다양한 형태의 웹 서버 클러스터 비교에 관한 연구

문종배*, 김명호*
*숭실대학교 컴퓨터학과
e-mail: comdoct@ss.ssu.ac.kr kmh@comp.ssu.ac.kr

A Study on Comparison between various forms of Web Server Clusters

Jong-Bae Moon*, Myung-Ho Kim*
*Dept of Computer Science, SoongSil University

요 약

최근 웹 사이트에 클러스터링 기술이 많이 적용되고 있다. 그 중 리눅스 가상 서버를 이용한 형태로 많이 구축되고 있는데 이 형태는 전면 서버가 한 대이므로 사용자 요청이 많이 증가하면 병목현상으로 더 이상 서비스를 할 수 없는 상황이 발생할 수 있다. 그래서 본 논문에서는 다수의 전면서버를 갖는 형태의 클러스터링 기술을 구현하고 기존의 형태에 비해 성능의 우수함을 보이기 위하여 여러 형태의 클러스터를 구성하여 비교 실험한다.

1. 서론

최근 인터넷 사용이 일반화되면서 인터넷 사용자의 수가 매년 증가하고, 인터넷은 새로운 산업의 형태로 발전하고 있다. 전자 상거래는 그 좋은 예가 될 수 있다. 현재 전자 상거래와 같은 서비스를 제공하는 사이트가 점차 늘어나고, 이 가운데 하루 평균 백만 건 이상의 접속이 이루어지고 있는 사이트도 생겨나고 있다. 이러한 사이트에서는 사용자의 요청을 처리하기 위해 많은 컴퓨팅 파워를 요구하고 있다[6]. 또한 이러한 사이트는 시스템이 다운되었을 경우 엄청난 경제적 손실을 입게 된다. 따라서 시스템의 다운 타임 최소화도 요구하고 있다. 이러한 측면에서 고성능과 고가용성을 제공하는 웹 서비스의 필요성이 대두되었다.

단일 시스템으로 구성된 기존의 사이트들은 단일 시스템의 한계 때문에 고성능과 고가용성의 서비스를 제공하는 것이 이미 한계에 도달하였다[2,4]. 이러한 단일 시스템의 한계를 극복하고 더욱 강력한 컴퓨팅 파워와 시스템의 안정적 서비스를 제공하기 위해 클러스터링 기술이 웹 사이트에 적용되고 있다[5].

웹 서버 클러스터는 사용자의 서비스 요청을 여러 대의 클러스터에 분배시켜주는 기능을 가지는 클러스터 시스템이다. 웹 서버 클러스터는 여러 형태로 나눌 수 있다. 본 연구에서는 기존 구성되었던 여러 형태의 웹 서버 클러스터와 본 연구에서 설계 및 구현한 웹 서버 클러스터 형태의 장단점을 비교분석한다.

2. 다양항 형태의 웹 서버 클러스터

웹 서버 클러스터는 사용자의 요청을 어떻게 분산시키는지에 따라 다음의 4가지 분류로 나눌 수 있다[4]. 본 장에서는 웹 서버 클러스터의 4가지 분류와 최근 많이 사용되고 있는 리눅스 가상 서버를 이용한 구성 방법에 대하여 살펴보고, 각각의 형태의 특징을 살펴본다.

2.1 클라이언트 기반 웹 서버 클러스터

웹 클라이언트가 자바 애플릿(Java Applet)을 통하여 각 서버들의 상태 정보를 수집하고 이 정보를 기반으로 하여 서버를 선택하고 그 서버로 웹 클라이언트의 요청을 전달하는 방법이다[4].

2.2 DNS 기반 웹 서버 클러스터

클라이언트 기반 웹 서버 클러스터의 단점을 없앤 분산 웹 서버 구조로 DNS(Domain Name Service)에서 사용자에게 서버들의 단일 가상 인터페이스를 제공하여 부하를 분산하는 구조이다.

이 형태의 대표적인 방식으로는 라운드 로빈 DNS 방식이 있다. NCSA에서 처음 사용한 이 방식은 DNS에 분산된 서버들의 IP 주소를 라운드 로빈 방식을 적용하여 순차적으로 대응시켜 서로 다른 서버를 선택하는 방법이다[7,8].

2.3 서버 기반 웹 서버 클러스터

서버 기반의 웹 서버 클러스터는 일반적으로 다음과 같은 2단계 분산 메커니즘을 가지고 있다. 우선 사용자의 요청은 웹 서버 시스템의 DNS에서 클러스터의 한 서버에 할당되고, 그 다음 사용자의 요청을 받은 서버는 다시 다른 서버에게 사용자의 요청을 다시 요청하게 된다. 이 방법을 사용하는 예로는 Scalable Server World Wide Web(SWEB)이 있다[9].

2.4 디스패쳐 기반 웹 서버 클러스터

사용자의 요청을 후면에 있는 웹 서버에 보내주는 디스패쳐를 통해 스케줄링하는 방식이다. 이 방법은 사용자에게는 하나의 가상 IP만을 보여줌으로써 웹 서버 클러스터 시스템에 대한 투명성을 제공한다. 이 방법은 다시 디스패쳐가 사용자의 요청을 처리하는 방법에 따라 Packet Rewriting, Packet Forwarding 등의 방식으로 나누어진다.

Packet Rewriting 방식의 예로는 Magicrouter와 Cisco의 LocalDirector가 있다[10]. Packet Forwarding 방식에는 IBM의 Network Dispatcher[11]와 ONE-IP가 대표적인 예이다[12].

2.5 리눅스 가상 서버를 이용한 웹 서버 클러스터

리눅스 가상 서버를 이용한 웹 서버 클러스터는 디스패쳐 기반 웹 서버 클러스터의 일종으로 최근 웹 서버 클러스터의 고성능과 투명성, 확장성, 안정성을 확보하기 위하여 많이 사용되고 있다.

이 방법에서 클러스터를 구성하는 서버들은 사용자의 요청을 받아들여 분배하는 전면 서버 부분과 사용자 요청을 실질적으로 처리하는 후면 서버 부분으로 구분되는 것이 일반적이다. 또한 디스패쳐 방식의 단점인 패킷의 변환에 따른 오버헤드를 해결하기 위하여 후면 서버가 처리한 응답을 전면 서버를 거치지 않고 사용자에게 직접 보낼 수 있도록 되어있다.

2.6 웹 서버 클러스터의 비교

앞에서 살펴본 다양한 형태의 웹 서버 클러스터들의 특징을 비교해 보면 다음과 같다.

클라이언트 기반 웹 서버 클러스터 형태는 사용자의 요청을 받아서 처리하는 부분이 따로 있지 않고 각 서버들이 직접 사용자 요청을 받아서 처리하기 때문에 사용자에게는 클러스터 시스템이 하나의 서버처럼 보이지 않는다. 또한 각 서버 노드들의 상태를 관찰하기 위한 네트워크의 부하량이 많은 단점이 있다.

DNS 기반 웹 서버 클러스터 형태는 단순하게 구성할 수 있고 확장성이 뛰어나다는 장점이 있지만 서버들의 상태를 알 수 없기 때문에 장애가 발생한 노드에 사용자의 요청을 계속 보낼 수 있다는 단점이 있다[4]. 또한 클라이언트의 캐싱에 의해 특정 서버에 사용자의 요청이 많이 할당되어 서버들의 불균형을 초래할 수 있다.

서버 기반 웹 서버 클러스터 형태는 각 서버들이 부하 분산을 하기 때문에 한 노드가 장애로 서비스를 하지 못할 경우에도 전체 클러스터는 계속 서비스를 할 수 있다. 즉 SPOF(Single Point Of Failure)가 발생하지 않는 장점이 있다. 그러나 HTTP- Redirection과 같은 응용프로그램 수준의 스케줄링 방법 사용함으로써 응답시간이 길어지게 되는 단점이 있다.

디스패쳐 기반 웹 서버 클러스터는 하나의 가상 IP(VIP)를 사용함으로써 사용자에게는 클러스터의 투명성을 쉽게 제공하고, DNS 기반 웹 서버 클러스터에서 발생했던 클라이언트 캐싱이 발생하지 않는 장점이 있다. 그러나 중앙집중식의 디스패쳐는 사용자의 요청이 많아지면 병목현상이 발생하여 SPOF가 발생할 수 있다. 또한 패킷의 변환에 오버헤드가 발생한다는 단점이 있다.

리눅스 가상 서버를 이용한 웹 서버 클러스터는 디스패쳐 기반 웹 서버 클러스터의 일종으로 하나의 가상 IP를 사용함으로써 손쉬운 투명성을 제공하고, IP 수준의 스케줄링과 후면 서버가 직접 사용자에게 응답을 보냄으로써 빠른 응답시간을 보장할 수 있고, 후면 서버의 확장성을 손쉽게 보장하는 장점이 있다. 그러나 중앙집중식의 부하 분배방법으로 하나의 전면 서버만을 사용하기 때문에 병목현상과 SPOF을 피할 수 없다. 또한 후면 서버들의 부하량과 상관없는 스케줄링 알고리즘을 사용하기 때문에 후면 서버들의 부하의 불균형을 초래할 수 있다.

3. 새로운 형태의 웹 서버 클러스터 시스템

본 연구에서는 여러 형태의 웹 서버 클러스터 중 리눅스 가상 서버 방식의 단점을 보완하여 다수의 전면 서버를 갖는 새로운 형태의 웹 서버 클러스터 시스템을 설계하고 구현하였다.

3.1 고가용성 웹 서버 클러스터 시스템의 전체 구조

본 논문에서 구현한 고가용성 웹 서버 클러스터 시스템은 그림 1에서 보는 것처럼 전면 서버와 후면 서버의 역할을 동시에 하는 다수의 노드들로 구성되어 있다. 각 노드들은 하트비트 데몬, 로드 모니터, 부하량에 따른 부하분배 방식의 스케줄링 모듈 등의 컴포넌트들로 구성되어 있다.

사용자의 요청은 일차적으로 라운드 로빈 DNS에 의해서 클러스터의 임의의 서버에 전달된다. 사용자 요청을 받은 서버는 자기 노드의 부하량을 측정하여 임계값보다 작으면 자기 노드에서 사용자 요청을 처리하여 응답을 직접 사용자에게 보내도록 한다. 그 값이 임계값보다 크면 로드 모니터와 스케줄링 모듈에 의해서 부하량이 가장 작은 노드를 선택하여 IP 터널을 통해 사용자 요청을 재전송한다.

(1) 하트비트(Heartbeat) 데몬

하트비트 데몬은 웹 서버 클러스터 시스템의 고가용성을 위한 컴포넌트이다. 하트비트 데몬은 클러스터 시스템의 안정성과 고가용성을 위하여 하트비트 체인을 구성하고 유지한다. 클러스터 시스템에서 임의의 노드가 다운된 경우 하트비트 체인에 의해서 결함 노드를 발견하게 되고, 결함 노드의 작업은 fake 기능을 사용하여 나머지 노드 중 임의의 노드에 의해서 수행

되도록 한다.

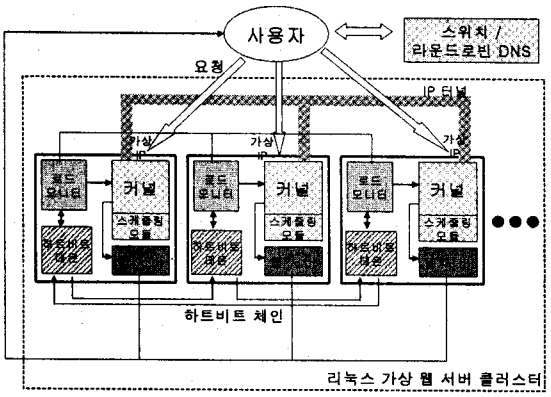

그림 1 전면 서버와 후면 서버를 통합한 고가용성 웹 서버 클러스터의 전체 구조

(2) 로드 모니터(Load Monitor)

로드 모니터는 주기적으로 자기 자신의 부하 정보를 측정하고 이 값을 커널에 보내어 유지하고, 부하의 변화가 있으면 다른 노드와 자신 노드에게 부하 정보를 알린다. 부하의 변화가 크지 않을 경우에도 부하 정보를 교환하는 것은 오히려 로드 모니터간의 네트워크 오버헤드를 늘릴 수 있다. 따라서 로드 모니터는 임계값을 설정해서 부하의 변화량이 임계값을 초과하는 경우에만 자신의 부하 정보를 다른 노드들에 보내도록 하도록 한다.

(3) 서버들의 부하를 고려한 스케줄링 모듈

부하량을 기반으로 하는 스케줄링을 하기 위해 각 노드의 부하 정보가 필요하다. 스케줄링 모듈은 부하 정보를 바탕으로 최적의 노드를 선택한다. 커널은 선택된 노드의 IP를 캡슐화(Encapsulation)하여 후면 서버로 패킷을 IP 터널을 통하여 전송한다.

4. 실험 및 분석

본 장에서는 다양한 웹 서버 클러스터를 구성하여 성능을 측정, 비교 분석하여 본 논문에서 구현한 웹 서버 클러스터 형태의 효율성을 보인다.

웹 서버 클러스터의 성능을 평가하기 위해서 SURGE라는 공개소스 프로그램을 사용하였다[13]. SURGE는 실제 사용자의 행동을 대신하는 쓰레드를 발생해서 웹 서버에 접속하여 주어진 테스트 시간동안 사용자 요청을 발생하는 프로그램이다.

실험에 사용한 클러스터는 7대의 P-III 800MHz, 256MB RAM, 100Mbps의 이더넷 허브로 구성되어 있고, 운영체제는 리눅스(커널 버전 2.4.13)를 사용하였다. 웹 서버는 아파치 웹 서버를 사용하였다.

4.1 사용자의 증가에 따른 성능 평가

각각의 웹 서버 클러스터의 최대 성능을 측정하기 위하여 우선 서버의 수를 고정시키고 사용자의 요청을 발생시키는 클라이언트의 수를 150개까지 증가시키면서 웹 서버 클러스터가 초당 처리하는 처리량을 측정하였다.

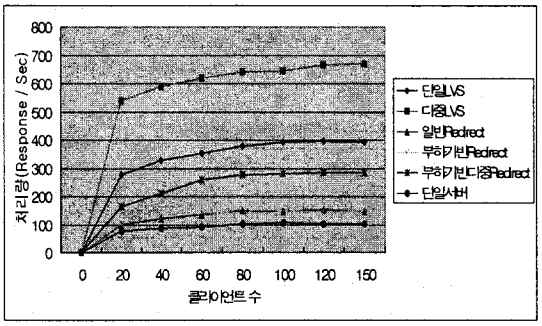

그림 2 클라이언트 수의 증가에 따른 여러 형태의 웹 서버 클러스터의 성능 평가

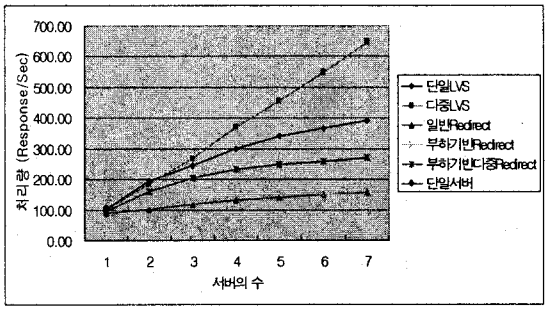

그림 3 서버의 증가에 따른 웹 서버 클러스터의 성능평가

그림 2는 전면 서버의 수가 하나인 형태와 다수인 형태 모두 클라이언트 수가 증가하면서 처리량이 증가하다가 어느 정도 되었을 때 포화되는 현상을 보여주고 있다.

리눅스 가상 서버를 이용한 웹 서버 클러스터는 전면 서버가 사용자 요청 패킷만 보고 스케줄링 해주기 때문에 빠른 응답시간과 많은 처리를 할 수 있다. 따라서 응용프로그램 수준의 스케줄링을 하는 HTTP 재지향 형태의 클러스터 형태보다 훨씬 많은 처리량을 보여주고 있다. 본 논문에서 구현한 다수의 전면 서버를 갖는 리눅스 가상 서버 형태의 웹 서버 클러스터는 전면 서버가 한 대인 형태보다 1.5배 이상의 성능 향상을 보여주고 있다. 이것은 사용자 요청을 분산하는 전면 서버를 다수로 두었기 때문이다.

이 실험으로 각 웹 서버 클러스터의 최대 성능을 가져오는 클라이언트 수를 알 수 있다. 또 전면에서 사용자 요청을 분산하는 서버가 한 대일 때보다는 다수일 때가 더 많은 사용자 요청을 처리해주는 것을 알 수 있다.

4.2 서버의 증가에 따른 성능 평가

리눅스 가상 서버를 이용한 형태의 웹 서버 클러스터들은 클라이언트 수가 120개 정도 되었을 때 포화가 되었고, 나머지 형태의 웹 서버 클러스터들은 100개

정도 되었을 때 포화가 되었다. 두 번째 실험에서는 이 클라이언트 수를 고정시키고 사용자 요청을 처리하는 서버를 7대까지 증가시키면서 성능을 측정하였다. 단일 LVS, 일반 HTTP 재지향, 부하기반 HTTP 재지향 방식은 사용자의 요청을 분산하는 전면 서버가 1대로 고정되어 있고 후면에서 실제로 웹 서비스를 하는 서버들의 수를 증가시키면서 실험하였다. 그리고 다중 LVS와 부하기반 다중 HTTP 재지향의 경우에는 전면 서버와 후면 서버의 역할을 동시에 하는 서버들을 증가시키면서 실험하였다. 실험 결과는 그림 3에서 나타내었다.

일반 HTTP 재지향 방법과 부하기반 HTTP 재지향 방법은 후면에 서비스를 처리하는 서버들이 증가하면서 성능향상이 조금 있지만 서버의 수가 증가해도 지속적인 성능향상은 보이지 않고 있다. 이것은 전면에 사용자 요청을 재지향시키는 서버가 과부하가 발생하여 더 이상의 성능은 보이지 않는 것이다. 부하기반의 다중 HTTP 재지향 방법은 사용자의 요청을 받는 전면 서버들을 증가시키면서 성능의 향상을 보이고 있다. 그러나 재지향 서버가 한 대인 형태보다 성능향상이 있지만 HTTP 재지향에 따른 네트워크의 오버헤드와 시간 지연 때문에 많은 성능향상을 보이지 않았고 지속적인 성능의 향상은 보이지 않았다.

리눅스 가상 서버를 이용하여 구성한 웹 서버 클러스터는 후면 서버의 수가 증가하면서 HTTP 재지향을 이용한 방법보다 많은 성능의 향상을 보이고 있다. 리눅스 가상 서버는 전면 서버가 IP 수준의 스케줄링을 하기 때문에 다른 형태의 웹 서버 클러스터보다 성능이 좋다. 그러나 서버의 수가 증가하면서 지속적인 증가를 보이지 않고 있다. 이것은 전면 서버가 한 대이므로 병목현상이 발생하여 지속적인 성능의 향상은 보이지 않는다.

이번 실험에서는 사용자 요청을 분산하는 서버를 다수로 설정하여 많은 성능 향상을 가져올 수 있다는 것을 볼 수 있다. 따라서 본 논문에서 구현한 웹 서버 클러스터의 형태가 가장 성능이 우수한 것을 보았다.

5. 결론 및 향후 과제

본 논문에서는 다양한 형태의 웹 서버 클러스터들과 본 논문에서 구현한 전면 서버와 후면 서버의 구분을 없앤 형태의 성능을 측정하고 비교 분석하였다. 성능 측정 결과 클러스터의 여러 형태에서 다수의 전면 서버를 두는 것이 하나의 전면 서버를 두는 것보다 성능이 좋다는 것을 볼 수 있었다. 또한 본 논문에서 구현한 리눅스 가상 서버를 기반으로 한 다수의 전면 서버를 갖는 형태의 웹 서버 클러스터가 가장 성능이 우수하다는 것을 보였다.

향후 과제로는 웹 서버 클러스터는 고가용성을 위한 하트비트 기능에서 장애 노드의 판단과 fake 할 때의 시간적 지연 동안에 사용자 요청 패킷들이 손실될 수 있다. 그래서 향후에는 장애 노드와 서비스 가능 서버들에 대한 판단 지연과 fake에 따른 사용자 요청 패킷들의 손실을 줄이는 연구가 좀 더 필요하다.

6. 참고문헌

[1] Raykumar Buyya, "High Performance Cluster Computing vol1." *Chap 36. A Scalable and Highly Available Clustered Web Server*, Prentice Hall, 1999.

[2] Huican Zhu, Ben Smith, Tao Yang, "A Scheduling Framework for Web Server Clusters with Intensive Dynamic Content Processing," *Processing of the 1999 ACM SIGMETRICS Conference*, May 1999.

[3] RADWARE Web Server Director Pro, URL: "http://www.radware.com/support/paper/What to Look For in an IP Load Balancer".

[4] Valera Cardellini, Michele Colajanni, Philip S. Yu, "Dynamic Load Balancing on Web-Server Systems," *IEEE Internet Computing*, Vol 3, No. 3, pp. 28-39, May/June 1999.

[5] Wensong Zhang, Shiyao Jin, and Quanyuan Wu, "Creating Linux Virtual Server," *The 5th Annual Linux Expo Conference*, May 1999.

[7] Thomas T. Kwan, Robert E. McGrath, and Daniel A. Reed, "NCSA's World Wide Web Server: Design and Performance," *IEEE Computer*, Vol. 28, No. 11, pp. 68-74, November 1995.

[8] Thomas T. Brisco, "DNS Support for Load Balancing," *Network Working Group RFC 1794*, April 1995.

[9] Daniel Andresen et al., "SWEB: Toward a Scalable World Wide Web-Server on Multicomputers," *Proceedings 10th IEEE International Symposium Parallel Processing*, pp. 850-856, 1996.

[10] Eric Anderson, Dave Patterson, and Eric Brewer, "The Magicrouter, An Application of Fast Packet Interposing," *Class Report*, University of California, Berkely, May 1996.

[11] Guerney D. H. Hunt et al., "Network Dispatcher: A Connection Router for Scalable Internet Services," *J. Computer Networks and ISDN Systems*, Vol. 30, Elsevier Science, Amsterdam, Netherlands, 1998.

[12] Om P. Damani et al., "ONE-IP: Techniques for Hosting a Service on a Cluster of Machines," *J. Computer Networks and ISDN Systems*, Vol. 29, Elsevier Science, Amsterdam, Netherlands, pp.1,019-1,027, Sepember. 1997.

[13] Paul Barford and Mark Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," *Processing of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 151-160, July 1998.