서버 성능 감소를 막는 웹 서버 디렉터 설계

조양환 °이철 이창규 안철우 박규호
한국과학기술원 전자 전산학과
{yhcho, chullee, cklee, cwahn}@core.kaist.ac.kr kpark@ee.kaist.ac.kr

Design of a Web Server Director to Prevent Performance Degradation

Yang-Hwan Cho °Chul Lee Chang-Kyu Lee Chul-Woo Ahn Kyu-Ho Park
Dept. of EECS, Korea Advanced Institutue of Science and Technology

요 약

본 논문의 웹 서버 디렉터는 웹 서버의 앞단에 설치되어 과부하 상태에서 서버의 성능 감소를 막는 역할을 한다. 본 논문을 통해 구현된 웹 서버 디렉터는 기존의 아파치와 같은 웹 서버의 수정 없이 그대로 사용하기 때문에, 그 동안 개발되어 온 웹 서버 모듈과 부가 기능을 모두 사용할 수 있는 장점이 있다. 또한, 서버의 I/O 모델로 가장 이상적이라고 검증된 단일 프로세스와 이산사건처리 방식을 채택하여, 일반 웹 서버에 비하여 최고 1.6배의 성능 향상을 보인다. 부가적으로 SYN Flooding 과 같은 악의적인 웹 요청에도 견고한 특성을 보인다.

1. 서 론

웹 사용량의 폭발적인 증가로 인해 인터넷의 병목 현상과 웹 서비스의 질은 점점 더 악화되고 있다. 현재 가장 많은 웹 호스트들이 사용하고 있는[1] 아파치 웹 서버와 같은 다중 프로세스 방식의 경우 급격히 증가한 웹 요청을 정상적으로 처리하지 못하는 단점이 있다. 그림 1에서 보는 것처럼 대부분의 웹 서버의 경우 동시 접속 클라이언트의 증가에 따라 throughput 곡선이 증가하다 최고점에 이른 뒤 과부하 때는 성능이 계속적으로 감소하게 된다.

그동안 많은 연구자들이 이런 성능 감소를 막고 뛰어난 성능을 실현하기 위해서는 단일 프로세스 구조로 논블로킹 입출력과 이산사건처리 방식의 웹 서버가 가장 이상적이라고 말하고 있다.[2][3] 이런 구조의 웹 서버가 급격히 증가하는 웹 요청이나 웹 서버의 용량을 넘어서는 요청에 대해서 견고하지만, 운영체제의 논블로킹 입출력 지원이 필요하고 이산사건 처리 방식으로 프로그래밍 하는데 어려움이 있다.

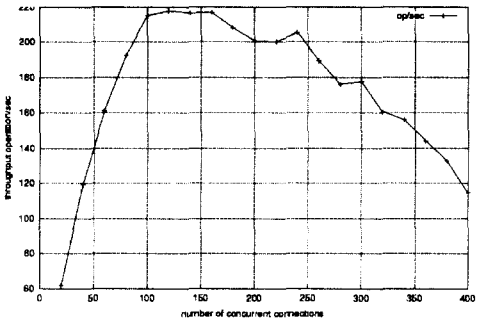

그림 1 과부하 아파치 웹 서버의 Throughput

WSD는 웹 서버와 분리된 네트워크 서버로서 웹 서버 앞단에 위치하여, 복잡한 웹 서버의 기능을 없애고 단순한 입출력 구조로서 연결을 중계하는 역할을 한다. 연결 중계과정에서의 TCP 연결을 모두 논블록킹 입출력으로 처리하며 이산사건처리 방식을 통해 단일 프로세스 방식임에도 불구하고 많은 웹 요청 처리의 동시성을 높이게 된다.

2. 웹 서버 디렉터 설계 및 구현

2.1 성능 감소를 막는 웹 서버 디렉터

웹 서버 디렉터(Web server director, 이하 WSD)는 인터넷 연결을 관리하는 측면에서는 프록시(proxy)와 같은 구조를 가지므로 디스크에 있는 파일을 읽기 위해 파일 시스템과 관련된 시스템 콜을 사용하지 않고, 소켓과 관련된 시스템 콜만 사용하게 된다. 따라서 작업과 관련된 데이터는 메모리에 적재된 상태로 실행되므로, 입출력을 수행하는데 있어 논블로킹(non-blocking) 네트워크 서버의 설계가 가능하다.

그림 2 는 WSD의 내부 함수들의 동작 과정을 설명한 것이다. 각 함수는 이산 사건 별로 나누어져 있으며 이산 사건의 상태에 따라 사용할 입출력 파일 기술자를 가지게 된다. 이 파일 기술자가 준비되어 있는지를 poll() 함수로 알아내고 처리 루프에서 준비된 파일 기술자에 연관된 이산 사건을 지정하여 처리될 함수를 호출하게 된다. 함수내의 모든 처리과정이 에러 없이 마치게 되면, 이산 사건은 다음 상태로 바뀌게 되고 앞에서와 같은 과정으로 클라이언트로부터 오는 웹 요청 연결을 뒷 단의 웹 서버로 전달하게 된다. 이산사건의 모든 정보는 그림 2 에서의 연결 객체(Connection Object)라는 내부 테이블에 저장되게 된다. 또한 toRealServer 함수는 현재 웹 서버와 미리 연결된 Keep-Alive Connection pool에서

쓰지 않는 연결을 찾아 웹 요청을 보내게 된다. 웹 서버와의 연결을 미리 해두는 방식은 두 번의 연결을 하는 프록시 구조로 구현된 WSD 의 연결 지연시간을 크게 줄여준다. WSD는 연결 객체를 4가지 상태(REQUEST, PEEK, RECEIVE, RESPONSE)로 바꿔가면서 웹 요청을 처리한다. PEEK 상태에서 뒷 단의 웹 서버가 과부하 상태로 모니터링 되었다면 그 연결 객체는 일정시간동안 처리가 연기되어 Pending Queue 에 저장되게 된다. 계속해서 웹 서버가 과부하 상태인 경우에는 클라이언트에 Busy page를 보내고 연결을 종료하게 된다.

WSD는 SYN state 의 증가량과 CPU로드와 같은 웹 서버의 성능 요소들의 정보를 수집하여 최적의 성능을 발휘할 수 있는 연결 상태를 찾아내게 된다. 웹 서버의 모니터링 데몬 프로세스(daemon Process)가 보내는 성능 요소에 대한 정보를 바탕으로 웹 서버가 서비스가 가능한지 판단하게 된다. 이 같은 과정에서 웹 서버보다 웹 요청의 양에 상관없이 일정 성능을 내는 WSD를 거쳐 웹 서비스가 할당되므로 일반 웹 서버처럼 웹 요청이 급속히 증가하는 경우 앞에서 설명한 것처럼 서비스를 하지 못하는 상태가 되는 것을 막을 수 있게 된다. 서버 용량을 최대한으로 발휘하면서 서비스 하지 못할 웹 요청을 WSD에서 필터링 해주는 역할을 하는 것이다.

2.2 웹 서버 모니터링

웹 서버가 과부하 상태인지를 확인하기 위해서는 서버가 가동되고 있는 동안의 여러 변화들을 수집하고 분석해야 한다. 리눅스 상에서는 /proc 파일 시스템으로서 커널 상의 여러 상태들을 사용자가 볼 수 있도록 기록하고 있다.

웹 서버의 상태를 모니터링 하기 위해서 사용된 리눅스의 /proc/stat 파일의 정보로는 user-level process가 사용한 cpu 시간, 시스템 콜과 하드웨어 인터럽트를 위해서 kernel 이 사용한 cpu 시간, 컨텍스트 스위칭(context switching)의 횟수 등이 있다.

위와 같은 서버의 정보를 통해서 웹 서버의 과부하 상태가 탐지되었을 때 WSD로 연결제어를 요청하는 rsInfo(웹 서버 모니터링 데몬) 이 웹 서버에 설치된다.

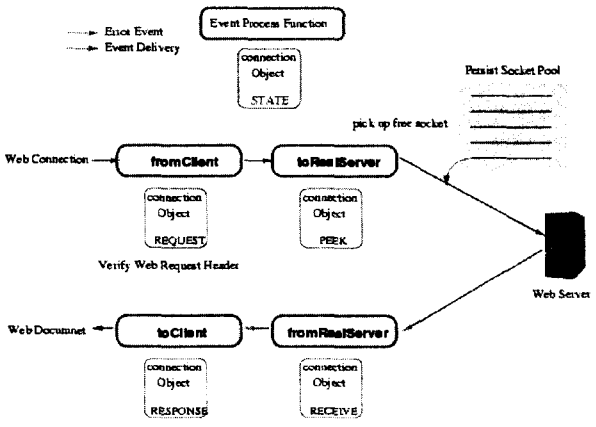

그림 2 웹 서버 디렉터 구조

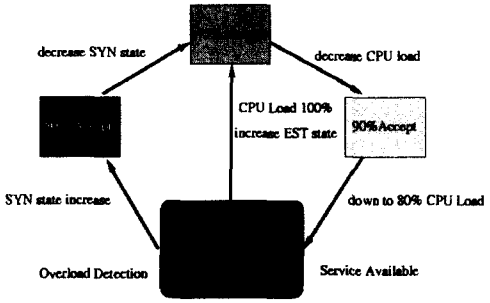

그림 3 웹 서버의 상태에 따른 연결전달 비율

일정 시간마다 웹 서버의 /proc/stat 정보를 읽어서 웹 서버가 성능 감소 상태인지를 알아내고 만일 성능 감소 상태인 경우에 WSD로 웹 서버 연결 비율을 낮추기 위한 경고 신호를 보내게 된다.

WSD의 경우 모든 기능을 논블로킹 입출력으로 수행하기 위해 기능들을 단순화하고 시스템 콜의 사용을 최소화하도록 했다. 이런 조건을 만족시키기 위해 WSD는 웹 서버로부터 전달되는 정보를 가공하는 부가적인 작업 없이 바로 연결 관리에 사용할 수 있도록 rsInfo 데몬에서 대부분의 작업을 하게 된다. 서비스가 시작되는 순간에 WSD로부터 Keep-Alive 연결을 받아서 그것을 통해 웹 서버의 정보를 바탕으로 도출된 서비스 가능 여부만을 코드화 하여 주기적으로 보고하게 된다.

3. 실험 및 결과

본 장에서는 과부하가 걸린 웹 서버의 특징과 이것으로 인해 생기는 성능 감소를 막기 위해 앞에서 제안된 WSD의 성능을 기존의 아파치 웹 서버만을 사용했을 때와 비교하였다.

3.1 실험 방법 및 구성

WSD의 성능을 측정하기 위해서 뒷단의 서버로 현재 가장 많이 사용되고 있는 아파치 웹 서버를 선택하였다. 실험의 구성은 그림 4 와 같이 워크로드 생성기로 SPECWeb 99 벤치마크 툴[4]을 클라이언트에 설치하고, 웹 서버로 리눅스를 운영체제로 사용하는 아파치 웹 서버를 사용하였으며 본 논문에서 제안한 WSD를 아파치 웹 서버의 앞단에 설치하였다. 실험은 WSD를 설치하였을 때와 아파치 웹 서버만을 사용하였을 때를 비교하였다. 실험을 위해 구성된 각 구성요소들의 사양은 표 1에 표시된 것과 같다.

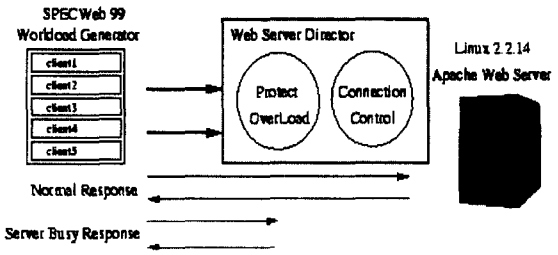

그림 4. 실험 환경

표 1 WSD, 웹서버, 클라이언트의 사양

	장비	사양
WSD & Web server	Web server	Apache 1.3.12
	Network	100Mbps Full-Duplex
	OS	Linux 2.2.14
	CPU	Intel Pentium II 450Mhz
	RAM	128M
Client	OS	Linux 2.2.14
	CPU	Intel Pentium II 450Mhz
	RAM	128M
	NIC	100M Full-Duplex
	Tool	SPECWeb99

3.2 실험 결과

그림 5에서 보는 것처럼 아파치 웹 서버와의 결과와 비교해 보면 throughput의 경우 1.6배 정도의 향상을 보인다. 또한, 동시에 연결되는 웹 요청이 증가하여도 WSD를 사용하지 않은 웹 서버에 비해 성능 감소 폭이 적은 것을 볼 수 있다.

4. 결 론

본 논문에서는 빠르게 증가하고 있는 인터넷 웹 요청을 안정적으로 처리하기 위해, 웹 서버 앞단에서 요청 연결을 관리하여 웹 서버가 최상의 성능을 유지할 수 있도록 하는 방법을 제안하였다. 이 방법을 사용하면 운영체제나 아파치와 같은 다중 프로세스 방식의 웹 서버가 가지고 있는 구조적인 문제점으로 인해 갑자기 증가한 웹 요청을 정상적으로 처리하지 못하고, throughput의 감소와 함께 latency가 증가하여 웹 서비스 질이 악화되는 현상을 막을 수 있게 된다. 또한, L7 수준에서 웹 요청의 정확성을 검사하여 웹 서버로의 잘못된 연결이나 악의적인 웹 요청으로 일어날 수 있는 웹 서버의 성능감소나 필요 없는 자원의 사용을 막는 기능을 가지게 된다.

실험 결과에서 보듯이 WSD를 웹 서버 앞단에 설치함으로서 throughput은 최고 1.6배 향상됐으며, 클라이언트의 증가로 인한 성능 감소 없이 일정하게 유지할 수 있게 되었다. 그리고, 프록시 구조의 중복된 TCP 연결로 인해 latency가 아파치 웹 서버보다 증가할 것이라고 생각할 수 있지만 실험 결과에서 보듯이 더 좋아질 뿐만 아니라 많은 웹 요청에도 거의 일정한 응답 시간을 유지

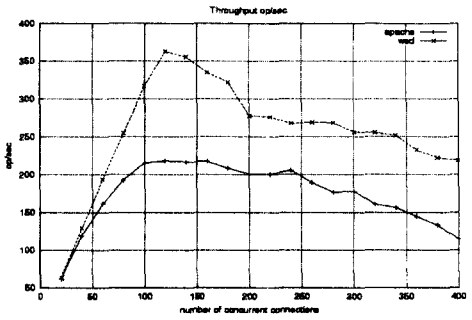

그림 5 동시접속자의 증가에 따른 throughput비교

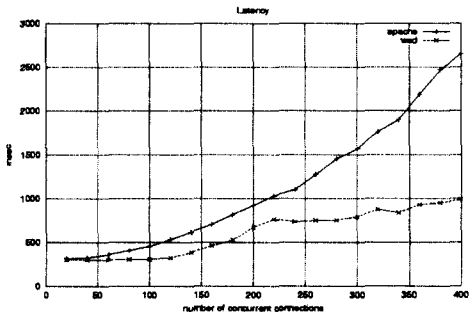

그림 6 동시접속자의 증가에 따른 Latency 비교

하는 결과를 얻었다.

이런 실험 결과들을 종합해 볼 때 WSD의 사용으로 급속한 웹 요청 증가에 유연하게 대처할 수 있으며, 중요한 기능을 하는 웹 서버를 악의적인 웹 요청에서 보호할 수 있을 것이다.

4. 참고 문헌

[1] Netcraft, Web server survey, http://www.netcraft.com/survey/

[2] V.S Pai, P. Druschel and W. Zwaenepoel, Flash: An Efficient and portable Web server, In Proceedings of the Annual USENIX Technical Conference(1999)

[3] J. C.. Hu and D. C. Schmidt, JAWS: A Framework for High-performance Web Servers, JAWS

[4] SPECWEB, http://www.spec.org/