웹 서버 QoS지원을 위한 커넥션매니저의 설계 및 구현

이종현 박창윤 정충일
중 앙 대 학 교 컴 퓨 터 공 학 과
jonghyun@orchid.cse.cau.ac.kr cypark@cau.ac.kr cijung@orchid.cse.cau.ac.kr

Design and Implementation of A Connection Manager Supporting Web QoS

Jong-Hyun Lee Chang-Yun Park Chung-Il jung
Dept. of Computer Science & Engineering, Chung-Ang Univ.

요 약

웹 서버에서의 서비스 품질을 지원하기 위하여 Web QoS를 보장하는 계층을 서버에 추가하는 방법에 대한 연구가 이루어지고 있다. 그러나 기존의 방법은 웹 서버의 자원을 소비하며, QoS를 보장하도록 웹 서버를 수정하여야 하는 단점이 있다. 본 논문에서는 기존의 웹 서버를 그대로 이용하면서 이러한 단점들을 보완하는 웹 서비스 연결 요청을 조정하는 별도의 커넥션 매니저를 설계하고 구현하여 웹 QoS에 대한 효용성을 점검하였다.

1. 서 론

현재의 인터넷을 이용한 웹 기반 서비스들에 QoS를 보장하기 위하여 네트워크에서의 QoS에 대한 연구가 이루어지고 있다. 그러나 웹 기반 서비스의 구조는 다수의 클라이언트와 하나의 웹 서버로 이루어져 있으므로 웹 서버에 부하가 집중되기 쉬우며 웹 서버에 부하가 집중되었을 경우 종단간의 네트워크 지연시간에 비하여 웹 서버에서의 지연시간이 커지기 때문에 네트워크에서의 QoS가 보장되어도 사용자들은 QoS를 보장 받을 수 없다.

이러한 이유로 웹 기반 서비스들에 웹 서버 QoS를 지원하는 계층을 웹 서버에 추가하여 QoS를 보장하는 방안에 대한 연구가 이루어지고 있다. 그러나 이러한 방법은 기존의 서버 구조를 수정해야 하며, QoS를 보장하기 위하여 웹 서버의 자원을 소비하게 되므로 웹 서버 성능의 저하를 초래하며, QoS를 보장하지 않는 웹 서버 시스템에 QoS를 보장하도록 웹 서버를 수정하는 것이므로 QoS보장에 한계가 있다는 단점이 있다. 본 논문에서는 기존 해결책들의 단점을 보완하면서 웹 서버에서의 서비스 품질을 지원하는 커넥션 매니저를 설계하고 구현하여 그 효용성을 점검해 본다.

2. 관련연구

2.1 Web QoS

웹 서비스에 QoS를 보장하기 위한 방안의 하나로 Web QoS[1]가 제안되었다. Web QoS에서 제안된 방법은 그림 1 에서 처럼 웹 서버의 HTTP계층을 QoS를 보장하도록 수정하는 방법이다. 그 결과 우선순위에 따른 차등화가 이루어지고 있으나 전체 R서비스 요청이 일정수준 이상으로 증가했을 경우 Web QoS 보장이 약화된다. 그 이유는 그림 1 에서 볼 수 있듯이 HTTP Request가 HTTP 계층으로 가기 전에 그 하위의 네트워크 계층을 거치기 때문에 웹 서버의 HTTP 계층에서의 차등화가 이루어져도 그 하위의 서버의 네트워크 계층에서 혼잡이 발생했을 경우 QoS보장이 약화되기 때문이다.

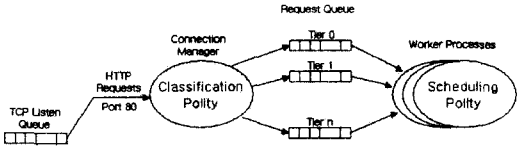

그림 1. Web QoS논문에서 제안한 방법의 구조

2.2 IP 패킷 필터링

네크워크를 지나가는 트래픽은 패킷의 형태이며 패킷은 헤더와 데이터로 구분되어 있다. 패킷 필터링이란 지나가는 패킷의 헤더를 보고 패킷의 운명을 결정짓는 작업이다.[4] 패킷의 헤더에 따라서 패킷의 통과를 허용하거나 거부하거나 패킷을 버리는 작업이 가능하다. IP 패킷 필터링을 사용하여 서버의 앞 단에서 지나가는 패킷의 통과를 허용하거나 거부하는 작업을 차등화 함으로서 Web QoS를 지원할 수 있다.

2.3 로드 밸런싱을 이용한 QoS 보장

로드 밸런싱이란 클러스터 내부의 노드들간에 작업이나 네트워크의 트래픽을 분산시키는 작업이다. 로드 밸런싱 시스템은 클라이언트에서의 리맵핑이나 패킷의 목적지 주소를 변경해 주는 방법으로 작업을 서로 다른 서버에 분산해 준다. 사용자의 우선순위에 따라서 작업을 서로 다른 프로세싱 노드에 분배 함으로서 차등화가 가능하지만

작업의 분산이 많은 부하가 걸리는 작업이라는 단점과 함께 QoS를 보장하는 작업의 분산이 힘들다는 단점이 있다.

3. Web QoS를 보장하기 위한 요구사항

3.1 HTTP Request에 대한 서버의 프로세싱

HTTP 서비스가 동작할 때 서버는 그림 2 의 과정처럼 사용자의 HTTP Request에 대하여 서버의 프로세싱 자원을 사용하여 HTTP Response를 생성하여 보내주게 된다.[5]

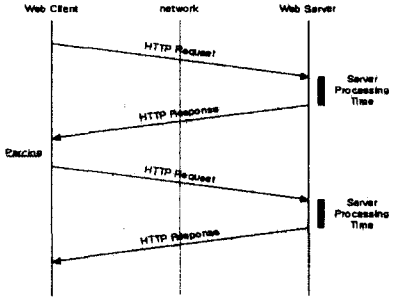

그림 2. HTTP Request에 대한 서버의 동작

웹 서버 에서의 QoS를 보장하는 것은 서버의 프로세싱 자원을 사용자의 우선순위에 맞게 사용할 수 있도록 해주는 것이며 사용자는 HTTP Request를 통하여 서버의 자원을 요구하고 서버는 사용자의 HTTP Request에 대하여 서버의 프로세싱 자원을 사용하여 HTTP Response를 생성하여 보내주게 된다. 이런 점에서 볼 때 사용자의 웹 클라이언트가 서버의 프로세싱 자원을 사용하는 방법은 웹 서버에게 HTTP Request를 전송하는 것이라고 할 수 있다.

HTTP Request의 전송은 일반적으로 TCP 프로토콜을 통하여 이루어진다. HTTP 1.0은 하나의 HTTP Request를 보내기 위하여 하나의 TCP 연결을 사용하며 HTTP 1.1은 여러 개의 HTTP Request를 전송하기 위하여 하나의 TCP 연결을 사용한다. 따라서 TCP 연결 요청에 대한 차등화로서 Web Server의 QoS를 지원할 수 있을 것이다.

3.2 웹 서버 혼잡시의 QoS보장

웹 서버가 처리할 수 있는 HTTP request보다 많은 양의 요청이 웹 클라이언트에서 웹 서버에게 보내진다면 웹 클라이언트의 요청이 TCP/IP계층까지 와서 버려지는 혼잡상황이 발생한다.[2]

TCP계층에서는 흐름제어로서 TCP 연결별로 클라이언트의 전송속도가 서버의 처리속도보다 커지는 현상을 막아준다. 그러나 TCP 연결 요청 패킷의 전송량은 네트워크의 성능에 따라 좌우된다.[2] 그러므로 웹 서버가 혼잡한 경우에는 많은 수의 TCP 연결이 생성되어 HTTP Request를 전송하므로 서버의 응답시간이 증가하며, TCP 프로토콜의 응답시간제한, 웹 클라이언트에서의 응답시간제한과 사용자의 재전송 요구 등으로 인하여 서버가 응답하는 도중에 TCP 연결이 재설정된다. 따라서 TCP 의 흐름제어는 혼잡상황을 해결할 수 없다.

이러한 혼잡상황에 대처하기 위한 방안은 서버의 부하와 상관없이 QoS를 보장할 수 있어야 하며, QoS를 보장하는 과정에서 낮은 지연시간을 가지며 서버의 자원사용을 최소화하는 것이 요구된다. 또한 기존의 운영되고 있는 웹 서버 시스템에 사용할 수 있는 적용성이나 웹 서버의 확장에 유연하게 대처할 수 있는 확장성을 위해서 서버의 수정을 최소화 하는 것이 필요하다.

기존의 서버 내부에서의 QoS 보장은 네트워크 계층에서의 혼잡으로 인한 Web QoS보장의 약화와 이를 보완하기 위한 네트워크 계층의 수정이 필요하며 Web QoS를 보장하기 위한 요구사항을 만족하는데 어려움이 있다. 그러므로 서버의 앞 단에 전용의 QoS를 지원하는 장치를 두어서 이 문제를 해결하는 것이 효과적일 것이다. QoS를 지원하는 전용의 장치는 서버의 앞 단에 설치하므로 서버의 부하와 상관없이 QoS를 지원하며, 서버의 자원을 사용하지 않으며 적용성 및 확장성이 좋다는 장점이 있다.

4. 커넥션 매니저 프레임워크

4.1 커넥션 매니저의 개념 및 구조

커넥션 매니저는 웹 서버와 웹 클라이언트의 사이에 위치하면서 웹 서버로 향하는 트래픽 중에서 TCP 연결 설정 요청 패킷의 통과나 지연여부를 조정한다. 그림 3 에 나타나 있는 것처럼 웹 클라이언트는 자신의 HTTP Request를 전송하며 이 HTTP Request는 커넥션 매니저를 거쳐서 웹 서버에 전달되게 된다.

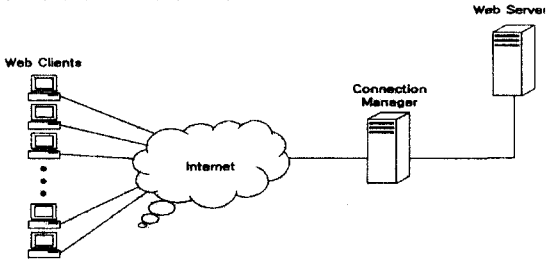

그림 3. Connection Manager의 네트워크상에서의 구조

그림 4 의 패킷의 흐름처럼 커넥션 매니저에 도착한 TCP 연결 설정 요청 패킷은 커넥션 매니저 내부의 커넥션 매니저 모듈로 전달된다. 커넥션 매니저 모듈로 전달된

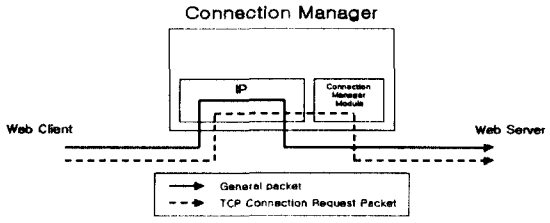

그림 4. 커넥션 매니저 내부에서의 처리

패킷들은 그림 5 의 처리과정을 거쳐서 서버로 보내진다. 커넥션 매니저 모듈은 사용자의 우선순위 정보를 유지하며 이 정보에 기초하여 TCP 연결 설정 요청 패킷의 우선순위를 정의한다. 우선순위가 정의된 TCP 연결 설정 요청 패킷은 admission control에서 현재 TCP 연결 설정 패킷의 우선순위에 해당하는 큐의 양에 따라서 받아들여지거나 거부되게 된다. Admission Control에서 받아들여진 패킷은 자신의 우선순위에 맞는 큐에 들어가게 된다. 스케줄러는 웹 서버의 네트워크 시스템이 처리할 수 있는 네트워크 트래픽의 양이나 서버의 프로세싱 능력 이상의 요청이 전달되지 못하도록 트래픽의 양을 제한하면서 패킷의 우선순위에 따른 비율로서 TCP 연결 설정 요청 패킷을 큐로부터 받아들여서 이를 웹 서버로 전송하는 역할을 하게 된다.

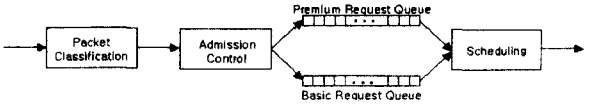

그림 5. 커넥션 매니저 내부 동작

4.2 커넥션 매니저에서 풀어야 할 문제들

커넥션 매니저는 서버의 앞 단에 위치하므로 먼저, 사용자들을 차별하기 위한 정보를 얻어오는 방법이 요구된다. CGI에서 전달해 주거나 웹 서버 내의 모듈, 또는 전용의 AGENT를 사용함 으로서 이를 해결할 수 있다. 또한, 커넥션 매니저는 웹 서버의 부하가 일정수준 이상 되었을 때 동작해야 하며 이를 위해서는 웹 서버의 부하를 커넥션 매니저에서 알 수 있는 방법이 요구된다. 웹 서버에 부하를 측정하여 보고하는 AGENT를 설치하거나 커넥션 매니저에서 직접 측정하는 방법이 사용될 수 있다. 그리고, 커넥션 매니저는 TCP 연결 설정 요청 패킷을 전송한 사용자의 우선순위에 따라 패킷의 우선순위를 인식 할 수 있어야 한다. 패킷의 발신지로서 패킷의 우선순위를 인식하거나 별도의 프로그램이나 HTTP Cookie를 사용하여 패킷을 인식할 수 있다.

5. 커넥션 매니저의 설계, 구현 및 실험

5.1 설계

커넥션 매니저에서는 서버의 부하가 일정수준 이상이 되었을 때 동작해야 하며 서버에서 전달된 TCP 연결 요청 패킷의 우선순위를 인식하여 이를 차등화 해줘야 한다. 클라이언트에서는 전송하는 패킷에 사용자의 우선순위에 대한 정보를 표시하여 전송하며, 커넥션 매니저는 이 정보를 이용하여 패킷의 우선순위를 인식한다. 웹 서버에 HTTP 요청의 전송량을 늘려가면서 처리량을 조사한 결과 초당 240개의 요청을 전송했을 경우 서버의 부하가 60%이상 발생하였다. 따라서 커넥션 매니저의 동작시점을 이 시점으로 하였다.

5.2 구현

커넥션 매니져는 리눅스 플랫폼을 사용하였으며 웹 서버 시스템과 커넥션 매니저를 연결하고 커넥션 매니저와 클라이언트 성능 측정 시스템에 연결하였다. IP Tables[4]를 사용하여 커넥션 매니저를 지나가는 패킷중에 TCP 연결 설정 요청 패킷을 커넥션 매니저 모듈에 전달하게 하였다. 커넥션 매니저 모듈은 전달된 패킷을 받아서 처리하고 이를 웹 서버로 전달한다.

5.3 실험 및 평가

클라이언트 시스템에서 httperf[3]을 사용하여 총 4개의 클라이언트에서 3개는 일반 사용자 요청을 보내고 1개는 고급 사용자의 요청을 전송하였다. 각각의 HTTP Request 전송량을 초당 20개에서부터 300개까지 동시에 증가시켜 나가면서 사용자별로 웹 서버의 처리량을 조사하였다. 실험 결과 그림 6에 나타난 것처럼 전체 요청이 증가하여도 각 사용자의 레벨에 따른 처리량은 일정하다는 것을 알 수 있다. 이는 웹 트래픽이 웹 서버에 도달하기 전에 커넥션 매니저에서 트래픽의 양을 조절 해주기 때문에 혼잡을 사전에 막아주기 때문인 것으로 생각된다.

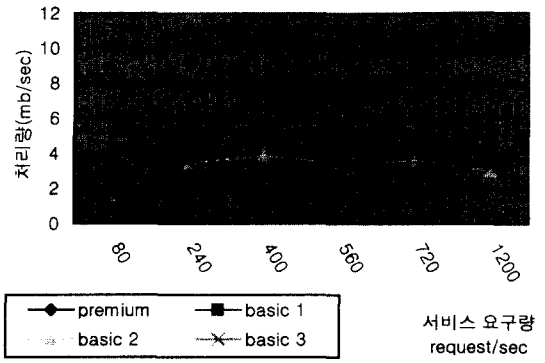

그림 6. 실험 결과

6. 결론

웹 서버의 QoS를 보장하기 위한 방안은 서버의 부하와 상관없이 QoS를 보장하면서 낮은 지연시간, 낮은 서버의 자원이용률, 적용성 및 확장성이 요구된다.

기존의 웹 서버 내에서의 해결책은 서버의 전체적인 수정이 불가피하며 서버의 자원을 사용하는 단점이 있으며 로드 밸런싱 장치 등의 외부 장치를 이용한 방법들은 외부 장치에 많은 부하가 발생하며 QoS를 보장하는 작업의 분산이 어렵다는 단점이 있다.

본 논문에서는 QoS를 지원해 줄 수 있는 커넥션 매니저를 제안하였다. 본 논문에서 제안된 커넥션 매니저는 서버의 처리량에 맞게 트레픽을 조정하여 서버에 보내주므로 서버의 혼잡을 발생시키지 않아서 QoS보장을 명확히 해주며 또한 서버의 앞 단에 위치하므로 서버의 자원을 사용하지 않는다. 또한 TCP 연결 설정 요청 패킷에 대한 처리만을 해줌으로써 커넥션 매니저에 걸리는 부하량이 적어서 확장성이 있으며 기존 서버를 수정하지 않으므로 적용성이 뛰어날 것으로 예상된다.

7. 참고문헌

[1] N. Bhatti, R. Friedrich, " Web Server Support for Tiered Services," in *IEEE Network*, Vol. 13,5 pp. 64-71, 1999

[2] P. Druschel, G. Banga, " Lazy Receiver Processing (LRP): A Network Subsytem Architecture for Server Systems," in *Proceedings of 2nd USENIX Symposium on Operating Systems Design and Implementation*, October 1997

[3] D. Mosberger, T. Jin, " httperf-A tool for measuring web server performance," in *Proceedings of the Internet Server Performance Workshop*, pp. 59-67, June 1998

[4] R. Russell, " Linux 2.4 Packet Filtering HOWTO," http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc.html, 2001

[5] D. A. Menasce, V. A. F. Almeida, *Capacity Planning for Web Performance*, Prentice Hall, pp. 71-97, 1998