웹 서버 시스템에서의 자원 정보를 이용한 효율적인 부하분산 기법

장 태 무[†] · 명 원 식[††] · 한 준 탁[†††]

요 약

웹을 사용하는 사람들의 기하급수적으로 증가하면서 확장이 용이하며 신뢰도가 높은 웹 서버가 절실히 요구된다. 사용자의 급증은 과중한 전송량과 시스템의 부하문제를 야기 시켰으며 이를 해결하기 위한 방안으로 클러스터 시스템이 연구되고 있다. 기존의 클러스터 시스템에서는 웹 서버 간 부하가 균등하더라도 멀티미디어나 CGI 등으로 요청 데이터 크기가 크면 특정 웹 서버의 부하와 응답 시간이 증가되는 경향이 있다. 본 논문에서는 웹 서버들이 각각 다른 컨텐츠를 갖고 CPU, 메모리 및 디스크 사용율 등의 웹 서버의 자원 정보를 이용하여 효율적으로 부하를 분산하는 기법을 제안한다. 각각 서로 다른 컨텐츠를 가지고 있는 웹 서버들은 컨텐츠들에 대한 수정, 삭제, 추가 등 자원 정보 변경으로 인하여 깨어질 수도 있는 자원 정보 일관성을 유지하기 위해 네트워크 파일 시스템에 연결되어 운영된다. 컨텐츠를 나누어 저장함으로써 생길 수 있는 각 컨텐츠 그룹 간의 부하의 불균형의 문제는 웹 서버에 대한 재설정으로 해결하였다. 성능 실험을 통해 기존의 RR방식과 LC방식보다 제안한 기법이 최대 50%의 처리율과 응답시간 향상을 보여주었다.

Efficient Load Balancing Scheme using Resource Information in Web Server System

Chang Tae-Mu[†] · Myung Won-Shig[††] · Han Jun-Tak[†††]

ABSTRACT

The exponential growth of Web users requires the web servers with high expandability and reliability. It leads to the excessive transmission traffic and system overload problems. To solve these problems, cluster systems are widely studied. In conventional cluster systems, when the request size is large owing to such types as multimedia and CGI, the particular server load and response time tend to increase even if the overall loads are distributed evenly. In this paper, a cluster system is proposed where each Web server in the system has different contents and loads are distributed efficiently using the Web server resource information such as CPU, memory and disk utilization. Web servers having different contents are mutually connected and managed with a network file system to maintain information consistency required to support resource information updates, deletions, and additions. Load unbalance among contents group owing to distribution of contents can be alleviated by reassignment of Web servers. Using a simulation method, we showed that our method shows up to 50 % about average throughput and processing time improvement comparing to systems using each LC method and RR method.

키워드 : Visual Server, Web Server, Load Balancing, NFS, Contents, RR(Round-Robin) method, LC(Least Connection)

1. 서 론

현재의 웹(Web)은 웹 사용자의 증가와 함께 웹을 통해 전달되는 데이터, 즉 그림, 멀티미디어 데이터, 웹 문서 등의 크기 또한 빠르게 증가되고 있다[1,2,12]. 또한 최근에는 인터넷 트래픽(Internet traffic)의 많은 부분이 웹 트래픽(Web traffic)으로 구성되고 있으며 이러한 웹의 급격한 보급과 각종 멀티미디어 데이터를 포함하는 웹 서비스의 팽창은 네트워크의 병목현상(Bottleneck)을 점점 더 가중시키고 있다. 따라서 트래픽의 증가로 인해 웹 서버의 성능(Performance)과 가용성(Availability)을 높이는 방안이 더욱 중요시되고 있다[20].

현재의 웹 클러스터 구조는 클라이언트의 요청을 적절한 웹 서버로 전달해주는 하나의 디스패처와 동일한 컨텐츠(contents)을 가진 여러 대의 서버 노드로 구성되어 있다. 이 방식은 클라이언트들로부터 서비스 요청을 디스패처에서 특정한 웹 서버에 집중되지 않도록 적절한 스케줄링 방식으로 전체 서버에게 전달하는 디스패처 방식으로 이루어지고 있다.

디스패처 방식은 모든 클라이언트의 요청을 제어할 수 있

※ 본 연구는 2004년도 동국대학교 교비연구비의 지원으로 이루어졌음.
† 정 회 원 : 동국대학교 컴퓨터멀티미디어공학과 교수
†† 정 회 원 : 동국대학교 컴퓨터공학과 졸업
††† 정 회 원 : 동해대학교 컴퓨터공학과 조교수
논문접수 : 2004년 9월 10일, 심사완료 : 2005년 2월 23일

고 디스패처의 처리 작업을 통해서 웹 서버에게 전달된다. 또한 디스패처로 도착하는 모든 패킷들을 관리하고 변경하는 작업들은 많은 처리 비용이 들고 클라이언트의 요청이 많아지면 병목 현상을 일으킬 수 있다. 나아가 문자(Text) 정보 같은 작은 데이터만 요구하는 패킷을 받는 경우는 별 문제가 되지는 않지만 요즘과 같이 기본적인 문자뿐만 아니라 동영상 파일, 음악 파일, MPEG, MP3등의 대용량의 멀티미디어 데이터를 요구하는 경우엔 웹 서버 간 부하가 균등하더라도 요청된 데이터의 크기가 크면 특정 웹 서버의 부하는 증가되고 전체 웹 서버의 부하는 불균등 하게 된다. 따라서 패킷을 처리하는데 많은 시간이 걸리거나 심각할 경우 더 이상 서비스를 할 수 없게 될 수도 있다.

본 논문에서 제안한 효율적인 부하분산 기법은 다음과 같은 기능들을 기반으로 설계되었다.

첫째, 다수의 노드로 구성된 클러스터 시스템을 단일 시스템처럼 인식할 수 있는 가상의 단일 시스템 환경을 제공하여야 한다. 이렇게 함으로써 사용자의 트랜잭션이 어느 노드에서 수행되는가에 구애를 받지 않는 투명성을 제공하게 된다.

둘째, 전체 시스템 구성과 각 노드의 부하 및 자원의 활용상태의 파악이 용이하여야 한다. 각 노드의 자원상태나 부하의 상태를 실시간으로 정확한 자원에 대한 모니터링 함으로써 각 노드의 부하를 측정하고 사용자의 트랜잭션 요구를 효율적으로 분산하여 클러스터의 성능을 향상해야 한다.

셋째, 모든 노드의 성능을 최대한 발휘하기 위해 사용자의 요구를 적절히 분산시키는 스케줄링 방법이 필요하다.

2. 관련연구

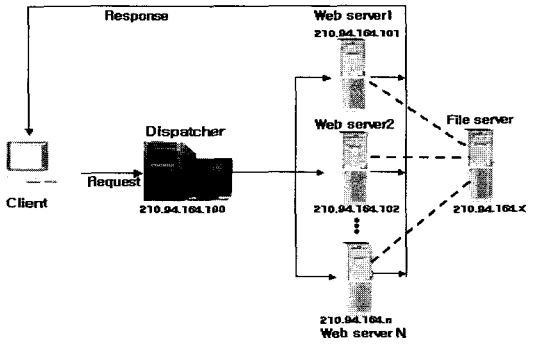
2.1 디스패처 방식

디스패처 방식은 웹 클러스터 앞 단에 중계 서버(front-end)가 존재한다. 장점으로는 모든 요청과 응답을 중계 서버가 감시하기 때문에 서버의 상태 파악이 수월하고, 서버의 이더넷 어댑터를 수정할 필요가 없으므로 운영체제에 종속적이지 않고 자유롭다. 단점으로는 중계 서버가 모든 트래픽을 처리하여야 하므로 단일 병목 지점으로 작용할 수가 있게 된다.

특히 문자 정보 같은 작은 데이터만 요구하는 패킷을 받는 경우는 별 문제가 되지는 않지만 요즘과 같이 기본적인 문자 뿐만 아니라 동영상 파일, 음악 파일, 이미지 파일과 같은 많은 양의 멀티미디어(MPEG, MP3)를 요구하는 경우엔 웹 서버 간 부하가 균등하더라도 요청된 데이터의 크기가 크면 특정 웹 서버의 부하는 증가되고 전체 웹 서버의 부하는 불균등 하게 된다. 따라서 패킷을 처리하는데 많은 시간이 걸리거나 심각할 경우 더 이상 서비스를 할 수 없게 될 수도 있다.

2.2 하나의 공유 파일 서버로 구성된 웹 서버 클러스터 방식

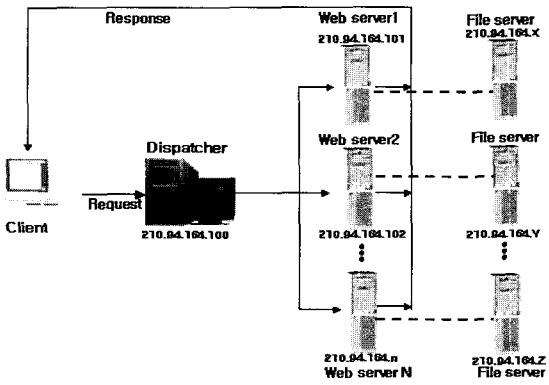
(그림 1)은 단일 공유 파일 서버로 구성된 웹 서버 클러스터의 모습이다. 공유 파일 서버를 가진다는 것은 임의의 웹 서버가 컨텐츠를 변경하여도 다른 웹 서버들이 변경된 컨텐츠를 그대로 사용할 수 있다는 것이다. 이런 구조는 상대적으로 구현하기 쉽고 비용을 줄일 수 있는 등의 장점을 갖고 있으나 웹 서버들의 개수가 많아질수록 하나의 공유 파일 서버로의 병목현상이 생길 수 있다. 특히 요즘 같이 멀티미디어, CGI 등의 동적 페이지의 생성이 많아지면 데이터베이스의 크기가 증가하게 되어 병목현상이 심화되고, 그에 따르는 확장성이 불가피하여 요구된다.



(그림 1) 하나의 공유 파일 서버로 구성된 웹 서버 클러스터 구조

2.3 동일 파일 서버로 구성된 웹 서버 클러스터 방식

(그림 2)는 여러 개의 동일 파일 서버로 구성된 웹 서버 클러스터 구조를 보여주고 있다. 이 구조은 웹 사이트에 대한 전체 컨텐츠를 동일하게 복사하여 여러 파일 서버로 가지고 있는 방식이다. 이는 각각의 웹 서버들이 자신의 파일 시스템에서 데이터를 읽어와 사용자의 요청을 서비스해주기 때문에 병목현상이 발생하지 않는다. 그러나 동적 웹 페이지의 생성의 증가에 따라 각각의 파일 서버마다 다른 내용을 가질 수 있어 새로운 웹 페이지가 생성되거나 데이터베이스가 갱신되었을 때마다 각 파일 서버들의 내용도 알맞게 갱신하는 동기화의 과정이 필요하다. 이것은 동적인 웹 서비스가 증가하는 경우 상당한 오버헤드(Overhead)로 작용될 수 있다.



(그림 2) 동일 파일 서버로 구성된 웹 서버 클러스터 구조

3. 컨텐츠 기반 웹 서버 클러스터 구조

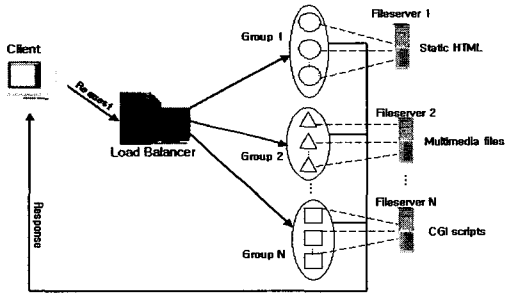
기존 방식에서는 실제 웹 서버의 부하 상태를 유지하며, 새로운 클라이언트 요청이 도착하면 부하가 가장 작은 웹 서버를 선택하여 서비스하는 방법으로 운용되고 있다. 이 방식은 실제 서버들 중에서 현재 가장 적은 수의 요청을 처리하고 있는 서버를 선택하여 요청 패킷을 할당하는 방식이다. 이는 트랙픽량에만 의존하므로 용량이 큰 멀티미디어 데이터(동영상)를 요구할 때는 응답시간이 길어지고 처리율이 저하되는 단점이 발생한다. 반면에 본 논문에서는 각각의 웹 서버가 주기적으로 자원상태(CPU 사용율과 메모리 사용율 그리고 DISK 사용율)를 부하 분산기에게로 보고하고 특정한 웹 서버에서 부하량이 임계치(Threshold)를 넘게 될 때 제안한 알고리즘에 의해 선택된 웹 서버로 새로운 요청을 전달하는 방법으로 설계되었다.

본 논문의 목적은 특정한 웹 서버 그룹이 기준 부하량을 넘을 때 클라이언트 요청을 다른 웹 서버에 전달함으로써 클러스터링 웹 서버의 처리율을 향상시킨다. 또한 부하를 다른 웹 서버에 분산할 때 웹 서버들의 자원 정보와 컨텐츠에 대한 접속 빈도수를 고려하여 선택된 웹 서버에게로 클라이언트의 요청을 전달함으로써 사용자의 응답시간을 줄이고 처리량에 대한 성능 향상에 있다.

(그림 3)은 하나의 부하 분산기와 부하 분산기가 관리하는 웹 서버들로 구성되어 있고, 웹 서버들은 컨텐츠의 종류에 따라 다시 그룹화 되었다. 한 그룹의 웹 서버들은 하나의 네트워크 파일 시스템에 링크되어 있고, 동일한 컨텐츠를 서비스한다. 부하 분산기가 관리하는 웹 서버와 그룹별 수는 각기 다를 수 있다.

3.1 동작원리

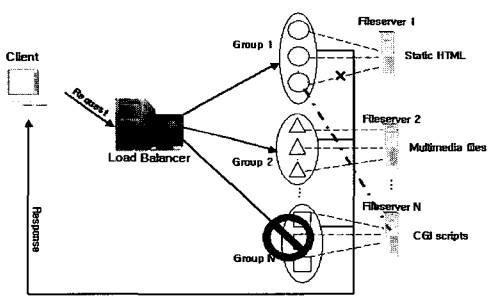
부하 분산기는 들어오는 클라이언트의 요청을 적절한 스케줄링 알고리즘에 의해 실제 서비스를 해주는 실제 웹 서버로 분산시켜주는 역할을 하는 노드를 말한다.



(그림 3) 제안한 웹 서버 클러스터 시스템 구조

본 논문에서는 그룹 내의 스케줄링 알고리즘으로 라운드로빈 방법을 채택하였다. 또한 특정한 그룹에서 병목현상이 발생할 경우 병목현상을 제거하고 원할한 서비스를 하기 위해 부하 분산기는 자신이 관리하는 웹 서버들의 부하 상태를 주기적으로 모니터링 할 수 있어야 한다.

웹 서버의 부하를 측정하는 정보로는 웹 서버들의 자원 정보를 사용한다. 즉, 각 웹 서버들에 대한 CPU 사용율과 메모리 사용율 그리고 디스크 사용율이다. 자원 정보를 사용하는 것은 실제 웹 서버의 유효 연결들이 멀티미디어 데이터(동영상) 서비스와 같이 큰 작업일 경우에는 그 수가 많지 않더라도 서비스되는 용량이 크기 때문에 처리시간이 오래 걸리며, 이에 새로이 요청되는 작업은 지연될 가능성이 아주 크다. 역으로, 실제 웹 서버의 유효 커넥션의 수가 많지만 대분분이 작은 작업을 처리하는 커넥션들이라면, 비록 과부하 상태일지라도 요청을 이 실제 웹 서버에서 수행하는 것이 사용자 응답시간을 줄일 수 있기 때문이다. 즉, 정적 컨텐츠(Static contents)를 서비스하는 웹 서버는 대부분의 작업이 디스크 접근(Disk I/O)이므로 입출력 병목이 발생할 수 있으며, 또한 멀티미디어 파일이나 CGI와 같은 동적 컨텐츠(Dynamic contents)를 요구하는 작업들만 서비스하는 웹 서버는 CPU와 메모리가 과부하 될 수 있다는 것이다[18]. 이러한 이유로 부하 분산기는 각 웹 서버들의 자원 정보(CPU 사용율과 메모리 사용율 그리고 디스크 사용율)들을 모니터링 해야 한다. 따라서 웹 서버들은 이와 같은 자원 정보 중에서 가장 큰 비율을 가진 자원 정보를 일정 시간마다 부하 분산기에게 보고한다. 또한 웹 서버들은 가장 큰 비율을 가진 자원 정보 중에서 어느 임계 값을 넘게 되면 부하 분산기에게 경고 메시지(Alarm Message)를 보낸다. 이는 현재 자기 웹 서버 상태에서 서비스할 부하량이 많음을 말한다. 경고 메시지를 받은 부하 분산기는 웹 서버들에 대한 일정량의 웹 로그(Web-Log)를 수집하여 각 웹 서버들에 대한 컨텐츠 접속 빈도수를 계산한다. 접속 빈도수를 계산한 값과 각 웹 서버에서 보고한 자원 정보의 값을 계산하여 가장 작은 값을 가진 웹 서버에게 새로운 클라이언트 요청을 전달하게 된다. 이 때 요청을 받게 될 웹 서버는 현재 네트워크 파일 시스템과의 커넥션을 해제한 후 부하량이 많은 네트워크 파일 시스템에 링크 재설정을 하여 서비스를 제공한다.



(그림 4) 병목현상을 제거하기 위한 링크 재설정

(그림 4)은 특정 그룹(CGI script)에 대한 부하량(자원정보)이 임계값보다 커짐에 따라 경고 메시지를 발생시킨다. 따라서 경고 메시지를 발생시킨 그룹 외의 다른 그룹들에서 컨텐츠 접속 빈도수와 각 웹 서버에 대한 자원 정보의 값을

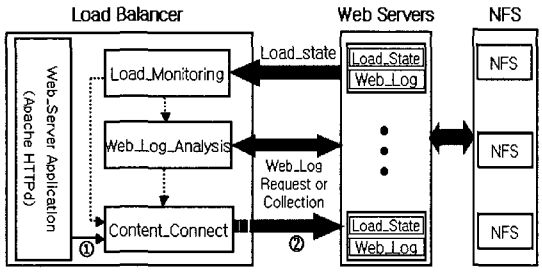
계산한 결과 값이 가장 작은 웹 서버를 선택하여 경고 메시지를 발생한 그룹의 네트워크 파일 시스템과 링크 재설정(Link-Reaction)을 통해 병목현상을 해결하는 것을 보여주고 있다. 예를 들면 특정 그룹의 웹 서버가 3개로 구성되어 있다면, 링크 재설정으로 4개의 웹 서버로 서비스하는 것이다. 결국 병목현상을 링크 재설정을 통해 해결함으로써 응답시간을 빠르게 할 수 있고, 처리 시간과 처리율도 향상될 수 있는 것이다.

3.2 제안한 시스템 구조의 프레임워크

(그림 5)는 부하 분산기와 웹 서버들 간의 모듈 관계를 보여 주고 있다. 부하 분산기 내의 ①과 ②의 흐름은 정상적인 동작일 때의 흐름이다. 즉, 웹 서버들 중 어느 것도 부하량에 대한 병목현상이 발생하지 않았을 경우를 말한다. 여기서 Web_Server Application은 Apache로 구동되며 웹 사이트의 HTTP_데몬 이다. 사용자는 HTTP_데몬을 통해서 접속할 수 있으며, 응답은 해당 컨텐츠를 가지고 있는 웹 서버로부터 응답을 받는다.

3.2.1 Load_State 모듈

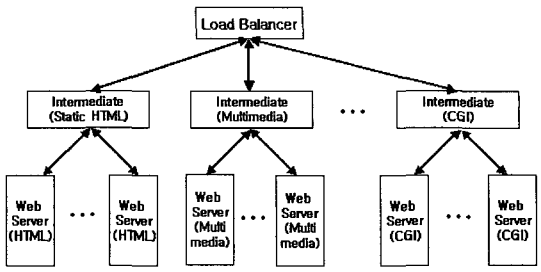
기존의 클러스터 방식에서는 실제 웹 서버의 부하상태를 유지하며, 새로운 요청이 도착하면 가장 적은 부하가 걸린 실제 웹 서버를 선택한다. 이는 해당 웹 서버에서 모든 처리가 가능한데도 부하 분산기에서 부하분산 스케줄링 기법으로 부하 균등화를 하고 있다.



(그림 5) 부하 분산기와 웹 서버들간의 프레임워크

부하 균등화를 위해 부하 분산기에서는 계속적인 웹 서버들에 대한 자원상태를 파악하고 계산해야 하므로 그 만큼 요청에 대한 처리 시간이 지연될 수 있다. 또한 요즘 같이 초당 접속률이 많은 경우엔 부하 분산기에서 자원정보를 파악하고 그에 따르는 웹 서버를 선택하는 시간 동안 요청되는 패킷에 대해서는 정확한 부하 분산을 할 수 없다.

따라서 좀 더 효과적으로 해결하기 위해 본 논문에서는 (그림 6)에서 보는 거와 같이 한 그룹마다 중계기를 두었다. 한 그룹 내의 웹 서버들 중 하나의 웹 서버가 중계기 역할을 담당한다. 중계기는 주기적으로 자기 그룹 내의 웹 서버들에 대한 자원 정보와 경고 메시지(Alarm Message)를 보고 받는다. 이러한 자원 정보와 경고 메시지는 주기적으로 부하 분산기의 Load_Monitoring 모듈로 전달된다. 여기서



(그림 6) 부하 분산기와 웹 서버들 간의 트리 구조

자원 정보를 모니터닝 하는 것과 경고 메시지는 아래의 식과 같다.

$$State_i = Max\ (CPU_i,\ 메모리_i,\ 디스크_i); \quad (1)$$

$$if\ (State_i >= THRESHOLD)\ then\ \ Alarm\_Message_i = 1; \quad (2)$$

i	: 웹 서버의 번호
$State_i$	: 각 웹 서버에 대한 자원 정보의 상태
CPU_i	: CPU 사용율 ($0 \leq CPU_i \leq 1$)
$메모리_i$	: 메모리 사용율 ($0 \leq 메모리_i \leq 1$)
$디스크_i$	: 디스크 입출력 사용율 ($0 \leq 디스크_i \leq 1$)
THRESHOLD	: 70% [3]

식 (1)은 각 웹 서버의 자원 정보 값들에서의 최대 값 이다. 가장 큰 값을 중계기에 전달한다. 컨텐츠별로 웹 서버들을 특화해서 서비스하기 때문에 정적 컨텐츠(StaticHTML)를 서비스하는 웹 서버는 대부분 디스크 접근(Disk I/O)이므로 입출력 병목이 발생하고 동적 컨텐츠(CGI, 멀티미디어)를 서비스하는 웹 서버에서는 CPU와 메모리의 과부하가 있기 때문이다. 식 (2)는 경고 메시지를 보낼 수 있는 상태이다. 각 웹 서버들의 자원 정보들 중 가장 큰 값을 가진 자원이 어떤 임계(Threshold) 값에 같거나 클 경우에는 경고 메시지를 중계기에 보내게 된다. 이는 현재 자기 웹 서버에서 부하량이 많을 경우를 말한다. 식 (2)을 만족하면 경고 메시지의 값은 1을 전달하게 된다.

3.2.2 Load_Monitoring 모듈

이 모듈은 각 그룹 내의 중계기로부터 자원 정보의 최대 값과 경고 메시지를 주기적으로 보고 받는다. 만약 경고 메시지 값이 1이면 현 모듈에서는 그룹별 웹 서버 자원 정보의 최대 값과 경고 메시지를 Content_Connect 모듈에게로 전달함과 동시에 Web_Log_Analysis 모듈에게로도 보낸다.

3.2.3 Web_Log_Analysis 모듈

이 모듈을 각 그룹에 대한 웹 로그 수집하여 컨텐츠에 대한 접속 빈도수를 분석하는 모듈이다. Load_Monitoring 모듈로부터 경고 메시지 1를 받으면, 각 그룹 내의 웹 서버에게 웹 로그를 요청하여 컨텐츠에 대한 접속 빈도수를 분석한다. 접속 빈도수를 구하는 식은 아래 식 (3), (4), (5)와 같다.

$$\text{Frequency\_StaticHTML} = \left(\frac{\sum_{i=1}^{n} StaticHTML_i}{N} \right) * Ws; \quad (3)$$

$$\text{Frequency\_멀티미디어} = \left(\frac{\sum_{i=1}^{n} Multimedia_i}{N} \right) * Wm; \quad (4)$$

$$\text{Frequency\_CGI} = \left(\frac{\sum_{i=1}^{n} CGI_i}{N} \right) * Wc; \quad (5)$$

Frequency_StaticHTML : StaticHTML 컨텐츠에 대한 접속 빈도수
Frequency_멀티미디어 : 멀티미디어 컨텐츠에 대한 접속 빈도수
Frequency_CGI : CGI 컨텐츠에 대한 접속 빈도수
StaticHTML : 정적 컨텐츠에 대한 로그 수
멀티미디어 : 멀티미디어 컨텐츠에 대한 로그 수
CGI : CGI 컨텐츠에 대한 로그 수
n : 그룹 내의 웹 서버 수
N : 제안된 시간동안 액세스 로그 수
W_s : StaticHTML에 대한 가중치
W_m : 멀티미디어에 대한 가중치
W_c : CGI에 대한 가중치

여기서 가중치(Weight)을 주는 것는 각 컨텐츠에 대한 접속 빈도수를 계산하여도 이 값으로 부하를 정확하게 해결할 수 없기 때문이다. 동일한 시간대에 정적 컨텐츠를 처리한 StaticHTML의 접속 빈도수와 멀티미디어와 CGI 같은 동적 컨텐츠를 처리한 접속 빈도수를 같은 조건으로 비교하면 정적 컨텐츠의 접속 빈도수가 높을 확률이 많다. 따라서 이를 해결하고자 각 컨텐츠에 대해 다른 가중치를 적용하여 계산한다. 위의 (3), (4), (5)식을 이용 각 그룹에 대한 접속 빈도수를 구한 값을 Content_Connect 모듈로 전달한다.

3.2.4 Content_Connect 모듈

이 모듈은 웹 서버 어플리케이션(Apache) HTTP_데몬으로 들어오는 클라이언트 요청을 해당 인덱스 테이블에 따라 서비스할 웹 서버를 선택하여 클라이언트의 요청을 전달한다. 또한 특정한 웹 서버에서 병목현상이 발생하여 부하를 분산할 경우에 Load_Monitoring으로부터 전달 받은 웹 서버들의 자원 정보에 대한 정보와 Web_Log_Analysis로부터 전달 받은 컨텐츠 접속 빈도수에 대한 정보를 가지고 서비스할 웹 서버를 선택하여 클라이언트 요청을 전달하는 역할을 한다. 또한 선택된 웹 서버를 링크 재설정(Link-Reaction)하는 부분도 이 모듈에서 행해진다. 아래의 식 (6)은 새로 선택될 실제 웹 서버를 구하는 식이다.

$$Selected\_Webserver = Min((Frequency\_StaticHTML * Statei), (Frequency\_멀티미디어 * Statei), (Frequency\_CGI * Statei)); \quad (6)$$

i : 웹 서버 번호
Selected_Webserver : 링크 재설정을 위한 선택되는 실제 웹 서버

여기서, 각 값을 구한 후 가장 작은 값(Min)을 가진 웹 서버를 선택(Selected_Webserver)하여 네트워크 파일 시스템에 링크 재설정을 하고 요청에 대한 서비스를 한다.

<표 1>은 클라이언트 요청이 부하 분산기에 도착 했을 때 요청에 서비스를 하기 위하여 해당 그룹을 선택해야 하는데, 이때 부하 분산기 서버 내의 인덱스 테이블 요소들을 가지고 해당 그룹을 선택하도록 함을 보여주고 있다. 예를 들면, 클라이언트 요청이 게시판을 접속하고자 한다면 부하 분산기는 자기가 관리하는 인덱스 테이블에 따라 그룹3의 웹 서버에게로 요청을 전달하게 된다. 그룹 내의 실제 웹 서버들은 동일한 컨텐츠를 가진다.

〈표 1〉 부하 분산기 내의 인덱스 테이블 요소들

웹 서버 클러스터 내의 그룹	Content
Group 1	Static HTML(HTML, text, document etc)
Group 2	멀티미디어 파일(gif, jpg, swf etc)
Group 3	CGI((Notice board, guest book, DB etc)

만약, 식(6)의 Min 값이 동일하게 나올 경우 최근 10분 동안의 접속 빈도수 변화율을 측정하여 접속 빈도수가 떨어지고 있는 웹 서버를 선택한다. 또한 같은 그룹 내의 웹 서버들 중에서 동일한 Min값이 나올 경우에는 라운드 로빈(RR) 스케줄링 방식으로 서비스할 웹 서버를 선택한다.

3.2.5 Web_Log 모듈

이 모듈은 각 웹 서버 상태에서 웹 로그를 수집한다. 웹 서버에 따라 한 개가 아닌 여러 개의 로그 파일을 만들 수 있는데, 본 논문에서는 액세스(access) 로그 파일만 관리한다. 액세스 로그 파일은 일반적인 사이트 방문기록 등을 기록하며 이를 토대로 웹 사이트 방문 시간 및 방문 경로 등을 파악한다. 이 모듈에서는 웹 로그로부터 각 파일에 대한 접근 빈도로 접속 빈도수를 생성하기 위해서, 페이지 뷰(page view)을 측정 단위로 한다.

4. 실험 및 성능 평가

본 장에서는 본 논문에서 제안된 웹 서버 클러스터 시스템 모델의 성능 실험을 측정하기 위해 WOW Linux 7.0 운영체제에서 UC Berkeley 대학의 네트워크 시뮬레이터인 NS-2.1b8a를 사용하여 실험 하였다. 시스템 사양은 Pentium Ⅲ 1.GHz, RAM 256MByte, HDD 30GB로 구성하였다. 제안한 시스템 구조에서의 모든 웹 서버 및 부하 분산기는 동일한 사양을 사용하였다.

4.1 실험 환경

본 논문에서는 기존의 클러스터링 시스템의 스케줄링 방식 중 라운드 로빈 방식(로드 밸런서에 들어오는 요청 패킷들을 차례대로 실제 서버에 할당하는 방법)과 최소 연결 방식(Least Connection : 웹 서버들 중에서 현재 가장 적은 수의 요청을 처리하고 있는 서버를 선택하여 요청 패킷을 할

당하는 방법)과의 비교를 통해 처리시간과 처리율에 대한 성능을 측정한다. 또한 부하량이 임계 값을 넘을 때 링크 재설정(Link Reaction)을 함으로써 기존의 모델과의 처리량을 대하여 평가한다.

<표 2>에서 제안된 시스템 모델의 구성이 클라이언트 요청을 해당 웹 서버에게 전달하는 부하 분산기는 1 대이고, 컨텐츠별로 나누어 가지고 있는 그룹은 세 그룹이다. 즉 CGI 컨텐츠만을 관리 서비스하는 그룹, 멀티미디어 컨텐츠만을 관리 서비스하는 그룹과 HTML관련 컨텐츠만을 관리 서비스하는 그룹이다. 한 그룹 내의 동일한 컨텐츠를 서비스하는 웹 서버는 3개씩 구성되어 있다. 각 그룹마다 한 개의 네트워크 파일 시스템(NFS)는 일대일로 링크되어 있으며, 한 그룹 내의 웹 서버들은 NFS와의 링크를 통해 요청의 데이터들을 클라이언트에게 서비스하게 된다. 제안 2는 임의적으로 특정 그룹에 대한 부하량을 증가하여 그룹별 서버 개수의 변경이 자동적으로 이루어지게 하였다. 제안 3:4:2은 CGI 파일들의 부하량을 증가시켰을 때 Multimedia을 서비스하는 웹 서버들 중 하나를 링크 재설정하여 4개의 웹 서버로 서비스를 한다. 제안 3:1:5은 Multimedia 파일들의 부하량을 증가시켰을 때 CGI을 서비스하는 웹 서버들 중 두 개를 링크 재설정하여 5개의 웹 서버로 서비스를 한다.

〈표 2〉 실험에 사용된 시스템 파라메터

	LC	RR	제안1	제안2
Load_Balancer	1	1	1	1
Web server	9	9	3:3:3	3:4:2,3:1:5
Group			3	3
요청 수	100~800	100~800	100~800	100~800
NFS			3	3

4.2 웹 로그 분석

기존의 모델과 제안한 웹 서버 클러스터 시스템 성능을 비교하기 위해 컨텐츠별로 웹 로그 파일들을 두 가지 패턴으로 임의의 입력 데이터를 만들어 실험하였다. 첫 번째로 본 대학 입시관련 서버에 대한 웹 로그 패턴을 이용하였다. 이는 2003년 4일 동안의 요청 기록이다. StaticHTML 34%, CGI 58%, 멀티미디어 파일 8%이다(실험 1). 두번째로 사용된 임의의 입력 데이터는 웹 로그 파일들의 패턴[4]을 기준으로 StaticHTML 31%, CGI 5%, 멀티미디어 파일 64%이다(실험 2).

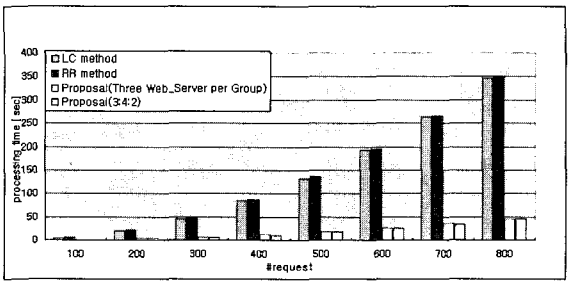
4.3 실험1

실험 1은 동국대학교 입시관련 서버에 대한 웹 로그를 분석하여 컨텐츠에 대한 사용자의 접근 패턴으로 입력 데이터를 사용하였다. 입력 데이터의 비율은 4.2절에서 기술한 실험 1과 동일하다. 총 요청 수는 64,662개로 에러 부분 3,230개를 제외한 접근된 로그만을 추출하였다.

추출된 접근된 웹 로그는 CGI가 가장 많은 35,468개이고, HTML이 20,796개이다. 멀티미디어는 5,168개로 요청 수가 가장 적다.

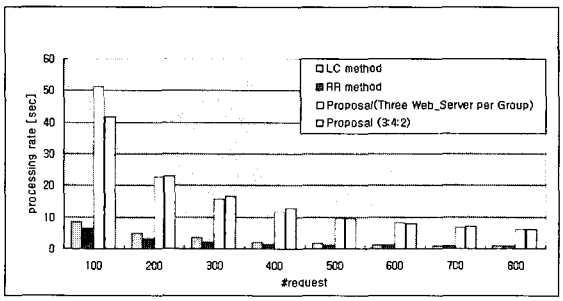
실험 1에서는 입력 데이터로 위와 같은 조건의 데이터를 임의로 생성하여 실험하였다. 즉 요청 수가 100개에서 CGI는 58개, HTML은 34개, 멀티미디어는 8개의 입력 데이터를 적용하였다.

(그림 7)에서 (그림 12)까지는 기존 모델과 제안한 그룹별 웹 서버 개수 그리고 요청 수가 100에서 800개에 대한 각각의 컨텐츠에 대한 처리시간과 초당 처리율에 대한 실험 결과이다.

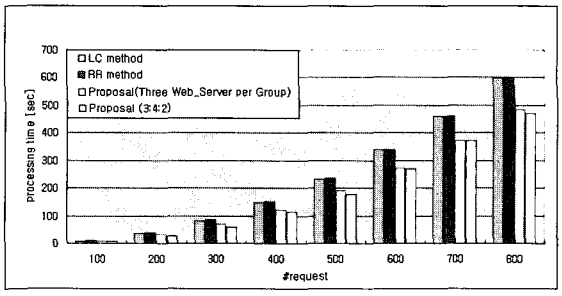


(그림 7) StaticHTML에 대한 요청 처리시간

(그림 7)에 보듯이 RR 방식과 LC 방식에서는 요청 수가 증가할수록 처리시간이 매우 크게 증가됨을 볼 수 있다. 실제 처리시간은 요청 수가 100개 일 때 RR은 5.4초이고, LC는 4초이며, 제안된 방식은 0.7초이다. 제안된 모델에서는 크게 변화됨이 없이 다른 기존 모델보다 처리시간이 빠름을 알 수 있다. (그림 8)은 초당 처리율이다. 800개 요청에 대해 제안 모델이 기존 모델보다 처리율이 약 7.5배 정도의 차이를 볼 수 있다.

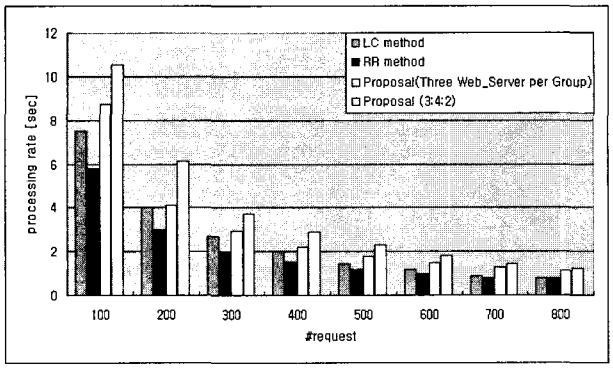


(그림 8) StaticHTML에 대한 요청 처리율

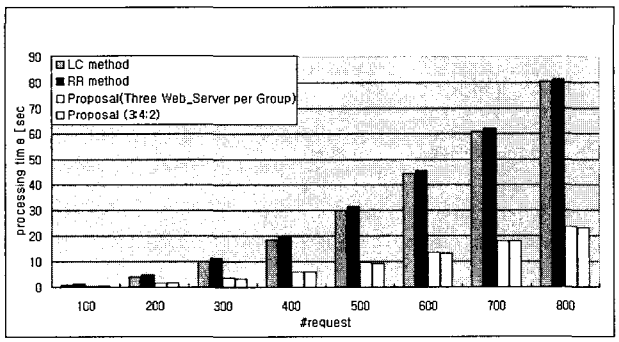


(그림 9) CGI에 대한 요청 처리시간

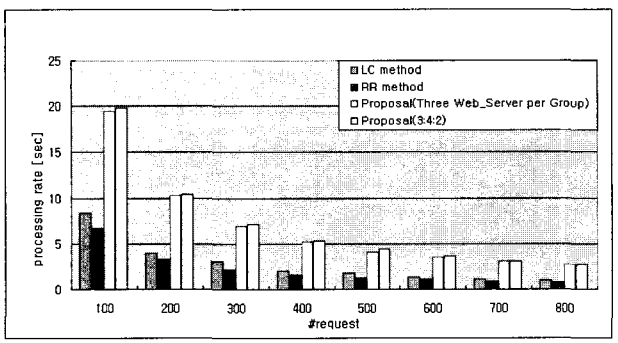
CGI 컨텐츠는 입력 데이터의 58%로 가장 많은 량의 요청의 수가 적용된다. (그림 9)에서 요청 개수가 증가할수록 RR 방식과 LC 방식이 제안한 모델과 거의 비슷한 처리시간을 볼 수 있다. 이는 RR 방식과 LC 방식이 구성하고 있는 웹 서버 수가 9대로써 전체 요청 수를 고루 나누어 처리하기 때문이다. 9대로 구성한 이유는 제안 모델과의 동일한 환경에서 실험하기 위함이다. 실제 처리시간은 요청 수가 100개 일 때 RR은 10초이고, LC는 7초이며, 제안된 방식의 3:3:3과 3:4:2에 대해 각각 7.7초와 5.6초이다. (그림 10)은 CGI에 대한 초당 처리율이다. 제안 모델이 기존 모델보다 처리율이 높음을 알 수 있다.



(그림 10) CGI에 대한 요청 처리율



(그림 11) 멀티미디어에 대한 요청 처리시간

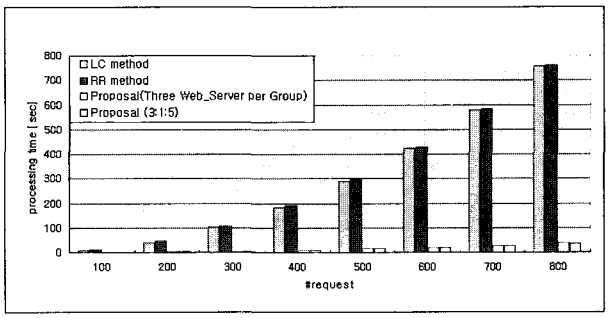


(그림 12) 멀티미디어에 대한 요청 처리율

(그림 11)과 (그림 12)는 멀티미디어에 대한 요청 처리시간과 처리율이다. 멀티미디어는 가장 적은 비율인 8%로 입력되었다. 제안(3:4:2) 모델과 제안(그룹: 웹 서버 3대) 모델의 처리율이 비슷하다. 이는 적은 입력 비율로 인해 웹 서버 수가 증가(링크 재설정)하여도 영향이 미치지 못함을 알 수 있다.

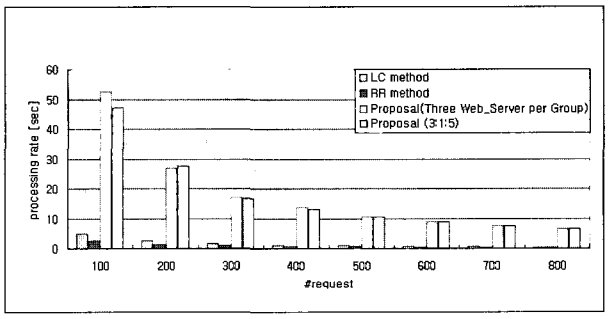
4.4 실험 2

실험 2는 [4]의 NASA Data의 패턴을 이용하여 임의의 입력 데이터를 사용하였다. 입력 데이터의 비율은 4.2절에서 기술한 실험 2과 동일하다.



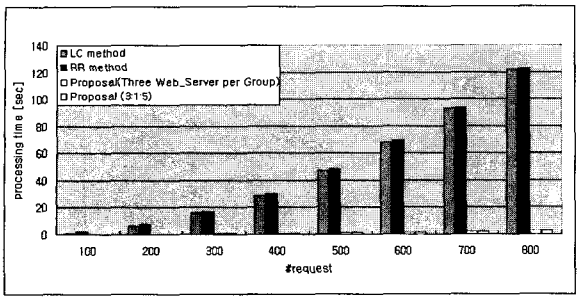
(그림 13) StaticHTML에 대한 요청 처리시간

(그림 13)과 (그림 14)는 StaticHTML에 대한 요청 처리시간과 초당 처리율이다. 앞의 실험 1과 실험 2의 Static HTML에 대한 실험 결과와 비슷한 형태이다. 제안 모델이 기존 모델보다 처리시간과 처리율에서 월등히 높음을 알 수 있다. 제안(3:1:5) 모델과 제안(1그룹: 웹 서버 3대) 모델은 StaticHTML의 처리시간과 처리율은 거의 동일하다. 이는 StaticHTML을 서비스하는 웹 서버 수가 동일하기 때문이다.

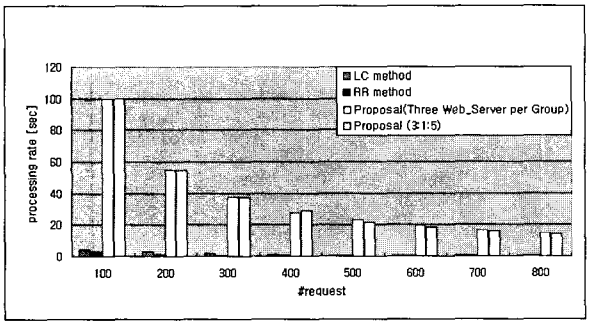


(그림 14) StaticHTML에 대한 요청 처리율

(그림 15)와 (그림 16)은 CGI에 대한 요청 처리시간과 처리율이다. 실험 2에서 CGI는 입력 데이터 중 가장 적은 5%이다. 제안 모델은 요청 수가 증가하여도 거의 변동이 없다. 이는 아주 적은 데이터가 입력되므로 이를 서비스하는 웹 서버에게 아무런 영향을 주지 못하기 때문이다. 제안(3:1:5) 모델은 웹 서버 1대가 모든 CGI들을 서비스한다.

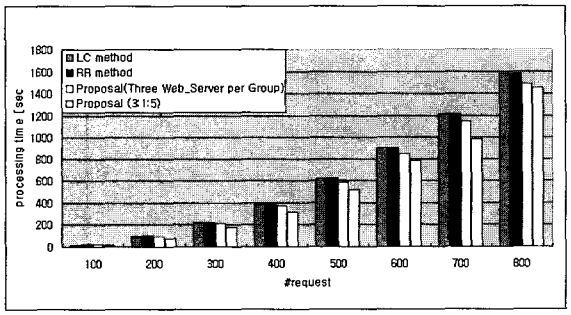


(그림 15) CGI에 대한 요청 처리시간

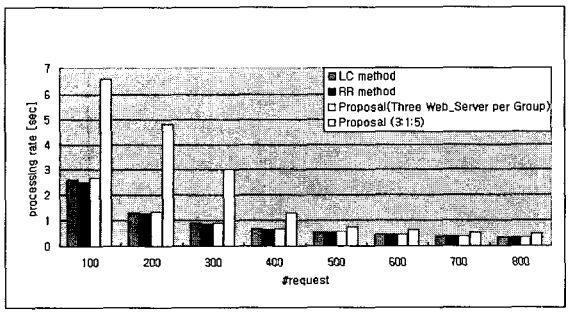


(그림 16) CGI에 대한 요청 처리율

(그림 16)은 CGI에 대한 처리율로서 CGI에 대한 입력 데이터가 아주 미약하기 때문에 웹 서버의 개수에 큰 영향을 주지 못하고 있다.



(그림 17) 멀티미디어에 대한 요청 처리시간



(그림 18) 멀티미디어에 대한 요청 처리율

(그림 17)과 (그림 18)은 멀티미디어에 대한 처리시간과 처리율에 대한 실험 결과이다. (그림 17)에서 제안(3:1:5) 모델과 제안(그룹: 웹 서버 3대) 모델과의 처리시간은 요청 수 100에서 보면, 전자는 15.14초이고 후자는 23.9이다. 약 1.6배 정도의 처리시간의 차이를 볼 수 있다. 전자는 멀티미디어 파일을 웹 서버 5대가 서비스하고 후자는 웹 서버 3대가 서비스한다. 그러나 멀티미디어 파일의 응답 전송이 대용량이므로 요청의 수가 증가할수록 기존 모델보다 크게 향상되지는 않는다. 처리율에 있어서도 일정 요청 수가 넘으면 크게 변화되지 않는 처리율을 보인다. 이와 같은 상황은 본 실험의 (그림 18)의 요청 500 이상이 입력되었을 때 발생됨을 볼 수 있다.

4.5 실험 결과

제안한 모델에서 그룹별로 정해진 웹 서버들은 동일한 컨텐츠을 제공하며, 또한 동일한 시스템 사양으로 구성되어 있다. 전체적으로 제안한 웹 서버 클러스터 시스템이 기존의 복사된(Replication) 웹 서버로 구성된 RR 방식과 LC 방식보다 처리시간과 처리율에서 월등히 향상되었음을 알 수 있었는데, 이러한 이유는 오래 걸리는 클라이언트 요청으로 인해 짧은 요청이 기다려야 하는 기존의 방식을 제안한 시스템 기법이 해결했기 때문이며, 또한 컨텐츠별로 웹 서버가 1대 이상의 그룹으로 구성되어 있기 때문에 클라이언트 요청에 대한 서비스 처리시간과 처리율이 향상됨을 알 수 있었다.

실험 1, 실험 2에서 StaticHTML, CGI, 멀티미디어에 대해 병목현상이 발생할 때 각각 3:4:2, 3:1:5으로 웹 서버가 재구성 되었다. 이때 특정한 그룹에 대한 컨텐츠 접속 빈도수와 자원 정보를 이용한 부하 분산 정책으로 다른 그룹을 링크 재설정으로 요청들을 분산했을 경우 기본의 방식보다 더 나은 처리시간과 처리율을 볼 수 있었다.

5. 결론 및 향후 연구 과제

사용자의 요청이 특정한 그룹에만 몰리면 해당 그룹에서는 처리량이 증가하여 처리시간도 매우 늦어지게 된다. 이에 대처하기 위해 부하 분산기에서 각각의 웹 서버의 부하 정보를 검사하고 주어진 임계 값 이상의 경우에 컨텐츠 접속 빈도수에 따라 접속 빈도수가 가장 적은 웹 서버를 부하가 많은 네트워크 파일 시스템(NFS)에 링크 재설정(Link Reaction)을 행하였다.

제안된 시스템 구조에 대해 성능 실험을 하였고, 기존의 동일한 컨텐츠를 가지고 있는 방식에 비해 처리율과 처리시간이 기존의 방식보다 50%의 향상됨을 관찰하였다.

본 논문의 실험은 실제 인터넷의 상황에서 이루어진 것이 아니기 때문에 사용자가 실제 느끼는 응답시간의 성능 향상은 미비할 수도 있어 보다 실제적인 환경에서 웹 서버로 동작시켜 실험해야 할 것이다. 또한 실험에서 사용한 컨텐츠에 대한 접속 빈도수를 구하는 알고리즘이나 링크 재설정시 부하에 대한 임계 값 설정 방법에 대해서 계속 연구 중이다.

참 고 문 헌

[1] T. Bray, "Measuring the Web," In Proc. of the Fifth International World Wide Web Conference, pp. 993-1005, Paris, France, May 1996.

[2] A. Woodruff, P. M. Aoki, E. Brewer, P.Gauthier and Lawrence A. Rowe, "An Investigation of Documents from the WWW," In Proc. of the Fifth International WWW Conference, pp. 963-979, Paris, France, May 1996.

[3] V. Cardellini, M. Colajanni and P. S. Yu, "Redirection Algorithms for Load Sharing in Distributed Web_server System," In Proc. of 19th IEEE International Conference on, 31 May-4 June 1999.

[4] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants (Extended Version)," 1996 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 126-137. Pennsylvania, U.S.A., May. 1996.

[5] Kangasharju and J. Ross and K. W., "A clustering structure for reliable multicasting," Computer Communications and Networks, 1999. Proceedings, Eight International Conference, pp.378-383, 1999.

[6] Dongeun Kim and Cheol Ho Park and Deayeon Park, "Request rate adaptive dispatching architecture for scalable Internet server," Cluster Computing, 2000. Proceedings. IEEE International Conference, pp.289-296, 2000.

[7] Canal, R. and Parcerisa, J. M. Conzalez and A., "Dynamic cluster assignment mechanisms," High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium, pp.133-142, 1999.

[8] Linux Virtual Server Project, http://www.linuxvirtualserver.org.

[9] V. Cardellini and M. Colajanni, P. Yu, "Geographic Load Balancing for Scalable Distributed Web Systems," Modeling, Analysis and Simulation of Computer and Telecommunication System, 2000, Proceedings 8th International Symposium on, 2000.

[10] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable Web server clustering technologies," IEEE network, pp. 38-45, May, 2000.

[11] A. Wong and T. Dillon, "Load balancing to Improve Dependability and Performance for Program Objects in Distributed Real-time Cooperation over the Internet," The 3rd IEEE International Symposium on Object-Oriented Real-time Distributed Computing, Mar., 2000.

[12] W. S. Myung and T. M. Chang, "New Web Cluster Technology Based on the Contents on the Web," Proceedings of the International Conference on Internet Computing, IC'03, Vol 2, pp. 655-660, June 2003.

[13] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable Web server clustering technologies," In Proc. IEEE network, Vol.14, no. 3, pp, 38-45, May-June 2000.

[14] E. Casalicchio and M. Colajanni, "A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services," In Proc. of 10th International World Wide Web Conference, May 2001.

[15] M. Colajanni, P. S Yu and V. Cardellini, "Dynamic load balancing in geographically distributed heterogeneous Web-servers," In Proc. of 18th IEEE International. Conference on Distributed Computing Systems, pp.295-302, Amsterdam, The Netherlands, May 1998.

[16] M. Garland, S. Grassia, R. Monroe and S. Puri, "Implementing Distributed Server Groups for the World Wide Web," Tech. Rep. CMU-CS-95-114, Carnegie Mellon University, School of Computer Science, Jan. 1998.

[17] C. S. Yang and M. Y. Luo, "Efficient Support for Content_based Routing in Web Server Clusters," In Proc. of 2th Symposium on Internet Techmologies & Systems, Colorado, USA, October 11-14, 1999.

[18] V. Cardellini, E. Casalicchio, M. Colajanni, M. Mambelli, "Web Switch Support for Differentiated Services," ACM Performance Evaluation Review. Selected from Performance and Achitecture of Web servers Workshop. 2001.

[19] Vivek S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel and E. Nahum, "Locality-Aware Request Distribution in Cluster-based Network Services," ACM 8th, ASPLOS Oct. 1998.

[20] 명원식, 장태무, "웹 서버 클러스터에서 내용 기반으로한 부하분산 기법," 정보처리학회논문지 A, 제10-A권 6호, Vol. 10-A, No 6, pp.729-736, Dec. 2003.

장 태 무

e-mail : jtm@dongguk.edu

1977년 서울대학교 전자공학과 졸업(학사)

1979년 한국과학기술원 전산학과 졸업 (이학석사)

1995년 서울대학교 컴퓨터공학과 졸업 (공학박사)

1979년~1981년 한국전자기술연구소 연구원

1998년 University of Southeastern Louisiana 교환교수

1981년~현재 동국대학교 컴퓨터.멀티미디어공학과 교수

관심분야 : 분산 및 병렬처리, 컴퓨터 구조, 입출력시스템 등

명 원 식

e-mail : wsmyung@dgu.edu

1996년 안양대학교 컴퓨터공학과 졸업(학사)

1998년 안양대학교 전산정보학과 졸업(공학석사)

2004년 동국대학교 컴퓨터공학과 졸업(공학박사)

관심분야 : 분산 및 병렬처리, 클러스터 웹 서버, 컴퓨터 구조, 온라인게임 등

한 준 탁

e-mail : okebary513@donghae.ac.kr

1989년 순천향대학교 전자계산학과(공학사)

1994년 한양대학교 전자계산학과(공학석사)

2003년 동국대학교 컴퓨터공학과(공학박사)

2004년 동해대학교 정보전산소 소장

1995년 3월~현재 동해대학교 컴퓨터공학과 조교수 재직 중

관심분야 : 분산 및 병렬 컴퓨팅, 입출력 시스템, 온라인 게임서버 등