

리눅스 상에서 고가용성 웹 서버 클러스터의 설계 및 구현

김동호 박 권 김명호
송실대학교 컴퓨터학과

(dhkim,terius}@amin.soongsil.ac.kr, kmh@comp.soongsil.ac.kr

Design and Implementation of HA Web Server Cluster on Linux

Dong-Ho Kim Kwon Park Myung-Ho Kim
Dept. of Computer Science, SoongSil University

요 약

인터넷 사용의 일반화로 인터넷을 통한 전자 상거래가 점차 늘고 있다. 이 가운데 많은 사용자들이 이용하는 사이트는 하루 수 백만건의 접속이 이루어지고 있고 사용자의 요구도 다양해지고 있다. 이러한 사이트에서는 사용자에게 최적의 성능을 보장해야 될 뿐만 아니라 시스템의 다운 타임을 최소화 할 수 있는 방안이 필요하게 된다. 기존의 단일 시스템 체계는 이런 면에서 한계에 도달하였다. 단일 시스템의 한계점을 극복하기 위해서 클러스터를 이용한 대안이 제시되고 있다. 따라서 본 논문에서는 리눅스 플랫폼 상에서 클러스터를 이용한 웹 서버를 설계 및 구현하였다.

1 서론

인터넷 사용이 일반화되면서 인터넷 사용자의 수가 매년 증가하고 있으며 인터넷은 새로운 산업의 형태로 발전하고 있다. 전자 상거래는 그 좋은 예가 될 수 있다. 현재 전자 상거래와 같은 서비스를 제공하는 사이트가 점차 늘어나고 있는 추세인데 이 가운데 많은 사용자들이 이용하는 사이트는 하루 평균 백만 건 이상의 접속이 이루어진다. 이러한 사이트는 사용자의 요청을 처리하기 위해 많은 컴퓨팅 파워를 요구하고 있으며 시스템의 다운 타임 최소화도 요구되고 있다.

단일 시스템으로 구성된 기존의 사이트들은 단일 시스템의 한계 때문에 고성능과 고가용성의 서비스를 제공하는 것이 이미 한계에 도달하였다[2]. 이러한 단일 시스템의 한계를 극복하고 더욱 강력한 컴퓨팅 파워와 시스템의 안정적 서비스 제공을 위해 클러스터링 기술이 웹 사이트에 적용되고 있다[1]. 기존의 방식은 RR_DNS를 사용하거나 지능형 스위치를 사용하여 클러스터를 구성하는데 이러한 방법은 추가적 비용 부담과 IP 캐싱 문제가 있다.

따라서 본 논문은 추가적인 비용 문제를 해결하기 위해 소프트웨어 디스패처를 사용하고 IP 캐싱과 클러스터의 안정성을 확보하기 위해 분산형 구조를 갖는 웹 서버 클러스터를 설계하도록 한다.

2. 클러스터링과 웹 서버 클러스터

클러스터는 확장성, 고성능, 안정성 등의 특징을 제공하는데 이러한 클러스터 시스템의 특징은 고성능, 고가용성 웹 서버를 구축하는데 적합하다. 사용자 요구의 증가로 인한 시스템 성능향상의 문제는 클러스터링의 확장성으로 해결할 수 있다. 클러스터링의 가용성을 이용해서 웹 서버의 다운타임을 최소화 할 수 있고 클러스터링의 고성능은 서버 쪽의 컴퓨팅 파워를 요구하는 사용자의 요청을 처리하는데 적합하다.

2.1 기존 연구

기존 연구들의 클러스터 구성 방식과 장단점을 표1에서 비교하였다.

표 1 웹 서버 클러스터 비교

	LocalDirector	NetDispatcher	SWEB
구성	스위치 방식	집중형 디스패처 방식	분산형 디스패처 방식
장점	관리용이 성능 우수 안정성 우수	비용부담 적음	비용부담 적음
단점	추가적 비용부담	SPOF 문제 발생	노드 결합 발생

2.2 제안하는 웹 서버 클러스터

본 논문의 목적은 추가적인 비용부담이 없으며 고성능과 고가용성을 보장하는 웹 서버 클러스터를 설계하고 구현하는 것이다. 따라서 소프트웨어 디스패처를 분산형 방식에 적용해 설계하였다.

3. 웹 서버 클러스터 설계 및 구현

본 논문에서 제시하는 웹 서버 클러스터는 디스패처와 로드 모니터 그리고 하트비트 데몬으로 구성된다.

3.1 디스패처의 설계 및 구현

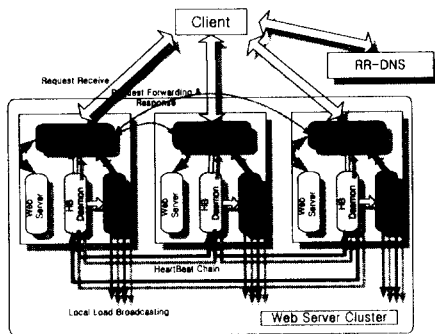


그림 1 웹 서버 클러스터 구성과 컴포넌트

디스패처는 사용자의 요청을 적당한 노드를 선택해 전달하고 서버의 응답을 다시 사용자에게 전달하는 기능을 한다. 그림 1에서처럼 사용자의 요청은 DNS에 의해 일차적으로 각 노드의 디스패처에 전달된다. 디스패처는 로드 모니터에 의해 전달받은 각 노드의 부하 정보를 바탕으로 스케줄링 한다. 각 노드의 부하 정보는 다

음 식에 의해 로드 모니터가 구하게 된다.

$$S_i(L) = A U_{Cpu} + B U_{Disk} \dots \dots \dots \text{식(1)}$$

U_{Cpu} : 노드의 Cpu 활용율

U_{Disk} : 노드의 $Disk$ 활용율

$P_{S_i}(R)$: 사용자 요청 R 이 서버 S_i 로 포워딩될 확률

A : CPU 가중치, B : $DISK I/O$ 가중치, $A + B = 1$

이때 각 서버의 부하를 나타내기 위해서 CPU의 활용율과 DISK I/O 활용율을 사용하였다. 디스패처는 식(1)에 의해 구해진 각 노드의 부하정보를 식(2)를 이용해서 각 노드가 사용자의 요청을 할당받을 확률을 계산하게 된다.

$$P_{S_i}(R) = \frac{S_i(L)}{\sum_{j=0}^n S_j(L)} \dots \dots \dots \text{식(2)}$$

$P_{S_i}(R)$: 사용자 요청 R 이 서버 S_i 로 포워딩 될 확률

3.2 로드 모니터의 설계 및 구현

로드 모니터는 주기적으로 자기 자신의 부하 정보를 측정하고 클러스터내의 다른 노드의 부하 정보 또한 관리한다. 구해진 부하정보와 클러스터 노드의 구성 정보를 디스패처에게 전달한다. 클러스터 노드의 구성 정보는 현재 결합이 발생해서 클러스터에서 제외된 노드정보나 결합이 복구되어 다시 클러스터에 진입한 노드 정보이다.

3.3 하트비트 데몬의 설계 및 구현

하트비트 데몬은 주기적으로 자기 자신의 디스패처와 로드 모니터를 검사하고 자신은 리눅스 시스템 명령인 Cron이 주기적으로 검사한다.

또한 클러스터의 안정성과 고 가용성을 위해 하트비트 체인을 구성하고 유지한다. 하트비트 데몬과 웹 서버 클러스터의 안정성 유지 구조와 하트비트 체인 구성은 그림 2와 같다.

4. 실험 및 분석

본 논문에서 구현한 웹 서버 클러스터의 성능 측정은 SURGE를 사용하였다[3]. 웹 서버 클러스터는 PI 400MHZ CPU와 128MB 메모리 LINUX 2.2.5 커널을 사용하는 PC 4대로 구성하였고 클라이언트는 표 2와

같이 구성하였다.

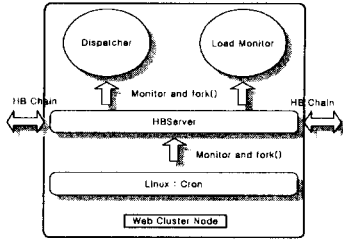


그림 2 웹 서버 클러스터의 고가용성

표 2 실험에서 사용한 클라이언트의 사양

	lily	young	outsider	milkyway	cosmos
CPU	P II 233	P 166	MMX 233	P II 266	P II 266
Memory	64 M	32 M	64 M	96 M	96 M
Kernel	2.2.9	2.2.5	2.2.10	2.2.5	2.2.9

본 논문에서 제시하는 웹 서버의 성능을 비교하기 위해 웹 서버를 단일 웹 서버, 집중형 디스패처 방식의 웹 서버, 라운드 로빈 DNS를 사용한 웹 서버, 분산형 디스패처 방식의 웹 서버 등 네 가지를 구성하여 비교하였다. 성능 비교 실험은 각 웹 서버에 부하가 전혀 걸리지 않은 경우, 웹 서버 클러스터 시스템에 결합 노드가 발생한 경우, 특정 노드에 부하가 어느 정도 있는 경우 등의 세 가지로 나누어서 수행하였다. 각 실험의 결과는 그림3, 그림4, 표3과 같다.

실험 결과를 보면 디스패처를 사용한 RR_DNS의 경우 평균 응답 시간이 작은 것을 볼 수 있고 두 표3을 보면 시스템의 가용성이 확보된 것을 볼 수 있다.

5. 결론 및 향후 과제

본 논문에서는 고성능과 고가용성의 웹 서버 클러스터를 구성하기 위해 소프트웨어 디스패처를 분산형으로 적용한 웹 서버 클러스터를 제안하였다. 실험을 통해 본 논문에서 구현한 웹 서버 클러스터가 가용성도 뛰어나고 성능 향상의 결과를 나타냄을 보였다.

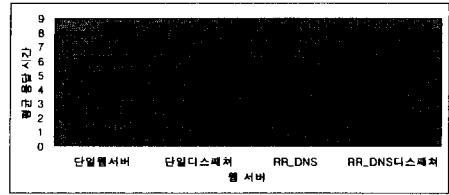


그림 3 평균 응답 시간

표 3 웹 서버의 가용성

	lily	young	outsider	milkyway	cosmos
RR_DNS	∞	5.74 / 529.21	∞	∞	∞
단일 디스패처	SPOF	SPOF	SPOF	SPOF	SPOF
RR_DNS 디스패처	1.45 / 13.06	3.14 / 170.28	1.47 / 14.54	1.29 / 18.78	1.00 / 12.13

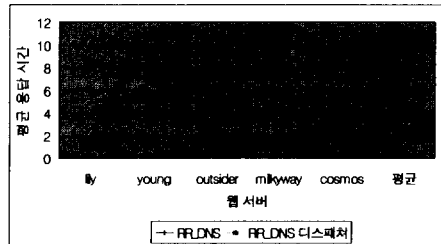


그림 4 특정노드에 부하가 클 때 평균 응답 시간

6. 참고 문헌

- [1] Raykumar Buyya, "A Scalable and Highly Available Clustered Web Server", Prentice Hall, Chap 36, 1999. *High Performance Cluster Computing voll.*
- [2] Huican Zhu, Ben Smith, Tao Yang, "A Scheduling Framework for Web Server Clusters with Intensive Dynamic Content Processing", Department of Computer Science University of California Santa Barbara, CA 93106.
- [3] Paul Barford and Mark Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", Computer Science Department, Boston University.