IP 필터링 방식을 사용하는 클러스터드 웹서버의 부하 분산 정책

김 재 천(金 在 天), 최 상 방(崔 相 昉)
인하대학교 전자공학과
전화 : (032) 860-7417 / 팩스 : (032) 868-3654

Load Balancing Policy in Clustered Web Server Using IP Filtering

Jae Cheon Kim, Sang Bang Choi
Electronic Engineering Inha University
E-mail : kjc0808@nownuri.net, sangbang@dragon.inha.ac.kr

Abstract

As Internet and WWW grow rapidly, the role of web servers is getting more important, and the number of users in a few popular site is also explosively increasing. Load balancing in clustered web server systems is important task to utilize whole system effectively, so dynamic load balancing is required to overcome the limit of static load balancing. In this paper, we propose two dynamic load balancing schemes, and analyzed load model and performance improvement and also compare existing load balancing methods and IP filtering method. In case of load balancing with threshold, little extra traffic was required for better performance, but in case of load balancing with load weight, we found that the performance mainly depends on information exchange rate.

I. 서론

현재 인터넷의 주류를 이루는 웹의 발전으로 보다 많은 요구를 보다 빠르게 처리하고자 하는 노력은 계속 되어져 왔으며 다양한 형태의 개선방향들이 나오고 있다.

개별서버의 성능을 높여서 증가하는 클라이언트(client)의 요구에 대처하기 힘들다는 것은 이미 알려진 사실이며 비용면에서도 클러스터링 방식이 훨씬 저렴하다. 응답속도를 줄이기 위해서 클라이언트 측에서는 사용하는 방법으로는 캐싱(caching)이나 프리팻칭(prefetching)등을 사용하는 방법이나 웹서버의 컨텐츠들을 클라이언트 쪽으로 좀더 가까이 배치함으로서 사용자가 매번 웹서버에 직접 가기보다는 가까이에 있는 프락시 서버를 이용하여 기존에 방문한 페이지 혹은 로컬네트웍내의 다른 사용자가 방문한 페이지를 바로 얻어오는 방법이 있다. 또한 컨텐츠 형식의 개선을 통하여도 사용자의 체감속도를 향상시킬 수 있으나 여기에도 한계가 있는 것이 사실이다.

IP 필터링 방식은 클러스터링되어 있는 서버들이 모두 고유의 필터(filter)를 가지고 자신이 처리할 패킷을 걸러내는 방식을 취하는데 추가적인 하드웨어를 필요로 하지 않으면서 네트웍 구조를 변경시키지 않고 모든 트래픽을 제어할 수 있기 때문에 가장 확장하기 쉬우면서도 저렴한 비용으로 부하분산을 실현할 수 있다. IP 필터링 방식은 개별서버에 부하분산기능을 탑재하여 독립적으로 동작하게 되므로 다른 방식에서 볼 수 있는 단일 병목지역(single point of bottleneck)이나 단일 오류 지점(single point of failure)이 발생하지 않는데 이를 통해 지속적으로 서비스를 유지해야 하는 경우 시스템의 신뢰성과 가용성을 높일 수 있다[1].

불규칙한 클라이언트의 요구로 인해 발생하는 부하

의 집중은 IP 필터링 방식을 사용하여 클러스터링된 서버들에서도 나타나게 되며 클러스터 내부에 있는 서버간의 균형 있는 부하분산을 위해서는 집중되는 부하를 상대적으로 부하량이 낮은 서버들에게 동적으로 분산시킬 필요가 있다. 정적인 부하분산기능이외에 동적인 부하분산기능을 추가하기 위해서는 서버의 현재 부하량을 알아야하며 이 정보를 상호 교환하는 과정이 필요하다[2],[3].

동적 부하분산 정책으로 사용한 방식은 첫째로 특정 서버가 허용 부하 임계치(threshold)를 넘어서는 경우 이를 클러스터링에 참여하는 서버들에게 브로드캐스팅하여 서버들이 가지고 있는 필터함수(filter function)를 수정하는 방식이며 부하가 다시 임계치 이하로 낮아지는 경우 다시 클러스터링에 참여하는 형태를 취한다. 이러한 방식은 서버에 과부하가 걸린 경우에만 추가 트래픽을 발생시키므로 이에 따른 오버헤드(overhead)는 아주 작은 값을 가진다. 두 번째 방식은 서버의 부하상태를 주기적으로 체크하여 이에 해당하는 비율대로 가중치를 계산하고 각 서버에 해당하는 가중치를 부가한다. 다음 주기까지는 이 가중치가 각 서버의 필터링함수에 적용되는데 이 과정에서 클러스터링에 참여하는 서버 중 하나가 마스터(master)가 되어 부하정보를 수집하고 재분배하는 책임을 담당한다. 짧은 주기를 가질수록 실시간으로 동적 부하분산을 할 수 있는 효과를 가지나 이를 위한 내부네트웍을 추가로 구성하지 않는다면 상대적으로 네트웍에 많은 트래픽을 발생시킨다.

임계치를 사용하는 경우 각 서버가 자신의 부하량을 모니터링하고 임계치를 넘어설 경우 더 이상 부하를 수용하지 않는 방식으로 과부하가 발생할 경우에만 트래픽이 발생하게 되므로 상대적으로 적은 트래픽으로도 동적 부하분산효과를 거둘 수 있었다. 이 방식은 과부하를 판단하는 시점 즉, 임계치의 설정에 따라 다른 부하분산효과를 가져오므로 전체 트래픽의 양을 고려하여 이 임계치를 결정하여야 한다. 가중치를 사용하는 경우는 각 서버의 부하상태를 수집하여 이에 따른 가중치를 두고 부하분산을 하는 방식이다. 서버간의 통신이 상대적으로 많이 발생하게 되므로 이에 따른 추가 트래픽을 별도의 네트웍을 구성하여 분리시킬 필요가 있다. 이 방식은 가중치를 재 계산하기 위해 부하량을 수집하는 간격과 이를 바탕으로 가중치를 계산하는 과정에서 사용하는 배수 값에 따라 다른 부하분산 효과를 보여주었다. 배수 값이 높을수록 서버간에 보다 선형적인 부하분산을 가져오는 반면 부하분산 효과는 낮아졌고 실시간으로 서버들의 부하상태를 모니터링할 경우에는 이상적인 부하분산효과를 볼 수 있었으나 재 계산간격이 늘어날수록 성능이 감소하게 되므로 가중치를 재계산 하는 간격을 최대한 줄여야 함을 알 수 있었다.

Ⅱ. 웹서버의 부하분산 기법

서버측에서 네트웍 트래픽을 분산시키는 기법 중 가장 단순하면서도 원시적인 방법은 서버의 주소를 나열하여 사용자로 하여금 임의로 선택하게 하는 방법이다. 물론 어떤 서버가 최선의 선택인지 사용자는 알 수 없으며 몇몇 서버에 부하가 집중되는 현상을 막을 수 없다. 이로 인해 외부에 단일한 주소로 알려진 서버군을 형성하여 사용자에게 투명성(transparency)을 제공하는 기법이 필요하다[3].

DNS를 이용한 부하분산방식에서는 DNS가 클라이언트에서 오는 요구를 분산시키는 역할을 하게되며, 서버측에 네트웍의 구조나 소프트웨어의 특별한 변경없이 적용할 수 있다는 장점이 있으나 DNS서버 고유의 특성상 모든 트래픽을 완벽하게 제어할 수 없는 단점을 가지고 있다. 또한 IP 분산기를 사용한 방식은 모든 트래픽을 IP 분산기가 제어할 수 있기 때문에 부하분산측면에서 높은 효율을 기대할 수 있고 부하의 불균형에 능동적으로 대처할 수 있다. 하지만 모든 트래픽이 한 노드(node)를 경유함에 따라 IP 분산기 자체가 병목지역이 될 수 있으며 IP 분산기 자체에 오류가 생길 경우 전체 서비스가 중단되는 치명적인 결함을 가지고 있기 때문에 고장허용(fault tolerance) 측면에서 다소 위험부담을 지닌 방식이라 할 수 있다[2]. 마지막으로 4계층 스위치(Layer 4 Switch)라고도 불리는 하드웨어적인 장치를 이용하여 트래픽을 분산시키는 방법은 모든 트래픽의 TCP/IP 헤더를 번역하여 적절한 서버로 맵핑시켜주는 기법으로 분산스위치는 각 커넥션 상태를 모니터링하고 있어야 하는데 클라이언트에서 서버로 오는 패킷뿐 아니라 서버로부터 클라이언트로 가는 트래픽 역시 재 번역과정을 거쳐 나가게 되므로 모든 패킷을 처리하는데 과중한 부하가 생기게 되어 스위치 자체가 서비스에 병목지역이 될 수 있다.

Ⅲ. IP 필터링 방식

IP 필터링 방식은 클러스터링된 서버들간에 공통된 하나의 가상주소(virtual address)를 필요로 한다. 이를 위해서 각 서버들이 고유의 주소(private address)이외에 서버군을 대표하는 하나의 주소를 가지게 되는데 이 주소가 클라이언트 측에 알려진다[4],[5].

동적으로 시스템간의 부하분산을 하기 위해서는 각 시스템의 부하상태를 알고 있어야 하지만 단순히 과부

하상태가 된 서버가 발생하였을 때 자신의 상태를 다른 서버에 알려줌으로써 클라이언트 IP 영역을 재분배하는 효과를 거둘 수 있다. 또한 각 서버의 부하상태에 따른 비율을 결정하여 이에 해당하는 비율로 IP 영역을 분배하면 더 높은 성능향상을 기대할 수 있지만 이때는 각 서버의 상태를 수집하여 분배 비율을 되돌려주기 위해 부하분산시점에서 특정서버가 각 서버의 부하 정보를 수집하여야만 한다. 서버 중 하나가 이를 담당하게되면 단일 오류지점이 생기는 문제점은 피할 수 있지만 서버상태를 수집하기 위한 트래픽의 증가문제는 여전히 남아 있게 된다.

Ⅳ. 부하분산 알고리즘 및 부하의 지표

IP 필터링 방식의 특성상 모든 트래픽을 모니터링하는 지점이 없기 때문에 부하의 분산은 나머지(modular)연산을 기반으로 한다. 정적인 부하분산 방식으로 발생된 부하의 불균형을 그때그때 조정하기란 미래의 트래픽을 예측하기 힘든 상황에서는 어려울 수밖에 없으며 이를 동적으로 조율해 줄 수 있는 알고리즘을 필요로 한다. 이때 각 서버의 부하량을 수집하여 각 서버에 대한 가중치를 부여하는 방법, 또는 응답속도나 우선순위에 따라 가중치를 부여하는 방법 등을 사용할 수 있다.

본 논문에서 사용한 부하의 지표는 웹서버에의 성능에 영향을 미칠 수 있는 변수들을 종합한 후 개별 변수들에 가중치를 주어 산출한 값을 기준으로 실험을 하였으며 동적 부하분산알고리즘은 서버의 부하량을 기준으로 가중치를 부가하였다. 모델 2 즉, 임계치에 의한 부하분산방식에서 부하분산기능을 탑재한 서버들은 부팅시에 고유의 호스트 ID를 가지게 되며 서버 수 n을 나머지연산에 사용한다. 과부하상태가 된 서버는 이를 주기적으로 체크하여 자신이 더 이상 추가적인 작업을 처리할 수 없음을 다른 서버들에게 브로드캐스팅하게 되고 모든 서버의 필터링 함수의 k_1=CA mod n이 k_2=CA mod (n-1)값으로 바뀌게 된다 차후 현재 처리중인 작업이 끝나고 다시 추가작업을 할 수 있는 상태가 되면 다시 자신의 상태를 다른 서버에게 알려 각서버들이 담당할 IP영역을 재 계산하게 된다.

모델 3 즉, 가중치에 의한 부하분산방식에서는 각 서버의 부하량에 따라 다른 가중치를 가지고 IP 영역을 배분하기 위해서는 클러스터링에 참가하는 서버 중 하나가 마스터가 되어 각 서버의 상태정보를 수집할 필요가 있다. 이때 상태정보를 보내오는 응답시간을 고려하면 시스템 부하상태 이외에도 네트웍 부하상태에 대한 정보를 얻을 수 있다 이러한 IP영역의 분배방법은 상대적으로 많은 양의 트래픽을 발생시키기 때문에 전체성능에 영향을 주지 않는 범위 내에서 주기를 결정하거나 별도의 내부네트웍을 구성하여 순수한 서비스 트래픽과 분리시킬 필요가 있다. 각 서버는 먼저 마스터가 될 것인가를 판단하고 마스터가 될 경우 각 서버에 부하량 전송요구를 보낸다. 마스터로 수집된 부하량을 토대로 각 서버에 대한 가중치를 산출한 후 이 값을 다시 전송하게 되며 마스터가 아닌 서버의 경우 마스터로부터의 부하전송요구에 응답한 후 자신의 가중치를 전송 받아 필터링에 반영한다. 서버간의 가중치를 계산하기 위해서는 서버들의 부하상태에 대한 비율을 먼저 계산하고 이에 대한 역수를 취하거나 일정 값에 대한 차를 구하는 방법으로 가중치를 계산한다. 이 가중치를 차기 부하분산에 적용할 경우 보다 선형적(linear)인 부하의 분산을 할 수 있다.

서버간의 부하가 고르게 분포되어 있을 경우 개별 서버가 담당하는 부하는 상대적으로 낮아지므로 전체 서비스의 성능이 향상되게 된다. 서버들의 부하분산이 얼마나 잘되어 있는가를 나타내는 지표로 LBM(Load Balance Metric)을 사용하는데 이는 평균 부하에 대한 최대부하의 비율이다.

Ⅴ. 모의실험 및 성능 비교

모의 실험에서 사용한 클라이언트의 분포는 비교적 고르게 분포되어 있으나 실제로 특정서버에 접속하는 클라이언트의 분포는 특정 IP 주소에 편중되는 현상을 나타내게 되며, 이 경우 정적인 부하분산방식만으로 부하분산을 할 경우는 일부 서버에 부하가 집중되는 현상이 더욱 커진다. 모델 2와 모델 3의 그래프에서는 일부서버에 집중되는 트래픽이 나머지 서버들로 분산되는 효과를 가져오기 때문에 서버간에 부하의 격차가 줄어들고 있음을 알 수 있다.

모델 2의 경우 시스템의 부하량이 임계치를 넘어서면 더 이상 트래픽을 수용하지 않는 방식이기 때문에 이 한계치의 설정이 웹서버 클러스터의 성능에 결정적인 역할을 하게 된다. 모델 3의 경우 실시간으로 가중치를 두고 부하분산을 하는 경우 이상적으로 부하분산이 이루어져 있음을 알 수 있었다. 하지만 가중치 계산간격이 늘어날수록 부하분산 효과가 떨어지게 되므로 서비스 트래픽에 영향을 주지 않는 내부네트웍을 통해 가중치 계산 간격을 좁혀야 한다. 목적부하를 산출하는 과정에 사용한 배수 값은 각 서버에 분산되는 부하를 얼마나 선형적으로 조정하는가를 결정하게 되는데 이 값이 클수록 서버간의 부하의 변화는 보다 선형적인 형태를 보여주고 있다. 모델 3에서 가중치의

재계산 간격은 부하분산에 가장 큰 영향을 미친다. 간격이 커질수록 부하분산효과는 계속 떨어지게 되므로 시스템의 성능에 영향을 주지 않는 범위 내에서 최대한 짧게 조정하여야 한다.

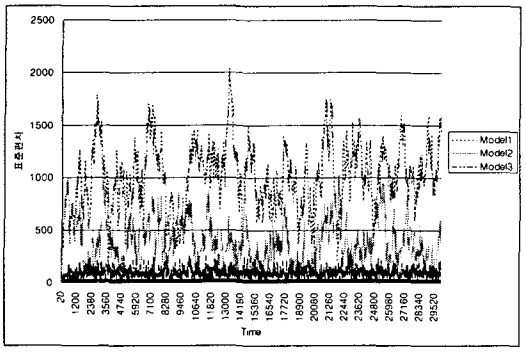

그림 1. 모델별 부하의 표준편차
Figure 1. Loads standard variation

그림 1은 각 모델의 표준편차를 나타낸 그래프이다. 각 모델에서 가장 우수한 성능을 보이는 방식을 기준으로 하였다. 모델 1의 경우 서버간의 표준편차가 가장 작은 값을 보이고 있으며 모델 2의 경우에도 정적인 부하분산 방식의 모델 1보다 전반적으로 낮은 표준편차 값을 유지하고 있다. 표준편차는 클러스터링 서버들의 LBM값과도 밀접하게 관련하고 있으며 서버간의 부하분산 정도를 나타내는 지표가 될 수 있다.

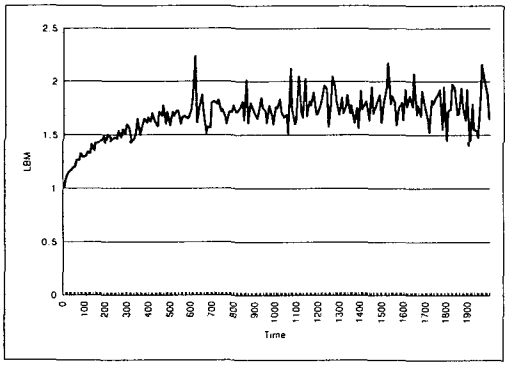

그림 2 가중치 재계산 간격에 따른 LBM의 변화(모델 3)
Figure 2. LBM by weight calculation interval

모델 2의 경우 임계치의 설정이 부하분산에 가장 많은 영향을 주며 모델 3의 경우 재계산 간격이 작을수록 높은 성능을, 가중치 계산 배수가 낮을수록 높은 성능을 보여주었는데 이들 값에 따라 모델 1이나 모델 2보다도 낮은 성능을 보여주었다.

VI. 결론

본 논문에서는 정적인 부하분산방식의 한계를 극복하기 위하여 2가지의 동적인 부하분산 방식을 제안하였다. 정적인 부하분산방식에 동적인 부하분산기법을 추가한 경우 정적인 부하분산방식에서 편중되는 부하를 다른 서버로 나누어줌으로써 전체 트래픽을 보다 효율적으로 배분하는 효과를 거둘 수 있었다.

모델 2의 경우 각 서버가 자신의 부하량을 모니터링하고 임계치를 넘어설 경우 더 이상 부하를 수용하지 않는 방식으로 과부하가 발생할 경우에만 트래픽이 발생하게 되므로 상대적으로 적은 트래픽으로도 동적 부하분산효과를 거둘 수 있었다. 모델 3은 각 서버의 부하상태를 수집하여 이에 따른 가중치를 두고 부하분산을 하는 방식이다. 서버간의 통신이 상대적으로 많이 발생하게 되므로 이에 따른 추가 트래픽을 별도의 네트웍을 구성하여 분리시킬 필요가 있다. 배수 값이 높을수록 서버간에 보다 선형적인 부하분산을 가져오는 반면 부하분산효과는 낮아졌고 실시간으로 서버들의 부하상태를 모니터링할 경우에는 이상적인 부하분산효과를 볼 수 있었으나 재 계산간격이 늘어날수록 성능이 감소하게 되므로 가중치를 재계산 하는 간격을 최대한 줄여야 함을 알 수 있었다.

참고문헌

[1] Richard B.Bunt, Gregory M. Oster, Derek L. Eager, Carey L. Williamson, Achieving Load Balance and Effective Caching in Clustered Web Servers, 1999

[2] German Goldszmidt and Guerney Hunt, NetDispatcher: A TCP Connection Router 1997

[3] Michele Colajanni, Phillip S.Yu, Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems, IEEE Transaction on Parallel and Distributed system vol.9 June 1998

[4] Om P. Damani, P. Emerald Chung, Yennun Huang, Chandra Kintala, and Yimin Wang, ONE-IP: Techniques for Hosting a Service on a Cluster of Machines, 6th International WWW Conference, April. 1997

[5] Shie-Yuan Wang and Rauel Hill, IP Clustering for building a scalable high-performance web server cluster on a BSD-derived UNIX router, 1997