Diehard: 인터넷 서비스를 위한 N-way 고가용성 시스템

(Diehard: N-way High Availability System for Internet Services)

최 종 명 † 한 주 현 † 최 재 영 ††

(Jong-Myung Choi) (Joohyun Han) (Jaeyoung Choi)

요 약 인터넷을 이용한 서비스와 산업이 급속도로 발전하면서 인터넷 서비스를 위한 효율적이며 확장성이 있는 고가용성 시스템이 매우 중요한 요소로 인식되고 있다. 인터넷 서비스를 위한 고가용성 시스템은 컴퓨터들이 쉬지 않고 서비스를 제공할 수 있는 active-active 형태로 구성되어야 하며, 확장성을 위해서 N개의 컴퓨터가 연결되어 클러스터를 구성하는 N-way 방식을 지원하여야 한다. 본 논문에서는 N개의 컴퓨터들이 클러스터로 연결된 N-way에서 active-active 형태의 구성을 위한 알고리즘을 제시하고, 이를 바탕으로 구현된 고가용성 시스템인 Diehard를 소개한다. Diehard는 인터넷 서비스의 특성을 고려해서 소프트웨어 SPOF를 모니터링하고, 부분적인 서비스 중단에 대한 다양한 복구 정책 등을 지원하도록 설계 및 구현되었다.

Abstract The high availability system becomes more important as the need for Internet services is rapidly increasing. The services should be provided as an "active-active" manner for continuity and efficiency, and computers, which provide the services, should be connected in a form of "N-way" for scalability. In this paper, we present an algorithm to provide high availability for the N-way and introduce a high-availability system called Diehard. The Diehard provides various functions that monitor software SPOFs including partial service failure, determine the failure levels, and recover the failures.

1. 서 론

고가용성(high availability)이란 하드웨어, 소프트웨어 혹은 네트워크 등에 문제가 발생하여도 서비스를 지속적으로 제공할 수 있도록 해주는 기술을 말한다[1]. 상업적으로 사용되는 시스템에서 서비스 중단은 심각한 손실을 초래할 수 있다. 1995년 오라클과 데이타메이션의 보고서에 따르면 갑작스러운 서비스 중단은 평균적으로 시간당 8,000~35,000 달러의 경제적 손실을 가져온다[2]. 표 1에서 보듯이 최근 데이타에 의하면 서비스 중단에 따른 손실은 더욱 증가되고 있다. 표 1은 서비스 중단에 따른 업체별 시간당 손실을 보여준다[3].

표 1 서비스 실패에 따른 비용

업체	시간당 비용
증개 사업	$6.45M
신용 카드/판매 인증	$2.6M
홈 쇼핑(TV)	$113K
홈 카달로그 판매	$90K
항공 예약	$89.5K
전화 티켓 판매	$69K
패키지 해운업	$28K
ATM 수수료	$14.5K

서비스 실패는 하드웨어/소프트웨어의 결함, 관리자의 실수, 시스템 백업 및 업그레이드 등의 원인 때문에 발

† 비 회 원 : 숭실대학교 컴퓨터학과
jmchoi@it.ssu.ac.kr
dossy@ss.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
choi@comp.ssu.ac.kr
논문접수 : 2000년 7월 19일
심사완료 : 2001년 6월 1일

생한다. 서비스 실패를 유발하는 하드웨어나 소프트웨어의 결함 요소를 SPOF(Single Point Of Failure)[4]라고 하고, 고가용성 서비스를 위해서는 SPOF를 제거하여야 한다. SPOF를 제거하는 가장 간단한 방법은 문제가 발생할 수 있는 부분을 중복해서 설치하는 것이다. 중복성과 고성능을 제공하기 위해서 클러스터 시스템이 사용되며, 클러스터 시스템의 컴퓨터들은 active-standby나 active-active 형태로 구성할 수 있다. active-standby 구성에서 standby 컴퓨터는 active 컴퓨터가 작동하는 동안에 다른 작업을 하지 않고 대기하기 때문에 시스템 활용 측면에서 비효율적이다. 또한 클러스터 시스템에서 확장성을 고려하면 N개의 컴퓨터를 연결할 수 있는 N-way 방식이 필요하다.

하드웨어의 SPOF는 하드웨어 기술의 발달로 MTBF (Mean Time Between Failure)가 커지고 있지만 소프트웨어는 아직 SPOF 문제를 해결하지 못하고 있다. 그림 1에서 보듯이 서버측 소프트웨어의 결함으로 인한 서비스 실패가 서비스 실패의 원인으로 가장 많은 부분을 차지하는 것으로 드러났다[3]. 따라서, 데이타베이스, 웹서버, 응용 프로그램 서버, HTML 문서, CGI(서블릿, PHP, ASP 등) 프로그램 등의 많은 소프트웨어 요소들로 구성된 인터넷 서비스는 소프트웨어 SPOF 문제를 해결할 수 있는 기능이 필요하다.

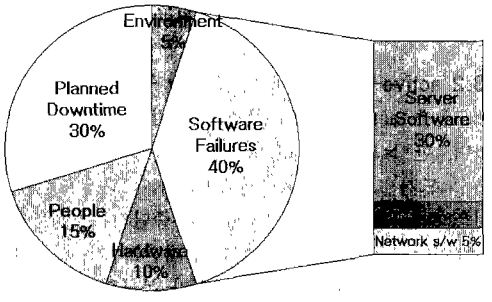


그림 1 서비스 중단의 원인

기존에 고가용성 NFS[5]나 워크플로우 시스템을 위한 고가용성 시스템[6]에 관한 연구들은 있었지만, active-active 구성의 N-way를 위한 시스템에서 fail-over나 fail-back 알고리즘에 관련된 연구는 발표된 사례가 적었다. 또한 인터넷 서비스의 특성을 고려한 고가용성 시스템에 관한 연구도 상대적으로 적었다.

본 논문에서는 active-active 형태로 동작되는 N-way를 지원하기 위한 알고리즘을 소개하고, 이 알고리즘을 구현한 고가용성 시스템인 Diehard를 소개한다. Diehard 시스템에서 N개의 컴퓨터들은 시리얼과 네트워크로 연결되어 클러스터를 구성하며, 각자 자신의 서비스를 수행한다. 만약 한 컴퓨터가 중지되는 경우에 클러스터에 포함된 다른 컴퓨터들 중에 하나가 선출되어 중지된 컴퓨터의 서비스를 인계받아 수행한다. Diehard는 소프트웨어의 SPOF 문제를 해결하기 위해서 소프트웨어 기능 감시와 복구에 중점을 두어 소프트웨어의 결함을 효과적으로 처리할 수 있다. 또한 내부적으로 서비스 가용여부를 판단할 수 없는 경우를 대비해서 원격 모니터링 기능을 두어 외부 클라이언트에서 실질적으로 서비스를 받을 수 있는지 여부를 판단할 수 있도록 한다.

본 논문은 2장에서 고가용성 시스템에 관련된 연구들을 소개하고, 3장에서 Diehard 시스템의 구성 및 fail-over와 fail-back을 위한 알고리즘을 설명한다. 4장에서는 구현 내용과 테스트에 대해서 기술하고, 마지막 5장에서는 결론 및 향후 연구 과제를 밝힌다.

2. 관련 연구

2.1 RSF-1

High-Availability.com사에서 개발한 리눅스 고가용성 시스템이다[7]. RSF-1(Redundant Server Facility) 에이전트들이 서버의 결함을 주기적으로 검사하고, 서버의 결함이 감지되면 10초 이내에 다른 서버에서 서비스를 인계받는다. 서버의 결함을 감지할 때 오류를 방지하기 위하여 TCP/IP, RS-232C 시리얼, 다중 공유 디스크 등의 다양한 heartbeat 메카니즘을 사용한다. RSF-1은 간단한 콘솔 인터페이스와 구성 도구, 그리고 서비스를 기동시키고 정지시키는 방법을 기술하는 스크립트 인터페이스를 제공하는데, 비교적 사용자의 편의성이 부족하다.

2.2 Linux HA

Linux-HA[8] 공개 프로젝트는 여러 방면에서 솔루션을 제공한다. 현재 Heartbeat[9], Fake[10], Heart[11] 등의 고가용성 시스템을 위한 프로젝트와 High availability RAID[12], GFS[13], LFS[14], CODA[15] 등의 파일 시스템, Mon[16], PIKT[17], NOCOL/SNIPS[18] 등의 모니터링 시스템 등 다양한 분야와 밀접하게 연계되어 진행되고 있다.

Heartbeat은 시리얼, UDP, PPP/UDP 미디어를 통한 서버 결함 검사 기능과 서버의 결함이 발생하였을 때, 결함 노드의 IP 주소를 인계받는(takeover) 기능, 리소스 그룹을 포함한 뛰어난 리소스 모델 등을 지원한다[9]. Fake은 부가적인 인터페이스나 ARP 스푸핑(spoofing)

을 사용하여 동일 네트워크에 있는 다른 기계의 IP 주소를 넘겨받는 기능을 가진 소프트웨어로 결함이 발생한 서버의 주소를 다른 서버에서 인계받을 수 있도록 한다[10]. Mon은 일반적인 자원 모니터링 시스템으로 네트워크 서비스의 가용성, 서버에서 제공하는 서비스의 가용성, 전산실의 온도같은 환경적인 조건 등을 모니터하는 기능을 지원한다[16].

Linux-HA 공개 프로젝트는 여러 방면에서 솔루션을 제공하지만, 상용 소프트웨어에 비해 기능이 뒤떨어지고, 각 소프트웨어가 개별적으로 개발되었기 때문에 통합하여 사용하기 어려우며, 사용자에게 편의성을 제공하지 못한다는 단점이 있다[19]. Linux-HA는 소프트웨어 SPOF를 고려하지 않고 있으며, active-standby 형태로 구성되어 있다.

2.3 Proxy 서버를 이용한 고가용성 웹서버

건국대학교에서 개발된 이 시스템은 주 서버와 백업 서버를 두고, 각 서버에 HA Proxy를 둔다[20]. 주 서버와 백업 서버는 shared nothing 방식으로 구성되어 있기 때문에 상태를 일치시키기 위하여 주 서버에 오는 요청을 그대로 백업 서버에서도 수행해주고, 주 서버나 백업 서버가 다운된 경우에는 로그를 남기는 방법을 사용한다. 상태를 일치시키기 위하여 클라이언트의 요청을 웹서버가 직접 받는 것이 아니고 주 서버의 HA Proxy가 받아서 백업 서버의 HA Proxy에 전달한다. 이 시스템은 기존의 웹서버를 그대로 사용할 수 있으며, 구현하기 간단하고, 저비용으로 구현할 수 있는 장점이 있다. 그러나 2-way의 active-standby 방식으로 되어 있기 때문에 확장성과 시스템 효율성에서 떨어지는 단점이 있으며, 웹서비스가 아닌 다른 서비스를 제공해야 하는 경우에는 HA Proxy 역할을 하는 다른 모듈을 개발해야 한다는 부담이 있다.

3. 본 론

3.1 Diehard 네트워크 구성

Diehard는 active-active 형태의 클러스터로 구성되어 고가용성 서비스를 제공한다. Diehard는 물리적으로 그림 3과 같은 형태로 구성되어 있다. 클러스터에 포함된 컴퓨터들은 네트워크와 시리얼 케이블로 연결되어 있고, 컴퓨터들은 각각 2개의 NIC를 가지고 있어서 외부 네트워크(public network)와 내부 네트워크(private network)로 연결되어 있다. 내부 네트워크와 시리얼 케이블은 노드들간의 heartbeat 통신을 위해서 사용된다. 클러스터에 포함된 컴퓨터들은 주기적으로 heartbeat 신호를 통해 다른 컴퓨터의 상태를 점검해서 서비스가 이루어지고 있는지 확인한다. 컴퓨터의 NIC도 SPOF가 될 수 있기 때문에 heartbeat는 내부 네트워크와 시리얼 케이블을 통해 다른 컴퓨터가 살아있는지 여부를 판단한다. 컴퓨터 노드들 간에는 서로를 감시하기 위해서 시리얼 케이블로 연결되어 있는데 각 노드가 모든 노드를 연결하는 것이 가장 확실한 방법이지만 이러한 경우에 N 개의 노드를 사용하면 N*(N-1) 개의 시리얼 라인이 필요하기 때문에 확장성이 떨어진다. 따라서 Diehard는 노드들을 원형으로 연결해서 바로 옆의 컴퓨터를 감시하도록 한다. 이러한 연결은 간단하면서 각 노드가 2개의 시리얼 라인으로만 연결되기 때문에 확장성면에서 뛰어나다.

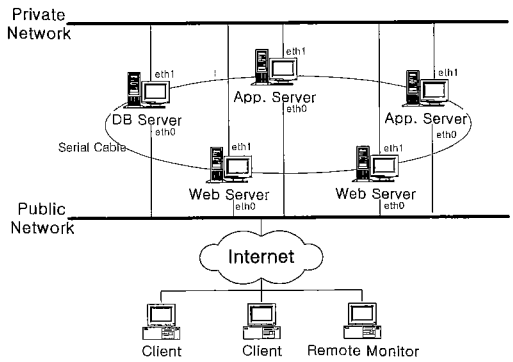


그림 2 Diehard 네트워크 구성

3.2 active-active 형태의 N-way 구성

노드들은 active-active 방식으로 각각의 서비스를 수행하며, 한 노드가 결함으로 서비스가 중단되는 경우에 다른 노드가 그 서비스를 인계받는다. 다음은 정상적인 서비스(그림 3)와 웹서버가 서비스 중단되는 경우에 데이타베이스 서버가 웹서비스를 인계받는 것(그림 4)을 보여준다. 서비스가 중단된 컴퓨터가 복구되면 그림 4의 상태에서 그림 3의 상태로 전이된다.

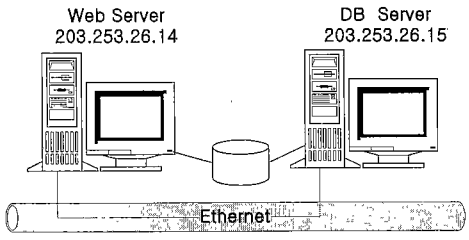


그림 3 fail-over가 일어나기 전

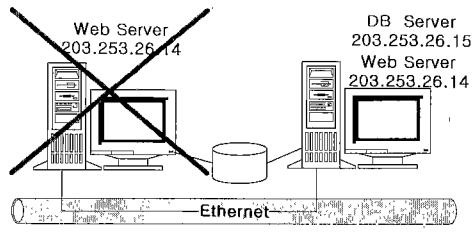


그림 4 fail-over가 발생한 후

클러스터에 포함된 노드들은 원형으로 연결되어 있으며, 주기적으로 HEART_BEAT_ PACKET을 전송해서 앞 뒤 노드들의 상태를 체크한다. 만약 어느 일정 시간 동안에 상대방에게서 응답이 없는 경우에 일시적인 장애나 시스템 부하 때문에 응답을 할 수 없기 때문에 미리 정해진 횟수만큼 다시 패킷을 전달하고, 기다리는 작업을 수행한다. 계속적으로 응답이 없는 경우에 이 노드는 중지된 것으로 간주하고 fail-over 작업을 수행한다. 노드들 간에 heartbeat 신호를 체크하는 알고리즘은 알고리즘 1과 같다.

알고리즘 1 heartbeat 신호 점검

```
Heartbeat algorithm
begin
    while( forever )
        send HEART_BEAT_PACKET to neighbor
        receive HEART_BEAT_PACKET from neighbor
        if( don't receive HEART_BEAT_PACKET for a
        fixed time )
            send HEART_BEAT_PACKET to neighbor
            wait for a fixed time to receive HEART_
            BEAT_PACKET
            if( don't receive HEART_BEAT_PACKET
            for a fixed time )
                start fail-over
            end if
        end if
    end while
end
```

중단된 노드가 발견되면, 클러스터에 소속된 모든 노드들에 장애 사실을 브로드캐스트하고, 동적으로 다시 정상적으로 수행되는 노드들로 클러스터를 구성한다. 클러스터에서 부여된 번호가 가장 작은 노드가 컨트롤러의 역할을 수행하면서, 시스템 로드가 가장 적은 노드를 찾아서, 이 노드가 중단된 서비스를 인계받도록 한다. fail-over는 알고리즘 2에 기술된 단계로 진행된다.

알고리즘 2 fail-over 단계

```
Fail-over algorithm
begin
    broadcast the system fault information to all nodes in
    the cluster
    rebuild cluster dynamically with active nodes
    if( this node == the first node in the cluster )
        mark this node as controller
        for( all nodes in the cluster )
            send SYSTEM_LOAD_PACKET
            receive SYSTEM_LOAD_PACKET
        end for
        find out the node with the lowest system load
        send FAIL_OVER_PACKET to the node
    else
        receive SYSTEM_LOAD_PACKET
        check system load
        send SYSTEM_LOAD_PACKET to the controller
    end if
    if( this node==the node with the lowest system load )
        start fake_process
        setup system for take-over services
        start services
    end if
end
```

fail-over[1]가 발생한 후에 다운되었던 시스템이 정상적으로 복귀하는 경우에, IP 주소의 충돌과 디스크 공유에 따른 문제들과 이에 따라 파생되는 문제들이 발생할 수 있다. 따라서 fail-over가 발생한 후에는 fail-back[1] 작업을 위한 적절한 절차를 거쳐야 한다. 다운되었던 서버 A를 내부 IP를 이용해서 부팅시킨다. 다음 단계로 Diehard의 구성 정보를 살펴보고, 클러스터에 포함되어 있는 다른 컴퓨터에 IP_CHECK_PACKET을 전달한다. IP_CHECK_PACKET을 받은 컴퓨터는 자신이 현재 서버 A의 서비스를 수행하고 있는지 여부를 CHECK_IP_ANSWER_PACKET을 통해 전달한다. 서버 A는 다른 컴퓨터가 자신의 서비스를 제공하지 않는 경우에, 외부 IP를 활성화시키고 서비스를 시작한다. 서버 A의 서비스를 다른 컴퓨터에서 제공하고 있는 경우에는 해당 컴퓨터에 FAIL_BACK 패킷을 전달한다. FAIL_BACK 패킷을 받은 컴퓨터는 서버 A에 서비스를 넘겨주기 위한 작업을 수행하고, 작업이 끝나는 경우에 RESPONSE를 서버 A에 전달한다. RESPONSE를 받은 서버 A는 외부 IP를 활성화시키고, 서비스를 제공한다. fail-back은 알고리즘 3에 기술된 절차에 따라 진행된다.

알고리즘 3 fail-back 단계

```
Fail-back algorithm
begin
    boot the system with a internal IP
    check Diehard configuration
    for( all nodes in the cluster )
        send CHECK_IP_PACKET
        receive CHECK_IP_ANSWER_PACKET
        find out take-over server
    end for
    if( take-over server exists )
        send FAIL_BACK_PACKET
        receive FAIL_BACK_OK_PACKET
    end if
    setup external IP
    check computing resources for services
    start services
end
```

4. Diehard 시스템 구성

Diehard 시스템의 구조는 다음 그림 5에서 보는 바와 같이 Diehard Heartbeat, SPOF 모니터 (SPOF Monitor), 복구 관리자 (Recovery Manager), 구성 관리자 (Configuration Manager), GUI 모니터 (GUI Monitor), 원격 모니터 (Remote Monitor)로 구성되어 있다.

인터넷 서비스는 클라이언트와 서버가 인터넷을 통해 연결되어 있으며, 독립적인 혹은 반 독립적인 서버 프로그램들과 이와 관련된 프로그램 및 자원들로 구성되어 있다. 따라서 인터넷 서비스는 다음과 같은 특성들을 가지고 있다.

- 많은 네트워크 장비로 구성되어 있다.
- 많은 소프트웨어 컴퓨팅 자원들로 구성되어 있다.
- 부분적인 서비스 실패가 일어날 가능성이 높다.
- 서비스 실패의 중요도가 여러 단계로 나누어질 수 있다.
- 로컬 컴퓨터에서는 서비스 중단을 식별할 수 없는 경우가 많다.
- 해킹에 대비해야 한다.

인터넷 서비스는 웹 서버 프로그램, 메일 서버 프로그램, 각종 응용프로그램 서버, 서블릿 /PHP/ASP 등의 CGI 관련 프로그램, DBMS, HTML 파일, 그림 파일 등의 많은 소프트웨어적인 자원들로 구성되어 있다. 이러한 자원들은 모두 SPOF로 문제가 발생할 수 있다. 따라서 주기적으로 자원들을 모니터링하고, 문제가 발생하는 경우에 적절한 조치를 취해야 한다. 또한 이러한 소프트웨어는 서로 독립적 혹은 반 독립적인 형태로 존재하기 때문에 일부 프로그램이나 일부 모듈만 실패하는 경우에 일부 서비스만 중단된다. 이러한 부분적인 실패를 파악하고, 복구하기 위한 다양한 복구 정책이 필요하다. 어떤 서비스 실패는 무시할 수 있는 반면에 어떤 서비스 실패는 매우 위급한 상황일 수 있다. 예를 들어, 해당 그림 파일이 없거나, 방문자 카운트를 표시하는 프로그램이 중단되는 것은 무시할 수 있는 서비스 실패일 수 있으나, 웹서버나 DBMS 서버의 실패는 치명적인 문제가 될 수 있다. 인터넷 서비스에서 해킹에 대비하기 위한 보안 관련 기술들이 필수적이다. 이러한 문제들은 주로 보안에 관련된 사항들이지만 고가용성 시스템측면에서도 해커에 의해 잘못된 서비스나 잘못된 정보를 제공하는 것을 방지할 필요가 있다.

4.1 Diehard Heartbeat

Diehard Heartbeat은 서버간에 통신을 통해 상대방 컴퓨터의 상태를 알아보는 역할을 수행한다. 상대방 컴퓨터가 일정 시간 동안 응답하지 않는 경우에 서버는 상대편 서버가 서비스 중단된 상태로 판단한다. 다른 서버가 중단된 상태로 판단되면 Diehard Heartbeat은 내장된 Fake 모듈을 이용하여 Fake를 수행해서, 상대방

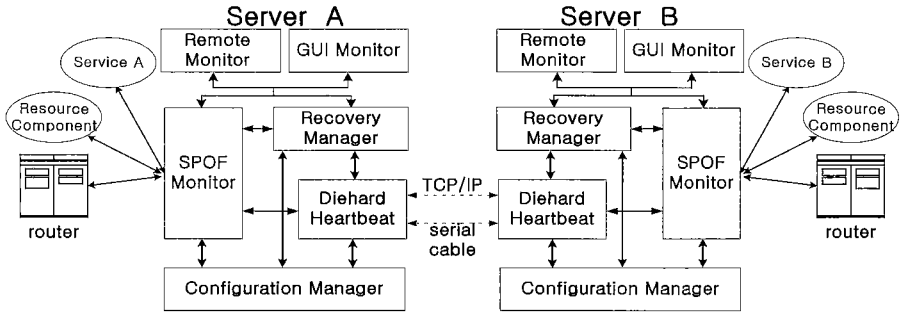


그림 5 Diehard 시스템 구성

컴퓨터의 IP 주소로 서비스한다.

Diehard Heartbeat 자체도 소프트웨어도 SPOF가 될 수 있다. 만약 서비스는 정상적으로 수행되는데 Diehard Heartbeat이 정상적으로 작동하지 않는 경우에 split-brain 문제[3]가 발생할 수 있다. 이러한 문제를 해결하기 위해서 Diehard에서는 Heartbeat을 모니터링하고, Diehard가 갑작스럽게 중지되는 경우에 다시 가동시켜주는 작업을 수행하는 Back-up 기능을 제공하고 있다.

4.2 SPOF 모니터

SPOF 모니터는 서비스의 상태와 컴퓨팅 자원들의 상태를 체크하는 역할을 담당한다. Diehard Heartbeat은 서버가 다운되었는지 여부를 모니터링하고, SPOF 모니터는 서버가 서비스하는 각 서비스와 이에 따라 필요한 컴퓨팅 자원들(서버 프로세스, 네트워크, 파일 시스템 등)의 상태를 주기적으로 모니터링한다.

SPOF 모니터는 HTML 페이지, 그림 파일 등의 정적인 자원들을 위해서는 해당 파일들이 정상적으로 존재하는지 여부를 판단하고, 응용 프로그램이나 서비스 데몬 등의 동적인 자원들을 위해서는 프로세스의 상태를 주기적으로 체크하는 것은 물론 서비스 데몬에 요청을 전달해서 정상적으로 처리되는지 여부를 판단한다. 만약 모니터링에서 서비스나 자원에 문제가 있다고 판단되면, 실제로 문제가 발생한 것인지 아니면 임시적으로 시스템 과부하 때문에 발생한 문제인지를 파악하기 위해서 즉시 재 테스트를 수행한다. 재 테스트에서도 역시 문제가 발생하면 문제가 있는 내용들을 구성 관리자에서 등록된 내용에 따라 "경고", "에러", "치명적 에러"라는 3가지 범주로 분류한다. 경고는 중요하지 않은 HTML 페이지나 그림 파일 등이 없는 경우이고, 에러는 중요하지 않은 응용 프로그램이 동작하지 않는 경우이다. 치명적 에러는 주 서비스가 동작하지 않는 경우이다. 에러와 치명적 에러가 발생하는 경우에 SPOF 모니터는 이 사실을 복구 관리자에 전달한다.

SPOF 모니터는 주기적으로 서비스와 자원 상태를 네트워크로 연결된 GUI 모니터에 전달한다. GUI 모니터에 전달되는 정보는 다음과 같은 것들이 있다.

- 모니터링되고 있는 서비스들 목록
- 서비스가 제공되고 있는 호스트 이름
- 서비스되고 있는 포트 번호
- 서비스들의 현재 가용성 상태

SPOF 모니터는 또한 서비스되는 데이타들에 대해 해커에 의해 데이타가 변경되었는지 여부를 체크한다. SPOF 모니터는 Diehard가 처음 시작될 때 중요한 데이타들과 파일들의 마지막 변경된 시간을 기억하고 있다가 해커에 의해 데이타가 변경되거나 삭제되면 관리자에게 알리고, 서비스를 다른 컴퓨터에게 넘겨주고, 시스템을 종료한다. 관리대상이 되는 중요 파일들로는 서비스를 위해 꼭 필요한 파일이나 서비스 첫 화면이 될 수 있고, 이 파일들은 구성 관리자에 등록할 수 있다.

4.3 복구 관리자

복구 관리자는 서비스나 자원에 에러가 발생한 경우에 문제를 해결하기 위한 모듈이다. 이 모듈은 SPOF 모니터로부터 에러가 발생한 부분을 찾고, 구성 관리자에 등록된 복구 정책을 결정한다. 복구 정책은 "무시", "재시작", "fail-over"로 나누어져 있다.

- 무시 - 자원 실패를 무시한다.
- 재시작 - 문제가 있는 서비스를 중지시켰다가 다시 시작시킨다.
- fail-over - 서비스를 다른 컴퓨터에 넘겨준다.

복구 관리자는 fail-over가 발생하는 경우에 서비스를 어느 컴퓨터에 넘겨줄 것인지 결정한다. 어느 컴퓨터가 서비스를 받을 것인지는 구성 정보와 동적인 시스템 부하에 따라 결정된다. fail-over가 일어날 때 하드디스크와 데이타 공유 등의 문제 때문에 클러스터에 포함된 모든 컴퓨터가 서비스를 인계받을 수 있는 것은 아니다. 따라서 서비스를 인계받을 수 있는 컴퓨터들은 서로 구성 정보에 이 내용들을 기술하고 있다. 서비스를 인계받을 수 있는 컴퓨터가 많은 경우에는 fail-over가 발생하는 시점에서 시스템 부하가 가장 적은 컴퓨터로 서비스가 넘어 간다.

4.4 GUI 모니터

GUI 모니터는 클러스터 밖의 다른 시스템에서 실행되며 현재 서비스가 이루어지고 있는 서버의 상태, 서비스 중단 상태, failover 과정 등을 화면에 그래픽으로 보여줌으로써 시스템 관리자가 시스템의 현재 상태를 알 수 있도록 해준다.

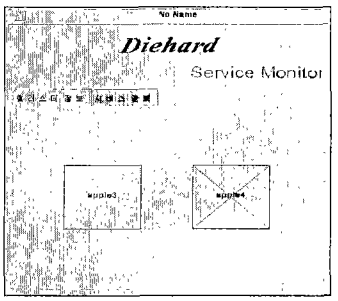


그림 6 클러스터 정보

GUI 모니터가 제공하는 정보는 크게 두 가지 부분으로 나누어 볼 수 있다. 첫 번째는 그림 6의 클러스터 부분으로 각 클러스터의 서비스 상황을 나타내어 주는 부분으로서 각 클러스터의 가용성을 표시하여 준다. 두 번째는 그림 7의 서비스 부분으로 현재 서버에서 제공하고 있는 서비스들의 가용성을 표시하여 준다.

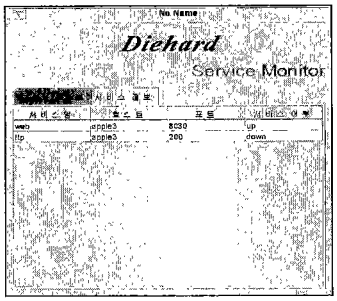


그림 7 서비스 정보

4.5 원격 모니터

컴퓨팅 자원들은 로컬과 원격에서 모니터링되어야 한다. 가용성 실패의 75%는 로컬에서 감지되지 않는다[21]. 이유는 대부분의 문제가 방화벽 밖의 네트워크에서 발생하기 때문이다. 이러한 이유 때문에 Diehard에서는 로컬에서 모니터링하는 SPOF 모니터 이외에 네트워크 외부에 존재하면서 서비스를 감시하는 원격 모니터를 두고 있다. 원격 모니터는 네트워크 외부에서 주기적으로 서버에 요청을 보냄으로서 서비스 가용성을 체크한다. 원격 모니터는 전적으로 외부적으로 드러나는 서비스만 감시하고, 서비스를 위해 내부적으로 필요한 자원들은 감시하지 않는다. 원격 모니터 측면에서는 서비스가 중단된 것과 네트워크 혹은 시스템 로드로 인해서 서비스가 느리게 처리되는 것을 구별할 수 없다. 따라서 원격 모니터는 사용자 입장에서 서비스 가용성을 판단할 수 있는 중요한 정보를 제공하기 때문에 서비스가 중단된 경우에 시스템 관리자에게 즉시 알리고, 중단된 서비스 내용을 로그 파일에 저장한다.

4.6 구성 관리자

구성 관리자는 Diehard 시스템을 위한 구성 파일을 설정해주는 역할을 한다. 구성 정보는 시스템에 대한 정보들과 쉘 스크립트 프로그램들로 구성되어 있다. 구성 관리자는 시스템 정보는 물론 간단한 쉘 프로그램도 자동적으로 생성해주는 역할을 한다. 구성 관리자를 이용해서 구성되는 정보들로는 다음과 같은 것들이 있다.

- 클러스터의 노드 정보
- fail-over 가능한 노드 그룹 정보
- 컴퓨팅 자원에 따른 에러 수준 정보
- 에러 복구 정책 정보
- 해킹에 대비해서 변경되어서는 안되는 파일 정보
- 기타

그림 8은 구성 관리자를 통해서 시스템 정보를 입력하는 모습이다.

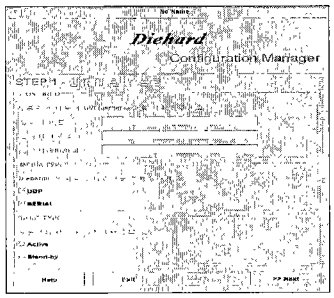
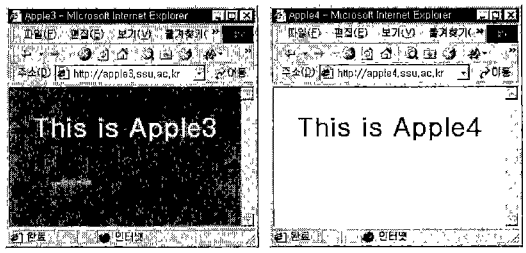


그림 8 Diehard Configuration 예

5. 실행 결과

Diehard에서 fail-over가 제대로 작동하는지를 확인하기 위해서 apple3과 apple4라는 두 대의 컴퓨터에서 웹 서비스를 제공하고 있다. apple3과 apple4가 정상적으로 수행하는 경우에 그림 9와 같이 서비스를 제공한다. 만약 apple4에 문제가 발생하는 경우에 apple3은 apple4의 서비스를 인계받아서 수행하게 된다. 그림 10은 apple3이 apple4를 위해서 서비스하는 것을 보여준다.

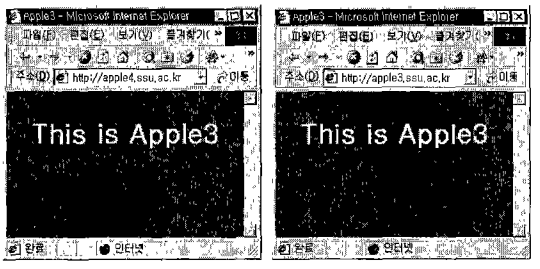


(a) apple3의 정상적인 서비스 (b) apple4의 정상적인 서비스

그림 9 정상적으로 웹 서비스를 제공하는 경우
(Fail-over가 일어나기 전)

그림 10은 테스트 결과를 시각적으로 확인하기 위해서 apple3과 apple4가 동일한 내용을 서비스하도록 하였지만, 실질적으로는 apple3에서는 웹서비스, apple4에

서는 데이타베이스 등의 서비스를 제공할 수 있다. 만약 apple4가 복구되어서 서비스를 수행할 수 있다면 fail-back 알고리즘을 통해서 그림 10 상태에서 그림 9의 상태로 전이된다.



(b) apple4의 서비스가 apple3 (a) apple3의 정상적인 서비스으로 fail-over된 경우

그림 10 Fail-over가 일어나서 apple3가 apple4 서비스를 동시에 제공하는 경우

6. 결론 및 향후 연구 과제

본 논문에서는 active-active 형식의 N-way 고가용성 클러스터 시스템을 위한 heartbeat, fail-over, fail-back 알고리즘을 소개하고, 이 알고리즘에 따라 구현된 Diehard 고가용성 시스템을 소개하였다. Diehard는 클러스터에 포함된 N대의 컴퓨터들이 쉬지 않고 효율적으로 서비스를 제공할 수 있다. 또한 Diehard는 인터넷 서비스의 특성을 고려해서 소프트웨어 SPOF를 모니터링하고, 부분적인 서비스 중단에 대한 다양한 복구 정책 등을 지원하도록 설계 및 구현되었다.

Diehard는 Diehard Heartbeat, SPOF 모니터, 복구 관리자, 구성 관리자, GUI 모니터, 원격 모니터로 구성되어 있다. Diehard는 운영체제 위에서 구현되었으며, GUI 모니터와 원격 모니터는 자바 언어를 이용하여 구현되었다. 그리고 다른 부분들은 C 언어와 POSIX 멀티 쓰레드를 이용해서 구현되었다.

인터넷 서비스와 같은 네트워크 기반의 시스템에서 사용자는 시스템의 실패와 네트워크 혹은 서버의 부하에 의해 서비스가 느려지는 것을 구별할 수 없을 뿐만 아니라, 기다리는 동안은 일종의 서비스 실패로 취급된다 [22]. 따라서 진정한 고가용성을 지원하기 위해서는 시스템의 고가용성뿐만 아니라 빠른 서비스가 가능하도록 지원하여야 한다. 향후에는 Diehard와 고성능 웹서버 및 리눅스 가상 서버와의 연계를 연구할 예정이다.

참 고 문 헌

[1] Gregory F. Pfister, *In search of Clusters*, Prentice Hall PTR, 1998.
[2] Linux High Availability HOWTO, http://metalab.unc.edu/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html
[3] Evan Marcus, Hal Stern, *Blueprints for high Availability*, Wiley & Sons, 2000.
[4] Peter S. Weygant, *Clusters for High Availability*, Prentice Hall PTR, 1996.
[5] Anupam Bhide, "A High Available Lock Manager For HA-NFS," in *Usenix Conf. Procdeedings*, pp. 177-184, Jun. 1992.
[6] M. Kamath, et al, "Providing High Availability in Very Large Workflow Management Systems," in *The 5th International Conference on Extending Database Technology (EDBT '96)*, Mar, 1996.
[7] High-availability.com, *RSF-1 Technical White-paper*, Whitepaper, 1998. 8.
[8] High-availability Linux Project, http://linux-ha.org/
[9] Heartbeat, http://linux-ha.org/comm/#Heartbeat
[10] Fake: Redundant Server Switch, http://linux.zipworld.com.au/fake/
[11] Heart a redundant, distributed cluster technonogy, http://www.lemuria.org/Heart/
[12] Software-RAID HOWTO, http://www.linas.org/linux/Software-RAID/Software- RAID.html
[13] The Global File System, http://gfs.lcse.umn.edu/
[14] The Logging Filesystem, http://hp.cso.uiuc.edu/~c-cook/prof/lfs
[15] Coda File System, http://www.coda.cs.cmu.edu/
[16] MON, Service Monitoring Deamon, http://www.kernel.org/software/mon/
[17] PIKT, http://pikt.uchicago.edu/pikt/
[18] nocol, http://www.netplex-tech.com/software/nocol/
[19] 최재영, 최종명, 김은희, 김민석, "고가용성 리눅스", *정보처리학회지, 제6권, 제6호*, pp.19-25, 1999.
[20] 정갑주, 이환묵, "고가용성 웹서버: Primary-Backup 설계 방법", *한국 정보과학회 컴퓨터시스템 연구회, 추계 학술발표회 논문집*, pp.112-119, Sep. 2000.
[21] red-alert keynote, http://www.redalert.com/
[22] Alan Wood, "Predicting Client/Server Availability," in *IEEE Computer*, pp.41-48, April 1995.
[23] Rodney Fountain, Richard Braithwaite, Philip Joyce, "Teaching Electronic Commerce: A New Focus For Business Computing," in *International Conference on Software Engineering: Education & Practice*, 1998.

최 종 명

1992년 숭실대학교 전자계산학과 학사. 1996년 숭실대학교 전자계산학과 석사. 1997년 ~ 현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 고가용성 시스템, 시각프로그래밍, 멀티패러다임 시스템

한 주 현

2000년 숭실대학교 컴퓨터학부 학사. 2000년 ~ 현재 숭실대학교 컴퓨터학과 석사과정. 관심분야는 분산/병렬 시스템, 시스템 소프트웨어, 클러스터링, 한글폰트기술

최 재 영

1984년 서울대학교 학사(제어계측공학). 1986년 미국 남가주대학교 석사(컴퓨터공학). 1991년 미국 코넬대학교 박사(컴퓨터공학). 1992년 1월 ~ 1994년 2월 미국 국립 오크리지연구소 연구원. 1994년 3월 ~ 1995년 2월 미국 테네시 주립대학교 연구교수. 1995년 3월 ~ 현재 숭실대학교 컴퓨터학부 부교수. 관심분야는 병렬/분산처리, 클러스터링, 시스템 소프트웨어