임무컴퓨터를 위한 고가용 시스템의 설계 및 구현

Design and Implementation of High-availability System

정재엽, 이철훈*
충남대학교 컴퓨터공학과

Jae-Yeop Jeong, Cheol-Hoon Lee*
Dept. of Computer Engineering, Chungnam National University

요약

임무컴퓨터는 항공전자시스템에서 전체 시스템을 관리하고, 특정 임무를 처리하는 중요한 역할을 수행한다. 일반적으로 단일 시스템에서 SPOF(Single Point Of Failure) 요소의 고장은 전체 시스템의 고장으로 이어질 수 있으며, 이는 서비스의 중단으로 인한 임무의 실패뿐만 아니라 조종사의 생명까지도 위협할 수 있다. 이에 본 논문에서는 SPOF 요소를 제거하기 위해 단일 시스템을 이중화하여 고장발생에 유연하게 대처하도록 설계하였다. 또한 이를 효율적으로 운영하기 위한 방안으로 리눅스 기반의 Heartbeat, Fake, DRBD(Distributed Replicated Block Device), Bonding 등의 기법을 이용하여 시스템을 관리한다.

Abstract

Mission Computer performs important role both managing a whole system and dealing with a specific mission in avionics system. In general, the fault of SPOF(Single Point Of Failure) in unity system can lead to failure of whole system. It can cause a failure of a mission and also can threaten to the life of the pilot. So, in this paper, we design the HA(High-availability) system so that dealing with the failure. And we use HA software like Heartbeat, Fake, DRBD and Bonding to managing HA system.

I. 서 론

최근 항공 전자장비 개발의 추세를 보면 디지털화가 가속되면서 이를 전체적으로 관리하기 위한 임무컴퓨터의 역할이 중요시되고 있다. 임무컴퓨터는 항공기의 임무 수행에 필요한 각종 전술 데이터의 처리, 비행 데이터, 항법정보, 영상처리, 관리 및 융합 등의 기능을 수행한다. 이러한 중요 시스템의 오동작이나 붕괴는 임무의 실패뿐만 아니라 조종사의 생명까지도 위협할 수 있다. 서비스의 중단은 계획된 중단과 갑작스런 중단으로 구분할 수 있다.

[표 1] 서비스 중단의 구분

계획된 중단	백업, 업그레이드, 유지보수 등 여러 가지 계획에 의해 서비스가 중단되는 경우
갑작스런 중단	정전, 하드웨어 및 소프트웨어적인 오류, 해킹, 사용자 오류, 자연재해 등의 원인으로 서비스가 중단되는 경우

서비스의 실패를 유발하는 하드웨어나 소프트웨어의 단일 결함 요소를 SPOF라 하며, 고가용 서비스를 위해서는 SPOF를 제거하여야 한다. 이를 위한 가장 간단한 방법은 문제가 발생할 수 있는 부분을 이중화하는 것이다. 본 논문에서는 단일 시스템의 이중화를 통해 SPOF 요소를 제거하고, 이를 효율적으로 관리하기 위해 리눅스 기반의 고가용 시스템을 설계하고 구현한 내용을 기술한다. 2장에서는 관련연구에 대해 살펴보고, 3장에서는 리눅스 기반의 고가용 시스템 설계와 구현을 다룬다. 4장에서는 실험환경 및 실험 결과를, 마지막으로 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

II. 관련연구

1. Linux HA

Linux-HA 프로젝트는 신뢰성(Reliability), 가용성(Availability), 내구성(Serviceability)을 위한 고가용성 솔루션을 제공한다. Heartbeat, Fake, Ipfail, DRBD[3]등의 소프트웨어를 이용하여 시스템에 고가용성을 보장하고 있으며, 이를 이용하여 데이터베이스, 웹, LVS, 메일, DNS, DHCP, Proxy Caching 서버를 구성할 수 있다[1]. 또한 Ultra Monkey[4], MON[5], Nagios[6], Clumon[7], Ganglia[8] 등의 클러스터 모니터링 시스템과도 밀접하게 연구가 진행되고 있다.

Linux-HA 프로젝트의 핵심 프로그램은 Heartbeat으로 시리얼(serial line)이나 UDP통신을 이용하여 서버의 결함을 검

사한다. 결함이 발생하여 노드의 동작이 불가능할 경우 해당 노드의 모든 작업을 대기하고 있던 노드로 모두 이전하여 시스템이 지속적으로 서비스를 수행하도록 함으로써 고가용성을 보장한다[1].

Fake는 ARP spoofing을 사용하여 결함이 발생한 서버의 주소를 다른 서버로 인계하여 지속적으로 처리를 가능하게 하며, Ipfail은 Heartbeat 프로그램에 plug-in 형태로 동작하여 네트워크를 지속적으로 감시하는 역할을 수행한다. DRBD는 각 시스템의 로컬디스크를 동기화 하는 기법으로 TCP/IP를 이용하여 실시간으로 데이터를 동기화 한다.

이렇게 구성된 고가용 시스템은 클러스터로 구성하여 관리하며, 백업 시스템의 운영방안은 Active/Standby와 Active/Active 두 가지 방식으로 나눌 수 있다[9].

Active/Standby 방식은 각 노드가 primary/backup으로 동작하며, primary 노드에 고장이 발생할 경우에만 backup 노드가 primary 노드의 모든 서비스를 이전받아 수행한다. 구현은 쉽지만 backup 노드가 아무런 처리를 하지 않으므로 성능대비 가격이 높다.

Active/Active 방식은 두 노드가 모두 각자의 서비스를 제공하다가 하나의 노드에 고장이 발생하면, 살아있는 다른 노드가 고장이 발생한 노드의 서비스를 모두 담당한다. 이는 고장이 발생한 노드의 작업을 모두 수행해야 하므로 과중한 오버헤드로 인한 이차적인 고장이 발생할 수 있다[1][2].

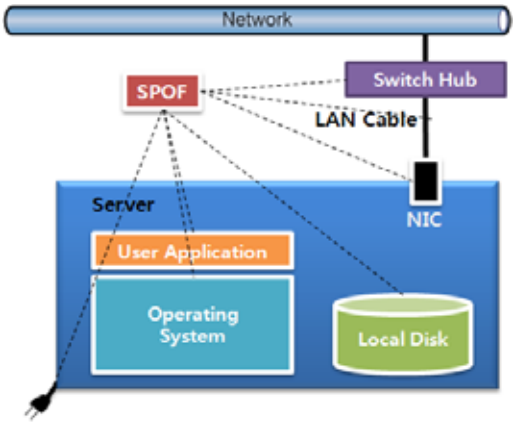
2. Bonding

Bonding 기법은 물리적으로 실제 존재하는 다수의 NIC (Network Interface Card)를 하나의 가상 NIC로 구성함으로써 네트워크를 이중화시킬 수 있는 방법이다. 설정에 따라 다수의 NIC로 들어오는 데이터를 로드밸런싱할 수 있으며, 고가용성을 제공하기 위해 Active/ Standby로 구성하여 Active NIC가 작업을 수행하다가 장애가 발생할 경우 Standby NIC가 계속해서 작업을 수행하도록 구성할 수 있다[10].

Ⅲ. 고가용 시스템의 설계 및 구현

1. 단일 시스템의 SPOF

매우 신뢰적인 단일 시스템이라도 SPOF를 갖는다. 단일 시스템에서는 SPOF 요소의 고장으로 인해 시스템이 붕괴될 수 있으며, 이는 서비스의 중단으로 이어져 상업적 시스템에서는 기업의 경제적 손해뿐만 아니라 사용자에게 불편함을 주고, 군용 시스템에서는 임무의 실패와 사용자의 생명까지도 위협할 수 있다.



▶▶ 그림 1. 단일 시스템의 SPOF

SPOF는 그림 1에서와 같이 네트워크, 운영체제, 전원 등 시스템의 다양한 부분에서 나타날 수 있으며, 표 2의 방법으로 이를 해결할 수 있다.

[표 2] SPOF 요소 제거

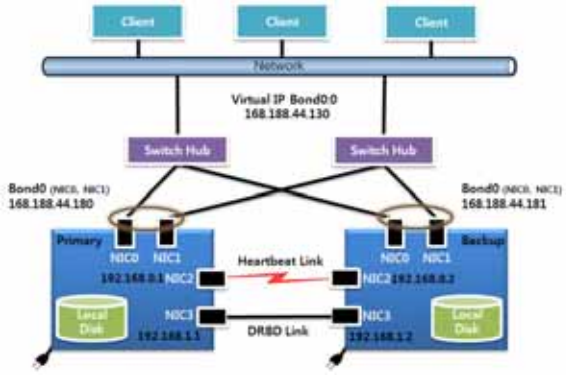
단일 컴포넌트	고장 제거 방법
CPU	- Backup CPU 제공
LAN, NIC	- 중복 NIC설치 - Stand-alone 인터페이스 구성
DISK	- RAID 기법 적용
전원	- 시스템에 UPS추가 - 전력원 추가 사용
운영체제	- 재시작과 복구를 위한 failover 기능
애플리케이션	- 자동으로 재시작하기 위한 기능제공 - 복구하기위한 기능 제공 - 코드 디버깅을 통하여 제공

2. 이중화를 통한 고가용 시스템의 설계 및 구현

단일 시스템의 모든 SPOF 요소를 제거하여 전체 시스템에 고가용성을 제공하기 위해 그림 2와 같이 시스템을 구성하였으며, 각 노드는 리눅스 기반의 Heartbeat, Ipfail, Bonding, DRBD등의 고가용 소프트웨어에 의해 관리된다.

1. 시스템 이중화 및 동작방식

시스템의 이중화를 위해 기존의 단일 시스템과 동일한 구성을 가진 시스템을 추가로 두어 고장에 대비할 수 있도록 한다. 두 개의 시스템은 클러스터로 구성되어 관리되며, 백업 시스템의 운영방안은 Active/Standby 방식을 사용한다.



▶▶ 그림 2. 고가용 시스템의 구성

시스템에서 가장 중요한 프로그램은 Heartbeat으로 표 3의 파일에 의해서 관리된다.

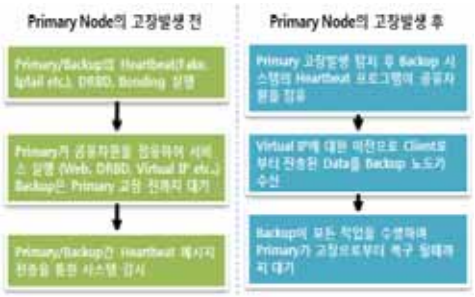
[표 3] Heartbeat 설정 파일

이름	기능
ha.cf	keepalive - 메시지전송주기 deadtime - 노드 실패의 판단시간 bcast - 메시지를 전송할 인터페이스
haresource	두 노드간의 공유자원 설정
authkeys	노드간 인증방법 설정

시스템이 처음 시작될 때 Heartbeat 프로그램은 haresource 파일에 등록된 DRBD, Virtual IP, Web, DB 등의 공유자원에 대해 그 모든 권한을 얻어 서비스를 실행한다. 또한 ha.cf 파일에 설정되어있는 keepalive 시간을 주기로 NIC2를 이용하여 Heartbeat 메시지를 전송하고, 설정된 deadtime 이내에 메시지가 수신되지 않으면 시스템에 고장이 발생한 것으로 판단하여 backup 노드로 이전(takeover)이 발생한다.

정상동작 시에 클라이언트로부터 전송되는 데이터는 Fake에 의해 생성된 가상IP (bond0:0 168.188.44.130)를 통해서 전송되며, 전송된 데이터는 primary 노드로 전송되어 이를 처리하게 된다. 또한 로컬디스크에 저장되는 데이터는 NIC3을 통해 실시간으로 동기화된다.

그림 3은 primary 노드의 고장발생 전과 고장발생 후의 전체적인 동작이다.



▶▶ 그림 3. 시스템의 동작방식

2. 네트워크 이중화 및 동작방식

본 논문에서 구성한 고가용 시스템에서는 Bonding 기법을 사용하여 네트워크를 이중화하고, Heartbeat 프로그램의 Ipfail 데몬을 이용하여 네트워크를 감시한다. 또한 두 개의 스위치 허브를 이용하여 허브자체의 고장에도 대비할 수 있도록 하였다.

Bonding은 리눅스 내부에 구현되어 있는 데몬으로 지정된 타깃(Target)에 ARP 패킷을 일정한 주기로 전송하고 그에 대한 응답으로 NIC또는 LAN Cable의 고장 여부를 파악할 수 있다.

```
alias bond0 bonding
options bond0 miimon=0
arp_interval = 20
arp_ip_target = 168.188.46.1
mode = 1
primary = eth0
```

▶▶ 그림 4. modprobe.conf 파일의 Bonding 설정

그림 4는 Bonding의 설정으로 NIC0와 NIC1을 하나의 IP로 묶어 관리하며, primary 이더넷을 NIC0로 설정하여 모든 패킷을 NIC0가 처리하다가 LAN Cable이나 NIC0에 고장이 발생하면 NIC1이 모든 작업을 이전받아 처리하게 된다.

Heartbeat의 plug-in 형태로 동작하는 Ipfail 데몬은 외부에 연결된 네트워크를 감시한다. 지정된 타깃에 ping echo request를 보내고, 그에 따른 응답으로 네트워크를 지속적으로 감시한다. ping 메시지의 전송주기는 ha.cf파일에서 설정했던 keepalive 시간을 이용하여 전송하며 deadtime 이내에 응답이 오지 않을 경우 네트워크에 고장이 발생한 것으로 판단하여 backup 노드로 이전이 발생한다.

3. 디스크 저장소 이중화

디스크 저장소의 특성에 따라 시스템은 여러 가지 구성으로 변경할 수 있다. 공유저장소를 사용하게 되면 두 노드가 저장소를 공유하여 데이터를 처리하게 되고, RAID방식을 사용하여 해당 저장소에 대한 미러 디스크를 만들어 관리할 수 있다.

본 논문에서는 각 노드에 로컬디스크를 두고 DRBD 기법을 이용하여 노드간의 데이터를 실시간으로 동기화 시킨다. DRBD는 TCP/IP프로토콜을 이용하여 동기화할 데이터를 전송하며, 공유데이터를 저장할 장소와 IP 및 Port번호 등을 drbd.conf 파일에서 설정할 수 있다. 그림 5는 drbd.conf 파일의 설정이다.

```
resource share{
    protocol C;

    on sslab1 { // Primary node
        device      /dev/drbd0;
        disk        /dev/hdc6;
        address     192.168.1.1:7788;
        meta-disk   /dev/hdc5[0];
    }

    on sslab2 { // Backup node
        device      /dev/drbd0;
        disk        /dev/hda6;
        address     192.168.1.2:7788;
        meta-disk   /dev/hda5[0];
    }
}
```

▶▶ 그림 5. drbd.conf 설정 파일

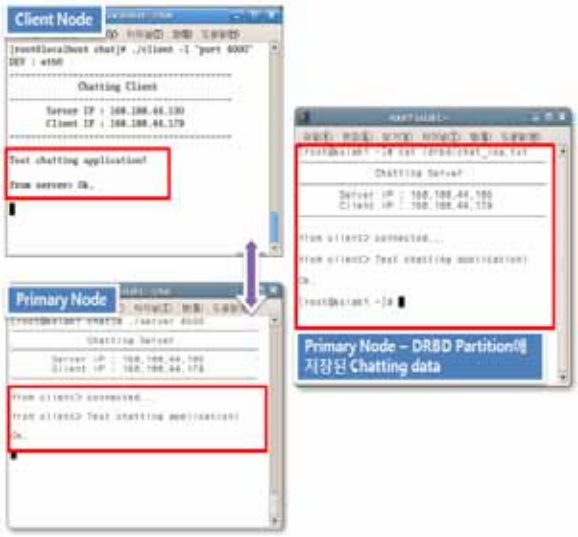
Ⅳ. 실험 환경 및 결과

본 논문에서 구현한 임무컴퓨터를 위한 고가용 시스템에서는 리눅스 운영체제를 기반으로 표 4와 같은 서버 시스템을 사용하였다. 네트워크는 최대 100Mbps까지 지원하는 LAN Cable과 NIC를 사용하였다.

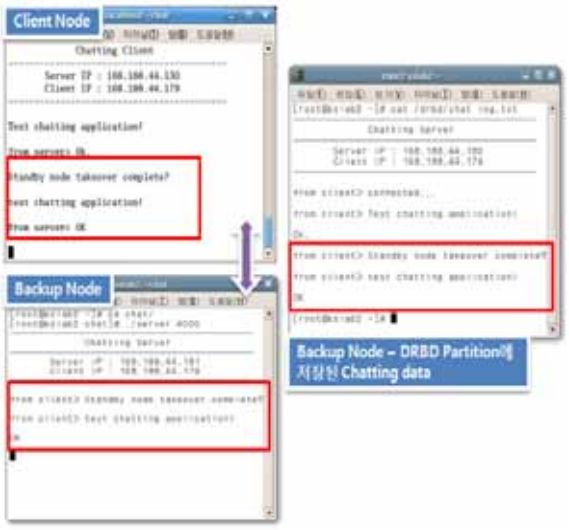
[표 4] 실험 서버환경

컴포넌트	특 징
CPU	Intel Pentium IV 3.00GHz
Cache Size	512KB
RAM	512MB
NIC	10Mbps and 100Mbps
LAN Cable	100Mbps

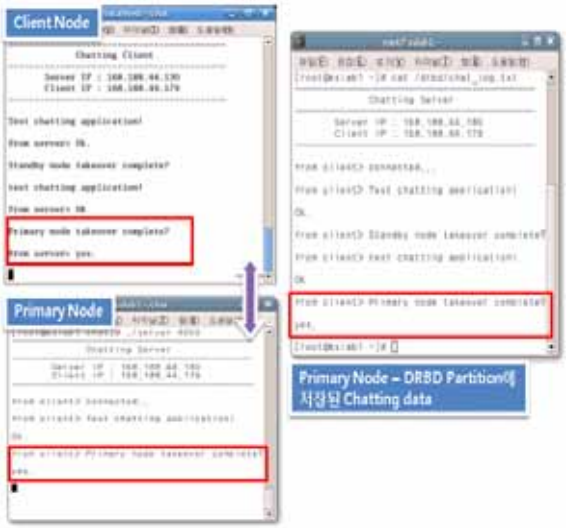
채팅 애플리케이션을 이용하여 클라이언트와 서버가 통신을 하며, 채팅 내용은 DRBD 파티션에 저장되어 두 노드가 공유할 수 있도록 구성하였다. 그 후 임의로 고장을 발생시켜 서버의 동작을 확인하였다.



▶▶ 그림 6. primary 노드 고장발생 전 채팅 프로그램



▶▶ 그림 7. primary 노드 고장발생 후 채팅 프로그램



▶▶ 그림 8. primary 노드 고장복구 후 채팅 프로그램

그림 6에서는 고장 발생 전의 상황으로 클라이언트(168.188.44.179)와 primary 노드(virtual IP : 168.188.44.130, Real IP : 168.188.44.180)간에 채팅이 이루어지며, 채팅 내용은 공유저장소인 DRBD 파티션의 chat_log.txt 파일에 저장된다.

그림 7에서 primary 노드에 고장이 발생하면 모든 공유자원이 backup 노드로 이전되며 모든 자원을 이전받은 후 클라이언트와 backup 노드(Virtual IP : 168.188.44.130, Real IP : 168.188.44.181)간에 채팅이 이루어진다. 또한 모든 채팅 내용은 backup 노드에 있는 로컬디스크에 저장된다.

그림 8은 primary 노드가 고장으로부터 복구된 후의 상태로써, 클라이언트와 primary간 통신이 다시 이루어지며, backup 시스템에서 저장하던 모든 내용은 primary의 공유저장소에 다시 동기화 된 것을 확인할 수 있다.

이와 같이 primary 노드에 고장이 발생하여 primary노드가 임무를 수행하지 못할지라도 backup 시스템을 이용하여 시스템에 고가용성을 제공함으로써, 고장으로부터 유연하게 대처할 수 있다.

V. 결론 및 향후 연구과제

본 논문에서는 이중화를 통하여 단일시스템에서 시스템 붕괴의 원인이 되는 SPOF 요소를 제거하였다. 또한 이중화된 시스템을 클러스터로 관리하고, 고가용성을 제공하는 여러 소프트웨어 기법을 적용하여 고가용 시스템을 구축하였다.

향후 연구과제로는 primary 노드에서 backup 노드로 이전이 발생할 때 디스크, 메모리, CPU 등의 자원을 효율적으로 복구할 수 있는 방법이 필요하며, 고가용 시스템의 중요부분인 성능측정부분에 대한 연구가 진행되어야 한다.

▌참 고 문 헌▐

[1] Alan Robertson, "The Evolution of The Linux-HA project", UKUUG, 2004
[2] High-availability Linux Project, http://linux-ha.org/
[3] DRBD, http://linbit.com, http://drbd.org/
[4] Ultra Monkey, http://www.ultramonkey.org/
[5] MON, http://mon.wiki.kernel.org/index.php/Main_Page
[6] Nagios, http://netsaint.sourceforge.net/
[7] Clumon, http://clumon.ncsa.uiuc.edu/
[8] Ganglia, http://ganglia.info/
[9] Alan Robertson, "Linux-HA Heartbeat System Design", USENIX, 2000
[10] Linux Ethernet Bonding Driver HOWTO, http://www.redhat.com/