대규모 웹서버에서 확장성 및 가용성을 고려한 계층형 클러스터링 방안

연문숙, 조진성, 정병수
경희대학교 전자정보학부 컴퓨터공학과
zoom97@dblab.khu.ac.kr, {chojs, jeong}@khu.ac.kr

A Scalable / Highly Available Hierarchical Clustering Scheme in Large-scale WWW Servers

Moonsuk Yeon, Jinsung Cho, Byeong-Soo Jeong
Dept. of Computer Engineering, Kyung Hee University

요 약

인터넷 사용자의 증가와 사용자 요구사항의 다양화로 확장성과 고가용성을 지닌 고성능의 웹서버가 필요하게 되었다. 이러한 요구사항을 만족시키기 위해 여러대의 웹서버를 클러스터링하는 방안이 제시되고 있다. 본 논문에서는 단일 계층 웹서버의 문제를 해결하고 확장성(Scalability)과 가용성(Availability)을 높이기 위해 Top level에 복수개의 로드밸런서를 갖는 계층형 웹서버 클러스터를 제안한다. 본 논문에서 제안하는 시스템에서는 사용자의 요청을 동시에 받는 복수개의 로드밸런서를 Top level에 위치시켜 병목현상 및 치명적 오류 문제를 해결한다. 이러한 로드밸런서의 구성은 n level까지 확장할 수 있으므로 사용자의 증가 및 자료의 증가에도 유연하게 대체할 수 있다. 또한 각 level마다 다른 스케줄링 알고리즘을 사용할 수 있으므로 적응력 높은 웹서버를 구축할 수 있다. 본 논문에서 제안한 서비스 대기시간의 합을 가중치로 하는 동적 스케쥴링 알고리즘을 사용하여, Bottom level에 있는 리얼서버들 간의 부하 불균형을 최소화한다.

1. 서론

인터넷의 지속적인 발전으로 인터넷에 저장된 정보의 양은 기하급수적으로 증가하게 되었다. 또한 그 정보를 이용하려는 사용자의 수도 증가하게 되었다. 인터넷에서 정보를 얻는 방법으로 가장 많이 사용되는 것은 웹 서비스인데, 사용자의 수가 증가함에 따라 이에 대응하는 웹 서버의 성능 향상이 필요하게 되었다. 많은 사용자의 요청에도 만족할 만한 응답시간을 제공하고, 중단없는 서비스를 보장하는 것은 웹 서버의 기본 요건이라 할 수 있다. 또한 웹 서비스의 증가하는 정보를 계속해서 제공할 수 있어야 하며, 사용자의 증가에 따라 유연성 있게 성능을 높일 수 있는 확장성(Scalability)도 중요한 요소 중에 하나이다.

단일 웹 서버로는 위와 같은 여러 가지 요소를 만족시키기는 충분하지 않다. 이를 해결하기 위한 방법으로 여러대의 웹 서버들을 클러스터링(Clustering)하는 방법이 대안으로 제시되고 있다. 클러스터형 웹 서버 시스템이란 다수의 웹 서버를 네트워크로 연결하여 사용자의 요청을 클러스터내의 특정 웹 서버로 분산시켜 주는 구조로, 구성 형식에 따라 크게 중앙 집중형과 분산형 클러스터로 나눌 수 있다.

중앙 집중형 클러스터 구조[1]는 특정 부하분산 서버(Load Balancer)를 두어 사용자로부터 들어오는 모든 요청을 하나의 부하분산 서버를 거쳐서 서비스하는 형태로 그림 1과 같은 LVS[2,3]가 전형적인 중앙 집중형 클러스터 구조이다. LVS는 리눅스에서 클러스터 형태의 서버를 구축할 때 쓰이는 소프트웨어로 클러스터링 되어있는 병렬의 서버들을 단일 IP의 가상 서버로 인식하게 함으로써, 클러스터에 노드를 투명하게 추가 및 삭제하여 확장성을 높일 수 있다. 또한 노드나 데몬에 문제가 생기면 이를 바로 인식하여 시스템을 재구성하므로써 높은 가용성을 얻을 수 있다. 하지만 중앙 집중형 구조이기 때문에 부하분산 서버에 부하가 집중되어 병목현상(bottleneck)이 발생할 수 있고, 고장이 발생하면 시스템 전체가 동작하지 않는 치명적 오류의 지점이 될 수 있다는 단점이 있다.

분산형 클러스터 구조[4]는 실제 서버들이 모두 부하분산 기능을 하면서 서비스도 처리하는 형태이다. 즉 모든 실제 서버들은 사용자의 요청을 동시에 받게되고, 클러스터내의 다른 서버들과의 통신을 통해서 자신이 처리(accept)할지 폐기(discard)할지를 결정하게 된다. 이러한 분산형 클러스터 구조는 중앙 집중형 클러스터의

문제점을 해결할 수 있지만, 모든 실제 서버들이 서로간의 부하정보를 유지해야 하므로 추가적인 오버헤드가 발생하며, 병렬 구조로만 연결되기 때문에 확장성의 문제점을 안고 있다.

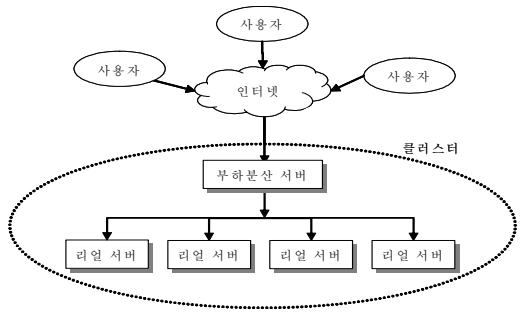

그림 1. LVS 구조도

본 논문에서는 대규모 웹서버에서 병목현상과 치명적 오류의 문제점을 해결하기 위해서 복수개의 로드밸런서를 두었으며, 트리(tree)구조의 계층형 웹서버 클러스터로 구성하므로써 확장성 및 가용성을 높이는 방안을 연구했다. 또한 복수개의 로드밸런서들 사이의 부하 균형을 맞추기 위한 방안과, 복수개의 리얼서버들 사이의 부하 불균형을 최소화하기 위한 동적 스케줄링 알고리즘을 제안하고자 한다.

본 논문의 구성은 서론과 2장에 단일 계층 웹서버 클러스터의 문제점에 대해 살펴보고, 3장에서는 계층형 클러스터 구조에 대해서 설명할 것이다. 4장에서는 본 논문에서 제안하고자 하는 계층형 웹서버 클러스터링 방안의 설계에 대해 기술할 것이며, 5장에서는 결론을 내리고 향후 연구 방향을 제시할 것이다.

2. 단일 계층 웹서버 클러스터의 문제점

클러스터형 웹 서버는 단일 웹 서버에 비교해서 확장성, 가용성, 경제성 등의 많은 장점을 가지고 있다. 그러나 로드밸런서와 리얼서버의 구조로 이루어진 클러스터형 웹 서버는 다음과 같은 몇가지 문제점을 내포하고 있다.

- **로드밸런서에 연결된 리얼서버 수의 제한**
  웹 서비스에는 일반적으로 사용자가 서버에 보내는 요청 패킷은 적은 양인데 반해, 서버가 사용자에게 보내는 응답 패킷은 매우 큰 경우가 많다. 이러한 비대칭적인 특징으로 한대의 로드밸런서가 여러대의 리얼서버를 관리 하는 것이 가능하다. 그러나 한대의 로드밸런서에 지나치게 많은 리얼서버를 연결시킨다면, 로드밸런서에 병목 현상이 발생할 것이다. 즉 로드밸런서 한대에 확장 가능한 리얼서버의 수는 로드밸런서의 성능에 따라 제한을 받기 때문에 대규모 웹 서비스를 지원하는 시스템에서 한대의 로드밸런서로는 성능 향상을 기대하기 어렵다.

- **부하 불균형**
  클러스터형 웹 서버의 경우 사용자의 접속 요청이 일정하지 않으며, 요청 파일의 패턴 또한 다양하기 때문에 서버들 사이의 부하 불균형이 발생할 가능성이 매우 크다. 부하 불균형이 발생을 하면, 시스템 전체로 봤을때는 자원이 효율적으로 사용되지 못하고, 사용자 입장에서는 기대할만한 응답시간(Response time)을 얻을 수 없고, 또한 부하 불균형이 개선되지 않는다면 더욱 가중되는 현상이 일어난다. 부하 불균형을 개선하기 위해서 기존에는 클라이언트와 서버 모두 특별한 변경이 필요없는 DNS 기반 웹 서버 클러스터를 많이 이용하였으나, DNS 캐싱에 의해서 같은 도메인의 요청은 항상 같은 웹 서버로만 몰리게 되는 편중현상이 발생하므로 최근에는 분배기를 이용한 부하분산 방식이 많이 쓰이고 있다. 분배기 기반의 부하분산 방식은 정적 부하분산 방식과 동적 부하분산 방식으로 나눌 수 있으며, 서버의 상태를 정확하게 반영할 수 있는 동적 부하분산 방식이 효율적으로 부하 불균형을 해소할 수 있다. 동적 부하 분산 방식에서 서버의 어느 정보를 부하로 정의 할 지는 신중히 고려해야 한다.

- **가용성**
  클러스터형 웹 서버는 여러 대의 서버로 구성되어 있기 때문에 특정 서버가 다운되어도 나머지 서버들로 계속해서 서비스를 해줄 수 있다는 장점을 가지고 있다. 그러나 이것은 리얼서버중에 한 서버가 다운되었을때 적용되는 경우이다. 만약에 한대의 로드밸런서와 다수개의 리얼서버 구조의 클러스터형 웹 서버에서, 로드밸런서의 다운은 시스템 전체의 중단을 의미한다.

3. 계층형 웹서버 클러스터 구조

계층형 웹서버 클러스터는 클러스터안의 웹 서버들이 전면서버와 후면서버의 단일 계층이 아니고, 그림 2와 같은 tree 형태의 다계층 구조를 갖는 클러스터를 말한다. 이 구조는 n level 까지 설계가 가능하므로 확장성을 최대한 높일 수 있고, 각 level마다 서로 다른 알고리즘을 사용할 수 있어 적응력 높은 웹서버 클러스터를 구축할 수 있다.

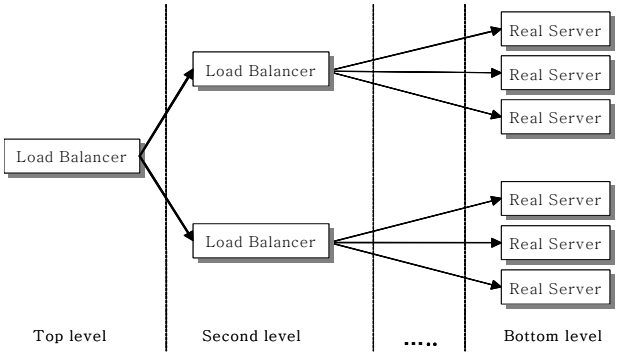

그림 2. 계층형 웹 서버 클러스터 구조

계층형 웹서버 클러스터는, 하나의 로드밸런서와 다수의 리얼서버를 갖는 중앙 집중형 클러스터와는 다르게 다수의 로드밸런서와 다수의 리얼서버를 갖는다. 즉 전체 노드중에 Bottom level의 노드들은 실제 서비스를 처리

하는 리얼서버가 되고, 나머지 앞 level의 노드들은 부하 분산을 담당하는 로드밸런서가 된다.

사용자의 요청을 받은 Top level의 노드는 Second level의 노드들에게 부하를 분산해 주고, Second level의 노드들은 다음 level의 노드들에게 부하를 분산해 준다. 이렇게 마지막까지 도착한 사용자의 요청은 Bottom level의 노드인 리얼서버에 의해서 처리된다.

4. 계층형 웹서버 클러스터링 방안의 설계

본 논문에서는 단일 계층 웹서버의 문제점을 해결하고 확장성(Scalability)과 가용성(Availability)을 높이기 위해서 그림 3과 같은 Top level에 복수개의 로드밸런서를 갖는 계층형 웹서버 클러스터를 제안하고자 한다.

그림 3. 제안하는 계층형 웹서버 클러스터

4.1 Load Balancing 방안

계층형 웹서버 클러스터는 여러 level로 설계할 수 있으며, level의 성격에 따라 서로 다른 알고리즘을 적용할 수 있다.

- **Top level의 Load Balancing 방안**

  사용자의 요청을 처음으로 받아 들이는 Top level에 복수개의 로드밸런서를 구성하므로써 병목현상 및 치명적 오류 문제를 해결할 수 있다[5]. 이 level의 로드밸런서들은 사용자의 요청을 동시에 받아서, 사용자의 요청을 분석하고 내용기반(Content aware) 알고리즘[6,7]을 이용하여 부하분산을 실행한다. 로드밸런서에서 이루어지는 작업을 모듈별로 나누어 보면 표 1과 같다. Top level의 로드밸런서들은 전체 웹 시스템이 서비스 하는 컨텐츠 리스트(Content List)를 중요도 순으로 정렬하여 가지고 있다. 중요도는 자신이 주로 서비스 하는 컨텐츠의 목록을 우선하며, 이때 서비스할 목록의 비율은 Top level의 로드밸런서 수에 기반하여 정한다. 예를 들어 Top level에 두개의 로드밸런서가 있으면 서비스 목록의 50%를 자신이 서비스할 목록으로 생각한다.

  각 모듈들의 세부 내용을 살펴보면, 연산 모듈내의 모듈들은 일정간격으로 백그라운드에서 연산을 실행하여, 분석 모듈과 분산 모듈에 필요한 데이터를 재구성한다. 분석모듈은 사용자 요청의 URL을 파싱(parsing)하여[8] 해당하는 서비스 요청이 자신이 서비스할 목록과 일치하면, 분산 모듈로 보내고 그렇지 않으면 폐기한다. 분산 모듈에서는 자신과 연결되어 있는 next level의 서버들중 부하가 가장 적은 서버를 선택하여 요청을 보낸다. 이와 같은 모듈들의 상호동작은 그림 4와 같다.

표 1. Top level의 Load Balancer 작업 모듈들

연산모듈	• 로드밸런서 연산모듈 : 로드밸런서들의 작업량 정보를 이용하여 자신이 서비스할 목록의 비율을 계산한다. • next level 연산모듈 : 다음 level 서버들의 부하정보를 받아 가중치를 계산한다. • current level 연산모듈 : 로드밸런서 자신이 처리한 작업량을 카운트한다.
분석모듈	• 사용자의 요청을 분석하여, 요청한 컨텐츠 목록이 자신의 서비스 목록과 일치하는지 비교한다.
통신모듈	• 다른 서버와의 통신을 담당한다.
분산모듈	• next level 연산모듈의 정보를 토대로 next level에서 최적의 서버를 선택한다.

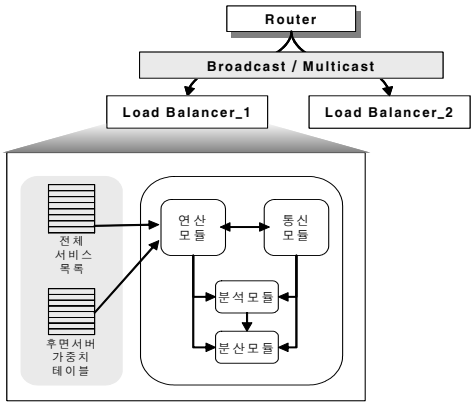

그림 4. 모듈들의 상호동작

- **중간 level의 Load Balancing 방안**

  구축하고자 하는 웹서버의 성격에 따라 다양한 부하분산 알고리즘을 적용할 수 있다. 상하 level의 알고리즘과 연계된 알고리즘을 적용하면 더욱 효과적인 부하분산을 기대할 수 있다.

- **Bottom level의 Load Balancing 방안**

  웹 서버의 부하 정보는 사용자 요청에 대한 응답시간에 영향을 미치는 모든 요소를 말한다[9]. 본 논문에서는 대기 행렬(Server Qeueue)에서 아직 처리되지 않고 남아 있는 요청들의 서비스 시간의 합(대기 시간)을 부하 정보로 정의한다.

  각 리얼서버들은 대기 시간의 합을 주기적으로 상위 level의 로드밸런서에 전송한다. 로드밸런서의 연산 모듈에서 부하 정도에 반비례하는 리얼서버의 가중치를 부여하고 부하 정보를 읽어 들일 때마다 동

적으로 가중치를 갱신해 준다. 결과적으로 부하가 가장 적은 리얼서버에 사용자의 요청을 할당한다.

4.2 부하 불균형 최소화 방안

사용자의 요청이 모든 서비스에 대하여 균일하게 요청되지 않는 다면, 내용기반으로 Load Balancing을 하는 Top level 로드밸런서들의 부하 불균형이 발생할 가능성이 높다. 이를 해결하기 위해서 일정시간 동안 처리한 요청의 건수를 카운트하여 다른 로드밸런서에 전송한다. 전송받은 다른 로드밸런서 카운트정보와 자신의 카운트 정보를 비교하여 서비스 목록의 비율을 재계산한다. 이렇게 함으로써 사용자의 요청이 많은 서버는 서비스할 목록을 줄이고, 요청이 적은 서버는 서비스할 목록을 늘리므로 로드밸런서간의 부하 불균형을 최소화 할 수 있다.

Bottom level 리얼서버들은 상위 level의 로드밸런서와 연결되어지는 sub cluster를 형성하고 있다. 그 sub cluster 안에서의 리얼서버들은 대기시간의 합을 부하정보로 하여 일정시간마다 동적으로 가중치를 재계산해 주므로써 sub cluster안의 리얼서버간의 부하불균형을 최소화 할 수 있다. sub cluster들 사이의 부하 균형은 그림 5의 (a)와 같이 부하가 적은 쪽의 리얼서버를 부하가 많은 sub cluster쪽으로 이주함으로써 해소할 수 있다.

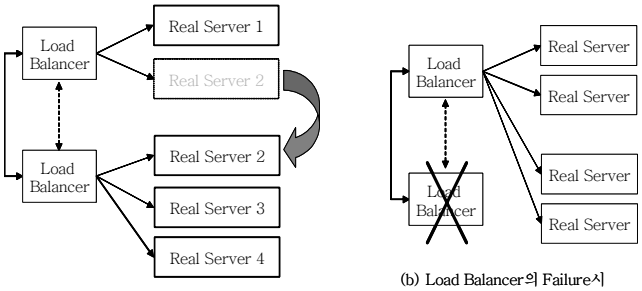

그림 5. Cluster의 재구성

4.3 고 가용성 확보를 위한 방안

웹서버의 신뢰성을 위해서는 클러스터안의 일부 서버에 고장이 발생하는 경우에도 중단 없는 서비스를 제공해야 하는데, 그러기 위해서 클러스터 안의 서버들은 서로간의 상태정보를 교환해야 한다.

Top level의 로드밸런서들은 자신이 처리한 요청의 건수를 카운트하고, 이 정보를 주기적으로 같은 level의 로드밸런서들에게 전송한다. 전송받은 정보와 자신의 처리건수를 비교하여 서비스 목록의 비율을 재계산하게 되는데, 이때 일정시간이 경과했는데도 다른 로드밸런서의 정보가 도착하지 않으면 failure로 간주하고 서비스 목록 비율을 확대하여 상대 로드밸런서의 서비스 목록까지 지원하여 그림 5의 (b)와 같이 지속적인 서비스를 제공할 수 있다.

리얼서버의 경우는 요청들의 대기시간의 합을 주기적으로 상위 level의 로드밸런서에 전송하게 된다. 이때 일정시간이 경과했는데도 부하정보가 도착하지 않으면 failure로 간주하고 가중치 계산시 0(zero)를 곱하여 주므로써 부하분산에서 제외시키다가 정상으로 회복이 되어 부하정보를 정상적으로 보내게 되면 다시 가중치를 계산하여 서비스에 참여시킨다.

5. 결론 및 향후 연구 과제

본 논문에서는 단일 계층 웹서버의 문제점을 해결하고 확장성(Scalability)과 가용성(Availability)을 높이기 위해서 Top level에 복수개의 로드밸런서를 갖는 계층형 웹서버 클러스터를 제안하였다. 또한 서비스 요청의 대기시간의 합을 가중치로 하는 동적 스케줄링 알고리즘을 사용하여 리얼서버들 간의 부하 불균형을 해소할 수 있는 방안을 제안하였다.

대규모 웹서버 클러스터 설계시 n개의 노드가 주어졌을 때, 로드밸런서와 리얼서버의 최적의 비율을 찾기위한 연구와, 리얼서버들 간의 가중치 기반의 스케줄링 알고리즘에서 과거시점의 부하정보를 이용하여 스케줄링하는 것이 아니고 현재의 부하를 바로 적용할 수 있는 방법에 대한 연구를 계획해 나갈 것이다.

참 고 문 헌

[1] D. Dias, W. Kish, R. Mukherjee, and R. Tewari, "A Scalable and highly available web server", Ieee International Conference on Data Engineering, New Orleans, February, 1996

[2] Linux Virtual Server open project sites, http://www.linuxvirtualserver.org/

[3] "Linux Virtual Servers for Scalavle Network Services" Ottawa Linux Symposium July 19th~20th, 2000.

[4] A. Dahlin, M. froberg, J. Walerud and P. Winroth, "EDDIEA:A Robust and Scalable Internet Server", 1998

[5] IA-LVS: Design of the Improving Availability for Linux virtual Server", 2nd International conference on Software Engineering,, Artificial Intelligence, Networking & Parallel/Distributed Computing August 20-22, 2001 Nagoya Institute of Technology, Japan

[6] Trevor Schoeder, Steve Goddard, Gyrav Ramamurthy, "Scalable Web Server Clustering Technologies" IEEE Network, May/June, 2000

[7] George Apostolopoulos, David Aubespin, Vinod Peris, Parshant Pradhan, Debajan Saha, "Design, Implementation and Performance of a Content-Based Switch", IEEE INFOCOM, March, 2000

[8] George Apostolopoulos, Vinod Peris, Prashant Pradhan, Debanjan Saha, "L5: A self learning Layer 5 switch", IBM research report

[9] Wenson Zhang, Shiyao Jin, Quanyuan Wu, "Creating Linux virtual servers", LinuxExpo Conference 1999