2000년도 대한전기학회 하계학술대회 논문집 2000.7.17-20

고가용성 클러스터 웹 서버의 로드밸런서에 대한 고장진단기법 연구

이 상 문*, 고 성 준*, 강 신 준**, 곽 태 영**, 김 학 배*
연세대학교 전기·컴퓨터공학과*, ACS Technology**

A Study on Diagnosis Methods for a High Available Clustering Web Server

Sangmoon Lee*, Soungjun Ko*, Sinjun Kang**, Taeyoung Kwak**, Hagbae Kim*
Dept. of Electrical and Computer Eng., Yonsei Univ.*, ACS Technology**

Abstract - 최근 웹의 사용이 일반화되면서 인터넷의 사용자가 급속히 증가하고 있어서, 기존의 단일 웹서버 방식에서는 막대한 접속 트래픽의 수용과 유연한 시스템 확장 등의 문제점이 예상되고 있다. 이와 함께 business-critical한 작업의 경우, 웹서버의 안정성 및 가용성 문제가 가장 중요한 문제로 지적되고 있어서 이러한 문제를 해결할 수 있는 웹전용 서버를 개발이 절대적으로 필요하다.

본 연구에서는 이를 위해, 급격한 트래픽 변화의 수용 및 웹서버의 확장성이 용이한 가상머신 개념과 고신뢰성의 시스템 운영을 위한 고장포용(fault-tolerant)기법을 적용하여 클러스터링 웹전용 서버를 구축하고, 특히 클러스터 웹서버의 부하를 분배해주는 로드밸런서의 고가용성 보장을 위해 heartbeat, fake, mon등의 기법을 이용하여 백업(backup)을 구현한다. 또한, 구현된 시스템의 고성능 및 고가용성을 극대화하고, 시스템의 고장시 데이터 손실의 최소화와 이의 복구를 위해 고장검출 및 진단 기법에 대한 방안을 제시한다.

1. 서 론

최근 웹을 통한 다양한 서비스로 인터넷 사용자 수가 급격히 증가하고 있으며, 통신기기, 가전/멀티미디어기기 등의 여러 분야에서도 인터넷을 이용한 응용프로그램의 개발이 확대되고 있는 추세이다. 또한, 초고속통신망 등의 인터넷관련 기본 인프라 구축되면서 그 사용환경이 크게 개선되고 있다. 따라서, 인터넷 사용의 증가로 인해 예상되는 트래픽(traffic)의 발생은 웹 서버단의 문제가 될 것이다. 즉, 웹 서버의 접속용량 및 작업처리 속도가 서비스의 질 향상에 큰 영향을 줄 수가 있으며, 특히 business-critical 한 경우에는 웹서버의 안정성 및 가용성 문제가 가장 중요한 요소로 대두되고 있다.

일반적으로 기존의 단일 웹서버에서는 고가·고성능의 시스템을 구축하여 웹서버에서 발생될 수 있는 이러한 문제들을 해결하고 있으나, 급격한 접속수에 따른 서버의 확장성 및 유연성 등이 부족하다. 이를 위해 최근에는 리눅스에 의해서 일반 PC를 클러스터링 하여 클러스터 가상 웹서버를 구축하고 있는 사례가 크게 증가하고 있다. 이러한 방법은 사용자측에서는 마치 하나의 서버에만 접속한 것처럼 투명성(transparency)을 제공하며 서버측에 대해서는 확장성을 보장할 수 있다[1]. 확장성을 목적으로 하는 이러한 클러스터 웹서버의 구조는 증가하는 부하를 모두 처리할 수 있어야 한다. 따라서 부하를 공유하는 독립된 서버들로 구성된 분산구조가 확장성 웹서버를 위해 보다 적절한 방안이 된다[2].

또한, 웹서버의 고가용성을 증가시켜 클라이언트에게 요청지연시간(request latency)을 줄이기 위해 제안된 많은 솔루션들이 있다[3]. 이와 함께, 클러스터 웹서버의 고가용성을 위해 각 요소의 고장 발생에 대해서 신속한 고장 진단 및 검출 그리고 적절한 복구에 대한 고려가 필수이다. 본 연구에서는 이러한 클러스터링 가상웹서버를 구현하고, 이를 위한 고가용성 및 고장포용기법 그리고, 여기에 적용된 고장 진단기법에 대한 방안을 제시한다.

본 연구는 2000년 정보통신연구진흥원 산업기술개발사업의 지원 하에 진행되었습니다.

2. 클러스터 웹서버

2.1 클러스터 웹서버의 구성

단일 서버와는 달리 고확장성 및 고가용성을 위해 독립적인 여러 서버들을 그룹화하여 마치 하나의 서버처럼 운영하는 것을 클러스터(cluster)라고 한다[4]. 본 연구에서는 리눅스를 기반으로 하여 그림 1과 같은 클러스터링 웹서버를 구현한다. 클러스터 시스템은 다양한 구조로 구현 될 수 있으나, 일반적으로 부하를 분산해주는 로드밸런서(load balancer)와 그에 딸린 여러 대의 실서버(real server)들로 구성될 수 있다[4,5]. 비록 많은 실서버들로 이루어져 있어도, 인터넷을 통해 웹서버에 접속하는 사용자에게는 단일 서버 또는 단일 서버이미지를 갖게 하며, 사용자는 클러스터를 하나의 서버인 것처럼 간주하게 된다. 이러한 클러스터링 기술은 저가의 시스템을 이용하여 고가용성 및 고성능의 시스템을 구현하기 위해 널리 사용되고 있다.

2.2 고가용성 기법

클러스터 웹서버의 로드밸런서는 웹서버의 가상IP를 가지고 있으며 사용자 접속 요청에 의한 부하를 각 실서버에 분배해주는 기능을 하고 있다. 로드밸런서에 고장이 발생했을 경우, 전체 시스템의 고가용성을 보장하기 힘들다. 그러므로 고가용성의 유지를 위해 그림 1에서와 같이 로드밸런서에 대해서 백업서버를 두고 있다. 로드

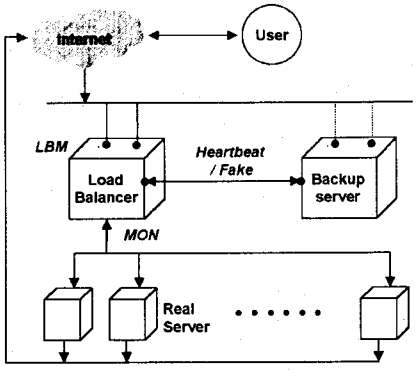

그림 1. 클러스터 웹 서버의 구성도

밸런서와 백업사이에는 heartbeat에 의해 주기적으로 'alive'신호를 주고받으며 항상 서로의 상태를 감시하며, 로드밸런서의 고장발생시 backup 서버가 fake에 의해 가상IP를 이어받음으로써 지속적으로 로드밸런서의 역할을 하게 된다.

mon은 각 실서버의 상태를 모니터링하는 기능을 하며, 이것에 의해서 로드밸런서는 각 실서버로부터 각각의 상태(작업량, 트래픽량 등)를 주기적으로 전송받아 다음 부하분배시 이를 이용하여 적절한 부하분배 스케쥴링을 하게 된다. 이때, 한 실서버에 고장이 발생할 경우, 고장이 발생한 서버는 클러스터에서 제외되며 부하분배 스케쥴이 재작성함으로써 사용자 접속요청에 대해서 지속적인 서비스를 해주게 된다. 이와 같이, 전반적으로 클러스터내의 각 요소들 상호간의 작용들은 GUI로 제공되는 LBM(Load Balancer Manager)에 의해 관리된다.

그러나, 이러한 구조가 예상하지 못한 고장과 열악한 서버 환경에서는 완벽하게 고가용성을 유지하지 못할 수도 있다. 그러므로, 고가용성 및 고신뢰성을 위해, 시스템에서 예측될 수 있는 결함 및 고장의 효과를 고려하여 일시적 또는 영구적인 결함 또는 고장이 발생하더라도 적절한 고장지역에 대한 검출 및 분리, 그리고 신속한 복구를 통해서 고장에 의한 영향을 최소화하면서 시스템에 진행중인 작업을 정상적으로 수행할 수 있도록 하기 위해 고장포용기법에 대한 고려가 필요하다.

3. 클러스터 웹서버를 위한 고장포용시스템

고장포용이란 시스템의 일부 요소에 고장이 일어나더라도 전체 시스템이 잘못된 행동을 일으키기 전에 고장을 회복시키거나 고장의 효력을 제거하여 안전한 작업수행을 보장할 수 있게 하는 기법을 말한다. 본 연구에서는 고장포용을 위해, 서버에서 고려되는 고장에 대해서 고가용성을 보장하기 위한 다음과 같은 공간 여분 및 시간 여분을 바탕으로 적절한 고장포용의 설계 기법을 적용하며, 고장포용 시스템의 성능을 평가하기 위한 다양한 기준 중 확률적 접근방식에 바탕을 둔 신뢰도(reliability)와 가용성(availability), 그리고 유지성(maintainability)에 대해 고려한다.

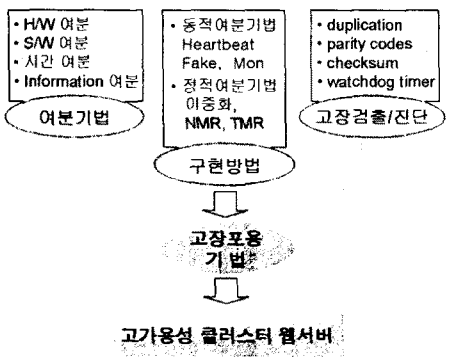

그림 2. 고가용성을 위한 고장포용기법

3.1 고장의 분류

고장은 발생원인, 일시적 행동 및 결과지속시간 등의 여러 관점에서 다양하게 분류될 수 있다. 본 연구에서는 웹서버의 로드밸런서가 처리하는 대부분의 작업인 트랜잭션(transaction)작업에서의 고장에 대해서 고려를 한다. 이러한 고장은 다음과 같은 형태로 분류될 수 있다[6].

- crash failures : 영구적인 서버의 고장
- crash + link failures : 서버가 crash되거나 link(network)에서 데이터손실이 발생
- receive-omission failures : 서버가 crash되거나 수신 단의 버퍼 overflow에 인해 수신된 데이터에 손실발생
- send-omission failures : 서버가 crash 되거나 송신단의 과부하에 의해 송신 데이터 손실발생
- general-omission failures : 서버가 receive-omission 또는 send-omission 또는 이들 모두에 의해 고장이 발생

3.2 고장포용기법

3.2.1 동적여분(dynamic redundancy)기법

기존의 동적여분기법은 로드밸런서의 고가용성을 위한 것으로, 로드밸런서에 고장이 발생했을 때 백업으로 있던 다른 하나가 heartbeat에 의해 고장을 인식하고 그 일을 이어받아 지속적인 동작을 하게 하는데, 그 범위가 두 대에 대해서만 적용되었다. 본 연구에서는 이를 확장하여, 다수의 로드밸런서 spare pool이나 multi-mode 기법을 적용한다. 기존의 단일 서버 및 백업된 서버와 비교하여 가용성과 신뢰성 측면에서 현저한 모듈의 고장발생률을 감소라는 효과를 얻어낼 수 있다.

3.2.2 정적여분(static redundancy)기법

정적여분기법은 다수의 서버에 대한 것이 아니라, 각각의 실제 서버에 즉 단일 모듈 내에서, 이중화, 삼중화, NMR 기법을 이용하여 구현되는데, 동일한 작업을 할 N개(≥3)의 프로세서나 모듈을 구성하여 고장이 발생하더라도 고장차폐를 통해서 고장과 관계없이 작업수행을 완결할 수 있도록 하거나 또는 여분을 추가설치해서 고장이 발생할 경우에 적절한 스위칭을 통해서 정상적인 작업을 수행한다. 이를 통해 단일모듈의 고장발생률을 현저히 줄일 수 있게 된다.

3.2.3 checkpoint

일정한 주기로 현재의 상태를 체크하여 그 상태를 기억장소에 저장하고 고장이 발생되었을 경우 발생시간 바로 전의 checkpoint가 저장한 상태부터 모듈의 작업을 다시 수행하는 일종의 시간여분 고장포용 기법이다. 즉, 어떤 작업의 중간에서부터 실행하기 위해 저장되는 정보의 양은 다양하게 변한다. 따라서 어떤 시간에 프로그램의 정보들을 저장하기 위해 캐시와 같은 빠른 입출력 시간을 갖는 메모리를 사용하고 이의 운용을 관찰하는 운영체제로서 리눅스를 통해 용이한 checkpointing 기법이 구현 가능하다.

본 연구에서는 로드밸런서의 고가용성을 보장하기 위해 제안된 공간여분(동적여분 및 정적여분) 뿐만 아니라 checkpoint기법을 이용한다. 즉, 로드밸런서의 부하분배 스케쥴을 위한 각 실서버의 정보는 주기적으로 백업서버에 기록하게 함으로써 로드밸런서의 고장시 백업서버의 재작업 시간을 최소화해 주도록 한다.

3.3 고장검출 및 회복기법

고장검출 및 회복기법(fault detection & correction or recovery)은 H/W적으로 구성되기도 하지만 가장 일반적인 형태는 coding theory를 기반으로 개발된 기법들이다. 온-라인 기법과는 다르게 오프-라인 고장검출은 고장진단을 하고 있을 때 작업을 수행할 수 없으므로 이에 대한 적절한 시스템 설계가 중요하다. 일반적으로 다음과 같은 기법들이 적용된다.

- Duplication : 가장 간단한 구조를 가지며 저렴한 비용뿐만 아니라 시스템 성능에 영향을 거의 주지 않기 때문에 가장 많이 사용되는 고장검출 기법으로서 동일한

작업을 각각 다른 프로세서 및 모듈에서 수행해서 그 결과를 서로 비교함으로써 고장발생 유무를 알아낼 수 있는 방법이다.

• Parity codes : 전체가 n-bit로 구성된 information code 가운데 하나의 bit 이상에서 동시발생이 가능한 고장을 k-bit라고 가정한다면, $({}_{n+c}C_{(k)})+1 \leq 2^c$의 식에 의해서 필요한 parity bit c를 설정한다. 즉, 만약 하나의 bit 만이 고장 가능하다면, $n+c+1 \leq 2^c$에 의해서 parity bit 설정이 가능하다.

• Checksum : 서로 데이터를 주고받을 때 전송단 쪽에서 모든 word를 더함으로써 checksum을 설정한 후 데이터와 함께 적절한 데이터 코딩을 통해서 전송하고, 또한 수신단 쪽에서는 전송된 데이터를 받아서 데이터와 checksum을 분리해서 전송된 데이터의 checksum을 전송단에서 받은 checksum값과 서로 비교함으로써 고장을 검출할 수 있다.

• Watchdog timers : 일정한 시간이내에 원하는 값이 어떤 특정 프로세서 및 모듈에 주어지지 않는 경우에 timeouts에 의한 고장으로 판단되는데, 이때 고장을 검출하기 위해서 어떤 프로세서의 데이터 및 어드레스 라인을 주기적 또는, 비주기적으로 watchdog timer를 통해서 체크해야 한다.

3.4 로드밸런서 및 실서버의 고장검출

로드밸런서에서 고장의 발생 및 시간을 감지하기 위해서 특별한 기법을 적용할 필요가 있다. 보통 고장포용을 위한 고장 발생 감지 기법은 먼저 결함의 종류(일시적, 간헐적, 영구적)를 찾아내고 이러한 결함에 의한 오류를 감지하게 된다. 오류의 발생을 감지한 후에야 비로소 결함의 위치를 파악할 수 있지만 이러한 과정이 매우 복잡하여 오류는 보통 전파(propagation)를 하게 된다. 따라서 결함 및 오류의 감지 메커니즘이 완전하지 않다면 다른 모듈로의 전파가 예상된다. 일반적인 감지 메커니즘은 TMR과 같은 voting기법을 사용하거나 BIT (Built-in Test)를 이용하여 주기적으로 자신의 상태를 진단하기도 한다.

로드밸런서 작업의 고신뢰성을 보장해주기 위해서 정적여분으로써 TMR을 적용하여 고장진단 및 감지를 하게 된다. 고장감지시 고장회복기법에 의해 고장이 복구되지만 이러한 고장이 반복되면 시스템의 성능저하가 지속되므로 watchdog timer를 통하여 로드밸런서를 재가동시키고 백업서버로 그 작업을 넘겨주는 것이 전체시스템의 성능저하를 최소화 시켜주는 효과를 준다.

이러한 정적여분인 TMR에 의한 고장감지는 개별 모듈 즉, 로드밸런서나 실서버에는 적용될 수 있으나, 고장감지의 범위를 다중 모듈 즉 클러스터내의 모든 모듈(로드밸런서, 실서버, 백업 등)에까지 확장을 시키기에는 한계를 가지고 있다. 이를 극복하기 위해서, 로드밸런서는 클러스터내의 각 모듈의 상태에 대한 정보를 mon에 의해 주기적으로 갱신시킨다. 이는 local의 전용 네트웍과 serial cable에 의한 heartbeat신호를 주고받음으로써 각 실서버와 백업 로드밸런서의 동작여부를 항상 체크를 하게 한다. 일반적으로 heartbeat신호는 상대방의 동작 또는 down상태를 확인하기 위해 최소한의 데이터를 주고받는데, 본 연구에서는 serial cable에 의한 heartbeat신호와 함께 전용 네트웍을 통해 로드밸런서가 각 실서버로부터 주기적으로 정보를 받고 동시에 백업서버에 전송함으로써, 백업의 가동시 checkpoint작업에 의해 재시작 시간을 최소화하게 한다.

4. 결 론

본 연구에서는 동시 접속 트래픽의 수용, 유연한 시스템 확장성과 business-critical한 작업의 경우 웹서버의 안정성 및 고가용성 문제를 보장해 줄 수 있는 클러스터 웹 서버를 구현하였다. 이를 위해서 heartbeat, fake, mon 등의 HA기법을 이용하여 로드밸런서의 백업을 두고, 로드밸런서에 고장 발생시 전체 시스템의 안정성 및 고가용성을 극대화하기 위해 데이터 손실을 최소화하고 이를 복구하기 위한 고장 검출 및 진단 기법에 대한 방안을 제시하였다.

[참 고 문 헌]

[1] 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.

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

[3] B. Narendran, S. Rangarajan and S. Yajnik, "Data Distribution Algorithms for Load Balanced Fault-Tolerant Web Access", *Reliable Distributed Systems, 1997*, pp.97-106, 1997.

[4] http://www.linux-vs.org

[5] M. Dias, W. Kish, R. Mukherjee and R. Tewari, " A scalable and highly available web server", *IEEE Proceedings of COMPCON '96*, pp.85-92, 1996.

[6] N. Budhiraja, K. Marzullo, "High-available services using the primary-backup approach", management of replicated data, pp.47-50, 1992.

[7] K. Shin and X. Cui, "Effects of computing time delay on real-time control systems", *Proc. of 1988 ACC*, pp.1071-1076, 1988.