공학석사 학위논문

시스템의 부하 예측을 통한 웹에서의 부하 분산 연구

지도교수 서 경 룡

이 논문을 공학석사 학위논문으로 제출함

2003년 2월

부경대학교 대학원

컴퓨터공학과

이 석 주

이석주의 공학석사 학위논문을 인준함

2002 년 12 월 26 일

주 심 공학박사 정목동 

위 원 공학박사 권오흠 

위 원 공학박사 서경룡 

목 차

그 림 목 차

표 목 차

시스템의 부하 예측을 통한

웹에서의 부하 분산 연구

이 석 주

부 경 대 학 교 대 학 원 컴 퓨 터 공학과

요약

본 논문에서는 급증하는 웹 서비스의 부하를 해결하기 위해서 LVS 시스템에서 부하 분산 서버의 새로운 스케쥴링을 제안한다. 새로운 부하 분산 스케쥴링은 부하 분산 서버가 RealServer와의 SNMP 메시지 교류를 통해 사용 가능한 메모리의 사용량과 1분 동안의 평균 부하량의 시스템 상태 정보를 획득한 후 Newton의 보간 다항식을 이용하여 부하 예측식을 생성하고 이를 통하여 균등 부하 분산을 하는 것을 목적으로 한다.

부하 예측을 통하여 시스템 부하에 미치는 영향을 최소로 하고 시뮬레이션 결과, 효율적인 부하 분산을 할 수 있는 것을 입증하였다. 또한, 실제 부하 값에는 미치지 못하지만 부하 예측 식을 통하여 오차를 감소시킴으로써 부하 예측값을 실제 부하 값에 근접할 수 있도록 하였다.

A Load balancing strategy on Web cluster with load prediction

Suk Joo Lee

Dept. of Computer Eng., Graduate School,

Pukyong National University

Abstract

In this thesis, we propose new scheduling method of load balancing server in LVS system for reduce load of web service. The new load distribution scheduling get available memory and system status information of average load for 1 min. using communication with RealServer. After it generate load prediction equation using Newton's interpolation and

distribute load.

We minize the effect to system load using load prediction. In the result of simulation, we proof that this method can distribute load effectively. Also, the load prediction is closed to real load by reducing error through load prediction equation.

1. 서 론

최근 인터넷의 사용이 급속히 확산되어 전자상거래 사이트와 같은 웹 서비스의 사용이 증가하고 있으며 QoS 서비스가 요구되는 인터넷 상영관과 음반 사이트 등의 활용 분야가 증가함에 따라 그에 따른 부하도 증가하고 있다. 또한, 인터넷 사용자들의 요구가 다양해지고 양질의 서비스를 받기 원함으로써 서비스 시스템의 처리 능력에 대한 수요가 더불어 증가하고 있다[1]. 하지만 서비스 시스템의 처리 능력이 사용자들의 요구를 충족시키기 위해서는 고가의 장비가 필요하고 기하급수적으로 늘어나는 시스템의 부하를 단일 시스템의 처리 능력만으로 해결할 수가 없다.

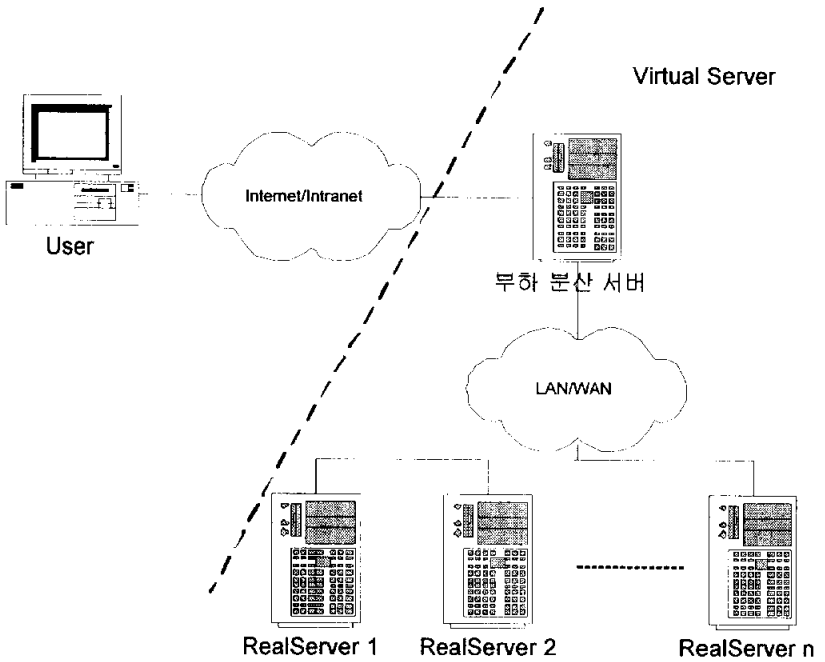

그림 1-1 LVS 시스템 구성도

이러한 문제를 해결하기 위해서 그림 1-1과 같이 저가 장비를 사용하면서도 고 성능과 고 가용성을 발휘할 수 있는 LVS 시스템 사용이 일반화되는 추세이다.

1.1 LVS 시스템

LVS는 Linux Virtual Server의 약자로서, 리눅스 서버를 통하여 구성된 고성능 고가용성의 부하 분산 시스템을 말한다. 하지만 요즘엔 부하 분산시스템 전체를 뜻하는 말로 통용되고 있으며 사용자에게는 서버가 한대만 있는 것으로 인식되지만, 실제로는 여러 대의 컴퓨터가 서비스를 수행하도록 구성된 것을 말한다.

LVS 시스템은 사용자들의 요청에 응답하는 실제 시스템인 RealServer와 이들 RealServer와 사용자들을 연결시켜주며 부하 분산을 담당하는 부하 분산 서버로 나누어진다. RealServer는 고속의 LAN이나 지역적으로 분산된 WAN에 연결되어있다. RealServer의 전면(front-end)에 있는 것이 부하 분산서버이다. 부하 분산서버는 들어오는 요청에 대하여 서로 다른 서버로 스케쥴링을 하며, 클러스터로 구성된 병렬의 서비스를 단일 IP의 가상 서버로 인식하도록 한다. 클러스터에 노드를 투명하게 추가 및 삭제하여 확장성을 높일 수 있다. 노드나 데몬에 문제가 생기면 이를 바로 인식하여 그 즉시 시스템을 재구성하므로 높은 가용성을 얻을 수 있다.

서버 클러스터 구성에도 몇가지 방법이 있다. 일반적으로 많이 사용되는 방법 중 하나가 Round-Robin DNS이다. 라운드 로빈 DNS는 여러 아이피에 단일한 이름을 주는 방법이다. 그러면 서로 다른 클라이언트는 서로 다른 서버에 접속을 하게 되는 것이다. 이런 방법을 이용하면 부하가 여러 서버사이에 분산된다. 그런데 클라이언트와 계층 DNS 시스템의 캐쉬 특성때문에 서버들 사

이에 부하가 불균등하게 분산된다. 그러므로 부하가 최고로 올라갈 때 서버를 관리하기가 쉽지 않다. TTL(Time To Live)값을 정하기가 쉽지 않다. 값을 작게 주면 DNS서버에서 병목이 걸리고 값을 높게주면 서버간의 부하불균형이 더 심해진다. TTL을 0으로 주더라도 스케쥴링의 기본은 호스트이기때문에 사용자들의 접근 형태때문에 부하의 불균형이 심해질 수 있다. 왜냐하면 어떤 사용자는 사이트에서 많은 정보를 가지고 갈 수 있고 어떤 사용자는 잠깐 보고 그냥 갈수도 있기 때문이다. 더군다나 신뢰하기 힘든 점은 특정한 서버 노드에 문제가 생겼을때 그 노드에 접속하는 사용자는 서버가 다운되었다고 생각할 것이며 브라우저에서 "reload"나 "refresh"를 눌러도 마찬가지로 문제는 해결되지 않는다.

이보다 좋은 방법은 클러스터의 서버사이에서 부하를 분산하는 부하 분산기를 사용하는 것이다. 서버는 병렬로 구성되었다고 하더라도 실제 사용자들은 하나의 IP만 가진 가상 서버만을 볼 수 있으며 실제 서버는 보지 못한다. 스케쥴링의 기초는 연결이며 이는 서버의 부하분산에 더 효과적이다. 한대 또는 그 이상의 서버에서 문제가 생겼을 경우 기록을 남긴다. 서버관리가 더 쉬워지고 관리자는 사용자들에게 계속 서비스를 제공하면서 언제라도 서버를 교체할 수 있다.

부하 분산은 애플리케이션 레벨과 IP 레벨로 구성할 수 있다. 예를 들어 리버시 프록시와 pWEB은 애플리케이션 레벨의 부하분산 방법이다. HTTP요구를 클러스터로 구성된 서로 다른 웹서버로 보내고 그 결과를 받아서 다시 클라이언트에게 넘겨준다. 애플리케이션 레벨에서는 HTTP요청과 결과를 돌려주는

과정에서 과부하가 크기 때문에 서버 노드가 각 서버의 총 처리량보다 증가하면 애플리케이션 레벨의 부하분산기 자체에서 병목현상이 생길 수 있다.

일반적으로 IP 부하분산을 하는것이 과부하가 적다. 그래서 서버노드의 최대 숫자가 25개에서 100개까지 올라갈 수 있다. LVS 시스템은 이러한 목적을 위해 설계가 되었다.
살펴본 바와 같이 LVS 시스템의 효율을 높이기 위해서는 RealServer의 시스템 성능도 중요하나 각 RealServer로의 부하 분산을 담당하는 부하 분산 서버의 역할이 크다 하겠다. 부하 분산 서버는 RealServer와 사용자 요청 사이에서 병목 현상을 유발시킬 수 있으며 적절한 부하 분산을 하지 못하면 RealServer의 부하를 특정 시스템에 가중시킴으로써 LVS 시스템의 효율을 저하시킨다. 이에 LVS 시스템에서는 균등 부하 분산을 위해서 라운드로빈, 가중치 기반 라운드로빈, 최소 접속, 가중치 기반 최소 접속 스케쥴링의 기법이 연구되어 활용되고 있다. 하지만 기존의 스케쥴링 방식은 급증하는 서버의 부하를 균등 분산하지 못해 사용자의 요구를 충족시켜주지 못하며 부하 분산 능력이 저하되고 있어 새로운 스케쥴링 연구가 필요하다

본 논문에서는 부하 분산 서버가 RealServer와의 SNMP 메시지 교류를 통해 사용 가능한 메모리의 사용량, 1분 동안의 평균 부하량의 시스템 상태 정보를 획득한 후 Newton의 보간 다항식을 통해 부하 예측 값을 측정하고 이를 통해 균등 부하 분산을 하는 스케쥴링을 제안한다.

본 논문의 구성은 2장에서 관련 연구를 통하여 기존의 스케쥴링 방식을 알

아보고 3장에서는 본 논문이 제안한 시스템을 살펴본다. 4장에서 제안 시스템의 시뮬레이션과 5장에서 시뮬레이션 결과를 분석하고 6장의 결론에서 끝을 맺는다.

2. 시스템 구성 방법

2.1 LVS 구성 방안

가상 서버는 세가지 방식으로 사용할 수 있다. 부하 분산서버에 함께 존재하는 IP 부하분산에는 패킷 포워딩 방법을 이용한 세가지 기법이 있다. NAT, IP 터널링, 다이렉트 라우팅의 구성 방안이 그것이며 작동 원리 및 설정방법은 다음과 같다.

I. NAT 이용한 가상서버

NAT는 Network address translation(네트웍 주소 변환)으로 특정한 IP 주소를 한 그룹에서 다른 그룹으로 매핑하는 기능이다. 주소를 N-to-N 형태로 매핑하는 경우를 정적 NAT라 하고 M-to-N(M>N)를 동적 NAT라고 한다. 네트웍 주소 포트 변환은 기본 NAT의 확장기능으로 여러 가지 네트웍 주소와 TCP/UDP 포트를 단일의 네트웍 주소와 TCP/UDP 포트로 변환한다. N-to-1 매핑이라고 하며 리눅스에서 IP 마스커레이딩도 이러한 방식을 이용한다

TCP/IP 프로토콜을 지원하는 어떠한 운영체제에서로 실제 서버로 사용할 수 있다. 실제 서버는 사설 IP를 써도 되며 오직 부하분산 서버만 실제 IP를 사용하면 된다.

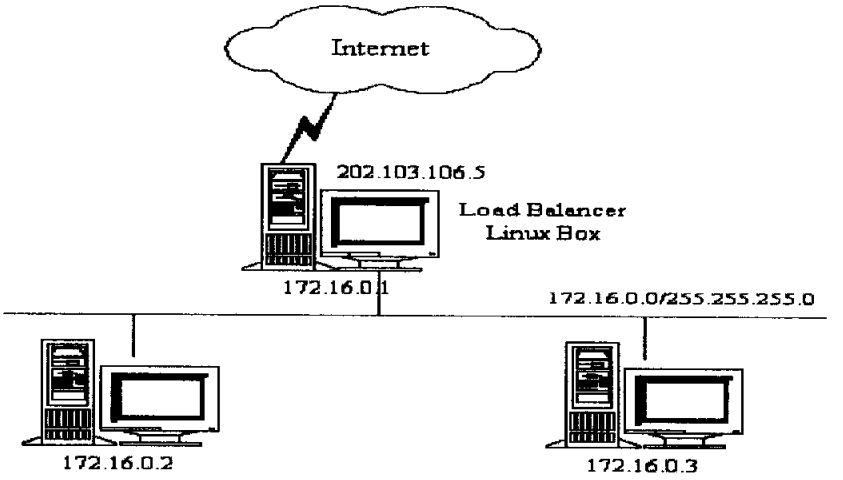

그림 2-1 NAT를 이용한 가상서버

단점은 확장성에 제한이 있다는 것이다. 일반적인 PC서버 기준으로 했을 때 서버노드가 20개 이상으로 증가할 경우 부하분산서버에서 병목이 생길 수 있다. 왜냐하면 패킷이 들어오고 나갈때마다 부하분산서버에서 패킷을 변경해야 하기 때문이다. 다음과 같이 가정해보자. TCP 패킷의 평균 길이가 536Bytes이다. 패킷 변경에 의한 지연시간은 60us(펜티엄 프로세서 기준이며 이보다 더 뛰어난 프로세서를 쓰면 지연시간이 감소될 것이다)이고 부하분한서버의 최대 처리량이 8.93MBytes/s이다. 실제 서버의 평균 처리량이 400Kbytes/s 일 때 부하분산서버는 22개의 실제 서버를 스케쥴링할 수 있다.

NAT를 이용한 가상서버에서 아주 많은 서버를 처리해야 할 경우가 있다.부하분산서버 자체가 전체 시스템에서 병목현상을 보이겠지만 이를 해결할 두가지 방법이 있다. 한가지는 혼합적인 방식을 사용하는 것이며 또 하나는 IP 터널링을 이용하거나 다이렉트 라우팅을 이용하는 것이다. DNS를 이용하는 경우,

서버클러스터를 구성하는 다수의 부하분한서버를 도입하고 그 부하분산서버를 Round-Round DNS로 이용해 묶는 것이다. 확장성을 위해 VS-터널닝이나 VS-다이렉트라우팅을 사용할 수 있으며 복합적인 부하분산방법을 생각할 수 있다. 전면에는 터널링이나 다이렉트 라우팅을 하는 부하분산서버를 구성하고 그 다음으로 NAT를 이용해 부하분산서버를 구성할 수 있다.

II. IP 터널링 이용한 가상 서버

IP 터널링 (IP encapsulation)은 IP 데이터그램안에 IP 데이터그램을 넣는 기술로서, 어떤 IP 주소를 향하는 데이터그램을 감싸 다른 IP 주소로 재지향할 수 있다. IP encapsulation은 현재 엑스트라넷, 모빌-IP, IP-멀티캐스트, tunnled 호스트나 네트웍 등에 일반적으로 사용되고 있다.

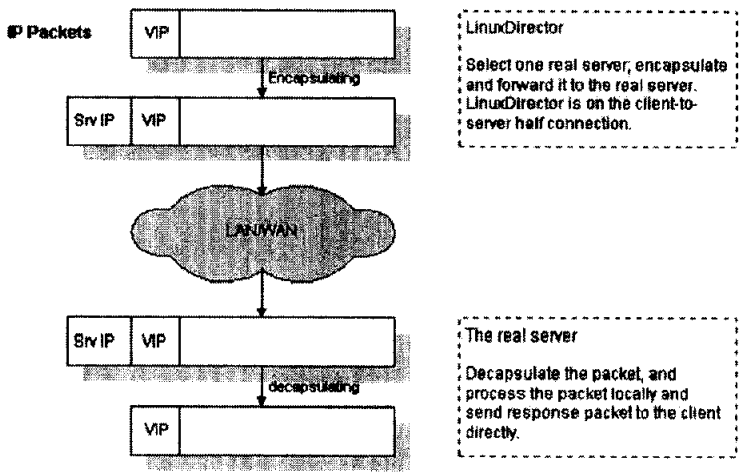

그림 2-2 IP 터널링을 이용한 가상서버

NAT를 이용할 경우 패킷이 들어오고 나갈때마다 부하 분산서버를 거쳐야한다. 네트웍 인터페이스의 처리량은 제한되어있기 때문에 서버 노드가 20대이상 늘어날경우 부하 분산서버가 병목현상의 새로운 주범이 될 수 있다. 일반적으로 웹 서비스같은 인터넷 서비스는 들어오는 요청는 작아도 그에 대한 응답은 자료양이 큰 경우가 대부분이다.

IP터널링을 이용하는 경우, 부하 분산서버는 단지 들어오는 요청에 대하여 각기 다른 실제 서버로 할당만을 할 뿐이고 실제 서버가 사용자에게 직접 응답을 보낸다. 그래서 부하 분산서버에서 더 많은 요청를 처리할 수 있으며 100개 이상의 실제 서버를 다를 수 있다. 또한 부하 분산서버 자체가 시스템의 병목지점이 되는 현상도 없앨 수 있다. 이렇게 IP 터널링을 이용하면 부하 분산서버에서 사용할 수 있는 서버 노드의 수를 크게 늘릴 수 있다. 설사 부하 분산서버가 100Mbps full-duplex 네트웍 어댑터를 가졌어도 가상 서버의 최대 처리량은 1Gbps에 달할 수 있다.

IP 터널링의 이러한 특성을 이용하면 아주 높은 성능의 가상 서버를 구축할 수 있으며, 특히 가상 프록시 서버에 적합하다. 프록시 서버에서 요청을 받는 경우, 인터넷에 직접 연결하여 개체를 가져오고 그것을 바로 사용자에게 보내줄 수 있다.

Ⅲ. 다이렉트 라우팅 이용한 가상 서버

RealServer와 부하 분산서버에서 가상 IP 주소를 공유한다. 부하 분산서버에는 마찬가지로 가상 IP 주소를 설정한 인터페이스가 있어야하며 이 인터페이스를 이용, 요청 패킷을 받아들이고 선택한 서버에 직접 라우팅 할 수 있다. 모든 RealServer는 가상 IP주소로 설정한 non-arp alias 인터페이스가 있거나 가상 IP 주소로 향하는 패킷을 지역 소켓으로 재지향한다. 그래서 RealServer에서 패킷을 지역적으로 처리할 수 있다. 부하 분산서버와 RealServer는 허브/스위치를 이용 물리적으로 링크된 그들만의 인터페이스를 가지고 하나 가지고 있어야한다.

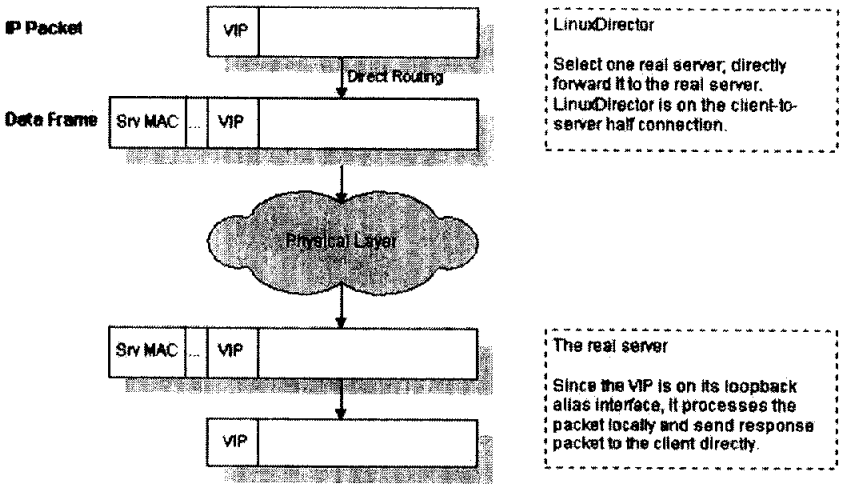

그림 2-3 다이렉트 라우팅을 이용한 가상서버

IP터널링을 이용한 가상서버와 마찬가지로, 리눅스다이렉터는 다이렉트 라우팅을 이용해 가상 서버로 가는 연결만 처리를 한다. 그리고 분리되어있는 네

트웍 라우터를 통해 사용자에게 응답 패킷을 보낸다. 이런 방법을 이용해 가상 서버의 확장성을 매우 높일 수 있다.

IP 터널링 방식에 비해 터널링에서 생기는 과부하가 없다. (실제로 이러한 과부하도 대부분 아주 작다) 그러나 부하분산서버에서 하나의 인터페이스가 더 필요하고 실제 서버의 인터페이스는 물리적인 세그멘트안에 있어야한다.

2.2 스케쥴링 필요성

LVS 시스템 구성에서 일반적으로 과부하를 극복하기 위해서 두 가지 방법이 사용된다.

Ⅰ. 고가의 고 성능의 서버 구성

실제적으로 작업을 수행하는 RealServer를 고가의 장비를 사용하여 시스템의 성능을 하드웨어적으로 향상시켜 수행력을 높이는 방법이다. 고 성능의 RealServer를 통하여 과부하를 극복하는 방법으로 이 구성 방식은 구성 비용이 높고 시스템의 성능에 의존하는 극복 방법은 한계가 있다.

Ⅱ. 저가의 다중 서버 구성

RealServer의 수를 고 성능의 서버를 사용할 때보다 많이 늘려서 저성능의 시스템으로 부하를 분산시켜 과부하를 극복하려는 방식의 구성이다.

고가의 장비로 구성하거나 저가의 장비로 구성된 LVS 시스템을 사용할 경우에 부하 분산 서버가 각각의 RealServer로의 부하분산이 요구되며 부하 분산을 위한 스케쥴링 알고리즘이 필요하게 된다.

2.3 기존 스케쥴링

기존에 연구되어 LVS 시스템에서 사용되고 있는 부하 분산 스케쥴링 알고리즘은 라운드 로빈, 가중치 기반 라운드 로빈 스케쥴링, 최소 접속 스케쥴링, 가중치 기반 최소 접속 스케쥴링이 있다.

Ⅰ. 라운드 로빈 스케쥴링[2]

사용자의 요구를 차례대로 각 서버에 균등하게 분배하는 방식으로 클라이언트의 연결에 대해 각 RealServer로 순차적으로 부하 부산을 한다.

이 경우 RealServer의 연결 개수나 반응시간 등은 고려를 하지 않는다. 그렇지만 약간의 차이가 있다. 라운드 로빈 DNS는 단일한 도메인을 서로 다른 IP로 해석을 하지만, 스케쥴링의 기초는 호스트 기반이며 캐싱때문에 알고리즘을 효율적으로 사용하기 힘들다. 그래서 RealServer 사이에 동적인 부하 불균형이 심각해질 수 있다. 가상 서버의 스케쥴링 기초는 네트웍 기반이며 라운드 로빈 DNS 에 비해 훨씬 더 훌륭하다

Ⅱ. 가중치 기반 라운드 로빈 스케쥴링[2]

각 RealServer의 처리 능력에 따라 가중치를 부여하여 라운드 로빈 방식처럼 부하 분산을 수행한다.

각 서버에 가중치를 부여할 수 있으며, 여기서 지정한 정수값을 통해 처리 용량을 정한다. 기본 가중치는 1이다. 예를 들어 실제 서버가 A,B,C 이고 각각의 가중치가 4,3,2 일 경우 스케쥴링 순서는 ABCABCABA 가 된다.

가중치가 있는 라운드 로빈 스케쥴링을 사용하면 RealServer에서 네트웍 접속을 셀 필요가 없고 동적 스케쥴링 알고리즘보다 스케쥴링의 과부하가 적으므로 더 많은 RealServer를 운영할 수 있다. 그러나 요청에 대한 부하가 매우 많을 경우 RealServer 사이에 동적인 부하 불균형 상태가 생길 수 있다.

라운드 로빈 스케쥴링은 가중치기반 라운드 로빈 스케쥴링의 특별한 한 종류이며 모든 가중치가 동일한 경우이다. 가상 서버의 규칙을 변경하고 나서 스케쥴링 순서를 생성하는데는 거의 과부하가 걸리지 않으며 실제 스케쥴링에 어떠한 과부하도 추가하지 않는다. 그러므로 라운드 로빈 스케쥴링만 단독으로 실행하는 것은 불필요한 일이다.

Ⅲ. 최소 접속 스케쥴링[2]

사용자가 연결된 RealServer 중에서 가장 적은 숫자의 연결이 이루어진 RealServer로 부하를 분산 시키는 동적 스케쥴링 방식이다.

각 서버에서 동적으로 실제 접속한 숫자를 세어야하므로 동적인 스케쥴링 알고리즘중의 하나이다. 비슷한 성능의 서버로 구성된 가상 서버는 아주 큰 요

구가 한 서버로만 집중되지 않기 때문에, 접속부하가 매우 큰 경우에도 아주 효과적으로 분산을 한다.

가장 빠른 서버에서 더 많은 네트웍 접속을 처리할 수 있다. 그러므로 다양한 처리 용량을 지닌 서버로 구성했을 경우에도 훌륭하게 작동한다는 것을 한눈에 알 수 있을 것이다. 그렇지만 실제로는 TCP의 TIME_WAIT 상태때문에 아주 좋은 성능을 낼 수는 없다. TCP의 TIME_WAIT는 보통 2분이다. 그런데 접속자가 아주 많은 웹 사이트는 2분 동안에 몇 천 개의 접속을 처리해야 할 경우가 있다. 서버 A는 서버 B보다 처리용량이 두 배일 경우 서버 A는 수천 개의 요청을 처리하고 TCP의 TIME_WAIT 상황에 직면하게 된다. 그렇지만 서버 B는 몇 천 개의 요청이 처리되기만을 기다리게 된다. 그래서 최소 접속 스케쥴링을 이용할 경우 다양한 처리용량을 지난 서버로 구성되었을 경우 부하분산이 효율적으로 되지 못할 수 있는 것이다.

Ⅳ. 가중치 기반 최소 접속 스케쥴링[2]

최소 접속 스케쥴링 방식에 시스템의 처리 능력에 따른 가중치를 부여하여 부하 분산을 하는 동적 스케쥴링 방식이다.

가중치가 높은 서버에서 더 많은 요청을 받을 수 있다. 가상 서버의 관리자는 각각의 RealServer에 가중치를 부여할 수 있다. 가중치의 비율인 실제 접속자 수에 따라 네트웍 접속이 할당된다. 기본 가중치는 1이다.

가중치가 있는 최소 접속 스케쥴링은 다음과 같이 작동한다:

n개의 실제 서버가 있는 경우 각 서버 i는 가중치 Wi $(i=1,\ldots,n)$를 가진다고 가정하자. 서버 i의 활동 접속은 Ci $(i=1,\ldots,n)$이고 모든 접속은 Ci $(i=1,\ldots,n)$의 합이다. 서버 j로 가는 넷트웍 접속은 아래와 같다.

$$(Cj/ALL\ CONNECTIONS)/Wj = \min (Ci/ALL\ CONNECTIONS)/Wi \quad (i=1,..,n)$$

… 수식 2-1

이 비교에서 ALL CONNECTIONS는 상수이므로 Ci를 모든 접속으로 나눠줄 필요가 없다. 그러면 다음과 같이 최적화될 것이다.

$$Cj/Wj = \min Ci/Wi \quad (i=1,..,n)$$

… 수식 2-2

가중치가 있는 최소 접속 스케쥴링 알고리즘은 최소 접속 스케쥴링 알고리즘에 비해 부가적인 배분작업이 필요하다. 서버들이 같은 처리 용량을 가졌을때는 작업 할당의 간접 비용을 최소화하기위해 최소 접속 스케쥴링과 가중치가 있는 최소 접속 스케쥴링 알고리즘 둘 다 사용할 수 있다.

2.4 기존 스케줄링 문제점

기존의 스케줄링의 가장 큰 문제점은 CPU, 메모리 등의 자원의 불균형의 정도가 심하다는 것이다. 이는 순차적인 부하 할당, 접속자의 수에 근거한 할당, 사용자들의 시스템 사용 정도의 차이가 RealServer로 사용되는 시스템의 성능에 따라 각 RealServer의 자원 사용의 불균형을 초래하고 하나의 RealServer에 과부하가 계속 걸리게 될 경우 LVS 시스템의 성능에도 많은 지장을 주게 된다.

3. 제안 시스템

1, 2장에서 살펴 본 LVS 시스템의 성능을 높이고 효과적인 스케줄링을 위해서 다음과 같은 시스템을 제안한다. 제안 시스템은 기존의 LVS 시스템을 기본으로 하여 구성된다.

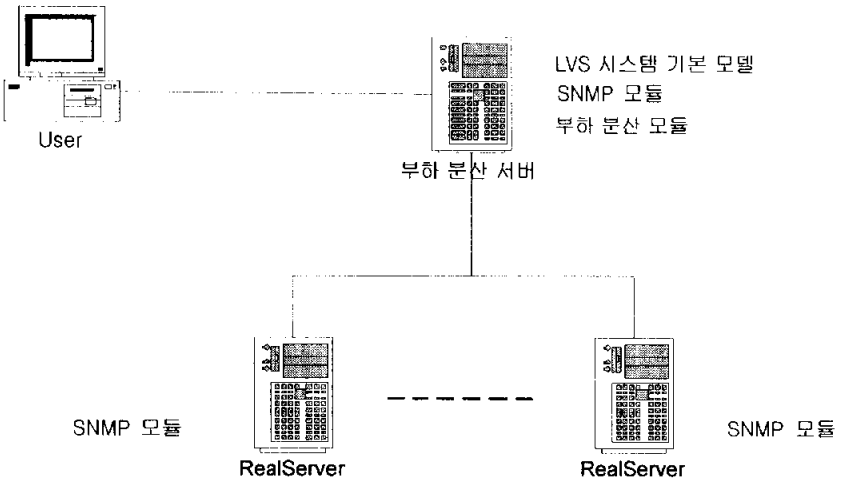

그림 3-1 제안 시스템 모델

3.1 SNMP 모듈

본 논문에서 제안한 스케줄링은 부하 분산 서버가 LVS 시스템에서 RealServer와의 SNMP 메시지 교환을 사용하여 획득한 시스템 정보를 계산하여 접속을 분산시키는 방식이다.

SNMP 프로토콜은 원래 TCP/IP 네트워크에서 라우터를 핸들링하기 위한 프로토콜로 디자인이 되었었다. 즉, SNMP는 프로토콜 독립적인 프로토콜이고 IP를 기반으로 인스톨된다. SNMP는 단일 프로토콜이 아니라 다음의 두가지 프로토콜과 더불어 패키지화 되는데 그것은 MIB(Management Information Base)와 SMI(Structure and identification of Management Protocol)이다. MIB는 네트워크의 상태를 저장하고 있는 일종의 데이터베이스를 의미하고 SMI는 서버와 네트워크용 디바이스 장치들의 통신 방법에 대한 규정을 의미한다.

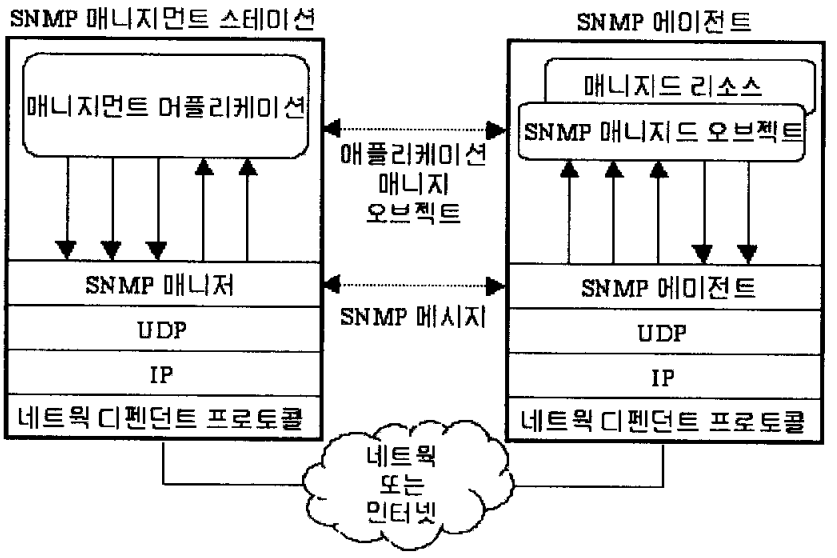

그림 3-2 SNMP 프로토콜 구조

SNMP의 구성

TCP/IP 네트워크를 관리하기 위한 4가지 모델은 관리 시스템, 관리대상 에이전트, 네트워크 관리 프로토콜, MIB로 구성된다.

Ⅰ. 관리 시스템은 네트워크 관리자에게 전 네트워크 상황을 볼 수 있는 인테페이스를 제공하며 관리 데이타의 분석, 장애관리 등의 기능수행을 위한 데이타베이스를 구축하고 있다.

Ⅱ. SNMP 에이전트는 피관리 대상장비, 즉 호스트, 라우터, 브릿지, 허브와 같은 네트워크 장비에 설치되어 관리 시스템의 요구에 따라 관리 정보를 송달하거나 관리 시스템의 액션 요구를 수행하며 문제 발생시 자동적으로 장애 상황을 관리 시스템에 통보한다.

Ⅲ. 네트워크 관리 프로토콜로서 현재 가장 널리 사용되는 SNMP는 OSI에서 정의된 CMIP와 공존하며 일정기간 계속 사용되리라 예상되고 있다.

Ⅳ. MIB(Management Information Base) 란 TCP/IP를 기초로한 관리 모델에서 각 피관리 대상장비의 관리 되어질 요소들에 대한 정보를 포함하고 있는 데이타베이스이다. 이때 관리되어지고 있는 각 정보들을 오브젝트라고 하며 MIB은 이러한 오브젝트들의 계층적 트리구조로 이루어져 있다. MIB은 SMI 규칙에 따라 5가지 기능분야로 분류된다.

구성관리 (Configuration Management)

네트웍상의 장비와 전반적인 물리구조를 Maping 하는 기능을 말한다.

성능관리 (Performance Management)

가용성, 응답시간, 사용량, 에러량, 처리속도 등 성능 분석에 필요한 통계 데 이타를 제공하는 기능

고장관리 (Fault Management)

문제의 검색, 추출 및 해결을 제공하는 기능

보안관리 (Security Management)

정보의 제어 및 보호 기능

계정관리 (Accounting)

각 노드별로 사용현황을 측정하는 기능

Simple Network Management Protocol

SNMP 프로토콜은 UDP (User Datagram Protocol)상에서 동작하는 비동기식 요청/응답 메시지 프로토콜로서 다음의 간단한 4가지 연산만 수행한다.

Get : 장비의 상태 및 가동시간 등의 관리 정보를 읽어 들인다. 특정 장비의 정보를 읽으려면 메시지의 송신자로서 관리자는 그 장비를 표시하는 작은 프로그램인 에이전트에 조회를 한다. 관리자는 MIB의 트리구조를 이용해 필요한 정보를 찾는 객체를 알아내고 응답을 해석한다.

Get Next : 정보가 계층적 구조를 가지므로 관리자가 장비에 조회를 해서 해

당 트리의 보다 하위층 정보를 얻도록한다.

Set : 장비의 MIB을 조작하여 장비를 제어한다. 관리자는 요청을 보내 다시 초기화 시키거나, 프로그램에 따라 스스로를 다시 재구성한다.

Trap : 관리자에게 보고하는 Treshold나 Event를 말한다. 장비 에이전트는 경고, 고장통지등 관리자가 미리 설정한 유형의 보고서를 생성한다.

SNMP 프로토콜은 일반적으로 시스템 모니터링을 하기 위해 많이 사용되고 있다. 본 논문에서 시스템 정보를 획득하기 위해서 SNMP 프로토콜을 사용하는 것은 다음과 같은 장점을 가지고 있기 때문이다.

SNMP 프로토콜 사용의 장점은 다음과 같다.

① SNMP는 관리 수행자 자체에 의해 실현되는 관리 기능의 수와 복잡성을 최소화하여 프로토콜을 지원하기 위한 개발 비용을 절감한다.

② 원격지에서 지원되는 관리 기능의 정도가 높아지면서 연결망 자원의 충분한 활용과 관리 기능의 단순성은 사용자들에게 편이를 제공한다.

③ 감시와 제어를 위한 기능적인 형태 활용이 확장 가능하고 구조가 특정 호스트 또는 게이트웨이의 구조와 메카니즘과는 독립적이다.

④ DES(Data Encryption Standard)로 전송되는 테이블 보안 기능을 위한 알고리즘과 MD5(Message Digest 5)를 사용하여 인증절차를 위한 보안 알고리즘을 사용한다.

다음 그림은 SNMP 메시지를 사용하여 시스템의 메모리 정보와 평균 부하량

정보를 획득하는 사용 예이다.

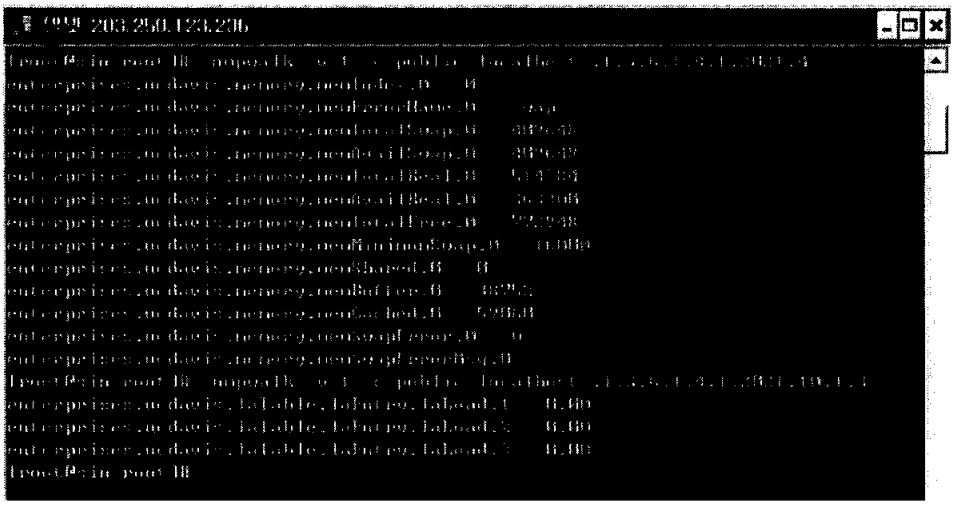

그림 3-3 SNMP 사용 예제

3.2 스케쥴링 순서도

제안 스케쥴링을 순서적으로 살펴보면 그림 3-1과 같이 나타낼 수가 있다

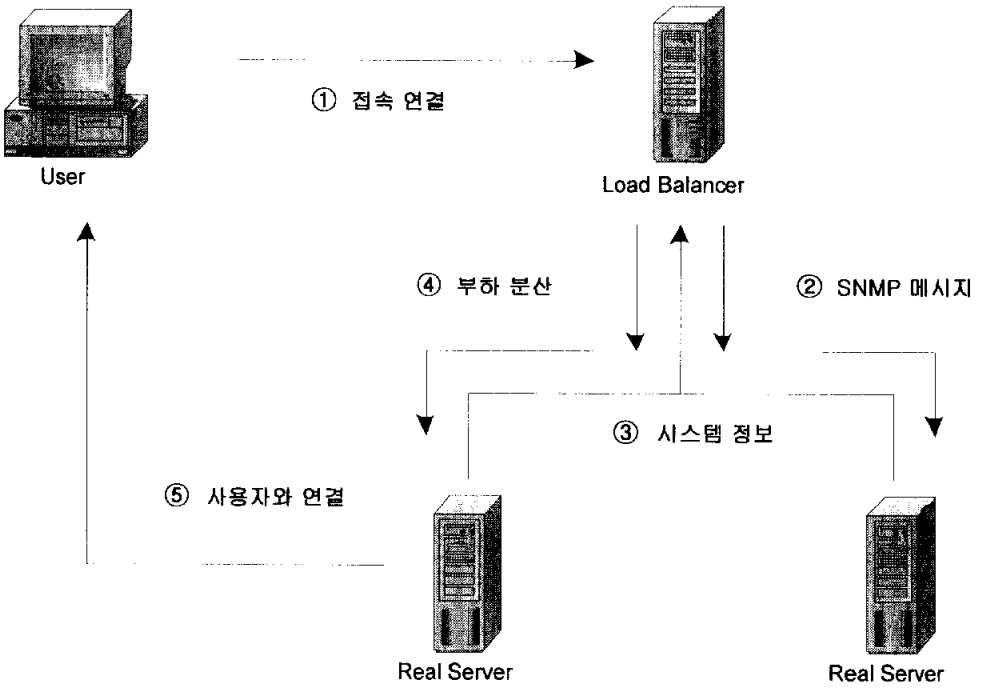

그림 3-4 스케쥴링 순서도

Ⅰ. 사용자의 접속 요청

일반적으로 LVS 시스템에서 사용자는 부하 분산 서버를 통해 접속 연결을 시도한다.

Ⅱ. 부하 예측값으로 부하 분산, SNMP 메시지 교환

미리 계산된 부하 예측값으로 부하 분산을 하고 오차를 줄이기 위해 주기적으로 SNMP 메시지 교환을 통해 예측값을 갱신하게 된다.

부하 분산을 위한 계산은 각 시스템의 사용 가능한 메모리 사용량, 1분 동안의 평균 부하량을 사용하여 Newton의 보간 다항식으로 측정한다.

Ⅲ. 시스템 정보 전송

부하 분산 서버의 SNMP 메시지가 시스템 정보를 요청하면 각 RealServer는 사용가능한 메모리 량과 1분 동안의 평균 부하량을 전송한다.

Ⅳ. 부하 분산

초기의 예측된 부하 값이 없거나 SNMP 메시지를 통하여 예측 부하값을 갱신하고 나면 부하 분산이 이루어진다.

Ⅴ. 사용자와의 연결

부하 분산 서버가 시스템 정보를 통하여 가중치를 계산 후, RealServer를 선택하면 이 RealServer는 접속을 요청한 사용자와 연결한다.

다음은 스케줄링 알고리즘을 나타낸 것이다.

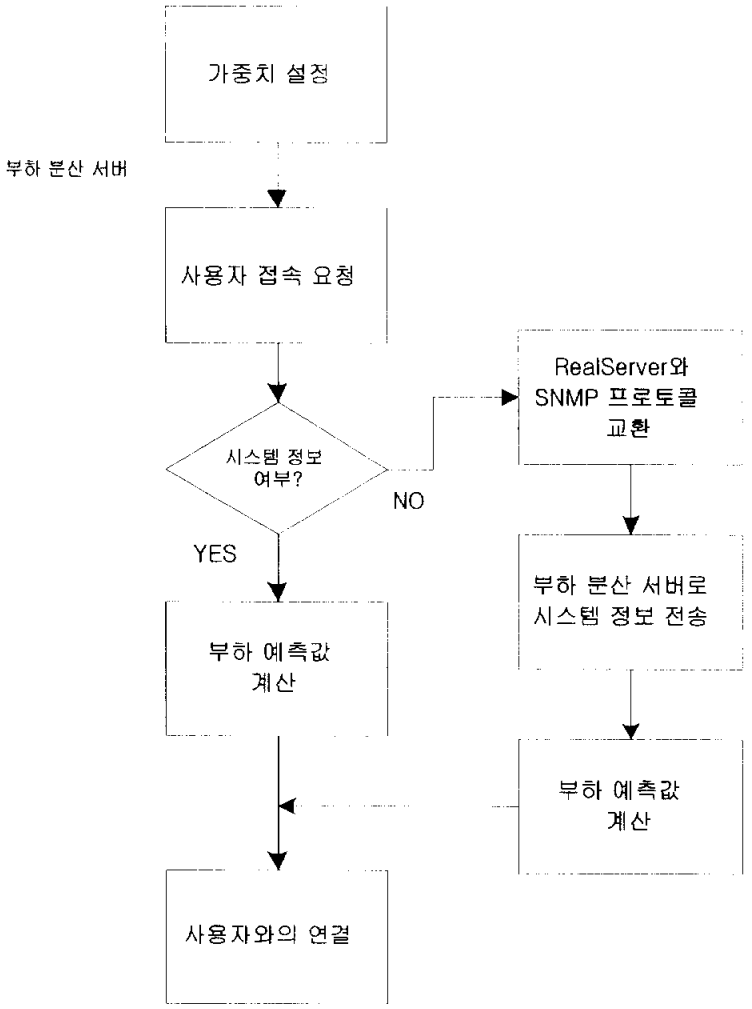

그림 3-5 알고리즘 순서도

3.3 부하 예측 계산

본 논문에서 제안하는 시스템의 부하 예측 계산 방법은 SNMP 메시지를 통해서 RealServer의 1분 동안의 평균 부하량과 사용 가능한 메모리의 사용량을 토대로 계산하여 예측한다.

부하 예측을 위한 계산을 위해서는 Newton의 보간 다항식을 사용한다. Newton의 보간 다항식은 주어진 데이터 점들의 정보로부터, 그 점들 사이의 정보를 유추하는 수치적 방법이다. 주어진 데이터 점들을 모두 지나는 식을 구하여 식으로부터 중간점들의 값을 계산할 수 있다. 보통 자료를 통하여 값을 예측할 경우에는 회귀분석 방법을 사용하지만 회귀분석법의 경우에 각각의 측정자료에 오차를 포함하고 있을 경우에 측정자료를 대표하는 함수를 구하여 효과적으로 사용할 수 있다. 반면에 보간법의 경우, 오차가 없다는 가정하에서 각 자료의 점을 지나는 보간 다항식을 구할 때 효과적이다. 여기서 측정된 시스템 부하값은 오차가 없다는 가정으로 보간법을 사용한다.

이러한 부하 예측값은 계속적인 시스템 정보를 통하여 얻은 값으로 부하 분산을 하는 것보다 신속하게 처리를 할 수가 있다. 또한, SNMP 프로토콜 교환에 할당되는시스템 자원 사용을 감소하는 효과를 볼 수 있다. 이는 동적 할당을 위한 시스템간의 메시지 지연 시에도 부하 분산이 가능한 장점을 가지게 된다.

다음의 표 3-1은 동적으로 획득한 실제 시스템 정보와 이를 바탕으로 부하를

예측하여 계산한 값을 비교하고 오차에 대해 알아본 것이다.

1분 동안의 평균부하량	사용 가능한 메모리량(Kbyte)	계산된 부하 예측값	오차
0.65	402884		
0.76	402868		
0.86	402864	402853.454545	-10.545455
0.95	402848	402840.363636	-7.636363

표 3-1 부하 예측값의 계산 예제

평균 부하량 0.65와 0.76에 따른 메모리량의 변화로 보간다항식에 적용하면

$$f(x) = 402978.545454 - 145.454545x$$

와 같이 함수를 구하여 부하 예측값을 계산할 수 있다.

3.4 가중치 결정

부하 예측 값을 계산한 후 부하 분산을 위해서는 각 RealServer의 시스템 성능에 따른 가중치가 필요하다. LVS 시스템에서 각 RealServer의 시스템 성능은 모두다 동일할 수도 있지만 모두 다를 수도 있다. 부하 예측을 통한 값을 획득한 후에 각 RealServer의 성능에 따른 부하 균등 분배를 위해서는 가중치를 설정하여 부하 처리 능력에 적합한 RealServer를 선택하여야 한다.

일반적으로 시스템 성능은 CPU 성능과 메모리량에 따라 결정이 된다. 하지만 웹 서비스의 경우 CPU 성능 보다는 메모리량에 의한 영향이 더 크다.[8] 본 논문에서는 각 RealServer의 시스템 가중치를 위해서 물리적 메모리량을 기준으로 하고 CPU 성능 또한 메모리량에 비해서 영향이 적지만 무시할 수 없기 때문에 일반적으로 사용하고 있는 인텔 CPU의 상대적 수치 표현 방식인 iCOMP 지수를 사용해서 결정한다.

다음 수식 3-7은 가중치 W를 구하는 식이다.

$$P_{(x)} = \frac{\text{사용 가능한 메모리 사용량}}{\text{물리적 총 메모리량}}$$

$$(P_{(x)}\min = 1, \cdots, P_{(x)}\max = n \text{라 둔다.}) \qquad \cdots \text{ 수}$$

$$W_{(x)} = \frac{P(x)}{RealServer \text{ 총 개수}}$$

(단, $P_{(s)} = P_{(t)}$ 면 $P_{(s)}iCOMP > P_{(t)}iCOMP$을 $P_{\max}$라 한다.)

식 3-7

그림 3-3은 인텔 CPU의 iCOMP 지수를 나타낸 것이다.

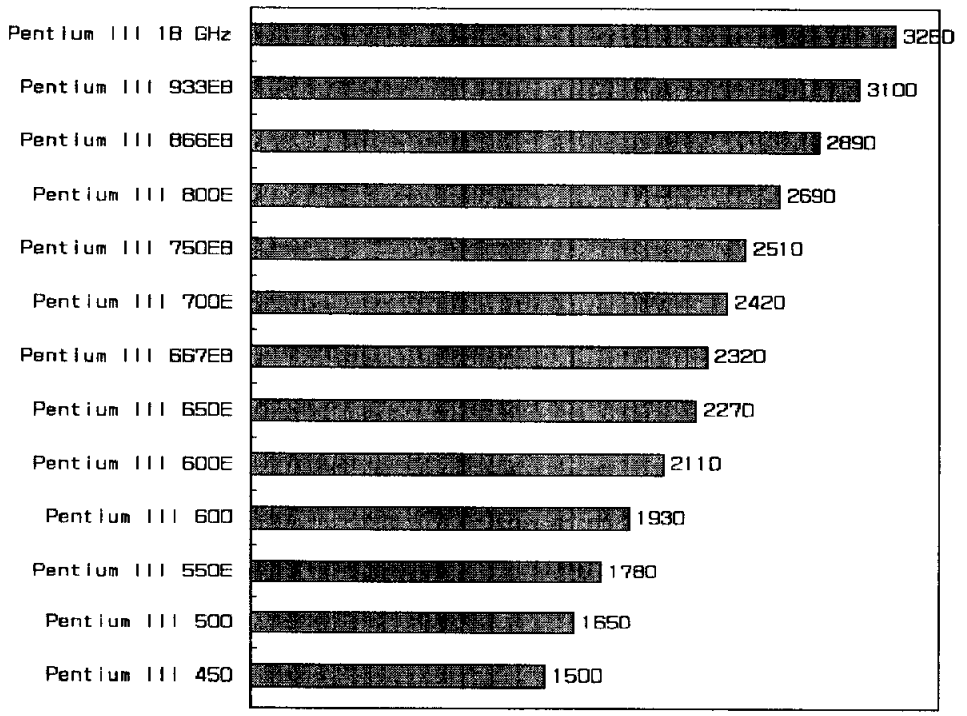

그림 3-6 인텔 CPU의 iCOMP 지수

4. 시뮬레이션

실험 환경은 셀러론 533 Mhz의 CPU와 512M 메모리로 구성된 리눅스 레드햇 7.3(커널 2.4X)의 시스템을 부하 분산 서버로 사용하였다. RealServer는 인텔 펜티엄 75 Mhz와 64M 메모리, 인텔 펜티엄Ⅱ 350 Mhz와 256M 메모리, 인텔 펜티엄Ⅱ 400 Mhz와 256M 메모리로 3대의 서버로 구성하였다.

다음 그림은 시뮬레이션 모델이다.

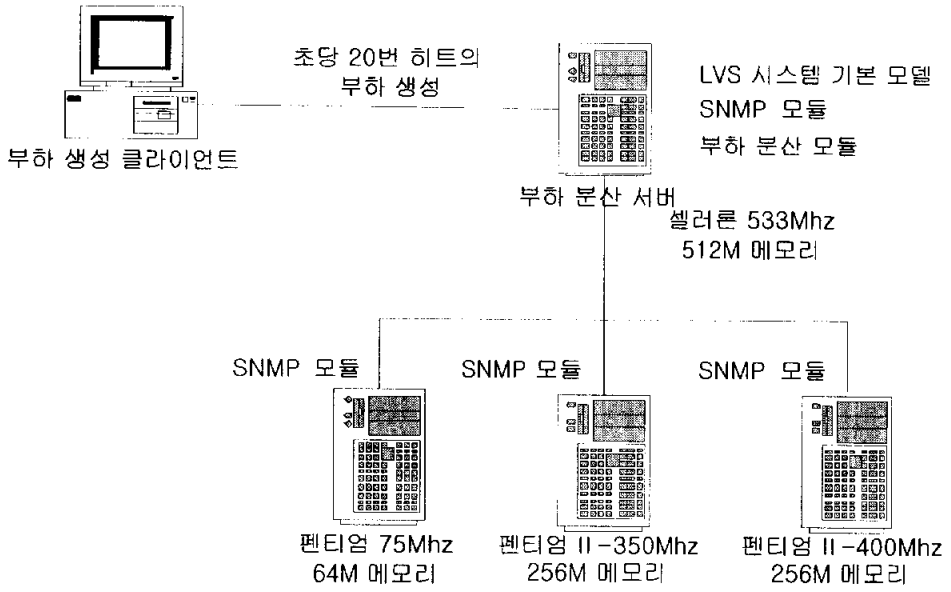

그림 4-1 시뮬레이션 모델

본 논문의 실험에서 CPU 성능은 고려하지 않고 메모리 사용량과 부하량의

정보에 초점을 맞추어 측정하였다. 이는 웹서버의 성능이 일반적으로 CPU의 성능보다는 메모리 사용량에 더 많은 영향을 받기 때문이다.[8]

부하 생성은 WebLoad라는 웹 부하 생성 응용 프로그램을 이용하여 10분 동안 초당 20번의 히트 수 만큼의 부하를 생성하여 1분 동안의 평균 부하량과 사용 가능한 메모리 사용량을 측정하였다.

WebLoad의 실행절차는 다음과 같다.

Ⅰ. 실제 클라이언트가 대상 사이트에 접속하여 실험에서 요청할 내용을 포함하는 Agenda file을 script 문을 이용하여 작성한다.

Ⅱ. 작성된 Agenda file의 내용을 그대로 반복하는 가상 클라이언트를 단위시간 당, 다수를 생성하여 LVS 클러스터 시스템에 접속시킨다.

Ⅲ. Agenda file의 선택된 컨텐츠들을 요청하여 대상 사이트에 부하를 부과하게 되고, 요청이 이루어지고 결과가 반환되면 기록하고, 가상 클라이언트는 소멸된다.

본 실험에서 사용된 Agenda file은 간단한 html 문서를 요청하는 script 파일로 구성이 되어 있다.

5. 시뮬레이션 결과 및 분석

다음 그림 5-1은 초당 20번의 히트 수의 시스템 부하를 라운드 로빈 방식으로 부하 분산한 것이다.

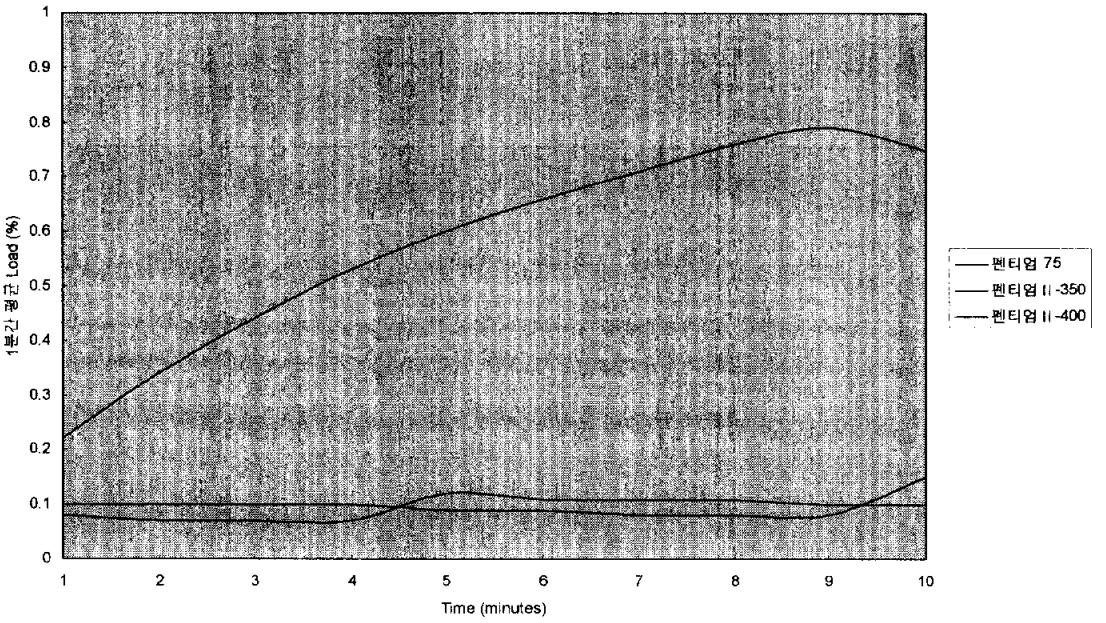

그림 5-1 초당 20번의 히트, 라운드 로빈

라운드 로빈 스케줄링에 의해 분산된 부하가 시간이 지남에 따라서 시스템 성능이 상대적으로 떨어지는 시스템에 부하가 적재되면서 각 시스템의 1분 동안의 평균 부하량 수치가 상당한 차이를 보인다. 이는 두 개의 시스템의 자원을 효율적으로 사용하지 못하고 있으며, 나머지 하나의 시스템은 과부하로 시스템 자원 사용에 있어서 불균형을 초래하고 있다.

다음 그림 5-2은 초당 20번의 히트 수의 시스템 부하를 제안한 방식으로 부하 분산한 것이다.

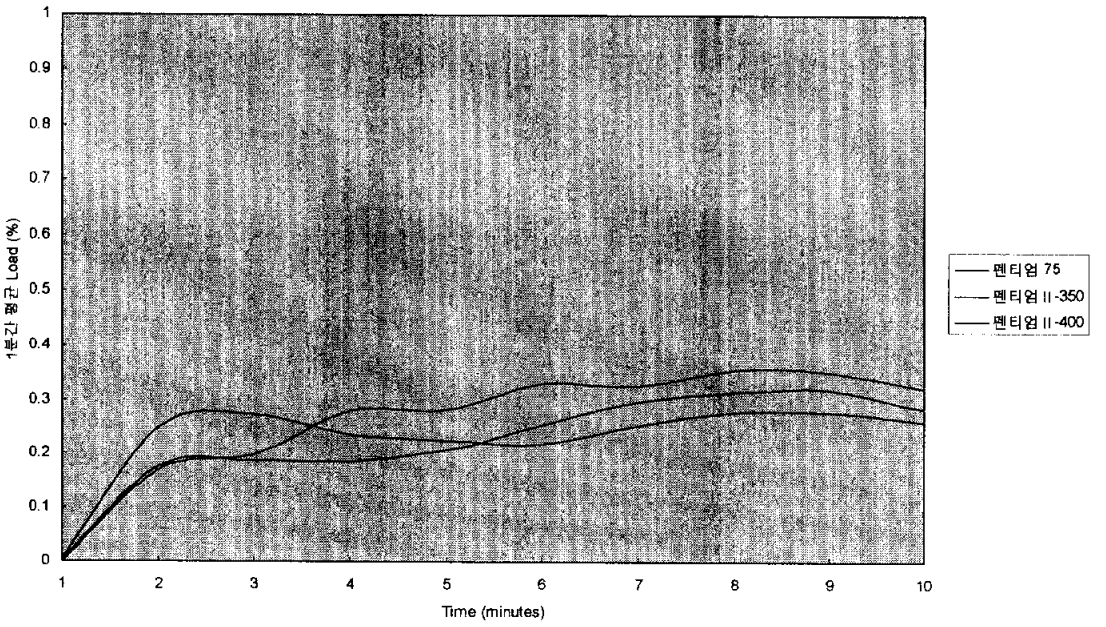

그림 5-2 초당 20번의 히트, 제안 방식

제안 시스템의 스케쥴링에 의해 분산된 부하는 라운드 로빈 방식과 비교해 볼 때 시스템의 성능에 따라 적절한 부하가 이루어 짐을 알 수 있다. 비슷한 수치의 1분 동안의 평균 부하량을 유지하면서 각 시스템의 자원 사용의 균형을 이루고 있다.

6. 결론

인터넷의 대중화 추세로 일상 생활에서 인터넷의 사용 비중이 점차 증가하고 있으며 이에 따라 인터넷의 부하에 미치는 영향도 급격히 증가하고 있다.

본 논문에서는 웹 부하 분산을 위해서 LVS 시스템에서 각 RealServer의 시스템 정보를 SNMP 프로토콜로 획득하고 이를 Newton의 보간 다항식을 사용하여 부하를 예측 계산하여 분산시킬 수 있는 부하 분산 스케쥴링을 제안하였다.

본 논문에서 제안한 방법으로 시뮬레이션 결과 실제 부하값에 해당하는 정확한 값을 유출해내지 못했지만 실제 부하값에 근접하는 부하 예측값을 측정할 수 있었다. 부하 예측값을 계산하는 측정식에서 오차의 차이를 줄인다면 시스템의 자원을 효율적으로 사용하면서 균등 부하 분산을 할 수 있을 것이다.

참고문헌

[1] 인터넷 통계 서비스,

http://www.i-biznet.co.kr/inet/inet-1999062819081.htm

[2] 리눅스 가상 서비스,

http://www.clunix.com/support/sw/about_lvs/index.html

[3] 김석찬, 이영, "동적 가중치에 기반을 둔 LVS 클러스터 시스템의 부하 분산에 관한 연구", 정보처리학회논문지A, 제8-A권, 제4호, 2001

[4] 박동국, 이용우, "리눅스 가상 서버의 구성방식과 스케쥴링 알고리즘의 성능평가", 한국정보과학회 봄 학술발표논문집, Vol.29, No.1, 2002

[5] 김성, 김경훈, 류재상, 남지승, "부하를 고려한 동적 가중치 기반 라운드로빈 스케쥴링 알고리즘", 한국정보처리학회 추계 학술발표논문집, 제8권, 제2호, 2001

[6] Wensong Zhang, "Linux Virtual Server Project",

http://www.linuxvirtualserver.org

[7] 장익진, 서경룡, "리눅스를 이용한 고 가용성 서버 시스템의 구축", 석사논문, 2001

[8] Mike Loukides, "System Performance Tuning", O'Reilly, 1990.

[9] 정낙수, 김정민, 이효진, 김성우, 황선태, "LVS 시스템 최적화"

[10] SNMP 프로토콜의 구성과 네트웍 관리,

http://my.netian.com/~yduvis/network/BASIC/com11.html

감사의 글

본 논문이 이루어지기까지 세심한 배려와 지도로 이끌어주시며 학문의 길을 가르쳐주신 서경룡 교수님께 한없는 감사를 드립니다. 그리고 본 논문을 심사해주시고 보완해주신 정목동 교수님, 권오흠 교수님께 진심으로 감사드립니다. 또한 학위과정 동안 많은 가르침으로 주셨던 학과의 여러 교수님들께도 감사드립니다.

길게 느껴졌지만 짧았던 대학원 생활동안 즐겁게 생활할 수 있도록 도와준 기호 형, 형상이 형, 현권이 형에게 감사드립니다. 항상 많은 조언을 아끼지 않았던 수진 선배, 봉기 선배에게 감사드립니다. 그리고 대학원 선후배인 창수 형, 희숙이 누나, 채주 형, 환조 형, 종욱이 형, 성주 형, 윤희 에게도 감사드립니다. 우리 연구실 식구들 익경이 형, 경민, 병길, 지산, 수진, 가끔 들러서 격려해준 현석 에게도 감사의 마음을 전합니다. 그 외 여러모로 도와주신 많은 분들에게 감사드립니다.

언제나 바로 설 수 있도록 도와주신 어머님과 동생과 기쁨을 함께 나누고 아울러 감사드립니다.

2003년 1월 이 석 주 드림