분산 웹 서버 시스템에서의 DNS 기반 동적 부하분산 기법

(DNS-based Dynamic Load Balancing Method on a Distributed Web-server System)

문 종 배 † 김 명 호 ††
(Jong Bae Moon) (Myung Ho Kim)

요 약 대부분의 분산 웹 시스템은 Domain Name System(DNS)을 이용하여 사용자 요청을 분산한다. DNS 기반 부하분산 시스템은 구성하기 쉬운 장점이 있지만, 주소 캐싱 매커니즘에 의해 서버들 사이의 부하 불균형이 발생한다. 또한, 서버의 상태를 파악하기 위해서 DNS의 수정이 필요하다. 본 논문에서는 DNS의 동적 갱신(dynamic update)과 라운드로빈 방법을 이용한 새로운 부하분산 기법을 제안한다. 본 논문에서 제안하는 방법은 DNS의 수정 없이 동적인 부하분산을 한다. 본 논문에서 제안하는 시스템은 서버의 부하량에 따라 서버를 DNS 리스트에 동적으로 추가, 삭제한다. 부하가 많은 서버를 DNS 리스트에서 제거함으로써 사용자 응답시간이 빠르다. 동적인 부하분산을 위하여 CPU와 메모리, 네트워크 자원의 사용률에 따른 부하분산 알고리즘을 제안한다. GUI 기반의 관리도구를 이용하여 손쉽게 제안하는 시스템을 관리할 수 있다. 실험을 통하여 본 논문에서 구현한 모듈들이 제안된 시스템의 성능에 많은 영향을 주지 않는다는 것을 보여준다. 또한 기존 라운드로빈 DNS와의 비교실험을 통하여 사용자 응답시간과 파일 전송률이 더 빠르다는 것을 보여준다.

키워드 : 도메인 네임서버, 분산 웹 시스템, 부하분산, 동적 갱신

Abstract In most existing distributed Web systems, incoming requests are distributed to servers via Domain Name System (DNS). Although such systems are simple to implement, the address caching mechanism easily results in load unbalancing among servers. Moreover, modification of the DNS is necessary to load considering the server's state. In this paper, we propose a new dynamic load balancing method using dynamic DNS update and round-robin mechanism. The proposed method performs effective load balancing without modification of the DNS. In this method, a server can dynamically be added to or removed from the DNS list according to the server's load. By removing the overloaded server from the DNS list, the response time becomes faster. For dynamic scheduling, we propose a scheduling algorithm that considers the CPU, memory, and network usage. We can select a scheduling policy based on resources usage. The proposed system can easily be managed by a GUI-based management tool. Experiments show that modules implemented in this paper have low impact on the proposed system. Furthermore, experiments show that both the response time and the file transfer rate of the proposed system are faster than those of a pure Round-Robin DNS.

Key words : DNS, distributed web system, load balancing, dynamic update

1. 서 론

인터넷의 발달로 인터넷 방송이나 온라인 교육, 온라인 게임, VOD, 스트리밍 서비스 등 많은 부하를 필요로 하는 서비스들이 생겨나고 있다. 또한 인터넷 컨텐츠의 유료화가 가속화 되면서 고품질 서비스를 제공하기 위한 방안을 필요로 하고 있다. 최근에는 두 대 이상의 서버를 지역적으로 분산하여 구성한 Content Distribution Networks(CDN)[1]과 같은 분산 웹 시스템[2-4]을 이용하여 웹 사이트의 컨텐츠와 부하를 분산하고 있다. 분산 웹 시스템은 확장성이 뛰어날 뿐만 아니라 하

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
† 학생회원 : 숭실대학교 컴퓨터학과
comdoct@ss.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
kmh@ssu.ac.kr
논문접수 : 2005년 1월 24일
심사완료 : 2005년 11월 27일

나의 서비스 이름으로 투명성을 제공한다. 그러나 분산 웹 시스템에서 가장 중요한 문제는 각 사용자 요청을 빠르게 처리할 수 있는 최적의 서버에 연결시켜 빠르고 안정된 서비스를 제공할 수 있도록 하는 것이다.

분산 웹 시스템은 클러스터 시스템과는 달리 전면서버(front-end)가 없기 때문에 DNS에서 사용자 요청을 서버로 할당한다[5]. DNS는 기본적으로 Fully-Qualified Domain Names(FQDN)을 IP 주소로 바꿔주는 역할을 수행한다. DNS는 FQDN을 IP 주소로 바꾸는데 투명성을 제공하기 때문에 DNS를 이용하면 클라이언트 프로그램이나 서버 프로토콜, 웹 응용프로그램을 수정하지 않고 사용자 요청을 적절한 서버에 할당할 수 있다. DNS 기반의 서버선택 방법은 기존 프로토콜을 바꾸지 않고 손쉽게 구성이 가능하고, 인터넷 환경에서 보편적으로 사용하는 IP 기반의 응용프로그램을 사용하는 모든 곳에서 사용이 가능하다는 장점이 있다. 그러나 DNS 기반 부하분산 방법은 다음과 같은 두 가지 문제점이 있다. 첫째, DNS는 서버의 상태를 모르기 때문에 부하가 많은 서버나 서비스를 할 수 없는 서버에 사용자 요청을 할당할 수 있다. 둘째, DNS의 캐싱은 DNS가 모든 사용자 요청을 제어하지 못하게 하므로 부하의 불균형을 가지고 온다.

이러한 DNS의 문제점을 해결하기 위해 DNS를 수정하는 방법[6,7]들과 DNS를 수정하지 않는 방법[2,8,9]들이 연구되었다. DNS를 수정하는 방법은 기존 DNS를 바꾸고 재구성해야하기 때문에 많은 시간과 비용이 소요된다. 또한 DNS 교체에 의해 서비스가 중단될 수 있다. DNS를 수정하지 않는 방법은 기존 DNS 스케줄링 알고리즘과 다른 스케줄링 기법을 혼합하는 방법이다. 이 방법은 2단계의 네트워크 연결이 필요하기 때문에 사용자 응답시간이 느리다는 단점이 있다.

따라서 본 논문에서는 기존 DNS를 이용한 부하분산 방식의 단점을 해결하기 위해 DNS를 수정하지 않고 사용자 응답시간이 빠른 새로운 부하분산 시스템을 제안한다. 제안하는 시스템은 사용자 요청을 균형적으로 분배하기 위해 라운드로빈 DNS를 기반으로 하며, DNS를 수정하지 않기 위해 BIND에서 제공하는 동적 DNS 갱신(dynamic DNS update)[10,11] 방식을 이용한 부하분산 모듈을 추가한다. 서버들의 상태를 파악하기 위해 모니터링 모듈을 추가한다. 부하분산 모듈은 DNS에 등록된 서버들을 상태에 따라 zone 데이타베이스에 동적으로 추가 또는 삭제한다. 부하가 많은 서버들을 DNS 리스트에서 삭제함으로써 시스템의 응답시간이 느려지는 것을 막을 수 있다. 동적 부하분산을 위하여 시스템이 제공하는 컨텐츠의 종류에 따라 CPU와 메모리, 네트워크의 자원 사용량에 가중치를 다르게 적용하여 서버들의 부하를 측정하는 알고리즘을 제안한다. 또한 본 논문에서 제안하는 시스템은 GUI 기반의 관리 도구를 사용하여 DNS를 손쉽게 관리할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 DNS를 이용한 부하분산 방법에 대한 기존 연구를 살펴본다. 제3장에서는 DNS를 이용한 부하분산 시스템의 요구사항을 살펴본다. 제4장에서는 본 논문에서 제안하는 DNS 기반 부하분산 시스템과 모듈들을 설명한다. 제5장에서는 여러 실험을 통하여 본 논문에서 제안하는 시스템의 효율성을 살펴본다. 끝으로 제6장에서는 결론을 맺는다.

2. 관련연구

NCSA에서 처음으로 DNS를 수정하여 라운드로빈 방식의 부하분산 방법을 사용하였다[4]. 라운드로빈 방식은 DNS에 여러 IP를 하나의 이름으로 연결하여 순차적으로 한 번씩 요청에 대한 응답을 할당하는 방식이다. 라운드로빈 방법은 설치가 간단하고 부하를 분산하는 디스패처가 없기 때문에 한 서버가 장애로 서비스를 할 수 없어도 전체 시스템에 영향을 미치지 않는다. 즉, SPOF(single point of failure)가 발생하지 않는 장점이 있다. 그러나 DNS의 특성상 캐싱으로 인한 부하의 불균형이 발생한다[3]. 또한 DNS는 서버들의 상태를 알지 못하기 때문에 장애가 발생한 서버에도 사용자 요청을 할당할 수 있는 단점이 있다. 라운드로빈 DNS는 부하분산 방법의 단점을 해결하기 위하여 아래와 같은 많은 연구들이 진행되어 왔다.

I2-DSI[12]와 Cisco 시스템의 DistributedDirector[8]는 근접거리 기반의 알고리즘을 사용한다. 근접거리 기반 방식은 DNS가 클라이언트의 IP 주소를 알 수 없기 때문에 클라이언트의 로컬 네임서버와의 거리를 측정하여 가까운 서버에 연결하는 방식이다. 실질적으로 서비스는 클라이언트와 웹 서버사이에서 이루어진다. 근접거리 기반 방식에서는 클라이언트와 로컬 네임서버의 거리가 가깝다는 가정이 있어야 한다. 따라서 DNS와 클라이언트의 로컬 네임서버와의 거리가 가깝다고 하여도 로컬 네임서버와 클라이언트의 거리가 멀다면 성능이 떨어진다[13,14].

TTL(Time-To-Live)을 이용한 방법은 크게 두 가지로 나눌 수 있다[3]. 고정 TTL(constant TTL) 알고리즘은 TTL 시간동안 같은 IP 주소를 반환하는 방법이다. 고정 TTL 알고리즘에는 RR2(two-tier round-robin)과 DAL(dynamically accumulated load), MRL(minimum residual load) 등이 있다[9]. 고정 TTL 알고리즘은 모든 IP 주소 요청에 대해 같은 TTL 값을 반환하기 때문에 TTL 시간동안 많은 사용자 요청이 한 서버로 집중되면 부하의 불균형이 발생한다. 변동 TTL(adaptive

TTL) 알고리즘은 각각의 주소 요청마다 다른 TTL 값을 할당하는 방법이다[15,16]. 사용자 접속이 많은 사이트에는 작은 TTL 값을 할당하고, 사용자 접속이 적은 사이트에는 큰 TTL 값을 할당한다. 사용자 요청이 동적으로 변하는 환경에서는 큰 TTL 값을 할당했을 때 많은 사용자 요청이 할당되어 부하가 급증할 수 있다. 또한 사용자 요청의 종류에 따라 서버의 부하가 동적으로 변할 수 있기 때문에 사용자 접속률에 따라 TTL 값을 할당하는 것은 적합하지 않다. TTL을 이용한 방법의 가장 큰 단점은 DNS를 수정해야 한다는 것이다.

SWEB[17]과 DPR[7,18], Sun-SCALR[19] 등과 같은 시스템은 DNS기반 알고리즘과 HTTP 재지향 방법 또는 패킷변환 방법을 복합한 방법을 사용한다. 이 방법에서는 우선 DNS에서 라운드로빈과 같은 방식으로 서버의 IP 주소를 할당한다. 그 다음에 해당 서버에서 다시 HTTP 재지향 방법이나 패킷변환 방법으로 부하를 분산하는 방법이다. 분산된 서버들이 스케줄링을 하기 때문에 병목현상이 없다. 그러나 HTTP 재지향 방법은 재지향에 따른 네트워크 지연이 발생하여 사용자 응답시간이 길어진다. 또한 패킷변환 방법은 패킷변환에 따른 오버헤드가 발생하여 사용자 응답시간이 길어진다.

lbnamed[20]는 스탠포드 대학에서 부하분산을 목적으로 펄(perl)을 이용하여 만든 DNS이다. lbnamed는 기존 DNS 프로토콜을 수정하지 않고 만들었으며, 기존 DNS 응용프로그램인 BIND와도 호환이 가능하도록 구현되었다. lbnamed는 서버들의 부하를 동적으로 파악하여 부하가 가장 작은 쪽으로 사용자 요청을 연결하는 방법을 사용한다. 그러나 펄로 작성되어 기존 BIND보다는 빠르지 못하고, 등록된 서버들의 다양한 부하를 고려하지 않아 사용자 요청이 다양하고 부하가 동적으로 변하는 환경에서는 적당하지 못하다.

표 1은 기존 DNS 기반 부하분산 방법들의 장점과 단점을 정리한 표이다. 본 논문에서는 이러한 단점을 보완한 DNS를 수정하지 않고 서버와 네트워크의 동적인 부하를 고려하는 부하분산 시스템을 제안한다.

3. DNS 기반 부하분산 시스템의 요구사항 분석

본 장에서는 기존 연구들의 단점을 보완하기 위하여 고려해야 할 요구사항들을 살펴본다. 본 논문에서는 기존 DNS를 이용하여 동적인 부하분산을 할 수 있는 시스템 구축을 목표로 하고 있다. DNS를 이용한 부하분산 시스템을 구현하기 위해서 DNS 프로토콜을 수정한다면 기존 DNS를 모두 바꾸고 재구성해야 하기 때문에 많은 시간과 비용이 소요된다. 또한 DNS 프로그램 교체에 의해 서비스가 중단될 수 있다. 따라서 본 논문에서는 DNS 프로토콜을 수정하지 않고 서비스 중단 없이 호환성을 유지하기 위하여 부하분산 모듈을 추가하는 방법을 사용한다.

기존 DNS기반 부하분산 방법들은 DNS의 설정파일에 등록되어있는 서버들만 고려한다. 새로운 서버를 등록할 때마다 또는 특정 서버를 삭제할 때마다 설정파일을 수정하고 DNS를 재시동해야 한다. 또한 DNS는 텍스트 기반의 명령어를 이용하여 관리하도록 되어있다. 관리자는 명령어들을 이용하여 설정파일을 수정하고 DNS를 재시동하는 것은 번거롭고 확장성도 떨어뜨린다. 따라서 DNS를 이용한 부하분산 시스템에서는 DNS를 재시동하지 않고 설정파일을 동적으로 수정할 수 있도록 해야 할 필요가 있다. 또한 관리자가 DNS 유지와 관리를 손쉽게 하기 위해 GUI 기반의 관리 도구가 필요하다.

DNS는 인터넷 환경에서 보편적으로 사용되는 서비스이다. 임의의 사용자가 DNS의 설정파일을 동적으로 수정하면 시스템에 큰 영향을 미칠 수 있다. 또한 DNS의 zone 파일을 임의 사용자가 가지고 간다면 서버들의 정

표 1 기존 DNS를 이용한 부하분산 방법의 비교

		장점	단점
라운드로빈(RR)		- 구성하기 쉬움	- 서버 상태를 고려하지 않음
근접거리 (Proximity)		- 네트워크 상태 고려	- 서버 성능은 고려하지 않음 - 사용자와 서버와의 거리 확인 어려움
TTL	고정 (Constant)	- 라운드로빈 방식보다 성능이 좋음	- TTL 시간동안 부하 불균형 초래 - BIND 수정 필요
	변동 (Adaptive)	- 서버의 성능 고려한 TTL 변경	- BIND 수정 필요
DNS 혼합	HTTP 재지향 (Redirection)	- 이기종 환경에서 균형적인 부하분산	- HTTP 재지향에 따른 시간 지연
	패킷 변환 (Rewriting)	- 이기종 환경에서 균형적인 부하분산	- 패킷 변환에 따른 오버헤드 발생
lbnamed		- DNS 프로토콜 수정하지 않음 - 기존 DNS와의 호환성	- 부하정보가 부족 - BIND보다 느림

보가 노출되기 때문에 악의적인 의도로 사용될 수 있다. 따라서 DNS를 이용한 부하분산 시스템에서는 관리자만 정보를 접근할 수 있도록 철저한 보안이 필요하다. 또한 관리자에 대한 인증도 반드시 필요하다.

DNS에서는 기본적으로 스케줄링 방식으로 라운드로빈 방식을 지원한다. 라운드로빈 방식은 사용자 요청을 서버들 사이에 균등하게 분산하는 방법이다. 그러나 서버들의 상태를 고려하지 않기 때문에 서버들의 불균형을 가져온다. 또한 장애가 발생한 서버에도 계속 사용자 요청을 할당할 수 있다. 따라서 DNS 기반의 부하분산 시스템에서는 고른 부하분산을 하기 위해서 DNS가 서버들의 상태를 고려해야 한다. 서버들의 상태를 고려할 때 CPU와 메모리, 네트워크의 사용량을 기반으로 부하를 측정하도록 한다. 분산 웹 시스템이 제공하는 서비스의 종류에 따라 CPU와 메모리, 네트워크 자원의 사용량이 달라지기 때문이다.

DNS 기반 부하분산 시스템에서는 캐싱에 의한 부하의 불균형을 막기 위해 TTL 값을 최소로 한다. TTL 값을 작게 함으로써 동적으로 변하는 서버의 상태에 빠른 대응을 할 수 있다. 반면에 TTL 값을 작게 함으로써 DNS의 처리가 많아져 부하가 증가할 수 있다. 하지만 인터넷 방송이나 VOD 서비스 같은 자원 사용이 많고 서비스 받는 시간이 긴 서비스에서는 크게 고려하지 않아도 된다고 가정한다.

4. 제안하는 DNS 기반의 동적 부하분산 시스템

본 장에서는 기존 DNS 기반 부하분산 시스템의 단점을 보완한 새로운 부하분산 시스템을 제안한다. 본 논문에서 제안하는 DNS 기반의 부하분산 시스템은 라운드로빈 방식을 기반으로 하며, 부하가 많은 서버들을 감시하여 사용자 요청을 받지 못하도록 한다. 제안하는 시스템은 DNS 소프트웨어인 BIND를 그대로 사용하면서 부하분산을 할 수 있도록 모듈을 추가한다. 본 논문에서는 BIND를 재시동 하지 않고 동적으로 DNS 리스트를 갱신할 수 있도록 BIND의 동적 갱신 프로토콜을 사용한다. 동적 갱신 프로토콜을 이용하여 동적으로 서버들의 추가 또는 삭제가 가능하기 때문에 손쉬운 확장성을 보장할 수 있다. 부하가 많은 서버에 사용자 요청을 할당하지 않음으로써 사용자 응답시간이 느려지는 것을 방지할 수 있다. 또한 제안하는 시스템의 가장 큰 장점은 DNS를 수정하지 않는다는 것이다.

본 논문에서 제안하는 DNS 기반 부하분산 시스템의 구조는 그림 1에서 보는 것처럼 BIND와 동적 갱신 모듈, 모니터링 도구, 관리 도구로 구성되어 있다. DNS 서버는 라운드로빈 방식으로 서비스하도록 설정한다. 라운드로빈 방식은 사용자 요청을 서버들 사이에 가장 균

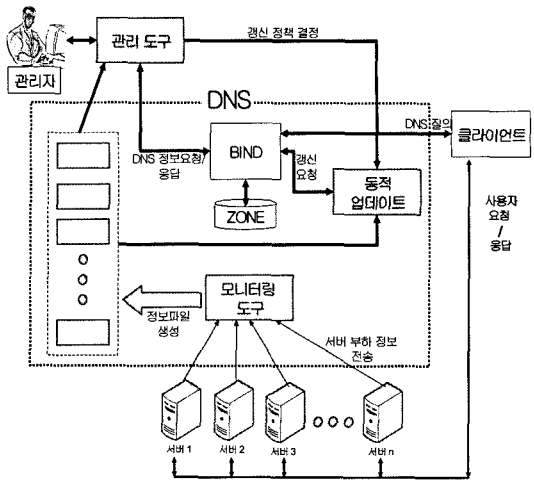

그림 1 DNS를 이용한 부하분산 시스템의 전체 구조

형적으로 분산하는 방법이다. 그러나 이 방법은 서버들의 상태를 모르기 때문에 문제가 발생한다. 이러한 문제를 해결하기 위해 모니터링 모듈을 추가한다. 모니터링 모듈은 서버들의 상태를 주기적으로 관찰하고 부하를 수집한다. 동적 갱신 모듈은 모니터링 모듈이 수집한 서버들의 부하 정보를 기반으로 라운드로빈 DNS가 사용자 요청을 빠르게 처리할 수 있는 서버들만을 유지하도록 한다. 관리 도구는 네트워크를 통해 원격지에서도 DNS의 손쉬운 관리를 할 수 있도록 한다.

4.1 동적 갱신 모듈의 설계 및 구현

동적 갱신 모듈은 제안하는 시스템이 DNS를 수정하지 않고 부하분산을 할 수 있도록 하는 가장 중요한 모듈이다. 이 모듈은 DNS에 등록된 서버들의 총부하량을 주기적으로 측정하여 DNS 리스트를 최적의 서버들로 유지하도록 한다. 최적의 서버들이란 사용자 요청을 빠르게 처리하여 응답할 수 있는 서버들을 말한다. 서버들의 총부하량은 모니터링 데몬이 수집한 3가지 중요 자원인 CPU와 메모리, 네트워크의 사용량을 기반으로 하여 계산한다. 이 때 모니터링 도구에 의해 생성된 서버들의 부하정보 파일을 참조한다.

그림 2는 동적 갱신 모듈의 수행과정을 보여주는 그림이다. BIND는 zone 파일의 수정 없이 DNS 리스트를 갱신할 수 있도록 동적 갱신 프로토콜을 제공한다. 동적 갱신 모듈은 DNS 리스트에서 추가 또는 삭제될 정보를 위해 갱신 요청 메시지를 구성하여 BIND에 요청한다. BIND는 요청한 메시지가 올바른지 확인한 후 요청을 수행한다. 서버들의 동적인 부하변화가 많은 환경에서는 zone 레코드의 수정이 많이 일어날 수 있기 때문에 최대, 최소 임계값을 정하여 동적 갱신의 발생 횟수를 줄일 수 있다.

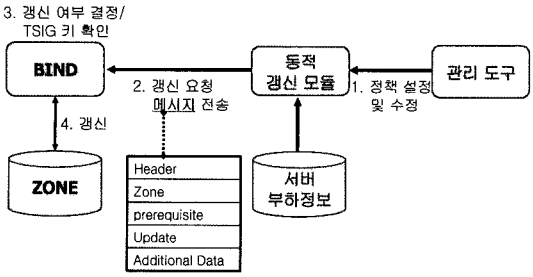

그림 2 동적 갱신 모듈

4.2 모니터링 도구의 설계 및 구현

모니터링 도구는 DNS에 등록된 서버들의 부하 상태 정보를 주기적으로 수집하는 역할을 한다. 그림 3에서 보는 것과 같이 모니터링 도구는 모니터링 데몬과 모니터링 에이전트로 구성된다. 서버의 증가에 따라 부하 상태 정보를 주고받는 네트워크 부하가 증가할 것을 고려하여 UDP 패킷을 이용하여 통신하도록 한다. 또한 서버의 상태를 파악하기 위한 하트비트 패킷의 역할도 하도록 하여 추가적인 네트워크 부하를 줄이도록 한다.

모니터링 데몬은 DNS가 설치된 서버에서 실행되며, DNS에 등록된 각 서버들의 부하정보를 수집한다. 동적 갱신 모듈에서 부하량이 많은 서버는 서버 리스트에서 삭제하지만 모니터링 데몬은 그 서버를 계속 모니터링 할 수 있도록 서버 정보를 계속 유지한다. 모니터링 데몬은 에이전트로부터 각 서버들의 부하정보를 받아들이고, 각 서버의 IP주소에 해당하는 파일을 생성하여 CPU와 메모리, 네트워크의 부하정보를 저장한다. 모니터링 데몬은 에이전트에서 보내는 부하정보 패킷을 이용하여 서버가 동작하는지 판단한다. 에이전트가 보내는 UDP 패킷이 3번의 주기 동안 도착하지 않았을 경우에는 생성하였던 정보 파일을 삭제하도록 한다. 삭제된 서버는 DNS 정보에서 자동으로 삭제되도록 한다.

모니터링 에이전트는 각 서버에서 실행되며, 서버들의 부하정보와 자신의 IP 주소를 포함한 UDP 패킷을 모니터링 데몬에 주기적으로 보낸다. 에이전트는 부하정보를 UDP 패킷으로 생성하여 보내기 때문에 실시간 부하정보를 빠르게 보낼 수 있으며 네트워크 부하를 줄일 수 있다.

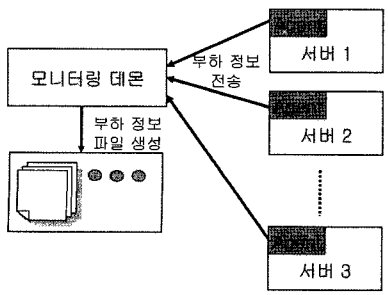

그림 3 모니터링 도구의 구성

4.3 DNS 관리 도구의 설계 및 구현

DNS는 리눅스 환경에서 수행되며, 여러 텍스트 기반의 명령어들을 이용하여 관리하도록 구성되어 있다. 이런 불편함을 덜기 위하여 관리자가 GUI를 통해 DNS를 손쉽게 관리할 수 있도록 관리 도구를 제공할 필요가 있다. 원격지에서도 관리할 수 있도록 네트워크 기반으로 구현한다. 관리 도구의 구조는 그림 4와 같다.

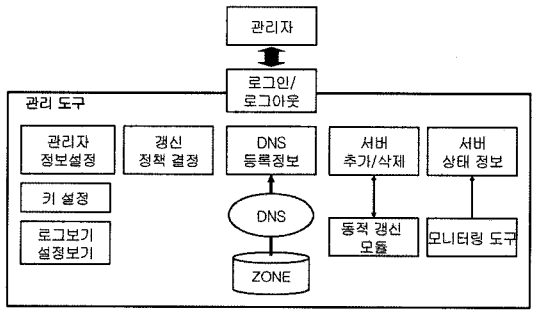

그림 4 관리 도구의 구조

관리 도구에서는 로그인 / 로그아웃 기능을 비롯하여 DNS 등록정보 보여주기 기능, 서버 상태 보여주기 기능, 로그 보기 기능, 관리자 정보 수정 기능, 갱신 정책 설정 기능, 서버 추가 / 삭제 요청 기능, 설정 정보 기능, 키 설정 기능을 관리자가 쉽게 사용할 수 있도록 구성되어 있다.

로그인 / 로그아웃 기능은 DNS 관리를 하기 위한 인증을 받는다. DNS 등록정보 보여주기 기능은 현재 DNS에 등록되어 있는 서버들의 리스트를 보여주는 기능이다. DNS 정보는 BIND에서 제공하는 'dig' 명령어로 출력되는 부분을 편집하여 화면에 보여준다. 서버 상태 보여주기 기능은 현재 리스트에 보이는 서버들의 부하정보를 실시간으로 보여준다. 서버들의 상태에 관한 정보는 모니터링 도구에서 생성한 서버 정보 파일을 읽어 들여 화면에 보여준다. 로그 보기 기능은 DNS의 추가 / 삭제에 대한 로그를 볼 수 있다. 동적 갱신 모듈에서 생성되는 로그 파일을 열어 그 내용을 보여준다. 이러한 로그 내용을 바탕으로 갱신 정책을 정하는데 참조한다. 갱신 정책 설정 기능은 DNS의 스케줄링 정책을 설정할 수 있다. 서버 추가 / 삭제 요청 기능은 새로운 서버의 추가 또는 수정에 관한 폼을 제공하며 동적 갱신 모듈에 실시간 갱신을 요청한다. 관리자 정보 설정 기능은 관리자의 아이디와 비밀번호를 변경할 수 있다. 설정 보기 기능은 현재 DNS의 모든 설정 정보를 보여

줌으로써 변경된 설정들이 적용되었는지 확인할 수 있다. 키 설정 기능은 DNS 보안을 위한 TSIG 키 값을 설정한다.

4.1 동적 부하분산 알고리즘

본 논문에서는 동적인 부하분산을 위하여 그림 5와 같은 최적의 서버 유지 알고리즘을 수행한다. DNS에 등록된 서버는 n 대가 있다고 가정하며, 사용된 기호의 의미는 다음과 같다.

$T_{i,Load}$ = i 번째 서버의 총부하량

α, β, γ : 가중치 ($\alpha+\beta+\gamma=100$), TH : 임계값

S_i = i 번째 서버

$L_{i,CPU}$: i 번째 서버의 CPU 부하량,

$L_{i,MEM}$: i 번째 서버의 메모리 부하량,

$L_{i,NET}$: i 번째 서버의 네트워크 부하량

우선 i 번째 서버 S_i 의 총부하량을 구한다. 각 서버의 부하를 측정할 때에는 CPU와 메모리, 네트워크의 사용률을 이용하여 측정한다. 각 서버의 자원 사용량은 모니터링 모듈이 수집한 정보를 사용한다. 분산 시스템이 제공하는 서비스의 내용에 따라 각 자원의 사용량에 다른 가중치를 두어 총부하량을 계산해야 한다. 온라인 방송이나 VOD 서비스와 같이 대용량의 파일 전송을 하는 서비스일 경우 CPU와 메모리 사용률 증가보다 네트워크 사용률의 증가폭이 크며, 온라인 게임과 같이 서버에서의 많은 처리를 요구하는 서비스일 경우는 네트워크 사용률의 증가보다 CPU 사용률의 증가폭이 크다. 따라서 총부하량을 계산할 때에는 서비스의 종류에 따라 사용률의 증가폭이 큰 자원에 가중치를 높여서 계산한다.

S_i가 DNS 리스트에 존재하고 총부하량이 임계값보다 크면 S_i를 리스트에서 삭제하고, S_i가 DNS 리스트에 존재하지 않고 총부하량이 임계값보다 작으면 부하가 다시 줄어들었다고 판단하여 다시 리스트에 추가한다. 그 외의 경우에는 서버의 현재 상태로 계속 유지하도록 한다. 임계값은 학습에 의한 값으로 관리자가 직접 정할 수 있도록 한다.

모니터링 도구에 의해 측정된 부하 정보를 기반으로 본 논문에서는 다음과 같은 갱신 정책들을 동적으로 선택하고 변경할 수 있다.

- CPU 기반 방식: $\alpha = 100$, $\beta = 0$, $\gamma = 0$, CPU 사용량이 대부분인 서비스의 경우 사용
- 메모리 기반 방식: $\alpha = 0$, $\beta = 100$, $\gamma = 0$, 메모리 사용량이 대부분인 서비스의 경우 사용
- 네트워크 기반 방식: $\alpha = 0$, $\beta = 0$, $\gamma = 100$, 네트워크 사용량이 대부분인 서비스의 경우 사용
- 복합 방식: $\alpha + \beta + \gamma = 100$, CPU와 메모리, 네트워크의 사용량이 모두 무시하지 못할 정도로 사용되는 서비스의 경우, 관리자에 의해 가중치가 설정

```
while (1) {
  for(i = 1 to n)
  {
    T_{i, Load} = α × L_{i, CPU} + β × L_{i, MEM} + γ × L_{i, NET}
    if ((S_i is in DNS List) and (T_{i, Load} > TH))
      then remove S_i from DNS list
    else if ((S_i is not in DNS list) and (T_{i, Load} < TH))
      then add S_i to DNS list
  }
}
```

그림 5 최적의 서버를 유지하기 위한 알고리즘

4.5 DNS 보안

동적 갱신 모듈은 DNS의 zone 파일을 직접 수정하지 않고 레코드를 원격에서 갱신할 수 있기 때문에 관리자만 DNS에 접근이 가능하도록 해야 한다. DNS 보안을 위해 동적 갱신 메시지는 DNS 메시지의 추가 정보(additional data)부분에 RFC2845에 정의된 TSIG 레코드를 추가하여 보낸다. TSIG 레코드는 메시지의 송신자가 수신자와 공유된 암호화된 키를 가지고 있으며 메시지가 송신자로부터 보내진 후에 변경되지 않았음을 보증한다.

DNS는 갱신 요청 메시지를 받으면 TSIG 키 값을 확인하여 DNS에 대한 접근 권한이 있는지를 확인한다. 접근 권한이 있으면 "prerequisite" 레코드 부분을 확인하여 요청한 메시지의 요구가 현재 DNS에서 가능한 요청인지 판단한다. 예를 들면, DNS에 "somewhere.com"이라는 도메인이 존재하지 않는데 "www.somewhere.com"이라는 레코드를 추가하라는 메시지가 오면 갱신을 하지 않는다.

DNS는 zone 파일 전송을 지원한다. DNS zone 파일 전송은 임의의 호스트에서 DNS의 zone 파일의 내용을 전송하거나 받을 수 있는 기능이다. DNS zone 파일은 회사나 조직의 네트워크 구조와 같은 정보를 포함하고 있을 수 있다. 따라서 zone 파일 전송을 허용한다면 DNS 스푸핑(spoofing)과 같은 악의적인 용도로 사용될 수 있기 때문에 보안에 문제가 된다. TSIG 키를 이용하여 zone 레코드 정보가 다른 서버로 유출되어 악의적인 의도로 사용되는 것을 막을 수 있다.

5. 실험 및 분석

본 장에서는 제안한 시스템의 성능을 측정하기 위하여 다양한 실험을 한다. 첫째, 본 논문에서 구현한 시스

템의 기능들을 수행하고 제대로 작동되는지 실험한다. 둘째, 본 논문에서 구현한 DNS 기반 부하분산 시스템의 효율성을 살펴보기 위하여 DNS의 네트워크와 CPU의 부하량 변화를 측정하였다. 셋째, 작은 TTL 값이 시스템에 미치는 영향을 측정한다. 마지막으로 기존 라운드로빈 DNS와의 성능을 비교 분석한다.

실험에 사용된 DNS 서버는 실제 네임서버 ns.grid.or.kr을 서비스하고 있는 시스템을 사용하였다. DNS 서버는 CPU가 P-III 800MHz이며 메모리 512MB, 100Mbps 이더넷 카드를 가지고 있다. DNS 응용프로그램을 위해 BIND 9를 설치하고 TSIG 키를 설정하였다. 실험에 사용한 도메인은 초기에 4대의 서버를 라운드로빈 방식으로 설정한 vdns.grid.or.kr을 사용하였다. 각 서버는 CPU가 P-IV 3GHz, 메모리 1GB, 100Mbps 이더넷 카드를 가지고 있으며, 운영체제는 리눅스를 설치하였다. 각 서버들은 100Mbps의 LAN으로 연결되어 있다. 웹 서버를 위해서 Apache 2.0을 설치하였다.

5.1 부하분산 DNS 시스템 기능 테스트

본 논문에서 구현한 DNS 기반 부하분산 시스템은 관리 도구의 GUI를 통해 DNS를 재시동 하지 않고 DNS 리스트를 손쉽게 관리할 수 있다. 그림 5는 관리 도구를 이용하여 DNS의 서버 리스트를 동적으로 추가하는 그림이다. 도메인 추가메뉴에서는 도메인 이름과 TTL, IP 주소를 입력한다. 그림 6은 도메인 삭제하는 그림이다. 삭제된 서버는 서버 리스트에서 삭제되지만 모니터링 부분에서는 계속 모니터링이 되는 것을 볼 수 있다. 삭제된 서버를 계속 모니터링하여 부하량이 줄어들면 서버 리스트에 다시 추가할 수 있다. 추가와 삭제 후에 nslookup 명령어로 변경된 리스트를 바로 확인할 수 있다. 그림 7은 갱신 정책을 설정하는 그림이다. 관리자가 서비스의 종류에 따라 CPU 기반과 메모리 기반, 네트워크 기반, 복합 방식을 선택할 수 있다. 그림 8은 DNS 기반 부하분산 시스템의 로그 정보를 보여주는 그림이다. 서버들이 동적으로 삽입 또는 삭제된 로그를 그래프나 텍스트로 볼 수 있다. 로그 정보를 통해 어떤 서버에 부하가 많이 발생했는지 통계적으로 파악할 수 있다.

그림 9와 그림 10은 관리 도구에서 갱신 정책을 설정하여 본 논문에서 구현한 시스템이 제대로 작동하는지 보여준다. 관리 도구에서 CPU 사용률의 최대 임계값을 80%, 최소 임계값을 70%로 설정한다. 임계값은 학습에 의해 관리자가 정해줄 수 있다. 특정 서버(213.253.21.75)에 CPU를 많이 사용하는 프로그램을 수행하여 CPU

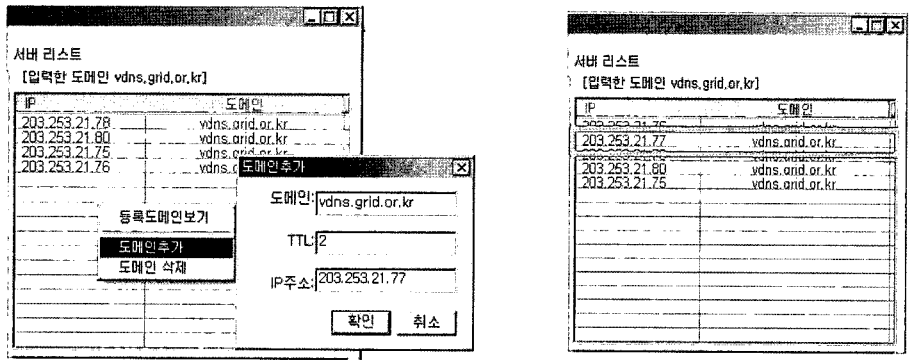

그림 5 도메인 추가를 위한 GUI

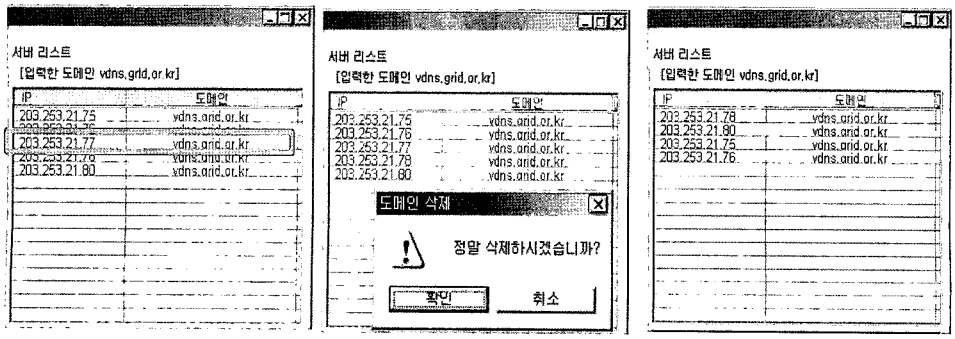

그림 6 서버의 동적인 삭제

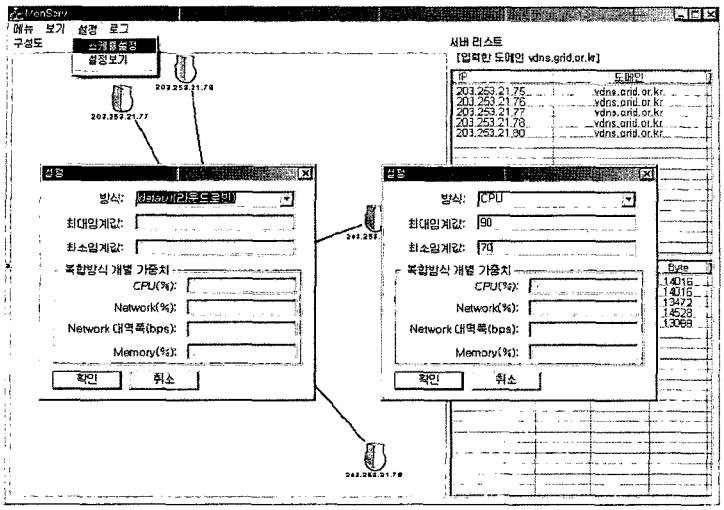

그림 7 갱신 정책 설정 화면

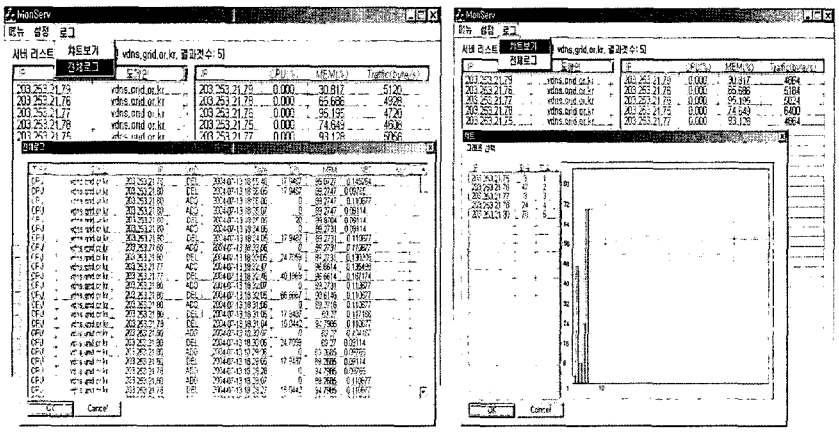

그림 8 DNS 로그 보기

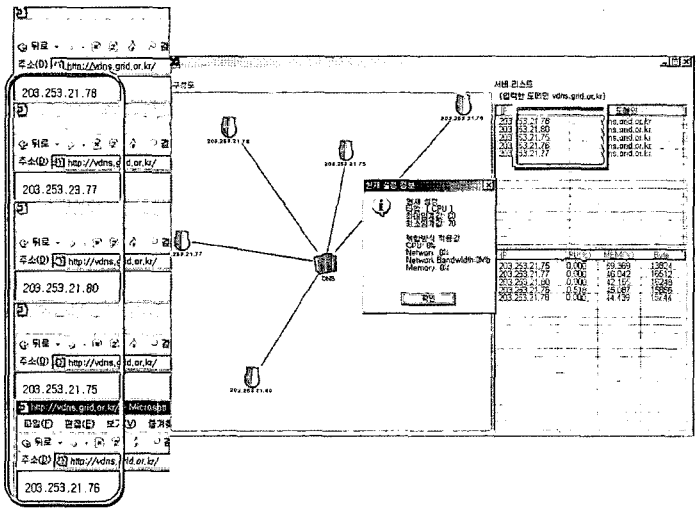

그림 9 서버들에 부하가 많지 않을 때의 화면

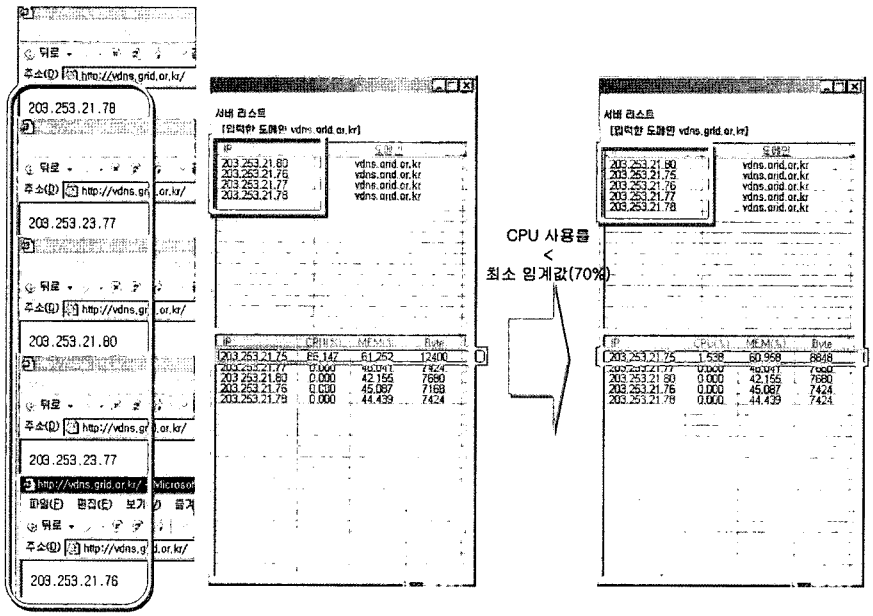

그림 10 특정 서버에 부하가 많이 발생할 때의 화면

사용률이 80% 초과하도록 하면 그 서버는 라운드로빈 리스트에서 삭제되며, 해당 도메인(vdns.grid.or.kr)으로 접속되지 않는 것을 볼 수 있다. CPU 사용률이 최소 임계값(70%)보다 작아졌을 경우 삭제된 서버는 서버 리스트에 추가되며 다시 접속이 이루어진다. 부하가 많은 서버를 라운드로빈 DNS 리스트에서 삭제함으로써 시스템의 전체적인 응답시간이 느려지는 것을 막을 수 있다.

5.2 부하분산 모듈 추가에 따른 DNS 오버헤드 측정

본 논문에서 제안한 부하분산 시스템의 효율성을 측정하기 위하여 DNS의 부하를 측정하는 실험을 하였다. DNS에 등록된 서버들을 증가시키면서 DNS의 CPU와 네트워크의 사용량을 측정하였다. 각 서버의 모니터링 에이전트가 DNS의 모니터링 데몬으로 부하정보를 주기적으로 보내기 때문에 부하정보 전송 주기에 따른 부하량도 측정하였다. 그림 11은 서버의 수가 20대에서 200대까지 증가할 때 부하정보 전송 주기에 따른 네트워크의 부하량을 보여준다. 서버의 수가 증가하면서 네트워크 사용량이 증가하는 것을 볼 수 있다. 서버가 200대까지 증가하면서 부하정보를 주고받는 부하가 전체 네트워크의 0.1% 정도밖에 되지 않는다. 서버가 더 증가할수록 사용량도 증가하겠지만 UDP 패킷으로 구성된 부하정보가 전체 시스템의 성능에는 크게 영향을 미치지 않는다. 부하정보 전송 주기가 길어질수록 네트워크 사용량이 줄어드는 것을 볼 수 있다. 서버들의 부하정보를 1초 단위로 측정하면 더 정확한 부하정보를 가질 수 있지만 네트워크의 부하가 2초 단위로 측정할 때보다 많

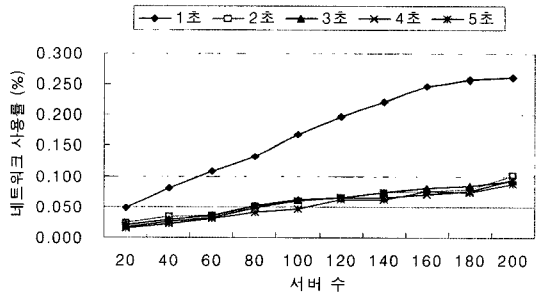

그림 11 서버의 증가에 따른 부하정보 전송 주기별 DNS의 네트워크 부하량

은 네트워크 부하를 발생하는 것을 볼 수 있다. 부하정보 전송 주기가 길어지면 네트워크의 부하는 줄어들지만 서버들의 정확한 부하정보를 얻을 수 없다. 따라서 부하정보 주기를 2초로 하면 시스템에 영향을 미치지 않으면서 정확한 부하정보를 얻을 수 있다.

그림 12는 서버의 수가 20대에서 200대까지 증가할 때 DNS에 등록된 서버 리스트와 부하정보를 유지하는 과정에서 발생하는 CPU의 부하량을 보여준다. 서버의 수가 증가하면서 CPU 사용률이 증가하는 것을 볼 수 있다. 네트워크 사용률과 마찬가지로 부하정보 전송 주기가 길어질수록 CPU 부하는 작아지고 전송 주기가 짧아질수록 CPU 사용률이 증가하는 것을 볼 수 있다. 서버가 200대일 경우 최대 1.3% 정도의 부하량을 나타내고 있다. 이는 전체 성능에 비해 영향을 미치지 않을 정

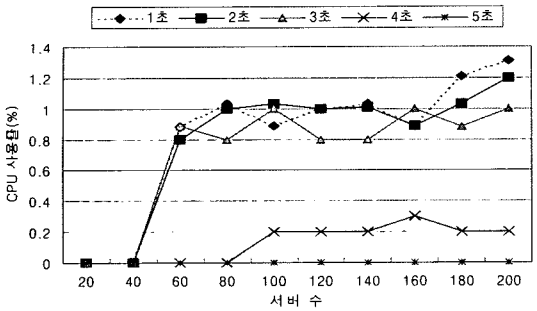

그림 12 서버 증가에 따른 부하정보 전송 주기별 DNS의 CPU 사용률

도의 작은 값이다. 모니터링 데몬이 각 서버의 부하정보를 받아 그 정보를 파일을 생성하여 유지하는데 발생하는 부하가 크면 DNS의 전체 성능을 떨어뜨린다. DNS의 동적 갱신 모듈은 부하분산 정책에 의해서 각 서버의 부하를 측정하여 리스트에서 삭제할 것인지를 결정한다. 서버 리스트를 유지하는 과정에서 부하가 많이 발생하면 시스템 전체에 영향을 미칠 수 있다. 본 실험에서 동적 갱신 모듈과 모니터링 도구가 DNS 시스템의 전체 성능에 미치는 영향이 크지 않다는 것을 알 수 있다. 또한 부하정보 전송 주기는 네트워크와 CPU의 사용률과 트레이드오프 관계가 있는 것을 볼 수 있다.

5.3 작은 TTL 값이 DNS에 미치는 영향

DNS 기반 부하분산 시스템에서 TTL 값을 크게 하면 캐싱에 의해 네트워크의 부하를 줄일 수 있지만 부하의 불균형이 발생한다. 본 논문에서는 캐싱에 의한 부하의 불균형을 막기 위하여 TTL 값을 최소로 설정한다. 작은 TTL 값이 시스템에 미치는 영향을 측정하기 위하여 TTL 값을 0에서 20초까지 증가하면서 CPU와 네트워크의 사용률을 측정하였다. 실험에 사용한 클라이언트는 4000개이며 gethostbyname() 시스템 콜을 이용하여 DNS에 쿼리를 보내어 평균을 구했다.

그림 13은 TTL 값의 변화에 따른 CPU와 네트워크의 사용률을 나타낸다. TTL 값이 증가하면서 캐싱에 의해 DNS에서 처리하는 요청의 수가 줄어들기 때문에 네트워크와 CPU 사용률이 줄어드는 것을 볼 수 있다. 캐싱을 없애기 위해 TTL 값을 0으로 설정하면 모든 사용자 요청에 대하여 제어를 할 수 있지만 CPU와 네트워크 사용률이 높다. 그러나 1초 이상일 경우에는 많이 줄어들어는 것을 볼 수 있다. 온라인 방송이나 온라인 교육과 같은 서비스들이 DNS에서 응답한 서버의 IP를 가지고 오는 서비스 초기화 시간보다 서비스 받는 시간이 길기 때문에 한 클라이언트가 작은 TTL 값 동안 많은 질의가 발생하지 않는다고 가정한다.

그림 14는 클라이언트가 증가하면서 TTL 값에 대한 사용자 응답시간의 변화를 나타낸 그림이다. TTL 값을 0으로 설정하였을 때보다 TTL 값을 크게 할수록 DNS에 부하를 줄여주기 때문에 응답시간이 빠르다는 것을 볼 수 있다. 본 논문에서 제안한 시스템은 앞에서 실험한 네트워크과 CPU의 사용률을 고려하면 TTL 값이 2초일 때 가장 효율적인 성능을 보인다는 것을 알 수 있다.

실험을 통하여 본 논문에서 구현한 DNS 기반 부하분산 시스템이 제대로 동작하는지 살펴보았다. 서버를 모니터링하고 부하를 측정하여 최적의 서버 리스트를 유지하는 동안 DNS에 부하가 많이 발생하지 않는 것을 보였다. 또한 작은 TTL 값이 시스템에 미치는 영향을 측정하여 적합한 최소의 TTL 값을 측정할 수 있다.

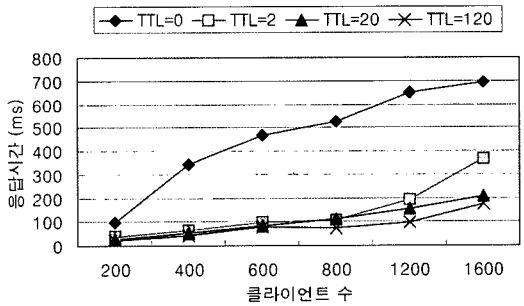

그림 14 클라이언트 증가에 따른 사용자 응답시간

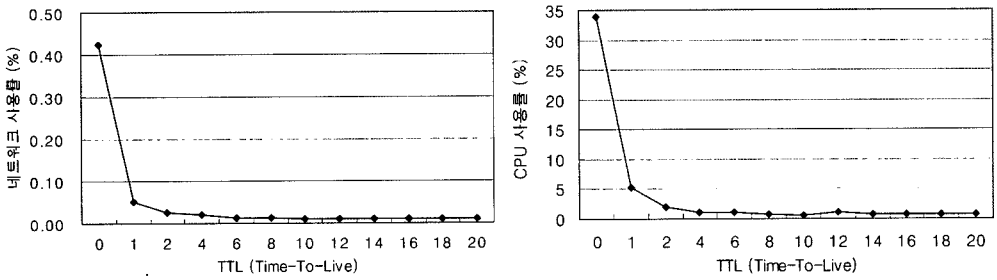

그림 13 TTL 값의 변화에 따른 CPU와 네트워크의 사용률

5.4 라운드로빈 DNS와의 성능 비교

다음은 본 논문에서 제안한 시스템인 DLBDNS와 라운드로빈 DNS를 이용한 부하분산 시스템의 평균 응답시간과 평균 파일 전송 속도를 측정하였다. 본 실험에서는 시스템의 성능을 측정하기 위하여 HTTP 부하생성 프로그램인 SIEGE[21]를 이용하였다. SIEGE는 실제 사용자의 행동을 대신하는 쓰레드를 생성하여 웹 서버에 접속하는 사용자 요청을 발생시킨다. 본 실험에서는 SIEGE를 이용하여 클라이언트가 30개에서 100개까지 증가하면서 DNS 시스템에서 등록된 웹 서버에 3MB의 미디어 파일들을 125번 요청하도록 하였다. DNS 시스템에 등록된 서버는 4대의 웹 서버들이다. 동적인 부하를 가진 환경을 구성하기 위하여 그 중 2대는 60%~90%의 CPU 부하를 가지고 있도록 하였다. 나머지 2대는 거의 부하가 없도록 하였다. 최소, 최대 임계값은 각각 70과 80으로 설정하였다.

그림 15는 라운드로빈 DNS와 DLBDNS의 사용자 응답시간과 파일 전송률에 관한 실험 결과이다. 본 실험 결과에서 사용자 응답시간과 파일 전송률이 DLBDNS가 라운드로빈 DNS보다 빠른 성능을 보였다. 클라이언트의 수가 증가하면서 네트워크 부하가 증가하기 때문에 사용자 응답시간이 증가하는 것을 볼 수 있다. 라운드로빈 DNS는 서버들의 상태를 고려하지 않기 때문에 부하가 많은 서버에도 사용자 요청을 동일하게 할당한다. 따라서 부하가 많은 서버에서는 처리속도가 느려지기 때문에 전체 시스템의 응답시간을 느리게 한다. 그러나 DLBDNS는 부하가 많은 서버를 DNS 리스트에서 삭제를 하기 때문에 부하가 많은 서버에는 사용자 요청이 많이 할당되지 않는다. 그리고 서버의 부하가 줄어들면 다시 사용자 요청을 처리할 수 있도록 한다. 클라이언트의 수가 증가하면서 파일 전송률은 증가하다가 네트워크 부하의 증가하기 때문에 감소하는 것을 볼 수 있다. 클라이언트 수가 증가하면서 파일 전송률의 감소폭이 라운드로빈 DNS는 크지만 DLBDNS는 작다. 따라서 DLBDNS가 라운드로빈 DNS보다 효율적인 부하분산을 한다는 것을 알 수 있다.

실험을 통하여 본 논문에서 구현한 DNS 기반 부하분산 시스템이 기존 라운드로빈 DNS보다 효율적으로 부하분산 하는 것을 보였다. 또한 본 논문에서 구현한 각 모듈들이 DNS에 많은 부하를 주지 않는 것을 볼 수 있었고 작은 TTL 값이 제안된 시스템에 미치는 영향을 측정하여 적합한 최소의 TTL 값을 측정하였다.

6. 결 론

DNS 기반 부하분산 시스템은 라운드로빈 방식을 이용하여 구성하기가 쉽고 분산 환경에 적합한 이점이 있지만 각 서버의 부하를 고려하지 않는다는 단점이 있다. 본 논문에서는 기존의 라운드로빈 DNS 방식의 단점을 보완하고 다양한 컨텐츠에 대하여 분산 웹 서버의 동적 부하분산을 위하여 DNS를 이용한 새로운 방식의 부하분산 시스템을 제안하였다. 이 시스템은 기존 라운드로빈 DNS에 부하분산 모듈을 추가하여 DNS를 수정하지 않고 부하분산 시스템을 구성하였다. DNS 리스트에 등록된 서버들의 부하 정보를 주기적으로 검사하여 한 서버로 집중될 수 있는 사용자의 요청을 효율적으로 분산한다. 분산 웹 시스템이 제공하는 서비스에 따라 CPU와 메모리, 네트워크 자원 사용량이 모두 다르다. 따라서 서비스의 종류에 따라 각 자원의 사용량에 가중치를 다르게 적용하여 부하를 측정하는 정책을 사용한다. 라운드로빈 방식과 동적 갱신 프로토콜을 사용하여 부하가 많은 서버는 DNS 리스트에서 동적으로 삭제한다. 이렇게 함으로써 전체적인 사용자 응답시간이 길어지는 것을 방지할 수 있다. GUI 기반의 관리 도구는 관리자가 손쉽게 DNS를 관리할 수 있도록 인터페이스를 제공한다. 실험을 통하여 본 논문에서 추가한 모듈들이 기존 DNS 시스템에 영향을 크게 미치지 않는 것을 보였다. 또한 본 논문에서 제안한 시스템이 기존 라운드로빈 DNS보다 성능이 우수하고 효율적으로 부하를 분산하는 것을 보였다.

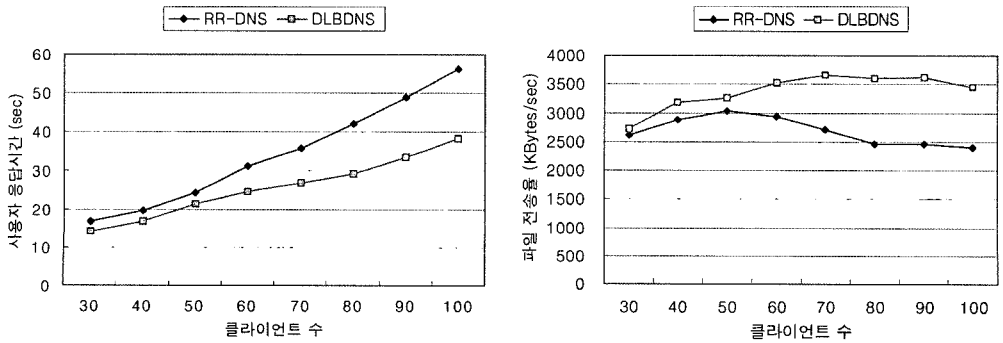

그림 15 라운드로빈 DNS와 DLBDNS의 성능 비교

참 고 문 헌

[1] Balachander Krishnamurthy, Craig Wills and Yin Zhang, "On the Use and Performance of Content Distribution Networks," Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement, pp. 169-182, 2001.

[2] Valerial Cardellini, Michele Colajanni and Philip S. Yu, "Geographic Load Balancing for Scalable Distributed Web Systems," Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 20-28, 2000.

[3] Valleria Cardellini, Michele Colajanni and Philip S. Yu, "Dynamic Load Balancing on Web-server Systems," IEEE Internet Computing, Vol. 3, No. 3, pp. 28-39, 1999.

[4] T. Kwan, R. McGrath and A. Reed, "NCSA's World Wide Web Server: Design and Performance," IEEE Computer, Vol. 28, No. 11, pp. 67-74, 1995.

[5] Valeria Cardellini, Emilinano Casalicchio, Michele Colajanni and Philip S. Yu, "The State of the Art in Locally Distributed Web-server System," ACM Computing Surveys (CSUR), Vol. 34, pp. 263-311, 2002.

[6] Yong Meng TEO, Rassul AYANI, "Comparison of Load Balancing Strategies on Cluster-based Web Servers," Transactions of the Society for Modeling and Simulation, 2000.

[7] Azer Bestavros, Mark Crovella, Jun Liu and David Martin, "Distributed Packet Rewriting and its Application to Scalable Server Architectures," Proceedings of the 6th International Conference on Network Protocols, Austin Texas, pp. 290-297, 1998.

[8] Cisco's DistributedDirector, http://www.cisco.com/

[9] Michele Colajanni, Philip S. Yu, "A Performance Study of Robust Load Sharing Strategies for Distributed Heterogeneous Web Server Systems," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 2, pp. 398-414, 2000.

[10] RFC 2136, "Dynamic Updates in the Domain Name System (DNS UPDATE)," http://www.ietf.org/ rfc/rfc2136.txt, 1997.

[11] RFC 2137, "Secure Domain Name Dynamic Update," http://www.ietf.org/rfc/rfc2137.txt, 1997.

[12] Micah Beck, Terry Moor, "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," Computer Networking and ISDN Systems, pp. 2141-2148, 1998.

[13] Anees Shaikh, Renu Tewari and Mukesh Agrawal, "On the Effectiveness of DNS-based Server Selection," Proceedings of IEEE INFOCOM, 2001.

[14] Zhuoquing Morley Mao, Charles D. Cranor, Fred Douglis, Michael Rabinovich, Olvier Spatscheck and Jia Wang, "A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers," Proceedings of USENIX Annual Technical Conference, 2002.

[15] Michele Colajanni, Philip S. Yu and Valeria Cardellini, "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers," Proceedings of the The 18th International Conference on Distributed Computing Systems, pp. 295-302, 1998.

[16] Valerial Cardellini, Michele Colajanni and Philip S. Yu, "DNS Dispatching Algorithms with State Estimators for Scalable Web-server Clusters," World Wide Web Journal, Baltzer Science, Vol. 2, No. 2, pp. 101-113, 1999.

[17] Daniel Andresen, Tao Yang and Oscar H. Ibarra, "Towards a Scalable WWW Server on Networked Workstations," Journal of Parallel and Distributed Computing, Vol. 42, pp. 91-100, 1997.

[18] Luis Aversa, Azer Bestavros, "Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting," IEEE International Performance, Computing, and Communications Conference, pp. 24-29, 2000.

[19] Ashish Singhai, Swee-Boon Lim and Sanjay R. Radia, "The SunSCALR Framework for Internet Servers," Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing, pp. 108-117, 1998.

[20] Roland J. Schemers, "lbnamed: A Load Balancing Name Server in Perl," Proceeding 9th systems Administration Conference, Monterey, CA, 1995.

[21] SIEGE, http://joedog.org/siege/

문 종 배

2000년 숭실대학교 컴퓨터학과(학사) 2002년 숭실대학교 컴퓨터학과(공학석사) 2002년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 분산/병렬 컴퓨팅, 그리드, 웹 서비스, 리눅스

김 명 호

1989년 숭실대학교 전자계산학과(학사) 1991년 포항공과대학교 전자계산학과(공학석사). 1995년 포항공과대학교 전자계산학과(공학박사). 1995년 한국전자통신연구소 선임연구원. 1998년~1999년 미국 테네시주립대 교환교수. 1995년~현재 숭실대학교 정보과학대학 컴퓨터학부 교수. 2005년~현재 미국 테네시주립대 교환교수. 관심분야는 분산/병렬 컴퓨팅, 그리드, 웹 서비스, BI, 보안