제 2 장 리눅스 가상 서버 시스템

본 절에서는 무정지, 무제한의 Web Server를 만들기 위한 Solution으로 제공되는 Linux Virtual Server(이하 LVS 또는 가상서버)에 대해서 소개한다. 리눅스 가상서버는 외부 사용자에게는 하나의 가상 IP만을 보여주면서, 내부적으로는 n개의 서버를 돌리는 것을 말한다. 가상서버는 고속의 LAN이나 지역적으로 분산된 WAN에 연결되어있는 여러 대의 실제서버(Real Server)와 실제서버의 전면에 있는 부하분산서버(Load Balancer)로 구성된다[4]. 부하분산서버는 사용자로부터 들어오는 요청에 대하여 정해진 스케줄링 알고리즘에 따라 각각의 서로 다른 실제서버로 요청을 분산시켜 서비스하도록 하여, 외부적으로는 클러스터로 구성된 병렬의 서비스를 단일 IP의 가상 서버로 인식하도록 한다. 또한, 클러스터 시스템은 여러 대의 실제서버들을 추가하거나 및 삭제하기가 용이하여 시스템 전체적인 확장성을 높일 수 있으며, 실제서버나 데몬에 문제가 생기면 이를 바로 인식하여 문제가 발생한 노드를 제거하고 그 즉시 시스템을 재구성하여 지속적인 서비스를 가능하게 하므로 높은 가용성과 신뢰성을 제공한다[2,4].

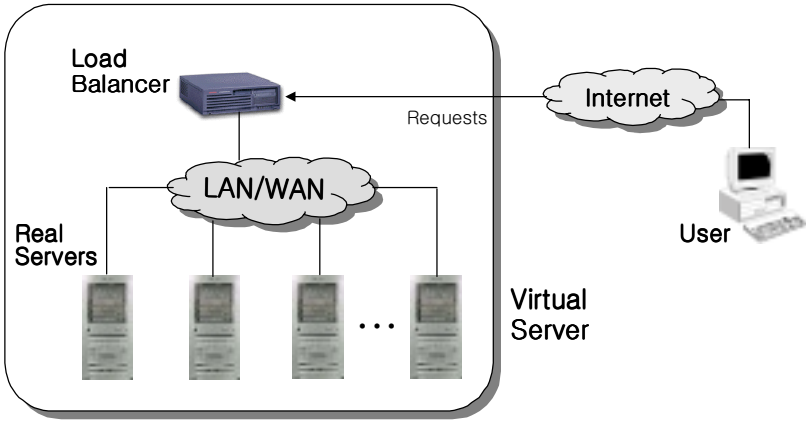

그림 2.1 가상서버 구성도
Fig. 2.1 Architecture of Virtual Server

가상서버를 구성하는 방법으로는 IP분산 방식에 따라 NAT (Network address translation)를 이용한 방법, IP 터널링을 이용한 방법, Direct Routing을 이용한 방법의 세 가지가 있다.

2.1 NAT를 이용한 가상서버

NAT(Network address translation - 네트웍 주소 변환) 방식은 TCP/IP 프로토콜을 지원하는 어떠한 운영체제 머신이라도 실제서버로 사용할 수 있다. 실제서버는 사설 IP를 써도 되며 오직 부하분산 서버만 실제 IP를 사용하면 된다. 따라서 실제 사용할 수 있는 IP주소가 부족할 경우, 사설 망에 있는 호스트에서 인터넷에 접속을 하거나 인터넷망에서 사설 망의 호스트에 접속하기 위해서 NAT기능을 이용하기도 한다.

NAT는 특정한 IP 주소를 한 그룹에서 다른 그룹으로 매핑하는 기능이다. 네트웍 주소 변환은 기본 NAT의 확장기능으로 여러 가지 네트웍 주소와 TCP/UDP 포트를 단일의 네트워크 주소와 TCP/UDP 포트로 변환한다. N-to-1 매핑이라고 하며 리눅스에서 IP 마스커레이딩도 이러한 방식을 이용하며 스티븐 클락(Steven Clarke)의 포트 포워딩 코드를 재사용하고 있다[4,5].

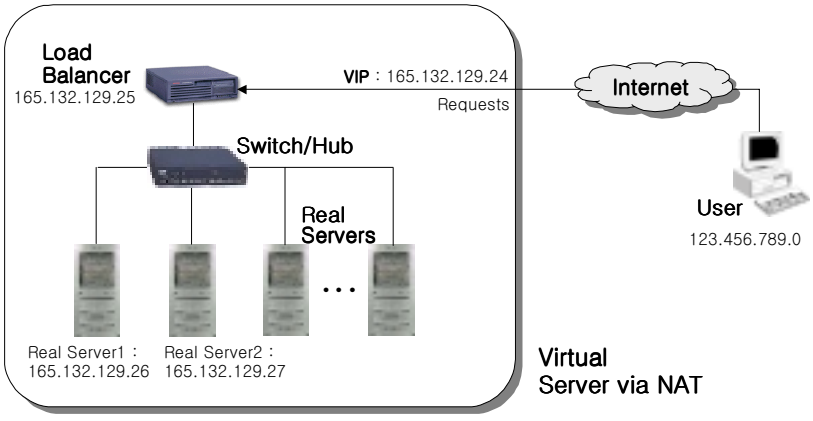

그림 2.2 NAT를 이용한 가상서버
Fig. 2.2 Virtual Server via NAT

사용자가 가상서버에서 제공하는 서비스를 요청할 때, 가상 IP 주소(부하분산서버의 외부 IP 주소)로 향하는 요구 패킷이 부하분산서버로 간다. 부하분산서버에서 패킷의 목적지 주소와 포트 번호를 검사한 후, 스케줄링 알고리즘에 따라 클러스터상의 실제서버를 선택한다. 그리고 접속이 이루어진 것을 기록하는 해쉬 테이블에 새로운 접속을 추가한 다음, 패킷의 목적지 주소와 포트를 선택한 실제서버에 맞게 변경하고 패킷을 해당 서버로 전송한다. 서비스 요청 패킷을 전달받은 실제서버는 요청 받은 서비스에 대한 응답을 다시 부하분산서버로 돌려보낸다. 실제서버로부터의 응답패킷을 받은 부하분산서버는 패킷의 출발 주소와 포트를 가상 서비스로 다시 변경하여 외부의 사용자에게 패킷을 전달한다. 이러한 네트웍 주소 변환 과정의 예를 표 1 에 나타내었다.

표 2.1 네트웍 주소 변환
Table 2.1 Network Address Translation

패킷의 이동	출발지 주소	도착지 주소
웹서비스 요청 패킷은 다음과 같은 출발지와 목적지 주소를 가지고 있다.		
사용자 → VIP(부하분산서버)	123.456.789.0	165.132.129.24
부하분산서버에서 실제 서버로 165.132.129.26를 선택하여 패킷을 재작성한다.		
부하분산서버 → 실제서버1	123.456.789.0	165.132.129.26
다음과 같이 부하분산 서버로 응답이 돌아온다.		
실제서버1 → 부하분산서버	165.132.129.26	123.456.789.0
패킷이 가상 서버 주소로 재 변경되어 사용자에게 다음과 같이 전송된다.		
VIP(부하분산서버) → 사용자	165.132.129.24	123.456.789.0

NAT를 이용한 가상서버는 TCP/IP를 지원하는 모든 머신을 사용할 수 있다는 장점이 있지만, 확장성에 제한이 있다는 치명적인 단점을 가진다. NAT 방식에서는 패킷이 들어오고 나갈 때마다 부하분산서버에서 패킷을 변경해야하기 때문에 . 일반적인 PC서버 기준으로 했을 때, 서버노드가 20개 이상으로 증가할 경우 부하분산서버에서 병목이 생길 수 있다. TCP 패킷의 평균 길이가 536Bytes라고 한다면, 패킷 변경에 의한 지연시간은 약 60us이고 부하분한서버의 1초당 최대 처리량은 8.93MBytes/s이다. 실제 서버의 평균 처리량이 400Kbytes/s 일 때 부하분산서

버는 약 22개의 실제 서버만을 스케줄링 할 수 있게 되는 것이다. 특히 웹서비스 같은 인터넷 서비스는 들어오는 요청은 작아도 그에 대한 응답은 자료의 크기가 큰 경우가 대부분이기 때문에 부하분산서버의 병목현상은 서비스의 지연을 가져오게 되어 전체 시스템의 성능저하를 초래하고 병목현상에 따른 각종 고장이 발생할 수 있기 때문에 이러한 병목현상을 극복하기 위해서 IP 터널링을 이용하거나 다이렉트 라우팅을 이용한다. 이러한 방식을 사용하면 전체 가상서버의 성능저하를 방지할 수 있으며 확장성 또한 크게 증가할 수 있다.

2.2 IP 터널링을 이용한 가상서버

IP 터널링을 이용하는 경우, 부하분산서버는 단지 들어오는 요청에 대하여 각기 다른 실제서버로 작업 할당만을 할뿐이고 실제서버가 사용자에게 직접 응답을 보낸다. 그래서 부하분산서버에서 더 많은 요청을 처리할 수 있으며 100개 이상의 실제 서버를 스케줄링 할 수 있으며 부하분산서버자체가 시스템의 병목지점이 되는 현상도 없앨 수 있다. 가령, 부하분산서버가 100Mbps full-duplex 네트웍 어댑터에 의해 동작하더라고 가상 서버의 최대 처리량은 1Gbps에 달할 수 있다[4]. IP 터널링의 이러한 특성을 이용하면 아주 높은 성능의 가상 서버를 구축할 수 있다.

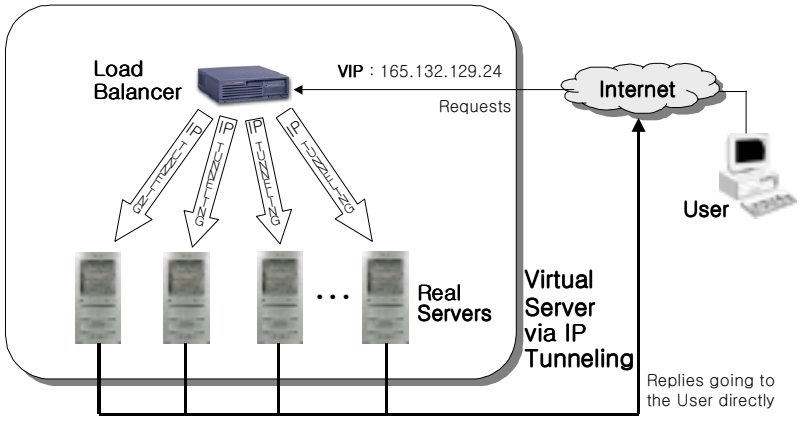

그림 2.3 IP 터널링을 이용한 가상서버
Fig. 2.3 Virtual Server via IP Tunneling

사용자가 가상서버에서 제공하는 서비스를 요청할 때, 가상 IP 주소(부하분산서버의 외부 IP 주소)로 향하는 요구 패킷이 부하분산서버로 간다. 부하분산서버에서 패킷의 목적지 주소와 포트 번호를 검사한 후, 스케줄링 알고리즘에 따라 클러스터상의 실제서버를 선택한다. 그리고 접속이 이루어진 것을 기록하는 해쉬 테이블에 새로운 접속을 추가한 다음, IP 데이터 그램 안에 패킷을 감싸 넣고(encapsulate) 실제 서버로 전송한다. 실제서버에서는 부하분산서버로부터 이 패킷을 받으면 패킷을 다시 풀고(decapsulate) 요청을 처리한 다음 최종적으로 실제서버의 라우팅 테이블에 따라 사용자에게 직접 결과를 돌려준다.

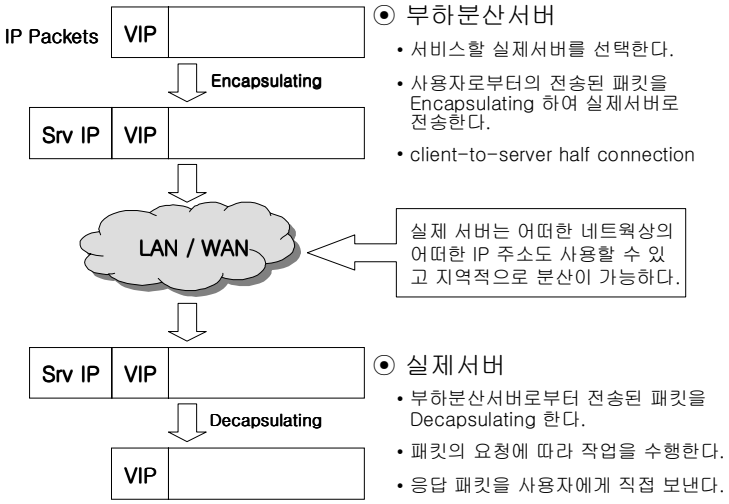

그림 2.4 IP 터널링의 패킷 작업 과정
Fig. 2.4 Packet Process via IP Tunneling

부하분산서버는 사용자(클라이언트)로부터 요청을 받을 때만 패킷을 변환하고 응답은 실제 서버에서 직접 사용자에게 전송된다. 이것을 Client-to-Server half connection 이라고 한다. NAT를 이용할 때는 들어오고 나가는 패킷이 모두 부하분산서버에서 변환되는데 이것은 Full connection이라고 할 수 있다. 실제 서버는 어떠한 네트웍의 어떠한 IP 주소도 사용할 수 있고 지역적으로 분산이 가능하여 부하분산서버에서 사용할 수 있는 서버 노드의 수를 크게 늘릴 수 있다.

2.3 다이렉트 라우팅을 이용한 가상서버

다이렉트 라우팅을 이용한 가상서버에서는 실제 서버와 부하분산 서버에서 가상 IP 주소를 공유한다. 부하분산서버에는 가상 IP 주소를 설정한 인터페이스가 있어야하며 이 인터페이스를 이용하여 요청 패킷을 받아들이고 선택한 실제서버에 직접 라우팅한다. 모든 실제 서버는 가상 IP주소로 설정한 non-arp alias 인터페이스가 있거나 가상 IP 주소로 향하는 패킷을 지역 소켓으로 재지향하여 패킷을 지역적으로 처리할 수 있다. 부하분산 서버와 실제 서버는 허브/스위치를 이용 물리적으로 링크된 그들만의 인터페이스를 가지고 있어야한다. 구조는 다음과 같다.

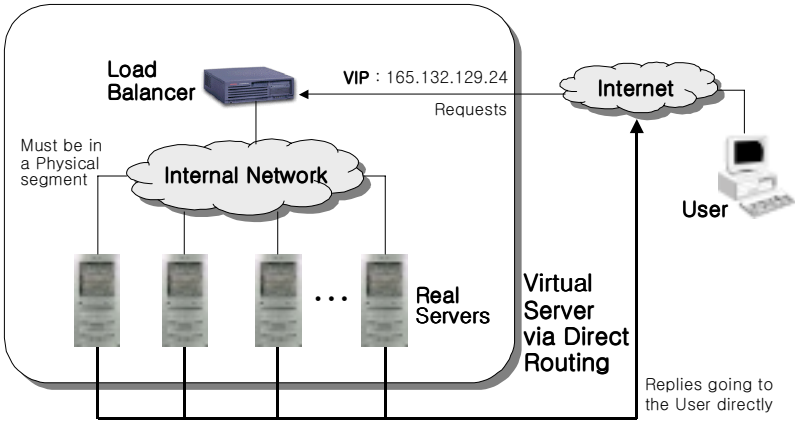

그림 2.5 다이렉트 라우팅을 이용한 가상서버
Fig. 2.5 Virtual Server via Direct Routing

IP 터널링을 이용한 가상서버와 마찬가지로, 부하분산서버는 다이렉트 라우팅을 이용하여 사용자 요청 패킷에 대한 실제서버와의 연결만을 처리하고, 실제서버가 서비스 요청을 처리하여 사용자에게 직접 응답을 보낸다. 다이렉트 라우팅을 이용한 가상서버에서의 부하분산서버는 단순히 데이터 프레임의 MAC 주소만 선택한 서버로 바꾸며 이를 다시 LAN에 재 전송한다. 이런 이유 때문에 부하분산서버와 각 서버가 단일한 물리적 세그먼트 안에서 연결되어야 한다. 이런 방법을 이용해 가상 서버의 확장성을 매우 높일 수 있으며 IP 터널링 방식에 비해 각 서버에서 발생할 수 있는 En(de)capsulating 과부하가 없다는 장점이 있다.

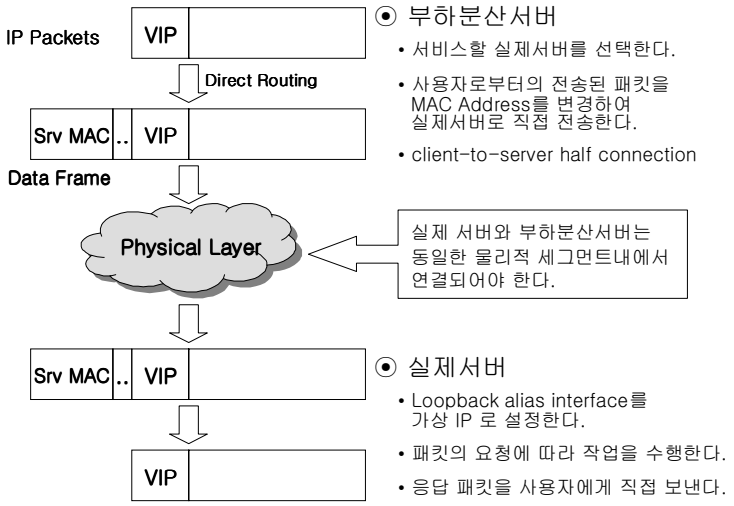

그림 2.6 다이렉트 라우팅의 패킷 작업 과정
Fig. 2.6 Packet Process via Direct Routing

표 2 에 NAT, IP 터널링, 다이렉트 라우팅 방법에 대해서 간단히 정리하였다. 참고로 본 논문에서는 다이렉트 라우팅을 이용하여 가상서버를 구축하였다.

표 2.2 NAT, IP 터널링, 다이렉트 라우팅에 대한 정리
Table 2.2 Comparison of NAT, IP Tunneling, Direct Routing

NAT		
IP 분산 방식	**장 점**	**단 점**
• 요청 패킷의 IP주소를 실제 서버의 IP주소로 직접 변경하여 실제서버로 전송한다.	• TCP/IP를 지원하는 모든 시스템에서 사용가능하다. • 실제서버의 외부적인 IP주소 없이도 사용 가능하다.	• 부하분산서버에서의 병목현상으로 20대 이상의 실제서버를 연결하기 어렵다.
IP Tunneling		
• 요청 패킷을 Encapsulating하고 실제서버의 주소를 추가하여 재전송한다.	• 부하분산서버의 병목현상이 적어 확장성이 뛰어나다. • LAN에 연결된 모든 서버가 실제서버의 역할을 할 수 있다.	• 모든 서버에서 IP 터널링 기능을 지원해야 한다. • En(de)capsulation과정에서 약간의 과부하가 발생할 수 있다.

Direct Routing		
IP 분산 방식	장 점	단 점
• 요청 패킷의 데이터 프레임의 MAC 주소만 실제서버로 바꾸어 재 전송한다.	• 부하분산서버의 병목현상이 적어 확장성이 뛰어나다. • IP 터널링에서 생기는 과부하가 없다.	• 부하분산서버와 실제 서버의 인터페이스는 동일한 물리적인 세그멘트안에 있어야한다.

2.4 가상서버 스케줄링 알고리즘

가상서버의 부하분산서버는 여러 가지 스케줄링 알고리즘을 사용하여 클러스터 내의 모든 실제서버들에게 부하를 분산한다. 여러 대의 실제서버가 클러스터를 이루어 하나의 고성능 가상서버를 이루기 위해서는 몇 대의 실제서버에만 부하가 편중되지 않고 모든 실제서버로 부하가 분산될 수 있도록 하는 부하분산 스케줄링 알고리즘이 가상서버의 전체적인 성능을 좌우할 수도 있다. 이번 절에서는 기본적인 4가지의 부하분산 스케줄링 알고리즘에 대해서 소개한다[2,4]. 본 논문에서는 라운드-로빈 방식의 스케줄링 기법을 사용하여 가상서버를 구축하였다.

2.4.1 Round-Robin 스케줄링

이 라운드-로빈 스케줄링 방식은 여러 가지 스케줄링 기법들 중에서도 가장 손쉽게 구현되고 자주 사용되는 스케줄링 기법으로, 클러스터내의 모든 실제서버들이 순서대로 부하를 분산 받는 방식을 말한다. 이 경우 실제서버의 기존 연결 개수나 시스템의 반응 시간 등은 고려되지 않는다. 라운드 로빈 스케줄링은 아래에서 설명될 가중치기반(Weighted) 라운드 로빈 스케줄링의 특별한 한 종류이며 모든 가중치가 동일한 경우이다.

2.4.2 Weighted Round-Robin 스케줄링

가중치기반(Weighted) 라운드 로빈 스케줄링은 실제 서버에 서로 다른 처리 용량을 지정할 수 있다. 즉, 각 서버에 가중치를 부여할 수 있으며, 여기서 지정한

가중치 정수 값을 통해 처리 용량을 정한다. 기본 가중치는 1이다. 예를 들어 실제 서버가 A, B, C 이고 각각의 가중치가 4,3,2 일 경우 스케줄링 순서는 ABCABCABA 가 된다.

가중치가 있는 라운드 로빈 스케줄링을 사용하면 실제 서버에서 네트웍 접속을 셀 필요가 없고 동적 스케줄링 알고리즘보다 스케줄링의 과부하가 적으므로 더 많은 실제 서버를 운영할 수 있다. 그러나 요청에 대한 부하가 매우 많을 경우 실제 서버사이에 동적인 부하 불균형 상태가 생길 수 있다.

2.4.3 Least-Connection 스케줄링

최소 접속(Least-Connection) 스케줄링은 각 실제서버의 상태를 파악하여 가장 접속이 적은 서버로 요청을 직접 연결하는 방식을 말한다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야하므로 동적인 스케줄링 알고리즘중의 하나이다. 비슷한 성능의 서버로 구성된 가상 서버에서는 매우 효과적인 분산알고리즘일 수 있다. 그러나 일반적으로 가장 빠른 서버에서 더 많은 네트웍 접속을 처리할 수 있기 때문에 다양한 처리 용량을 지닌 실제서버들로 가상서버를 구성했을 경우에는 TCP의 TIME_WAIT 상태 때문에 아주 좋은 성능을 낼 수 없을 수도 있다. 또한 연결의 수가 적더라도 그 몇 개의 연결에 의해 많은 서비스를 요청 받을 수도 있어서 오히려 부하 불균형이 심화될 수도 있으며, 또한 동적인 스케줄링 알고리즘이기 때문에 부하분산서버는 수시로 실제서버의 상태를 파악해야 하고 여기서 오는 시간적인 처리 지연 현상도 발생할 수 있다.

2.4.4 Weighted Least-Connection 스케줄링

가중치 기반 최소 접속(Weighted Least-Connection) 스케줄링은 최소 접속 스케줄링의 한 부분으로서 각각의 실제 서버에 성능 가중치를 부여할 수 있다. 언제라도 가중치가 높은 서버에서 더 많은 요청을 받을 수 있으며, 가상 서버의 관리자는 각각의 실제 서버에 가중치를 부여할 수 있다. 가중치의 비율인 실제 접속자 수에 따라 네트웍 접속이 할당된다. 기본 가중치는 1이며 가중치가 있는 최소 접

속 스케줄링 알고리즘은 최소 접속 스케줄링 알고리즘에 비해 부가적인 배분작업이 필요하다.

2.5 부하분산서버와 실제서버의 설정

가상서버의 부하분산서버가 여러 가지 스케줄링 알고리즘을 사용하여 클러스터내의 여러 실제서버들에게 부하를 분산하고 부하를 분산 받은 실제서버들이 실제로 서비스를 수행하기 위해서는 리눅스 커널이 클러스터 시스템을 지원할 수 있도록 컴파일 되어있어야 한다. 실제로 커널 소스를 다운 받아 클러스터 가상서버를 구동시킬 수 있는 기능들을 지원하도록 컴파일 하여 구축하였으나, 여기서 사용한 WOW Linux 6.1 plus (Kernel Version 2.2.14-5.0)은 기본적으로 클러스터 기능을 모듈화하여 컴파일 되어 있어서 추가적인 커널 컴파일 과정 없이도 가상서버를 구축할 수 있었다[6-8]. 따라서 가상서버를 구축하기 위한 첫 번째 전제조건인 클러스터링을 위한 커널 컴파일 과정에 대해서는 다루지 않도록 하겠다.

본 논문에서는 다이렉트 라우팅을 이용한 가상서버를 구축하였으며 부하분산서버의 스케줄링 알고리즘으로는 라운드 로빈 방식을 사용하였다. 가상서버 구축을 위해서 모든 서버에 클러스터링을 지원하는 커널을 설치한 후, Virtual Server patch for Linux 2.2.14 -Version 0.9.7- 을 Linux Virtual Server Project 사이트에서 Download 하여 설치하였다. 다음은 각각의 서버에서 자신의 역할을 수행하기 위해 실행되어야 할 스크립트 파일에 대하여 설명하도록 하겠다.

2.5.1 부하분산서버의 설정

부하분산서버로는 165.132.129.25의 IP 주소를 사용하였으며 외부에서 접속하게 되는 가상 IP는 165.132.129.24이다. 실제서버들로는 165.132.129.26, 165.132.129.27, 165.132.129.28을 사용하였다. 모든 IP들이 교내 DNS 서버에 Host-name이 등록되어 있으나 여기서는 IP 만을 이용해 설정한 예를 보도록 하겠다. 다음은 부하분산서버에서 실행된 부하분산을 위한 스크립트 파일이다[9].

```
ifconfig eth0:0 165.132.129.24 netmask 255.255.255.255 broadcast 165.132.129.24 up
route add -host 165.132.129.24 dev eth0:0
echo 1 > /proc/sys/net/ipv4/ip_forward
ipvsadm -A -t 165.132.129.24:80 -s rr
ipvsadm -a -t 165.132.129.24:80 -r 165.132.129.26 -g
ipvsadm -a -t 165.132.129.24:80 -r 165.132.129.27-g
ipvsadm -a -t 165.132.129.24:80 -r 165.132.129.28 -g
```

다음은 각각의 명령에 대한 간략한 설명이다.

- *ifconfig eth0:0 165.132.129.24 netmask 255.255.255.255 broadcast 165.132.129.24 up*

  eth0:0 이라는 것은 지금 현재 리눅스 머신에 붙어있는 네트워크 인터페이스 장치에 또다른 IP를 부여하겠다는 의미이다. 다시 말하면, IP 앨리어스를 사용하여 하나의 랜카드에 IP를 두 개 부여하겠다는 의미이다. eth0:1 , eth0:2 ... 이런식으로 여러개의 IP를 하나의 랜카드에 부여할 수도 있다. 따라서 이 명령은 이미 165.132.129.25 IP를 할당받은 부하분산서버의 이더넷 카드에 IP 앨리어스를 이용하여 165.132.129.24 라는 가상 IP를 eth0:0 에 부여하겠다는 명령이 될 것이다.

- *route add -host 165.132.129.24 dev eth0:0*

  가상 IP 165.132.129.24를 장비 eth0:0 의 기본 host 네트워크로 설정한다.

- *echo 1 > /proc/sys/net/ipv4/ip_forward*

  ip_forward 는 네트워크 서비스시 사용되는 변수중의 하나이다. 이 값을 1로 부여하면 ip_forward 기능을 사용할 수 있는 'enable' 상태가 된다.

- *ipvsadm -A -t 165.132.129.24:80 -s rr*

  ipvsadm 명령에서 -A 옵션은 라우팅 테이블에 public service address를 추가하는 것이다. service address는 일반적으로 IP address와 port number, protocol type 으로 정의된다. -t 라는 parameter는 TCP service를 이용하겠다는 것이다. -s 라는 parameter는 scheduling-method를 지정하는 것인데 여기서는 Round Robin 방식을 지정하였다.

  165.132.129.28:80 에서 port number를 80으로 설정하였는데, 이는 가상 IP에서 웹 서비스를 하는데 있어서 지정된 방식의 스케줄링 알고리즘 에 의해서 부하 분산을 수

행하겠다는 것이다.

또, 165.132.129.28:23 과 같이, port number가 23으로 설정하면, 요청되는 Telnet Service에 대해서 지정된 방식의 스케줄링 알고리즘에 의해서 부하 분산을 수행하겠다는 것을 의미한다.

- *ipvsadm -a -t 165.132.129.24:80 -r 165.132.129.26 -g*

  -a 옵션은 존재하는 165.132.129.24:80 라는 public service address에 대해서 redirected 받을 수 있는 서버들을 서로 연계시키겠다는 것을 의미한다. 뒤에 나오는 -r 옵션은 실제로 redirected 되어 동작할 실제서버를 지정하는 것으로 여기서 지정된 165.132.129.26 는 첫 번째 실제서버이다. -g 옵션은 packet-forwarding-method 중에서 gatewaying을 이용한 다이렉트 라우팅을 지정한 것이다.

  실제로 n 개의 Rear Server 가 존재할 경우 이 명령을 n 번 실행하면 된다.

2.5.2 실제서버들의 설정

실제서버들의 서비스 처리를 위한 스크립트는 모두 동일하다. 여기서는 다이렉트 라우팅을 이용하여 서비스를 분산시키기 때문에 각 실제서버들은 자신의 IP를 가상 IP 로 바꾸어 서비스를 제공한다는 것을 확인할 수 있다. 다음은 실제서버에서 실행된 스크립트 파일이다.

```
ifconfig lo:0 165.132.129.24 netmask 255.255.255.255 broadcast 165.132.129.24 up
route add -host 165.132.129.24 dev lo:0
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
```

다음은 각각의 명령에 대한 간략한 설명이다.

- *ifconfig lo:0 165.132.129.24 netmask 255.255.255.255 broadcast 165.132.129.24 up*

  route add -host 165.132.129.24 dev lo:0

  lo은 127.0.0.1과 같은 로컬 루프백 장치를 의미하는데, 여기서는 lo:0 이라는 로컬 루프백 장치에 IP 엘리어스를 이용하여 자기자신의 IP 를 가상 IP로 대체 하겠다는 것을 의미한다. 다시 말하면, 다이렉트 라우팅의 경우 각각의 실제서버가 사용자에게 직접 액세스하여 응답하게 되므로, 각각의 실제서버가 사용자에게 응답할 때에는 실

제서버가 마치 애초에 사용자가 접속한 가상 IP인 것처럼 보여지기 위한 명령이라고 할 수 있다. 이 명령은 IP 터널링 기법에는 사용되지 않음을 유념하자.

- *echo 1 > /proc/sys/net/ipv4/conf/all/hidden*
  echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
  : 가상서버 전체에서 하나의 가상 IP를 부하분산서버와 여러 대의 실제서버가 함께 공유하기 때문에 발생하는 ARP 문제로부터 실제서버의 네트워크 인터페이스를 숨기기 위한 명령이다.

지금까지 가상서버 구축에 대해서 간단히 살펴보았다. 이제 가상서버를 이루는 클러스터 서버가 아닌 다른 임의의 컴퓨터에서 가상 IP로 접속해보면 클러스터링이 잘 수행되고 있는지 확인할 수 있을 것이다. 웹 서비스의 테스트는 클라이언트 측의 캐시기능 때문에 클러스터링이 잘 수행되는지 의문이 갈 수도 있으나, 이런 경우 여러 대의 컴퓨터에서 가상 IP로 웹서비스를 요청하면 클러스터링되는 것을 확인할 수 있을 것이다. 보다 확실한 클러스터링을 확인하고자 한다면 포트번호를 23으로 설정하여 텔넷으로 접속하여보면 세대의 실제서버들로 순차적으로 접속되는 것을 확인할 수 있을 것이다.