Journal of Korea Multimedia Society Vol. 7, No. 8, August 2004(pp. 1097-1106)

웹 클러스터 시스템의 효율적인 동적 작업분배

서 경 룡†

요 약

전형적인 웹 클러스터 시스템은 여러 대의 실제서버와 클라이언트에 작업을 분배하는 가상서버로 구성된다. 본 논문에서는 고 가용성 웹 클러스터를 구성하기 위한 부하예측방식을 사용한 동적 작업 분배 방식을 제안하였다. 가상서버는 적절한 주기로 상태요청 메시지를 실제서버에 전송하여 부하 상태를 알아낸다. 하지만 작업분배 과정에서 실제의 부하상태의 변화를 정확히 알 수 없으므로 클라이언트의 요청에 대하여 실제서버의 부하상태를 예측하여 최소 부하상태 인 서버에 작업을 할당한다. 제안된 작업 분배방식은 실제서버의 부분적 고장과 무관하게 동작하며 부분적인 고장이 발생하여도 전체 시스템에 영향을 주지 않고 계속적인 서비스를 제공한다. 또한 다양한 성능평가를 수행하여 실제서버 확장에 따른 성능확장능력과 작업처리능력이 향상되었음을 보였다.

An Efficient Dynamic Load Distribution for the Web Cluster Systems

Kyungryong Seo†

ABSTRACT

The typical web cluster architecture consists of replicated real servers and a virtual server that routes client requests among the real servers. In this paper, we proposed an efficient dynamic load distribution method with load prediction for the web cluster systems. The virtual server transmit status request message to real servers in other to get load states. However the load states dose not accurate during load distribution, thus the virtual server predict the load status of real servers and assign a request of the client to the minimum loaded real server. The proposed distribution methods works not related to partial breakdown of real servers, thus the system works with high availability. We also show that the proposed distribution method preserve scalable property and improve the throughput through a set of simulations.

Key words: Web Cluster(웹 클러스터), Virtual Server(가상서버), Dynamic Load Distribution(동적작업분배), High Availability(고 가용성)

1. 서 론

웹의 개발로 인터넷의 사용이 쉬워지면서 일반 사용자들의 인터넷 활용이 급증하고 있다. 개인적인 목적으로 사용하는 인터넷 서버도 증가하는 추세이고 상업적 목적으로 운용되는 서버의 수도 나날이 증가하고 있다. 웹 서버는 비교적 단순한 프로토콜을 사용하여 HTML로 작성된 문서를 사용자의 요청에 제공하는 기능을 수행하며 개인적 서버의 경우 저렴한 가격의 컴퓨터로도 쉽게 구현이 가능하다. 하지만 상업적 서버나 고 수준의 서비스를 제공하는 서버의 경우 보다 고급화된 기능을 수행하여야 한다.

상업적인 서버가 요구하는 주요 기능은 고성능

※ 교신저자(Corresponding Author) : 서경룡, 주소 : 부산광역시 남구 대연3동 599-1(608-737), 전화 : 051)620-6885, E-mail : krseo@pknu.ac.kr
접수일 : 2004년 4월 21일, 완료일 : 2004년 6월 7일
† 정회원, 부경대학교 전자컴퓨터 정보통신공학부 부교수

[1-7]과 고 가용성이다[9,10]. 컴퓨터는 전자부품으로 구성되어 고장이 발생할 여지가 있는데 고장이 발생하면 서버는 지속적으로 서비스를 수행할 수 없으며 상업적 서버의 경우 매우 나쁜 결과를 초래한다. 고장을 원천적으로 봉쇄할 수는 없지만 고장이 발생하여도 외부의 사용자가 고장을 인지하지 못하도록 지속적인 서비스를 제공할 수 있다. 이를 고 가용성이라고 하는데 보통의 컴퓨터에서는 제공되지 않는 기능이다. 이러한 고성능, 고 가용성의 전용 서버는 일반적인 컴퓨터에 비하여 구현이 어렵고 가격 또한 매우 비싸다. 따라서 저렴한 가격으로 이러한 성능을 구비한 서버를 구축하는 방법에 관한 연구가 지속되어 왔다.

클러스터 기술은 이러한 문제를 해결하는 훌륭한 방법으로 여러 대의 컴퓨터로 통신망을 통하여 결합하여 특정한 작업을 수행하도록 구성된 시스템을 말한다. 클러스터는 여러 컴퓨터가 동시에 작업을 수행하기 때문에 뛰어난 성능을 제공할 뿐 아니라 내부의 컴퓨터에 장애가 발생하더라도 정상적인 동작이 가능한 여분의 컴퓨터가 지속적인 서비스를 제공하여 고 가용성을 얻을 수 있다.

클러스터의 구성에는 실제 작업을 수행하는 실제 서버와 작업의 분배와 관리를 수행하는 가상 서버가 있으며 각각에는 전문적인 기능을 수행하는 소프트웨어들이 필요하다. 외부의 사용자에게는 가상서버 한 대만 존재하는 것처럼 보이지만 서비스의 요청에 따른 응답은 클러스터 내부의 실제서버가 수행한다. 만약 모든 외부요청에 어떤 한 대만이 응답하도록 구성되었다면 클러스터 내부의 다른 서버는 아무런 일도 수행하지 않고 시스템 전체의 성능은 한 대의 컴퓨터보다 향상될 수 없다. 따라서 가상서버의 작업 분배방식이 시스템의 성능을 결정하게 된다.

[1]에서 Linux 시스템을 사용한 초기형태의 웹 클러스터 시스템을 보였다. 여기서는 확장 가능한 시스템을 구성하는데 주 목적을 두었으며 Linux 환경의 시스템에 잘 적용되어 웹 클러스터 시스템의 모태가 되었다. 하지만 시스템 구성을 편리하게 하기 위하여 동적인 작업분배방식은 전혀 고려하지 않았으며 성능평가도 이루어 지지 않았다. 이후 발표된 몇몇 시스템도 주로 정적인 분배방식을 사용하였는데 구현이 쉽고 시스템 구축에는 용이하지만 최적의 작업분배를 보장하지 못하고 실제서버의 고장을 알 수 없기 때문에 중요한 클러스터 시스템의 특징인 고 가용성을 얻기 어렵다[2,6].

최근에 연구되고 있는 동적인 분배방식은 가상서버가 실제서버와 통신을 수행하며 실제서버의 부하상태를 확인하고 최소부하의 서버를 선정하여 작업을 할당하는 방식이다. 하지만 상태를 알아내는 작업을 수행하기 때문에 부가적인 기능을 가진 프로그램의 설치가 필요하며 부하확인 요청과 응답과정이 시스템의 자원을 사용하는 작업이기 때문에 이로 인한 성능저하가 발생할 수 있다[3-7,9].

본 논문에서는 시스템이 확장성과 고 가용성을 보장하기 위하여 새롭게 웹 클러스터 시스템을 구축하고 여기에 적합한 동적 작업분배 방식을 제안하였다. 동적 작업 분배방식에는 가상서버와 실제 서버사이에 적절한 메시지 교환이 필요하게 된다. 본 논문에서는 표준화된 인터넷 프로토콜인 SNMP(Simple Network Management Protocol)를 활용한다. 가상서버는 주기적으로 실제 서버에 SNMP 메시지를 전송하여 실제서버의 부하 상태를 알아낸다. 작업이 할당된 서버의 부하는 변하게 되는데 제안된 시스템에서는 부하의 변화량을 예측하여 변경시킨다. 따라서 다음 주기에 부하 값을 정확하게 알 때 까지 오류를 줄인다. SNMP 메시지를 통하여 고장을 쉽게 알아낼 수 있으며 손쉽게 작업할당을 배제 할 수 있어 고 가용성의 특성을 얻는다.

본 논문은 2장에서 웹 클러스터 구성에 관한 일반적인 연구결과를 소개하며 3장에서는 제안된 시스템의 구성 방법을 보이고 효율적인 동적 작업할당방법을 제시한다. 시뮬레이션 결과와 분석이 4장에서 다루어지고 5장 결론으로 끝을 맺는다.

2. 관련 연구

웹 서비스를 위한 클러스터 시스템에는 DNS를 이용하여 구축한 시스템과 IP/TCP/HTTP 재전송 방식을 이용한 시스템이 연구되어 왔다[1,6].

DNS를 이용한 시스템은 R-R DNS라는 확장된 DNS 개념을 사용하는데 클라이언트에서 웹 페이지를 요청하기 위하여 특정한 서버의 IP 주소를 DNS 게이트웨이에 요청하면 DNS게이트웨이 는 내부의 서버들의 IP 주소를 관리하는 R-R DNS 서버에 IP를 요청한다. 이때 R-R DNS 서버는 실제서버 중에서 하나의 IP주소를 지정하여 응답하면 클라이언트는

그 서버에 접속된다. R-R DNS는 보통 라운드 로빈 방식으로 내부서버를 지정하게 되는데 외부의 요청이 내부의 서버들에 순차적으로 분배한다.

이 방식은 내부의 서버들에는 별도의 시스템이 필요하지 않고 일반적인 서버를 그대로 사용 가능하다. 또한 R-R DNS 기능을 구현하는데도 큰 어려움이 없기 때문에 시스템 구축이 매우 쉬운 편이다. 하지만 DNS의 경우 보다 좋은 성능을 얻을 수 있는 스케줄링 방식을 적용하기 어렵고 비록 적용한다 하더라도 성능 문제로 IP 주소를 캐시 하는 경우가 일반적이기 때문에 좋은 성능을 보장하기 어렵다.

R-R DNS 방식의 단점을 해결할 수 있고 보다 다양한 작업분배 전략을 적용할 수 있도록 연구되고 있는 구성방법은 그림 1의 IP/TCP/HTTP 전환 방식이다.

여기서는 외부의 클라이언트는 시스템의 가상서버만을 인식한다. 따라서 클라이언트는 가상서버에게 HTTP 메시지를 전송하여 웹 페이지 요청을 한다. 이때 HTTP 접속을 요청 받은 가상서버는 자신이 웹 서비스를 지원하지 않고 시스템 내부의 실제 서버들에서 하나를 선정하여 HTTP 접속을 넘겨버린다. 그 이후의 HTTP 전송은 클라이언트와 접속을 넘겨받은 실제 서버에서 이루어진다.

HTTP는 IP나 TCP의 상위계층 프로토콜로 접속 서버의 전환은 TCP 계층이나 IP 계층에서도 수행이 가능하다. 가상서버는 강력한 성능이 필요하지 않기 때문에 일반적인 컴퓨터를 활용하거나 비교적 단순한 기능의 전용 하드웨어를 사용할 수 있다. 서비스를 직접 담당하는 실제서버는 가상서버가 분배한 서비스만 지원하면 되고 가상서버가 잘 동작 한다면 전체적인 서비스 수행능력은 실제서버의 수에 비례하여 성능이 향상된다. 또 시스템 구성에서 볼 수 있듯이 가상서버는 실제서버와 네트워크를 통하여 연결되어 있으므로 실제서버의 상태를 파악하여 고 가용성을 얻거나 실제서버에 대한 원격관리기능을 얻을 수 있는 여지가 있다.

이 형태의 시스템에서 특별히 중요한 것은 가상서버의 동작이다. 가상서버는 내부의 자원을 효율적으로 관리하고 외부의 서비스 요청에 따른 작업들을 내부의 실제서버에 잘 분배하여야 시스템의 성능을 높일 수 있다. 이를 위한 부하 분배 방식은 크게 정적 방식과 동적방식으로 분류된다.

정적 분배방식은 가상서버가 실제서버의 상황을 전혀 고려하지 않고 초기 설정과 지난 분배과정을 고려한다. 앞에서 설명한 R-R DNS의 경우도 이와 유사한데, 라운드 로빈, 가중치 기반 라운드 로빈 스

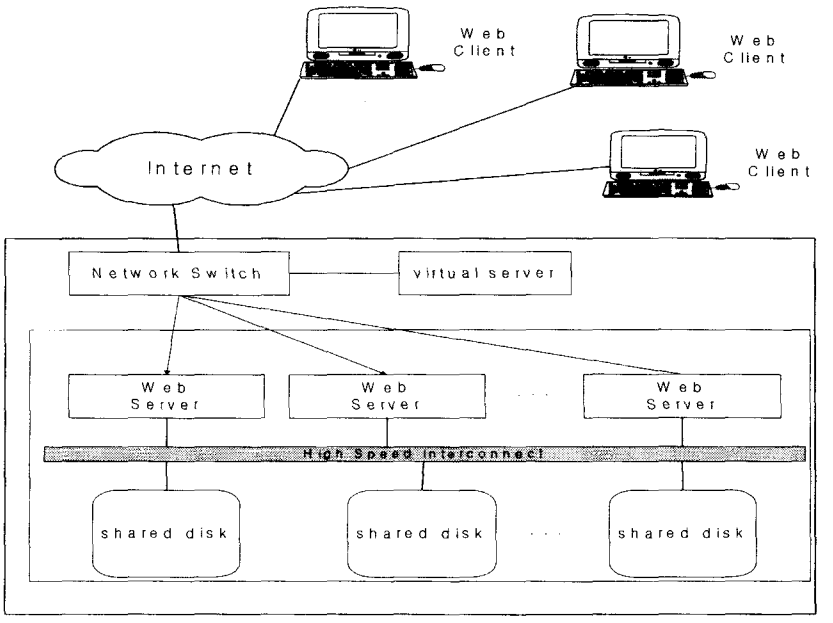


그림 1. IP/TCP/HTTP 전환방식의 웹 클러스터.

케줄링, 최소 접속 스케줄링, 가중치 기반 최소 접속 스케줄링 등이 있다. 또한 최근에 연구된 FLEX 도 정적 방식의 하나이다.

라운드 로빈 스케줄링 : 사용자의 요구를 차례대로 각 서버에 균등하게 분배하는 방식으로 클라이언트의 연결에 대해 각 실제서버에 순차적으로 부하 부산을 한다.

가중치 기반 라운드 로빈 스케줄링 : 각 실제서버가 서로 다른 시스템으로 구성되어 있을 때 각 서버의 처리 능력에 따라 가중치를 부여하여 라운드 로빈 방식처럼 부하 분산을 수행한다. 각 서버에 가중치를 부여할 수 있으며, 여기서 지정한 정수 값을 통해 처리 용량을 정한다. 기본 가중치는 1이다. 예를 들어 실제 서버가 A,B,C 이고 각각의 가중치가 4,3,2일 경우 스케줄링 순서는 ABCABCABA가 된다.

최소 접속 스케줄링, 가중치 기반 최소접속 스케줄링 : 사용자가 연결된 실제서버에서 가장 적은 숫자의 연결이 이루어진 서버로 부하를 할당하는 방식이다. 각 서버에서 동적으로 실제 접속한 숫자를 알아야 하므로 시스템 구성이 복잡해진다.

FLEX : 최근에 제안된 FLEX는 기존의 방식이 현재의 상황에만 의존하는데 반하여 그림 2와 같이 지난 스케줄 결과를 로그파일에 저장하여 두고 지난 상황을 고려하여 실제서버를 결정하는 정적 방식의 하나이다.

그림 2의 flex-alpha는 작업분배 알고리즘으로 서비스를 요청하는 클라이언트의 수와 실제서버의 수를 고려하고 저장된 로그를 참조하여 최소의 부하를 가진 실제서버를 현재요청중인 클라이언트에 할당한다. flex-alpha 알고리즘에서 각 서버의 성능과 현재부하를 고려하므로 기존의 방식보다 성능이 우수한 것으로 알려져 있다[2].

동적 방식은 정적방식이 실제서버를 선정하는 과정에서 실제서버의 부하나 상태를 확인하지 않고 가상서버가 임의로 정하는데서 발생하는 문제를 해결하기 위하여 만들어 졌다[3-5]. 실제 서버 중 특정한 한 대의 서버를 선정하여 작업을 할당하기 위하여 현재 최소의 부하 상태인 서버를 결정하는 것이 매우 중요하다. 각 서버의 부하는 현재 할당된 작업의 수와 종류에 따라 변한다. 정적인 방식은 주로 작업의 수에 의존하여 분배를 수행하는데 알고리즘에 따라 결정된 서버가 최소부하 상태라고 보기 어렵다. 가장 좋은 방법은 현재 각 서버의 부하상태를 확인한 다음 최소의 부하인 서버를 결정하여 여기에 작업을 할당하는 것인데 바로 동적인 방식이다.

동적인 방식의 문제점은 각 서버의 부하를 확인하는데 있다. 서버의 부하를 확인하기 위하여 가상서버는 실제서버와 메시지를 교환하여야 하는데 이것이 새로운 부하를 발생시킨다. 또한 메시지교환 시 전달지연이 발생하면 서버의 부하는 순간 변화하므로 획득한 부하 상태가 지속적으로 유지되지 않는다. 따라서 시스템 성능을 최대한으로 유지하면서 최적의 할당을 수행할 수 있는 전략이 필요하다. 동적인 방식으로 얻을 수 있는 또 다른 장점은 서버의 장애를 알아낼 수 있다는 것이다[9]. 메시지를 전달하지 못하는 서버나 장애상황을 알리는 메시지를 전달한 서버는 장애가 발생한 것으로 판단하고 서버 풀에서 배제시킨다면 모든 서버가 고장 나지 않는 한 웹 서비스를 지속적으로 수행할 수 있다.

3. 고 가용성 웹 클러스터 시스템

3.1 하드웨어 구성

본 논문에서 고려한 시스템 의 기본 구성은 그림 3과 같다. 시스템에 사용된 모든 서버는 Linux를 탑재한 IBM 호환기종의 컴퓨터를 사용하였다.

위의 그림 3과 같이 가상서버와 실제서버모두 네트워크 스위치를 통하여 인터넷에 접속되어 있다. 이러한 구성은 전송변경을 통하여 가상서버는 응답메시지를 전송하는데 관여하지 않고 실제서버가 직접

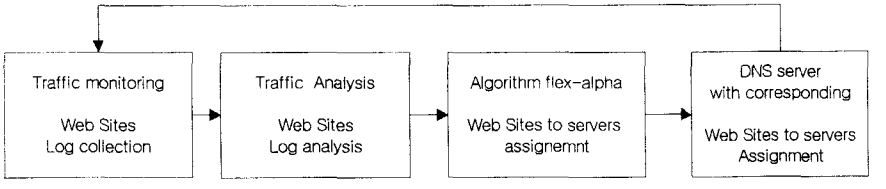


그림 2. FLEX의 동작개요.

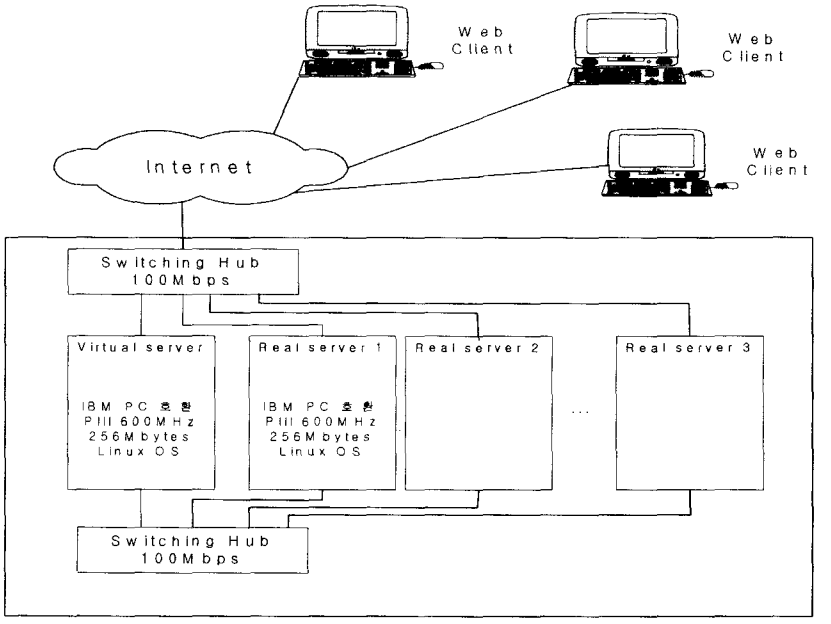


그림 3. 웹 클러스터 시스템

클라이언트에 응답할 수 있기 때문에 좋은 성능을 얻을 수 있는 방식이다. 또한 각 실제서버와 가상서버는 내부의 사설네트워크를 통하여 연결되어있는데 가상서버와 실제 서버간의 메시지 전송은 이를 통하여 이루어진다.

시스템 구성에 필요한 소프트웨어는 가상서버와 실제서버가 서로 다르다. 그림 4와 같이 실제서버에는 실제로 웹 서비스를 수행할 수 있도록 Apatch 웹 서버를 설치하였다. 시스템 구성에서 가장 중요한 부분은 가상서버와 실제 서버와의 메시지 교환기능을 담당하는 부분과 작업분배를 수행하는 분배 모듈이다. 몇몇의 연구에서는 전용의 프로그램을 구성하거나 원격명령을 수행하여 해결 하였는데 본 논문에서는 시스템 구성이 용이하고 다양한 기능을 수용 할

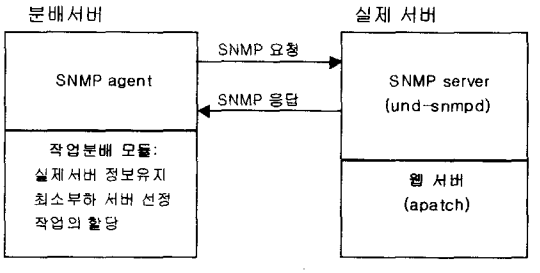


그림 4. 가상서버와 실제서버의 내부구성.

수 있도록 SNMP 프로그램을 활용하였다.

가상서버는 클러스터에 연결된 모든 실제서버의 부하상태를 사설네트워크를 통하여 알아내는데 이를 위하여 가상서버에는 Linux용으로 개발된 ucd-snmp 라이브러리를 사용하여 SNMP 에이전트를 구성하였다. SNMP 에이전트는 SNMP 요청 메시지를 실제 서버에 전송하여 실제서버의 상태를 확인할 수 있다. 또한 수집된 정보를 이용하여 효율적인 분배작업을 수행하는 분배모듈을 작성하였다. 시스템 구성에서 실제 새롭게 작성된 부분은 가상서버의 이 부분이다. 실제서버에는 미리 만들어진 SNMP 에이전트를 설치만 하면 된다.

3.2 부하분배

가상서버는 실제서버의 상태를 알아내고 이 정보를 유지하며 외부의 요청에 따라 최적의 실제서버를 찾아서 작업을 할당하는 과정을 수행하여야 한다. 본 시스템에는 SNMP 에이전트를 활용한 동적 분배 방식을 채택하였다. 따라서 각 실제서버의 부하 값을 유지하는 작업이 매우 중요한데 여기에는 해결하기 어려운 문제점이 있다.

우선 단순한 다음의 작업분배 시나리오를 살펴보자.

(1) 가상서버가 모든 실제서버에 SNMP 메시지를 전송하여 부하 값을 요청하면 실제서버는 자신의 부하 값을 가상서버에 알린다. 부하 값을 전송 받은 가상서버는 이 값을 기록 유지한다.
(2) 만약 외부의 요청이 발생하면 최소의 부하 상태인 가상서버를 선정하여 그 서버에 작업을 할당한다.

웹 클러스터 시스템의 작업 분배에서 가장 중요한 점은 최소부하의 실제서버를 찾아서 여기에 작업을 할당하는 것 이다. 그런데 위의 시나리오에서는 실제 서버에서 얻은 부하 값이 시간에 따라 변화하기 특정한 시점에서 최소부하상태인 서버를 선정하기 어렵다. 또한 SNMP 요청을 수행하기 위하여 실제서버는 시스템 자원을 사용하게 되고 이 결과로 자신의 부하 값이 상승하게 된다. 이와 같이 단순한 작업분배 방식은 실제로 잘 적용되지 않는 것을 알 수 있다.

부하변화 상태를 정성적으로 살펴보기 위하여 서버i의 총 부하를 Δ_i라고 표시한다. 서버i에는 여러 개의 작업이 할당되어 수행 되고 있으며 특정 작업이 종료 되거나 새로운 작업이 부가 되면 이 값은 변하게 된다. Δ_i의 단위는 특정한 값으로 표시하기 매우 어렵지만 보통의 경우 시스템에서 제공하는 부하 값을 사용하거나 가용메모리 양으로 나타내기도 한다.

가상서버에서 서버i 에 작업을 할당하면 Δ_i는 증가 하고 이미 분배된 작업이 종료되면 Δ_i감소하게 되므로 특정한 시점에 어떤 서버의 Δ_i를 정확히 알아내는 것은 불가능하다. 제안된 시스템 에서도 Δ_i를 알기 위하여 SNMP 요청 메시지를 서버 i에 전달하게 되고 이 메시지에 응답하기 위하여 SNMP 서버가 동작하게 되며 이때 Δ_i는 증가한다. 뿐만 아니라 이 응답을 분배 서버가 수신하였을 때 일정한 시간 지연이 발생하고 이때 종료한 작업이 발생한다면 역시 수신된 값은 정확한 서버 i 의 부하 값을 나타내지 못한다. 이러한 점들을 고려하더라도 Δ_i는 서버 의 부하상태에 관한 상당한 정보를 가지므로 이를 부하 분배에 중요한 단서로 삼을 수 있다.

동적인 부하 분배에서 중요한 점은 현재 부하 값을 알아내기 위한 메시지 교환을 최소화 하면서 최소의 부하 값을 가진 서버를 선정 하는 것이다. 예를 들면 두 대의 실제서버 에서 $\Delta_1=4, \Delta_2=5$ 일 때 서버1의 부하가 서버2보다 낮기 때문에 서버1에 작업을 할당하게 된다. 하지만 지속적으로 서버1에 부하를 할당하면 서버1의 부하는 계속 증가하게 되고 서버2의 부하는 작업의 종료로 감소하게 되어 사실상 서버2의 부하가 낮아지게 된다. 이러한 부하의 변화를 메시지 교환을 통하여 감지하지 못한다면 부가가 높은 서버에 부하를 집중시키는 경우가 발생한다. 메시지 교환을 자주 수행한다면 이러한 문제는 해결되겠지만 서버에 불필요한 부하를 가중시키게 되므로 이 방법 또한 좋지 않다.

제안된 시스템은 각 기능을 모듈로 구성하여 쉽게 다양한 분배 방식을 적용할 수 있다. 그림 4에서 알 수 있듯이 메시지 교환 모듈의 기능을 정지하고 작업 분배 모듈을 정적인 분배방식을 사용하도록 하면 [1,2]에서 제안한 정적분배 시스템이 된다. 또한 메시지 교환 부분을 가동하고 작업분배 모듈을 적절히 구성하면 다양한 동적 분배 시스템의 구성이 가능하다.

여기서는 [4]에서 구성한 동적 분배 시스템을 단순분배방식이라 하고 다음과 같이 구현하였다.

단순분배방식

(1) 가상서버는 일정한 시간간격으로 모든 서버i, $i=1,2,\ldots,N$,에 대하여 SNMP 요청을 수행하여 Δ_i 값을 알아낸다.
(2) 이 값을 토대로 다음과 같이 서버 i의 동적 가중치 W_i를 결정한다.

$$\Delta_T=\sum_{i=1}^{N}\Delta_i,$$
$$W_i=\frac{\Delta_i}{\Delta_T}$$

(3) W_i 값을 토대로 가중치 기반 정적 스케줄링을 수행한다.

단순 분배 방식은 단위시간별로 서버의 부하를 측정하여 그 상태의 부하 값으로 각 서버의 가중치를 계산한 다음 단위시간 동안 계산된 가중치를 토대로 가중치 기반 정적 스케줄링을 수행 하는 방식이다. 이 방식은 [4]에서 제안된 방식과 동일하며 구조가 단순하고 구현이 용이하지만 부하의 급격한 변화에 잘 적응 하지 못하는 단점이 있다. 즉 동적 가중치 W_i 가 최소인 서버 i는 단위분배기간에는 동적가중치가 유지되므로 부하가 집중되는 경향을 보인다.

따라서 이러한 방식은 부하측정 단위시간을 조정해야 하는데 부하의 요청이 많아질수록 측정 단위시

간을 짧게 가져가야 한다. 이는 부하의 불필요한 부하의 증대를 가져오며 전체 성능이 저하된다. 또한 실제 서버에 장애 가 발생하면 계산된 동적 가중치가 잘 적용되지 않는다. 그 이유는 동적 가중치가 전체 시스템의 부하상태와 상관관계가 있기 때문이다.

다음은 이러한 문제점을 극복하기 위하여 본 논문에서 고려한 전략이다.

● 실제서버에 대한 SNMP 메시지 전송빈도를 최소로 유지한다. SNMP 메시지는 실제서버의 부하를 상승시킨다. 실험을 통하여 SNMP 응답에는 10Kbyte의 정적 웹 페이지 전송과 비슷한 부하를 유발하는 것을 알 수 있었다. 하지만 SNMP 전송을 요청하였을 때 얻은 부하 값은 매우 정확한 값이다. 따라서 정확한 부하 값을 얻을 수 있으면서 실제서버에 부하를 최소화하는 전송구간을 구하여야 한다.

● 가상서버는 실제서버를 선택할 때 보관하고 있는 부하 값을 보정한다. SNMP 메시지의 응답으로 얻은 부하 값은 모든 실제서버에 동일한 조건이므로 정확한 부하상태를 알린다고 볼 수 있다. 하지만 시간이 지나면서 부하 값은 실제와 다르게 된다. 새로 요청 받은 작업을 수행하면 부하가 상승하고 작업을 끝내면 부하가 감소한다. 이렇게 변화된 실제서버의 부하상태를 가상서버는 알지 못한다. 따라서 부하변화를 예측하여 적절한 보정을 수행한다.

● 가상서버와 각 실제서버와의 관계를 독립적으로 유지한다. 고 가용성 시스템에서는 실제서버에서 고장이 발생하여도 전체 시스템에 영향을 주어서는 안 된다. 또한 새로운 실제서버를 추가하여 시스템 성능을 향상시킬 때에도 실제 서버간의 연관이 없도록 시스템을 구성하는 것이 좋다.

다음의 부하예측 분배방식은 앞에서 설명한 전략을 채택하였다.

부하예측 분배방식

(1) 가상서버는 일정한 시간간격으로 모든 서버i, $i=1,2,\ldots,N$,에 대하여 SNMP 요청을 수행하여 Δ_i값을 알아낸다.

(2) 각 서버i, $i=1,2,\ldots,N$, 에 대하여 부하밀도 ξ_i를 다음과 같이 구한다.

$$\xi_i = \frac{|\Delta_i^{old} - \Delta_i^{new}|}{\sum job_i}$$

여기서 Δ_i^{old}는 서버 의 바로 전의 SNMP 요청으로 구한 부하 값 이며 Δ_i^{new}는 지금 구한 부하 값이다. 또한 $\sum job_i$는 서버i에 지난 구간 할당된 작업의 총 수 이다.

(3) 서비스 요청이 발생하면 가상서버는 각 서버에 대하여 Δ_i가 최소인 서버i를 찾아 작업을 할당한다. 작업이 할당된 서버i에 대하여 Δ_i를 다음과 같이 변화 시킨다. $\Delta_i = \Delta_i + \xi_i$.

부하 예측 분배방식에서는 SNMP 메시지 교환을 통하여 서버의 부하를 알게 되는 구간부터 다음의 SNMP 메시지 교환을 수행할 때까지의 단위 구간에서 각 서버에 대한 부하의 변화량을 고려 적절한 보상을 수행한다. 즉 부하가 할당된 서버는 부하가 증가 하게 될 것 이므로 그 값을 예상하여 더하여 현재 부하 값을 예측한다. 이때 가상서버는 보상된 값을 토대로 최소 부하 서버를 예측하여 새로운 작업을 수행할 서버를 결정한다. 이때 중요한 점은 작업이 할당된 서버의 부하를 어떻게 보상 하는가 하는 것이다. 또한 작업을 할당 받지 않은 서버도 부분적으로 수행하던 작업의 종료로 인하여 부하 값이 변한다. 물론 앞에서 제시한 방식은 이러한 문제점을 완벽하게 고려 할 수는 없다. 하지만 SNMP 메시지 교환전의 일정한 구간에서만 예측을 수행하여 작업을 할당하기 때문에 비교적 단기간에만 최소부하의 서버를 예측하면 소기의 목적을 달성할 수 있다.

앞에서 제안된 부하 밀도 ξ_i는 서버i 에서 작업과 부하와의 상관 변화를 나타내는 값이다. 단순하게 부하밀도는 단위기간 부과된 작업 수에 따른 부하의 변화로 물리적인 의미는 하나의 작업이 서버에 발생시킨 부하변화이다. 만약 단위구간에서 100번의 작업할당을 수행하여 5의 부하변화를 가져 왔다면 이때 부하밀도는 5/100 = 0.05가 된다. 이때의 의미는 한번의 부하할당이 서버에 0.05의 부하를 증가 시켰다고 말할 수 있지만 이것이 항상 옳은 것은 아니다. 실험에 의하면 서버는 적절한 부하 값에서 지속적인 작업을 할당 받을 때 부하의 변동이 없는 평형상태에 도달한다. 즉 부하 밀도가 0인 경우도 가능하다. 이 경우는 작업의 할당과 작업의 종료가 균등하게 발생할 때 발생한다.

서버의 작업할당에 따른 부하 변화를 고찰하여 보면 최소부하의 상태에서 작업을 할당하면 부하가 증

가하기 시작한다. 일정한 주기로 부하를 할당하면 그 부하 값을 유지하며, 부하밀도는 0, 그보다 느린 주기로 부하를 할당하면 다시 부하가 감소하며 보다 빠른 주기로 부하를 할당하면 부하밀도는 증가하며 부하 값은 상승한다. 여기서 알 수 있는 것은 부하 값이 클수록 보다 하나의 작업이 부하의 변화량에 미치는 영향이 적다는 것이다. 부하 밀도 값은 100회의 부하가 1의 부하 값을 변화 시켰을 때는 0.01이지만 1000회의 부하가 1의 부하를 변화 시켰을 때는 0.001로 낮아짐을 알 수 있다. 앞에서 제안한 부하밀도에는 작업의 종료에 따른 부하 양의 감소분은 전혀 고려하지 않았다. 작업종료에 의한 변화분도 고려하면 예측 부하 값이 실제 부하의 변화와 비슷하게 되고 더욱 정밀한 부하예측이 되겠지만 계산 알고리즘이 지나치게 복잡해지고 이점을 고려하지 않더라도 각 서버에서 종료되는 작업의 비율은 비슷하므로 최소부하의 서버를 결정하는 데는 큰 문제가 없다.

4. 실험결과 및 분석

제안된 부하분산 방식을 검정하기 위하여 다양한 환경에서 시뮬레이션을 수행하였다. 시뮬레이션 환경은 그림 3과같이 구성된 실제 시스템으로 하였으며 실제서버를 8대 연결하였다. 성능평가에 사용되는 부하의 척도로는 메모리양이나 시스템에서 제공하는 시스템부하 값을 사용할 수 있다. 시스템 부하 값은 가상서버가 서버에 SNMP 요청 메시지를 보내어 알게 되는데 메모리양을 부하 척도로 활용할 경우에는 enterprises.ucdavis.memory를 요청한다. 만약 시스템에서 제공하는 부하를 사용할 때는 enterprises.ucdavis.laTable.laEntry.laLoad.1를 요청한다.

본 논문에서는 각각의 경우로 실험하였으나 그 결과가 유사하여 여기서는 사용하기 간편한 시스템 부하를 평가 척도로 사용하였다.

부하 생성은 WebLoad라는 웹 부하 생성 응용 프로그램을 이용하여 요청 빈도를 다양하게 변화시키면서 서비스 수행 여부를 조사하였다. 또한 공정성을 기하기 위하여 실제서버에는 3종류의 웹 서비스를 설치하였는데 작업 A는 단순한 HTML 문서 요청 서비스 이며 약 14.5Kbytes 의 문서를 전송한다. 이 값은 웹 서버의 성능을 평가하는 SpecWeb96[8]의 평균 문서 크기이다. 작업 B는 단순한 CGI 프로그램이며 서버의 약간의 CPU 부하를 요구 하며 1Kbytes의 문서를 전송한다. 세 번째 작업 C는 데이터베이스 쿼리를 수행하며 상당한 CPU 서비스를 필요로 하며 전달 문서는 100Bytes 정도 이다.

성능비교를 위하여 3가지의 분배 방식이 채용하였다. [1,2]에서 제안된 정적분배방식과 [4]에서 제안된 동적 분배방식을 시스템에 맞게 재구성한 단순분배방식, 그리고 부하예측 분배 방식이다. 실험에 중요한 요소로 메시지 교환 주기가 있는데 이 값은 동적 분배작업의 성능과 밀접한 관계가 있다. 최적의 교환주기를 구하는 것은 불가능하고 다양한 주기를 사용하여 성능평가를 수행하였는데 실험결과 0.3초에서 1.4초 정도의 주기에는 크게 변화를 보이지 않고 이보다 길게 되면 단순분배방식은 성능이 급격히 저하 한다. 본 실험에서는 적절한 값인 0.5초를 선정하였다.

시스템의 성능 확장 능력을 알아보기 위하여 실제 서버의 대수를 변화시키며 WebLoad 를 사용하여 5분간 지속적인 작업요청을 수행하도록 하여 성공한 작업의 초당평균을 구하였다. 그림 5는 실험결과를 보여 주는데 실험 1은 서버의 작업을 모두 A 타입으로 하여 수행하였으며 실험 2는 작업을 A, B, C 타입을 동일한 비율로 발생시켜서 수행하였다.

실험 결과에서와 같이 서버가 확장 될수록 초당

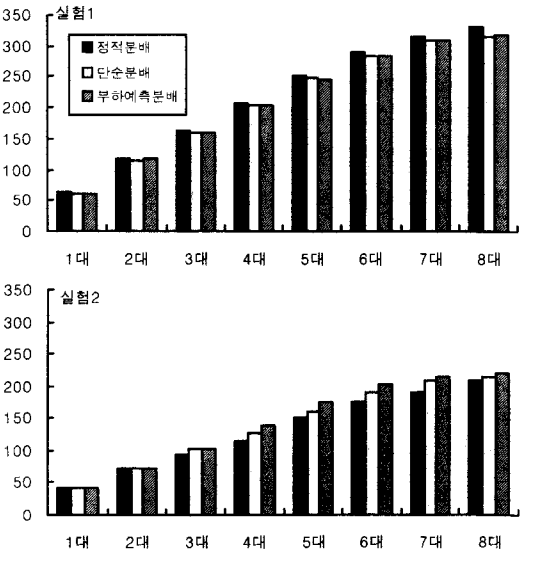


그림 5. 서버 확장에 따른 처리량 변화.

처리량이 증대되었음을 볼 수 있었다. 이 결과는 이미 예측 되었으며 실험 1의 경우 1대의 서버를 사용할 경우 평균적으로 초당 62회 정도의 처리를 수행하였다. 이 수치는 한대의 서버에서 수행할 수 있는 최대 처리량에 해당된다. 서버의 확장에 따라 비교적 균등한 성능 향상이 이루어지는 것을 알 수 있는데 6대 이상의 서버를 접속한 경우에는 성능향상 폭이 눈에 뜨이게 감소한다. 조사결과 이는 가상서버에 집중되는 작업요청에 따른 TCP 접속이 증가하기 때문이었다. 즉 직접적인 작업을 처리를 수행 하지는 않지만 가상서버의 성능이 전체 시스템의 영향을 미치는 것으로 조사되었다.

실험 1에서는 정적 분배방식을 사용하는 것이 가장 좋은 결과를 보인다. 라운드 로빈 방식에 의한 순차적인 작업 분배가 동일한 성능의 시스템과 동일한 작업인 경우에는 최적이고 동적인 방식에서는 SNMP 메시지 교환에 따른 부가작업이 필요하므로 필연적인 결과로 보인다.

이러한 점을 고려하여 수행한 실험 2에는 부하 량이 서로 다른 3종류의 작업을 분배하도록 하였는데 이 결과는 부하의 적절한 분배가 시스템의 성능에 중요한 요소로 작용하는 것을 보여준다. SNMP 메시지 교환에 따른 부가 작업에도 불구하고 동적인 분배방식이 정적인 방식보다 뛰어나며 같은 동적인 방식이라도 작업분배 방식에 따라 성능차이가 나타나는 것을 볼 수 있다. 실험결과 부하예측 분배방식이 다른 방식에 비하여 6% 에서 10% 정도 우수한 처리능력을 보였다.

분배방식이 최적으로 동작하면 실제서버의 부하가 균등하게 분산되어야 한다. 이를 측정하기 위하여 실험 3에서는 3대의 서버를 연결하고 실험 2와 같은 형태의 작업을 분배할 때 각 서버의 부하에 대한 분산 값을 나타내었다. 각 서버의 부하상태를 초단위로 측정하여 다음의 식으로 분산을 계산하였다

$$\sigma^2=[(m-\Delta_1)^2+(m-\Delta_2)^2+(m-\Delta_3)^2]/3$$
$$m=\Delta_1+\Delta_2+\Delta_3)/3$$

그림 6에서는 초기에 3대의 서버는 부하 평형상태를 유지하다가 가상서버로부터 작업이 할당되면 평형상태가 깨어지고 시간의 변화에 따라 분산이 변하는 것을 볼 수 있는데 정적분배 방식이 가장 나쁘며 부하예측 분배방식이 단순분배 방식보다 좀더 나은 결과를 보인다.

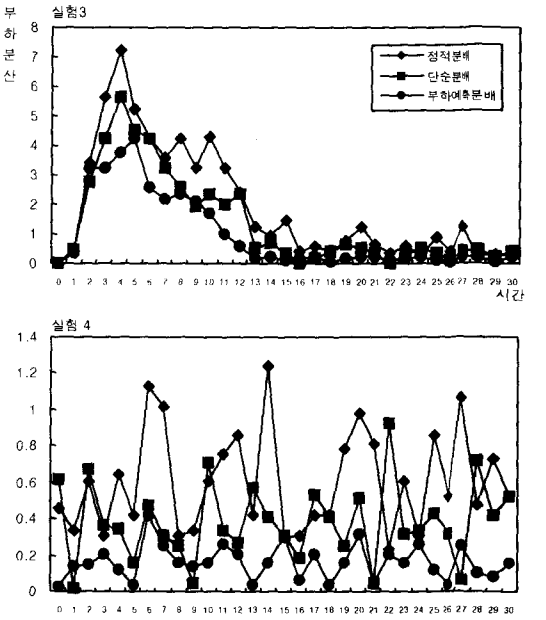


그림 6. 시간 변화에 따른 부하분산.

실험 4는 요청부하를 무작위로 선정하고 일정한 시간이 지연된 후의 30초간 부하 변화를 보여 준다. 실험 결과에서와 같이 정적 분배 시스템은 부하의 변화가 심할 때 각 서버의 부하상태 가 균등하지 않음을 쉽게 확인할 수 있다. 반면에 제안된 방식은 부하의 상태변화를 감안하여 작업을 할당할 수 있으므로 각 서버가 비교적 균등한 부하상태를 유지하는 것을 볼 수 있다. 이 실험은 각 서버의 처리능력이 동일한 경우인데 만약 서버의 처리능력이 다르다면 더욱 큰 차이를 보이게 됨은 자명하다.

5. 결 론

본 논문에서는 웹 클러스터 시스템에 효율적인 동적인 부하분배 방식을 적용할 수 있도록 SNMP을 기반으로 한 분배시스템 구성을 제안하였다. SNMP는 잘 알려진 표준 프로토콜로 시스템에 어렵지 않게 설치할 수 있다. SNMP가 설치된 시스템에서 얻을 수 있는 이득은 서버의 부하상태뿐 각 서버의 고장유무도 알아 낼 수 있다. 만약 실제서버에 장애가 발생하면 그 서버에는 작업이 할당되어서는 안 된다. 정적인 방식에서는 실제서버의 고장을 알아낼 방법이 존재하지 않으며 이러한 경우에도 장애가 발생한 서버에 계속적인 작업분배가 이루어진다. 제안된 시

스템은 SNMP의 사용으로 서버의 장애를 확인하여 작업분배를 중지하고 관리자에게 서버의 장애를 통보할 수 있다.

동적인 작업분배가 가능한 경우에도 작업분배방식에 따라 성능이 달라질 수 있다. 본 논문에서는 효율적인 작업 분배방식을 제안하였는데 각 실제서버와 상호작용 없이 가상서버와의 메시지 전달만으로 분배 작업을 수행한다. 따라서 실제서버의 고장이 전체시스템에 영향을 미치지 게 되므로 고 가용성의 특성을 얻을 수 있다. 또한 제안된 분배방식의 성능을 평가 하기 위하여 몇 가지 시뮬레이션을 수행하였다. 우선적으로 시스템 확장에 따른 성능개선의 정도를 알아보았는데 5~6대 정도의 확장까지는 비교적 예상대로 시스템 처리능력이 확장되었다. 하지만 그 이상의 시스템을 확장한 경우에는 가상서버의 처리능력으로 인하여 지속적으로 성능이 향상되지 않는 것을 알 수 있었다. 또 다른 실험으로 분배방식의 균등분배정도를 측정하였는데 기존의 방식에 비하여 더 나은 성능을 보임을 알 수 있었다. 실험결과 부하예측 분배방식은 각 서버의 부하를 더욱 균일하게 유지하며 이 결과로 정적분배방식에 비하여 10%, 단순분배 방식에 비하여 6% 정도의 처리능력이 향상됨을 확인 하였다.

참 고 문 헌

[1] W. Zhang, S. jin, Q. Wu, "Creating Linux Virtual Servers", http://www.linuxvirtualserver.org/linuxexpo.html.

[2] Ludmila Cherkasova, "FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service", In proceeding of the Fifth International Symposium on Computers and Communications, July, 2000.

[3] YunHee Shin, "Agent-based Sophisticated Scheduling Mechanism in LVS", The Third International Network conference, INC2002, July 16-18, 2000.

[4] 김석찬, 이영, "동적 가중치에 기반을 둔 LVS 클러스터 시스템의 부하분산에 관한 연구", 정보처리학회눈문지 A 제8-A권 제4호, 2001.

[5] V. Cardellini, M. Colajanni and P. S. Yu, "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, May-June, 1999.

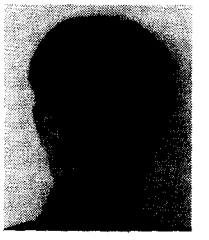
[6] T. Brisco, "DNS Support for Load Balancing", RFC 1794, Rutgers University, April, 1995.

[7] Y. M. Teo, R. Ayani, "Comparison of Load Balancing Strategies on Cluster-based Web Servers", Transactions of the Society for modeling and simulation, 2001.

[8] The workload for the SPECweb96 benchmark, http://www.specbench.org/osg/web96/workload.html.

[9] Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, "A Unified Framework for Clustering Heterogeneous Web Objects", http:// citeseer.nj.nec.com/zeng02unified.html, 2002.

[10] Rajkumar Buyya, *High Performance Cluster Computing: Architecture and Systems*, Prentice Hall. 1999.



서 경 룡

1983년 2월 부산대학교 전기기계공학과(공학사)
1990년 2월 한국과학기술원(KAIST) 전자공학과(공학석사)
1995년 8월 한국과학기술원(KAIST) 전자공학과(공학박사)
1991년~현재 부경대학교 전자컴퓨터 정보통신공학부 부교수
관심분야 : 분산시스템, 컴퓨터 네트워크