韓國컴퓨터情報學會 論文誌
第16卷 第8號, 2011. 8.

2011-16-8-1-2

분산 웹 클러스터 시스템에서의 효율적인 부하 균등 기법에 관한 연구

이 복 기*

A Study on Efficient Load Balancing Mechanism in Distributed Web Cluster System

Bog-Ki Lee*

요 약

웹 사용자의 급증은 과중한 전송량과 시스템의 부하문제를 야기 시켰으며 이를 해결하기 위한 방안으로 클러스터 시스템이 연구되고 있다. 기존 웹 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 클러스터 시스템의 효율적인 자원사용 방법에 관하여 연구하였다. 메모리 사용을 균형적으로 하는 웹 클러스터 시스템을 구현하기 위한 부하 분산 알고리즘을 제안하여 검토하고 다양한 클러스터링 모델에서 반응시간을 성능 측정의 지수로 제시하였다. 또한 웹 클러스터 시스템의 동시사용자 수를 기반으로 반응시간은 사용자 수를 증가시키면서 실험 하였으며, 성능 실험을 통해 기존의 방식보다 제안한 기법이 처리율과 응답시간에서 보다 향상된 것으로 나타났다.

▶ 키워드 : 웹서버, 부하조절, 웹 트래픽

Abstract

The increasing of web users load to the excessive transmission traffic and system overload problems. To solve these problems, cluster systems are 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, we consider the methodology of efficient resource usage, specially distributed web cluster system. We develope an algorithm that distributes the load on the web

• 제1저자 : 이복기
• 투고일 : 2011. 06. 07, 심사일 : 2011. 07. 22, 게재확정일 : 2011. 08. 16.
* 경원대학교 컴퓨터공학과(Dept. of Computer Science, Kyungwon University)
※ 본 논문은 경원대학교 지원에 의한 결과임.

cluster system to use the system resources, such as system memory equally. The response time is chosen as a performance measure on the various clustering models. And based on the concurrent user to the web cluster system, the response time is also examined as the number of users increases. Simulation experience with this algorithm shows that the response time and average throughput seems to have a good results compare to those with the other algorithm.

▶ keyword : Web Server, Load Balancing, Web traffic

I. 서 론

현재 인기 있는 웹 사이트에서는 매일 수백만의 사용자 요구를 수용하기 위해 다수의 서버를 사용하고 있으며, 하나의 가상 URL을 이용하여 사용자에게 투명성을 제공하는 방식을 일반적으로 채택하고 있다[1][2]. 클러스터링을 이용한 웹 서버 구축 시 중요하게 고려해야 할 사항은 각 웹 서버 노드에 적절한 부하 분산이 이루어지도록 하는 것이며 웹 서버 노드에 결함이 발생할 경우에 대비하여 적절한 서비스 시스템 이주(Migration)가 이루어지게 해야 한다. 문자와 같이 용량이 작은 데이터만 요구하는 패킷을 부하 분산기에서 처리하는 경우는 별 문제 없지만 동영상 파일, MPEG, MP3등의 대용량 멀티미디어 데이터를 요구하는 경우에는 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[3][4]. 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 부하 상태를 실시간으로 정확히 자원을 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.

II. 관련연구

2.1 직접 라우팅 방식

직접 라우팅 방식은 요청을 중계해 주는 서버가 존재하여 클라이언트의 요청(Request)을 서비스 서버에 연결하여 주는 방식이다. 클라이언트의 최초 요청은 중계 서버를 거쳐 서버에 연결이 되지만 요청에 대한 서버의 응답은 클라이언트와 직접 통신하는 특성을 갖는다. 장점으로는 중계 서버의 트래픽 집중화를 막을 수가 있으며, 중계 서버의 다양한 트래픽 분배 정책에 따라 트래픽을 어느 정도 효율적으로 분배할 수 있게 된다. 단점으로는 브로드캐스팅 방식과 같은 이유로 서버의 운영체제에 제약이 있으며, 요청에 대한 응답은 직접 클라이언트와 이루어지기 때문에 서버의 트래픽 처리량을 정확하게 계산할 수 없게 된다[5].

2.2 디스패처 방식

디스패처 방식은 직접 라우팅 방식과 구조적으로 유사하게 웹 클러스터 앞 단에 중계 서버가 존재한다. 그러나 직접 라우팅 방식과는 다르게 모든 클라이언트의 요청과 서버의 응답은 중계 서버를 통한다는 특성을 갖는다. 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[6].

2.3 부하 분배 스케줄링 알고리즘

부하 분배 스케줄링에 이용되는 방법 중 라운드 로빈(Round Robin) 방식은 클러스터로 구성된 모든 웹 서버들이

동등하게 한 번씩 돌아가며 요구에 대하여 서비스하는 방식이다. 이는 각 웹 서버의 서비스 처리 용량과 관계없이 서비스되므로 부하 불균형이 발생할 수 있는 단점이 있다. 이러한 문제점을 고려하여 라운드 로빈 스케줄링과 유사하나 클러스터에 속한 웹 서버들의 처리 용량에 따라 가중치를 둔 뒤 연결하는 가중기반 라운드 로빈 스케줄링 방식이 있다[7].

III. 콘텐츠 기반 웹 서버 클러스터 구조

본 논문에서는 각각의 웹 서버가 주기적으로 자원상태(CPU 사용율과 메모리 사용 율)를 부하 분산기(Load balancer)에게 보고하고 특정 웹 서버에서 부하 량이 임계치(Threshold)를 넘게 될 때 제안한 알고리즘에 의해 선택된 웹 서버로 새로운 요청을 전달하도록 설계되었다. 또한 특정 웹 서버 그룹이 기준 부하 량을 초과할 때 클라이언트 요청을 다른 웹 서버로 전달함으로써 클러스터링 웹 서버의 처리율을 향상시킨다. 그리고 부하를 다른 웹 서버에 분산할 때 웹 서버들의 자원 정보와 콘텐츠에 대한 접속 빈도수를 고려하여 선택된 웹 서버에게로 클라이언트의 요청을 전달함으로써 사용자의 응답시간을 줄이고 처리량에 대한 성능을 향상시킨다.

3.1 동작원리

부하 분산기는 들어오는 클라이언트의 요청을 적절한 스케줄링 알고리즘에 의해 서비스 해주는 실제 웹 서버로 분산시켜주는 역할을 하는 노드를 말하며, 그룹 내의 스케줄링 알고리즘으로 라운드 로빈 방법을 채택하였다. 또한 특정 그룹에서 병목현상이 발생할 경우 이를 제거하고 원활한 서비스를 위해 부하 분산기는 자신이 관리하는 웹 서버들의 부하 상태를 주기적으로 모니터링 할 수 있어야 한다.

웹 서버의 부하를 측정하는 정보는 각 웹 서버의 CPU 사용율과 메모리 사용률이다. 자원 정보를 사용하는 것은 실제 웹 서버의 유효 연결이 멀티미디어 데이터 서비스와 같이 큰 작업일 경우에는 그 수가 많지 않더라도 서비스되는 용량이 크기 때문에 처리시간이 오래 걸리게 된다. 따라서 신규 요청되는 작업은 지연될 가능성이 아주 크다. 즉, 멀티미디어 파일이나 CGI와 같은 동적 콘텐츠(Dynamic contents)를 요구하는 작업들만 서비스하는 웹 서버는 CPU와 메모리가 과부하 될 수 있다는 것이다. 그러므로 부하 분산기는 각 웹 서버 자원 정보(CPU 사용율과 메모리 사용율)를 모니터링 해야 한다. 또한 웹 서버는 가장 큰 비율을 가진 자원 정보 중에서 어느 임계값을 넘게 되면 부하 분산기로 경고 메시지(Alarm Message)를 보낸다. 이는 현재 자신이 서비스할 부하량이 많음을 의미한다. 경고 메시지를 받은 부하 분산기는 웹 서버에 대한 일정량의 웹 로그(Web-Log)를 수집하여 각 웹 서버에 대한 콘텐츠 접속 빈도수를 계산한다. 빈도수를 계산한 값과 각 웹 서버에서 보고한 자원 정보의 값을 계산하여 가장 작은 값을 가진 웹 서버에게 새로운 클라이언트 요청을 전달하게 된다.

이 때 요청 받게 될 웹 서버는 현재 네트워크 파일 시스템과의 커넥션을 해제한 후 부하 량이 많은 네트워크 파일 시스템에 링크를 재설정(Link-Reaction)하여 서비스를 제공한다.

[그림 1]은 특정 그룹(CGI script)에 대한 부하 량(자원정보)이 임계값보다 커짐에 따라 경고 메시지를 발생시킨다.

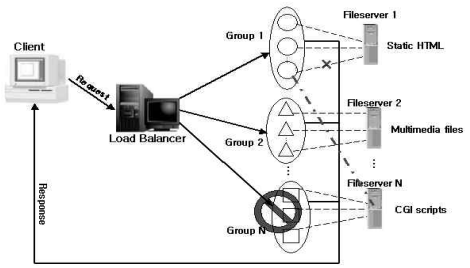

그림 1. 병목현상을 제거하기 위한 링크 재설정
Fig. 1. Link-Reaction for deletion bottle-nek

그 외의 다른 그룹에서 콘텐츠 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 재설정을 통해 병목현상을 해결하는 것을 보여주고 있다. 결국 링크 재설정을 통해 병목현상을 해결함으로써 응답시간을 빠르게 할 수 있고, 처리 시간과 처리율도 향상될 수 있는 것이다.

3.2 제안한 시스템 구조의 프레임워크

[그림 2]는 부하 분산기와 웹 서버들 간의 모듈 관계를 보여 주고 있다.

부하 분산기 내의 ①과 ②의 흐름은 정상적인 동작일 때의 흐름이다. 즉, 웹 서버 중 어느 것도 부하 량에 대한 병목현상이 발생하지 않았을 경우 이다. 여기서 Web_Server Application은 Apache로 구동되며 웹 사이트의 HTTP_데몬이다. 사용자는 HTTP_데몬을 통해 접속할 수 있으며, 응답은 해당 콘텐츠를 가진 웹 서버로부터 받는다.

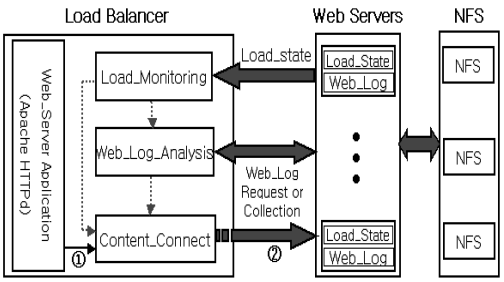

그림 2. 부하 분산기와 웹 서버들 간의 프레임워크
Fig. 2. Frame work load balancer and Web-sever

3.2.1 Load_State 모듈

부하 균등화를 위해 부하 분산기에서는 각 웹 서버의 자원상태를 계속해서 파악하고 계산해야 하므로 그 만큼의 처리 시간이 지연될 수 있다. 따라서 보다 효과적인 해결을 위해 본 논문에서는 각 그룹마다 중계기를 두었다. 중계기는 주기적으로 자기 그룹 내의 웹 서버들에 대한 자원 정보와 경고 메시지를 보고 받는다. 이러한 자원 정보와 경고 메시지는 주기적으로 부하 분산기의 Load_Monitoring 모듈로 전달된다.

여기서 자원 정보를 모니터닝 하는 것과 경고 메시지는 다음과 같이 처리한다.

Statei = Max (CPUi, 메모리i); ……(1)

if (Statei >= THRESHOLD) then Alarm_Messagei = 1; ……(2)

i : 웹 서버의 번호
Statei : 각 웹 서버에 대한 자원 정보의 상태
CPUi : CPU 사용율 (0 ≦ CPUi ≦ 1)
메모리i: 메모리 사용율 (0 ≦ 메모리i ≦ 1)
THRESHOLD : 70% [2]

(1)은 각 웹 서버의 자원 정보 값의 최대값을 중계기에 전달한다. 콘텐츠별로 웹 서버를 특화하여 서비스하기 때문에 정적 콘텐츠(StaticHTML)를 서비스하는 웹 서버와 동적 콘텐츠(CGI, 멀티미디어)를 서비스하는 웹 서버에서 CPU와 메모리의 과부하가 있기 때문이다.

(2)는 경고 메시지를 보낼 수 있는 상태이다. 각 웹 서버의 자원 정보 중 가장 큰 값을 가진 자원이 어떤 임계값과 같거나 클 경우 경고 메시지를 중계기에 보내게 된다. 이는 현 웹 서버에서 부하량이 많을 경우이며, 경고 메시지의 값은 1을 전달하게 된다.

3.2.2 Load_Monitoring 모듈

이 모듈은 각 그룹 내의 중계기로부터 자원 정보의 최대값과 경고 메시지를 주기적으로 보고 받는다.

만약 경고 메시지 값이 1이면 Content_Connect 모듈과 Web_Log_Analysis 모듈에게 현 모듈의 그룹별 웹 서버 자원 정보의 최대값과 경고 메시지를 동시에 전달한다.

3.2.3 Web_Log_Analysis 모듈

이 모듈은 각 그룹에 대한 웹 로그를 수집하여 콘텐츠에 대한 접속 빈도수를 분석하는 모듈이다.

Load_Monitoring 모듈로부터 경고 메시지 1를 받으면, 각 그룹 내의 웹 서버에게 웹 로그를 요청하여 콘텐츠에 대한 접속 빈도수를 분석한다. 접속 빈도수를 구하는 식은 식 (3), (4), (5)와 같다.

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

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

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

Frequency_StaticHTML : StaticHTML 콘텐츠의 접속 빈도수
Frequency_멀티미디어 : 멀티미디어 콘텐츠의 접속 빈도수
Frequency_CGI : CGI 콘텐츠의 접속 빈도수
StaticHTML : 정적 콘텐츠의 대한 로그 수
멀티미디어 : 멀티미디어 콘텐츠의 로그 수
CGI : CGI 콘콘텐츠의 로그 수
n : 그룹 내의 웹 서버 수
N : 제안된 시간동안 액세스 로그 수
Ws : StaticHTML에 대한 가중치
Wm: 멀티미디어에 대한 가중치
Wc : CGI에 대한 가중치

여기서 가중치(Weight)을 주는 것는 각 콘텐츠에 대한 접

속 빈도수를 계산하여도 이 값으로 부하를 정확하게 해결할 수 없기 때문이다.

동일 시간대에 정적 콘텐츠를 처리한 StaticHTML의 접속 빈도수와 멀티미디어와 CGI 같은 동적 콘텐츠를 처리한 접속 빈도수를 같은 조건으로 비교하면 정적 콘텐츠의 접속 빈도수가 높을 확률이 많다.

따라서 이를 해결하고자 각 콘텐츠에 대해 다른 가중치를 적용하여 계산한다.

식 (3), (4), (5)에서 각 그룹에 대한 접속 빈도수를 구한 값은 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 특정 웹 서버에서 병목현상이 발생하여 부하를 분산할 경우에 Load_ Monitoring으로부터 전달 받은 웹 서버의 자원 정보에 대한 정보와 Web_Log_ Analysis로부터 전달 받은 콘텐츠 접속 빈도수에 대한 정보를 가지고 서비스할 웹 서버를 선택하여 클라이언트 요청을 전달하는 역할을 한다. 그리고 선택된 웹 서버를 링크 재설정하는 부분도 이 모듈에서 행해진다.

식 (6)은 신규 선택될 웹 서버를 구하는 식이다.

Selected_Webserver = Min((Frequency_StaticHTML * Statei), (Frequency_멀티미디어 * Statei), (Frequency_CGI * Statei)) …… (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 서비스하기 위해 해당 그룹을 선택해야 하는데, 이때 부하 분산기 서버 내의 인덱스 테이블 요소들을 가지고 해당 그룹을 선택하도록 함을 보여주고 있다.

표 1. 부하 분산기 내의 인덱스 테이블 요소
Table 1. Index table in load balancer

웹 서버	Contents
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 빈도수가 떨어지고 있는 웹 서버를 선택한다. 또한 같은 그룹 내의 웹 서버 중에서 동일한 Min값이 나올 경우는 라운드 로빈(RR) 스케줄링 방식으로 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 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 위한 페이지 뷰(page view)를 측정 단위로 한다.

IV. 실험 및 성능 평가

본 논문에서 제안된 웹 서버 클러스터 시스템 모델의 성능을 측정하기 위해 WOW Linux 7.0 운영체제에서 UC Berkeley 대학의 네트워크 시뮬레이터인 NS-2.1b8a를 사용하여 실험 하였다. 시스템 사양은 Intel 2.66.GHz, RAM 3.25GByte, HDD 300GB로 구성하였다. 제안한 시스템 구조에서의 모든 웹 서버 및 부하 분산기는 동일한 사양을 사용하였다.

4.1 로그 파일 분석

본 논문에서 제안한 내용 기반의 웹 서버 클러스터 시스템을 실험하기 위해 본 대학의 웹 서버 로그 파일을 분석하였다. 웹 로그 파일의 기간은 4일간이었다. 총 요청 수는 64,600여 개로 에러부분을 제외한 콘텐츠만을 추출하였다. 콘텐츠의 분포를 이해하기 위해 실제 서비스된 웹 사이트를 구성하는 콘텐츠의 종류를 조사하였다. 이를 통해 웹 서버에 대

한 사용자의 접근 패턴을 알 수 있다.

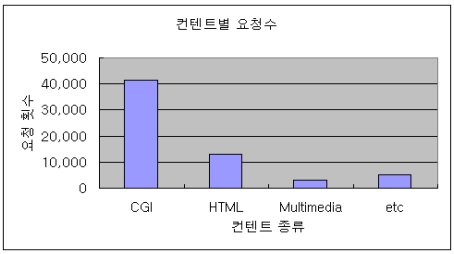

그림 3. 콘텐츠별 요청 회수
Fig. 3. Number of content require

[그림 3]에서 나타난 것과 같이 CGI가 가장 많은 41,344개이고, Multimedia는 3,230개로 요청이 가장 적다. [그림 4]는 콘텐츠별 총 전송된 비율로 분석한 것으로 멀티미디어 데이터의 경우 요청 수는 적으나 전송 비율은 높음을 알 수 있다. 이것은 작은 콘텐츠들을 많이 요청함으로써 네트워크 트래픽이 증가할 수 있지만, 크기가 큰 콘텐츠들은 몇 번만 요청하면 통신량이 증가함을 의미한다.

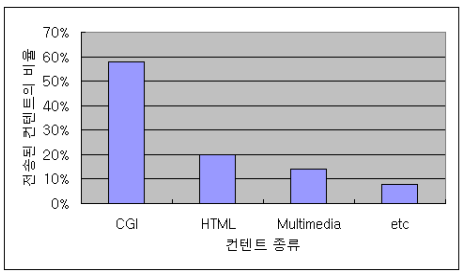

그림 4. 콘텐츠별 총 전송량 비율
Fig. 4. Ratio of content transfer

지금까지의 웹 로그 부하 분석이 다른 웹 서버의 작업 부하와 정확하게 일치한다고 단정할 수는 없다. 다른 인터넷 서버를 이용하는 사용자의 경향이나 웹 서버의 역할에 따라 다르기 때문이다.

4.2 실험 결과 및 평가

기존의 RR 스케줄링 방식과 LC 스케줄링 방식, 그리고 제안한 시스템 구조에 대해 응답시간과 요청 수에 대한 초당 응답 수를 비교 실험하였다. 시뮬레이션 결과로Static HTML, CGI, Multimedia files 각각에 대한 초당 응답 수를 살펴보았다.

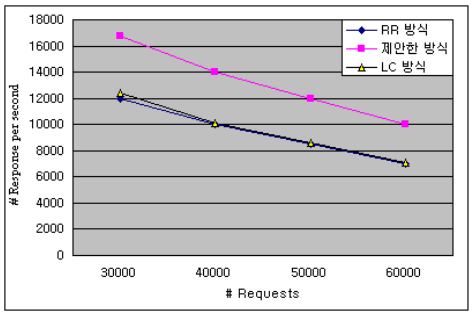

그림 5. Static HTML의 실험 결과
Fig. 5. Test result of static HTML

[그림 5]는 Static HTML 콘텐츠를 가지고 요청 수에 대한 초당 응답 수를 실험한 결과이다. 처음의 30,000개의 요청 수에 대해 RR방식은 12,000개, LC 방식은 12,400개의 초당 응답 수를 보였다. 그러나 제안한 방식은 16,780개의 초당 응답 수를 보임으로써 기존의 방식 보다 더 나은 것으로 나타났다. LC 방식의 경우 부하 분산기 서버가 각 웹 서버에 대한 현재 연결 수를 분석해야 하는 오버 헤드가 추가 되어 그 성능이 떨어지게 된다.

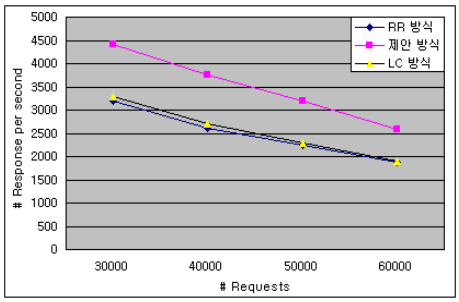

그림 6. CGI의 실험 결과
Fig. 6. Test result of CGI

[그림 6]은 CGI 콘텐트를 가지고 요청 수에 대한 초당 응답 수를 실험한 결과이다. 30,000 개의 요청 수에 따라 RR 방식은 3,200개, LC 방식은 3,290개 그리고 제안한 방식은 4,413개의 초당 응답 수를 보였다. CGI 콘텐츠는 Static HTML 콘텐츠에 비해 아주 적은 응답 수를 볼 수 있는데, 이는 CGI 콘텐츠가 더 많은 실행 시간을 가지기 때문이다. Static HTML 콘텐츠는 실질적으로 문서(text, html 등)를 보여주지만, CGI 콘텐츠는 DB관련 등을 처리하기 때문에 많은 프로세스가 요구된다.

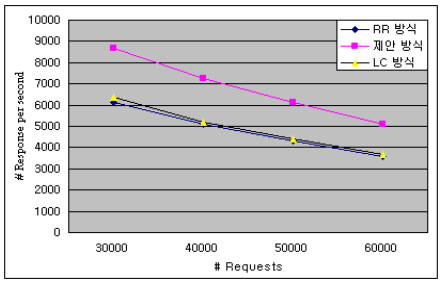

그림 7. Multimedia files의 실험 결과
Fig. 7. Test result of multimedia files

위 실험결과와 같이 기존의 RR 스케줄링 방식과 LC스케줄링 방식은 거의 비슷하게 나타났으나 본 논문에서 제안한 내용 기반으로 한 웹 서버 클러스터 기법이 약 39%의 성능이 향상됨을 알 수 있다.

V. 결 론

웹 서버(실행 서버)간의 서로 다른 콘텐트를 가지고 각 시스템의 자원 정보 기반으로 한 웹 서버 클러스터 기법을 소개하였다. 본 논문에서는 웹 서버마다 서로 다른 콘텐트를 갖는 콘텐트 웹서버 기반으로 한 클러스터 기법을 제안하였다. 이 기법은 성능 실험을 통해 기존 방식보다 약 39%의 성능이 향상되는 것으로 나타났다. 제안한 콘텐츠 기반 웹클러스터 기법을 적용할 경우 기존 솔루션에 비해 다음과 같은 장점이 있다.

첫 번째, 제안한 웹 서버 클러스터 구조는 각각의 웹 서버가 서로 다른 콘텐트를 가지고 있으므로 부하 분산기에 대한 오버헤드가 적어 전체 웹 서버 클러스터 성능을 효율적으로 운용할 수 있다. 두 번째, 사용자의 요청에 대한 응답이 부하 분산기를 거치지 않고 해당 웹 서버에서 바로 사용자에게 전달되기 때문에 응답 시간이 감소된다. 세 번째, 각 웹 서버들은 서로 다른 콘텐트를 가지고 있어 짧은 응답을 요구하는 사용자는 빠르게 응답을 받게 된다.

향후 서버 규모를 증가할 때 더 큰 성능의 향상을 얻도록 개선되어야 한다. 앞으로의 연구 과제로는 웹 서버 클러스터링에 대한 안전성과 신뢰성을 더하고, 또한 링크 재설정시 부하에 대한 임계 값 설정 방법에 대해서 연구도 진행되어야 하겠다.

참고문헌

[1] V. Carellini and M. Cloajanni,and P. Yu, “Red irection Algorithms for Load Sharing in Distributed Web-server Systems,” Proceedings of the 19thIEEE International Conferenceon Distributed Computing Systems, pp. 528-535, May 1999.

[2] Beowulf Project, http://www.beowulf.org

[3] 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 Symposiumon Object-Oriented Real-time Distributed Computing. Mar.2000.

[4] Jaesung Park, Yujin Lim,"A Stochastic Serving MPP Selection Method for Increasing the Efficiency of a Wireless Mesh Network", Journal of The Korea Society of Computer and Information, Vol. 14, No. 6, pp. 83-90, Jun 2009.

[5] Young-Bok Cho, Jae-min Choi, Sang-ho Lee,"An Efficient Dynamic Prediction Clustering Algorithm Using Skyline Queries in Sensor Network Environment", Journal of The Korea Society of Computer and Information, Vol. 13, No. 7, pp. 139-148, Dec 2008.

[6] J. Challenger et al., “Efficiently Serving Dynamic Data at Highly Accessed Web Sites,” IEEE/ACM Trans. on Networking, vol. 12, 2004, pp. 223-233.

[7] L. Chen, Y. Lu, and T.F. Abdelzaher, “Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers,” IEEE Parallel and Distributed Systems, vol. 17, 2006, pp. 1014-1027.

저 자 소 개

이 복 기

2007년 : 아주대학교 컴퓨터공학과 박사 수료

현 재 : 경원대학교 컴퓨터공학과 교수

관심분야 : 병렬처리, 클러스터시스템, 데이터 마이닝

E-mail : bglee@kyungwon.ac.kr