클러스터 웹 서버 상에서 히스토그램 변환을 이용한 내용 기반 부하 분산 기법☆

A Content-Aware Load Balancing Technique Based on Histogram Transformation in a Cluster Web Server

홍 기 호* 권 춘 자** 최 황 규***
Gi Ho Hong Chun Ja Kwon Hwang Kyu Choi

요 약

최근 인터넷 사용자의 기하급수적 증가에 따라 저렴한 가격의 고성능 대용량 클러스터 웹 서버 시스템에 관심이 증대되고 있다. 클러스터 웹 서버 시스템은 저렴한 비용, 높은 확장성과 가용성 등의 장점과 더불어 대규모 사용자에 대한 성능의 극대화를 목적으로 연구 개발되고 있으며, 최근에는 성능 향상을 위한 내용 기반의 부하 분산 기법에 관심이 모아지고 있다. 본 논문에서는 이러한 클러스터 웹 서버 상에서 사용자의 접근 빈도와 파일의 크기를 고려하여 각 서버 노드에 부하를 균등하게 할당하는 새로운 내용 기반의 부하 분산 기법을 제안한다. 제안된 기법은 웹 서버 로그의 각 URL 항목에 해시 함수를 적용하여 얻어지는 해시 값에 그 빈도와 전송된 파일의 크기를 고려한 누적 히스토그램을 생성한다. 사용자 요청은 (해시 값-서버 노드) 매핑에 의한 히스토그램 변환 과정을 통하여 각 서버 노드에 균등하게 할당된다. 제안된 기법은 누적 히스토그램을 주기적으로 갱신함으로써 동적으로 클러스터 웹 서버 시스템의 부하를 고르게 분산시킬 수 있으며, 또한 서버 노드의 캐시를 활용함으로써 전체 클러스터 시스템의 성능을 향상시킬 수 있다. 시뮬레이션을 통한 성능 분석에서 제안된 기법은 전통적인 라운드 로빈 방법보다는 월등히 우수함을 보이고, 기존의 내용 기반 WARD 방법보다는 약 10% 정도의 우수한 성능을 나타낸다.

Abstract

As the Internet users are increasing rapidly, a cluster web server system is attracted by many researchers and Internet service providers. The cluster web server has been developed to efficiently support a larger number of users as well as to provide high scalable and available system. In order to provide the high performance in the cluster web server, efficient load distribution is important, and recently many content-aware request distribution techniques have been proposed. In this paper, we propose a new content-aware load balancing technique that can evenly distribute the workload to each node in the cluster web server. The proposed technique is based on the hash histogram transformation, in which each URL entry of the web log file is hashed, and the access frequency and file size are accumulated as a histogram. Each user request is assigned into a node by mapping of (hashed value-server node) in the histogram transformation. In the proposed technique, the histogram is updated periodically and then the even distribution of user requests can be maintained continuously. In addition to the load balancing, our technique can exploit the cache effect to improve the performance. The simulation results show that the performance of our technique is quite better than that of the traditional round-robin method and we can improve the performance more than 10% compared with the existing workload-aware load balancing(WARD) method.

☞ Keyword : Cluster Web Server, Content-Aware Load Balancing, Histogram Transformation

* 정 회 원 : (주)엘비에스 플러스 연구소 솔루션 개발팀 주임연구원
toeema@hanafos.com(제 1저자)
** 정 회 원 : 강릉영동대학 사이버경찰과 초빙교수
kwoncj@mail.kangwon..ac.kr(공동저자)
*** 정 회 원 : 강원대학교 전기전자정보통신공학부 교수
hkchoi@kangwon.ac.kr(공동저자)
☆ 본 논문은 강원대학교 BK21 사업단과 2004년 정보통신 기초기술연구지원사업(과제번호 04-기초-004) 지원 연구 결과의 일부임.
[2004년/07/20 투고 - 2004/08/13,14,16 심사 - 2004/09/16 심사완료]

1. 서 론

최근 저장장치, 통신, 데이터 압축 기술 등의 발달로 웹(WWW)을 통한 인터넷 사용자의 수는 급격히 증가하고 있으며, 국내외 인기 사이트 경우 다양한 서비스 요구를 만족시키기 위하여 많은 컴퓨팅 자원을 필요로 하게 되었다. 이와 같이 수많은 인터넷 사용자의 요구를 수용하기 위해서는 기존의 고성능 단일 서버의 자원을 업그레이드하거나 추가하는 것으로 해결할 수 있다. 그러나 짧은 기간 폭발적으로 증가하는 서비스 요구를 수용함에 한계가 있으므로 기존의 서버 대신 더욱 성능이 우수한 고성능 서버를 구입하여 대체해야 한다. 이러한 해결책은 점증적으로 증가하는 사용자의 수요를 만족시키기 어렵고, 업그레이드 절차도 복잡하며 비용이 많이 들어가는 단점을 안고 있다.

여러 대의 고성능 PC 또는 워크스테이션을 고속의 네트워크로 결합하여 구성하는 클러스터 웹 서버는 점증적인 확장성과 가격 대 성능이 우수한 웹 서버 시스템을 제공한다. 따라서 클러스터 웹 서버를 사용할 경우 적은 비용으로 서버 성능을 증대시키므로 고성능 단일 서버로 서비스하는 경우보다 사용자 증가에 따른 서비스 부하를 줄이는데 훨씬 효율적이다. 지난 몇 년간 클러스터 서버의 기술은 상업적인 솔루션뿐만 아니라 실험적인 연구 결과에 의한 적용 방법에서도 주목할 만한 성장을 보여 왔다.

클러스터 웹 서버는 서버들 간의 균형적인 부하 분산이 서비스 시간의 단축에 가장 크게 기여한다. 클러스터 웹 서버의 전통적인 부하 분산 기술은 RR-DNS, Dispatching 등의 부하 분산 방법으로 서버 확장성 및 클라이언트 투명성을 제공한다[1-3]. 그러나 최근의 연구에서는 내용 기반 부하 분산(Content-Aware Request Distribution) 방법이 더 우수한 기법으로 대두되고 있다[4]. 기존의 부하 분산 기술은 클러스터 내 서버 노드의 부하 정보(예: CPU 타임, 메모리 크기, 접속의 수 등)만을 이용하는 반면에, 내용 기반 부하 분산 방법은 내용(contents) 타입이나 서비스에 따라 클라이언트 요구를 분배한다. 즉, 클라이언트 특정 내용 또는 서비스 요구를 처리할 특정 서버 노드를 선택하는 특화된 서비스에 의한 서버 분할이 가능하게 함으로써 전체적인 서버 시스템의 성능 향상 및 유연성을 제공한다.

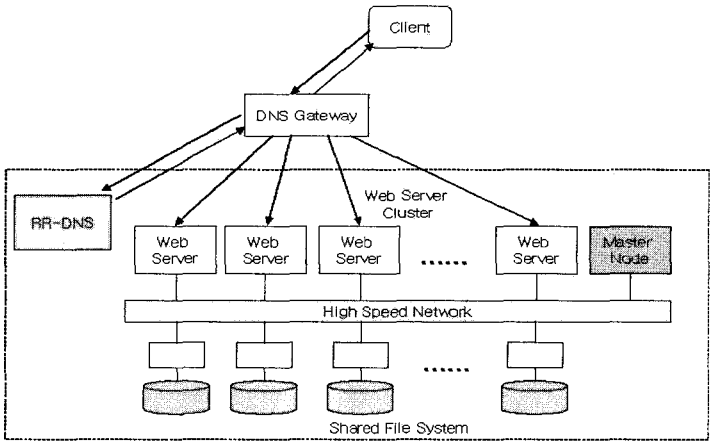
본 논문에서는 새로운 내용 기반 부하 분산 방법으로서 히스토그램 변환을 이용한 부하 분산 기법을 제안한다. 제안하는 기법은 웹 서버 로그의 URL 항목에 해시(hash) 함수를 적용하여 얻어지는 해시 값에 빈도와 전송된 바이트 항목에 대한 파일 크기를 누적하여 히스토그램을 생성한다. 생성된 히스토그램의 누적 분포에 히스토그램 변환 함수를 적용하여 얻어지는 각 서버 노드의 해시 값에 따라 부하를 균등하게 할당할 수 있다. 히스토그램 변환을 이용한 부하 분산 기법에 의해 얻어지는 [해시 값-서버 노드]의 설정을 주기적으로 갱신함으로써, 클러스터 웹 서버 시스템의 부하를 동적으로 고르게 분산할 수 있으며, 또한 각 서버 노드의 캐시를 효율적으로 활용함으로써 전체 클러스터 시스템의 성능을 향상시킬 수 있다.

본 논문은 먼저 2장에서 클러스터 웹 서버 시스템의 구조와 내용 기반 부하 분산에 대하여 기술하며, 3장에서는 본 논문에서 제한하는 히스토그램 변환을 이용한 부하 분산 기법에 대하여 설명한다. 또한 제안된 기법의 성능 평가를 위하여 시뮬레이션 모델과 성능 분석 결과를 4장에서 기술하고, 마지막으로 5장에서 결론을 맺는다.

2. 클러스터 웹 서버 시스템과 부하 분산

2.1 클러스터 시스템 구조

본 논문에서 제안하는 부하 분산 기법을 위한 클러스터 웹 서버 시스템의 구조와 환경은 그림 1과 같다. 클라이언트의 요청이 DNS Gateway에 도착하면 RR-DNS에 의해 클라이언트 요청을 라운드 로빈 방법에 의해 서비스를 수행할 웹 서



〈그림 1〉 클러스터 시스템의 구조 및 환경

버 노드를 결정한다. RR-DNS에 의해 수신된 정보를 바탕으로 DNS Gateway는 네트워크상의 트래픽을 통제하고 클라이언트 요청을 해당 웹 서버 노드에 라우팅(routing)하여 할당한다. 클라이언트 요청을 할당받은 웹 서버는 서비스를 수행할 결과를 DNS Gateway를 통해 수신 받을 클라이언트의 IP 주소를 찾아 요청 결과를 전송한다. 또한 웹 서버 클러스터 내에는 여러 대의 웹 서버 노드들과 하나의 마스터 노드(Master Node)가 고속의 네트워크에 의해 연결되어 있다. 그리고 클러스터 내 각 웹 서버 노드들은 자신의 로컬 디스크를 가지며, 두 개의 논리적인 타입으로 나뉜다. 논리적으로 front-end와 back-end로 나누어 관리를 하며, front-end는 HTTP 서버로 공유 파일 시스템을 통해 back-end로부터 파일을 읽어와 서비스를 하게 된다.

2.2 내용 기반 부하 분산 기법

라운드 로빈(Round-Robin : RR)과 같은 전통적인 부하 분산 방법은 순수하게 서버 노드간의 부하 정보를 바탕으로 서버 노드에 클라이언트 요청을 할당한다. 이와 달리 클라이언트 요청을 처리할 서버 노드를 선택하기 위하여 요청의 타입이나 서비스에 따라 부하를 분산하는 내용 기반 부하 분산 방법을 CARD(Content-Aware Request Distribution) 라 한다[1,5]. 이러한 내용 기반 부하 분산의 장점은 첫째, 서버 노드의 메인 메모리 캐시의 적중 율(Cache Hit Rates) 향상으로 클러스터 시스템의 성능이 향상된다. 둘째, 서버 노드의 분할을 통해 2차 저장장치의 확장성을 향상시킬 수 있다. 셋째, 요청의 타입(예, 오디오, 비디오 등)에 따라 특정한 서버 노드를 배치 운영할 수 있다[5].

Locality-Aware Request Distribution(LARD)[7]은 위의 첫 번째 장점에 초점을 맞춘 내용 기반 부하 분산 방법이다. 중앙 집중적인 방식의 서버 구조에서 front-end는 모든 클라이언트의 요청을 TCP hand-off 메커니즘[6]을 통해 back-end 노드에 데이터를 전달하는 역할을 한다. 또한 front-end는 모든 클라이언트와 back-end 노드의 연결을 관리하므로, 동적으로 변화하는 back-end 노드의 부하 정보를 고려하여 클라이언트 요청을 서비스할 노드에 할당한다. 즉, 최근에 처리한 노드 중 부하가 가장 적은 노드에 요청을 할당한다. 전통적인 부하 분산 방법의 RR 등과 달리 전체의 작업문서가 아닌 부분적인 작업문서를 메인 메모리에 캐시 하므로 캐시의 적중 율이 높아지는 것은 자명한 사실이며, LARD는 캐시의 지역성을 효과적으로 활용하여 전체 시스템의 성능 향상을

가져온다. LARD가 캐시의 지역성과 노드간의 부하 분산이 우수한 방법이나, 모든 요청을 중앙의 front-end에서 back-end로 포워딩하므로 병목 현상이 발생하기 쉽다. 또한 front-end의 고장에 의해 전체 클러스터 시스템이 실패(failure)하는 원인이 된다.

FLEX[8]는 웹 호스팅 서비스를 위한 부하 분산 방법으로, 웹 서버 로그를 수집하고 트래픽 패턴을 분석하여 서버 노드에 웹 사이트를 균등하게 할당한다. 주기적으로 DNS 서버의 [사이트-서버] 할당에 대한 설정만을 갱신함으로써 추가적인 비용 없이 부하 분산이 가능하다.

Workload-Aware Request Distribution(WARD)[9]는 사용자의 액세스 패턴과 작업부하 특성을 파악하기 위해 웹 서버 로그를 분석한다. 분석된 로그에 의해 가장 빈번하게 접근하는 파일 셋의 일부분인 코어(core)를 클러스터 내 모든 서버 노드에 중복 배치한다. 그리고 파일의 나머지는 클러스터 램에 분할 배치한다. 즉, WARD는 웹 서버 로그를 분석하여 작업부하 특성에 따라 파일을 배치하는 알고리즘으로 서버간의 포워딩을 위한 TCP hand-off와 디스크 접근 오버헤드를 줄이기 위해 서버 노드의 램에 의존한다.

이와 같은 기존의 내용 기반 부하 분산 방법은 요청을 서비스할 서버를 선택하기 위해 요청에 포함된 타겟 URL을 읽고, 이전에 생성된 클라이언트와의 연결정보를 유지하거나 미리 설정된 테이블로부터 검색하여야 한다. 따라서 URL 테이블을 관리하거나 검색하기 위해 상당한 오버헤드가 소요된다. 또한 서버 노드간의 메시지 교환에 의해 부하 정보를 공유하므로 이에 따른 오버헤드도 뒤따르게 된다.

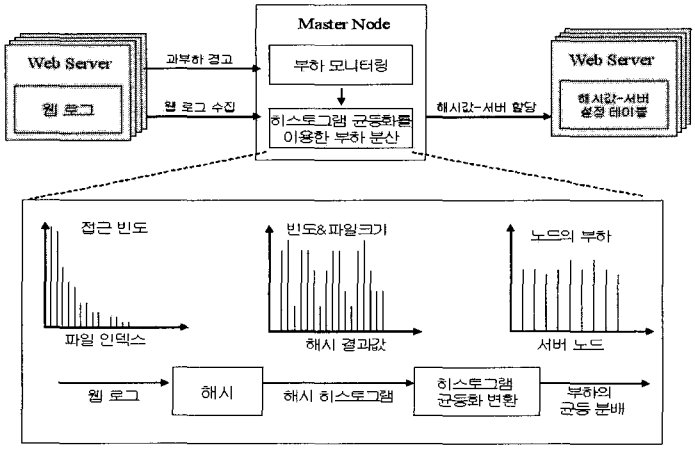
3. 히스토그램 변환을 이용한 부하 분산 기법

클라이언트 요청의 내용 타입 또는 서비스에 따라 클러스터 서버 노드에 요청을 분배하는 내용 기반 클러스터 웹 서버 시스템이 기존의 클러스터 웹 서버 시스템보다 웹 서버의 캐시 메모리를 더욱 효율적으로 이용할 수 있다. 본 논문에서도 단순히 서버의 접속 수와 같은 부하정보에 의존하지 않고 요청 파일의 접근 빈도와 파일 크기를 기반으로 부하를 분산시킨다. 즉, 내용 기반 부하 분산 방법의 장점인 서버 노드의 캐시를 효율적으로 활용하고, 해시 함수와 히스토그램 변환과정을 통해 서비스할 노드를 결정하는 새로운 부하 분산 기법을 제안한다. 본 장에서는 히스토그램 변환을 이용한 부하 분산 기법을 각 단계별로 기술하고, 히스토그램의 분포가 편재하는 경우의 확장된 부하 분산 기법과 서버의 성능이 다른 이종의 클러스터 환경에서의 부하 분산 기법에 대하여 기술한다.

3.1 히스토그램 변환 기법

본 논문에서 제안한 내용 기반 부하 분산 기법은 그림 1과 같은 클러스터 시스템 구조 하에서 이루어지며, 그림 2는 제안한 히스토그램 변환을 이용한 부하 분산 기법에 대한 전체적인 프레임워크를 나타낸다. 클러스터 웹 서버 노드들의 설정 테이블 갱신은 마스터 노드(Master Node)에서 수행된다. 마스터 노드는 각 웹 서버의 부하 상태를 모니터 하고 각 웹 서버로부터 로그를 수집한다. 수집한 웹 로그의 분석은 히스토그램 변환 과정을 적용하며, 새로운 부하 할당 테이블로 각 웹 서버의 설정을 갱신하여 준다. 웹 로그에 없는 새로운 URL의 요청도 기존의 로그에 있는 URL들과 동일한 해시 함수를 적용함으로써 결과 해시 값을 구하며, 이 값에 따라 이 URL은 특정 노드에 할당된다.

클라이언트 요청이 주어졌을 때 트래픽의 흐름은 다음과 같이 처리된다. 먼저, RR-DNS에 의해 클라이언트의 요청을 서비스할 웹 서버 노드를 결정한다. 마스터 노드는 결정된 웹 서버 노드의 요청을 읽어 웹 로그에 따른 타겟(target) URL에 해시 함수를 적용한다. 여기에 사용되는 파라미터



〈그림 2〉 히스토그램 변환을 이용한 부하 분산 프레임 워크

는 URL의 접근빈도와 파일 크기가 되며, 해시 함수의 결과인 해시 값에 히스토그램 변환 함수를 적용하면 서비스할 서버 노드가 결정된다. 이에 따른 결과인 [해시 값-서버 노드]는 실제 서비스를 수행할 웹 서버 노드의 아이디가 된다. 또한 과부하 상황이 발생하거나 일정 주기마다(24시간 단위) 미리 웹 로그를 수집하여 제안기법을 적용한다. 서버 노드 아이디가 현재 요청을 할당받은 웹 서버 노드인 경우는 로컬에서 처리가 되며, 다른 서버 노드인 경우는 해당 서버 노드로 포워딩된다. 각 웹 서버 노드는 요청을 처리하여 응답을 DNS Gateway를 통하여 클라이언트에게 전송한다. 본 논문에서 제안한 내용 기반 부하 분산 기법은 그림 3과 같이 웹 로그 분석을 통해 얻어지는 부하 정보에 기반하여 서비스할 파일을 해시 값에 따라 각 서버 노드에 균등하게 할당하게 된다.

전체 프레임 워크를 다시 설명하면, 주기적으로(24시간 단위) 웹 로그를 수집하고 수집된 로그에 해시 함수를 적용하여 파일에 대한 접근 빈도와 파일 크기를 누적하여 해시 히스토그램을 생성한다. 이 히스토그램에 히스토그램 변환 함수를 적용하여 클러스터 웹 서버 노드의 부하를 균등하게 분산되도록 한다. 주기적으로 또는 각 서버의 과부하 임계 값(threshold value)에 의해 과부하 상황에 놓인 웹 서버 노드로부터 과부하 경고 피드백을 받았을 때, 즉 동적으로 변하는 사용자의 요청 패턴에 따라 균등하게 부하 분산을 수행하게 된다. 주어진 요청을 서비스할 서버 노드에 할당시키거나 부하를 균등하게 할당하도록 수행하는 방법이 매우 빠르고 간결하다. 따라서 부하 분산을 위해 메시지 교환이나 요청의 파싱과 검색 및 URL 정보 테이블을 유지하고 관리하는 등의 오버헤드를 줄일 수 있다. 본 논문에서 제안한 히스토그램 변환 함수를 이용한 내용 기반 부하 분산 기법을 단계별로 설명한다.

단계 1: 웹 로그 수집 및 분석

웹 로그 파일은 사용자가 해당 웹 서버에 접근하여 웹 서비스를 요구하였을 때 접근한 파일 및 객체에 대하여 서버와 사용자간의 정보가 시간 순으로 기록되는 파일이다. 웹 로그는 사용자가 요청한 파일과 파일에 포함된 여러 객체들에 대하여 하나의 레코드로 기록된다. 하나의 레코드에는 여러 항목을 포함하고 있으며, 로그 형식에 따라 다음과 같은 항목을 포함한다.

- 소스 IP 주소
- 날짜 및 시간
- HTTP 메소드
- 요청한 URL

· 응답 코드

· 전송된 파일크기

하나의 로그 엔트리에서 소스 IP 주소는 요청 패킷을 보낸 클라이언트 주소이고, 파일에 접근한 날짜 및 시간 항목을 포함한다. 그리고 'GET', 'POST'와 같은 HTTP 메소드, 응답이 성공적으로 이루어 졌는지를 나타내는 응답 코드와 요청한 파일의 경로를 포함한 URL(또는 파일명)을 가지고 성공적으로 전송이 이루어졌을 경우 전송된 파일 크기가 포함된다. 그림 3은 본 논문의 시뮬레이션에 사용된 실제 웹 로그의 일부분이다. 웹 로그로부터 해시 히스토그램을 생성하기 위해 성공적으로 응답한 로그 레코드의 URL 항목과 클라이언트로 전송된 파일크기 항목을 추출한다.

단계 2: 해시 히스토그램 생성

추출한 각 URL 항목에 해시 함수를 적용하여 얻어지는 해시 값에 전송된 파일 크기를 누적한다. 파일 크기에 따라 서버의 프로세싱 시간이 다르므로 각 파일의 접근 빈도에 파일 크기를 가중치로 적용하여 부하가 랜덤하게 해시 범위 내에 분포하도록 히스토그램을 생성한다.

하루에 수백만의 요청을 받는 웹 서버의 로그양은 방대할 수밖에 없다. WARD 알고리즘도 이러한 웹 로그를 수집 분석하는 작업에 오랜 시간을 소비한다[9]. 반면 제안하는 기법은 전체 웹 로그를 한번만 검색하여 해시 히스토그램을 생성하게 되므로 처리시간이 줄어든다.

단계 3: 히스토그램 변환

웹 로그의 해시 히스토그램에 따른 파일의 접근 빈도와 크기에 대한 부하를 각 서버 노드에 분할하기 위해 히스토그램 변환(Histogram Transformation : HT) 기법을 사용한다. 먼저, 히스토그램 변환 기법을 위해 웹 로그에 대한 해시 히스토그램의 확률 분포를 구한다. 히스토그램의 확률 분포에 누적 분포 함수(Cumulative Distribution Function)를 적용하여 히스토그램의 확률 누적 분포를 구한다. 각 해시 입력 값 $\overline{x}$에 대한 URL의 접근빈도와 파일크기를 누적한 확률 분포 $H_{\overline{x}}(m)$를 구한다. 각 해시 값에 따른 확률 누적 분포로부터 전체 클러스터 서버 노드에 균등하게 부하를 할당하기 위해 다음과 같은 히스토그램 변환 함수를 사용한다.

$$y = \lceil P \sum_{m=0}^{x} H_{\overline{x}}(m) \rceil \quad (1)$$

여기서, $\lceil\ \rceil$는 상한(ceiling) 함수이고, P는 크기 벡터로써 서버 노드가 동일한 성능을 가진다고 가정하므로 서버 노드의 개수가 된다. 해시 히스토그램으로부터 해시 범위 $(0 \le m \le x)$에 히스토그램 변환 함수를 적용하여 해시 값에 대응되는 서버 노드 y를 구한다. 같은 서버 노드 y의 해시 값에 따른 확률 분포 값(파일 접근빈도와 크기 누적) 들을 더하여 각 서버 노드의 부하 정보가 되어 전체 클러스터 서버 노드의 각 서버 노드에 균등하게 분산되도록 한다. 또한 웹 로그에 없는 최초의 사용자의 요청 URL이거나 일정시간이 흐른 후의 웹 로그나 해시 함수에 적용시킨다. 즉 모든 요청 URL에 대해 해시 함수를 적용하여 해시 값에 따라 바로 서비스할 노드가 결정되어진다. 이와 같이, 히스토그램 변환을 이용한 내용 기

```
199.72.81.55 - - [01/Jul/1995:00:00:59 -0400] "GET /history/ HTTP/1.0" 200 1382
port26.annex2.nwlink.com - - [01/Jul/1995:00:01:02 -0400] "GET /software/winvn/winvn.html HTTP/1.0" 200 9867
port26.annex2.nwlink.com - - [01/Jul/1995:00:01:04 -0400] "GET /software/winvn/winvn.gif HTTP/1.0" 200 25218
port26.annex2.nwlink.com - - [01/Jul/1995:00:01:04 -0400] "GET /images/construct.gif HTTP/1.0" 200 1414
port26.annex2.nwlink.com - - [01/Jul/1995:00:01:04 -0400] "GET /software/winvn/bluemarb.gif HTTP/1.0" 200 4441
dd14-012.compuserve.com - - [01/Jul/1995:00:01:05 -0400] "GET /shuttle/technology/images/srb_16-small.gif HTTP/1.
205.189.154.54 - - [01/Jul/1995:00:01:06 -0400] "GET /cgi-bin/imagemap/countdown?99,176 HTTP/1.0" 302 110
```

〈그림 3〉 웹 로그의 예

반 부하 분산 기법에 의하여 얻어진 [해시 값-서버 노드] 테이블은 주기적으로(24시간 간격) 또는 부하 모니터링으로 빠르게 갱신함으로써 유동적인 사용자의 접근 패턴에 따라 부하를 각 서버에 균등하게 할당하므로 더욱 효율적이 된다.

3.2 히스토그램 변환 기법의 확장

히스토그램 변환을 이용한 부하 분산 기법이 해시 히스토그램 분포가 크게 편재되지 않은 경우 매우 효율적이다. 따라서 해시 히스토그램의 결과 분포가 편재되는 경우와 클러스터 내 서버의 성능이 서로 다른 경우에 대하여 제안 기법을 확장한다.

3.2.1 편재된 히스토그램 분포를 위한 부하 분산 기법

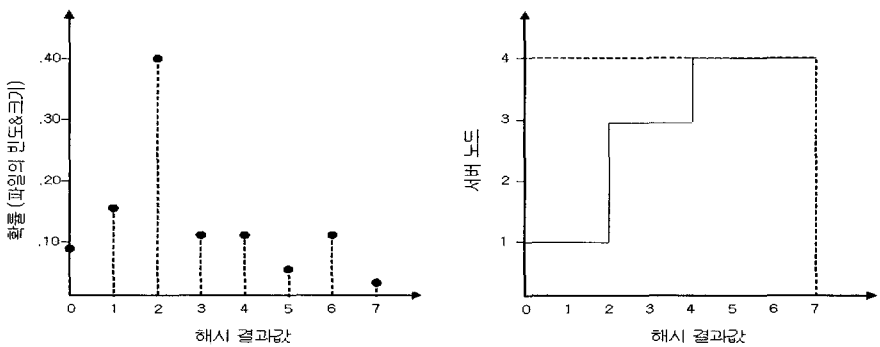
히스토그램 변환 기법을 해시 함수에 의한 히스토그램 분포가 급하게 기울어지는 경우(heavy skew case)로 확장한다. 웹 로그의 해시에 의해 생성된 히스토그램 분포에 영향을 미치는 요인은 여러 가지가 있다. 그 요인으로는 특정 웹 파일에 대하여 접근 빈도가 현저하게 많은 경우, 또는 접근 빈도에 따른 가중치로 작용되는 파일 크기가 비교적 큰 경우 발생할 수 있다. 그러므로 히스토그램의 분포가 하나의 해시 결과 값에 몰리는 경우가 생길 수 있다. 일반적으로 서버 노드의 개수보다 큰 해시 값의 범위를 작게 하였을 경우 더욱 발생 확률이 높아진다.

편재된 해시 히스토그램에 히스토그램 변환 함수를 적용한 예를 보자. 웹 로그의 해시 결과 값에 따른 히스토그램 분포는 그림 4(a)와 같고, 히스토그램의 해시 결과 값은 표 1과 같다. 여기서 "파일의 빈도 & 크기 누적(확률)"은 웹 로그에 생성된 각 파일의 요청 빈도수와 각 파일의 크기를 곱하여 나온 값을 각 파일의 요청 빈도수와 각 파일의 크기를 곱하여 나온 값 전체를 누적한 값으로 나눈 확률 값을 의미한다. 4개의 서버 노드에 대하여 부하를 균등하게 할당하고자 한다면, 식 (1)의 히스토그램 변환 함수를 적용하여야 한다.

〈표 1〉 편재된 히스토그램의 해시 결과 값

해시 결과 값	파일의 빈도&크기 누적(확률)
0	0.08
1	0.15
2	0.40
3	0.10
4	0.10
5	0.05
6	0.10
7	0.02

$$y_0 = \lceil 4\times 0.08 \rceil = \lceil 0.32 \rceil = 1,$$
$$y_1 = \lceil 4\times 0.23 \rceil = \lceil 0.92 \rceil = 1,$$
$$y_2 = \lceil 4\times 0.63 \rceil = \lceil 2.52 \rceil = 3,$$
$$y_3 = \lceil 4\times 0.73 \rceil = \lceil 2.92 \rceil = 3,$$



(a) 해시 히스토그램 (b) 히스토그램 균등화 변환

〈그림 4〉 편재된 경우의 히스토그램 변환을 이용한 부하 분산 기법의 확장

$y_4 = \lceil 4 \times 0.83 \rceil = \lceil 3.32 \rceil = 4,$

$y_5 = 4,\ y_6 = 4,\ y_7 = 4$

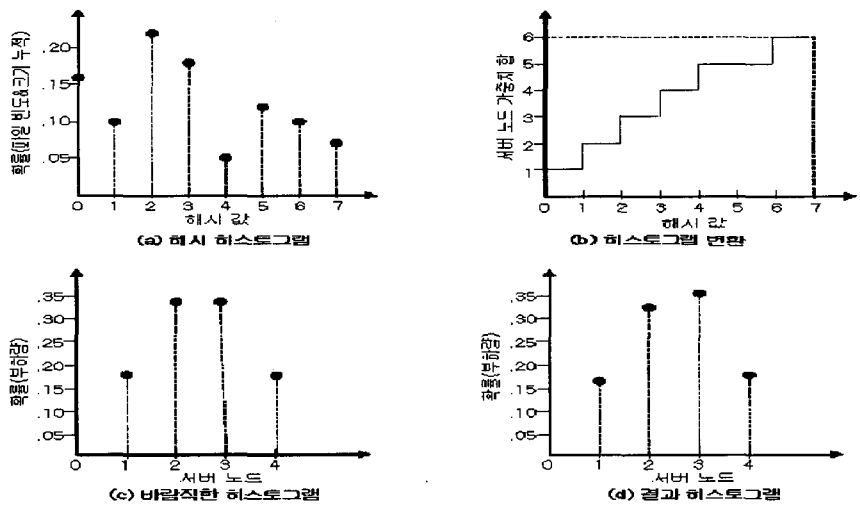
와 같이 서버 노드 경계 값이 1, 3, 4로 추출된다. 서버 노드 2는 경계 값에서 제외되고, 노드 3에 부하가 0.5로 몰리게 된다. 이러한 경우 서버 노드 2, 3을 그룹으로 묶어 부하를 배정하게 된다. 결과적으로 그림 4(b)와 같이 서버 노드 1에는 해시 값 0~1의 부하 0.23이 할당되고, 그룹으로 묶여진 서버 노드 2, 3에는 해시 값 2~3의 부하 0.5가 할당되고, 마지막으로 서버 노드 4에는 해시 값 4~7의 부하 0.27이 할당된다. 그룹으로 묶여진 서버 노드들은 그룹 사이에 라운드 로빈에 의해 클라이언트 요청을 할당하도록 설정을 변경하여 준다. 따라서 해시 히스토그램의 결과가 편재되는 경우 특정 서버 노드에 부하가 집중되는 것을 분산 시킬 수 있도록 확장하여 적용한다.

3.2.2 이종의 서버 환경에서의 부하 분산 기법

서버 노드의 성능이 다른 클러스터 환경에서 효율적인 부하 분산이 필요하다. 클러스터 시스템은 사용자의 수요 증가에 따라 손쉽게 서버를 추가하거나 교체함으로써 점진적인 확장성을 제공한다. 점진적인 확장에 의해 새로 추가된 서버의 성능은 기존의 서버 성능에 비하여 우수할 것이며, 이러한 시스템의 확장으로 발생하는 서버 노드간의 성능차이를 고려한 부하 분산 방법이 필요하게 된다. 히스토그램 변환을 이용한 내용 기반 부하 분산 기법은 서버 노드의 성능이 동일한 경우에 대하여 부하를 균등하게 분배함으로써 성능 향상을 가져온다. 그러나 서버의 성능이 다른 경우 다른 서버 노드에 비하여 성능이 우수한 서버 노드의 자원을 효율적으로 사용할 수 없다. 따라서 이종의 서버 환경에서 서버의 성능에 따라 가중치를 부여하고, 부하를 가중치에 따라 서버에 할당하도록 한다. 즉, 본 논문에서 제안한 부하 분산 기법의 3단계 중 히스토그램 변환 단계에서 히스토그램 변환 함수를 확장하여 적용한다.

성능이 다른 서버 환경을 위한 확장된 히스토그램 변환을 이용한 부하 분산 기법은 서버 성능에 따라 서버의 가중치를 정수 값으로 부여한다. CPU 처리속도, 메모리 량을 고려하여 히스토그램 변환 함수의 서버 노드의 개수 P를 각 서버 노드의 가중치의 합으로 대치한다. 즉,

$$y = \lceil P_w \sum_{m=0}^{x} H_{x(m)} \rceil \qquad (2)$$



〈그림 5〉 이종의 서버 환경을 위한 제안 기법의 확장

여기서, P_w는 각 서버 노드의 가중치 합이 된다. 그러므로 실제 서버 노드에 할당되어지는 해시 결과 값은 서버 성능의 가중치에 따라 고려되므로 클러스터 웹 서버 시스템의 확장시 더욱 효율적인 부하 분산이 가능하게 된다.

그림 5는 성능이 다른 4개의 서버 노드에 대하여 가중치를 1, 2, 2, 1을 갖는다고 가정할 때의 각 단계와 식 (2)를 적용한 결과 그래프이다. 서버 노드의 가중치의 합은 $P_w = 1+2+2+1=6$이 되고, 그림 5(a)의 각 해시 값에 히스토그램 변환함수를 적용하여 각 서버 노드의 가중치에 따라 부하를 할당하면 그림 5(d)와 같이 나온다. 그림 5(c)의 바람직한 히스토그램의 부하 분산과 그림 5(d)의 결과 히스토그램과 비교 결과 오차의 범위가 거의 없는 것으로 나타남을 알 수 있다.

4. 성능 평가

본 논문에서 제안한 히스토그램 변환을 이용한 부하 분산 기법의 성능 평가를 위한 시뮬레이션을 수행하였다. 시뮬레이션을 위한 웹 로그는 임의 로그와 실제 로그에 대해 전통적인 부하 방법인 RR과 웹 로그 분석에 의한 부하 분산 방법인 WARD와의 비교 분석이 이루어진다.

4.1 시뮬레이션 모델

웹 서버의 작업부하(Workload) 특성은 웹 로그의 분석을 통해 연구되어 왔다[13]. 웹 로그 분석에 의한 연구에서 대부분 웹 서버의 작업부하는 파일의 접근 빈도와 크기에 밀접한 관계를 가지고 있으며, 일부분의 파일에 대하여 웹 서버 요청의 많은 부분을 차지한다. 즉, 웹 서버의 파일 모두가 동일한 요청 빈도를 가지는 것이 아니라, 10% 정도의 파일에 대하여 웹 서버가 받은 요청의 80~95%의 접근 빈도를 보인다. 본 논문에서는 웹 서버의 작업부하 특성으로 접근 빈도와 파일크기를 고려하여 생성한 임의로그와 실제 웹 로그를 대상으로 시뮬레이션을 수행한다. 시뮬레이션에 사용된 임의 로그 생성 방법을 기술하고, 또한 실제 웹 로그 특성에 대하여 기술한다.

1) 로그 모델

임의 로그 생성을 위하여 Zipf 분포를 이용한다. 웹 서버의 N개 파일 중 i번째 문서에 대한 접근 빈도 확률 P_i는 다음 식으로 나타내진다[10].

$$p_i = \frac{c}{i^{(1-\theta)}}, \quad 1 \le i \le N, \tag{3}$$

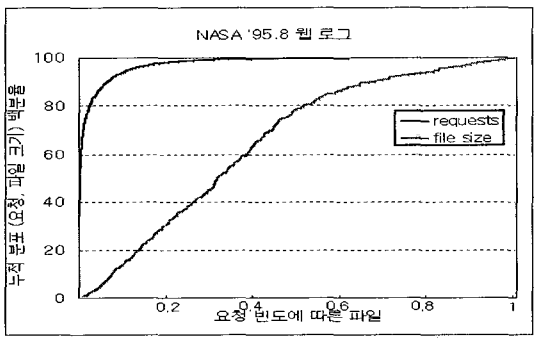
$$c = \frac{1}{H_N^{(1-\theta)}}, \quad H_N^{(1-\theta)} = \sum_{i=0}^{N} \frac{c}{i^{(1-\theta)}}$$

식 (3)에서 가장 기울어진 경우(heavy skew)의 Zipf 분포 상수는 $\Theta = 0.0$으로 클라이언트들의 파일에 대한 요청빈도가 급격하게 몰리는 경우의 로그를 생성하며, $\Theta = 0.4$는 요청빈도가 급격함의 정도가 다소 둔화된 경우(mild skew)에 대해 로그를 생성한다. 또한 클라이언트들의 파일에 대한 요청빈도가 균일한 경우(normal skew)의 $\Theta = 1.0$에 대하여 임의 로그를 생성한다. 식 (3)의 c는 임의 URL의 인덱스 값이며, $H_N^{(1-\Theta)}$는 N개 파일에 대한 조화수(Harmonic Number)로 N개 파일에서 i번째 문서까지의 Zipf 분포 상수를 적용한 값이다. 접근 빈도와 파일 크기와의 관계는 4개의 클래스로 구분하며, 각 클래스의 파일 크기와 접근 빈도는 표 2와 같다[10].

〈표 2〉 접근 빈도와 파일 크기

클래스	파일크기	접근빈도
클래스 0	100 ~ 900 Byte	35%
클래스 1	1 ~ 9 KB	50%
클래스 2	10 ~ 90 KB	14%
클래스 3	100 ~ 900 KB	1%

실제 웹 로그는 NASA 항공 우주 연구소의 웹 서버에 대한 한달 분량의 웹 로그를 사용한다[11]. 실제 로그 파일의 크기는 160MB이고, 이중 성공적으로 응답을 보낸 경우, 즉, 응답코드가 200인 로그 엔트리만을 추출하였다. 추출한 로그 파일의 크기는 140MB이다. NASA 웹 로그의 파일 개수는 6,561개로 파일의 전체 크기는 223MB이다. 평균 파일 크기는 34.1KB이고, 전체 요청 수는 139만으로 요청에 대한 평균 전송량은 19.8KB이다. 시뮬레이션의 부하특성 파라미터 요소를 알아보기 위해 NASA의 웹 로그를 파일에 대한 요청 빈도에 따른 파일크기를 누적 분포로 분석해 보았다. 요청수와 파일크기에 대한 부하량을 총 요청수에 대해 각 파일에 대한 요청수를 구해 전체파일로 누적해 나간 것을 백분율로 구하고, 전체파일 크기에 대해서도 각 파일크기에서 전체파일로 누적해 나간 것을 백분율로 구해 확률분포 그래프로 나타냈다. 그림 6에서와 같이 요청빈도 율에 따라 파일크기가 비례함을 알 수 있었다. 즉, 부하특성으로 파일의 요청수인 접근빈도와 파일크기가 영향이 있음을 알 수 있었다. 따라서 시뮬레이션에 접근빈도와 파일 크기를 파라미터 요소로 사용하였다.



〈그림 6〉 이종의 서버 환경을 위한 제안 기법의 확장

2) 시뮬레이션 환경

시뮬레이션은 VC++와 SMPL을 이용하여 수행하였다. 클러스터 시스템은 그림 1과 같이 RR-DNS 서버와 웹 서버 노드들로 구성된다. 웹 서버 노드는 CPU와 로컬 디스크를 가지며 각 로컬 디스크는 고속의 네트워크에 의해 공유되어진다. 웹 서버 노드들이 디스크에 있는 파일을 읽어올 때 네트워크 오버헤드는 동일하다고 가정하며, 각 노드 자신의 메인 메모리를 이용해 캐시 정책을 수행한다. 시뮬레이션을 단순하게 하기 위해 캐싱은 파일 단위로 수행된다고 가정한다. 각 서버 노드의 대기 큐에 라운드 로빈(RR) 방법으로 클라이언트 요청이 도착하면 RR-DNS 서버에 의해 각 서버 노드에 동일한 개수의 요청이 할당되며, 다음 세 가지 방법에 따라 요청을 분배한다.

첫째, 전통적인 방법의 RR은 DNS 서버에 의해 할당받은 요청을 모두 로컬에서 서비스한다.

둘째, 웹 로그 분석에 의한 방법의 WARD는 로그 분석에 의한 URL 테이블에 따라 core에 대 한 요청은 로컬에서 서비스하고, 다른 노드로의 분배의 경우는 해당 노드로 포워딩(hand off)하게 되고, 그 외에 디스크에 있는 것은 로컬 노드에서 서비스 된다.

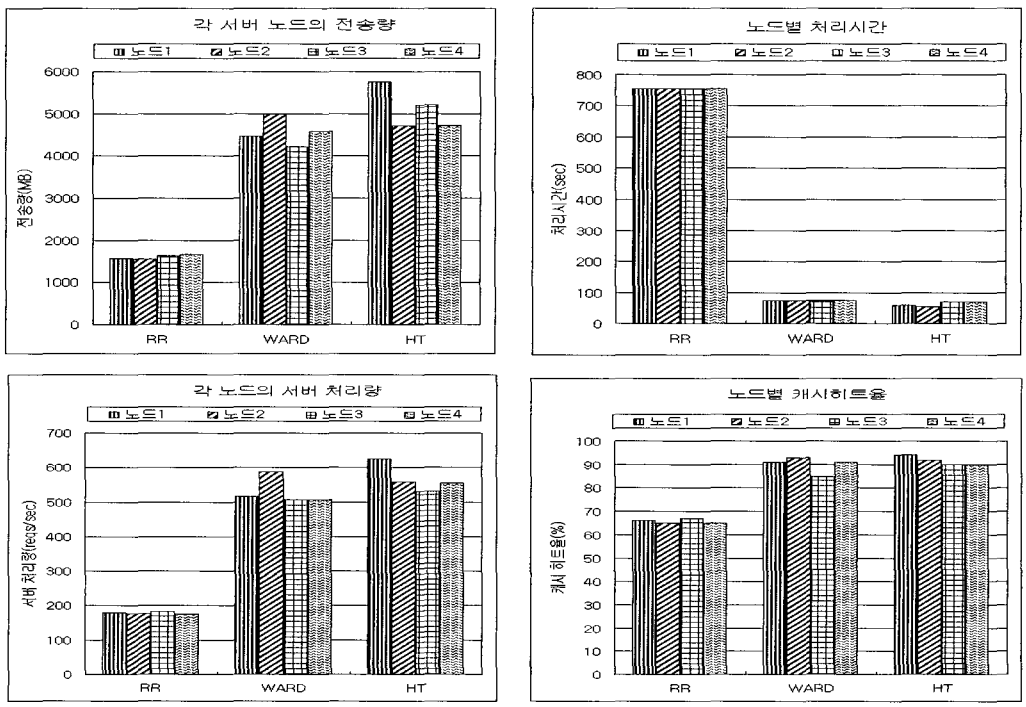
셋째, 히스토그램 변환 기법은 요청된 URL 항목을 해시 함수에 적용하여 URL-해시 테이블 에 따라 로컬 노드인 경우는 로컬에서 서비스가 이루어지고, 원격의 다른 노드인 경우 는 해당 노드로 포워딩이 이루어진다.

시뮬레이션을 위한 파라미터는 WARD[9]를 참조하였다. 포워딩을 위한 오버헤드는 WARD에서 사용한 가정치로 CARD의 실험 값(약 300 μ sec) 에 근거한 수치인 276 μ sec 를 사용하였다. 이 수치는 본 논문에서 제안한 기법과 WARD 방법에 동일하게 적용하였다. RR 방법과 히스토그램 변환 기법과는 달리, WARD 방법에서는 하나의 요청을 파싱하고 URL 테이블을 검색하여 라우팅

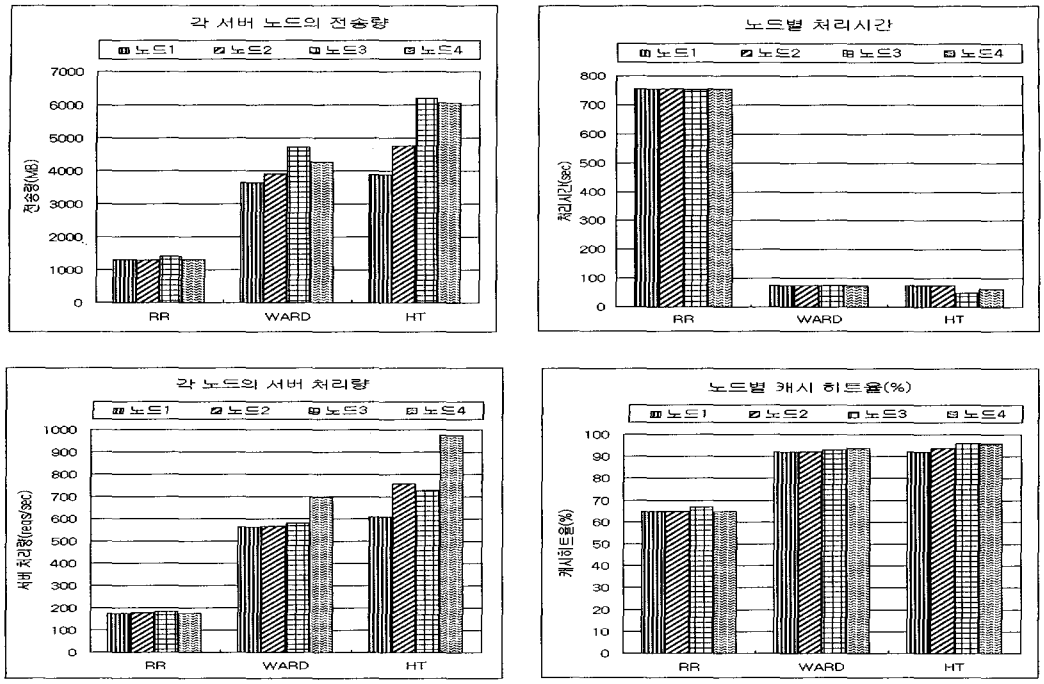
을 결정하는데 따른 오버헤드가 필요하다. 이 오버헤드 값은 200μsec로 설정하였다. 서버 노드에서 요청을 처리하기 위한 비용으로 평균 파일 크기가 14.5KB일 때 메모리에 캐시된 파일의 요청을 처리하는 경우는 초당 900개 정도의 요청을 서비스할 수 있으며, 디스크로 파일을 검색하는 경우는 초당 60개의 요청을 처리할 수 있다고 가정한다. 파일의 서비스 시간은 파일크기에 비례한다고 가정한다. 또한 서버 노드의 캐시 정책은 일반적인 LRU(Least Recently Used)를 사용한다. 시뮬레이션은 위의 세 가지 방법을 적용하였을 때의 서버 노드의 평균 처리량, 평균 전송량, 처리시간 등을 측정한다.

4.2 내용 기반 부하 분산 기법의 성능 평가 결과

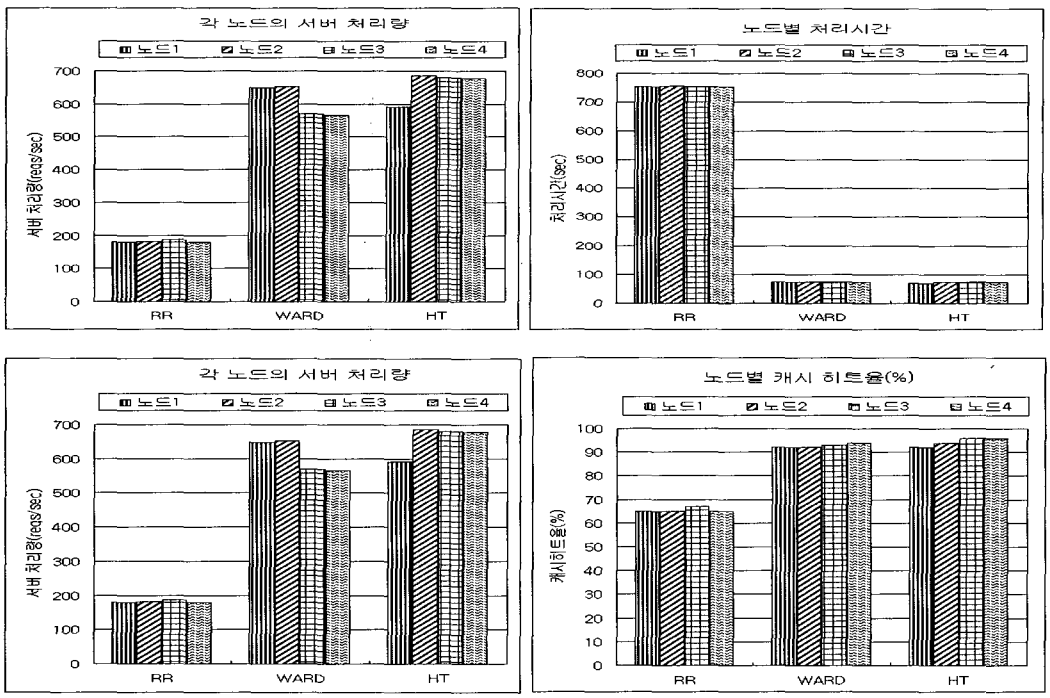
클러스터 웹 서버의 부하 분산을 위해 RR, WARD, 히스토그램 변환(HT) 기법을 각각 적용하여 시뮬레이션 함으로써 비교 분석한다. 먼저, 임의 생성한 동일한 파일 셋에 대하여 Zipf 분포 상수 값(Θ)을 다르게 하여 파일에 대한 접근 패턴에 변화를 주어 세 가지 방법에 의한 부하 분산 성능을 측정하였다. 시뮬레이션에 사용된 임의 로그의 작업부하 특성은 파일 셋의 크기가 10,000개이고, 로그 엔트리의 수는 50만으로 평균 파일 크기는 13.45KB, 요청에 대한 평균 전송량은 10.15KB이다. Zipf 분포 상수는 가장 heavy skew한 경우를 0.0, mild skew한 경우 0.4, normal skew한 경우 1.0에 대하여 4개 서버 노드를 대상으로 시뮬레이션 하였다. 단, 서버 노드의 메모리는 32MB로 동일한 크기를 갖는다고 가정한다. 각각의 Θ에 따라 임의 생성한 로그를 세 가지 기법(RR, WARD, HT)에 동일하게 적용하여 전체 로그를 처리한 후, 서버 노드 당 전송량(Transferred Bytes), 처리시간(Processing Time), 처리량(Server Throughput), 캐시 히트 율(Cache Hit



〈그림 7〉 (a) Zipf 상수 0.0의 임의 로그에 대한 RR, WARD, HT 기법의 적용



〈그림 7〉 (b) Zipf 상수 0.4의 임의 로그에 대한 RR, WARD, HT 기법의 적용 결과

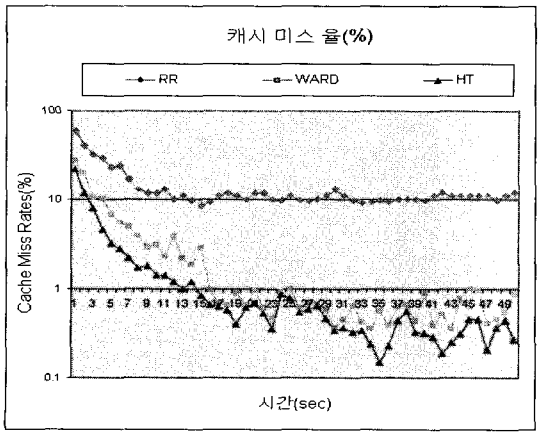


〈그림 7〉 (c) Zipf 상수 1.0의 임의 로그에 대한 RR, WARD, HT 기법의 적용 결과

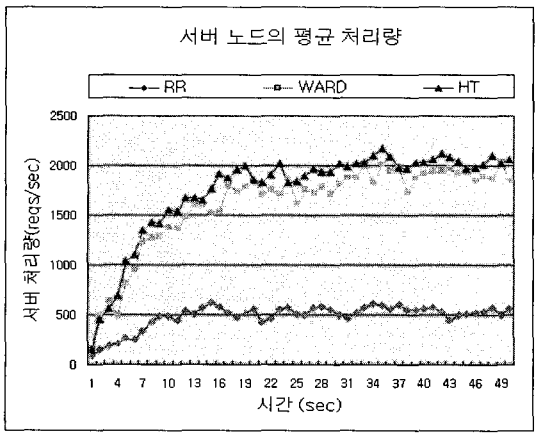
Rates)을 측정하였다.

그림 7에서 (a)는 Zipf 상수가 0.0일 때, (b)는 0.4일 때, (c)는 1.0일 때를 각각 나타낸다. 본 논문에서 제안한 해시 값과 히스토그램 변환 함수를 이용한 부하 분산 기법은 작업부하 특성으로 파일의 접근 빈도와 파일 크기를 고려한다. 서버의 부하 요인은 서버에 할당되는 요청의 수와 서버에서 요청을 처리하는데 소비되는 시간이 된다. 따라서 시뮬레이션을 통해 서버 당 전송한 전체 파일 크기를 비교한다. 그림 7에서 보는 바와 같이 RR, WARD, HT 방법 중 가장 고른 전송량을 보이는 것은 RR 방법이다. RR 방법은 RR-DNS에 의하여 서버에 일정하게 할당되는 요청을 포워딩 없이 지역적으로 처리하기 때문이다. 그러나 일반적으로 서버 노드가 전체 파일 셋에 대해 서비스하므로 캐시 히트 율이 감소하여 전체 파일 중 접근 빈도가 높은 일부 파일 셋만을 서비스하는 내용 기반 부하 분산 기법보다 전송량뿐만 아니라 처리시간도 높으며 서버의 처리량도 크게 떨어진다. 제안된 HT 기법은 WARD 부하 분산 방법과 같이 서버의 디스크에 의존하지 않고 파일을 캐시하여 서비스하므로 처리시간과 처리량에서 RR과 큰 차이를 보인다. 즉, HT 기법의 경우 하나의 서버 노드뿐만 아니라 그룹핑 되어진 서버 노드에도 캐시가 되어 캐시 히트 율이 높기 때문에 처리시간이 적게 소비되고, 각 서버의 단위 시간당 처리량도 크게 증가하는 것을 볼 수 있다. Zipf 상수 값의 변화에 따라 생성된 로그에 대하여 사용자의 접근 패턴의 변화에 RR 기법은 전혀 부응하지 못하여 그림 7의 (a), (b), (c)에서 큰 변화 없이 거의 일정한 결과를 보여준다. 반면에 가장 사용자가 몰리는 경우인 그림 7(a)에 대해 제안된 HT 기법은 RR 방법에 비해 노드별 전송량과 처리량이 약 60% 높게 나왔으며, 캐시 히트 율은 20% 이상 높았고 처리시간도 70% 이상 빠르게 나왔다. 또한 WARD 방법에 비해 노드별 전송량과 처리량 및 캐시 히트 율이 약 10% 정도 높게 나왔으며, 처리시간도 1%정도 빠르게 나왔다. 이와 같이 HT 기법의 경우 사용자의 접근 패턴의 변화에 따라 서버 노드가 빠르게 대응하는 것을 볼 수 있다. 특히, Zipf 상수 값의 변화에 의해 생성된 임의 로그에 대하여 처리량과 처리시간 및 전송량의 결과는 해시 함수의 특성으로 빠르고 간결한 방법의 결과로 볼 수 있으며, 해시 함수의 범위 값의 오차를 최소로 하는 경우 더 좋은 성능을 가져온다.

그림 8, 9, 10은 NASA 웹 로그에 대한 시뮬레이션으로 RR, WARD, HT 기법을 각각 적용하여 서버의 성능을 평가한다. 서버 노드의 개수는 8개로 모든 서버 노드는 32MB 메인 메모리를



〈그림 8〉 서버 노드들의 평균 캐시 미스 율

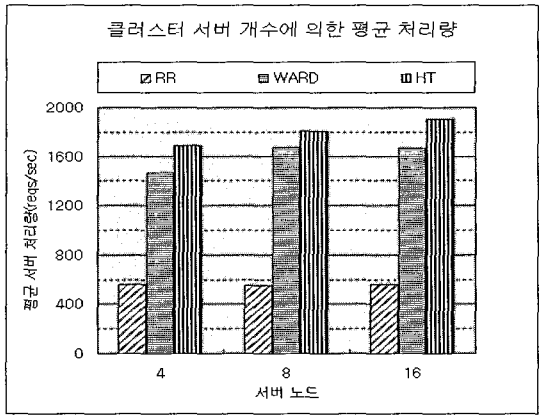


〈그림 9〉 서버 노드들의 평균 처리량

갖는다고 가정한다. 또한 NASA 웹 로그는 6561개의 파일로 전체 문서의 크기가 223MB인 비교적 적은 규모이다. 그림 10은 50초 동안 일정시간 간격으로 서버 노드들의 평균 캐시 미스 율(Cache Miss Rates)을 측정하였다. RR 방법은 전체 문서에 대하여 요청을 처리하므로 WARD, 제안된 HT 기법보다 상대적으로 높은 평균 13%의 캐시 미스 율을 나타냈고, HT 기법은 평균 1.6%의 캐시 미스 율로 WARD는 평균 2.6%의 캐시 미스 율을 나타냈다. WARD 방법이 캐시의 일부분에 대하여 파일을 중복하여 캐시 하므로 RR에 비해서는 훨씬 낮지만, HT기법에 비해서는 평균 1% 높게 나왔다. 그림 11은 50초 동안 일정시간 간격으로 서버 노드들의 평균 처리량을 측정하였다. 시간이 지날수록 RR 방법에 비해 제안된 HT 기법의 서버 처리량이 월등함을 알 수 있다. 특히 HT 기법은 WARD 방법보다 최대 10% 이상 좋은 서버 처리 결과 값을 얻은 것으로 나타났다.

그림 8과 9를 비교하여 볼 때, 캐시 미스 율이 높은 RR 방법은 서버 노드의 처리량이 작게 나타났다. 캐시 미스로 인해 디스크 접근 빈도가 높아지기 때문이다. 가장 접근 빈도가 높고 파일 크기가 작은 파일들을 코어(core)로 설정하기 때문에 WARD 방법은 포워딩을 줄이고, 캐시를 효과적으로 사용하여 디스크 접근 시간의 오버헤드를 줄임으로써 좋은 성능을 발휘한다. 그러나 이전에 생성된 클라이언트와의 연결정보를 유지하거나 미리 설정된 테이블로부터 검색하여야 한다. 이러한 URL 테이블을 관리하거나 검색하기 위한 오버헤드가 소요된다. 또한 서버 노드간의 메시지 교환에 의해 부하 정보를 공유하므로 이에 따른 오버헤드도 뒤따르게 된다. 따라서 요청을 서비스할 서버를 선택하기 위한 오버헤드를 줄이기 위해 해시 값과 히스토그램 균등화 함수를 이용한 HT 기법의 매핑 테이블 방식이 효율적이다. 내용 기반으로 하는 HT 기법은 바로 서버 노드로의 부하 분산이 가능하므로 빠르고 균등하게 부하를 분산시킬 수 있다. 본 논문에서는 내용기반 부하 분산 기법인 LARD 방법은 웹 로그를 이용하지 않으므로 비교대상으로 삼지 않았다.

그림 10은 서버 노드의 개수를 변화시키면서 RR, WARD, 제안된 HT 기법에 대한 서버 노드의 평균 서버 처리량을 측정하였다. RR 방법은 노드 개수를 증가시켜도 서버의 처리량이 크게 증가하지 않고 일정한 것을 볼 수 있고, WARD와 HT 기법은 비교할 수 없을 정도로 우수한 것을 볼 수 있다. 특히 HT 기법은 간결한 해시 함수에 의해 히스토그램 변환을 적용하여 요청을 할당함으로써 요청의 파싱에 대한 오버헤드를 상대적으로 무시할 수 있으므로, 요청의 파싱과 라우팅 결정을 위한 오버헤드를 적용한 WARD와 비교하여 약 10% 정도의 성능 향상을 보였다.



〈그림 10〉 클러스터 서버 개수에 따른 평균 서버 처리량

5. 결 론

본 논문은 클러스터 웹 서버 상에서 서버 노드의 캐시 효율을 향상시키는 방법으로 해시 값과 히스토그램 변환 함수를 이용하는 내용 기반 부하 분산 기법을 제안하였다. 동적 부하 분산을 위해 주기적으로 또는 과부하 발생 시에 웹 로그를 수집하여 분석한다. 하루에도 몇 백만의 요청을 받는 웹 서버의 방대한 로그를 수집하여 간결하게

각 로그 엔트리의 URL 항목에 해시 함수를 적용하여 얻은 해시 결과 값과 파일 접근 빈도와 크기 값을 히스토그램으로 생성할 수 있다. 히스토그램을 균등화하기 위해 각 해시 값에 대해 파일의 접근 빈도와 파일 크기에 대한 확률 값을 누적하여 가며 히스토그램 변환 함수를 적용한다. 같은 노드 결과 값끼리 각 해시 값에 해당하는 확률(부하 정보) 값을 더해 각 서버 노드에 균등하게 할당한다. 이와 같은 웹 로그에 대한 간결한 분석 절차는 변화하는 요청의 패턴에 따라 동적으로 대응할 수 있게 해준다.

내용 기반 부하 분산을 위해서는 어떤 요청에 대해 서비스할 서버 노드를 선택하기 위해 타겟 문서에 대한 정보를 가지고 있게 된다. 따라서 클라이언트로부터 요청이 들어왔을 때, 요청을 파싱하여 타겟 문서에 대한 정보를 바탕으로 부하 분산이 이루어진다. 이러한 정보들을 공유하여 요청을 파싱 하게 되는데, 제안한 기법은 해시 함수를 이용함으로써 메시지 교환이나 검색에 대한 오버헤드를 줄일 수 있다.

본 논문은 디스크 액세스 타임을 줄이기 위해 서버 노드의 메모리 캐시를 적극 활용하며, 성능 평가를 통해 해시 함수에 의한 간결한 부하 분산을 행하므로 전통적인 방법인 RR과 기존의 내용 기반 부하 분산 방법인 WARD에 비해 전체적인 서버 노드의 성능이 향상되는 것을 확인하였다. 또한, 히스토그램 변환을 이용한 내용 기반 부하 분산 기법은 점진적인 확장성에 의해서 발생하는 이종의 서버 환경에서도 좋은 성능을 보인다. 기존의 내용 기반 부하 분산 방법인 WARD는 서버 노드의 메모리 사용에 최적의 성능을 보이지만, 서버의 성능 차이에 따른 이종의 서버 환경은 고려하지 않고 있다. HT 기법은 정적인 문서에 초점을 두고 웹 서비스에서 증가하고 있는 동적인 문서(CGI 스크립트, ASP, JSP)에 대해 고려하지 않았으나, 시뮬레이션 결과 웹 로그를 이용한 우수한 내용 기반 부하 분산 방법인 WARD보다 약 10%의 성능 향상을 보임을 증명하였다.

참 고 문 헌

[1] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic Load Balancing on Web Server Systems," IEEE Internet Computing, Vol.3, No.3, pp. 28-39, 1999.

[2] T. Schroeder, S. Goddard, and B. Ramamurthy, "Scalable Web server Clustering Technologies," IEEE Network, pp. 38-45, May-June 2000.

[3] IBM Corporation, "The IBM Interactive Network Dispatcher", http://www.ics.raleigh.ibm.com/netdispatch, 1998.

[4] L. Cherkasova, "Optimizing a Content-Aware Load Balancing Strategy for Shared Web Hosting Service," In Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems(MASCOTS2000), San Francisco, California, August/September 2000.

[5] M. Aron, D. Sanders, P Druschel, and W. Zwaenepoel, "Scalable Content-aware Request Distribution in Cluster-based Network Servers," In Proceedings of USENIX2000 Technical Conference, June 2000.

[6] A. Cohen, S. Rangarajan, and H. Slye, "On the Performance of TCP Splicing for URL-Aware Redirection," In Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO, Oct. 1999.

[7] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-Aware Request Distribution in Cluster-based Network

Servers," In Proceedings of the 8th Conference on Architectural Support for Programming Languages and Operating System, San Jose, CA, Oct. 1998.

[8] L. Cherkasova, "FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service," In Proceedings of the Fifth International Symposium on Computers and Communications(ISCC00), pp. 8-13, July, 2000.

[9] L. Cherkasova and M. Karlsson, "Scalable Web Server Cluster Design with Workload-Aware Request Distribution Strategy WARD," In Proceedings of the 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information System, June, 2001.

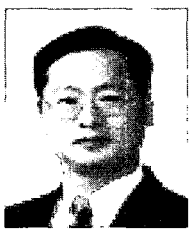
[10] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants," In Proceedings of the ACM SIGMETRICS '96 Conference, Philadelphia, PA, Apr. 1996.

[11] T. T. Kwan, R. E. McGrath, and D. A. Reed, "NCSA's World Wide Web server: Design and Performance, Computer," Vol. 28, No. 11, pp. 68-74, Nov. 1995.

◐ 저 자 소 개 ◑

홍 기 호 (Hong Gi Ho)
2000년 2월 강원대학교 컴퓨터공학과 졸업(학사)
2002년 2월 강원대학교 대학원 컴퓨터정보통신공학과 졸업(석사)
2002년 3월~2004년 7월 (주)씨엘씨소프트 연구/개발팀 주임연구원
2004년 9월~현재 (주)엘비에스 플러스 연구소 솔루션 개발팀 주임연구원
관심분야 : 멀티미디어 시스템, 데이터베이스 시스템 etc.
E-mail : toeema@hanafos.com

권 춘 자 (Kwon Chun JA)
1986년 2월 한양대학교 전자공학과 졸업(학사)
1991년 2월 한양대학교 대학원 전자계산학과 졸업(석사)
1991년 4월~1994년 4월 한국과학기술정보연구원 데이터베이스운영실 연구원
2000년 3월~현재 강원대학교 대학원 컴퓨터정보통신공학과 박사과정
2002년 3월~현재 한림성심대학 컴퓨터정보과 초빙교수
관심분야 : 멀티미디어 VOD 시스템, 데이터베이스 시스템, 클러스터 웹 서버 시스템 etc.
E-mail : kwoncj@mail.kangwon..ac.kr



최 황 규 (Choi hwang Kyu)
1984년 2월 경북대학교 전자공학과 졸업(학사)
1986년 2월 한국과학기술원 전기및전자공학과 졸업(석사)
1989년 8월 한국과학기술원 전기및전자공학과 졸업(박사)
1994년 7월~1995년 7월 Univ. of Florida Database R&D Center 방문교수
1999년 3월~2001년 2월 강원대학교 전자계산소 소장
2002년 7월~2003년 8월 Univ. of Minnesota 방문교수
1990년 3월~현재 강원대학교 전기전자정보통신공학부 교수
관심분야 : 멀티미디어 시스템, 데이터베이스 시스템, Intelligent Storage System etc.
E-mail : hkchoi@kangwon.ac.kr