpenSSL이 커널에서 정상적으로 동작하게 하기 위해 사용자 레벨에서는 존재하지만 커널 영역에서 존재하지 않는 외부 함수나 시스템 콜들의 구현을 포함하고 있다.

SSL 요청 처리 모듈은 HTTPS 접속을 받아서 서로의 데이터를 협상(negotiation)하고 클라이언트에게 온 메시지를 복호화하며 웹 서버로부터 전해진 데이터를 암호화해서 다시 클라이언트에게 보내는(send) 역할을 한다. SSL 커널 가속 모듈은 커널 SSL 라이브러리 부분과 클라이언트와의 통신, 웹 서버와의 연동, 쓰레드 관리 등의 일을 하는 SSL

요청 핸들러로 구성된다. (그림 3)은 SSL 커널 가속 프로그램의 상세 구조를 도식화한 것이다.



(그림 3) SSL 커널 가속 프로그램의 상세 구조

4.2 커널 레벨의 SSL 암호화/복호화 라이브러리 모듈

OpenSSL은 SSLeay를 기반의 공개용 암호화 라이브러리이다. 미국에서 상용으로 개발된 RSA 알고리즘을 거의 완벽하게 호환하면서 미국 밖에서도 사용 가능한 프리웨어로 소스코드 및 각 암호화 알고리즘 자체가 완벽히 공개되어 있다. 본 연구에 사용된 이유도 오픈 소스라는 점과 많은 아파치 웹 서버와 같이 사용되면서 안정성과 호환성에 대한 검증을 받았다는 것을 들 수 있다. 약 40가지 이상의 암호화 알고리즘과 4가지 이상의 디지털 서명 알고리즘, 3가지 메시지 인증코드(MAC) 알고리즘이 구현되어 있다. 본 논문에서 사용한 OpenSSL은 <표 1>과 같다.

〈표 1〉 본 논문에서 사용된 OpenSSL

이 름	OpenSSL-0.9.6d
프로토콜 지원	SSLv2, SSLv3, TLSv1
공개키구조 관련(PKI) 알고리즘	RSA, EDH, DH, etc
관습적 암호화 알고리즘 (Conventional Encryption Algorithm)	DES, 3DES, RC4, RC2, RC5, etc
메시지 인증 알고리즘 (Message Authentication Code Algorithm)	MAC, HMAC, SHA, SHA1, etc

본 논문에서 수행된 커널 SSL 라이브러리 모듈은 보안 알고리즘을 수행하기 위한 코드들이며 OpenSSL을 기반으로 구성된다. 이 모듈은 OpenSSL이 커널에서 정상적으로 동작하게 하기 위해 커널레벨로 구현된다. 이는 실제 사용자 레벨에서의 SSL 라이브러리 프로그램은 read, write, send, recv와 같은 시스템 콜을 통해 커널과 인터페이스를 가지는데 반해, 커널레벨의 SSL 라이브러리를 구현할 경우는 이러한 시스템 콜을 이용한 인터페이스, 독점적 CPU 사용 및 오버헤드 제거 등을 통해 보통 IO 오퍼레이션의 속도를 크게 증가시킬 수 있다. 그러므로 CPU 집중적인 SSL 프로토콜의 특성상, CPU 및 네트워크 데이터 속도에 중요한 recv, send 등의 함수를 커널레벨에서 수행함으로써 전체 성능향상에 도움이 된다. 이러한 경우, 가장 중요한 문제는 기존의 SSL 라이브러리가 커널에서 완벽하게 동작할 수 있도록 커널 레벨로 구현해야 하는 것으로, 기존 사용자 레벨에서 gcc 라이브러리를 통해서 사용되는 OpenSSL의 라이브러리를 커널레벨에서 동작할 수 있도록 바꾸는 것이다. 이러한 구현을 위해선 기존 OpenSSL 라이브러리 소스에서, 사용자 영역의 여러 라이브러리와의 관련성을 없애고, 라이브러리 전체가 모두 커널에서 동작하도록 커널 레벨로 수정해야 한다. 또한, 사용자 영역 라이브러리와의 연관성(dependency)이 없기 때문에, 메모리/파일/시간/네트워크 오퍼레이션에 대한 함수 등 몇 가지가 커널에서는 존재하지 않으므로, 이 기능들을 커널에서 동작되도록 직접 구현하여야 한다.

결과적으로 표준(standard) I/O, 시간에 관한 라이브러리(time, localtime, gmtime), 메모리 할당 및 해제에 관한 함수호출 그리고 네트워크 소켓에 관한 함수 호출들을 커널 레벨에서 구현하여 SSL 라이브러리에서 사용할 수 있도록 하여야 한다. 이러한 커널 레벨의 SSL 라이브러리 구현 과정은 다음과 같이 요약된다.

- 표준 헤더 파일들을 모두 커널 헤더 파일로 변환
- 표준 라이브러리 및 공유 라이브러리와의 링킹(linking) 및 의존도(dependency) 삭제
- 커널에 구현되어 있는 API만을 사용하여 SSL 라이브러리를 구현
- Apache와 같이 멀티 프로세스에 맞춰진 SSL 라이브러리 구현 방식을 커널쓰레드 방식으로 구현

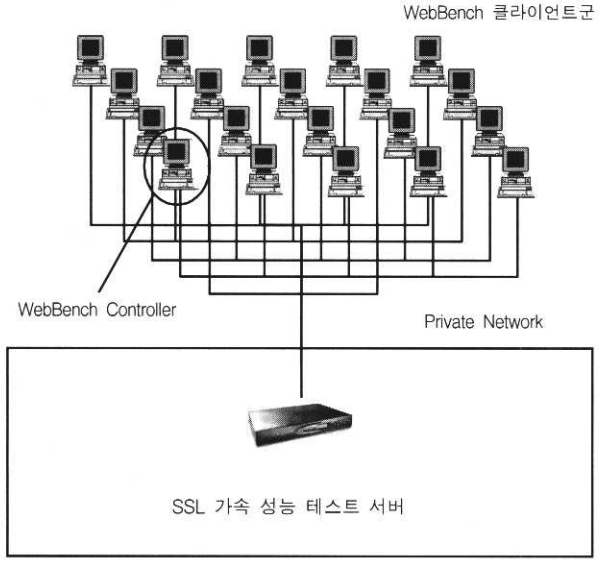
5. 성능 평가

제안된 기법의 SSL 가속 커널 모듈의 SSL 처리속도 성능 검증을 위하여 기존 SSL 웹 서버와(Apache + modSSL + OpenSSL) 제안된 SSL 가속 커널 모듈을 탑재한 서버의 성능을 비교하였다. 성능 평가에 사용된 서버의 주요 사양은 <표 2>와 같다.

〈표 2〉 테스트에 사용된 서버 사양

구 분	P3 서버
CPU	Dual P3-933Mhz
Memory	512M
Hard Disk	SCSI Ultra160 LVD/SE
Lan Card	3Com Giga lan
배포판	Redhat 7.2
SSL 라이브러리	OpenSSL0.9.6d/또는 본 웹가속기의 SSL 라이브러리
리눅스 커널	2.4.18

테스트 환경은 부하를 거는 서버로 10대의 서버가 사용되었고, 같은 하드웨어의 서버에서 기존 SSL 처리구조와 본 SSL 가속 커널 모듈을 번갈아 적용하면서 테스트되었다. 웹 서버 프로그램으로는 가장 많이 사용되고 있는 아파치(apache)를 사용했으며, 레드햇(Redhat) 7.2에 기본으로 설치되어 있는 1.3.20으로 테스트를 수행하였다. Test 프로그램은 Zdnet의 Webbench 4.1(128bit)을 사용하였고 협상 암호문(Negotiation cipher)은 EDH-DSS-DES-CBC3-SHA을 이용하였다[4]. 또한, 테스트한 SSL 버전은 TLS1.0, SSL 암호화 비트는 128bit를 사용하였다. (그림 4)는 HTTPS 서비스에 대한 웹가속기의 성능을 테스트하기 위한 환경 구성도이다. gigabit LAN 환경하에서 총 10대의 테스트 클라이언트 서버를 통해 이루어졌다.

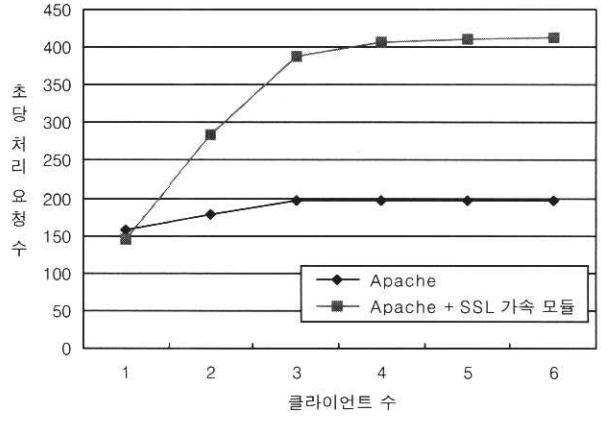


(그림 4) HTTPS 성능 테스트를 위한 구성도

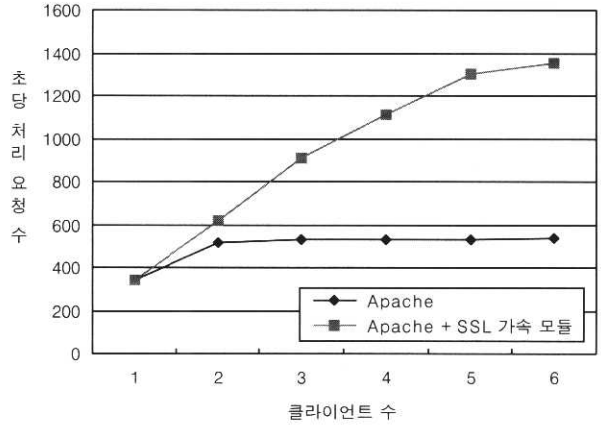
테스트는 HTTP 1.0, HTTP 1.1(persistence connection, pipeline) 그리고 두 가지를 혼합한 총 4가지 경우에 대해서 수행하였다.

- case 1 : HTTP 1.0 100%
- case 2 : HTTP 1.1 지속연결(persistence) 100%
- case 3 : HTTP 1.1 연속요청(pipeline) 100%
- case 4 : HTTP 1.0/1.1 혼합

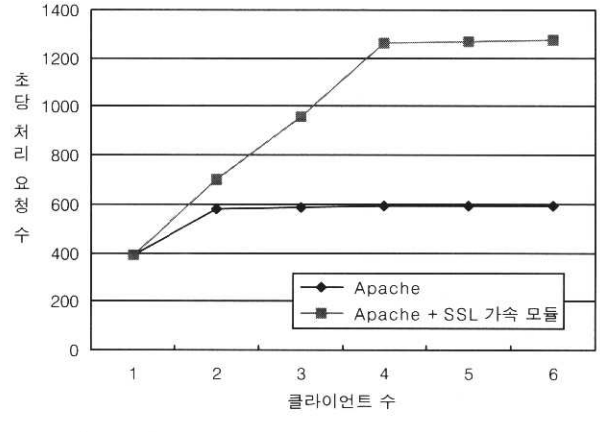
HTTP 1.0에 비교하여 지속 연결과 연속 요청 기능이 추가된 HTTP 1.1상에서 본 가속 모듈이 어느 정도 유효한 지에 대하여 테스트하였다. 클라이언트를 1대부터 10대까지 증가시키며 부하를 주고 그에 따라 1초당 처리되어 나온 요청의 수를 기록하였다. 즉, 웹 서버의 성능 향상을 보여주는 기준으로 초당 처리수를 측정하였다. (그림 5)~(그림 8)은 각각의 경우 테스트 클라이언트당 초당 처리 요청수를 보여주고 있다. 어느 경우나 제안된 기법을 적용한 서버가 기존 Apache 웹 서버보다 초당 처리수가 많음을 알 수 있다.



(그림 5) 테스트 클라이언트당 초당 처리 요청수 (HTTP 1.0 100%인 경우)



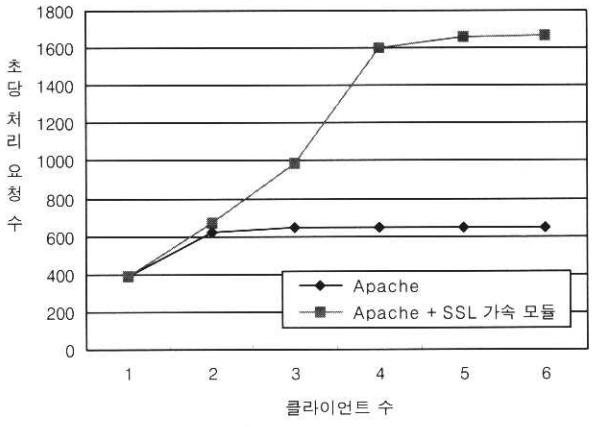
(그림 6) 테스트 클라이언트당 초당 처리 요청수 (HTTP 1.1 persistent 100%)



(그림 7) 테스트 클라이언트당 초당 처리 요청수 (HTTP 1.1 pipeline 100%)

성능 시험 결과, 각 경우별로 차이는 있었지만 125%~200% 정도의 초당 처리 요청수가 증가했으며 제안된 SSL 가속 커널 모듈의 처리 성능이 기존 웹 서버 보다 뛰어남

을 알 수 있다. 결국 기존 웹 가속기에 HTTPS 요청 처리 핸들러 및 SSL 라이브러리 기능을 추가함으로써, HTTPS 연결시 처리 성능을 향상시킬 수 있다.



(그림 8) 테스트 클라이언트당 초당 처리 요청수 (HTTP 1.0/1.1)

6. 결 론

본 논문에서는 복잡한 암호화 과정 및 접속시 메시지 핸드쉐이킹 증가로 웹 서버의 처리 성능이 현격히 감소하는 기존 SSL 처리의 문제점을 해결하기 위한 방법으로 부가적인 하드웨어나 장비의 도움 없이 시스템 내부 구조를 변화시켜, SSL 웹 서버의 성능을 증대시키는 커널 레벨의 SSL 가속 모듈을 제시하고, 이를 구현하였다.

전체적으로 웹 서버 시스템의 성능을 향상시키기 위하여 아래의 방법들이 사용되었다.

첫째, 사용자 영역 멀티 쓰레드 처리 구조를 커널 영역 단일 쓰레드 구조로 변경하고 매우 중요한 코드 블록을 non-preemptive하게 하여 CPU를 아주 짧은 시간동안 독점하여 요청을 처리한다. 또한, 각 요청마다 새로운 쓰레드를 구동시키는 구조 대신 요청을 받아들이는 커널 쓰레드를 하나씩으로 처리하도록 한다. CPU를 독점할 수 있는 커널 영역의 쓰레드로 구현되어 있다는 점과 짧은 시간에 동작되는 매우 중요한 처리 코드를 블럭화하여 이러한 단위 블럭들이 비선점형으로 실행되게 함으로써 처리 속도를 극대화하였다.

둘째, 기존의 SSL 처리구조에서는 파일, 메모리, 시간 관련 연산 등에 대한 시스템 콜을 사용하였지만, 커널로 구현하였을 경우에는 시스템 콜로 구현하는 것이 아니라 직접 커널 함수를 사용하므로, 시스템 콜을 사용하지 않고 커널 모드와 사용자 영역 모드와의 스위칭이 필요 없으므로 이로 인한 성능 이득을 얻을 수 있었다.

셋째, 사용자와 서버와의 주고받는 데이터의 보내기(send)/받기(receive) 연산을 직접 커널에서 수행하여 사용자 영역에서 하는 경우에 비해 호출되는 함수 콜이 감소되고, 사용자 영역과 커널 영역사이로의 버퍼 카피 연산이 줄어들게 되어, 이로 인한 이득을 얻을 수 있다.

본 논문에서는 위의 방법들과 커널의 최적화를 통하여 사용자 프로그램 형태의 웹 서버 프로그램에서 보다 약 두 배 이상의 처리 속도 향상을 얻었다.

향후 과제로는 커널쓰레드로 구현된 프로그램을 더욱더 최적화함과 동시에 성능 변화 요인에 대한 정량적 분석을 통한 증명이 필요하다.

참 고 문 헌

[1] A. O. Freier, P. Karlton and P. C. Kocher, "The SSL Protocol, V3.0," IETF draft at www.netscape.com/eng/ssl3/draft302.txt.

[2] K. Kant, R. Iyer and P. Mohapatra, "Architectural Impact of Secure Socket Layer on Internet Servers," Proc. IEEE 2000 International Conference on Computer Design, pp.7-14, 2000.

[3] A. J. Elbirt, W. Yip, B. Chetwynd and C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists," IEEE Transacions on VLSI Systems.

[4] http://www.etestinglabs.com, Ziff Davis Media Web Site.

[5] http://www.openssl.org.

[6] H. Yiming, A. Nanda and Q. Yang, "Measurement, Analysis and Performance Improvement of the Apache Web Server," Proc. IEEE International Conf. Performance, Computing and Communications, pp.261-267, 1999.

[7] http://www.modssl.org.

[8] J. Park, H. Lim and H. Kim, "Development of kernel thread web accelerator," IEE Electronic Letters, pp.672-673, Vol.38, No.13, June, 2002.

[9] 황 준, 민병조, 남의석, 김학배, 장 훠, "리눅스 커널에서 구현한 웹 서버 암호화 가속 기법에 대한 연구", 한국정보처리학회 제 18회 추계학술발표대회논문지, 제9권 제2호, pp.489-492, 2002.

남 의 석

e-mail : nahmes@kdu.ac.kr

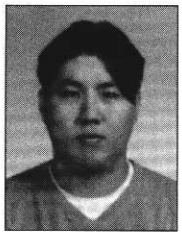
1991년 연세대학교 전기공학과(학사)

1993년 연세대학교 대학원 전기공학과 (공학석사)

1998년 연세대학교 대학원 전기공학과 (공학박사)

2003년~현재 극동대학교 정보통신학부 전임강사

관심분야 : 시스템 제어 및 응용, 지능형 모델링



민 병 조

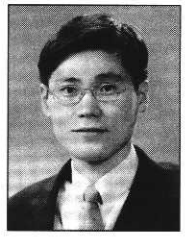
e-mail : bjmin@yonsei.ac.kr

1998년 연세대학교 전기공학과(학사)

2001년 연세대학교 대학원 전기컴퓨터 공학과(공학석사)

2001년~현재 연세대학교 대학원 전기 전자공학과 박사과정

관심분야 : 인터넷 웹 서버 기술, 임베디드 시스템, 정보보안



김 학 배

e-mail : hbkim@yonsei.ac.kr

1988년 서울대학교 전자공학과(학사)

1990년 미국 미시간대학교 전기 및 컴퓨터공학과 석사

1994년 미국 미시간대학교 전기 및 컴퓨터공학과 박사

1996년~현재 연세대학교 전지전자공학과 부교수

관심분야 : 실시간 시스템, 인터넷 웹 서버 기술, 디지털 시스템 고장포용 및 신뢰도 평가분야