웹 클러스터 시스템의 실시간 서버 상태를 기반으로 한 부하분산 방안

윤 천 균†

요 약

웹 클러스터 시스템에서 사용자들의 접속 요구를 효율적으로 real 서버에 분산시키기 위하여 기존의 부하분산 알고리즘들과 현재 많이 사용중인 WLC(Weighted Least Connection) 알고리즘에 대하여 연구하였다. 이 부하분산 알고리즘들은 서버들의 부정확한 부하상태 파악과 파악 시점에 문제가 있어 서버들 간에 부하 불균형이 발생함으로써 효율성이 저하된다. 본 연구에서는 사용자의 접속 요청 시 broadcasting RPC(Remote Procedure Call)를 이용하여 서버들의 다양한 부하상태를 실시간으로 파악하여 부하를 분산하는 알고리즘을 제안하고, prototype을 구현하여 성능을 실험하였다. 실험 결과, real 서버 간 부하 불균형 현상이 기존 방법에 비하여 크게 향상되었고, 응답시간이 단축되는 효과로 웹 클러스터 시스템의 성능이 향상되었다.

키워드 : 부하분산, 클러스터링

A Load Distribution Technique of Web Clustering System based on the Real Time Status of Real Server

Youn Chunkyun†

ABSTRACT

I studied about existent load distribution algorithms and the WLC(Weighted Least Connection) algorithm that is using much at present to distribute the connection request of users to real servers efficiently in web cluster system. The efficiency of web cluster system is fallen by load imbalance between servers, because there is problem in inaccurate load status measuring of servers and measuring timing at these load distribution algorithms. In this paper, I suggest an algorithm that distributes load base on various load state of servers by real time using broadcasting RPC(Remote Procedure Call) when a user requests connection, and implement a prototype and experiment its performance. The experiment result shows that load imbalance phenomenon between real servers was improved greatly than existing method, and performance of web cluster system was improved by efficiency that response time is shortened.

Key Words : Load Distribution, Clustering

1. 서 론

최근 인터넷 사용자의 급속한 증가, 고속 네트워크 기반의 다양한 웹 서비스들의 등장, 인터넷을 통해 송수신되는 데이터의 멀티미디어화 등에 따라 네트워크 트래픽이 급증하고 있다. 이러한 트래픽의 증가로 인하여 다양한 서비스를 제공하는 서버와 이를 클라이언트와 연결하는 네트워크에서 병목현상이 집중되고 있다. 이에 대한 해결책으로 얼마 전까지는 백본의 대역폭 확장에 주력하였으나, 서버의 수용 한계를 넘는 트래픽에 대한 해결책으로 서버의 성능 향상과 가용성을 확보하는 방안이 중요시 되고 있다[1]. 이를 위해 고성능 프로세서를 장착하거나 여러 개의 CPU를 이용한 병렬처리 기법을 통하여 시스템의 성능을 향상하여 왔다. 최근 저가형 서버들을 고속 네트워크로 연결하고, 이 서버들에 부하분산 기법을 적용하여 고성능 컴퓨팅을 제공하는 클러스터 시스템 구축이 증가하고 있다[2, 3].

클러스터란 여러 대의 시스템을 고속 네트워크를 통해 연결하여 하나의 단일 시스템처럼 동작하는 시스템을 말한다. 클러스터로 연결된 컴퓨터들은 클라이언트 입장에서는 단일 호스트와 같은 단일 시스템 이미지로 보이고, 클러스터 group에 저렴한 장비들을 계속 추가할 수 있어 슈퍼컴퓨터나 병렬컴퓨터보다 확장성 측면에서 우수하고 고속 컴퓨팅 및 멀티미디어 데이터를 효과적으로 처리할 수 있다. 이러한 장점을 이용하여 웹 서버를 클러스터 형태로 구성하였을

† 정 회 원 : 호남대학교 인터넷소프트웨어학과 조교수
논문접수 : 2004년 11월 25일, 심사완료 : 2005년 5월 16일

때 기존 웹 서버에서 취약한 확장성, 가용성, 경제적인 효율성을 만족시킨다. 클러스터 시스템을 사용 목적에 따라 과학 계산용과 부하분산용으로 구분할 수 있는데, 부하분산이 목적인 클러스터는 단일 서버를 사용할 때 발생할 수 있는 부하를 여러 서버로 분산시켜 웹 서버의 과부하를 해결한다[4].

본 연구에서는 클러스터 시스템으로 구성된 웹 서버의 상태를 broadcasting RPC를[5] 이용하여 각 real 서버들의 부하를 실시간으로 측정하여 이를 부하분산에 적용함으로써 서버의 부하 편중 현상이 개선되는 효율적인 부하분산 방안을 제안하고, prototype을 구현하여 성능을 분석하고자 한다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구 및 문제점에 대하여 알아보고, 제3장에서는 본 논문에서 제안하는 웹 클러스터 시스템의 부하분산 방법에 대해 설명한다. 제4장에서는 제안한 방식에 대한 prototype을 설계 구현하고 기존 방법과 제안된 방법의 성능을 실험하고 그 결과를 분석한다. 마지막으로 제5장에서는 결론을 소개한다.

2. 관련 연구 및 문제점

2.1 서버 시스템의 성능 (부하) 측정

클러스터 시스템에서 효율적인 부하분산을 하기 위해서는 각 Real 서버들의 정확한 부하 상태 또는 성능을 파악하여 가장 부하가 적은 시스템에 클라이언트들의 요구를 분산시켜야 한다. 이에 대한 연구로 웹 서버에 많은 사용자들이 동시에 접근할 경우나 멀티프로세서 환경에서 운영되는 경우의 서버 부하를 분산하기 위한 방법들이 연구되었다[6, 7]. 또한, 웹 서버의 성능에 미치는 요소를 정의하거나, 다양한 벤치마크를 통해 웹 서버의 성능을 측정하는 방법들에 관한 연구들도 이루어져 왔다[8]. 클라이언트에 Agent를 심어 서버의 응답시간을 측정하거나 ping, traceroute 등을 이용하여 노드 사이의 네트워크의 성능을 분석하게 하는 등 여러 가지 방식을 제시하고 있다[9].

2.2 기존 클러스터 시스템의 부하분산 기법 및 문제점

웹 클러스터 시스템의 부하분산을 담당하는 핵심적인 기능으로 다음과 같은 부하분산 알고리즘이 있다.

Round-Robin(RR) 알고리즘은 real 서버의 접속 수나 응답시간을 고려하지 않고 모두 동일하다는 가정 하에 순서적으로 작업 서버를 선택하는 방식이다. 스케줄링에서 소요되는 처리 시간을 효과적으로 단축시킬 수 있다[6, 10, 11].

Weighted Round-Robin(WRR) 알고리즘은 각 real 서버에 처리용량에 비례한 가중치를 부여하여 부하를 분배하는 방식이다. 따라서 각 서버들은 가중치에 따라 작업할당 순서가 정해지며 네트워크의 접속은 정해진 순서에 따라 Round-Robin 방식으로 real 서버들로 보내진다. 하지만 부하가 폭주하는 경우, 부하의 크기를 고려하지 않고 각 요청을 동일하게 간주하므로 단 시간에 폭주하는 시스템의 경우 부적합하다고 할 수 있다[6, 10, 11].

Least-Connection(LC) 알고리즘은 새로운 서비스 요청이 발생한 경우, 웹 클러스터 시스템을 구성하는 각각의 서버 중 현재 가장 작은 접속량을 가진 서버를 선택하여 작업을 할당하는 방식이다[6, 10, 11].

Weight Least-Connection(WLC) 알고리즘은 웹 클러스터 시스템을 구성하는 서버의 성능에 따라 가중치를 부여하고 동적으로 작업 접속 상황을 조사하는 방식으로 서버를 선택한다. 부하분산 과정은 Wi를 n개의 서버에 대한 가중치, Ci(i=1,2,...,n)를 n개의 서버가 현재 가지고 있는 접속 수라 하고 S를 웹 클러스터 시스템의 전체 접속수라 할 때, 새로운 서비스 요청이 들어올 경우, n개의 서버 중 접속 수와 가중치의 비율이 최소인 임의의 서버 j에게 접속을 할당하는 방식이다[6, 11].

위와 같이 기존의 클러스터 시스템 부하분산 알고리즘은 방식에 따라 다소 차이는 있지만 서버의 물리적 성능을 고려하여 가중치를 고정하고 접속 수의 변화에 의거하여 real 서버를 선택하는 방식을 취하고 있다. 이는 real 서버들의 다양한 부하요소들을 반영하지 못하기 때문에 정확하게 서버의 부하상태를 파악하지 못하고 있으며, 상태정보 파악 시점도 디렉터가 주기적으로 real 서버들의 접속수를 파악하고 있는 형태이기 때문에 실시간 부하를 반영하지 못하고 있다고 볼 수 있다.

3. 효율적인 부하분산 방안 제안

3.1 시스템의 다양한 부하측정 요소 검토 및 적용방안

본 논문에서는 기존 방식들에서 채택하고 있는 시스템의 부하요소들 중 서버의 물리적 성능을 고려한 가중치와 접속수에 의한 서버의 부하보다 더 정확한 부하상태 측정을 위하여 일반적으로 웹 서버로 많이 사용되고 있는 유닉스(리눅스) 기반의 서버들에 대한 성능 관련 요소들 중 클러스터 부하분산에 적절한 요소를 도출하여 측정 방법들을 파악하고 이를 제안하는 부하분산 방안에 적용하고자 한다.

시스템의 성능 분석은 컴퓨터가 운영되고 있는 상태에서 시스템이 언제, 어디에, 얼마만큼의 부하가 걸리는가를 확인하기 위한 다양한 이벤트를 측정하는 과정으로, 시스템의 성능에 영향을 주는 요소들은 다양하게 존재한다. 이 요소들은 여러 가지로 파악될 수 있지만 그 중에서도 시스템의 프로그램들이 사용하는 리소스의 측면에서 파악하는 것이 일반적이다. 시스템의 성능에 영향을 주는 요소들을 리소스 측면에서 파악하면 CPU, 메모리, 하드 디스크 I/O 량, 네트워크 등이 있다. 이들 중 하드 디스크의 I/O 량에 따른 부하는 디스크의 물리적 특성에 따라 결정되는 부하이므로 실시간으로 측정한 결과를 웹 클러스터 시스템의 부하분산에 적용하는데 있어 상대적으로 적정하지 않고, CPU, 메모리 및 네트워크의 상태 정보를 측정하여 부하부산에 적용하는 것이 효율적이다[8, 12, 13].

시스템의 부하요소들을 웹 클러스터 시스템의 부하분산에 적용하는데 있어 비교적 신속하고 정확한 서버의 상태정보

를 측정할 수 있는 3가지 항목에 대한 구체적인 부하 요소는 다음과 같다.

3.1.1 CPU의 부하

일반적으로 CPU의 부하를 측정하기 위해서는 CPU 전체 사용 정도, 평균 부하, 프로세스 별 CPU 사용 정도 등 다양한 상태 정보를 수집해야 하나 클러스터링 시스템의 부하분산을 위해 실시간 측정 및 적용을 고려할 때 현재와 향후의 CPU부하 상태를 비교적 정확하게 파악할 수 있는 항목으로 현재 수행을 기다리는 프로세스 수와 CPU의 idle율을 들 수 있다. 클러스터 시스템 구성 및 사용자 접속 빈도 등에 따라 차이가 있지만, 일반적으로 대기 프로세스 항목에 수치가 표현 되는 것은 찾아보기가 힘들다. 그 이유는 현재의 서버 시스템들은 거의 고성능 멀티프로세서로 운용되기 때문에 실시간으로 처리가 되기 때문이다. 따라서 일반적으로는 idle 항목만 확인하면 CPU의 부하가 적은 서버를 알 수 있다[12, 13].

3.1.2 메모리의 부하

메모리 항목에서는 최근 20초 안에 수행된 프로세스가 이용중인 가상 메모리 량과 여유(free) 메모리 량, 페이징 활동에 대한 다양한 상태 정보를 파악할 수 있다. 이들 정보 중 여유 메모리 량을 이용하면 현재의 메모리 부하 상태를 비교적 정확하게 확인할 수 있다[12, 13].

3.1.3 네트워크의 부하

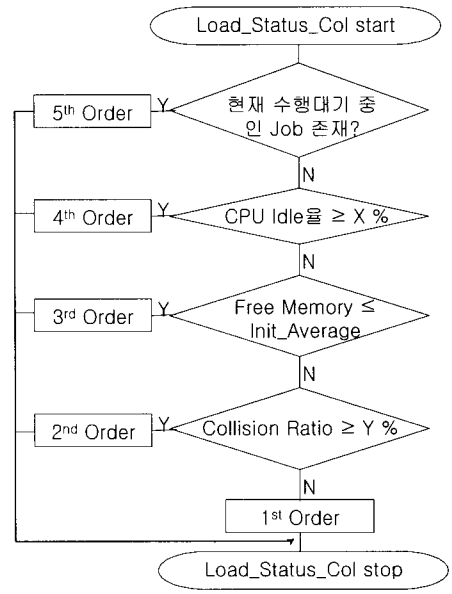
네트워크의 부하상태를 알 수 있는 항목으로 각 서버 네트워크 인터페이스의 패킷 I/O량, 패킷 에러율, 충돌율 등이 있다. 이들 중 충돌율이 5~10%에 가까워지면 네트워크에 과부하가 초래된 것이라고 볼 수 있으며, 필요 시 패킷 I/O량을 이용하면 네트워크의 부하상태를 확인할 수 있다[12, 13].

3.2 부하 요소를 고려한 서버 부하 측정 방안

서버 시스템의 주요 자원 중 윈도우용 응용 프로그램의 처리에 가장 큰 영향을 주는 것은 <표 1>에서와 같이 단연 CPU이고, 다음으로 메모리 용량이다. RAM의 속도도 관련이 있지만, 특히 메모리 용량이 큰 영향을 끼친다. <표 1>은 한 시스템 내에서의 물리적 자원만을 고려하였고 인터넷 환경은 고려하지 않았다. 그러나 우리가 관심 있는 클러스터링된 웹 서버 시스템에서는 네트워크에 대한 부하요소도 매우 중요하다. 따라서 본 연구에서는 이미 언급한대로 네트워크 부하를 포함하여 3.1 절에서 파악한 각 항목의 구체적인 내용과 <표 1>을 바탕으로 (그림 1)과 같은 처리 방법을 통하여 부하가 작은 서버 시스템을 선정할 수 있도록 하였다.

〈표 1〉 윈도우용 응용 프로그램의 처리속도에 영향을 주는 하드웨어 비율(자료: 인텔)

CPU=54%	메모리=25%	그래픽카드=12%	하드disk=9%



(그림 1) 다양한 부하요소를 고려한 부하 산정 방안

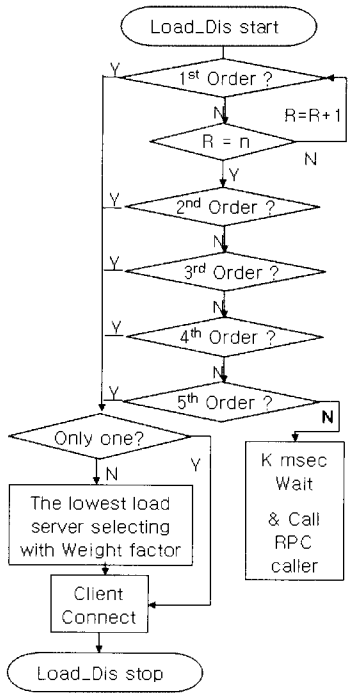
이 알고리즘은 각 real 서버들에 Agent 형태로 탑재되고, Director에서 broadcasting RPC를 이용하여 호출된다. 호출된 각 real 서버들은 (그림 1)에 준하여 자신의 부하상태를 파악하여 Director에게 전송한다. 여기서 "CPU idle율"과 "Collision ratio"의 비교 기준값(X, Y)은 구성된 클러스터 시스템과 사용자들의 환경에 따라 적절하게 조정되어야 하며, "Free Memory"에서의 Init_Average는 클러스터 시스템 구축 후 안정화된 시점에서 적정 값을 설정하여 사용하여야 한다.

3.3 실시간 서버 상태를 고려한 부하분산 방안

본 연구에서는 기존 부하분산 방식 중 WLC 방식을 개선하여 사용자의 접속요구 수신 시 실시간으로 각 real 서버의 정확한 부하상태를 파악하여 이를 부하분산 알고리즘에 적용함으로써 최저 부하 상태인 서버에 사용자의 접속요구가 처리될 수 있도록 하는 부하분산 방안을 제안하고자 한다.

제안하는 클러스터 부하분산 시스템은 (그림 2)와 같이 Direct routing 기반의 클러스터 환경으로 구성하였다.

(그림 2) 제안 클러스터 시스템 구성 방식



(그림 3) 제안 부하분산 알고리즘

(그림 3)은 3.2절에서 파악된 서버의 부하상태를 기반으로 사용자의 접속요구를 어떻게 최저 부하를 갖은 서버에 할당할 것인지에 대한 처리 알고리즘을 나타낸 것으로 상세 내용은 다음과 같다.

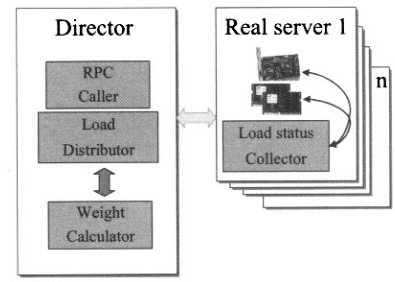
① Director는 가상 서버를 생성하고 가용한 real 서버들을 등록한다. 이 때 각 real 서버의 CPU 성능, 메모리 량, 지원 네트워크 대역폭 정보를 수집하여 물리적 사양에 따른 가중치 요소(Weight Factor)를 기존의 WLC를 준용하여 계산하여 저장하고, (그림 3)에서 보듯이 동일 order 발생 시 사용한다.

② Director는 사용자의 웹 접속 요청을 받으면 등록된 real 서버들에게 broadcasting RPC (RPC Caller) 루틴을 이용하여 각 real 서버들의 부하상태 수집기 (Load status collector) 모듈을 호출하여 그 결과를 수신한다.

③ 부하분산(Load distributor) 모듈은 (그림 3)과 같이 real 서버로부터 결과 수신 시 마다 1st order가 있는지 확인하여 있을 시 모든 real 서버들의 부하상태 정보를 수신할 때까지 기다리지 않고 바로 해당 서버에 사용자 요청을 할당한다. 없을 시는 모든 real 서버(n개)로부터 정보를 수신한 다음 2nd order가 있는지 확인한다. 여기서 R의 초기값은 0이고, n은 real서버의 수를 말한다.

④ 2nd order가 1개만 있으면 해당 서버에 사용자 요청을 할당하고, 2개 이상 있을 경우에는 ①에서 계산해둔 가중치 요소를 적용하여 가장 성능이 우수한 서버에 해당 사용자 요청을 할당한다.

⑤ 3rd~5th order에 대해서도 ④항과 같은 처리를 한다. 단, 최악의 경우 5th order까지 해당되는 서버가 없을 시에는 일정시간(K msec) 기다린 후 RPC Caller를 다시 기동하여 서버들의 상태 정보를 재수신하여 결정.



(그림 4) RPC기반 웹 클러스터 구성 모듈

상기와 같이 실시간으로 정확한 real 서버의 상태를 파악하여 사용자들의 요청을 분산하기 때문에 서버들 간 부하편중 현상이 개선되어 더 효율적인 부하분산이 가능하다.

3.4 Prototype 구현을 위한 모듈 구성 및 기능

제안된 부하분산 방안을 구현하기 위한 클러스터 시스템의 Director와 real 서버의 모듈 구성은 (그림 4)와 같다.

각 모듈들의 기능과 동작은 다음과 같다.

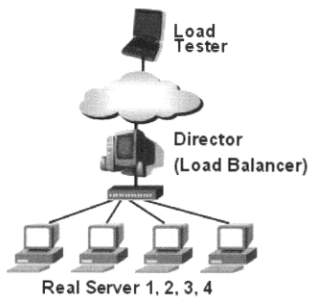
- 각 Real 서버의 부하상태 수집기 모듈은 Director에 RPC 서비스를 등록한 후 RPC caller 모듈 호출 시 (그림 1)과 같이 자신의 부하상태를 파악하여 결과를 전송한다.
- Director의 RPC caller 모듈은 사용자의 요청 수신 시 전 real 서버의 부하상태 수집기 모듈을 호출하는 broadcasting RPC 루틴이다.
- Load Distributor 모듈은 등록된 각 real 서버로부터 수신한 부하상태 정보를 이용하여 (그림 3)과 같이 가장 부하가 적은 real 서버에 사용자의 요청을 할당한다.
- Weight Calculator 모듈은 각 real 서버들이 서비스 등록 시 해당 서버의 CPU 성능, 메모리 량, 지원 네트워크 대역폭 정보를 수집하여 물리적 사양에 따른 가중치 요소를 계산하여 저장하고, (그림 3)에서와 같이 동일 order 발생 시 최고 성능의 서버를 선정한다.

4. 시험 및 결과 분석

4.1 시험 환경

본 연구에서 제안한 알고리즘의 성능 시험을 위해 비교대상으로 고정 가중치 방법과 접속수를 기반으로 한 기존의 WRR 및 WLC 알고리즘을 사용하였다. 성능 비교 방법은 각각의 부하 분산 알고리즘을 (그림 5)의 prototype 시스템에 적용하여 부하분산 정도를 측정하기 위해 3. 항에서 알아본 각 real 서버들의 가용 메모리 변화와 클러스터 시스템의 응답시간을 측정 비교하였다. 메모리의 고른 사용은 부하분산이 비교적 잘 되었다는 것과 응답시간 비교로 알고리즘간의 성능비교를 알 수 있기 때문이다.

시험 시스템 구성은 (그림 5)와 같으며, 본 연구에서는 시



(그림 5) 시험 시스템 구성도

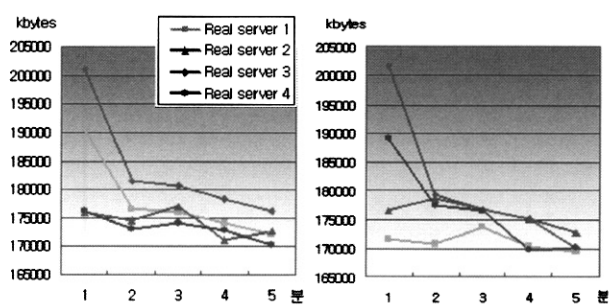
스템의 물리적 성능에 따른 고정 가중치 요소는 관심사항이 아니므로 real 서버들의 시스템 사양을 동일하게 하여 가중치 요소를 1로 고정하였다. Ultra Monkey Project 의 LVS (Linux Virtual Server) 웹 클러스터를 구성하였고, 논리적인 topology는 Direct routing 방식을 적용하였다[14]. load tester 서버에는 부하 생성과 모니터링이 가능한 OpenSTA 1.4를 설치하였다[15].

각 real 서버 상에 탑재되는 웹 서비스는 일반적인 html 문서와 CGI 게시판으로 구성하였으며, load tester에서 동시 접속자수를 100, 200, 300, 400으로 증가시키면서 각 알고리즘을 적용하여 웹 클러스터 시스템의 응답시간 및 각 real 서버의 가용 메모리 변화를 측정하였다. 한 알고리즘에 대한 실험이 끝난 후 이전 부하의 영향을 제거하기 위해 시스템을 재부팅하였다.

(그림 1)의 "CPU idle율"의 비교 기준값 X는 30%, "Collision ratio"의 Y는 10%로 설정하였으며, "Free Memory"에서의 Init_Average는 각 real 서버들의 free 메모리 값을 조사하여 100 Mbyte로 설정하였다.

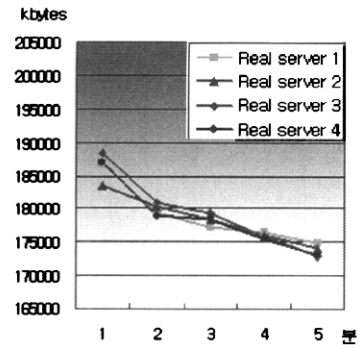
4.2 서버의 가용 메모리 변화 시험 결과 분석

실험 결과, 동시 접속자수가 100이하일 경우에는 기존 알고리즘 방식이나 제안된 알고리즘 방식 모두 각 서버의 가용메모리 변화 차가 크지 않았지만, 동시 접속자수가 점점 증가하여 400일 경우 (그림 6)에서 알 수 있듯이 기존 알고리즘에서는 각 서버의 가용메모리가 불균일한 반면, 제안 알고리즘의 경우는 (그림 7)과 같이 훨씬 균일하게 나타난다. 이는 제안 알고리즘에서 더 효율적인 부하분산이 이루어졌다는 것을 보여준다.



(a) WRR (b) WLC

(그림 6) 기존 알고리즘의 가용 메모리(400 기준)



(c) RPC-WLC

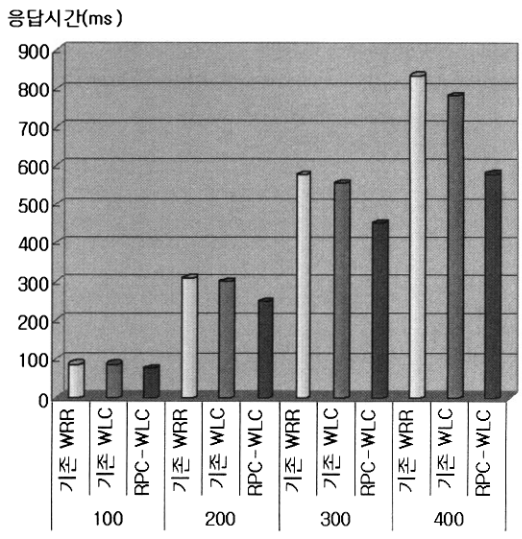
(그림 7) 제안 알고리즘의 가용 메모리(400 기준)

4.3 응답시간 시험 결과 분석

응답시간은 부하분산에 따른 클러스터 시스템의 성능을 시험하기 위해 real 서버에 구성된 웹 페이지에 접속 시 소요되는 응답시간을 50회 측정하여 평균한 것이다.

동시 접속자 수의 증가에 따른 알고리즘 별 평균 응답시간에 대한 실험에서 (그림 8)과 같은 결과를 얻었다. 동시 접속자수가 100일 경우 기존 알고리즘 WLC에 비해 제안한 알고리즘(RPC-WLC)에서 응답시간이 9.3ms개선된 반면, 동시 접속자수가 400으로 증가한 경우 203.1ms가 개선되었다. 즉, 동시 접속자수가 적은 경우 기존 알고리즘과 제안된 알고리즘의 차이가 별로 없으나, 동시 접속자수가 증가할수록 제안 알고리즘에서 더 빠른 응답시간을 나타냄을 알 수 있다. 이는 제안 알고리즘에서 부하분산이 더 효율적으로 이루어져 클러스터 시스템 전체의 성능이 보다 최적화되었다고 할 수 있다.

상기 실험 결과 기존의 가중치 고정 방식과 접속 수에 의한 부하분산 알고리즘 보다 본 연구에서 제안한 사용자의 접속 요구 시 각 real 서버의 정확한 부하 상태 정보를 실시간으로 파악하여 부하를 분산하는 방식이 더 효과적이라는 것을 알 수 있다.



(그림 8) 적용 알고리즘 별 응답시간 비교

5. 결 론

본 연구에서는 리눅스 기반 웹 클러스터 시스템에서 부하분산 시 특정 서버에 부하가 편중되는 현상을 줄이기 위한 효율적인 부하분산 방안을 제안하였다. 기존의 리눅스 기반 클러스터링 기법에서 연구된 동적 가중치 방법은 real 서버의 고정적인 물리적인 자원을 접속 수에 따라 평준화하려는 반면, 본 논문에서 제안한 알고리즘은 보다 정확한 real 서버의 부하상태를 파악하기 위하여 각 서버의 CPU 부하, 가용 메모리, 네트워크 부하를 실시간으로 측정하였고, 이를 바탕으로 가장 부하가 적은 real 서버에 빨리 사용자의 요청이 할당될 수 있도록 하는 효율적인 부하분산을 통해 해당 클러스터시스템의 성능을 최적화 시킬 수 있도록 하였다. 실시간으로 real 서버의 상태를 파악하기 위해 사용한 Director의 RPC Caller 모듈에서 서버들의 "Load Status Collector" 모듈을 호출하여 그 결과 값을 받는데 소요되는 시간은 동시 접속자 수가 400일 때 평균 약 0.2msec 이고 최대 1msec를 넘지 않았다. 이 값은 크지는 않지만 대규모 클러스터 시스템 구성 시에는 부분적인 오버헤드가 될 수도 있다.

실험 결과, Director에서 부하분산 시 보다 정확한 각 real 서버의 실시간 상태정보를 바탕으로 부하가 적은 서버에 사용자 요청을 할당하기 때문에 서버들 간 효율적인 부하분산과 응답시간 개선 효과를 얻을 수 있었으며, 동시 접속자 수가 증가할수록 부하부산 효과가 커지며 클러스터 시스템의 성능 역시 향상되는 결과를 나타내었다.

참 고 문 헌

[1] Wensong Jhang, The paper "Linux Virtual Servers for Scalable Network Services," Ottawa Linux Symposium, 2000.

[2] "Delivering High Availability Solutions with Red Hat Enterprise Linux AS 2.1," RedHat, 2003.

[3] Jian liu, Lorghu Xu, Baogen Gu, Jing Zhang, "A scalable, high performance cluster server," High performance computing in the Asia-Pacific region, 2000. Proceedings. The firth International Conference/Exhibition, Vol.2, pp.941-944, 2000.

[4] 권오영, "클러스터 시스템 개요" 기술·정책 동향, 한국과학기술정보연구원 소식지, 2000.

[5] John Bloomer, "Power Programming with RPC," O'Relly & Associates, Inc., pp.12-13, Sep., 1992.

[6] "Virtual Server Scheduling Algorithms," http://www.linuxvirtualserver.org/docs/scheduling.html, Dec., 2003.

[7] 권원상, 역, "웹 성능 최적화 by Patrick Killelea", 한빛미디어, 2000.

[8] Wimon ho, WayneM.Loucks and Ajit Singh "Monitoring The Performance of a Web Service," IEEE, 1998.

[9] L Cottrell and C Logg. "Tutorial on WAN Monitoring at SLAC," http://www.slac.stanford.edu/comp/net/wanmon/tutorial.html, March, 1997.

[10] 명원식, 장태무, "웹 서버 클러스터에서 내용 기반으로 한 부하분산 기법", 한국정보처리학회논문지A, 제10-A권 제6호, pp.729~736, 2003.

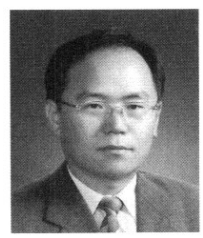
[11] 김석찬, 이영, "동적 가중치에 기반을 둔 LVS 클러스터 시스템의 부하 분산에 관한 연구", 한국정보처리학회논문지A 제10-A권 제2호, pp.302-303, 2001.

[12] http://www.sun.com/sun-on-net/performance.html

[13] http://www.junsoft.com/

[14] Simon Horman, "Linux Virtual Server Tutorial," http://www.ultramonkey.org/papers/lvs_tutorial/html/ July, 2003.

[15] Daniel Sutcliffe, Geoff Hall, Olaf Kock, Richard Clarke, "OpenSTA" http://sourceforge.net/projects/opensta, Sep., 2000.



윤 천 균

e-mail : chqyoun@honam.ac.kr

1982년 인하대학교 전자공학과(학사)

1997년 포항공과대학 정보통신학과 (공학석사)

2003년 조선대학교 전자계산학과 (이학박사)

1982년~1998년 포항종합제철(주) 과장

1998년~현재 호남대학교 인터넷소프트웨어학과 조교수

관심분야 : 네트워크, 정보가전, 정보보안, 공장자동화 시스템