제 3 장 고가용성 클러스터 시스템

고가용성(High Availability)은 하드웨어, 소프트웨어, 네트워크 등에 문제가 발생하더라도 기존의 작업을 계속 수행하며 서비스를 지속할 수 있도록 해주는 기술을 말한다[10]. 고가용성은 정보화 시대의 도래, 전자 상거래의 발전 및 정보 및 멀티미디어 서비스 등의 컴퓨팅 환경의 변화에 따라 점차 중요시되고 있다. 정보화 사회를 이끌고 있는 여러 가지 인터넷 산업 중에서도 그 기반을 형성하고 있는 컴퓨터 산업은 인터넷 서버 분야라 할 수 있다. 인터넷 서버는 E-business, 전자상거래(EC), 콘텐츠, 인터넷 데이터센터(IDC), 인터넷 서비스공급(ISP)등은 IT시장의 핵심을 이루고 있으며, 통신 기기, 가전/멀티미디어 기기, 자동화/제조 기기 등 다양한 응용분야에서 인터넷을 통한 웹 응용 프로그램을 활용하기 위한 가장 기본적인 요소이다. 따라서 24시간 일년 내내 서비스를 제공해주어야 하는 네트워크 컴퓨팅 부분에서 서버의 고가용성은 필수 불가결한 것으로 인식되고 있다.

초고속통신망이 폭넓게 보급되고 인터넷관련 기본 인프라 구축이 확장되면서 인터넷 사용환경이 크게 개선되고 있으며 이에 따라 인터넷 사용인구도 크게 증가하고 있다. 1998년도 3월에 수행된 IDC의 보고서에 따르면, 1998년도에 약 9천7백만명이 인터넷을 사용하고 있으며, 2002년에는 약 3억 2천만명의 인터넷 인구가 존재할 것으로 예측했다. 인터넷 사용인구의 폭발적인 증가에 비례하여 인터넷 사용 증가로 인해 예상되는 트래픽(traffic)의 발생은 웹 서버단의 큰 문제를 발생시킬 것이다. 특정 사이트에 사용자가 폭주하여 서비스의 병목현상이 발생하고 웹서버의 처리 불가로 사이트가 다운되는 경우를 자주 접할 수 있다. *Contingency Planning Research*에서 제공하는 연구에 의하면, 평균적인 시스템의 중단으로 기업체들은 몇 백만 달러의 손해를 입는다고 나와 있다[1]. 따라서 웹 서버의 접속용량 및 작업처리 속도의 향상은 전체 서비스의 질을 높이고 사용자를 만족시키는데 아주 중요한 의미를 가지게 된다. 일반적으로 기존의 단일 웹서버에서는 고가·고성능의 시스템을 구축하여 웹서버에서 발생될 수 있는 이러한 문제들을 해

결하고 있으나, 급격한 접속 수에 따른 서버의 확장성 및 고가용성 확보가 어려운 실정이다. 고 가용성의 기본적인 목표는 적당한 가격에 고장 포용 시스템을 제공하는 것이다. 고가용성의 기본적인 방법은 컴퓨터 시스템 여분을 이용하여 SPOF (Single Point of Failure)가 발생하는 경우에 이 여분의 요소로 대체하는 것이다. 이를 위해 리눅스 클러스터링 기술이 고가용성을 보장할 수 있는 시스템으로 폭넓게 개발되고 있다. 이러한 가상 서버 구축 방법은 2장에서 소개하였다.

3.1 고가용성 클러스터 시스템 개요

클러스터 가상서버 시스템의 고가용성을 위하여 여러 가지 고가용성 기법들이 제안되고 있다. High-Availability Linux Project[11], Linux Virtual Server Project[4], Linux Scalability Project[12] 등 수많은 분야에서 이러한 고가용성 기법들이 공개적으로 개발되고 있으며, 이를 위한 Monitoring 기법과 Administration 소프트웨어들도 다양하게 개발되고 있다. 본 논문에서는 여러 가지 고가용성 기법들 중에서도"MON"[13], "Heartbeat"[11], "Fake"[14], "Coda"[15] 소프트웨어를 이용하여 고가용성 가상 서버를 구축하는 과정을 설명한다. 고가용성(High Availability) 클러스터 시스템의 구성도는 다음과 같다.

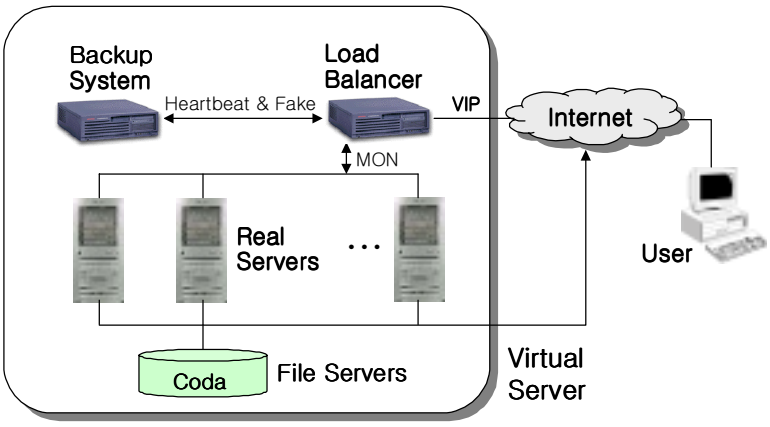

그림 3.1 고가용성 가상서버 구성도
Fig. 3.1 Architecture of High Availability Virtual Server

MON은 네트워크 서비스 이용, 실제 서버의 상태, 그 밖의 환경적 문제 등 각각의 서버에서 제공하는 서비스의 가용성을 모니터링하는 시스템이다. 부하분산서버는 MON을 이용하여 클러스터를 구성하는 모든 실제서버들의 부하상태를 파악하고 데몬의 동작여부를 파악하여 고장이 발생한 서버는 서비스 요청을 할당하지 않고 그 역할을 다른 서버가 대신하게 함으로서 가상서버의 전반적인 가용성을 높인다.

Heartbeat는 시리얼 라인이나 이더넷을 통하여 UDP 신호(heartbeat)를 주고받음으로서 두 서버의 상황을 확인한다. 만약 주기적으로 전송되어야 할 신호가 계속해서 얼마동안 전송되지 않으면 백업 시스템은 주 시스템에 장애가 발생한 것으로 간주하여 자신이 부하분산서버의 역할을 수행하게 된다. 백업 시스템이 부하분산서버의 역할을 수행할 수 있도록 하기 위해서 Fake를 사용한다.

Fake는 ARP spoofing을 이용해 IP를 이전하는 소프트웨어이다. 다시 말하면, 부하분산서버가 165.132.129.24의 가상 IP를 가지고 서비스를 하다가 부하분산서버에 고장이 발생하여 Heartbeat 신호발생이 중지되면 백업 시스템은 부하분산서버의 가상 IP인 165.132.129.24를 넘겨받아서 서비스를 계속해서 수행하게 된다. 가상서버를 구성하는 여러 대의 서버들 중에서 부하분산서버에 고장이 발생하여 그 역할을 수행할 수 없게 되면 가상서버 전체가 그 역할을 수행 할 수 없기 때문에 부하분산서버에서의 SPOF (Single Point of Failure)를 방지하기 위하여 Heartbeat 과 Fake를 이용하여 가상서버 전체의 고가용성을 보장한다.

Coda는 AFS(Andrew File System)를 기반으로 하여 현재 Carnegie Mellon 대학의 M. Satyanarayanan 팀에 의해 개발되고 있는 Advanced Networked File System으로 분산된 많은 서버들의 자원을 효과적으로 관리하고 자료의 안정성을 보장함과 동시에 네트웍 단절이나 서버의 장애 발생시에도 유연하게 대처할 수 있는 고장 포용(Fault Tolerance) 파일 시스템이다. 특히, Coda의 고장 포용(Fault Tolerance) 기법과 고가용성 보장 등의 내용에 대해서 4장에서 집중적으로 분석하여 소개 할 것이며, 5장에서는 이를 바탕으로 실제 구축된 고가용성 클러스터 웹 서버에 대해서 소개하도록 하겠다.

3.2 실제서버의 장애 극복(Fail-Over)

MON은 서버에서 제공하는 서비스의 가용성을 모니터링 하는 기능을 한다. MON은 모니터링과 액션을 계속해서 유지할 수 있도록 설계되었으며, 수행하는 작업을 크게 두 가지로 나누어볼 수 있다. 하나는 서비스의 상태를 모니터링 하는 것으로 이것을 'Monitors'라고 한다. 또 하나는 모니터링 된 정보를 바탕으로 어떠한 적절한 경고 작업을 실행하는 것으로 이것은 'Alerts'라고 한다. 이는 스케줄러로써 구현되고, 상태 모니터링과 경고 작업은 필요한 경우에 추가할 수 있다[13-16]. MON은 다음과 같은 요소들로 구성된다.

- **Monitors** : "Monitors"는 특정 상태를 체크하고, 특정 출력장치를 통해 서버의 성공 또는 실패를 보고하는 프로그램이다. MON과는 독립적이기 때문에 새로운 서비스의 테스트를 추가하려면, 해당되는 monitor 스크립트를 작성해서 적당한 디렉토리에 삽입해 주면 동작한다.
- **Alerts** : "Alerts"는 MON이 실패를 감지했을 때, 메시지를 보낸다거나, 그 밖의 다른 행동을 취하는 스크립트이다. Monitors와 마찬가지로 MON에 대해 독립적이기 때문에 적당한 스크립트를 쉽게 추가할 수 있다.

MON은 클라이언트/서버 형태로 구성되어 작동하며, 클라이언트와 서버는 각각 다음과 같은 역할을 수행하다.

- **Server** : 실제적인 모니터링과 경고 작업이 행해지는 부분이다. MON스케줄러가 시작하면, monitor에 필요한 서비스들을 결정하기 위해 환경 파일을 읽어 들인다. 그 후 스케줄러는 루프를 돌면서 클라이언트 연결, monitor 작동, 그리고 실패에 대한 경고 작업을 수행한다. 각 서비스들은 환경파일에 설정되어 있는 정해진 타이머를 가지고 있고, 이는 스케줄러가 monitor 프로세스를 얼마나 자주 실행해야 하는지를 알려준다.
- **Client** : MON 클라이언트는 MON 스케줄러에 특정 명령을 내려서 원하는 정보를 얻을 수 있도록 할 수 있다. 예를 들면 특정 호스트나 호스트 그룹, 서비스들에 대해 활성화(enabling)와 비활성화(disabling)를 명령할 수 있고,

상태를 저장하거나 로드(load)하거나, 서버를 리셋(reset)할 수도 있다. 클라이언트는 여러 기반에서 동작할 수 있기 때문에 원하는 클라이언트를 직접 만들어서 사용하는 것도 가능하다.

MON 서버를 데몬의 형태로 구동시키면 MON은 정해진 설정파일(mon.cf)을 읽어들인 후, 설정파일에 정의되어 있는 여러 가지 옵션에 따라 작업을 주기적으로 수행하고, 모니터링하고 있는 서비스의 상태에 따라 적절한 경고 작업을 수행하게 된다. 또한 모니터링하고 있는 현황을 콘솔상의 텍스트 형태나, 웹 상의 CGI 형태의 모니터링 툴을 이용하여 관리자에게 제공한다.

그림 3.2 는 클러스터를 이루는 실제서버들 중에서 임의의 실제서버에서 고장이 발생하였을 경우, 고장난 실제서버를 MON 데몬이 발견하여 부하분산서버가 부하분산 라우팅 테이블에서 삭제하여 고장난 실제서버로는 서비스 요청을 분배하지 않도록 하는 과정을 나타내고 있다. 또, 고장났던 실제서버가 다시 복귀되면 부하분산서버가 다시 복귀된 실제서버를 라우팅 테이블에 추가하여 서비스 요청을 분배하도록 되어있다.

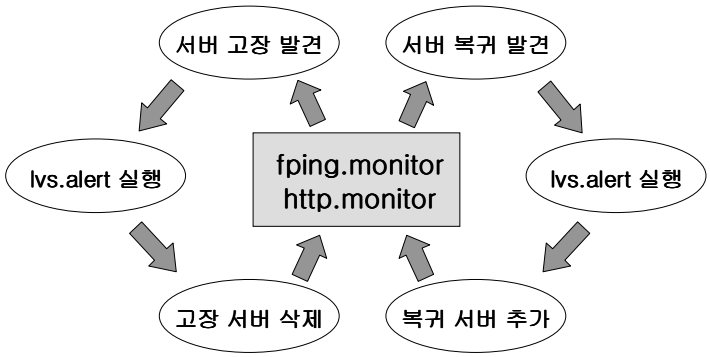

그림 3.2 MON의 실제서버 관리 모델
Fig. 3.2 Real Server Management Model of MON

여기서 서버들의 웹서버 데몬의 상태를 모니터링하는 프로그램으로는 *http.monitor*라는 스크립트를 사용하였으며, 서버들의 동작여부를 모니터링하는 프로그램으로는 *fping.monitor*라는 스크립트를 사용하였다. 이러한 모니터링 프로그

랩은 60초를 주기로 실행되고 이들 모니터링 프로그램으로부터 임의의 실제서버가 동작하지 않는다는 신호를 받으면 *lvs.alert*이라는 스크립트를 실행시켜 해당 서버를 부하분산 라우팅 테이블에서 삭제하여 고장난 실제서버로는 서비스 요청을 분배하지 않도록 하고, 반대로 고장났던 실제서버가 복귀되거나 새로운 서버가 추가되었을 경우에도 *lvs.alert* 스크립트를 실행시켜 복귀된 실제서버가 서비스 요청을 처리할 수 있도록 하였다. 따라서 클러스터내의 실제서버의 상태에 따라 서비스 요청이 할당되어질 것인지가 자동으로 결정되어 언제나 최적의 상태로 가상서버가 동작하게 되어 고가용성을 보장하게 된다[4,13].

· 경고 작업 스크립트 ***lvs.alert***

```
#!/usr/bin/perl
#
use Getopt::Std;
getopts ("s:g:h:t:I:P:V:R:W:F:u");
$ipvsadm = "/sbin/ipvsadm";
$protocol = $opt_P;
$virtual_service = $opt_V;
$remote = $opt_R;
if ($opt_u) {
    $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 $forwarding");
    } else {
        system("$ipvsadm -a -u $virtual_service -r $remote -w $weight $forwarding");
    }
} else {
    if ($protocol eq "tcp") {
        system("$ipvsadm -d -t $virtual_service -r $remote");
    } else {
        system("$ipvsadm -d -u $virtual_service -r $remote");
    }
};
```

· 설정파일 *mon.cf* 의 일부

```
watch Real_Server_1
    service ping
          interval 60s
          monitor fping.monitor
          period wd {Sun-Sat}
          alert mail.alert jihyunee@yonsei.ac.kr → 서버 failure가 검출되면 e-mail 발송.
            alert lvs.alert -P tcp -V 165.132.129.24:80 -R 165.132.129.26 -W 5 -F dr
            upalert lvs.alert -P tcp -V 165.132.129.24:80 -R 165.132.129.26 -W 5 -F dr
    service http
          interval 60s
          monitor http.monitor
          period wd {Sun-Sat}
          alert lvs.alert -P tcp -V 165.132.129.24:80 -R 165.132.129.26 -W 5 -F dr
          upalert lvs.alert -P tcp -V 165.132.129.24:80 -R 165.132.129.26 -W 5 -F dr
```

3.3 부하분산서버의 장애 극복(Fail-Over)

백업 시스템은 부하분산서버와 Heartbeat 신호를 주기적으로 주고받으며 상태를 파악하며 항상 준비상태를 유지하다가, 부하분산서버에서 고장이 발생하여 주기적으로 발생하던 Heartbeat 신호가 전송되지 않으면 Fake를 이용하여 자신이 부하분산서버의 가상 IP를 획득하고 부하분산서버의 역할을 대신하게 된다[11,17].

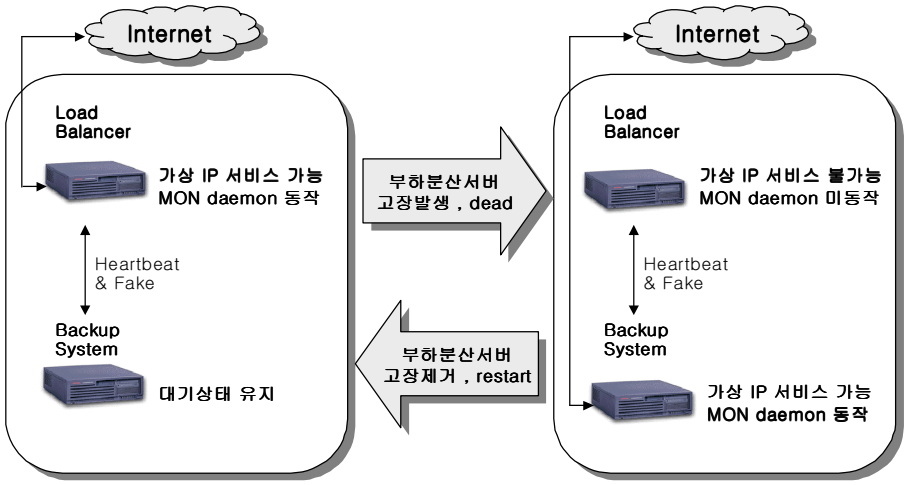

그림 3.3 부하분산서버의 장애 극복
Fig. 3.3 The failover of Load Balancer

그러다가 부하분산 서버가 고장이 제거되어 다시 동작하게 되면 백업 시스템은 다시 대기상태를 유지하게 된다. 이렇게 Heartbeat와 Fake는 상호 긴밀한 관계를 가지고 동작한다. Heartbeat와 Fake는 따로 인스톨하여 사용할 수도 있으나 이미 Fake의 코드는 Heartbeat 패키지에 포함되어 IP Takeover 기능을 가지고 있으므로 별도로 Fake를 설정할 필요는 없다. 본 논문에서도 Heartbeat 패키지에 포함된 IP Takeover 기능을 이용하여 부하분산서버의 SPOF (Single Point of Failure)에 대비한 백업시스템을 구축하였다. Heartbeat를 구성하는 방법으로는 시리얼 라인을 이용한 구성 방법과 LAN을 이용한 구성 방법 등 두 가지가 있다. 이중에서 한가지 방법을 사용하여도 되고, 두 가지 모두를 사용하여도 된다. 본 논문에서는 LAN을 이용하여 Heartbeat를 구성하여 부하분산서버의 고가용성을 보장하였다.

Heartbeat 데몬이 동작하기 위해서는 세 개의 설정파일이 필요하다. 이에 추가적으로 IP Takeover의 기능을 하는 lvs 스크립트와 IP Takeover 가 일어났을 경우에 앞에서 전술한 MON 데몬을 활성화(또는 비활성화)시킬 수 있는 mon 스크립트를 작성하여 부하분산서버와 백업 시스템을 구성하였다.

· 설정파일 ***ha.cf*** 의 일부

```
keepalive 5  → heartbeat 신호를 주고받는 주기
#
deadtime 20  → heartbeat 신호 미발생시 상대 서버가 다운되었다고 결정하는 시간
#
udp     eth0  → heartbeat 신호를 교환할 인터페이스 결정
#
logfile  /var/log/ha-log → heartbeat 신호에 대한 log 파일 설정
#
#       node      nodename ...       -- must match uname -n
node    master_server.yonsei.ac.kr
node    backup_server.yonsei.ac.kr
```

· 설정파일 ***haresources*** 의 일부

```
#       Simple case : One service address, default subnet and netmask
#       No servers that go up and down with the IP address
#
master_server   165.132.129.24   lvs   mon
                ( 가상 IP )        (실행할 스크립트)
```

Heartbeat가 실행중일 때, 만약 감시중인 Master 머신이 dead 된면, Backup 머신의 Heartbeat Daemon은 내부적으로 'start' 라는 string을 생성하여 위에 적은 lvs 와 mon script를 실행한다.

(⇒ /etc/rc.d/init.d/lvs start , /etc/rc.d/init.d/mon start)

dead 되었던 Master 머신이 다시 start up 되면, Backup 머신의 Heartbeat Daemon은 내부적으로 'stop' 이라는 string을 생성하여 실행중인 lvs 와 mon script를 중지시킨다.

(⇒ /etc/rc.d/init.d/lvs stop , /etc/rc.d/init.d/mon stop)

- ***/etc/rc.d/init.d/lvs***

  이 스크립트는 Heartbeat Daemon 에 의해 부하분산서버의 고장이 감지되었을 경우, 대기하고 있던 백업 시스템이 VIP를 이어받고, 실제서버들에 대한 Load balancing을 수행하도록 하는 역할을 한다.

```
#!/bin/sh
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH
IPVSADM=/sbin/ipvsadm
case "$1" in
  start)
    if [ -x $IPVSADM ]
    then
    ifconfig eth0 165.132.129.25 netmask 255.255.252.0 broadcast 165.132.131.255 up
    route add -net 165.132.128.0 netmask 255.255.252.0 dev eth0
    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 → 가상 IP는 165.132.129.24 이다.
    /usr/local/apache/bin/httpd start
    fi
    ;;
 stop)
    if [ -x $IPVSADM ]
    then
    $IPVSADM -C
    killall httpd
    fi
    ;;
*)
    echo "Usage: lvs {start|stop}"
    exit 1
esac
exit 0
```

· */etc/rc.d/init.d/mon*

이 스크립트는 Heartbeat 데몬에 의해 부하분산서버의 고장이 감지되었을 경우, 대기하고 있던 백업 시스템이 각각의 실제서버들을 Monitoring 할 수 있도록 하기 위해서 MON 데몬을 실행시키고 중지시키는 역할을 한다.

```
#!/bin/sh
PATH=/bin:/usr/bin:/usr/local/fping-2.2b1:/usr/local/apache/htdocs/mon-0.38.16
export PATH
MON=/usr/local/apache/htdocs/mon-0.38.16/mon
MONDIR=/usr/local/apache/htdocs/mon-0.38.16
case "$1" in
        start)
                if [ -x $MON ]
                then
                $MONDIR/mon -f -c $MONDIR/mon.cf -b $MONDIR
                fi
                ;;
        stop)
                if [ -x $MON ]
                then
                killall mon
                fi
                ;;
        *)
                echo "Usage: mon {start|stop}"
                exit 1
esac
exit 0
```

이제 Heartbeat 데몬을 직접 구동해보고 동작 확인을 위한 log 파일을 확인해 보도록 하겠다. 가상 IP는 165.132.129.24 이며, Heartbeat log 파일의 경로는 /var/log/ha-log 이다. 아래는 backup_server에서 확인한 log 파일이다. 부하분산서버의 LAN 선을 제거함으로서 인위적으로 고장을 발생시켰으며, 이때 백업시스템은 부하분산서버의 고장을 감지하여 'dead'신호를 보내고①, 바로 자신이 부하분산서버의 역할을 수행하기 시작한다.② 이제 부하분산서버의 LAN 선을 다시 연결하여 Heartbeat 신호가 재개되면③ 백업시스템은 부하분산서버의 역할을 중단하고 다시 대기상태로 돌아간다.④

지금까지 MON, Heartbeat, Fake를 이용하여 클러스터 시스템의 고가용성을 보장하기 위한 기본적인 설정에 대해서 알아보았다. 이제 다음 장에서 고장 포용(Fault Tolerance) 네트워크 파일 시스템인 Coda에 대해서 알아보도록 하겠다.

· */var/log/ha-log*

```
heartbeat: info: ************************
heartbeat: info: Configuration validated. Starting heartbeat.
heartbeat: notice: UDP heartbeat started on port 1001 interface eth0
heartbeat: info: Requesting our resources.
heartbeat: warn: node master_server.yonsei.ac.kr: is dead  → ①
heartbeat: INFO: Running /etc/ha.d/rc.d/status status
heartbeat: Taking over resource group 165.132.129.24
heartbeat: Acquiring resource group: master_server.yonsei.ac.kr 165.132.129.24 lvs mon
heartbeat: INFO: Running /etc/rc.d/init.d/mon  start  → ②
heartbeat: INFO: Running /etc/rc.d/init.d/lvs  start  → ②
heartbeat: INFO: Running /etc/ha.d/resource.d/IPaddr 165.132.129.24 start
heartbeat: notice: node master_server.yonsei.ac.kr seq restart 1 vs 251
heartbeat: info: node master_server.yonsei.ac.kr: status up  → ③
heartbeat: INFO: Running /etc/ha.d/rc.d/status status
heartbeat: INFO: Running /etc/ha.d/rc.d/ip-request ip-request
heartbeat: INFO: Running /etc/ha.d/resource.d/IPaddr 165.132.129.24 status
heartbeat: Releasing resource group: master_server.yonsei.ac.kr 165.132.129.24 lvs mon
heartbeat: INFO: Running /etc/ha.d/resource.d/IPaddr 165.132.129.24 stop
heartbeat: IP Address 165.132.129.24 released
heartbeat: INFO: Running /etc/rc.d/init.d/lvs  stop  → ④
heartbeat: INFO: Running /etc/rc.d/init.d/mon  stop  → ④
heartbeat: WARNING: Return code 143 from /etc/rc.d/init.d/mon
heartbeat: info: Heartbeat shutdown in progress.
heartbeat: info: Giving up all HA resources.
heartbeat: info: All HA resources relinquished.
heartbeat: info: Heartbeat shutdown complete.
```