LINUX에서 확장성 있는 웹 클러스터링 시스템의 설계

홍일구[0] 염미령 노삼혁

홍익대학교 정보 컴퓨터 공학부

{ighong[0], miryeom}@cs.hongik.ac.kr samhnoh@hongik.ac.kr

Extensible Web Clustering Systems in Linux

Il-gu Hong[0], Mi-ryeong Yeom, Sam H. Noh

Dept. of Computer Engineering, Dept. of Computer Science, School of Information and Computer Engineering, Hong-Ik University

요 약

웹의 이용율은 매년 100%이상씩 증가하고 있으며 대부분의 경제 활동도 웹으로 옮겨가고 있는 추세이다. 그러나 고속의 웹 이용율의 증가에 따른 웹서버의 성능 향상이 뒤따르지 못하면 웹서버는 과적이 된다. 이것은 웹을 통한 경제 활동에 매우 큰 부정적인 영향을 미치고 있다. 본 논문에서는 웹서버의 과적 상태를 대처하기 위하여 비교적 저렴한 가격의 시스템들로 구성되며 확장성이 높은 웹 클러스터링 시스템을 설계하였다. 본 논문에서 제시하는 웹 클러스터링 시스템은 사용자의 요청을 보고 부하를 분산하는 Laye-7 switch 방식을 LINUX 시스템에 적용한다.

1. 서 론

웹 클러스터링 기술이 일반화되면서 다양한 부하 분산 기법들이 소개되고 있다. 최근의 논의들은 사용자 요청의 지역성을 반영시킬 수 있는 layer-7 switch 방식이 기존의 layer-4 switch를 사용한 방식보다 우수한 성능을 보여주고 있다[1][2][3][4][5][6]. Layer-7 switch를 사용하기 위해서 사용자와 부하분산 서버 사이에 커넥션이 우선 생성된다. 이 커넥션을 통해 사용자의 요청이 부하분산 서버로 전달되며 부하분산서버는 사용자의 요청을 처리할 후위노드를 결정한다.

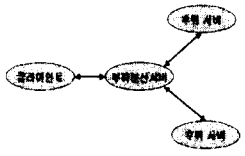

그림 2.1 TCP splicing 방식

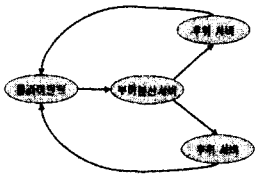

그림 1.2 TCP handoff 방식

그 다음, 부하분산서버와 사용자 사이에 설정된 커넥션을 후위노드에 전달해야 한다. 문제는 리다이렉션(redirection)을 사용하지 않고 사용자와 후위서버간에 3-way handshake를 통해 커넥션을 다시 설정 과정이 존재할 수 없다는데 있다. 이를 지원하기 위한 특별한 기법이 요구된다.

커널(kernel)계층에서 이를 지원하기 위한 방법으로 그림 1.1에서 볼 수 있는 것처럼 TCP splicing[4][5][6]방식과 그림 1.2 TCP handoff 방식이 있다[1][2][3]. TCP splicing 방식은 부하분산서버가 사용자의 요구를 후위서버에게 전달하기 위해서 후위서버와 부하분산서버 사이에 커넥션을 형성하고 이를 통해서 부하분산서버가 사용자와 후위서버 사이에 모든 메시지를 중개하고 관리하게 된다. 이 경우 부하분산서버는 메시지의 크기가 커짐에 따라 처리율이 급격히 떨어지게 되고 사용자의 요구가 증가할수록 병목현상이 나타나 확장성이 떨어진다. 이에 비해 TCP handoff 방식은 부하분산서버는 단순히 사용자의 요구를 후위서버에 전달하고 후위서버로부터 사용자에게로의 메시지 전달은 후위서버에 의해 처리됨으로서 효율적인 것으로 알려져 있다. 하지만 여전히 부하분산서버는 클러스터시스템의 상태정보와 부하분산을 관리하여야하고 handoff 프로토콜을 통해 커넥션을 후위서버에 전달하는 오버헤드(overhead)로 인해 병목현상이 나타날 수밖에 없다. 이 논문에서 확장성이 있는 TCP handoff 모듈을 LINUX 에 구현하기 위한 방법을 제시한다[3]. 제 2절에서는 확장성이 있는 TCP handoff 방식에 대한 전체적인 특징을 살펴보고, 제 3절에서는 각 모듈의 구성과 역할에 대해 살펴보며, 제 4절에서는 리눅스에 구현 시 필요사항을 언급하며 마지막 제 5절에서 결론 및 향후 추진 과제를 언급한

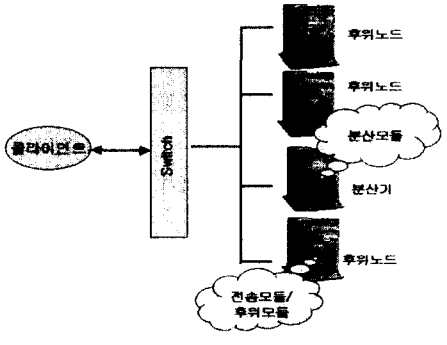

그림 2 확장성 있는 TCP handoff 구성

다.

2. 확장성 있는 TCP handoff 기법의 특징

본 논문에서 제시할 확장성 있는 TCP handoff 구성은 그림 2에서 볼 수는 것처럼 switch와 n-1개의 후위노드와 1개의 분산기로 구성된 클러스터시스템이다. 사용자의 요청은 switch를 통해서 후위노드에 전달된다. 후위노드는 전송모듈과 후위모듈로 구성된다. 전송모듈은 분산모듈을 통해서 어떤 후위노드에게 커넥션을 넘겨야 할지를 결정하고 handoff를 통해서 해당노드로 커넥션을 넘긴다. 기존의 부하분산서버가 가지고 있던 기능을 분산 모듈과 전송모듈로 분리한다. 분산모듈은 전체 시스템의 각 서버의 상태 정보와 부하분산의 책임을 갖게되며 전송모듈은 TCP handoff 프로토콜을 구현하며 사용자와 후위서버간에 패킷전송을 TCP 하위계층에서 지원하여 속도를 향상시킨다. 후위모듈은 HTTP ver1.1 persistent connection을 지원한다. 후위모듈의 역할은 handoff 과정이 끝나고 사용자와 후위서버 사이에 커넥션이 설정된 후 추가적인 사용자 요청이 들어올 경우 분산기를 통해 생성된 변형된 요구(tagged request)를 Web 서버에 전달 서비스를 제공하며 persistent 커넥션을 지원한다.

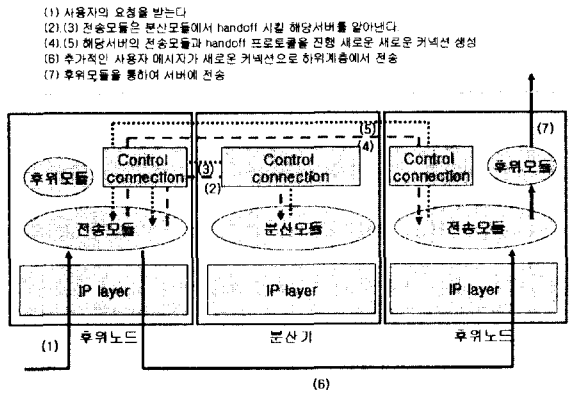

그림 3 TCP handoff 과정

3. TCP handoff 프로토콜

3.1 제어 커넥션(control connection)

TCP handoff protocol을 지원하기 위해서 기본적으로 클러스터시스템을 구성하는 각 서버 혹은 모듈간에 제어정보를 주고받을 수 있는 커넥션이 필요하다. 즉 n 개로 구성된 클러스터 시스템을 가지고 있다면 n^n 개의 제어 커넥션(control connection)이 형성되어야 한다. 제어 커넥션은 TCP handoff 프로토콜을 통한 커넥션 전달과정과 관계된 모든 메시지가 교환되는 통로 역할을 한다.

3.2 TCP handoff 프로토콜 프로세스

3.2.1 커넥션의 전달

그림 3은 TCP handoff 과정을 설명하고 있다. 전송모듈은 새로이 설정된 커넥션을 통해서 들어오는 사용자의 요청을 제어 커넥션을 통해서 분산기로 전송한다. 분산 모듈은 사용자의 요구를 처리할 해당 후위 서버를 결정하고 전송 모듈에 이 결과를 통보한다. 분산 모듈을 통해서 결정된 후위서버가 현재 사용자의 요구를 받은 노드와 일치하지 않을 경우 전송 모듈은 제어 커넥션을 통해 해당 서버에 handoff 요청을 전달한다. 이 요청 메시지는 새로운 커넥션을 생성하기 위한 몇 가지 정보를 포함해야 한다.

첫째, 기존의 커넥션에 대한 ID가 전송된다. 각각의 커넥션은 클러스터 시스템에서 자신을 구별할 수 있는 ID가 필요하게 된다. ID는 socketID에 대한 정보와 각 서버에서 부가하는 커넥션ID로 구성된다. socket ID는 사용자와 해당 서버 사이에 생성된 소켓의 정보, 즉 사용자의 IP주소와 port번호 그리고 서버의 IP주소, port번호로 구성된다.

둘째, 기존에 설정된 커넥션을 통해 전달된 요청(request)이 전달된다. 사용자의 요청은 분산 모듈을 통해 이미 변형된 요구(tagged request)이다. 이에 대해서는 뒤에서 다루기로 한다.

셋째, 기존 커넥션에 의해 설정된 TCP 계층 정보가 전달된다. TCP 계층정보는 새로운 커넥션을 위한 소켓을 생성하는데 사용되며 소켓의 sequence number와 커넥션 제어를 위해 사용되는 window size, timer, send/receive buff의 크기 등으로 구성된다.

3.2.3 Handoff된 커넥션으로 메시지 전송

전송모듈은 각 패킷들이 전송될 때 소켓을 통해 현재 커넥션의 상태를 확인하게 된다. 이때 아래의 몇 가지 상태가 존재할 수 있다. 커넥션이 생성될 때, 커넥션이 handoff 프로토콜이 진행중일 때, 이 커넥션이 후위 모듈에 의해 관리될 때, 마지막으로 handoff 프로토콜이 완료되었을 때이다. 커넥션이 생성될 때는 위에서 언급한 handoff 프로토콜 과정이 시작되며, handoff 프로토콜이 진행중일 때는 저장공간에 임시 보관된다. 후위모듈에 의해 관리될 때는 후위 모듈을 호출하여 패킷에 대한 처리를 요청하고 handoff 프로토콜이 완료된 상태면 상위계층을 거치지 않고 해당 후위서버로 전송하여 노드의 부하를 최소한으로 줄이게 된다.

3.2.4 handoff된 커넥션의 종료

전송모듈은 handoff된 커넥션이 종료되었는지를 알 수 없다. 단지 기존의 커넥션으로 SYN 플래그가 설정되 패킷이 전송되면 기존의 커넥션이 종료되고 새로운 커넥션이 요구되고 있다고 생각하게 된다. 이 경우 TCP handoff 프로토콜을 통해 사용자와 연결된 후위서버에 RST 플디를 설정한 패킷을 전송하여 handoff된 커넥션을 강제 종료하게 하고 분산기에게도 제어 커넥션을 통하여 기존에 생성된 커넥션이 종료하였다는 사실을 알려주게 된다. 분산 모듈은 커넥션에 대한 정보를 제거하고 이 커넥션을 통해 할당되었던 부하정보를 갱신하게 된다.

3.3 분산 모듈(dispatcher module)

분산 모듈의 역할은 사용자와 후위 서버 사이에 설정된 커넥션에 대한 관리와 부하를 적절히 분배한다. 이를 위해서 분산 모듈은 각 후위 서버의 할당된 커넥션 정보와 현재 부하 상태에 대한 정보를 가지고 있어야 한다.

3.3.1 후위서버 부하의 측정

후위서버의 부하의 측정 여러 가지를 고려해야 한다. 서버의 CPU 사용시간이나 디스크 입출력 연산의 수행 숫자들을 관찰하는 것이 필요하게 된다. 여기서는 LARD 수행한 방식을 따른다[1][2][3].

3.3.2 후위서버의 결정

전송모듈 혹은 후위모듈로 새로운 요청이 도착하게 되면 분산기로 이 요청이 전달된다. 분산기는 현재 각 후위서버의

부하정도와 할당된 사용자 요청과 서버의 사상관계를 이용하여 새로운 사용자 요청을 후위 서버에 할당하고 요청은 적절한 부하분산을 위해서 변형된 요청(tagged request) 형태로 해당 모듈에 전달한다..

3.3.3 persistent 커넥션의 지원

HTTP 1.1 persistent connection을 지원하기 위하여 두 가지 해결 방법이 있을 수 있다. 첫째는 후위모듈에 handoff 프로토콜을 수행할 수 있는 기능을 추가해서 매번 사용자의 요청이 있을 때마다 handoff 프로토콜을 이용하여 커넥션을 새로운 서버로 넘겨주는 방식이다. 두 번째 방식은 변형된 요청(tagged request)을 이용하여 파일을 해당 후위서버에서 읽어오는 것이다. 예를 들면 현재 후위노드 B(192.169.75.3)가 b 파일을 메모리 캐쉬하고 있다고 가정해 보자. 사용자가 후위노드 A(192.168.75.1)에 b파일을 요청했을 때 분산 모듈은 후위노드 A가 b 파일을 처리하는 것이 B에서 네트워크를 통해 처리하는 것보다 오버헤드(overhead)가 더 있다고 판단한다면 디스크 입출력 연산을 통해 후위노드 A가 서비스를 제공하기보다 /192.168.75.3/b 형태로 변형된 요청(tagged request)을 생성하여 A에 전달하고 후위노드 A의 Web 서버가 네트워크 통해서 파일을 얻고 사용자에 제공하기를 바라게 될 것이다. 전자의 경우 매 사용자 요구마다 handoff 프로토콜을 적용하여야 하기 때문에 관리가 어렵고 또한 비용이 많이 들게 되므로 후자의 경우가 선호된다.

3.4 전송모듈(forwarding module)

전송 모듈의 역할은 크게 두 가지로 생각할 수 있다. 첫째 handoff 프로토콜을 처리를 담당한다. 둘째 hanodoff 된 커넥션에 대해서 하위계층(low-layer)에서의 전달을 지원한다. 위의 기능을 지원하기 위해서 전송모듈은 첫째 TCP 3-way-handshake 과정에 개입 할 수 있어야 한다. 커넥션이 생성되면 사용자의 요구가 도착할 때마다 전송모듈이 개입할 수 있도록 핸들러를 설정해야 한다. 둘째 커넥션으로 전달되는 메시지를 가로채서 전송 모듈에 전달 할 수 있어야 한다. 이를 지원하기 위해서 웹 서버의 listen 소켓에 전송모듈이 설정되어야 한다.

3.5 후위모듈(back-end module)

전송모듈사이에 handoff 프로토콜이 완료되면 새로운 커넥션이 생성되고 이 커넥션을 통해 전달되는 사용자의 요청은 후위모듈에 의해 관리된다. 후위모듈은 사용자의 요구를 Web 서버에 전달하는 단순한 작업을 진행하지만 persistent connection을 효과적으로 지원하기 위해 사용자의 요청을 변형된 요청(tagged request)로 전달하기 위해서 분산기와 상호 동작하여야 한다.

4. LINUX network code에서의 구현

4.1 분산기와 전송모듈의 형성

클러스터시스템의 초기화 과정에서 각 서버사이에 제어커넥션이 형성된다. 분산기 시스템 콜을 통해서 이 제어 커넥션의 설정된 소켓에 핸들러를 추가한다. 다음 전송모듈은 잘 알려진 Web 서버 포트로 소켓을 생성하고 사용자의 요청을 가로챈다. 그러기 위해서 Web 서버의 포트를 변경해야하고 소켓에 분산기와 마찬가지로 시스템 콜을 이용 핸들러를 설정한다. 전송모듈은 3-way handshake를 통해 이 소켓으로 커넥션이 생성되어 새로운 소켓이 생성되면 위에서 설정된 핸들러는 새로 생성된 소켓을 처리하기 위한 핸들러를 설정하여 준다. 3 - way handshake 과정을 통해 ESTABLISHED 상태가 되면 전송모듈 핸들러는 새로 생성된 소켓에 사용자의 요청을 처리하기 위한 핸들러를 설정한다. 설정된 커넥션을 통해 전달되는 사용자 메시지가 있을 때마다 소켓에 설정된 핸들러가 호출된다. 그러기 위해서 커넥션으로 전송되는 사용자 패킷을 처리하는 tcp_rcv_established와 tcp_rcv_state_process 함수에 약간의 코드가 변형되어야 한다. 또한 소켓의 receive queue에 sk_buff 형태로 저장된 메시지를 임으로 삽입/삭제 할 수 있는 기능이 추가되어야 한다. 여기서 하나 생각해 볼 수 있는 것은 전송모듈을 통해 전달된 변형된 요구(tagged request)로 인해 sequence number가 증가하게 된다는 점에 있다. 이 것을 처리하기 위하여 TCP layer의 tcp_recvmsg 함수를 적절히 변형시켜야 한다.

또한 handoff 과정에 있는 커넥션으로 추가적인 메시지가 전달 될 수 있다는 점을 생각해야 한다. 이 것을 처리하기 위해서 서 handoff된 커넥션으로 전송되는 메시지는 상위 계층으로 전송되지 않고 네트워크를 통해서 해당 후위 서버로 직접 전송하는 코드가 삽입되어야 하며 또한 handoff 프로토콜이 진행 중인 커넥션으로 전달되는 메시지는 임시로 보관되어야 한다.

5. 결론 및 향후 추진 과제

현재 LINUX에서는 LVS(Linux Virtual Server)프로젝트를 통해서 Layer-4 switch 기법을 이용한 클러스터링시스템을 제공하고 있다. 아직까지 Linux에서 Layer-7 방식에 기반을 둔 클러스터시스템 구성은 실현되지 않고 있다. 이 논문에서 구현중인 클러스터링기법은 기존의 LVS 방식의 시스템보다 우수한 성능을 보여줄 것으로 기대된다.

사용자 요청을 이용하여 부하분산을 하는 방식은 메모리에 캐쉬된 파일을 이용함으로 서버의 부하를 줄이고 처리량을 높일 수 있다.[1][2][3][4][5][6] 사용자 요청을 이용하기 위해서 기존에 설정된 커넥션을 사용자와 후위 서버에 설정된 새로운 커넥션으로 넘겨줄 수 있는 방법이 필요하며 TCP handoff 기법은 이를 효과적으로 수행할 수 있다.

참 고 논 문

[1] Vivek S. Pai, Mohit Aron, Gauray Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel and Erich Nahum,"Locality-aware Request Distribution in Cluster-based Network Servers". In Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII), 1998.

[2] Mohit Aron, Peter Druschel, and Willy Zwaenepoel, "Efficient Support for P-HTTP in Cluster-Based Web Servers". In Proceedings of the USENIX 1999 Annual Technical Conference, 1999.

[3] Mohit Aron, Darren Sanders, Peter Druschel and Willy Zwaenepoel,"Scalable Content-aware Request Distribution in Cluster-based Network Servers". In Proceedings of the USENIX 2000 Annual Technical Conference , 2000.

[4] C. S. Yang, M. Y. Luo,"Efficient Support for Content-based Routing in Web Server Cluster". In Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems. 1999

[5] Zhang X., Barrientos M, Chen B., Seltzer M. "HACC: An Architecture for Cluster-Based Web Server". In Proceedings of the 3rd USENIX Windows NT Symposium,1999

[6] Ariel Cohen, Sampath Rangarajan, and Hamiltom Slye, "On the Performance of TCP splicing for URL-aware Redirection". In Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems 1999.