3. 부하분산 – Load Balancing 클러스터 시스템 구축

3.1 리눅스 부하 분산 클러스터 개론

3.1.1 클러스터 소개

인터넷의 발달과 더불어 생활의 정보 전달 매체로 웹이란 수단을 선택하기 시작한지 10 년에 다 되어가면서 매년 인터넷의 사용 증가는 100%를 넘고 있다. 뿐만 아니라 근래에 들어 인터넷 초기 시절의 단순한 정보 공유 매개체로서의 역할에서 사업상의 중요한 업무 시스템 구성의 중요한 요소까지 차지하게 되었다. 인터넷의 발달과 더불어 하드웨어와 소프트웨어 그리고 네트워크의 발달 속도 역시 빠른 추세로 증가 하고 있지만 기하급수적으로 늘어난 서비스 사용량을 처리 하기에 보편적인 시스템의 성능과 시스템 환경의 부족함이 많이 존재하고 있다.

이에 보다 높은 가용성 (Availability) 과 확장성(scalable)을 가진 서버의 요구가 절실히 높아지고 있다. 이에 고성능 SMP 서버를 이용하는 경우도 있지만 비 경제적인 가격 문제와 실제 시스템 병목 원인이 꼭 서버의 성능에만 있는 것이 아니기 때문에 고성능 SMP 서버만이 진정한 해결 방법은 아니다.

빠른 네트워크의 발달과 함께 분산된 서버의 자원을 하나의 작업에 이용하는 클러스터링 기술이 발달하게 되고 인터넷 서비스 역시 클러스터링 기술이 부각되기 시작하였다.

클러스터란 여러 대의 컴퓨터를 네트워크를 통해 연결하여 하나의 단일 컴퓨터처럼 작동하게 하는 기술을 말한다. 클러스터는 사용 목적에 따라 구현 방법 및 소프트웨어가 달라지며, 클러스터의 기능과 운영이 달라진다. 따라서 클러스터는 그 사용 목적에 따라 베어울프와 같은 과학 계산용 클러스터와 인터넷 서비스에서 사용되는 부하분산 클러스터로 나눌 수 있다.

이런 클러스터 시스템의 장점으로 추가 확장성과 높은 가용성 그리고 비용-효율성을 두루 갖추고 있어 인터넷 서비스에 사용되는 시스템의 요구 조건을 충족 시켜 줄 수 있다.

부하 분산 클러스터는 이전부터 L4 Switch 란 하드웨어를 이용하여 시스템 구축을 해 왔지만 워낙 고가의 시스템 이여서 일반 사이트에서는 적용이 힘든 방식으로 알려져 있었다. 하지만 Linux 의 커널에서 L4 Switch 의 기능과 동일하고 더 월등한 성능의 LVS(Linux virtual server) 기능을 지원함에

따라 일반 사이트에서도 손쉽게 클러스터 기술을 이용하여 고성능의 서버를 이용할 수 있게 되었다. 리눅스의 LVS 커널은 커널 2.0.x 때부터 지원 되었고, 사용 된지도 7~8 년에 이르고 있다.

이에 지속적인 안정성과 성능이 개선 되어져 리눅스의 수많은 기능 중 가장 선호하는 기능으로 손꼽고 있다.

이런 리눅스 부하 분산 클러스터도 여러 가지 방식이 있고 리눅스 LVS 커널과 다른 리눅스 응용 프로그램을 조합하여 고 성능 뿐만 아니라 고 가용성의 기능까지 탑재하게 된다.

본문 에서는 네트워크 구성에 따른 부하 분산 방식과 작업 활당 방식 그리고 고 가용성을 구현하는 방법에 대해 자세히 알아보도록 하겠다.

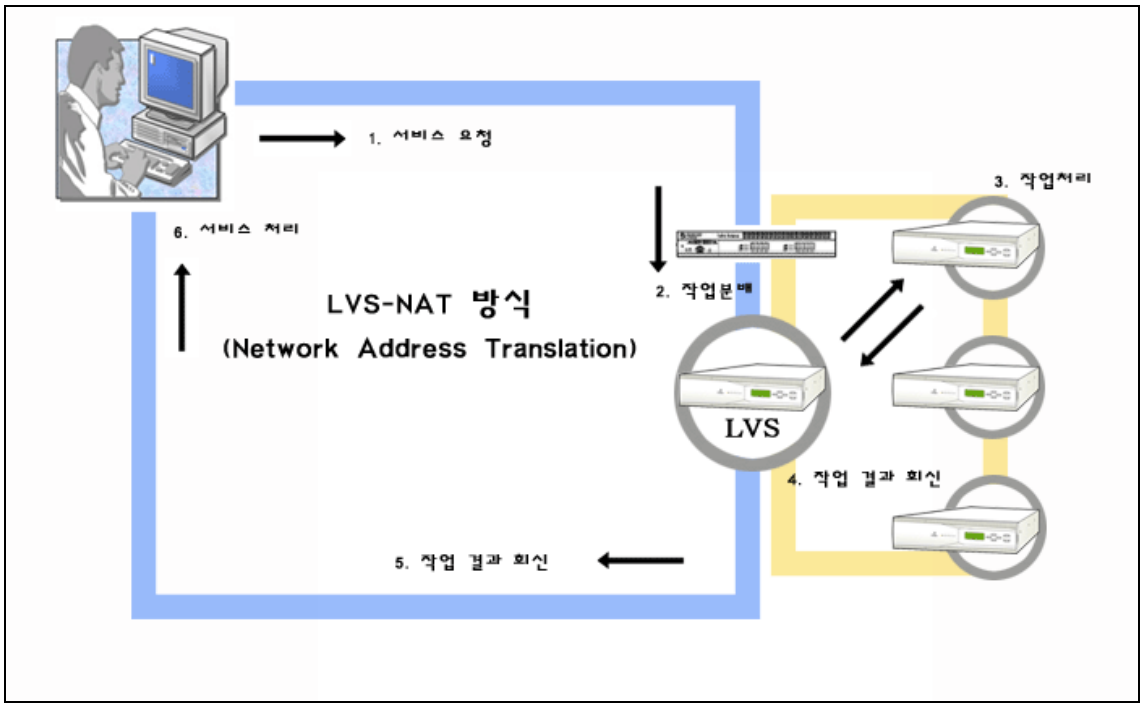
3.1.2 네트워크 구성에 따른 부하 분산 클러스터 종류

부하 분산 클러스터를 생각하면 대부분 Load Balancer 을 앞 단에 두고 backend 의 Real Server 에 작업을 분산하는 서버 측면의 부하 분산 방식을 알고 있다. 하지만 클라이언트에 부하분산 모듈을 넣어 서버의 상황을 클라이언트가 서버로부터 얻어내어 클라이언트가 직접 서버로 분산해서 들어가는 클라이언트 측면의 부하분산 방식도 있다. (예: Clunix 의 LB Less , 버클리의 스마트 클라이언트, 보스턴 대학의 클라이언트) 서버 측면의 부하분산 방식으로는 DNS 의 호스트를 이용한 Round-Robin DNS 방법과 Reverse-Proxy 와 같은 응용프로그램 계층의 부하분산 그리고 L4 Switch 와 같은 IP 계층의 부하분산 방식이 있다. 리눅스 가상 서버 역시 IP 계층의 부하 분산 방식에 속한다.

리눅스 가상 서버에서도 네트워크 구성에 따라 주소 변환 방식 (NAT)과 IP tunneling 에 의한 방식
그리고 Direct Rounting dispatching 기술을 이용한 방식 등이 있다.

3.1.2.1 Network address translation 부하 분산 방식 (네트워크 주소 변환 방식)

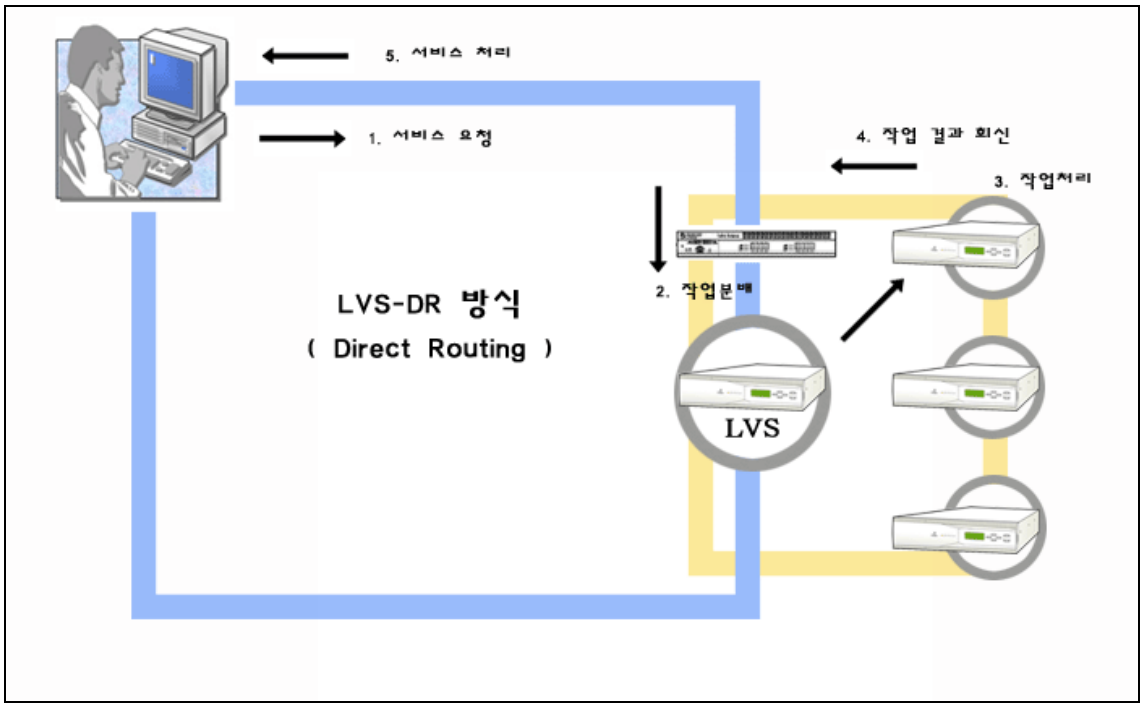
IPv4 에서 IP 주소가 부족하고 보안상에 몇가지 문제가 있어서 인터넷에서 사용할 수 없는 사설 IP (10.0.0.0/255.0.0.0 , 172.16.0.0/255.240.0.0 , 192.168.0.0 /255.255.0.0)를 사용하는 경우가 많아 지고 있다. 이때 인터넷 망에서 사설망의 호스트에 접근하기 위해서 네트워크 주소 변한 (NAT) 기능이 필요하게 된다. 이런 리눅스에서의 NAT 방식의 부하분산 방식은 초기의 리눅스 IP 마스커레이딩 코드와 Steven Clarke 의 포트 포워딩 코드를 재사용 하여 구현하고 있다.



사용자가 LVS 에서 가지고 있는 VIP 를 향해 서비스 요청을 하면 LVS 에서는 그 서비스 요청된 IP 와 Port 가 가상 서버 정책에 있는지를 확인하고 있다면 이 정책에 정해진 스케줄에 따라 실제 작업 서버로 패킷을 포워딩 시키게 된다. 이때 LVS 서버는 사용자가 요청한 패킷의 네트워크 주소 부분을 공인 IP 에서 사설 IP 로 주소를 변환 시켜 같은 사설 망에 존재하는 작업 서버로 보내게 된다. 작업서버는 LVS 서버로부터 패킷을 전달 받고 이에 대한 요청 처리된 내용을 다시 LVS 에 보내게 된다. 그럼 LVS 가 사설 IP 로 된 패킷을 다시 받아 다시 공인 IP 로 주소를 변환시켜서 사용자에게 돌려 보내는 방식이다.

3.1.2.2 Direct Routing 부하 분산 방식

실제 서버와 부하 분산 서버 사이에서 가상 IP 를 공유하는 방식으로 부하 분산 서버와 실제 작업 서버에 모두 동일한 가상 IP 를 가지도록 네트워크 구성해야 한다. 가상 IP 가 설정된 인터페이스를 통해 사용자의 요청을 받아 들여 이 요청이 가상 서버 정책에 적용되는 요청인지를 확인 후 적용되는 패킷이면 정책에 따라 실제 작업 서버로 패킷을 포워딩하게 된다. 이때 실제 서버 역시 별도의 Arp 캐싱을 남기지 않는 Alias 인터페이스를 통해 부하분산서버로부터 포워딩되는 패킷을 받아 들이게 된다.
만일 가상 IP 가 설정된 인터페이스에 arp caching 이 남는다고 하면 사용자가 부하 분산 서버를 통해 초기에 연결된 실제 작업와 연결되면 그 이후부터는 초기 접속한 작업 서버와만 통신이 이루어 지게 된다. 그렇기 때문에 반드시 커널 차원에서의 arp hidden patch 를 시켜 줘야 한다.



DR 방식은 패킷의 데이터 프레임 부분의 MAC 주소만을 변경하여 패킷을 포워딩 하기 때문에 물리적인 세크먼트 (같은 subnet)안에서만 동작이 가능하다. DR 방식의 arp 문제 해결 방법에 대해서는 실제 DR 구축 문서를 참조하길 바란다.

3.1.2.3 IP Tunneling 에 의한 부하 분산 방식

IP 터너링은 IP 화된 정보 (IP datagram)안에 IP 화된 정보를 감싸 넣는 (encapsulate) 기술이다. 리눅스 커널에서 지원하고 이를 이용하여 보다 WAN 을 대상으로 작업 서버 노드를 구성할 수 있게 된다.

실제 가상 IP 주소로 향하는 요구 패킷이 부하분산 서버로 전달 되면 부하분산 서버에서 패킷의 목적지 주소와 포트를 검사하여 가상 서버 정책과 일치 하면 스케줄링에 따라 실제 작업 서버로 전달하게 된다. 이때 패킷을 일반적인 Stream 방식으로 전달하는 것이 아닌 캣슐 형식으로 싸서 전달하게 된다. 이리 전달 되면 작업 서버에서는 감싸진 패킷을 다시 풀고 요청을 처리한 다음 실제 서버의 라우팅 테이블에 따라 사용자에게 직접 결과를 돌려주는 방식이다.

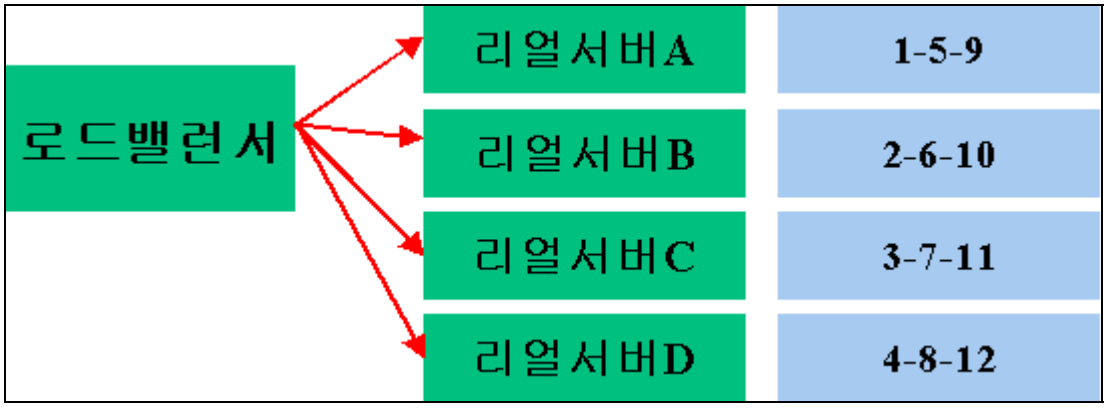
IP 터너링 방식은 DR 방식의 물리적인 세그먼트의 영향을 받지 않고 어디든지 작업을 분산 할 수 있는 장점이 있다. 그래서 거의 무한한 확장성을 가지게 된다. 하지만 모든 서버가 IP 터널링 전송 규약을 지원해야 하기에 지원하지 못하는 OS 에서는 문제가 발생하게 된다. .

3.1.3 작업 할당 방식 (Scheduling Algorithms)

클러스터로부터 서버를 선택하기 위한 네 개의 작업 할당 방식들(scheduling algorithms)이 준비되어있다: Round-Robin, 가중치가 있는(weighted) Round-Robin, 최소-접속(Least-Connection) 그리고 가중치가 있는 최소-접속의 방식들이다. 처음의 두 방식들은 서버에 대한 어떠한 부하 정보(load information)도 가지고 있지 않기 때문에 자명(self-explanatory)하다. 뒤의 두 방식들은 각 서버에 대한 활동 접속 수(active connection number)를 세어 이러한 접속 수에 의해 그들의 부하를 추정한다.

3.1.3.1 Round-Robin Scheduling (라운드 로빈 스케쥴링)

말그대로 라운드-로빈 방식을 이용해 네트웍 연결을 서로 다른 서버에 연결하는 것을 말한 다. 이경우 실제서버의 연결갯수나 반응시간등은 고려를 하지 않는다. 그렇지만 약간의 차이 가 있다. 라운드 로빈 DNS 는 단일한 도메인을 서로 다른 IP 로 해석을 하지만, 스케쥴링의 기초는 호스트 기반이며 캐싱때문에 알고리즘을 효율적으로 사용하기 힘들다. 그래서 실제 서버사이에 동적인 부하 불균형이 심각해 질수 있다. 가상 서버의 스케쥴링 기초는 네트웍 기반이며 라운드 로빈 DNS 에 비해 훨씬 더 훌륭하다.



3.1.3.2 Weighted Round-Robin Scheduling (가중치기반 라운드 로빈 스케쥴링)

가중치기반 라운드 로빈 스케쥴링은 실제 서버에 서로 다른 처리 용량을 지정할 수 있다. 각 서버에 가중치를 부여할 수 있으며, 여기서 지정한 정수값을 통해 처리 용량을 정한다. 기본 가중치는 1 이다. 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 실제 서버에서 네트웍 접속을 셀 필요가 없고 동적 스케쥴링 알고리즘보다 스케쥴링의 과부하가 적으므로 더 많은 실제 서버를 운영 할 수 있다.
그러나 요청에 대한 부하가 매우 많을 경우 실제 서버 사이에 동적인 부하 불균형 상태가 생길 수 있다.
라운드 로빈 스케쥴링은 가중치기반 라운드 로빈 스케쥴링의 특별한 한 종류이며 모든 가중 치가

동일한 경우이다. 가상 서버의 규칙을 변경 하고나서 스케쥴링 순서를 생성 하는 데는 거의 과부하가 걸리지 않으며 실제 스케쥴링에 어떠한 과부하도 추가하지 않는다. 그러므로 라운드 로빈 스케쥴링만 단독으로 실행하는 것은 불필요한 일이다.

3.1.3.3 Least-Connection Scheduling (최소 접속 스케쥴링)

최소 접속 스케쥴링은 가장 접속이 적은 서버로 요청을 직접 연결하는 방식을 말한다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야하므로 동적인 스케쥴링 알고리즘중의 하나이다. 비슷한 성능의 서버로 구성된 가상 서버는 아주 큰 요구가 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 작동 한다는 것을 한눈에 알 수 있을 것이다. 그렇지만 실제로는 TCP 의 TIME_WAIT 상태때문에 아주 좋은 성능을 낼수는 없다. TCP 의 TIME_WAIT 는 보통 2 분이다. 그런데 접속자가 아주 많은 웹 사이트는 2 분동안에 몇천개의 접속을 처리해야 할 경우가 있다. 서버 A 는 서버 B 보다 처리용량이 두배일 경우 서버 A 는 수천개의 요청을 처리하고 TCP 의 TIME_WAIT 상황에 직면하게 된다. 그렇지만 서버 B 는 몇천개의 요청이 처리되기만을 기다리게 된다. 그래서 최소 접속 스케쥴링을 이용할 경우 다양한 처리용량을 지난 서버로 구성되었을 경우 부하분산이 효율적으로 되지 못할 수 있는 것이다.

3.1.3.4 Weighted Least-Connection Scheduling (가중치 기반 최소 접속 스케쥴링)

가중치 기반 최소 접속 스케쥴링은 최소 접속 스케쥴링의 한 부분으로서 각각의 실제 서버에 성능 가중치를 부여할 수 있다. 언제라도 가중치가 높은 서버에서 더 많은 요청을 받을 수 있다. 가상 서버의 관리자는 각각의 실제 서버에 가중치를 부여할 수 있다. 가중치의 비율인 실제 접속자수에 따라 네트웍 접속이 할당된다. 기본 가중치는 1 이다. 가중치가 있는 최소 접속 스케쥴링은 다음과 같이 작동한다:

n 개의 실제 서버가 있는 경우 각 서버 i 는 가중치 Wi (i = 1, ... , n)를 가진다고 가정하자. 서버 i 의 활동 접속(active connection)은 Ci (i = 1, ... , n)이고 모든_접속 은 Ci (i = 1, ... , n)의 합이다. 서버 j 로 가는 넷트웍 접속은 아래와 같다.

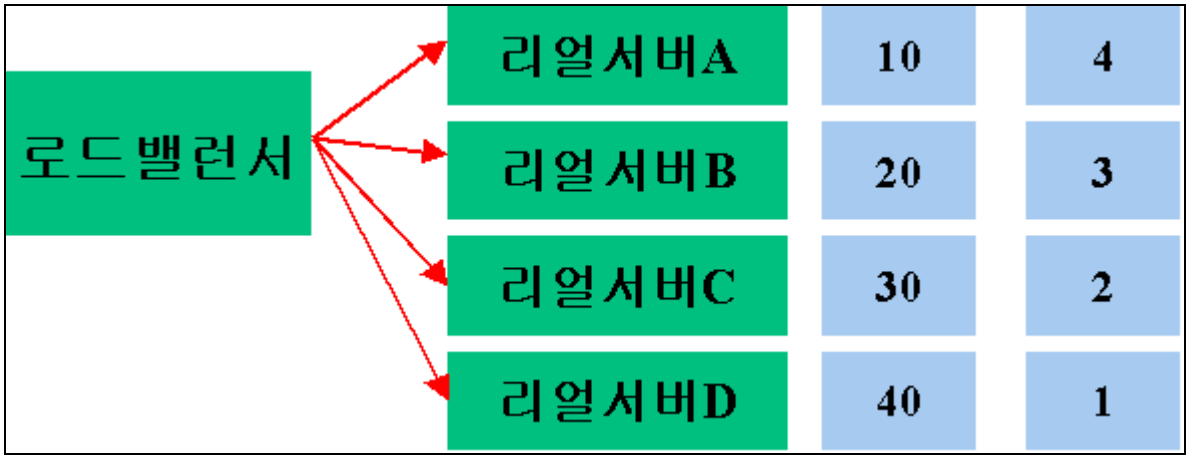
(Cj/ALL_CONNECTIONS)/Wj = min { (Ci/ALL_CONNECTIONS)/Wi } (i=1,..,n)

이 비교에서 ALL_CONNECTIONS 는 상수이므로 Ci 를 모든_접속으로 나눠줄 필요가 없다.

그러면 다음과 같이 최적화될 것이다.

Cj/Wj = min { Ci/Wi } (i=1,..,n)

가중치가 있는 최소 접속 스케쥴링 알고리즘은 최소 접속 스케쥴링 알고리즘에 비해 부가적인 분배 작업이 필요하다. 서버들이 같은 처리 용량을 가졌을 때는 작업 할당의 간접 비용을 최소화하기 위해 최소 접속 스케쥴링과 가중치가 있는 최소 접속 스케쥴링 알고리즘 둘 다 사용할 수 있다.



3.1.4 고 가용성 (High Availability)

중요한 상업적 응용 분야가 인터넷으로 점점 더 많이 옮아옴에 따라 가용성이 높은 서버들을 제공하는 것이 점점 더 중요해지고 있다. 클러스터화된 시스템의 장점중 하나는 하드웨어와 소프트웨어의 여유분이 있다는 것이다. 높은 가용성은 노드(node)나 데몬(deamon)의 장애들(failures)이나 시스템의 재구성(reconfiguring)을 적절히 감지해 클러스터에 남아있는 노드로 작업 부하(workload)를 이전하는 것으로 이룰 수 있다. 가상 서버의 높은 가용성을 현재 "mon"과

"fake" 소프트웨어를 사용해 제공한다. "mon"은 넷트웍 서비스 가용성과 서버 노드를 모니터할 수 있는 범용 자원 모니터링 시스템(general-purpose resource monitoring system)이다. "fake"는 ARP 속이기(spoofing)를 사용해 IP 를 이전하는 소프트웨어다.

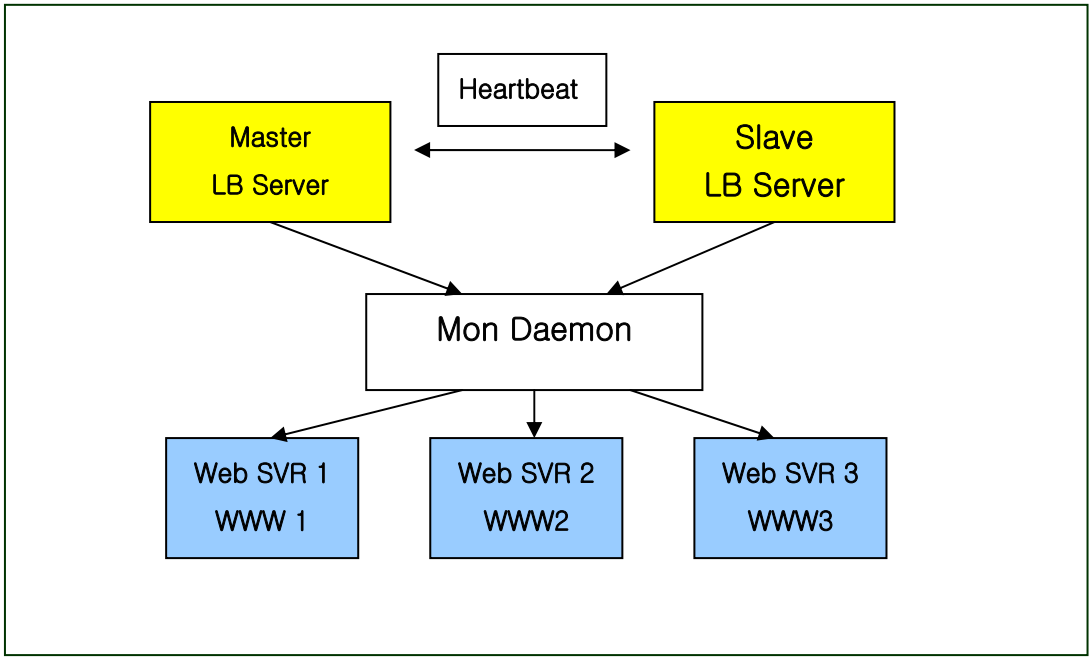
서버의 장애 극복(failover)은 다음과 같이 제어한다: 부하 분산기(load balancer)에 클러스터의 서비스 데몬들과 서버 노드들을 모니터하기 위한 "mon" 데몬이 운영된다. 매 t 초마다 서버 노드들이 살아있는지 감지하도록 fping.monitor 를 구성하고 매 m 분마다 모든 노드의 서비스 데몬들을 감지하기 위해 연관된 모니터 프로그램 역시 구성한다. 예를 들어 http 서비스를 점검하기위해 http.monitor 를, ftp 서비스를 위해 ftp.monitor 를 또, 기타 다른 모니터 프로그램을 사용할 수 있다. 서버 노드나 데몬이 새로 사용 불능 상태(down)가 되거나 사용 가능 상태(up)가 되면 가상 서버 테이블에 규칙을 제거하거나 추가하기위한 경보가 작성된다. 따라서 부하 분산기가 서비스 데몬이나 서버를 제거하거나 복구되었을 때 서비스에 추가하는 것을 자동으로 할 수 있다

3.1.5 클러스터 시스템 기본 구축 환경

앞으로 진행할 리눅스 클러스터 시스템 구축을 위해 테스트 환경으로 4 대 (최소 3 대) 서버가 필요합니다. Load Balance Server 2 대와 일반 웹서버 2 대로 구성하여 리눅스 부하분산 클러스터 시스템을 구축할 것 입니다.

시스템 구성은 Master LB 서버와 Slave LB 서버를 두어 Heartbeat 를 이용하여 LB 서버의 이중화 (High availabiltiy)를 구현할 것이고 Mon 을 이용하여 2 대의 웹서버에 작업 부하 분산 및 Service Fail Over 가 될수 있도록 구현할 것이다.

3.1.5.1 시스템 구성도



3.1.5.2 시스템 기본 정보

Operating System : Redhat Linux 9 (Kernel Version : 2.4.27)
Master LB ip : 211.238.41.167
Slave LB ip : 211.238.41.166
Web Server-1 : 211.238.41.165
Web Server-2 : 211.238.41.164
Virtual ip : 211.238.41.170
Cluster ip (Encluster 시 필요) : 211.238.41.168

LB 서버간의 heartbeat 와 웹서버간의 Data Sync, LB 와 웹서버간의 Mon server monitoring 에 사용되는 네트워크 대역을 Private Network 로 분리할 수도 있다.
이 방법에 대해서는 아래 내용을 충분히 숙지하고, 기본적인 리눅스 네트워킹에 대한 이해만 있으면 충분히 구축할 수 있을 것이다.

3.1.5.3 설치 시 필요 패키지

Kernel : 2.4.26 이상 (본 문서에서는 2.4.27 사용)
ftp://ftp.kernel.org/pub/linux

ipvs : Kernel 2.4.23 이하 버전에만 해당함
http://linuxvirtualserver.org/software

ipvsadm-1.21
http://linuxvirtualserver.org/software

arp hidden patch : 해당 커널에 맞는 패치를 다운 받는다.
http://www.ssi.bg/~ja

Mon 관련 Perl 모듈
ftp://ftp.bora.net/pub/CPAN
ftp://ftp.kornet.net/pub/CPAN
ftp://ftp.nuri.net/pub/CPAN
CPAN 을 이용한 다운 방식도 있음 -> 본문 참조

Mon
ftp://ftp.kernel.org/pub/software/admin/mon

Mon.cgi
http://www.nam-shub.com/files/

fping
ftp://ftp.kernel.org/pub/software/admin/mon

heartbeat
http://linux-ha.org/download

rsync
http://rsync.samba.org/ftp/rsync

Total package
ftp://syszone.co.kr/pub/linux/server/lvspkg

3.2. Open Source 를 이용한 리눅스 부하분산 클러스터 구축

3.2.1. 리눅스 클러스터 커널 환경 구축

이전에는 리눅스 커널에서 기본적으로 ipvs 기능을 지원 하지 않았기에 별도의 커널 패치를 해주어야 한다. ipvs 를 정상적으로 수행하기 위해서는 ipvs 자체 커널 패치와 direct routing 방식을 지원하기 위해서 arp 문제를 해결해 주는 non-arp alias 인터페이스를 만들 때 사용되는 arp hidden 패치를 해주어야 한다.

패치는 각 커널 버전에 맞는 패치를 해야 커널 컴파일 시 에러가 발생하지 않는다.

ipvs 는 커널 2.4.23 까지 ipvs-1.0.10 대 Patch 로 사용이 가능하다.
2.4.26 이후 부터는 ipvs-1.0.11 버전이 커널 속에 기본적으로 포함되어 있다.

3.2.1.1 Kernel 2.4.23 이하 버전일 경우 Kernel patch 방법

ipvs 패치는 로드밸랜싱 서버에만 해주면 되지만 arp 패치는 로드밸랜싱 서버와 리얼 서버(서비스 서버)에 모두 해준다. 여기서는 시스템 초기 셋팅을 동일하게 해야 한다.
두 대가 모두 lvs, real_server 역할을 해야 하기 때문이다.

IPVS module 의 설치 방법에는 두 가지가 있다. 우선 하나는 모듈로 올리는 방법과 커널 안에 추가 하는 방법이다.

- 모듈에 올리는 방법 :

// <path-name> 은 ipvs patch directory

```
# tar xzvf ipvs-0.9.6.tar.gz
# cd ipvs-0.9.6/ipvs
# make
# make -C ipvsadm

# make install
# insmod ip_vs_wlc.o
```

- 커널에 적재하는 방법 :

```
# cd /usr/src
# tar xjvf linux-2.4.23.tar.bz2
# ln -sf linux-2.4.23 linux
# cd linux
# make menuconfig
->        save -> exit
```

zcat ../linux-2.4.23-ipvs.patch.gz | patch p1
cat ../hidden-2.4.23-1.diff | patch p1
만일 커널 버전에 맞지 않는 Patch 를 할경우 에러가 발생할 수 있으니 반드시 커널 버전에 맞는 patch 를 적용하도록 한다.

3.2.1.2 커널 컴파일 하기

아래 커널 컴파일 환경은 Kernel 2.4.27 환경이고, 커널 컴파일 관련 옵션 선택 작업은 2.1.1 에서 다룬 kernel 2.4.23 이전 커널의 ipvs 패치 이후 진행 단계와 동일하게 작업 하면 된다.

이후 모든 설명은 커널 2.4.26 이상 환경을 기준을 설명한다.
tar xjvf linux-2.4.27.tar.bz2
ln -sf linux-2.4.27 linux
cd linux
make menuconfig
아래와 같은 초기 메뉴 화면이 나온다.

```
        Code maturity level options  --->
        Loadable module support  --->
        Processor type and features  --->
        General setup  --->
        Memory Technology Devices (MTD)  --->
        Parallel port support  --->
        Plug and Play configuration  --->
        Block devices  --->
        Multi-device support (RAID and LVM)  --->
        Networking options  --->
        Telephony Support  --->
        ATA/IDE/MFM/RLL support  --->
```

기본적은 커널 옵션은 해당 시스템 사항에 맞게 적용하고 여기서는 ipvs 에 관련된 부분만 적용하도록 하겠다.
먼저 Networking options 메뉴로 가서 아래와 같이 옵션을 체크 한다.

```
        <*> Packet socket
        [*]   Packet socket: mmapped IO
        < > Netlink device emulation
```

```
          [*] Network packet filtering (replaces ipchains)
          [ ]   Network packet filtering debugging
          [*] Socket Filtering
          <*> Unix domain sockets
          [*] TCP/IP networking
          [*]   IP: multicasting
          [ ]   IP: advanced router
          [ ]   IP: kernel level autoconfiguration
          <M>   IP: tunneling
          <M>   IP: GRE tunnels over IP
          [*]     IP: broadcast GRE over IP
          [ ]   IP: multicast routing
          [*]   IP: ARP daemon support (EXPERIMENTAL)
          [ ]   IP: TCP Explicit Congestion Notification support
          [*]   IP: TCP syncookie support (disabled per default)
            IP: Netfilter Configuration  --->
            IP: Virtual Server Configuration  --->
          < >   The IPv6 protocol (EXPERIMENTAL)
          < >   Kernel httpd acceleration (EXPERIMENTAL)
             SCTP Configuration (EXPERIMENTAL)  --->
          < > Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
          < > 802.1Q VLAN Support
          ---
          < > The IPX protocol
          < > Appletalk protocol support
          Appletalk devices  --->
          < > DECnet Support
          < > 802.1d Ethernet Bridging
          < > CCITT X.25 Packet Layer (EXPERIMENTAL)
          < > LAPB Data Link Driver (EXPERIMENTAL)
          [ ] 802.2 LLC (EXPERIMENTAL)
          [ ] Frame Diverter (EXPERIMENTAL)
          < > Acorn Econet/AUN protocols (EXPERIMENTAL)
          < > WAN router
          [ ] Fast switching (read help!)
          [ ] Forwarding between high speed interfaces
          QoS and/or fair queueing  --->
```

```
            Network testing  --->
          IP: Netfilter Configuration  --->
```

IP: Netfilter Configuration 항목으로 가서 iptables 에 관련된 옵션을 아래 와 같이 체크 한다. 실제 iptables 의 경우 패킷의 포워딩 및 NAT, DR 방식에서 중요한 요소로 사용되게 된다.

```
            <*> Connection tracking (required for masq/NAT)
            <M>   FTP protocol support
            <M>   Amanda protocol support
            <M>   TFTP protocol support
            <M>   IRC protocol support
            <M> Userspace queueing via NETLINK (EXPERIMENTAL)
            <*> IP tables support (required for filtering/masq/NAT)
            <M>   limit match support
            <M>   MAC address match support
            <M>   Packet type match support
            <M>   netfilter MARK match support
            <M>   Multiple port match support
            <M>   TOS match support
            <M>   recent match support
            <M>   ECN match support
            <M>   DSCP match support
            <M>   AH/ESP match support
            <M>   LENGTH match support
            <M>   TTL match support
            <M>   tcpmss match support
            <M>   Helper match support
            <M>   Connection state match support
            <M>   Connection tracking match support
            <M>   Unclean match support (EXPERIMENTAL)
            <M>   Owner match support (EXPERIMENTAL)
            <*>   Packet filtering
            <M>     REJECT target support
            <M>     MIRROR target support (EXPERIMENTAL)
            <*>   Full NAT
            <M>     MASQUERADE target support
```

```
        <M>     REDIRECT target support
        [*]     NAT of local connections (READ HELP)
        <M>     Basic SNMP-ALG support (EXPERIMENTAL)
        <*>   Packet mangling
        <M>     TOS target support
        <M>     ECN target support
        <M>     DSCP target support
        <M>     MARK target support
        <M>   LOG target support
        <M>   ULOG target support
        <M>   TCPMSS target support
        <*> ARP tables support
        <M>   ARP packet filtering
```

```
      IP: Virtual Server Configuration  ---> 로 가서 ipvs 옵션을 체크한다.
    <*> virtual server support (EXPERIMENTAL)
    [*]   IP virtual server debugging
    (12)   IPVS connection table size (the Nth power of 2)
    --- IPVS scheduler
    <*>   round-robin scheduling
    <*>   weighted round-robin scheduling
    <*>   least-connection scheduling
    <*>   weighted least-connection scheduling
    <*>   locality-based least-connection scheduling
    <*>   locality-based least-connection with replication scheduling
    <*>   destination hashing scheduling
    <*>   source hashing scheduling
    <*>   shortest expected delay scheduling
    <*>   never queue scheduling
    --- IPVS application helper
    <*>   FTP protocol helper
```

이밖에 Reiserfs, NFS, Coda, XFS, Samba 등등 클러스터 파일 시스템 구성에 관련된 유용한 기능을 추가적을 체크해 주면 된다.

저장하고 나온다. 이런 일련의 작업이 귀찮을 경우 ..

ftp://syszone.co.kr/pub/linux/server/lvspkg/xeon-p4-aic-usb 를 다운 받아

/usr/src/linux 에 놓아두고

--> Load an Alternate Configuration File 에서 불려와서 적용해도 된다.

단 위 커널 설정 파일은 Intel Xeon CPU, Aic79xx SCSI Adapter 환경에 맞는 설정이다.

```
# make dep
# make clean
# make bzImage
# make modules
# make modules_install
```

커널 컴파일 완료 후

```
# cp System.map  /boot/System.map-2.4.27-lvs
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.27-lvs
# cd /boot
# ln -sf System.map-2.4.27-lvs  System.map
# ln -sf vmlinuz-2.4.27-lvs vmlinuz
```

lilo 설정

```
# vi /etc/lilo.conf
---------------------------------------------------------------
prompt
timeout=50
default=linux-lvs
boot=/dev/sda
map=/boot/map
install=/boot/boot.b
message=/boot/message
image=/boot/vmlinuz-2.4.20-8smp
label=linux
initrd=/boot/initrd-2.4.20-8smp.img
```

```
read-only
append="root=LABEL=/"
image=/boot/vmlinuz
label=linux-lvs
read-only
root=/dev/sda2
-----------------------------------------------------------------
```

lilo

리부팅을 한다.

3.2.2. ipvsadm 설치 및 사용 방법

ipvsadm 은 LB 서버에서 클라이언트의 서비스 요청을 받아 실제 작업 서버로 요청 패킷을 포워팅 시키는 정책을 정의하는 명령어 이다. 부하분산 서버의 Virtual Server 정책은 모두 이 명령어로 정의하게 된다.

3.2.2.1 ipvsadm 설치

ipvsadm-1.21.tar.gz 파일을 /usr/local/src 밑에 옮겨 놓는다.

```
# cd /usr/local/src
# tar xzvf ipvsadm-1.21.tar.gz
# cd ipvsadm-1.21
# make
# make install
# ln -sf /sbin/ipvsadm /usr/sbin/ipvsadm    -> Encluster 사용 시 필요함.
# ipvsadm
-----------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
```

위와 같은 내용이 나오면 정상적으로 설치가 완료 된것이다.

3.2.2.2 ipvsadm 사용 방법

ipvsadm 옵션 설명
-A : 서비스를 추가 한다.
-D : 서비스 삭제
-E : 서비스 수정
-C : 정의된 모든 정책 삭제
-s : 스케줄러 선택 (wlc : weight least connection scheduling)
-a : add server (-e : EDIT , -d : Delete)
-t : TCP service 추가 (-u : UDP , -f : Firewall)
-r : 옵션 다음에 Real Server ip address 를 적는다.
-i : 패킷전달 방식 (ip tunneling 방식) , -g : Direct Routing 방식, -m : NAT
-w : wlc 방식에서 가중치 (기본 가중치는 1 로 주어짐)
--set tcp tcpfin udp : 연결 시간 제한 설정 (set connection timeout value)
-p : persistent 시간 지정

예) 정책 설정 방법

```
# ipvsadm -A -t 211.238.41.170:80 -s wlc -p 360
# ipvsadm
-------------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
          -> RemoteAddress:Port               Forward Weight ActiveConn InActConn
         TCP   211.238.41.170:http wlc persistent 360
-------------------------------------------------------------------

# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.165 -g -w 100
# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.166 -g -w 100
# ipvsadm
-------------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port               Forward Weight ActiveConn InActConn
TCP   211.238.41.170:http wlc persistent 360
  -> 211.238.41.166:http            Route    100      0          0
  -> 211.238.41.165:http            Route    100      0          0
-------------------------------------------------------------------
```

3.2.3 LB 서버와 Real Server 의 네트워크 구성 설정

3.2.3.1 LB 서버 네트워크 구성

LB 서버에는 특정 인터페이스에 VIP 를 설정하여야 한다. 여기서는 eth0:0 인터페이스에 VIP 를 설정하도록 한다.

-> eth0 : real IP
-> eth0:0 virtual IP
-> VIP : 211.238.41.170
-> Web Server-1 : 211.238.41.165
-> Web Server-2 : 211.238.41.164

ifconfig eth0:0 211.238.41.170 netmask 255.2555.255.255 broadcast 211.238.41.170
route add -host 211.238.41.170 dev eth0:0
route -n 으로 확인하여 해당 flags 값이 UH 면 된다.

route –n

```
------------------------------------------------------------------------
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
211.238.41.170  0.0.0.0         255.255.255.255 UH    0      0        0 eth0
211.238.41.160  0.0.0.0         255.255.255.224 U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         211.238.41.190  0.0.0.0         UG    0      0        0 eth0
------------------------------------------------------------------------
```

kernel forward parameter 값을 1 로 수정한다.
echo 1 > /proc/sys/net/ipv4/ip_forward
매번 부팅때마다 적용하기 위해서는 /etc/sysctl.conf 에서 해당 부분 설정을
0 에서 1 로 수정한다.

net.ipv4.ip_forward = 1
*** 만일 NAT 구성 시에는 /etc/sysconfig/network 파일을 열어 아래 내용을 추가해 준다.
IP_FORWARD(대문자)=true(소문자)

- **ipvsadm 정책 설정**

```
# ipvsadm -A -t 211.238.41.170:80 -s wlc
# ipvsadm -A -t 211.238.41.170:22 -s 짖
# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.165 -g
# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.164 –g
# ipvsadm -a -t 211.238.41.170:22 -r 211.238.41.165 –g
# ipvsadm
---------------------------------------------------------------------- IP Virtual Server version
1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  211.238.41.170:ssh 짖
  -> 211.238.41.165:ssh           Route   1      0          0
TCP  211.238.41.170:http wlc
  -> 211.238.41.165:http          Route   1      0          0
  -> 211.238.41.164:http          Route   1      0          0
---------------------------------------------------------------------
```

3.2.3.2 Real Server 네트워크 구성

Real Server 에서는 LB 로 부터 분배 되어지는 패킷을 받기 위해 DR, NAT, Tun 별로 모두 네트워크 구성이 달라진다.
NAT 의 경우는 별도의 VIP 에 대한 Alias Device 를 가질 필요가 없다. 단지 네트워크 구성에서 Gateway 를 LB 서버의 통신 가능한 IP 로 정해주면 된다.
DR 이나 Tun 의 경우에는 Real Server 에서도 VIP 에 대한 IP 를 Non arp alias Device 에 설정을 해주어야 한다. 그래야 LB 서버에서 포워딩되는 패킷을 받아 들일수가 있다

여기서는 Direct routing 의 방식으로 설정을 하도록 하겠다.

-> eth0 : Real IP
-> lo:0 : VIP

```
# ifconfig lo:0 211.238.41.170 netmask 255.255.255.255 broadcast 211.238.41.170
# route add -host 211.238.41.170 dev lo:0
```

여기서 별도의 non arp alias device 를 잡지 않아도 iptables 의 rediect 기능을 이용하여 Direct routeing 방식으로 네트워크를 구성할 수 있다.
Encluster 에서는 이 방법을 사용하고 있다.

iptables -t nat -A PREROUTING -j REDIRECT -d 211.238.41.170 -p tcp
iptables -t nat -A PREROUTING -j REDIRECT -d 211.238.41.170 -p u 에
Real Server 네트워크 구성에서 VIP 가 설정 되는 Device 의 경우 반드시 arp caching 남으면 안된다.
arp caching 남게 되면 vip 를 통해 초기 접속된 Real Server 로만 계속 접속을 시도하게 됨으로 LB 서버의 분배 스케줄링에 문제가 발생하게 된다.
다음 장에서 DR 과 tun 방식에서의 Arp 캐싱 문제 해결 방법에 대해 알아보자.

3.2.4. Direct Routing 방식에서의 ARP 문제 해결

arp caching 문제 해결은 2.1 장에서 얘기한 hidden patch 를 통해 해결 할 수 있다.
하지만 커널 2.4.26 이상부터는 별도의 hidden patch 없이 arp_ignore , arp_announce 값을 이용하여 arp 문제를 해결 할 수 있다

**** 참고 ****

하지만 http://www.ssi.com/~ja 에는 2.4.26~27 버전대의 hidden patch 가 올려져 있지만 hidden patch 를 한경우와 arp_ignore , arp_announce 통해 처리한 경우와 다른 점은 찾지 못했당.

- hidden patch 를 통해 arp 캐싱 문제 해결
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
- arp_ignore , arp_announce 통해 해결하는 방법
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

위 설정은 커널 파라메터 설정을 변경하는 것으로 1 장에서 언급 했듯이 매 부팅 때 마다 위 설정을 유지 하기 위해서는 위의 명령 내용을 /etc/rc.d/rc.local 에 적어 주던지 아님 /etc/sysctl.conf 파일 내용에 해당 내용을 아래와 같이 추가하도록 한다.

cat /etc/sysctl.conf
---
net.ipv4.ip_forward = 1 ## 0 에서 1 로 수정
# 아래는 추가

```
net.ipv4.conf.lo.arp_ignore = 1
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
# hidden patch 적용 경우에는 아래 추가
net.ipv4.conf.all.hidden = 1
net.ipv4.conf.lo.hidden = 1
```

3.2.5. 부하 분산 테스트

앞장에서 설정한 정책에 따라 잘 분산이 되는지 확인해 보도록 하자.
확인에 앞서 먼저 정확한 분산 확인을 위해 apache 의 httpd.conf 의 설정 값을 수정하도록 한다.

```
KeepAlive = Off
MaxKeepAliveRequests 1
```

그리고 웹 페이지 테스트를 할수 있는 index 페이지를 만들어 보도록 하자.

```
# vi index.html
```

```
<html>
<head>
<title>node01</title>
</head>
<body>
<font size=5 color=red> node01 </font>
</body>
</html>
```

그런 후 브라우저를 열어서 VIP 로 웹 접속을 해보도록 하자.
웹 접속이 이루어 졌으면 브라우저 "새로 고침"을 계속 눌려보아, 화면이 서버 별로 변경이 되는지를 확인하면 된다.
이밖에 아래 Perl 스크립터를 이용하여 확인을 해보아도 된다. 아래 스크립터로 테스트를 하기 위해서는 index 웹 페이지의 title 을 반드시 적어 주어야 한다.
아래 스크립터는 웹 페이지의 title 내용을 출력하라는 내용의 스크립트다.

```
# vi get_title.pl
-------------------------------------------------------------------------
#!/usr/bin/perl –w
#
#클러스터 서버들의 타이틀을 가져오는 프로그램.
#
      use LWP::UserAgent;
      use HTTP::Request;
      use HTTP::Response;
      use URI::Heuristic;
      my $raw_url = shift or die "usage: $0 url₩n";
      for ( $i = 0 ; $i <= 100 ; $i++){      # 이곳의 숫자를 변경해 반복횟수를 변경한다.
      my $url = URI::Heuristic::uf_urlstr($raw_url);
      $| = 1;
      printf "%s =>₩n₩t", $url;
      my $ua = LWP::UserAgent->new();
      $ua->agent("schmozilla/v9.14 Platinum");
      my $req = HTTP::Request->new(GET=>$url);
      $req->referer("http://wizard.yellowbrick.oz");
      my $response = $ua->request($req);
      if ($response->is_error()){
          printf "%s₩n", $response->status_line;
          }
      else{
          my $count;
          my $bytes;
    my $content = $response->content();
    $bytes = length $content;
    $count = ($content =~ tr/₩n/₩n/);
    printf "%s (%d lines, %d bytes)₩n", $response->title(), $count, $bytes;
        }
-------------------------------------------------------------------------

# chmod 755 get_title.pl
# ./get_title.pl http://211.238.41.170
```

```
http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
```

3.2.6. Server failed 를 감지하는 MON 적용 하기

지금까지 기본적인 로드 밸런싱(작업분배)은 설정에 대해 알아보았다.
하지만 real_server(앞으로 real 로 표기 하겠다.)중 하나가 다운되어도 master_lvs 에서는 스케줄 규칙에 따라 패킷을 죽은 real 에도 정해진 규칙대로 분배하게 된다. 그렇기 때문에 패킷이 죽은 real 로 보내질 때는 접속이 되지 않는다.
이를 방지하기 위해서 mon 이란 프로그램을 설치한다. mon 은 정해진 시간마다 각 real 서버들을 감시 하면서 죽은 real 서버가 발견대면 ipvsadm 에서 죽은 real 서버로 분배 되는 작업 규칙을 제거 하여 더 이상 작업을 분배하지 않도록 한다.
그리고 죽었던 real 서버가 살아나면 이또한 감지하고 다시 정해진 작업분배를 계속 하게 된다.
이제 mon 설치 방법에 대하여 알아보도록 하자.

3.2.6.1 Mon 설치 준비 하기

mon 은 기본적으로 perl 로 만들어진 각 서비스 체크 모듈을 가지고 서비스의 상태를 파악하여 해당 서비스가 죽은 경우 특정 액션을 취할 수 있게 해주는 프로그램이다.
이를 이용하면 ipvsadm 정책을 새로이 만들 수도 있고, 기타 다른 곳에도 응용할 수 있다.

mon 을 이용하기 위해서는 기본적으로 perl ver 5.x 이상이 설치 되어 있어야 한다.

또 몇 가지 기본 perl modules 과 fping 정도의 프로그램을 먼저 설치 해 두어야 한다.
perl modules 은 CPAN 을 이용하여 쉽게 설치 할수 있고, 직접 다운 받아 설치할 수도 있다.
perl modules 설치 전에 perl 용 시스템 header 파일을 만들어야 한다.

```
# cd /usr/include
# h2ph *.h sys/*.h asm/*.h
```

- CPAN 이용방법

```
# perl-MCPAN -e shell
```

처음 이용할 경우 다음과 같은 기본 설정 단계가 나온다.

```
-----------------------------------------------------------------------------
/usr/lib/perl5/5.8.0/CPAN/Config.pm initialized.
CPAN is the world-wide archive of perl resources. It consists of about
100 sites that all replicate the same contents all around the globe.
Many countries have at least one CPAN site already.
The resources found on CPAN are easily accessible with the CPAN.pm module. If you

want to use CPAN.pm, you have to configure it properly. If you do not want to enter a dialog now,
you can answer 'no' to this question and I'll try to autoconfigure.

(Note: you can revisit this dialog anytime later by typing 'o conf init' at the cpan prompt.)
```

Are you ready for manual configuration? [yes] -> yes

```
The following questions are intended to help you with the
configuration. The CPAN module needs a directory of its own to cache
important index files and maybe keep a temporary mirror of CPAN files.
This may be a site-wide directory or a personal directory.
First of all, I'd like to create this directory. Where?
CPAN build and cache directory? [/root/.cpan].
```

기본 내용으로 설정한다.

```
cpan shell -- CPAN exploration and modules installation (v1.61)
ReadLine support available (try 'install Bundle::CPAN')
```

cpan>

기본 설정이 완료 되면 Cpan> 프롬프트 상태로 전환 된다. 여기서 원한는 모듈을 Install 명령으로 아래와 같이 설치 하면 된다.

cpan> install Time::Period
cpan> install Time::HiRes
cpan> install Convert::BER
cpan> install Mon::Client
cpan> install Mon::Protocol
cpan> install Mon::SNMP

- 직접 다운 받아 설치 하기

ftp://ftp.bora.net/pub/CPAN 에서 원하는 perl modules 을 다운 받는다.

tar xzvf Time-Hires-01.20.tar.gz
cd Time-Hires-01-20
perl Makefile.PL
make

make test
make install

- fping 설치 하기

cd /usr/local/src
tar xjvf fping-2.2b1.tar.bz2
cd fping-2.2.b1
./configure
make

CONFIG_FILES= CONFIG_HEADERS=./config.h ./config.status
creating ./config.h
gcc -c -DHAVE_CONFIG_H -I. -I. -I. -g -O2 fping.c

fping.c:222: conflicting types for `sys_errlist'
/usr/include/bits/sys_errlist.h:28: previous declaration of `sys_errlist'
make: *** [fping.o] 오류 1

-> 이와 같이 오류가 발생하면 fping.c 파일의 222번 줄을 수정한다.

/* externals */
extern char *optarg;
extern int optind,opterr;
extern char *_sys_errlist[]; -> *sys_errlist[];
extern int h_errno;
#ifdef __cplusplus
}
.. 그런 후 ..

---

make
make install

3.2.6.2 Mon 설치 하기

tar xzvf mon-0.99.2.tar.gz -c /usr/lib
cd /usr/lib
mv mon-0.99.2 mon
cd mon
mkdir /etc/mon /var/log/mon
install m644 doc/mon.8 /usr/man/man8
install m644 doc/moncmd.1 /usr/man/man1
install m644 doc/monshow.1 /usr/man/man1
install m644 etc/auth.cf /etc/mon
install m644 etc/example.cf /etc/mon
install m700 etc/S99mon /etc/rc.d/init.d/mon
vi /etc/rc.d/init.d/mon 을 수정한다.

#!/bin/sh
#
start/stop the mon server
#

```
# You probably want to set the path to include
# nothing but local filesystems.

        #
# chkconfig: 2345 99 10
# description: mon system monitoring daemon
# processname: mon
# config: /etc/mon/mon.cf
# pidfile: /var/run/mon.pid
#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH

# Source function library.
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
start)
echo -n "Starting mon daemon: "
daemon /usr/lib/mon/mon -f -c /etc/mon/mon.cf   --> 이부분 -f 추가
```

```
# vi /etc/service
```

맨 아래다가..

```
        mon             2583/tcp        # MON
        mon             2583/udp         # MON traps
```

추가 ..

```
# cp /etc/mon/example.cf /etc/mon/mon.cf
```

이것으로 mon 설치를 완료한다.
간단한 테스트를 해보도록 하자

mon 의 fping modules 을 이용한 서버 상태 확인 하는 테스트이다.

```
# /usr/lib/mon/mon.d/fping.monitor syszone.co.kr

start time: Thu Aug 26 18:11:37 2004
end time   : Thu Aug 26 18:11:37 2004
duration   : 0 seconds
------------------------------------------------------------------------
reachable hosts                          rtt
------------------------------------------------------------------------
syszone.co.kr                            0.19 ms
```

다음 장에서는 실제 mon 을 이용하여 LB 서버에 Fail Over 기능을 추가 해보자.

3.2.6.3 Mon 을 이용한 Load Balance 에 Fail Over 기능 추가하기

/usr/lib/mon/mon.d 디렉토리 안의 file 들은 각 시스템의 해당 서비스를 모니터링 하는 perl scripts 이다. mon 의 장점과 단점이 될수 있는 부분이 기본적으로 포함되어져 있는 monitor scripts 가 없는 경우 일반인들이 싶게 개별 사이트의 특수한 프로그램을 감시 할수 없다는 것이다.
하지만 약간의 Perl 지식 만 갖춘 상태라고 하면 어떤 형태의 moniter modules 도 쉽게 만들 수 있을 것 이다.
이장에서는 mon을 이용하여 virtual address 정책을 실제 작업가 죽어 있을 경우 정책에서 자동 제거하고 다시 살아나면 다시 포함하는 기능을 추가하는 방법 에 대해 알아 보겠다.

mon 의 기본 설정 파일은 한개이다. /etc/mon/mon.cf 가 그것이다.
이 설정 파일의 내용만 이해하면 시스템 엔지니어가 원하는 어떤 형태의 감시툴도 만들 수 있을 것이다.

```
# vi /etc/mon/mon.cf
------------------------------------------------------------------------
# global options
#
cfbasedir     = /etc/mon            # mon.cf 위치
logdir        = /var/log/mon
alertdir      = /usr/lib/mon/alert.d
```

```
mondir          = /usr/lib/mon/mon.d
maxprocs        = 20
histlength = 100
randstart = 30s

# authentication types:
#     getpwnam          standard Unix passwd, NOT for shadow passwords
#     shadow            Unix shadow passwords (not implemented)
#     userfile          "mon" user file


authtype = userfile        # mon.cgi 사용시 로그인 가능 기능
userfile = /etc/mon/mon.user

dtlogging = yes
dtlogfile = downtime.log

############### 호스트 그룹 지정 ####################
# hostgroup 은 실제 모니터링을 해야할 서버와 그 서버의 별칭을 정하는 곳
# 아래에서 HTTP-server1 은 서비스 별칭이고 www1 은 호스트네임이다.
# 서비스 별칭은 실제 서비스에 대한 모니터링 관련 설정을 하는 watch 에서
# 사용된다.

hostgroup HTTP-server1 www1

hostgroup HTTP-server2 www2

watch HTTP-server1
service ping
            description ping servers in www1
            interval 5s
            monitor fping.monitor
            period wd {Sun-Sat}
#           alert mail.alert alang@clunix.com
#           alert page.alert alang@clunix.com
            alertevery 10m
            alertafter 2 30m
```

```
        alert test.alert "www1 Server is DOWN"
        upalert test.alert "www1 Server is UP"
service HTTP
        description http to servers in www1
        interval 3s
        monitor http.monitor
#   depend HTTP-server1:ping
        period wd {Sun-Sat}
        alertevery 30m
#       alert mail.alert alang@clunix.com
#       alert page.alert alang@clunix.com
        alert test.alert "HTTPd is DOWN"
        upalert test.alert "HTTPd is UP"
        alert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.165:80 ₩
                -W 100 -F dr -X down
        upalert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.165:80 ₩
                -W 100 -F dr

watch HTTP-server2
service ping
        description ping servers in www2
        interval 5s
        monitor fping.monitor
        period wd {Sun-Sat}
#       alert mail.alert alang@clunix.com
#       alert page.alert alang@clunix.com
        alertevery 10m
        alertafter 2 30m
        alert test.alert "www2 Server is DOWN"
        upalert test.alert "www2 Server is UP"
service HTTP
        description http to servers in www2
        interval 3s
        monitor http.monitor
#       depend HTTP-server2:ping
        period wd {Sun-Sat}
        alertevery 30m
```

```
#           alert mail.alert alang@clunix.com
#           alert page.alert alang@clunix.com
            alert test.alert "www2 HTTPd is DOWN"
            upalert test.alert "www2 HTTPd is UP"
            alert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.166:80 ₩
                        -W 100 -F dr -X down
            upalert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.166:80 ₩
                        -W 100 -F dr
```

Mon 설정이 완료된 후 Mon 설정에서 웹서버에 이상이 있을 때 호출하는 스크립터인 lvs.alert를 만들도록 한다.

```
# vi /usr/local/mon/alert.d/lvs.alert
```

```
#!/usr/bin/perl
#
# lvs.alert - Linux Virtual Server alert for mon
#
# It can be activated by mon to remove a real server when the
# service is down, or add the server when the service is up.
#
#
use Getopt::Std;
    getopts ("s:g:h:t:l:P:V:R:W:F:X:u");
        $ipvsadm = "/sbin/ipvsadm";
        $protocol = $opt_P;
        $virtual_service = $opt_V;
        $remote = $opt_R;
        $status = $opt_X;
if ($status eq "down"){

# <"udp 를 사용 하시려면 이 부분에 프로토콜 체크를 한번 더 하셔야
합니다.">

        system("$ipvsadm -d -t $virtual_service -r $remote");
        exit 0;
        }
```

```
else {
        $weight = $opt_W;
                if ($opt_F eq "nat") {
   $forwarding = "-m";
   }
                elsif ($opt_F eq "tun") {
   $forwarding = "-i";
   }
                else {
   $forwarding = "-g";
   }

   if ($protocol eq "tcp") {
   system("$ipvsadm -a -t $virtual_service -r $remote -w $weight
$forwarding");
   exit 0;
                }
   else {
   system("$ipvsadm -a -u $virtual_service -r $remote -w $weight
$forwarding");
   exit 0;
                }
exit 0;
};
```

해당 스크립터의 퍼미션을 755하여 실행 권한을 준다.

이제 설정을 저장 후 Mon 데몬을 시작해 보자

```
# /etc/rc.d/init.d/mon start
```

restart 는 불안하니 stop, start 만 사용하도록 하자.

2.6.4 Mon.cgi 를 이용하여 웹에서 Mon 제어하기

moncgi란 test-mode가 아닌 그래픽 모드에서 서버를 모니터링 할 수 있게끔

해주는 툴이다. 물런 설치 하지 않아도 LB Fail over 기능에는 전혀 상관 없다.

설치 과정에 대해 알아보자

Mon 웹프로그램을 관리할 시스템 계정 생성

```
# adduser -u 90 -c "MonCGI Apache User" -d /usr/local/moncgi -s /sbin/nologin moncgi
```

mon 웹프로그램에 로그인할 계정 생성

```
# /usr/local/apache/bin/htpasswd -c /etc/mon/mon.user moncgi
# cd /usr/local/src
# tar xzvf apache_1.3.31.tar.gz
# cd apache_1.3.31
# vi config.layout      -> 아래와 부분을 추가함
```

```
----------------------------------------------------------------------
        <Layout moncgi>
            prefix:           /usr/local/moncgi
            exec_prefix:      $prefix
            bindir:           $exec_prefix/bin
            sbindir:          $exec_prefix/bin
            libexecdir:       $exec_prefix/libexec
            mandir:           $prefix/man
            sysconfdir:       /etc/mon/moncgi
            datadir:          $prefix
            iconsdir:         $datadir/icons
            htdocsdir:        $datadir/htdocs
            manualdir:        $htdocsdir/manual
            cgidir:           $datadir/cgi-bin
            includedir:       $prefix/include
            localstatedir: /var/log/mon
            runtimedir:       $localstatedir/logs
            logfiledir:       $localstatedir/logs
            proxycachedir: $localstatedir/proxy
        </Layout>
----------------------------------------------------------------------
# ./configure --with-layout=moncgi --show-layout
# ./configure --with-layout=moncgi --with-port=9000 --server-uid=moncgi  ₩
--server-gid=moncgi --disable-module=autoindex --disable-module=imap ₩
```

```
--disable-module=include --disable-module=negotiation ₩
--disable-module=status --disable-module=userdir
# make
# make install
# cd /etc/mon/moncgi
# rm -rf access* srm* magic*
# grep -v "#" httpd.conf | grep ^. > tt      --> 주석  처리된 내용을 삭제함.
# mv ./tt ./httpd.conf
# vi httpd.conf
```

```
        ########################## 전체 환경 설정 ####################
        ServerType                  standalone
        ServerRoot                  "/usr/local/moncgi"
        PidFile                     /var/log/mon/moncgi/httpd.pid
        ScoreBoardFile              /var/log/mon/moncgi/httpd.scoreboard

        Timeout                     300
        KeepAlive                   On
        MaxKeepAliveRequests        100
        KeepAliveTimeout            15
        MinSpareServers             1
        MaxSpareServers             2
        StartServers                0
        MaxClients                  10
        MaxRequestsPerChild         0

        Listen              211.238.41.167:9000

        ########################## 주 서버 설정 ######################

        Port 9000
        User moncgi
        Group moncgi
        ServerAdmin root@www3.clunix.org

        DocumentRoot "/usr/local/moncgi/cgi-bin"
```

```
<Directory "/usr/local/moncgi/cgi-bin">
        Options ExecCGI
        AllowOverride None
        AddHandler cgi-script .cgi
#       Order deny,allow
#       Deny from all
#       Allow from 211.238.41.180
</Directory>

DirectoryIndex mon.cgi

UseCanonicalName On

TypesConfig /etc/mon/moncgi/mime.types
DefaultType text/plain

HostnameLookups Off

ErrorLog /var/log/mon/moncgi/error_log
LogLevel warn
LogFormat "%h %l %u %t ₩"%r₩" %>s %b" common
CustomLog /var/log/mon/moncgi/access_log common

ServerSignature On

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4₩.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4₩.0" force-response-1.0
BrowserMatch "Java/1₩.0" force-response-1.0
BrowserMatch "JDK/1₩.0" force-response-1.0
```

```
# cd /usr/local/moncgi 로 이동
# rm -rf htdocs icons libexec man cgi-bin/*
# cd /etc/rc.d/init.d/
# ln -sf /usr/local/moncgi/bin/apachectl moncgi
```

/etc/rc.d/init.d/moncgi restart

이제 moncgi 가 재대로 동작하는지 확인

lsof -i | grep 9000

```
------------------------------------------------------------------------
httpd      3805     root    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
httpd      3816 moncgi    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
httpd      3817 moncgi    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
httpd      3818 moncgi    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
httpd      3819 moncgi    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
httpd      3820 moncgi    16u  IPv4 419398         TCP www3.clunix.org:9000 (LISTEN)
```

chkconfig --add moncgi
chkconfig --level 3 moncgi on

moncgi를 설치 하기 위해서는 몇가지 perl modules 이 더 필요하다.
필요한 모듈로는 Crypt::TripleDES , Math::TrulyRandom 이 있다.
앞서 설명한 CPAN 을 이용하도록 하자.

perl -MCPAN -e shell
cpan > install Crypt::TripleDES
cpan > look Math::TrulyRandom
perl Makefile.PL
make && make install

mon.cgi-1.52.tar.gz 을 다운 받는다. (http://www.nam-shub.com/files/)

mkdir /etc/mon/moncgi
/usr/local/src
tar xzvf mon.cgi-1.52.tar.gz
cd mon.cgi-1.52
install m755 mon.cgi /usr/local/moncgi/cgi-bin
install m644 config/mon.cgi.cf /etc/mon/moncgi
install m700 util/moncgi-appsecret.pl /usr/sbin/moncgi-appsecret
cd .. ; rm -rf mon.cgi*
cd /usr/local/moncgi/cgi-bin

mon.cgi 를 수정한다.

vi mon.cgi

$moncgi_config_file = "/etc/mon/moncgi/mon.cgi.cf" 176 줄

/etc/mon/moncgi/mon.cgi.cf 에서 Reloas_time, must_login , Login_expire_time 을 설정이 되어 있는지 확인을 해본다. 환경에 맞게 수정하라.

그런후 http://IP:9000/mon.cgi

(접근후 계정과 password를 넣는 창이 있을 것이다. 이것은 MON에서 추가해준 유저를 넣으면 된다.)

Mon 데몬을 자동으로 실행하는 init script 를 만들어 보자

vi /etc/rc.d/init.d/puck

```
#!/bin/sh
#
# load balancer daemon scripts
#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH
IPVSADM=/sbin/ipvsadm
MON=/usr/lib/mon/mon
RETVAL=0
#Source function library.
. /etc/rc.d/init.d/functions
case "$1" in
start)
   if [ -x $IPVSADM ]
   then
#
# ipvs system 설정 부분
#
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ifconfig eth0:0 211.238.41.170 netmask 255.255.255.255 ₩
```

```
      broadcast 211.238.41.170 up
   route add -host 211.238.41.170 dev eth0:0
#   echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
#   echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
#   echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
#   echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce

   $IPVSADM -A -t 211.238.41.170:80 -s wlc
   $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.165:80 -g -w 100
   $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.166:80 -g -w 100
#
# 추가되는 real_server 는 아래에 같은형식으로 추가하면 된다.
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.165 -g -w 2
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.166-g -w 2
#


   echo -n "started loadbalancer daemon:"
   daemon   $MON -f -c /etc/mon/mon.cf
   RETVAL=$?
   echo
   [ $RETVAL = 0 ] && touch /var/lock/subsys/punk
   echo
   fi
   ;;
   stop)
   if [ -x $IPVSADM ]
then
echo -n "puck daemon stopping..."
$IPVSADM -C
ifconfig eth0:0 down
killproc mon
rm -f /var/lock/subsys/punk
killall http.monitor
echo -n "puck daemon killed"
echo
   fi
```

```
        ;;
        *)
        echo "Usage : puck {start|stop}"

      exit 1
      esac
      exit 0
```

이제 /etc/rc.d/init.d/puck initscripts 로 mon 을 손쉽게 제어할 수 있다.
LB 서버에서 /etc/rc.d/init.d/puck start 한 후 Web Server-1, Web Server-2의 httpd 데몬을 stop, start 시키면서 ipvsadm tables 에 상황에 맞게 적용되는지를 확인한다.

3.2.7. LB 서버간의 이중화 HA 구축 하기 (Heartbeat)

Load Balance 방식의 최대 단점 중 하나가 LB 서버가 다운 될을 경우 모든 서비스가 정지된다. 그러므로 LB 서버의 이중화는 필수적인 사항이다. 이를 위해 heartbeat 란 프로그램을 이용하여 LB 서버를 이중화 해야 한다.

heartbeat 프로그램을 다운 받아 /usr/local/src 에 놓아둔다.

heartbeat 에 이용되는 프로그램은 다음과 같다.
http://www.linux-ha.org/ 에서 heartbeat 최신버전을 다운 받는다.

heartbeat-pils-1.2.2-8.rh.9
heartbeat-stonith-1.2.2-8.rh.9
heartbeat-1.2.2-8.rh.9

3.2.7.1 Heartbeat 설치 및 기본 설정

Heartbeat는 Master_lvs 와 Slave_lvs 에 모두 설치한다.

```
# rpm -Uvh heartbeat-*
```

설정파일은 /etc/ha.d 에 있다.

cd /etc/ha.d

heartbeat 설정은

ha.cf
haresources
authkeys

rpm -q heartbeat -d 라고 하면 설정파일의 샘플파일이 /usr/share/doc/packages/heartbeat란 디렉토리에 있다는 것을 알수 있다.

샘플 파일을 /etc/ha.d 로 복사한다.

cp ha.cf /etc/ha.d
cp haresources /etc/ha.d
cp authkeys /etc/ha.d

복사가 완료되었으면 다시 /etc/ha.d 디렉토리돌아온다.

/etc/ha.d/ha.cf 설정 내용 ----------

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 5
hopfudge 1
udpport 1001
udp     eth0
node    www3    # Master LB Server
node    www2    # Slave LB Server
```

/etc/ha.d/authkeys 설정 내용 ----------

```
#auth 1
```

```
        #1 crc
        #2 sha1 HI!
        #3 md5 Hello!
        auth 1
        1 sha1 HI!
```

/etc/ha.d/haresource 설정 내용 ----------

```
# 설정 형식에 맞게 설정해야 합니다. 설정형식은...
# masternode_name    VIP                DAEMON_name
# DAEMON_name 은 /etc/rc.d/init.d 밑에 있는 initscripts 를 적으면 된다

www3      211.238.41.170   puck
```

여기까지 Master_LB 에 대한 설정은 모두 끝났다.

3.2.7.2 Slave LB 서버가 Director 와 Real Service 의 두 가지 역할 담당 시 Slave LB 서버 설정

만일 Slave_LB 서버를 LB 역할 로만 둔 다라고 하면 Master_LB 서버와 동일하게 Mon 설정과 Heartbeat
설정을 해주면 된다. 하지만 Slave_LB 서버를 평소에 RealServer 역할을 수행하다가 Master LB 서버에
문제 발생 시 Real Server 에서 LB 서버로 역할을 전환하는 방식으로 한다고 하면 Slave_LB 서버의
Mon 과 Heartbeat 설정에서 몇 가지 수정을 해주어야 한다.

먼저 Slave_LB 서버가 Real Server 역할을 수행 시 lo:0 를 추가할 때 쓰이는 스크립터를 만들도록
한다.

/etc/ha.d/lvs-r.sh----------

```
#!/bin/sh
VIP=211.238.41.170
ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP up
```

```
route add -host $VIP dev lo:0

/sbin/sysctl -w net.ipv4.conf.all.arp_ignore=1
/sbin/sysctl -w net.ipv4.conf.all.arp_announce=2
/sbin/sysctl -w net.ipv4.conf.lo.arp_ignore=1
/sbin/sysctl -w net.ipv4.conf.lo.arp_announce=2

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

lvs-r.sh 의 퍼미션에 실행 권한을 준다.

```
# chmod 755 /etc/ha.d/lvs-r.sh
```

/etc/rc.d/init.d/puck 을 수정한다.

```
/etc/rc.d/init.d/puck ------------------------------------------
case "$1" in
start)
        if [ -x $IPVSADM ]
            then
#
# ipvs system 설정 부분
#
            echo 1 > /proc/sys/net/ipv4/ip_forward
            ifconfig eth0:0 211.238.41.170 netmask 255.255.255.255 broadcast 211.238.41.170 up
            route add -host 211.238.41.170 dev eth0:0
            /sbin/sysctl -w net.ipv4.conf.all.arp_ignore=0
            /sbin/sysctl -w net.ipv4.conf.all.arp_announce=0
            /sbin/sysctl -w net.ipv4.conf.eth0.arp_announce=0
            /sbin/sysctl -w net.ipv4.conf.eth0.arp_announce=0
            $IPVSADM -A -t 211.238.41.170:80 -s wlc
            $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.165:80 -g -w 100
            $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.166:80 -g -w 100
#
# 추가되는 real_server 는 아래에 같은형식으로 추가하면 된다.
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.165 -g -w 2
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.166-g -w 2
```

```
#

        echo -n "started loadbalancer daemon:"
        daemon   $MON -f -c /etc/mon/mon.cf
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && touch /var/lock/subsys/punk
        echo
    fi
;;
stop)
    if [ -x $IPVSADM ]
        then
        echo -n "puck daemon stopping..."
        $IPVSADM -C

        ######## HA 설정 추가 내용 ###################################

        ifconfig eth0:0 down
        /etc/ha.d/lvs-r.sh

        ############################################################

        killproc mon
        rm -f /var/lock/subsys/punk
        killall http.monitor
```

그런 후에 /etc/rc.d/init.d/heartbeat 파일에 한 줄만 추가해 준다.

/etc/rc.d/init.d/heartbeat 수정 내용 ----------

```
.

        case "$1" in
          start)
```

```
                StartHA
                RC=$?
        .
        .


          stop)
            RunStartStop "pre-stop"
            StopHA
            RC=$?
            Echo
                if
                  [ $RC -eq 0 ]
                then
                  rm -f $LOCKDIR/$SUBSYS
                fi
                RunStartStop post-stop $RC

        ######### Slave-HA 추가 사항 ######################

                /etc/ha.d/lvs-r.sh

        ######################################################

            ;;
```

여기까지가 설정의 모든 것이다.

이제 Master_LB 와 Slave_LB 의 heartbeat 데몬을 가동시킨다.

/etc/rc.d/init.d/heartbeat start

이제 High Availability 가 이루어 지는지 테스트를 해보자.

A. 정상적으로 작업분배가 이루어 지는지 테스트를 한다.

B. Master_LB 의 Lan 선을 뽑아 버린다. 그럼..Slave_LB 가 Master_LB의 권한을 무사히 위임 받는지
를 확인한다.
C. 다시 Master_LB 를 살려 보도록 하자. 그럼..Slave_LB 가 Master_LB의 역할을 다시 반납하고 다시 real 서버로 돌아가는지를 확인한다.
D. 다른 노드들도 각각 Lan선을 뽑고 난뒤에도 서비스에 지장이 없는지를 확인 하도록 한다.

위의 4가지 테스트가 모두 성공적으로 완료 된다면 heartbeat 와 mon 을 이용한 고 가용성 시스템 구축이 완성 된 것이다.

마지막으로 서버 환경에 맞추어서 heartbeat 와 mon 의 데몬 감시 주기 시간을 조절하면 된다.