2000년도 대한전기학회 하계학술대회 논문집 2000.7.17-20

고가용성 클러스터링 가상서버의 로드밸런서를 위한 고장극복 기법에 관한 연구

*홍 태 화, *구 본 준, *김 학 배, **곽 태 영, **강 신 준
*연세대학교 전기·컴퓨터공학과, **(주) ACS Technology

A Failure-Recovery Method in the Load Balancer of a Clustering Virtual Server with High Availability

*Hong Taehwa, *Koo Bonjun, *Kim Hagbae, **Kwak Taeyoung, **Kang Shinjoon
*Dept. of Electrical & Computer Engineering, Yonsei University
**ACS Technology

Abstract - 최근에 인터넷의 급격한 수요증가로 인하여 웹서버의 고가용성(high availability)이 절실히 요구되고 있다. 이를 위한 방안으로 클러스터링 가상 서버가 핫이슈로 대두되고 있는 상황에서 이의 가장 핵심부분인 로드밸런서(load balancer)의 고가용성을 위해 고장극복(fault-tolerant) 기법 연구는 필수적이라 할 수 있다. 본 연구에서는 클러스터링 웹서버의 구성과 로드밸런서의 운영방안을 제시하고 특히, 로드밸런서가 고장났을 경우 로드밸런서의 작업을 신속하게 대체하는 방안을 모색한다. 로드밸런서의 구성이 마스터 로드밸런서와 백업 로드밸런서로 구성된다는 가정 하에 백업 로드밸런서가 마스터 로드밸런서의 작업을 신속히 대체하는 방안을 위해 체크포인트(checkpoint) 기법을 적용한다.

핵심어: 고가용성, 클러스터링 가상서버, 로드밸런서, 고장극복, 체크포인트

1. 서 론

근래에 들어 급속한 인터넷 보급에 의한 웹서비스 트래픽의 폭발로 웹서버들에 대한 부하(load)는 엄청난 속도로 증가되고 있다[1]. 요사이 운영되고 있는 대중적인 웹사이트들은 공통적으로 하나의 가상 URL 인터페이스를 유지하면서 분산 서버 구조를 이용하여 이와 같은 수많은 클라이언트의 접속 요청을 효율적으로 서비스하고 있다. 이러한 방법은 클라이언트에 대해 마치 하나의 서버에만 접속한 것처럼 투명성(transparency)을 공급하고 서버측에 대해서는 확장성을 보장할 수 있다[2]. 이와 함께 웹서버의 트래픽 분산 문제를 해결하기 위해 많은 솔루션들이 연구되고 있고 웹서버의 운영 구조와 방법이 큰 이슈가 되고 있는데 이에 대한 범용적 솔루션으로 고가용성과 확장성을 보장할 수 있는 클러스터링 가상서버가 크게 각광받고 있다.

확장성을 목적으로 하는 웹서버의 구조는 웹사이트들의 증가되는 부하를 모두 처리할 수 있어야 한다. 그러나 현재 멀티프로세서에 의한 웹서버 어플리케이션의 경우, 이러한 웹서비스의 확장성을 만족하지 못하는데, 그것은 TCP/IP가 웹서버 어플리케이션에 대해 멀티프로세서의 처리속도를 심하게 제한하는 single-thread 형태이기 때문이다. 따라서 부하를 공유하는 독립된 서버들로 구성된 분산구조가 확장성 웹서버를 위해 보다 적절한 방안이 된다[3]. 웹서버의 고가용성을 증가시켜 클라이언트에게 request latency를 줄이기 위해 제안된 많은 솔루션들이 있다[4]. 이러한 솔루션들은 크게 서버단과 클라이언트단에 적용된 솔루션으로 구분지을 수 있는데, 서버단의 솔루션은 LAN(local area network)을 통해 웹서버의 클러스터를 구성하고 그 상위에 또 하나의 서버를 두어 클라이언트의 요구를 웹서버 클러스터에 분산하는 방식을 사용한다. 한편 클라이언트단의 솔루션은 클라이언트 캐싱(caching), proxy 캐싱, pre-fetching 등을 이용하여 클라이언트에게 request latency를 줄이는 방법을 사용한다[4]. 이들 중 서버단에 대한 연구가 주류를 이루고 있고 많은 장점을 가지고 있다.

클러스터링 웹서버의 가장 효시라고 할 수 있는 NCSA 서버는 하나의 클러스터로 묶여진 웹서버들이 존재하고 이들은 AFS(Andrew File System)를 사용하여 HTML 문서들을 공유한다[1,3]. 또한 RR-DNS(Round-Robin Domain Name Server)를 이용하여 클러스터 안의 각 서버에 클라이언트의 접속요청을 분산하게 된다. 특히 클라이언트들은 하나의 URL로 접속하고 이를 받아들인 RR-DNS가 서로 다른 IP 주소를 가진 웹서버에 RR 방식으로 접속시켜 주는 것이다. 이와 같은 방식으로 웹서버들 사이에 부하가 분산되고 웹서버들의 각각은 AFS에 접속하여 자신들이 원하는 서비스를 수행하게 되는 것이다[1,4]. 그러나 이러한 솔루션은 클라이언트와 RR-DNS 사이에 존재하는 네임 서버 캐싱에 의해 보다 높은 수준의 부하 분산이 수행되기 어렵고 또한 웹서버에 고장이 발생했을 경우, 고가용성을 보장하기 힘들다[1].

따라서 기존의 클러스터링 웹서버를 운영함에 있어서 이들이 고가용성을 보장하기 위해 고장포용 기법을 적용하는 방안이 매우 중시되고 있다. 현재 리눅스 HA 사이트를 통해 'Heartbeat'과 'Fake'라는 로드밸런서의 고장진단 및 고장극복 방안이 제시되어 있다[7]. 본 연구에서는 이를 수용하고 체크포인트를 이용하여 백업 로드밸런서가 메인의 정보를 그대로 유지함으로써 클러스터링 웹서버가 보다 신속하고 신뢰성 있는 서비스를 보장하는 방안을 제시하고자 한다.

2. 본 론

2.1 클러스터링 가상서버의 구성

클러스터링 가상서버(clustering virtual server)는 하나의 시스템처럼 행동하도록 하기 위한 독립적인 컴퓨터들의 집합을 의미한다. 웹서비스와 같이 대량의 트래픽을 처리해야 하는 경우, 많은 서버가 필요하지만 서버가 많은 시스템으로 구성되었을지라도, 각 서버가 IP를 가지고 서비스를 수행한다면 하나의 서버가 고장을 일으켰을 경우, 그 서버에 대한 서비스는 중단되어 전체 서버에 대한 고가용성(high availability) 및 고성능 서비스를 수행하지 못할 것이다. 이에 반해, 클러스터링 가상서버는 사용자에게는 단일 서버 또는 단일 서버 이미지를 갖게 하여 클러스터를 하나의 서버인 것처럼 간주하게 하기 때문에 고가용성이 높은 서비스를 수행할 수 있다. 또한 클러스터링 가상서버는 구조의 특성상 서버의 용이한 확장이 가능하여 사용자 요청이 증가하거나 복잡한 작업을 처리하기 위하여, 필요에 따라 여분의 시

본 연구는 2000년 정보통신연구진흥원 산업기술개발사업의 지원 하에 진행되었습니다.

스템이 추가될 수 있다. 클러스터의 한 시스템이 에러를 일으키면, 이 시스템의 작업은 자동적으로 다른 시스템에 분산되며, 클러스터는 사용자에 대해 투명하다. 이러한 클러스터링 기술은 저가의 PC를 이용하여 고가용성 및 고성능의 시스템을 위해 널리 활용되고 있는 실정이다.

본 연구에서는 그림 1과 같은 구조의 클러스터링 가상 서버를 고려한다. 클러스터 시스템은 다양한 아키텍처로 구성될 수 있으나, 일반적으로, 앞단에 부하를 분산해 주는 로드밸런서(load balancer)와 그에 딸린 여러 대의 실서버(real server)들로 구성되어 있다. 클라이언트가 서비스 요청을 하면 로드밸런서가 모든 요청을 받아들이고 적당한 스케쥴링 알고리즘에 의해 클라이언트의 서비스 요청을 실서버에 분산한다. 물론 클라이언트는 최종적으로 연결된 실서버와 직접적으로 연결된 것으로 판단하게 된다. 이러한 작업을 수행하는 과정에서 로드밸런서는 각 실서버의 상태를 파악하고 있어야 하고 이러한 정보는 실제 클러스터링 가상서버에 중대한 영향을 미친다. 특히, 급격한 부하 증가, 시스템의 오동작, 통신 장애 등 열악한 서버 환경에서는 고장발생률이 매우 높아지므로 고장났을 경우 전체 시스템의 가용성에 치명적인 영향을 미치는 로드밸런서의 고장복구 문제는 매우 중대한 사안이다. 이러한 문제를 해결하기 위해 메인 로드밸런서와 더불어 백업 로드밸런서를 hot-standby 상태로 두어 백업 로드밸런서가 메인 로드밸런서의 작업을 대신 수행하는 방안이 이용되고 있다. 이와 함께 본 연구에서는 메인 로드밸런가 고장났을 경우 백업이 메인 로드밸런서가 실서버에 대해 가지고 있던 정보를 신속하게 상속받는 방안을 제시한다.

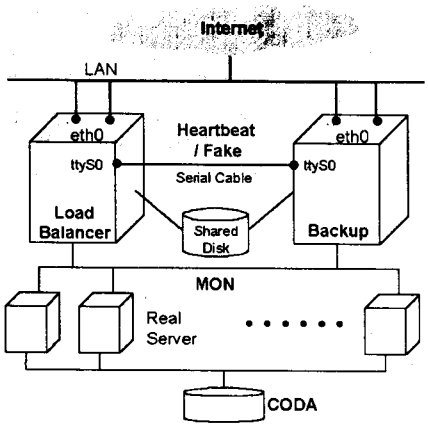

그림 1. 클러스터 웹 서버의 구성도

2.2 로드밸런서의 고장 검출 및 고장 회복 - Heartbeat, Fake

고장포용(fault-tolerant)은 시스템에 고장이 발생했을 때 이를 검출하고 복구하는 작업을 의미한다. 고장포용의 기법은 대표적으로 여분(redundancy)을 두는 공간적 방법과 retry, restart, rollback과 같은 시간적으로 수행되는 방법이 있다[6]. 웹서버와 같이 고가용성을 요구하는 경우에 고장포용기법은 필수적이고 본 연구 대상에는 로드밸런서의 고장포용을 위해 Heartbeat, Fake, Checkpoint 등의 고장포용 기법들이 적용되고 있다. 또한 로드밸런서를 위해 백업 로드밸런서를 적용하고 있다.

로드밸런서는 메인과 백업의 두 동일한 시스템으로 구성된다. 이는 고장포용의 공간여분으로서 메인 로드밸런서가 고장났을 경우 일정한 알고리즘에 의해 백업 로드밸런서가 작업을 이어받게 된다. 따라서 백업 로드밸런서는 메인 로드밸런서의 상태를 항상 확인하고 있어야 한다. 이러한 메인 로드밸런서의 고장검출을 위해 'Heartbeat'이라는 방법이 적용되는데, 앞의 그림 1과 같이 Heartbeat은 메인과 백업 사이에 두 개의 NIC(Network Interface Card)와 하나의 SL(Serial Line)을 통해 수행된다[7]. 즉, 백업은 메인에 3가지 네트워크 통로를 통해 주기적으로 신호를 전달하고 이에 대한 응답을 원하게 된다. 만약 응답이 오지 않았을 경우 백업은 메인이 고장난 것으로 간주하고 자신이 메인의 작업을 대체 수행하게 된다. 이를 위해 'Fake'라는 기법이 적용되는데 이는 Heartbeat을 통해 메인의 고장을 확인한 백업이 메인의 IP주소를 가져오게 된다[7]. 따라서 클라이언트들이 메인의 IP주소로 연결하더라도 백업이 이를 받게 되는 것이다.

2.3 로드밸런서에 체크포인트 기법 적용

위에서 설명된 Heartbeat과 Fake는 앞으로의 클라이언트 요청을 백업 로드밸런서가 수행할 수 있게 하지만, 메인 로드밸런서가 수행하던 기존의 작업에 대한 정보는 무시할 수밖에 없다. 즉 로드밸런서의 핵심적인 기능은 클라이언트의 서비스 요청을 자신이 가지고 있던 실서버들의 정보에 따라 최소 부하 최고 수용력을 가진 서버에 배분하는 역할을 하는데, 고장으로 인해 메인의 기능을 대체하는 백업은 실서버로부터 이러한 정보를 처음부터 다시 받아야만 한다. 이러한 문제를 해결하기 위해 메인은 주기적으로 체크포인트를 실시하여 메인의 정보를 공유 디스크(shared disk)나 또는 자신의 하드디스크에 저장하게 된다.

2.3.1 체크포인트 기법

체크포인트 기법은 일정한 주기로 현재의 상태를 체크하여 그 상태를 안정된 기억장소에 저장하고 고장이 발생되었을 경우 발생시간 바로 전의 체크포인트가 저장한 상태부터 모듈의 작업을 다시 수행하는 일종의 시간여분 고장포용 기법이다[6].

2.3.2 로드밸런서의 고가용성을 위한 체크포인트 알고리즘 적용

로드밸런서의 고가용성을 유지하기 위해 백업 로드밸런서를 두고 백업은 Heartbeat 기법을 이용하여 현재 작업중인 메인 로드밸런서의 상태를 감시하게 되고 동시에 체크포인트 기법을 적용한다. 즉 고장이 발생할 경우, Heartbeat을 통해 메인 로드밸런서의 상태를 감시하고 있던 백업 로드밸런서는 즉시 메인 로드밸런서의 작업 수행 데이터 및 메커니즘을 이어 받아 고장난 시간 바로 전의 체크포인트에 의해 저장된 내용부터 다시 작업을 수행하게 되는 것이다. 이와 같이 로드밸런서 내에 체크포인트 기법을 구현하기 위해서는 다음과 같은 내용을 고려하여야 한다.

- 정보내용과 저장장치

로드밸런서는 주기적으로 각 실서버의 상태를 전달받아 자신의 스케쥴링 표를 갱신한다. 각 실서버들은 자신의 live/dead 상태뿐만 아니라, CPU 점유율, hit 수, 네트워크 속도 등의 정량화된 부하량을 측정하여 이를 로드밸런서에 전달하게 되는 것이다. 한편, 로드밸런서에 체크포인트 기법을 적용하여 일정한 간격을 통해 백업 로드밸런서는 메인 로드밸런서의 스케쥴표를 읽어 안정된 저장장치에 저장하는 것이다. 이러한 과정은 메인에 고장이 발생하더라도 백업이 단순히 메인의 작업을 대신 수행하는 역할만 있는 것이 아니라 메인과 유사한 상태, 즉 고장나기 전 메인이 보유하고 있는 실서버의 상태조차도 어느 정도 복구한 상태에서 클라이언트의 서비스를 수행하고자 하는 것이다.

체크포인트의 구현을 위해 저장장치의 안정성과 빠른 접근 사이의 상충관계가 요구된다. 보통 어떤 시간에 프로그램의 정보들을 저장하기 위해 캐시와 같은 빠른 입출력 시간을 갖는 메모리를 사용하지만, 정보의 중요도를 고려할 때 백업과 메인 사이에 공유디스크를 통한 체크포인팅이 고려되고 있다.

• 최적화된 체크포인트 간격 설정

체크포인트 기법의 적용은 고장의 검출 및 포용 능력을 증대시켜주지만 시간지연이 생기는 것을 고려해야 한다. 따라서 최적의 고장포용을 위해선 체크포인트 간격을 적절히 선정하여야 한다. 체크포인트의 간격은 작업의 평균실행시간이 최소일 때의 간격으로 규정한다. 만약 메인 로드밸런서에 고장이 발생했다면 백업 로드밸런서는 앞에서 정의된 시스템 상태 중 하나에 위치하고 롤백(rollback) 작업이 수행된다. 롤백의 성공적 수행 후 로드밸런서는 다른 상태로 천이되고 이에 따른 상태확률 값은 변화하게 된다.

2.3.3 체크포인트를 포함한 작업흐름의 pseudo code

앞에서 설명했던 것처럼 클러스터링 가상서버의 고장포용을 위해 적용되는 Heartbeat, Fake, Mon, checkpoint, rollback 등의 기법들은 다음과 같은 절차를 통해 연관성을 갖게된다.

```
count == 0; /* 메인으로부터 응답이 없는 회수 체크 */
/* T_h 주기마다 Heartbeat 수행 */
/* T_m 주기마다 mon 수행 */
/* T_c 주기마다 checkpoint 수행 */
do heartbeat per T_h && mon per T_m && checkpoint per T_c
   if (no arrived ack signal from main load balancer)
      count ++;
      if (count == n)
         fake();
         rollback();
      end if
   end if
while (count != n)
```

그림 2. 고장포용기법들을 적용하기 위한 절차

위의 pseudo-code에서 체크포인트의 주기가 가장 길고 Heartbeat의 주기가 가장 짧다. 그리고 메인 로드밸런서가 dead 상태라고 판단하는 근거는 보내진 Heartbeat 신호의 ack 신호가 n번 연속 오지 않았을 경우로 판단하다. (n은 보통 3 정도가 적당하다.) 그리고 메인 로드밸런서가 dead 상태라고 판단되면, 백업 로드밸런서가 즉시, fake를 실행하여 메인 로드밸런서의 IP 주소를 가져오고, 동시에 롤백을 수행하여 기존의 실서버 정보를 복구하여 사용하게 된다.

3. 결 론

신뢰성 있는 웹서비스를 공급하기 위해 클러스터링 가상서버는 가장 인기 있는 솔루션으로서 클라이언트에게는 투명성을, 서버관리자에게는 고가용성과 확장성을 제공한다. 특히, 클러스터링 가상서버를 구성하고 있는 로드밸런서의 고가용성을 유지하는 문제는 가장 중요시되고 있는데, 이를 해결하기 위해 본 연구에서는 고장검출과 고장회복기법으로 제안된 Heartbeat, Fake 기법을 수용하고 복구의 신속성을 위해 체크포인트 기법을 제시하였다. 제시된 체크포인트 기법이 기존의 시간여분 고장포용기법의 대표 격인 롤백보다 간단한 개념이지만 이를 통해 백업 로드밸런서의 신속하고 신뢰성 있는 서비스 상속이 가능하다. 그러나 체크포인트 작업을 수행하는 과정에서 시간적, 공간적 오버헤드가 발생하는 단점을 가지고 있다. 따라서 각각의 체크포인트 시점에서 변화된 값만을 적절히 저장해 진단시간 및 메모리의 요구를 크게 줄인 복구캐시(recovery cache)를 이용해서 시간 지연에 대한 단점을 보완하는 연구가 진행될 필요가 있다.

[참 고 문 헌]

[1] D. M Dias, W. Kish, R. Mukherjee and R. Tewari, "A Scalable and Highly Available Web Server", Compcon '96. 'Technologies for the Information Superhighway' Digest of Papers, pp. 85 -92, 1996

[2] V. Cardellinu, M. Colajanni and P. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-Server Systems", Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on, pp. 528 -535, 1999

[3] A. Mourad and H. Liu, "Scalable Web Server Architectures", Computers and Communications, 1997. Proceedings., Second IEEE Symposium on, pp. 12 -16, 1997

[4] B. Narendran, S. Rangarajan and S. Yajnik, "Data Distribution Algorithms for Load Balanced Fault-Tolerant Web Access", Reliable Distributed Systems, 1997. Proceedings., The Sixteenth Symposium on, pp. 97 -106, 1997

[5] M.T. Kwan et.al, "NCSA's World Wide Web Server: Design and Performance", IEEE Computer, pp. 68-74, 1995

[6] 윤재영, "TMR 시스템에서의 시간여분 고장포용기법 혼용에 대한 연구", 연세대학교 전기공학과 석사학위논문, 1996

[7] http:\\www.linux-ha.org