임무컴퓨터를 위한 고가용 시스템의 구현 및 성능분석

Implementation and Performance Analysis of High-availability System for Mission Computer

정재엽, 박성종, 임재석, 이철훈
충남대학교 컴퓨터공학과

Jae-Yeop Jeong(iwis023@cnu.ac.kr), Seong-Jong Park(prime@cnu.ac.kr),
Jae-Seok Lim(jslim@cnu.ac.kr), Cheol-Hoon Lee(clee@cnu.ac.kr)

요약

임무컴퓨터는 항공전자시스템에서 임무 수행에 필요한 각종 전술데이터 처리, 영상처리, 항법정보의 관리 및 융합 등의 매우 중요한 기능을 수행한다. 이러한 중요 시스템이 단일시스템으로 구성되면, 여러 가지 SPOF(Single Point Of Failure) 요소의 고장으로 인해 전체 시스템의 고장으로 이어질 수 있다. 이는 서비스 중단으로 인한 임무의 실패뿐만 아니라 조종사의 생명까지도 위협할 수 있다. 본 논문에서는 단일 시스템의 이중화를 통해 SPOF 요소를 제거하고, 이를 운영하기 위한 방안으로 리눅스 기반의 Heartbeat, Fake, DRBD(Distributed Replicated Block Device), Bonding 등의 기법을 이용하여 고가용 시스템을 구현하였다. 또한, 구현한 고가용 시스템에서 빠른 고장 탐지를 위한 FDT(Fault Detection Time)와 고장 발생 시 임무 연속성을 위해 중요한 요소인 MTTR(Mean Time To Repair)의 평균값을 측정하고, 그에 따른 성능분석 결과를 제시한다.

■ 중심어 : | 임무컴퓨터 | 고가용 시스템 | Heartbeat | SPOF |

Abstract

MC(Mission Computer) performs important function in avionics system which tactic data processing, image processing and managing navigation system etc. 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(Hight-availability) system so that dealing with the failure. And we use HA software like Heartbeat, Fake, DRBD and Bonding to manage HA system. Also we analyze the performance of HA system using the FDT(Fault Detection Time) for fast fault detection and MTTR(Mean Time To Repair) for mission continuity.

■ keyword : | Mission Computer | High-availability System | Heartbeat | SPOF |

I. 서 론

최근 항공 전자장비의 개발의 추세를 보면 미사일, 센서, 레이더, 항법정보 등 항공기의 임무 수행에 필요한 각종 전술 데이터양이 많아짐에 따라 이를 처리하는 임무컴퓨터의 중요성이 증가하고 있다. 이러한 중요 시

접수번호 : #080520-004
접수일자 : 2008년 05월 20일
심사완료일 : 2008년 07월 15일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

스템인 임무컴퓨터의 오동작이나 시스템 붕괴로 인한 전체 시스템의 중단은 임무의 실패뿐만 아니라 조종사의 생명까지도 위협할 수 있다. 시스템의 중단은 백업, 업그레이드, 유지보수 등의 계획에 의한 중단과 정전, 하드웨어 및 소프트웨어 오류, 해킹, 자연재해 등의 갑작스러운 원인으로 인한 중단으로 나눌 수 있다. 이러한 시스템의 중단으로부터 임무컴퓨터를 보호하기 위해 고가용 시스템으로 구성하여 관리해야 한다[24].

시스템의 실패를 유발하는 하드웨어 또는 소프트웨어의 단일 결함 요소를 SPOF(Single Point Of Failure)라 하며, 고가용 시스템을 구축하기 위해서는 SPOF를 제거하여야 한다[1]. 이를 위한 가장 간단하면서 신뢰할 수 있는 방법은 문제가 발생할 수 있는 부분을 이중화하는 것이다[2].

본 논문에서는 단일 시스템의 이중화를 통해 SPOF 요소를 제거하여 시스템의 고가용성을 보장하고, 이를 효율적으로 관리하기 위해 리눅스 기반의 고가용 소프트웨어를 구현한다. 또한 고가용 시스템의 성능측정을 위해 빠른 고장탐지를 위한 FDT(Fault Detection Time)와 고장 발생 시 임무 연속성에 중요한 영향을 미치는 MTTR(Mean Time To Repair)을 측정하여 분석한 내용을 기술한다.

본 논문의 구성은 2장에서 관련연구에 대해 살펴보고, 3장에서는 리눅스 기반의 고가용 시스템 구현과 성능측정 방법에 대해 기술한다. 4장에서는 실험환경 및 결과를, 마지막으로 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

II. 관련연구

1. Linux-HA

Linx-HA 프로젝트는 신뢰성(Reliability), 가용성(Availability), 내구성(Serviceability)을 위한 고가용 솔루션을 제공한다[3]. Linux-HA 프로젝트의 핵심 소프트웨어인 Heartbeat[4], Fake[5], Ipfail[6], DRBD[7]등을 이용하여 시스템에 고가용성을 보장하기 위한 방안을 제시하고 있으며, 이를 이용하여 데이터베이스, 웹, LVS, 메일, DNS, DHCP, Proxy Caching 서버를 구성할 수 있다[1][4]. 또한 RAID[8], GFS[9], LFS[10], CODA[11]등의 파일시스템과 MON[13], Nagios[14], Clumon[15], Ganglia[16], Ultra Monkey[12]등의 클러스터 모니터링 시스템과도 밀접하게 연구가 진행되고 있다.

1.1 Heartbeat

Heartbeat은 Linux-HA 프로젝트의 핵심 프로그램으로 시리얼(Serial line)이나 UDP 통신을 이용하여 고가용 시스템에서 특정 노드의 결함을 지속적으로 감시하는 기능을 한다. 고가용 시스템 노드들의 전체적인 구성과 운영방법에 따라 Active/Active방식과 Active/Standby방식으로 구분할 수 있다[17][21][23].

Active/Active방식은 각각의 노드가 각각 자신의 서비스를 제공하다가 하나의 노드에 고장이 발생하면, 고장이 발생하지 않은 노드가 고장이 발생한 노드의 서비스를 모두 담당한다. 이는 고장이 발생하지 않은 노드가 고장이 발생한 노드의 작업을 모두 수행해야 하므로 과중한 오버헤드로 인한 이차적인 고장이 발생할 수 있다.

Active/Standby방식은 각 노드가 primary/backup 방식으로 동작하며, primary 노드에 고장이 발생할 경우 대기하고 있던 backup 노드가 모든 서비스를 이전받아 수행한다. 이 방식은 구현은 쉽지만 primary 노드에 고장이 발생하기 전에 backup 노드는 아무런 처리를 하지 않고 대기해야 하므로 성능대비 가격이 높은 단점이 있다[22].

1.2 Fake

고가용 시스템은 최소 두 개 이상의 노드로 구성된다. 이러한 고가용 시스템에서 client노드가 primary 상태의 노드로 접근할 수 있게 하기 위하여 Fake 기법을 사용한다.

Fake는 ARP spoofing을 사용하여, 두 개의 노드를 하나의 가상 IP로 설정한다. 시스템이 구동되면 Fake는 가상 IP를 primary 노드가 사용하도록 설정하며, 외부로부터 전송되는 데이터는 primary 노드로 전송된다. primary 노드에 고장이 발생하면 Heartbeat 소프트웨

어는 이를 탐지하고, 모든 자원을 backup 노드로 이전하게 되는데, 이때 Fake에 의해 가상 IP역시 backup 노드로 이전된다. 그 후 외부로부터 모든 데이터는 backup 노드로 전송된다[5].

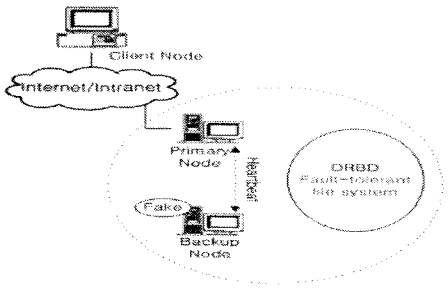


그림 1. Fake의 동작

1.3 DRBD (Distributed Replicated Block Device)

고가용 시스템에서는 각 노드에서 처리하는 데이터가 항상 일관성과 무결성을 유지해야 한다. 이를 위해 시스템의 설계에 따라 공유디스크를 사용하는 방법과 로컬 디스크를 사용하여 동기화 하는 방법으로 나눌 수 있다. DRBD는 각 노드의 로컬디스크를 동기화하는 기법으로 자신의 로컬디스크에 DRBD를 위한 파티션을 생성하여, 처리되는 데이터를 저장하고 실시간으로 다른 노드에 동기화하는 역할을 한다. [그림 2]는 DRBD의 동작이다.

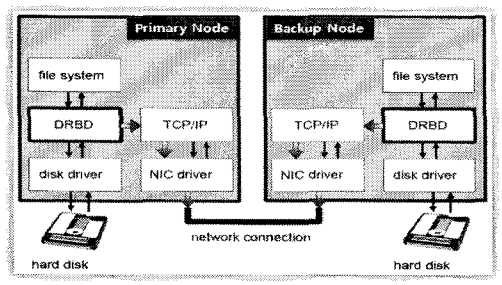


그림 2. DRBD의 동작

DRBD는 자체 파일시스템을 이용하여 모든 데이터를 관리하며, TCP/IP 프로토콜을 사용하여 실시간으로 데이터를 동기화 한다[7].

2. 기존의 고가용 시스템

heartbeat을 이용한 기존의 상용 고가용 시스템은 IBM사의 iSeries, zSeries, DB2와 HP사의 Service Guard for Linux등이 있으며 오픈소스 프로젝트로써 Beowulf cluster System을 강화하여 시스템의 고가용성과, computing 성능을 개선하기 위해 만들어진 HA-OSCAR (High Availability Open Source Cluster Application Resources)등이 있다. 각각의 고가용 시스템은 heartbeat 메커니즘을 사용하고 failover, fail-back 기능을 제공하며, 각각의 고유한 H/W 및 S/W 특성을 갖는다[25-28].

3. 주요 성능 척도

시스템의 성능측정을 위한 척도로는 가용성, 신뢰성, 관리 효율성, 안정성, 확장성, 보안성 등이 있다. 고가용 시스템의 특성상 서비스를 지속적으로 수행할 수 있는 가용성과 최대한 장애가 발생하지 않아야 하는 신뢰성이 중요한 척도가 된다.

3.1 가용성(Availability)

가용성이란 사용자가 주어진 자원이 필요한 시점에 그 자원을 사용할 수 있는 확률로써 수식 1과 같이 계산할 수 있다.

$$A(availability) = MTBF/(MTBF+MTTR) \quad (1)$$

MTBF(Mean Time Between Failures)는 평균 장애 간격 시간이며, MTTR(Mean Time To Repair)은 평균 복구 시간을 나타낸다. 또한, MTTF(Mean Time to Failure)는 어떤 시스템이나 시스템의 구성 요소가 고장 없이 수행을 계속할 수 있는 시간, 또는 고장 날 때까지의 기대 수명을 나타낸다[18][19].

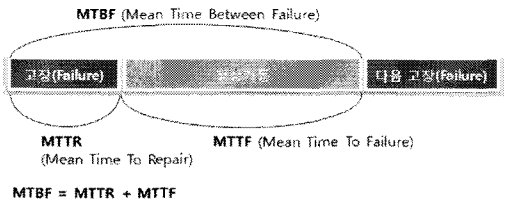


그림 3. MTBF, MTTR, MTTF의 관계

수식 1을 보면 가용성은 MTTR이 0에 가까워질수록 100%에 가까워진다는 것을 알 수 있다. 가용성은 가동 중지 시간 감소 및 제거에 초점을 맞추고 있으며, 고가용 시스템에서 가용성은 임무연속성을 위해 매우 중요한 요소이다.

3.2 신뢰도(Reliability)

신뢰도는 주어진 기간 동안 장애가 발생하지 않을 확률을 나타낸다. 수식 2를 이용하여 주어진 시간(t)동안의 신뢰도를 계산할 수 있다.

$$R(t) = e^{-(1/MTBF)(t)} \quad (2)$$

신뢰도는 중요한 임무를 수행해야하는 시스템에서 매우 중요한 요소로 MTBF를 증가시킴으로써, 시스템의 신뢰도를 높일 수 있다.

Ⅲ. 임무컴퓨터를 위한 고가용 시스템의 구현 및 성능평가 방법

1. 단일 시스템의 SPOF

매우 신뢰적인 단일 시스템이라도 SPOF가 존재한다. 시스템의 SPOF 에 고장이 발생하면 복구될 수 있는 장애가 발생하거나, 혹은 전체 시스템의 붕괴로 이어질 수 있다.

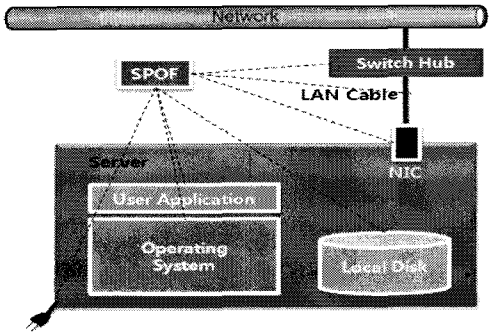


그림 4. 단일 시스템의 SPOF

전체 시스템의 붕괴는 곧 서비스의 중단으로 이어져, 상업적 시스템에서는 기업의 경제적 손실과 사용자의 불편함을 초래하고, 군용시스템에서는 임무의 실패와 사용자의 생명까지도 위협할 수 있다.

SPOF는 [그림 4]에서와 같이 네트워크, 운영체제, 전원 등 시스템의 다양한 부분에서 나타날 수 있으며 [표 1]의 방법으로 이를 해결할 수 있다.

표 1. SPOF 요소 제거

단일 컴포넌트	고장 제거 방법
CPU	- Backup CPU 제공
LAN, NIC	- 중복 NIC 설치 - Stand-alone 인터페이스 구성
DISK	- RAID 기법 적용
Power	- 시스템에 UPS 추가 - 전력원 추가 사용
Operating System	- 재시작과 복구를 위한 failover 기능
Application	- 자동으로 재시작하기 위한 기능제공 - 복구하기위한 기능 제공 - 코드 디버깅을 통하여 제공

2. 이중화를 통한 고가용 시스템의 설계 및 구현

시스템에 고가용성을 제공하기 위해 단일 시스템의 모든 SPOF를 이중화하여 제거하였다. [그림 5]와 같이 CPU, Disk, Power, Network 등을 이중화하여 시스템의 고장으로부터 유연하게 대처할 수 있다.

또한 설계한 고가용 시스템을 효율적으로 관리하기 위하여 리눅스 기반의 Heartbeat, Ipfail, DRBD, Bonding 등을 이용하여 관리한다.

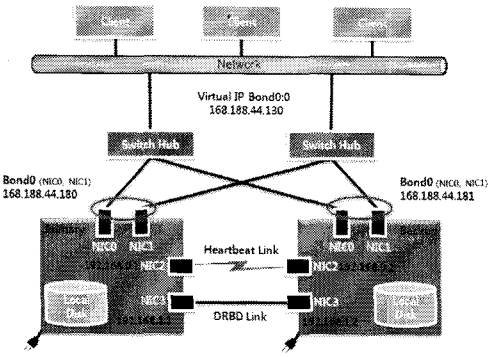


그림 5. 고가용 시스템의 구성

2.1 시스템 이중화 및 동작

시스템의 이중화를 위해 기존의 단일 시스템과 동일한 구성을 가진 시스템을 추가로 두어 고장에 대비할 수 있도록 한다. 두 개의 시스템은 클러스터로 구성되어 관리되며, 백업시스템의 운영방안은 Active /Standby 방식을 사용한다. 시스템에서 가장 중요한 프로그램은 Heartbeat으로 [표 2]의 파일에 의해서 관리된다.

표 2. Heartbeat 설정 파일

이 름	기 능
ha.cf	keepalive - 메시지전송주기 deadtime - 노드 실패의 판단시간 bcast - 메시지를 전송할 인터페이스
haresource	두 노두간의 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(take over)이 발생한다.

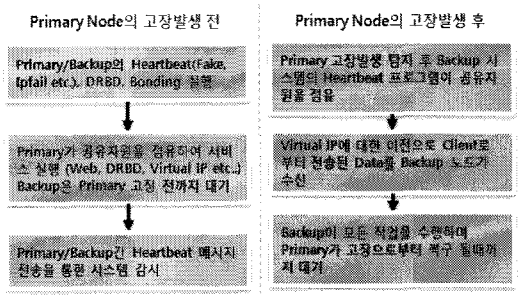


그림 6. 시스템의 동작방식

정상동작 시에 클라이언트로부터 전송되는 데이터는 Fake에 의해 생성된 가상 IP (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 6]은 primary 노드의 고장발생 전과 고장발생 후의 전체적인 동작이다.

2.2 네트워크 이중화 및 동작

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 NIC 또는 LAN Cable의 고장 여부를 파악한다.

본 시스템에서는 NIC0와 NIC1을 하나의 IP로 묶어 관리하며, pirmary 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 노드로의 이전이 발생한다.

2.3 디스크 저장소 이중화 및 동작

디스크 저장소의 특성에 따라 시스템은 여러 가지 방법으로 구성될 수 있다. 공유저장소를 사용하게 되면 두 노드가 저장소를 공유하여 데이터를 처리하게 되고, RAID 방식을 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/IP 프로토콜을 이용하여 동기화할 데이터를 전송하며, 공유데이터를 저장할 장소와 IP 및 port번호 등을 drbd.conf 파일에서 설정할 수 있다.

3. 고가용 시스템의 성능측정

3.1 성능측정 시 고려사항

고가용 시스템에서 성능측정을 위한 중요한 요소로 FDT와 MTTR을 고려해야한다. FDT는 빠른 고장 탐지를 위한 요소로서 고가용 시스템이 고장을 빠르게 탐지하지 못하게 되면 잘못된 데이터로 인한 시스템의 오동작과 같은 치명적인 오류가 발생할 수 있다. 또한 MTTR은 임무 연속성을 위해 중요한 요소로서 빠르게 Backup 시스템으로 이전하여 서비스를 계속해서 수행할 수 있어야 한다. HA-OSCAR의 경우 가용성을 측정하기 위해 MTTR을 고려하여 측정하였지만 FDT를 고려하지 않았다. 이에 본 논문에서는 성능측정 요소로 FDT와 MTTR을 고려하였다.

3.2 FDT (Fault Detection Time)

고장을 신속하게 탐지하지 못할 경우 시스템의 잘못된 연산이나 오동작으로 인해 고가용 시스템에 치명적인 오류가 발생하게 되고, 임무의 실패뿐만 아니라 막대한 인적 및 물적 자원의 피해를 가져온다.

구성한 고가용 시스템에서 고장 탐지는 Heartbeat에 의해 관리되며, keepalive 시간을 주기로 메시지를 보내고, deadtime 이내에 응답이 오지 않을 경우 고장으로 판단하므로, 고장 탐지 속도를 최소화하기 위해서는 최적의 keepalive 시간과 deadtime을 설정하는 것이 매우 중요한 일이다.

■ Heartbeat 시간 설정

Heartbeat에서는 keepalive 시간과 deadtime을 밀리세컨드(ms) 단위까지 설정할 수 있으며 항상 수식 3의 관계가 성립해야 한다.

$$deadtime > 2*keepalive \quad (3)$$

deadtime이 수식 3의 관계를 만족하지 못하면 네트워크가 과부하 상태에 있을 때 Heartbeat 메시지의 지연이 발생하여 노드에 고장이 발생하지 않았음에도 불구하고 고장으로 판단하게 될 수도 있다.

최적의 keepalive 시간과 deadtime을 측정하기 위하여 Heartbeat의 로그 메시지를 모니터링하여 에러메시지가 발생하지 않는 범위 내에서 keepalive 시간과 deadtime값을 줄여가며 시스템의 상태를 모니터링 하였다.

■ FDT 측정 방법

위에서 설정한 최적의 Heartbeat 시간으로 고가용 시스템을 구동시킨 후 리눅스에서 "kill" 명령어를 이용하여 관련 프로세스의 실행을 강제로 멈추고 타임스탬프를 저장한다. Heartbeat 프로그램은 시스템에 고장이 발생하였음을 판단하게 되고, primary 노드가 고장을 처음 판단한 시점을 로그 메시지를 이용하여 알아내고 타임스탬프를 저장한다. 두 개의 타임스탬프 차이를 계산하여 FDT를 측정하였다.

3.3 MTTR (Mean Time To Repair)

구성한 고가용 시스템에서는 primary 노드에 고장이 발생하면 모든 작업을 backup 노드로 이전하여 고가용성을 보장한다. 만약 backup 노드로의 이전 시에 많은 시간이 소요된다면 실제 임무를 수행하는 사용자 입장에서는 연속적으로 임무를 수행할 수 없게 된다. 이는

사용자에게 매우 불편함을 줄 뿐만 아니라 중요한 정보의 손실을 야기할 수 있다. 그러므로 backup 노드로의 이전은 사용자가 고장을 인지할 수 없을 만큼 빠른 시간에 이전을 완료하여 임무 연속성을 보장해야 한다.

■ 시간동기화

MTTR을 측정하기 위해서는 primary 노드와 backup 노드의 시간동기화가 필요하다. 이에 본 논문에서는 리눅스에서 제공하는 시간동기화 프로토콜인 NTP(Network Time Protocol)[20]를 사용하여 두 노드간의 시간을 동기화 한다.

■ MTTR에 영향을 미치는 요소

MTTR은 CPU의 성능, 네트워크의 상태, Heartbeat interval, fail over time, primary 노드에서 실행되고 있는 애플리케이션 중 복구가 필요한 애플리케이션의 개수에 의해 결정된다.

■ MTTR 측정 방법

NTP를 이용하여 각 노드의 시간을 동기화 하고 Heartbeat 소스를 수정하여 primary 노드에서 고장이 발생하였음을 판단하는 시점과 backup 노드로 이전이 완료되는 시점에 각각 타임스탬프를 설정한다. 타임스탬프 정보를 시스템의 로그 메시지에 저장하여 그 시간차를 계산하였다. [표 3]은 고가용 시스템의 MTTR과 시스템 비가용 시간을 나타낸다.

표 3. MTTR과 시스템 비가용 시간

	측정시간
MTTR	backup 노드의 take over 완료시간 - primary 노드의 이상 판단시간
비가용시간	deadtime + MTTR

Ⅳ. 실험 환경 및 결과

1. 시스템 구축비용 및 효율성

본 논문에서 구현한 고가용 시스템은 리눅스 운영체제를 기반으로 [표 4]와 같은 H/W를 사용하였다. 또한 Backup시스템의 운영방안으로 Active/Standby 구조를 사용하였기 때문에 Backup 노드는 Primary 노드에 고장이 발생하기 전까지 아무 일도 수행하지 않고 대기해야한다. 일반적인 단일 시스템과 비교하여 가용성과 신뢰성을 보장할 수 있지만, 전체적인 시스템의 이중화로 인한 추가적인 H/W의 비용이 필요하다.

표 4. 실험 서버환경

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

2. 고가용 시스템의 동작

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.

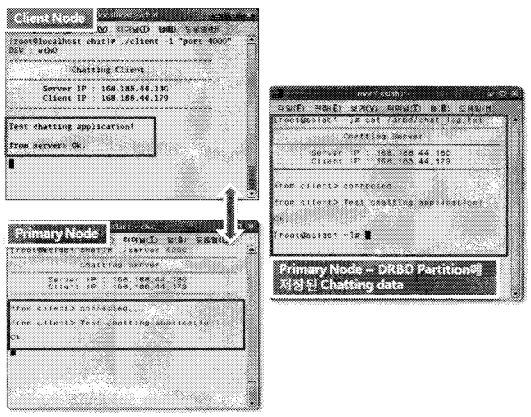


그림 7. primary 노드 고장발생 전 채팅 프로그램

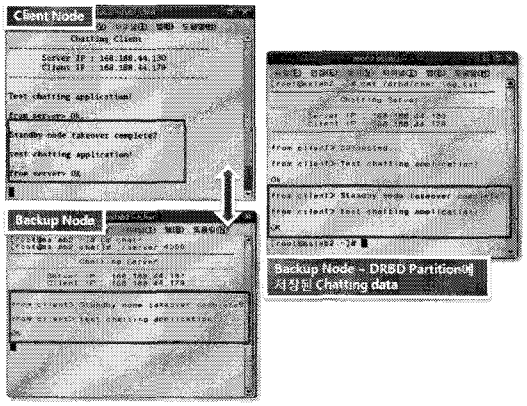


그림 8. primary 노드 고장발생 후 채팅 프로그램

[그림 7]에서는 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 8]에서 primary 노드에 고장이 발생하면 모든 공유자원이 backup 노드로 이전되며 모든 자원을 이전받은 후 클라이언트와 backup노드(Virtual IP : 168.188.44.130, Real IP : 168.188.44.181)간에 채팅이 이루어진다. 또한 모든 채팅 내용은 backup 노드에 있는 로컬디스크에 저장된다.

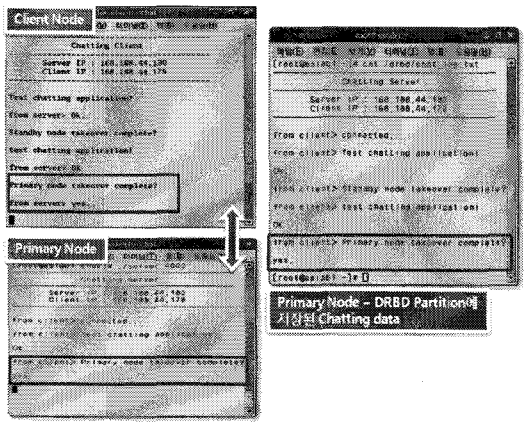


그림 9. primary 노드 고장복구 후 채팅 프로그램

[그림 9]는 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 primary 노드가 임무를 수행하지 못할지라도 backup 시스템을 이용하여 시스템에 고가용성을 제공함으로써, 고장으로부터 유연하게 대처할 수 있다.

3. 고가용 시스템 성능측정

3.1 Heartbeat 시간 설정

구성한 고가용 시스템에서 keepalive 시간은 40ms이상으로 deadtime은 180ms이상으로 설정해야 하며, 그 이하로 설정할 경우 [표 5]와 같은 문제점이 발생한다.

표 5. Heartbeat 시간설정 제한

Heartbeat 시간	문 제 점
keepalive time 〈 40ms	메시지를 수신하는 read_child process에 버퍼 오버플로우가 발생하여 메시지 처리 불가
deadtime 〈 180ms	네트워크를 감시하는 Ipfail 데몬에서 메시지 지연 발생으로 인한 시스템 불안정

3.2 FDT (Fault Detection Time)

primary 노드에 임의의 고장을 발생시킴으로써 FDT를 측정하였다. 측정회수는 100회이며, 현재 구축한 고가용 시스템에서 평균 4.3ms의 속도로 고장을 탐지하는 것을 확인하였다. 이는 하드웨어의 성능에 따라 달라질 것으로 예상되며 결과는 [표 6]과 같다.

표 6. FDT 측정

	시 간(ms)
최대 소요시간	5.857
평균 소요시간	4.321
최소 소요시간	2.699

3.3 MTTR (Mean Time To Repair)

시스템의 가용성은 수식 1에 의해 계산되며, 오랜 시간동안 시스템을 구동시켜 테스트를 해야 한다. 본 논문에서는 단기간동안의 시스템 구동으로 인해 정확한

가용성은 측정하지 못하였고, 고가용 시스템에서 임무 연속성을 위해 중요한 MTTR을 측정하여 성능을 평가하였다.

표 7. MTTR 측정

	시 간(ms)
최대 소요시간	1551.330
평균 소요시간	964.985
최소 소요시간	717.036

구현한 고가용 시스템에서는 간단한 채팅 애플리케이션을 이용하여 MTTR을 측정하였다. 기존의 고가용 시스템은 그 특성에 따라 각각의 MTTR을 갖는다. IBM의 DB2의 MTTR은 9초, HP의 Guard for Linux의 경우는 5초 이내이다. 또한 HA-OSCAR는 3~5초의 MTTR을 갖는다[25-28].

MTTR에 영향을 미치는 다양한 요소와 구현 H/W의 차이로 단순 비교는 무의미 할 수 있지만, 기존의 고가용 시스템과 비교하여 본 논문에서 구현한 고가용 시스템의 성능은 FDT는 평균 4.3ms, MTTR는 0.7~1.6초로 성능상의 오버헤드는 크지 않다.

V. 결론

본 논문에서는 단일시스템의 이중화를 통하여 시스템 붕괴의 원인이 되는 SPOF 요소를 제거하였다. 또한 이중화 된 시스템을 클러스터로 관리하고, Heartbeat, Fake, DRBD, Bonding등의 고가용성을 제공하는 여러 소프트웨어 기법을 적용하여 고가용 시스템을 설계하였다. 고가용 시스템의 성능측정을 위해 Heartbeat의 keepalive 시간과 deadtime을 조절하여 시스템을 최적화 시키고, FDT와 MTTR을 측정하였다. FDT는 고가용 시스템에서 매우 중요한 요소로 빠르게 고장을 탐지해야 시스템에 고장이 확산되는 것을 방지하고, 그 고장으로부터 복구될 수 있다. 고장으로부터 복구될 수 없는 경우에는 backup 노드로의 이전을 통하여 시스템을 지속적으로 구동시킨다. MTTR은 임무연속성을 위해 매우 중요한 요소로 MTTR이 짧아야 빠르게 고장으로부터 복구되어 시스템의 가용성을 높일 수 있다. 이로써 구축한 고가용 시스템에서의 평균 FDT는 4.321ms이고, 평균 MTTR은 965ms 가 소요되는 것을 확인하였다.

향후 연구과제로는 primary 노드에서 backup 노드로 이전이 발생할 때 디스크, 메모리, CPU등의 자원을 효율적으로 복구할 수 있는 방안에 대해 연구가 이루어져야 한다. 또한 실제 사용자에게 서비스를 제공하는 상용시스템에 리눅스 기반의 고가용 시스템을 구축하여 장기간 모니터링 함으로써, 전체 시스템의 가용성과 신뢰성을 측정하여야 한다.

참 고 문 헌

[1] P. S. Weygant, *Clusters for High Availability : A Primer of HP Solutions*, Prentice Hall PTR, 2001.

[2] D. K. Pradhan, *Fault-Tolerant Computer System Design*, Prentice Hall PTR, 1996.

[3] http://linux-ha.org

[4] http://www.linux-ha.org/Heartbeat

[5] http://www.vergenet.net/linux/fake/

[6] http://www.linux-ha.org/ipfail

[7] http://linbit.com/, http://drbd.org/

[8] http://www.linas.org/linux/Software-RAID/Software-RAID.html

[9] http://www.redhat.com/gfs/

[10] http://logfs.sourceforge.net/

[11] http://www.coda.cs.cmu.edu/

[12] http://www.ultramonkey.org/

[13] http://mon.wiki.kernel.org/index.php/Main_Page

[14] http://netsaint.sourceforge.net/

[15] http://clumon.ncsa.uiuc.edu/

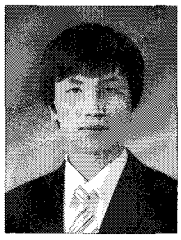
[16] http://ganglia.info/

[17] A. Robertson, "Linux-HA Heartbeat System Design," USENIX, pp.305-316, 2000.

[18] http://en.wikipedia.org/wiki/Mean_time between_failures
[19] http://en.wikipedia.org/wiki/Mean_time_to recovery
[20] http://www.ntp.org/
[21] J. E. J. Bottomley, "Implementing Clusters for High Availability," USENIX, pp.237-244, 2004.
[22] D. P. Siewiorek, "Architecture of fault-tolerant computers : an historical perspective," IEEE, Vol.79, No.1, pp.1710-1734, 1991.
[23] 최종명, 한주현, 최재영, "Diehard:인터넷 서비스를 위한 N-way 고가용성 시스템", 정보과학회 논문지, 제28권, 제8호, pp.390-398, 2001.
[24] 배재환, "멀티미디어 관광정보시스템을 위한 고가용성 리눅스 서버에 관한 연구", 한국통신학회 논문지, Vol.29, No.9B, pp.818-825, 2004.
[25] http://xcr.cenit.latech.edu/ha-oscar/index.html
[26] L. Chokchai, S. Lixin, L. Tong, S. Hertong, and L. S. Stephen, "Availability Prediction and Modeling of High Availability OSCAR Cluster," IEEE, pp.380-387, 2003.
[27] ftp://ftp.software.ibm.com/software/data/pubs/papers/10sfailover.pdf
[28] http://h71028.www7.hp.com/enterprise/downloads /Optimizing%20failover_6-22.pdf

저 자 소 개

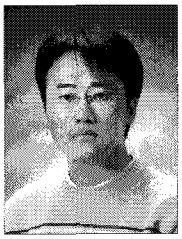
정 재 엽(Jae-Yeop Jeong) 준회원



- 2007년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2007년 3월 ~ 현재 : 충남대학교 컴퓨터 공학과 석사과정 재학

<관심분야> : 실시간 운영체제, 임베디드 시스템, 고장감내 시스템

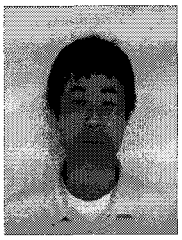
박 성 종(Seong-Jong Park) 준회원



- 2007년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2007년 3월 ~ 현재 : 충남대학교 컴퓨터 공학과 석사과정 재학

<관심분야> : 실시간 운영체제, 임베디드 시스템, 고장허용 컴퓨팅

임 재 석(Jae-Seok Lim) 준회원



- 2008년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2008년 3월 ~ 현재 : 충남대학교 컴퓨터 공학과 석사과정 재학

<관심분야> : 실시간 운영체제, 고장허용 컴퓨팅

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과(공학사)
- 1988년 2월 : 한국과학기술원 전기및전자공학과(공학석사)
- 1992년 2월 : 한국과학기술원 전기및전자공학과(공학박사)
- 1983년 3월 ~ 1986년 2월 : 삼성전자 컴퓨터사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 선임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원 연구원
- 1995년 2월 ~ 현재 : 충남대학교 컴퓨터공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙 연구원

<관심분야> : 실시간시스템, 운영체제, 고장허용 컴퓨팅