제 5 장 Coda 분산 파일 시스템을 적용한 고가용성 클러스터 웹 서버

지금까지 리눅스 클러스터링 가상서버와 이의 고가용성을 보장하기 위하여 제안된 기법인 MON, Heartbeat, Fake, Coda 에 대하여 설명하였다. 이번 장에서는 지금까지 설명한 모든 고가용성 기법을 이용하여 실제로 고가용성 클러스터 웹 전용 서버를 구축하는 과정을 설명하도록 하겠다. 단, 부하분산서버와 실제서버들의 설정은 제 2 장에서 다루었고, 마스터 부하분산서버와 백업 부하분산서버간의 Heartbeat & Fake 기법과 실제서버들의 모니터링기법을 이용한 고가용성 구현은 제 3 장에서 다루었기 때문에, 본 장에서는 이들 내용에 대해서는 추가적으로 설명하지 않고 Coda 파일 시스템을 이용한 고가용성 클러스터 가상서버 구축에 대해서만 설명하도록 하겠다. 특히, Coda 파일 시스템을 이용한 실제서버들의 자원 공유에 있어서의 고장 포용 기법에 대해서 중점적으로 설명하도록 하겠다. 가상서버 구축에 사용된 5대의 리눅스 머신의 기능을 표 3 에 정리하였다.

표 5.1 가상서버를 구성하는 리눅스 머신
Table 5.1 Linux Machines of Virtual Server

Hostname	IP	In Coda	In Cluster
linuxlab	165.132.129.25	NO	Load Balancer (Master)
autogold	165.132.129.35	NO	Load Balancer (Backup)
c-eisa	165.132.129.28	SCM	Real Server 1
cim	165.132.129.27	nonSCM	Real Server 2
bode	165.132.129.26	nonSCM	Real Server 3
가상 IP	165.132.129.24	외부에서 접속하는 가상서버의 주소	

테스트에 사용한 리눅스 배포판은 WOW Linux 6.1 plus (Kernel Version 2.2.14-5.0) 이다. 리눅스 클러스터와 마찬가지로, Coda 파일 시스템을 사용하기 위해서는 기본적으로 리눅스 커널이 Coda 파일 시스템을 지원할 수 있도록 컴파일 되어 있어야 한다. 실제로 커널 소스를 다운 받아 클러스터 가상서버를 구동시킬 수 있는 기능들과 Coda 파일 시스템을 지원하도록 컴파일 하여 구축하였으나, 여기서 사용한 리눅스 배포판의 커널은 기본적으로 클러스터 기능과 Coda 파일 시스템을 모듈화하여 컴파일 되어 있어서 추가적인 커널 컴파일 과정 없이도 가상서버를 구축할 수 있었다. 또한 커널 컴파일 옵션에 대해서 설명하자면 많은 지면이 필요하여 본 논문에서는 커널 컴파일 과정의 설명은 생략하고 WOW Linux 배포판에 포함되어 있는 기존의 커널을 사용하는 것으로 대신하도록 하겠다.

5.1 파일서버와 분리된 클러스터 웹서버

지금 현재 클러스터 솔루션으로 제공되어 시판되는 제품들의 대부분은 그림과 같이 클러스터 서버와 파일서버를 따로 분리하여 구성하고 있으며, 파일서버의 역할을 하기 위해서 NFS(Network File System)나 SAN(Storage Area Network) 등과 같은 솔루션을 사용하여 가상서버를 구성하고 있다[21,28,29].

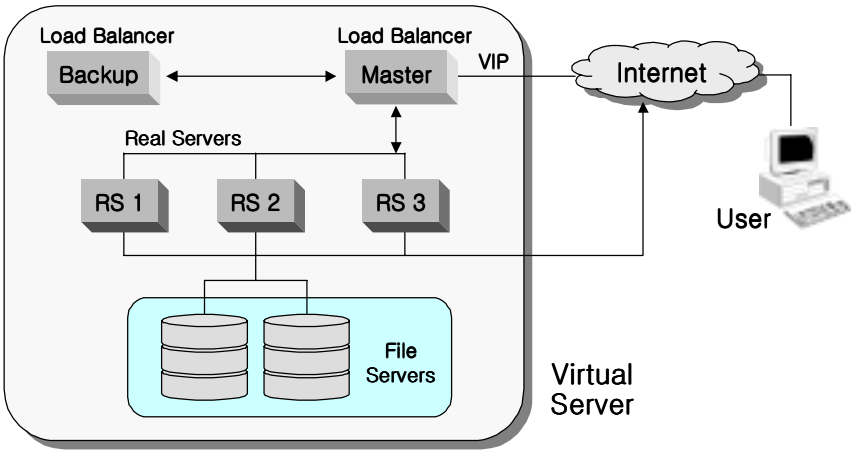

그림 5.1 파일서버와 분리된 클러스터 웹서버
Fig. 5.1 Cluster Web Server differ from File Server

그러나 파일서버를 클러스터 서버들과 분리해서 사용할 경우 파일서버단의 네트워크에서 병목현상이 발생할 수 있어서 전체적인 성능이 저하될 우려가 있으며, 클러스터 서버들과는 별도로 대용량의 파일서버를 따로 구입하여 설치해야 하므로 가상서버 구축비용이 상당히 비쌀 수 있게된다. 또한 NFS 와 SAN 등의 파일시스템에는 클라이언트의 캐시 기능이나 서버 복제 기능 등이 포함되어 있지 않기 때문에 파일서버 고장이나 네트워크 단절 등의 급작스런 시스템 고장이 발생할 경우 데이터가 손실되거나 사라질 수 있는 위험이 있다. SAN 같은 경우는 광채널을 고속의 데이터 전송망을 이용하여 데이터를 공유할 수 있어서 어느 정도의 가용성을 보장하고는 있으나 초기 설치비용이 막대하다는 단점을 가지고 있다[28]. 따라서 다음 절 에서는 클러스터를 이루는 실제서버들의 자원을 Coda 파일시스템으로 구성하여 클러스터 실제서버의 역할을 하면서 동시에 파일서버의 역할도 할 수 있도록 구성하였다.

5.2 Coda Cell에서 동작하는 클러스터 웹서버

그림 5.2 에서 본 논문에서 구현될 Coda 파일 시스템을 적용한 고가용성 클러스터 웹 전용 서버에 대한 구성도가 나타내었다.

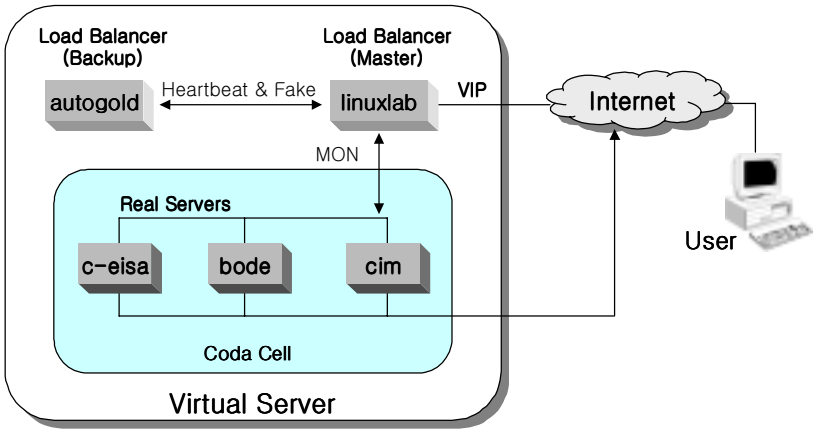

그림 5.2 Coda Cell에서 동작하는 클러스터 웹서버
Fig. 5.2 Cluster Web Server in Coda Cell

그림 5.2 에서 보여지듯이, 클러스터를 이루는 실제서버들은 Coda Cell을 이루며 Coda 서버의 역할을 수행한다. 또한 이와 동시에 실제서버들은 Coda 클라이언트의 역할도 수행하면서 부하분산서버에 의해 서비스 요청을 분배받아 각자 사용자에게 서비스를 제공한다. 만약 하나의 실제서버가 고장을 일으켜 다운되는 경우에는 그 실제서버가 지금까지 수행하고 있던 Coda 서버의 역할, 가상서버를 이루는 실제서버로의 역할을 수행하지 못하게 된다. 그러나, 서버 복제(Replication) 정책에 의해 다른 실제서버가 Coda 서버의 역할을 수행할 수 있으며, 클러스터링 정책과 모니터링 프로그램인 MON을 이용하여 다른 서버가 서비스를 제공할 수 있게 되어 가상서버의 전반적인 고가용성을 보장할 수 있게 된다. 또한 클러스터를 이루는 실제서버들의 여유자원을 공유하여 사용하기 때문에 파일서버를 설치하기 위한 추가비용이나 고속 전송로를 구축하기 위한 비용에 대한 절감 효과도 노릴 수 있다.

5.3 고장포용 기법을 적용한 Coda 서버의 볼륨 구성

고장포용(Fault Tolerance)이란 시스템의 일부 요소에 고장이 일어나더라도 전체 시스템이 잘못된 행동을 일으키기 전에 고장을 회복시키거나 고장의 효력을 제거하여 안전한 작업수행을 보장할 수 있게 하는 기법을 말한다. 초기에는 이러한 고장을 일으키는 문제를 해결하기 위해 고장회피(Fault-Avoidance) 시스템에 대한 고려가 제기되었으나, 경제적인 문제와 제조상의 기술적인 한계 등으로 인해 실용화에 어려움이 있으며, 완벽하게 모든 종류의 고장을 피할 수 있는 시스템을 만드는 것이 현실적으로 불가능하였다. 그래서 디지털 컴퓨터의 급속한 발달과 관련 응용분야의 확대와 함께 고장포용분야에 대한 관심도가 매우 커지고 있다[30].

고장포용기법 설계시의 기본개념은 시스템에 여분(Redundancy)의 자원(Resource)을 제공하는 것이다. 여분은 H/W, S/W 또는 시간으로 구성되며 모든 구성요소가 정상적으로 작동하는 동안에는 여분의 사용에 따른 어떤 성능상의 이득도 시스템에 주지 못한다. 물론 고장포용기법에 사용되는 어떤 여분도 시스템의 설계 및 구현 비용을 증가시키므로 시스템은 많은 여분을 가질 수 없으며 주어진

비용에서 시스템의 신뢰도를 극대화시킬 수 있는 고장포용기법을 사용해야만 한다. H/W로 구성되는 여분시스템은 동일한 작업을 할 N개(≥3)의 프로세서나 모듈을 구성하여 고장이 발생하더라도 고장차폐를 통해서 고장과 관계없이 작업수행을 완결할 수 있도록 하거나 여분모듈(Spare Module)을 추가 설치하여 고장이 발생할 경우에 적절한 스위칭을 통해서 정상적인 작업수행을 도모하는 시스템을 말한다. 본 논문에서는 H/W 여분 시스템을 응용하여 각각의 Coda 서버에 Replication 볼륨을 사용함으로서 임의의 Coda 서버가 고장을 일으켜 작업을 중지하더라도 Replication 서버가 계속해서 그 작업을 수행할 수 있도록 하였다. 또, 그 Replication 서버는 독립적으로 모든 작업을 수행하다가 고장난 Coda 서버가 다시 동작하기 시작하면 그 동안 변경되었던 모든 사항들에 대해서 자동으로 업데이트를 하여 고장 발생 이전의 H/W 여분 상태를 유지하게 된다.

표 5.2 는 클러스터를 구성하고 있는 세 대의 실제서버를 Coda 서버로 동작시키는데 있어서 상호간의 Replication 볼륨을 구축하기 위한 파티션 및 볼륨 정보를 나타내고 있다. 각각의 서버는 Coda 시스템을 위한 저장공간으로서 /vicep□ 로 명명된 파티션을 총 4개씩 보유하고 있으며 하나의 Coda 서버는 각각 다른 두 서버와 두 개씩의 Replication 볼륨을 공유한다. 각각의 볼륨은 클라이언트의 /coda 디렉토리에 하부적으로 마운트되어 사용되며, 하나의 서버가 고장나면 고장난 서버의 역할을 나머지 두 서버가 절반씩 나누어 수행하게 된다.

표 5.2 Coda 서버들의 복제 볼륨
Table 5.2 Replication Volumes of Coda Servers

c-eisa	cim	bode	volume name
/vicep**a**	/vicep**a**		eisa_cim_a
	/vicep**b**	/vicep**b**	cim_bode_b
/vicep**c**		/vicep**c**	eisa_bode_c
/vicep**d**	/vicep**d**		eisa_cim_d
/vicep**e**		/vicep**e**	eisa_bode_e
	/vicep**f**	/vicep**f**	cim_bode_f

5.4 복제 볼륨 마운트

클러스터를 이루는 실제서버들이 표 와 같이 replication 볼륨을 생성하게 되면 이제 Coda 서버로서의 역할을 수행할 준비가 다 된 것이다. 이제 실제서버들은 Coda 클라이언트로서의 역할을 수행하기 위해서 생성된 볼륨을 마운트하여 사용하게 된다. 클라이언트에서는 ***cfs*** *mkmount* 명령을 이용하여 생성된 볼륨을 마운트하게 되는데, 이 명령은 /coda 디렉토리 아래에 마운트하고자 하는 볼륨 이름으로 새로운 디렉토리를 생성한다. 그림 5.3 은 클라이언트 측에서 바라보는 볼륨 마운트에 대한 구조를 나타내고 있다.

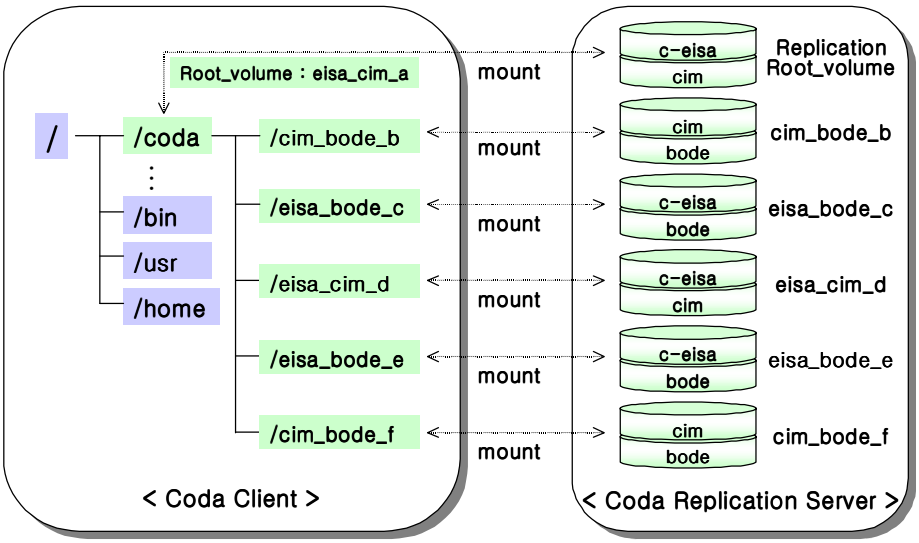

그림 5.3 Coda 클라이언트의 복제 볼륨 마운트
Fig. 5.3 Replication Volumes Mount of Coda Clients

그림 5.3 에서 볼 수 있듯이, 모든 볼륨은 replication 볼륨을 가지고 있어서 하나의 Coda 서버가 동작을 멈추더라도 다른 서버가 Coda 서버의 역할을 대신 수행할 수 있다. Coda Replication 서버가 어떠한 방식으로 동작하는지 간단한 예제를 통하여 알아보도록 하겠다. 위에서 보여지는 여러 디렉토리 중에서 ***/coda/cim_bode_b*** 디렉토리는 Coda 서버 *cim* 과 *bode* 에 의해 replication 볼륨을 통하여 각 서버에 동일한 데이터가 저장된다. Coda 클라이언트로 동작하는

*c-eisa*에서 *vi* 명령을 이용하여 test1 라는 파일을 */coda/cim_bode_b* 디렉토리에 생성하여 보겠다.

```
[root@c-eisa /coda/cim_bode_b]# vi test1

This is test1 file in /coda/cim_bode_b directory.     : wq (파일 저장)

[root@c-eisa /coda/cim_bode_b]# cat test1
This is test1 file in /coda/cim_bode_b directory.

[root@c-eisa /coda/cim_bode_b]#
```

그러면 *test1* 이라는 파일은 Coda 서버 *cim* 과 *bode* 에 함께 저장된다. Coda 서버가 볼륨의 형태로 파일을 저장할 때는 숫자를 이용하여 파일을 구분하여 저장한다. 클라이언트에서 *test1*으로 저장하였던 파일은 각각의 서버에서는 서버의 데이터 저장 디렉토리 */vicepb* 에 */vicepb/0/0/0/1* 과 같은 형태로 저장되어 관리된다. 이제 Coda 서버 *cim* 과 *bode*에서 각각 데이터를 확인해 보면 두 서버 모두에 동일한 데이터가 저장되었음을 알 수 있다.

```
[root@cim /vicepb]# ls
0/    FTREEDB
[root@cim /vicepb]# cd 0/0/0/
[root@cim /0]# ls
1
[root@cim /0]# cat 1
This is test1 file in /coda/cim_bode_b directory.

[root@cim /0]#
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[root@bode /vicepb]# ls
0/    FTREEDB
[root@bode /vicepb]# cd 0/0/0/
[root@bode /0]# ls
1
[root@bode /0]# cat 1
This is test1 file in /coda/cim_bode_b directory.

[root@cim /0]#
```

이러한 Coda 서버의 Replication 볼륨 정책은 replication 서버가 Coda Cell에서 떨어져 네트워크를 형성할 수 없는 경우에도 유용하게 사용된다. 예를 들어, Coda 서버 *bode*를 네트워크로부터 분리하여 인위적인 고장을 발생시켜 서버의 역할을 수행할 수 없도록 하고 Coda 클라이언트에서 /coda 디렉토리 아래에 파일을 생성할 경우, 그 파일은 Coda 서버 *cim*에만 저장되어 있다가 Coda 서버 *bode*가 네트워크 연결이 복원되면 바로 업데이트하여 동일한 데이터를 소유하게 되는 것을 확인할 수 있다. 이와 같이 Replication Coda 서버의 이용으로 Coda 시스템은 돌발적인 시스템 고장으로 인하여 일부 서버의 작동 불능 상태가 발생하더라도 고장에 상관없이 지속적인 서비스를 제공하기 때문에 Coda 파일 시스템을 클러스터 가상서버에 적용하면 보다 높은 고가용성을 보장하는 가상서버를 구축할 수 있을 것이다.