Diehard: N-웨이 웹 클러스터

임재훈 최종명 최재영
숭실대학교 컴퓨터학과
{suplim, jmchoi , choi}@ss.ssu.ac.kr

Diehard: N-way Web Cluster

Jae-hoon Lim, Jong-myoung Choi, Jae-Young Choi
School of Computing, Soongsil University

요 약

인터넷 서비스를 위한 시스템은 지속적인 서비스가 이루어져야 하므로, 보다 효율적으로 중단 없는 인터넷 서비스의 확장성을 제공하기 위한 고가용성 시스템의 필요성이 증가하고 있다. 기존에 많은 연구가 이루어진 Active-Active형태의 클러스터 시스템은 단순히 두 대의 시스템만으로 운용되기 때문에 확장이 용이하지 못하다. 그래서 단순하게 시스템의 증가만으로 클러스터를 확장할 수 있는 N-way 방식의 클러스터를 지원하는 방식이 필요하게 된다. 본 논문은 이러한 확장을 고려하여 소프트웨어적으로 N개의 시스템을 클러스터로 구축하기 위해 개발된 알고리즘을 제시한다.

1. 서 론

인터넷의 폭발적인 증가와 더불어 웹(WWW)을 통한 서비스의 증가로 특정 서비스를 제공하는 서버에 많은 부하가 발생하여 원활한 서비스를 제공하지 못하게 되는 경우가 종종 발생하고 있다. 이에 따른 심각한 경제적 손실을 초래할 수 있다. 그림 1 에서 서비스 실패를 발생원인의 대부분은 소프트웨어의 실패에 기인한다[2].

이러한 문제를 제거하기 위한 방법의 하나로, 저 비용으로 클러스터 컴퓨터들로 구성하여 하드웨어, 소프트웨어 혹은 네트워크 등에 문제가 발생하더라도 서비스를 지속적으로 제공할 수 있도록 고가용성(High Availability)을 제공한다.

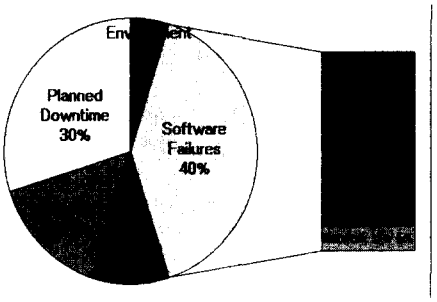

그림 1. 서비스 실패의 원인

본 논문에서는 이러한 고가용성을 제공하기 위한 기존의 Diehard 시스템 (Active-Active방식의 클러스터[3])을 확장하여 N개의 시스템으로 클러스터를 구성하도록 구현한다. 2장에서는 고가용성에 관한 연구를 간단히 소개하고, 3장에서는 Active-Active형태의 클러스터 시스템을 위한 N-way Diehard 시스템의 구조 및 구현 알고리즘을 소개한다. 마지막으로 4장에서는 결론과 추가적으로 연구되어야 할 과제를 소개한다.

2. 관련 연구

최근 공개 프로젝트로서 고가용성 솔루션을 개발하고 제공하는 Linux-HA[4]를 소개한다. 현재 Heartbeat[5], Fake[7] 등 고가용성의 핵심 모듈을 개발하였고 Mon[8] 등의 모니터링 시스템과 밀접하게 연계되어 개발되고 있다.

Heartbeat은 system의 결함을 검사하기 위한 기능을 하여, 결함 발생시 결함이 발생한 system의 IP 주소를 다른 system이 넘겨받는 기능을 수행한다. Fake는 ARP spoofing기술을 이용하여 가상IP를 이용하여 다운된 시스템의 IP주소를 넘겨받는 기능을 수행한다.

그러나, Linux-HA는 각 소프트웨어가 개별적으로 개발되어 왔기 때문에 완성도가 떨어지며, 통합하여 사용하기에 쉽지 않으며 사용 제품에 비하여 기능과 성능 면에서 미약하고 사용상의 편의를 제공하지 못하는 단점이 있다.

3. 설계 및 구현

3. 1 Diehard 시스템 구조

기본적으로 Diehard 시스템은 Active-Active형태의 클러스터를 기본으로 N-way로 확장된다.

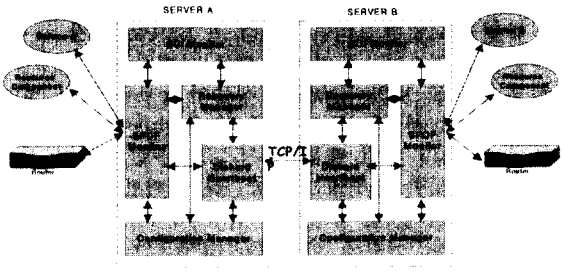

그림 1. Diehard 시스템 구성도

그림 1은 Active-Active기반의 고가용성 클러스터인 Diehard 시스템의 구성을 보여준다. 각 시스템은 TCP/IP로 서로 통신을 하게 되고, Heartbeat, SPOF 모니터[1], 복구 관리자, GUI Monitor, Configuration Manager로 구성되어 서로 간의 관계를 보여준다.

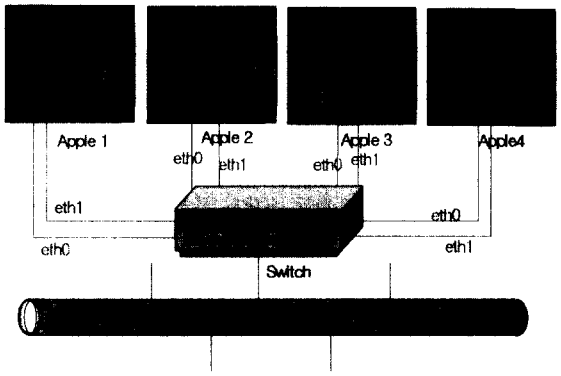

그림 2. N-way Diehard 네트워크 구성도

Diehard System의 특징은 소프트웨어로서 고가용성을 실현하는데 있다. 그림 2는 Diehard의 기본구성으로 4대의 서버가 스위치 장비에 연결되어 있고, 각 서버는 2대의 이더넷 카드만을 사용함으로서 저 비용으로 효율적인 클러스터를 구성하기에 적합하도록 구현되었다. 또한 효율적인 알고리즘을 고안하여 적용함으로써 자동적으로 부하 분산을 이루고, 다운된 시스템을 클러스터 시스템에서 제외시켜 동적으로 클러스터를 새롭게 구성하여 안전하게 지속적인 서비스를 제공하도록 구현되었다.

또한, 기본적으로 LAN환경에서 시스템간에 메시지를 전송함으로써 시스템과 네트워크의 부하를 최소화하는 방향으로 클러스터 환경을 구축하였다. N-way에서의 두 가지방식의 메시지 전송방식을 사용하게 되는데, 우선 Ring Heartbeat은 이웃한 시스템 노드에 대한 Fail을 확인하기 위해 패킷을 Unicasting하는 방식을 사용하고, 또 다른 하나는 Broadcasting방식으로 시스템 상태를 다른 시스템에게 전달하거나 Fake하기 위한 수단으로 이용된다. 이러한 전달 방식은 Diehard 클러스터를 적절하게 제어하기 위한 수단이 된다.

3. 2 알고리즘

각 시스템은 Active-Active형태를 기본으로 Heartbeat 패킷을 단방향으로 전송하며, N-way로 확장된 Diehard는 Ring형태의 Heartbeat 패킷을 사용하게 된다.

알고리즘 1 은 Heartbeat을 체크하는 알고리즘으로 상대 시스템의 활동 상태를 점검하게 된다. 전송된 HELLO 패킷의 일정한 횟수를 체크하여, 이웃 시스템 노드의 상태를 감지하게 된다. 일정한 횟수에 미치지 못하면 모든 시스템들에게 FAIL_OVER 패킷을 전송하게 되고, 클러스터 내의 특정한 시스템이 Controller가 되어 부하 분산을 고려하여 Fake를 수행하기 위한 절차를 시행한다[1].

Heartbeat Algorithm

```
begin
  while( forever )
    send HELLO (heartbeat packet) to neighbor
    receive HELLO from neighbor
    if(dont receive HELLO packet more than 3 count)
      send FAIL_DOWN packet to all cluster
      start to fake
    endif
  end while
end
```

[알고리즘 1]

알고리즘 2는 시스템이 다운되면 이웃한 시스템이 이 사실을 클러스터 안의 모든 시스템들에게 FAIL_DOWN 패킷을 보내 알리게 된다. 클러스터의 최초 구성에서 특정 시스템은 클러스터를 관리하는 역할을 맡게 되는데, 시스템은 클러스터내의 부하를 고려하여 부하가 적은 시스템에게 다운된 시스템의 서비스를 대신해 떠맡도록 TAKE_OVER 패킷을 전송한다[6]. 이 패킷을 받은 시스템은 다운된 시스템의 서비스를 대신하게 되며, 이러한 Fake 정보는 다른 모든 시스템에게 broadcating하여 알리게 된다.

Fail-over Algorithm

```
begin
  broadcast the node fault information to all nodes in the cluster
  receive FAIL_DOWN packet from some neighbor of the failed node
  if (this node == the node of high priority in the cluster)
    check the load all node in cluster in table of state of fake
    if (this node is not faking)
      start fake procedure
      send TAKE_OVER packet all node for table of state of fake
    else
      send FAIL_OVER packet to the node of lowest priority
    endif
  else
    if (receive FAIL_OVER packet from controller)
      start fake procedure
      start service
    endif
  endif
  rebuild the cluster dynamically
end
```

[알고리즘 2]

```
Rebuild cluster Algorithm
begin
  if (receive IP_CHECK packet from neighbor is alive)
    if (failed node is my right neighbor)
      change my right neighbor
      send I_AM_LEFT packet to node of right direction
    elseif (failed node is left neighbor)
      change my left neighbor
      send I_AM_RIGHT packet to node of left direction
    endif
  elseif (receive I_AM_RIGHT packet from left neighbor)
    change my right neighbor
  elseif (receive I_AM_LEFT packet from right neighbor)
    change my left neighbor
  endif
end
```

[알고리즘 3]

알고리즘 3 의 경우는 시스템이 다운되거나 다시 클러스터로 복구되었을 경우 클러스터를 새롭게 구성하도록 한다. 각 시스템은 초기 구성에서 IP_CHECK 패킷과 시스템 구성 파일로 클러스터의 시스템구성과 자신의 이웃한 시스템을 알게된다. 시스템이 다운되면 클러스터 내에서 제거되고 클러스터를 재구성하게 된다. 이때 다운된 시스템의 양쪽 이웃 시스템간 I_AM_LEFT 패킷과 I_AM_RIGHT 패킷을 주고받음으로서 새로운 클러스터를 구성을 하게 된다.

```
Fail-back Algorithm
begin
  check diehard configuration
  boot the system with a internal IP
  if (receive IP_CHECK packet from node is alive)
    if (faking failed nodes service)
      send FAIL_BACK packet to node is alive
      remove failed nodes service
    endif
  elseif (receive FAIL_BACK from node is taking over)
    setup external IP
    start services
    send FAILBACK_OK packet to all node for update
state of fake
  endif
end
```

[알고리즘 4]

알고리즘 4 는 Fail-back알고리즘으로서 다운되었던 시스템이 다시 살아나서 자신의 서비스를 다른 시스템이 대신하고 있는지 즉, 자신의 외부 IP를 다른 시스템이 사용하고 있는지를 검사하게된다. IP_CHECK 패킷을 모든 시스템에 보내게 되고, 만약 다운된 시스템을 대신해서 서비스를 하고 있는 시스템은 FAIL_BACK 패킷을 살아난 시스템에게 전달하고, 이를 받은 살아난 시스템은 모든 시스템에 FAIL_BACK_OK 패킷을 전달하며 클러스터에 복귀하게 된다.

N-way 고가용성 클러스터인 Diehard는 위 4가지의 알고리즘을 결합하여 상호 동작함으로서 결과적으로 소프트웨어적인 고가용성을 실현하게 되고 또한, 확장성을 고려하여 쉽게 시스템을 추가할 수 있도록 설계되었다..

4. 결론

고가용성 클러스터는 인터넷 시대의 필수적인 사항이 되었다. 클러스터는 저 비용으로 고가용성을 실현할 수 있는 좋은 대안으로 등장하였다. Diehard는 N개의 시스템을 클러스터로 구성함으로서, 하나의 단일한 시스템인 것처럼 구성할 수 있게 하는 소프트웨어이다. Diehard 시스템은 효율을 고려하여 C언어 기반으로 소켓 프로그램[10]과 POSIX 쓰레드[11]로 구현되었다. 모든 작업이 소프트웨어로 이루어지므로 추가적인 하드웨어 개발이 필요하지 않는 장점이 있고, 효율적인 N-way 알고리즘의 개발로 충분히 고가용성 클러스터를 구현하였다. 그러나 진정한 고가용성을 구현하려면 모니터링 툴과 같은 추가적인 소프트웨어 개발이 필요성이 절실하다. 마지막으로 제품으로 상용화된 많은 소프트웨어가 있으나 Linux-HA 공개 소프트웨어를 기반으로 독자적인 소프트웨어를 개발함으로써 국내 소프트웨어 산업에 도움이 되리라 본다.

5. 참고 문헌

[1] Gregory F. Pfister, In Search of Cluster, Prentice all PTR, 1998.

[2] Evan Marcus, Hal Sterm, Blueprints for High Availability, WILEY, 2000.

[3] Linux High Availability HOWTO, http://www.linuxdoc.org/HOWTO/HOWTO-INDEX/index.html

[4] High-availability Linux Project, http://linux-ha.org/

[5] Heartbeat, http://linux-ha.org/comm/#Heartbeat

[6] Failover, http://www.linux-ha.org/failover/

[7] Fake: Redundant Server Switch, http://linux.zipworld.com.au/fake

[8] Mon: Service Monitoring Deamon, http://www.kernel.org/software/mon/

[9] 최재영, 최종명, 김은희, 김민석, "리눅스 기반 고가용성 클러스터(Diehard)에서의 Active-Active 방식의 설계 및 구현", 한국정보과학회 컴퓨터 시스템 연구회, 추계 학술발표회 논문집, pp.120~126, 2000년 9월

[10] W. Richard Stevens, UNIX NETWORK PROGRAMMING Volume 1, Prentice Hall PTR, 1997

[11] David R. Butenhof, Programming with POSIX Threads, Addison Wesley, 1997