트래픽 관리를 위한 부본서버 성능평가 및 분배 알고리즘

한 정 혜[†] · 이 경 희[††]

요 약

최근 들어 다수의 클라이언트에게 컨텐츠를 안정적으로 서비스하기 위하여 여러 부본서버를 운영한다. 이 부본서버 중 시공간적으로 다양한 환경으로부터 요청된 클라이언트의 요구를 가장 효과적으로 처리할 수 있는 서버를 선택하기 위한 서버 선택 알고리즘이 여러 분야에서 활발히 연구되고 있다. 본 논문은 비교적 큰 용량의 컨텐츠를 서비스 하는 부본서버 간의 하드웨어 성능 차이가 큰 경우에 QoS를 향상시킬 수 있는 방법을 제안하였다. 또, 부본서버의 하드웨어 성능평가 알고리즘과 HTTP 응답시간의 평균과 편차를 동시에 고려한 알고리즘을 제안하였다. 그리고 시뮬레이션을 통하여 일반적인 경우에 가장 성능이 우수한 것으로 알려진 HTTP 반응시간 평균과 편차에 근거한 부본선택 알고리즘이 비교적 큰 사이즈의 컨텐츠 서비스할 경우와 부본 서버들간의 하드웨어 성능차가 큰 경우에 더 나은 성능을 보인다는 것을 확인하였다.

Evaluating and Distributing Algorithms based on Capacities of Duplicated Servers for Traffic Management

Jeong-Hye Han[†] · Kyung-Hee Lee[††]

ABSTRACT

Most of the existing algorithms try to disseminate the multimedia contents of internet service provider (ISP), without taking into account characteristics and capacities of duplicated servers. However, they are less reliable without prior information on capacities of duplicated servers. In this paper we propose two algorithms, performance rating algorithm of hardware and capacity algorithm, inspired by the need of improving QoS of delivering multimedia contents without incurring long access delays when the capacities of duplicated servers are significantly different and clients locate in a fixed geographical domain. Our simulation results show that they are better than HTTP response time algorithm when the multimedia contents are large and quite different from performances between duplicated servers.

키워드 : 부본서버(Duplicated server), 서버선택 알고리즘(Server selection algorithm), QoS(Quality of Service), 서버성능평가(Capacity of server)

1. 서 론

모든 정보가 인터넷기반으로 교환되고 있다고 해도 과언이 아닐 만큼 인터넷은 정보교환의 중요한 수단으로 자리잡았다. 또한 인터넷 트래픽은 TCP, UDP 및 기타 프로토콜용 트래픽으로 분류되며 이들을 이용하는 응용 프로토콜 중 HTTP용 트래픽이 전체 트래픽의 70%로 가장 많은 부분을 차지하고 기타 DNS, SMTP, FTP, NNTP 등의 트래픽은 5%이내의 값을 보이는 것으로 나타났다[2]. 인터넷 서비스가 활발해짐에 따라 네트워크의 물리적인 성능이 빠르게 향상되고 있는 것과 마찬가지로 보다 효과적인 정보전달을 위하여 데이터 용량 또한 증가하고 있다.

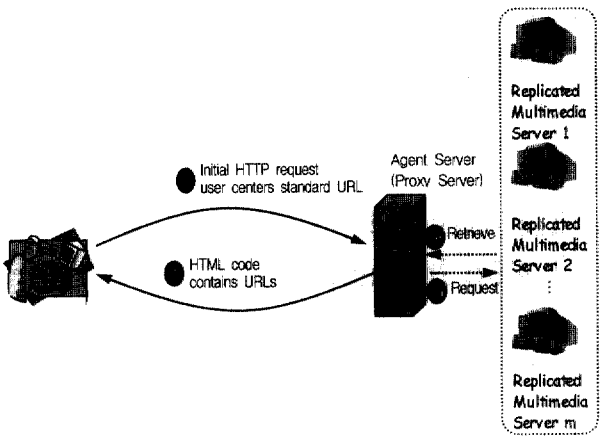
다양한 연구내용 중에서 오래전부터 채택되었으며, 많은 ISP(Internet Service Provider)들이 네트워크의 기반구조나 프로토콜 자체의 변경 없이 또는 변경을 최소화하면서 좋은 서비스 성능을 낼 수 있는 방법으로 선택한 것은 같은 서비스를 수행할 수 있는 서버인 부본서버(Duplicated Server)를 이용하는 것이다. 아래 (그림 1)과 같이 하나의 서버에서 모든 클라이언트의 요청을 처리하는 것이 아니라 클라이언트 요청을 가까운 부본서버로 분산시켜 서비스를 제공하도록 하는 방법이다. 이미 여러 웹 사이트에서 서비스 성능과 신뢰도 향상을 위하여 웹 문서를 분산시키고, 부본서버를 통해 서비스하는 방법을 사용하고 있다[4].

부본서버를 이용하는 방법도 크게 두 가지로 나뉘는데 하나는 서버를 지역적으로 분산시켜 요청한 클라이언트와 거리상 가까운 서버를 선정하여 서비스를 제공하는 방법이 있고, 다른 하나는 같은 장소에 여러 개의 서버를 두고 들어온 요청을 적절히 나누어 처리함으로써 좋은 QoS(Quality of Service)를 제공하는 방법이다.

그런데 현재 대부분의 ISP사들이 QoS를 요구하는 경우

※ 이 논문은 2001년도 한국학술진흥재단의 지원에 의해서 연구되었음 (2001-003-E00261).
† 종신회원 : 청주교육대학교 컴퓨터 교육과 교수
†† 준 회 원 : 서원대학교 교양학부 교수
논문접수 : 2002년 9월 3일, 심사완료 : 2002년 9월 30일

는 클라이언트 동시요청이 많을 때 큰 파일의 완전한 로딩 성공률을 높이고자 하는 것이다. 왜냐하면 다운로딩시 잦은 실패여부는 클라이언트의 만족과 QoS에 직접적이고 매우 큰 영향을 주기 때문이다. 따라서 이렇게 비교적 큰 데이터를 전송하는 경우에는 일반적인 경우에 가장 효과적이라 할 수 있는 HTTP 반응시간(latency time) 평균을 이용한 알고리즘([4])보다는 기존의 HTTP 응답시간(response time)을 고려한 알고리즘이 더욱 신뢰도가 높아질 것이다



(그림 1) 부본서버를 이용한 웹 서비스 아키텍처 .

한편, ISP사들의 부본서버들은 일시에 구입하여 설치하는 것이 아니라 클라이언트의 증가에 따라 추가적인 설치가 많으므로, 부본서버의 하드웨어 성능에 따른 편차가 고려된다면 보다 효과적인 분배가 될 것이다. 즉, 각 부본서버의 성능 및 신뢰도가 서로 다르기 때문에 이를 평가하고, 서버의 성능 및 상태를 파악하여 요청에 대한 서비스를 가장 잘 제공할 수 있는 서버를 선택하도록 하는 것이 중요하다.

따라서 본 연구에서는 부본서버의 서비스 성능을 하드웨어 메트릭과 HTTP 응답시간 평균과 분산을 동시에 고려한 메트릭으로써 평가하고, 이에 따라 트래픽을 분산시킬 수 있는 알고리즘을 제안하고자 한다. 이를 위한 본 논문의 구성은 2장에서 부본서버를 이용한 트래픽 신뢰도 향상에 대한 관련 선행연구와 알고리즘을 요약하고, 3장에서는 부본서버성능에 영향을 미치는 외부요인을 측정하기 위한 사전 실험연구를 실시하였다. 그리고 4장에는 부본서버성능을 측정하기 위한 HTTP 반응시간의 2차원 벡터정보의 시뮬레이션 결과와 활용을 제안하고, 마지막 5장에서는 결론과 향후 연구를 제시하겠다.

2. 관련 연구

2.1 부본서버선택 알고리즘

클라이언트의 요청에 가장 적합한 부본서버 하나를 선택하기 위한 기준으로 근접척도(approximate metric)라는 것이 있는데, 이러한 근접척도는 지리적 위치, 네트워크 연결상태, 트래픽 부하 등 다양하게 설정될 수 있다. 먼저 [3]은 지리적 거리, 연결 홉(hop)의 개수, 난수발생, 핑 요청의 라운드 트립 시간(round trip measurement)의 평균값을 제안하였다.

그러나 [4]은 홉의 수나 핑 요청의 라운드 트립 시간이 웹 문서의 크기가 아주 작을 경우에는 적당하지만, 근접한 부본서버를 결정하는 일반적으로 좋은 척도가 아님을 보였다. 즉, 요청 후 완전한 문서가 오기까지의 HTTP 응답시간보다 요청 후 첫 바이트가 오기까지의 HTTP 반응시간의 사용이 더 효과적인 것, HTTP 반응시간과 HTTP 응답시간의 상관관계는 매우 높다는 것, 그리고 HTTP 반응시간과 문서크기의 상관관계는 거의 없는 것을 보였다. HTTP 반응시간을 이용한 부본서버 동적선택 알고리즘을 정리하면 다음 <표 1>과 같다[4, 5].

위의 부본서버 선택 알고리즘 외에도 동일확률 분배의 고정 알고리즘과 홉의 개수를 이용한 정적 알고리즘은 [3]를 참고하면 된다. 그러나 이러한 부본선택 알고리즘들은 기존의 서버운영 로그결과 분석이 선행되어야 하므로, 서비스를 처음 시작하는 경우에는 적용하기 적당치 않은 것도 있다. 예를 들어, 문서의 인기도인 이용접속 확률분포에 의하여 부본서버를 선택하거나[1], 문서의 멀티미디어 개체 크기와 문서의 이용접속 확률행렬을 고려하여 부본에서 다운로드 시간을 최소화하기 위한 비용모델(cost model)이 요구된다[6].

〈표 1〉 동적선택 알고리즘

	정의 및 장단점
병 렬 (parallel)	모든 부본서버에 매번 요청을 보내 가장 빠른 HTTP 반응시간을 보이는 부본서버 선택. 가장 효과적이나 동시 사용자에게 같은 부본서버를 선택하게 할 위험.
확 률 (probabilistic)	가장 빠른 HTTP 반응시간을 갖는 부본서버가 선택될 확률을 크게 배정. 부본서버의 정보를 갱신할 추가 요청이 필요 없으나, 최선이 아닌 부본서버에 보내지는 요청회수를 통제할 수 없음.
재 시 도 (refresh)	제한된 시간에 HTTP 반응시간 내 반응이 오면 그 부본서버를 선택하고, 초과하면 새로운 HTTP 반응시간 값을 얻는다. 추가 요청을 생성하는 단점이 있으며, 리플래시 요청 회수는 HTTP 반응시간 값으로 조정할 수 있음.
확장된 재시도 (extended refresh)	여러 부본서버의 HTTP 반응시간 평균은 유사하나 서비스 질을 의미하는 분산은 크게 다르므로, S-percentile기법을 이용하여 HTTP 반응시간 평균을 추정하는 재시도 알고리즘의 확장[5].

2.2 부본서버 성능평가

인터넷 서비스를 제공하는 서버의 성능을 측정하기 위해 고려할 중요한 것은 서버의 하드웨어, 소프트웨어, 네트워크 상태이다. 또 각 부분마다 상태를 측정하기 위한 메트릭은 다르다. 예를 들어 하드웨어 성능을 평가하여야 한다면 CPU의 명령처리속도, 메모리용량, 캐시 버퍼의 용량, 하드디스크의 속도 등이 있다.

본 연구에서는 네트워크의 상태와 같이 서버의 외부 환

경적 요인과 별도로 서버마다의 처리능력을 평가하기 위한 시뮬레이션을 수행하였다. 같은 하드웨어와 같은 소프트웨어를 쓴다하더라도 서버의 성능이 완벽히 동일하지 않을 수 있으며, ISP나 기업 등에서 서버를 추가할 때 기존의 서버와 동일한 서버를 추가하는 경우는 드물기 때문에 통제된 실험을 통하여 서버의 성능을 측정하고 이를 바탕으로 요청을 분산시킬 수 있는 기준을 제시하는 것이 의미 있다.

인터넷 서비스를 제공하기 위한 m개의 부본서버 $\overrightarrow{R}=(R_1, R_2, \cdots, R_m)$가 있다고 했을 때, 임의의 지리적 위치에서 서비스를 요청한 클라이언트 C에 대하여 최적의 서비스를 제공해줄 수 있는 서버를 선택하기 위하여 다음 식 (1)의 서비스 부하 SL의 값이 최소인 부본서버를 선택하여야 한다[8]. 즉, 서비스 부하 SL은 클라이언트와 서버간의 네트워크 부하량과 부본서버 부하의 논리합이며, 이를 최소화하는 부본서버를 선택하는 것이다.

$$SL = \min(Network\ Load) + \min(RL_1, RL_2, \cdots, RL_m) \quad (1)$$

단, RL_i : i 번째 부본서버 부하량

$i = 1, 2, \cdots, m$

식 (1)의 네트워크 부하는 [3]과 같이 접속하는 클라이언트의 지리적 위치 등의 많은 요인에 의해 영향을 받는 외부요인으로 서버관리자가 제어할 수 없다. 만약 대부분의 클라이언트들이 네트워크 부하가 동일한 지리적 도메인 내에 위치한다면, 이 항을 상수로 가정할 수 있으며 각 부본서버 중 부하량이 최소인 서버를 선택하는 것이 최적의 선택이 될 수 있다.

식 (1)의 두 번째 항인 부본서버의 부하량 RL_i가 문서의 요청확률, 단위시간당 서비스된 문서의 크기(byte), 서버저장능력의 함수로 보고 최소화를 위한 알고리즘[1], RL_i를 HTTP 반응시간의 함수로 보고 평균을 이용한 알고리즘[4], 그리고 서비스하는 문서에 포함된 멀티미디어 오브젝트의 크기에 따른 전송비용을 이용한 알고리즘[6] 등이 제안되었다. 그러나 [1]은 너무 많은 양의 계산이 요구되어 현실적이지 못하며, [4]은 부본서버들간의 하드웨어 성능차가 매우 커서 클라이언트 요청에 의한 HTTP 반응시간 평균값의 변화가 매우 심한 경우에는 효과적이지 않고, [6]은 문서가 바뀔 때마다 매번 전송비용을 재계산해야 하는 불편함이 있다.

3. 부본서버 성능평가

본 연구에서는 먼저 지리적 요인에 따른 네트워크 부하의 영향을 알아보고, 대부분의 클라이언트들이 지리적으로 유사한 도메인내에 위치하고 있는 경우에 RL_i에 영향을 주는 요인을 고려한 알고리즘을 제안하고자 한다. 즉 RL_i의 하드웨어 성능을 고려한 알고리즘과 HTTP 응답시간의 평균과 분산을 동시에 고려한 알고리즘을 제안·비교함으로써, VOD 서비스와 같이 용량이 큰 컨텐츠를 서비스하는 경우의 QoS가 어떻게 달라지는지를 보이고자 한다.

3.1 부본서버 성능측정 실험

실험을 통하여 하드웨어의 성능이 현격히 다른 부본서버의 부하량을 측정비교함으로써 네트워크 부하의 영향이 유의한지를 측정함으로써 식 (1)에 대한 실험결과를 얻고자 한다.

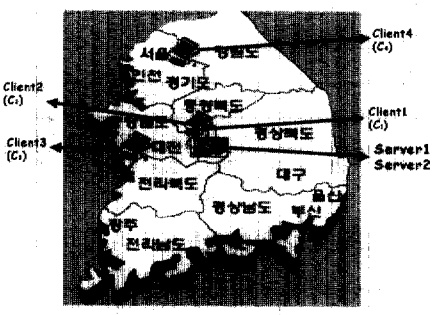
실험에 사용한 웹 서버는 하드웨어인 성능이 확연히 구분되도록 다음 <표 2>와 같이 구성하고 운영체제는 Windows 2000 Professional이며, 웹 서버는 IIS 5.0을 사용하여 C지역에 설치하였다.

〈표 2〉 서버 환경

	Server 1	Server 2
CPU	Intel Celeron 301MHz	Intel Pentium4 1.7GHz
Cache	L2 123KB	L2/L3 256KB
Performance Rating	PR361	PR1863
Memory	128MB SDRAM	256MB SDRAM
Memory Bus Speed	67MHz	133MHz
Hard Disk	4.8GB 10%Free	55.9GB 90%Free

이때 Performance Rating값은 펜티엄 표준 프로세서가 해당 서버의 프로세서의 처리수준과 동일하도록 계산된 것으로써, 다른 시스템간의 프로세서를 비교하는데 유용하다. 이는 실험대상 서버에 실제 응용 프로그램을 사용한 벤치마킹 시스템을 실행하여 얻어지는 것으로, 본 실험에서는 Sandra를 이용하였다[9].

클라이언트 프로그램은 서버에 있는 웹 문서를 일정기간 동안 3초마다 한번씩 요청하도록 Visual C++를 이용하여 개발하였으며, 클라이언트의 동시적 접근을 위하여 시스템 시간을 동기화 하고 시뮬레이션 시작/종료시간을 동시지정함으로써, 모든 클라이언트들이 같은 기간동안 서버에 요청을 보내고 서버의 반응시간과 서비스 성공여부를 체크하였다.

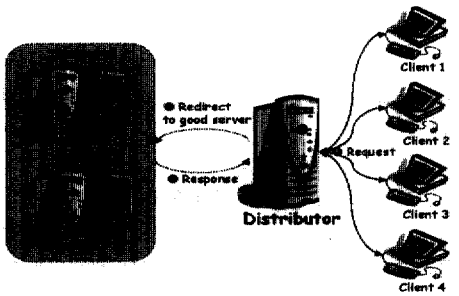


(그림 2) 부본서버 성능측정을 위한 지리환경

클라이언트는 (그림 2)과 같이 지리적으로 서로 다른 4곳에 위치하고 있으며, 4곳의 위치는 서버와 같은 LAN에 속하는 한곳과 C지역을 중심으로, $1km$(C_1), $4km$이상(C_2), $40Km$이상(C_3), $160km$이상(C_4)에서 서비스 요청을 발생하도록 실험환경을 구축하였다.

(그림 3)과 같이 클라이언트의 요청은 분배기로 먼저 전달되고 분배알고리즘에 의하여 좋은 서비스를 제공할 수

있을 것으로 판단되는 서버를 선택한다. 그리고 선택된 서버로 요청을 넘기고 서버로부터 서비스가 완료되는 시간과 서비스의 성공여부를 기록하여 각 분배알고리즘의 효율성을 비교할 수 있도록 하였다.



(그림 3) 실험 환경

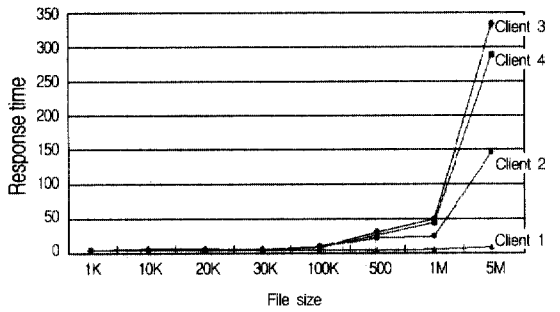
이 실험으로써 식 (1)의 서버-클라이언트간의 거리에 따른 서비스 반응시간에 미치는 영향인 *SL*값의 *Network Load*를 분석할 수 있다.

3.2 부본서버의 성능평가 결과

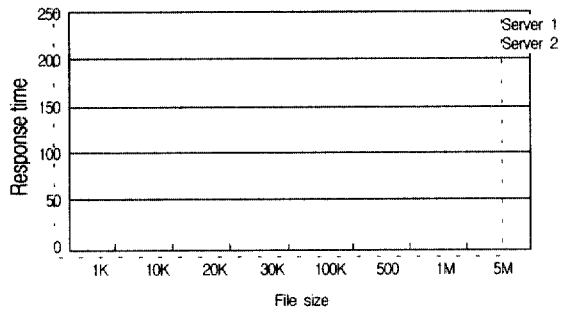
다음의 (그림 4)와 (그림 5)는 지역별, 서버성능별 부본서버의 HTTP 응답시간 평균을 실험한 결과이다.

먼저 (그림 4)를 보면, 파일 용량이 500*KB* 이하는 큰 차이가 없으나 1*MB* 이상은 클라이언트의 지리적 거리에 따라 HTTP 응답시간이 지수적으로 증가함을 보인다. 즉 식 (1)의 *SL*은 지리적인 요인인 *Network Load*의 지수함수와 비례함을 알 수 있다.

서버별 HTTP 응답시간의 평균을 나타낸 (그림 5)을 보면, Server 2가 Server 1보다 하드웨어 성능이 좋은 서버이고 HTTP 응답시간이 더 빠른 것을 알 수 있다. 역시 파일사이즈가 500*KB* 이상인 경우 두 부본서버의 성능에 따라 부하량이 일정하게 차이를 유지함을 볼 수 있다.



(그림 4) 지역별 반응시간 비교그래프



(그림 5) 서버성능별 반응시간 비교그래프

하드웨어 성능에 많은 차이를 보이는 두 서버의 HTTP 응답시간의 평균과 분산을 비교한 결과는 다음 <표 3>과 같다.

〈표 3〉 부본서버의 반응시간

		client 1	client 2	client 3	client 4	합 계
Server 1	평 균	0.8	25.6	48.1	59.3	48.3
	분 산	2.5	2768.1	11723.5	19178.9	108.1
Server 2	평 균	0.7	22.5	41.2	43.9	42.5
	분 산	2.2	2127.3	8254.7	8398.8	91.2

즉, Server 2가 Server 1보다 하드웨어 성능이 높기 때문에 HTTP 응답시간이 평균적으로 더 빠른 것을 알 수 있으며, 응답시간의 분산 또한 하드웨어 성능이 높은 서버의 응답시간 편차가 작아 안정적인 서비스를 제공하는 것으로 나타났다. 본 시뮬레이션 결과에서는 두 하드웨어의 성능이 트래픽 처리 능력간에 통계적으로 유의한 영향을 보이지는 않았다(p-값 0.3782).

따라서 (그림 4)와 (그림 5)의 실험결과에 의하여, 식 (1)은 다음 식 (2)와 같이 가정할 수 있다.

$$Q = \min(a\exp^{bx}) + \min(RL_1, RL_2, \cdots, RL_m) \qquad (2)$$

즉, 클라이언트의 요청이 증가하고 서비스하는 파일사이즈가 클 경우 트래픽이 증가함에 따라 처리능력은 매우 유의한 영향을 끼칠 것이다. 다음 절에서는 식 (2)의 가정 하에서 가장 효과적인 분배 알고리즘을 제안하고자 한다.

4. 성능평가 분배 알고리즘

4.1 분배 알고리즘

본 연구에서는 클라이언트가 요청한 임의의 고정된 시점에 있어서, 서비스를 수행중인 부본서버 RL_i는 다음 식 (3)과 같이 하드웨어의 성능, HTTP 응답시간의 평균과 분산의 함수로 가정하고자 한다.

$$RL_i = f(HW_i, MLSW_i, SLSW_i) \qquad (3)$$

단, HW_i : i번째 하드웨어 성능,

$MLSW$: 응답시간평균, $SLSW$: 응답시간편차

왜냐하면 앞 장에서의 시뮬레이션 결과를 살펴보면, 부본서버들간의 하드웨어 성능의 차가 큰 경우와 서비스해야할 파일의 크기가 클수록 각 부본 서버들간의 서비스 처리속도는 평균뿐만 아니라 분산 역시 매우 크게 다르므로, [4]와 같이 단순히 HTTP 응답시간 평균값만을 고려한다면 효과

적이지 못하다. 실제로 많은 상업적 사이트의 부본서버들은 예산에 따라 추가구입 및 업그레이드가 되므로 하드웨어 성능차가 큰 경우가 많고, 멀티미디어 컨텐츠의 증가로 파일사이즈가 큰 경우가 많기 때문이다.

여기서 식 (3)의 *HW* 성능을 이용한 메트릭으로는 <표 2>의 Performance Rating을 활용할 수 있다. 또한 큰 사이즈의 파일전송이 잦은 경우 *MLSW*(Mean of Latency Time for Software)와 *SLS*(Standard Deviation of Latency Time for Software)를 동시에 고려하기 위하여, 다음 식 (4)와 같이 공정능력지수(Process Capability Index)를 새로운 메트릭으로 적용하고자 한다.

$$C_p = \frac{USL - LSL}{6\sigma}, \quad (\geq 0) \tag{4}$$

단, $USL(LSL)$: 최대(소)허용 *HTTP* 반응시간,
σ : HTTP 반응시간의 모표준편차,

m개의 부본서버의 공정능력지수(C_p)를 추정하여 이 값이 1보다 크거나 같으면, 부본서버의 HTTP 응답시간의 변동이 작으므로 i번째 부본서버가 선택될 확률, $P(S_i)$,을 크게 한다. 이와 반대로 $C_p < 1$이면 서버의 HTTP 응답시간의 변동이 크므로 그만큼 하드웨어 성능과 접속량 처리가 늦으므로 QoS를 좋게 하기 위하여 $P(S_i)$를 작게 하는 것이다[7].

분배 알고리즘은 (그림 6)과 같이 C_p값을 통해 계산된 $P(S_i)$과 1의 차가 허용한계 안에 들어오는 적당한 k값에 의해 각 부본서버의 선택확률을 계산·분배한다.

Algorithm distributor( ) **returns** server_no
1. **for** ($k = 0, m-1, \varepsilon$) **do**
2. **for** ($1, m, i^{++}$) **do**
3. **if** ($C_{p_i} \geq 1$) **then** $P(S_i) = \frac{1}{m-k}$
4. **else** $P(S_i) = \frac{1}{m+k}$
5. **end for**
6. **if** $\sum_{i=1}^{m} P(S_i) - 1 < \varepsilon$ **then**
7. p = prob_alloc ($P(S_1)$, $P(S_2)$, ···, $P(S_m)$)
8. **return** p
9. **end for**

단, $0 \leq k < m$이며 $\sum_{i=1}^{m} P(S_i) = 1$인 실수
$\varepsilon > 0$은 $\sum_{i=1}^{m} P(S_i) = 1$의 허용한계
C_{p_i}는 서버 S_i의 공정능력지수

(그림 6) PCI 적용분배를 위한 계산 알고리즘

4.2 분배 알고리즘의 성능평가

본 절은 부본서버를 이용하는 다수의 클라이언트들이 지리적으로 넓게 분포하지 않는 도메인 내에서 용량이 1*MB* 이상의 비교적 큰 컨텐츠 서비스를 하는 경우를 가정할 때 적용 가능한 알고리즘들을 비교·평가하였다.

즉, 기존의 HTTP 평균 응답시간 알고리즘, 하드웨어 성능을 나타내는 Performance Rating 알고리즘, 앞 절에서 HTTP 응답시간 평균과 분산을 동시에 고려한 PCI, C_p를 적용한 세 알고리즘에 대하여 다음 <표 4>와 같이 비교하였다.

〈표 4〉 부본서버 선택 알고리즘의 장단점

	HTTP 응답분배	HW 성능분배	PCI 적용분배
장 점	• 요청하는 파일의 크기가 작은 경우 적당	• 계산이 간단 • 사전실험결과 불필요 • 부본서버 HW 성능차가 클 경우 효과적	• 사이즈가 큰 파일일 경우 적당 • 부본서버 HW 성능차가 클 경우 효과적
단 점	• 파일 사이즈가 클 경우 계산량 과다 • 부본서버간 HW 성능차이 비고려	• 부본서버 HW가 유사한 경우 효과 없음 • 부본서버 HW 성능계산방법이 주관적임	• 계산량이 많음 • PCI의 추정량값과 확률배분값의 조정이 필요함

위의 세 가지 메트릭에 대한 분배 알고리즘의 성능을 비교평가하기 위하여, 다음 <표 5>와 같이 분배 확률을 설정하였다. 이때, HTTP 평균 응답시간분배는 사전실험(pilot test)에 의해 얻어진 결과이며, HW 성능은 Sandra에서 제공하는 performance rating값을 활용하였고, 마지막으로 PCI는 Server1에는 $C_p = 1.1$을 그리고 Server 2에는 $C_p = 0.9$를 세팅하여 계산하였다. <표 5>의 분배확률은 다음 (그림 1)의 프락시 서버의 분배기에 탑재되어 클라이언트의 요청을 분배한다.

〈표 5〉 부본서버의 요청분배 확률

분배법 / 분배확률	HTTP 평균 응답시간분배	HW 성능분배	PCI 적용분배
Prob(Server 1)	0.47	0.16	0.42
Prob(Server 2)	0.53	0.84	0.58

용량이 큰 컨텐츠의 경우 다운로딩 중간에 끊기는 경우는 클라이언트가 체감하는 QoS에 아주 중대한 영향을 끼치므로, 성공률여부로 알고리즘을 다음 <표 6>와 같이 비교·평가하였다. 즉, 분배기가 파일크기 500*KB*, 1*MB*에 대하여, 동시 1,000번의 요청 시뮬레이션을 했을 때 그 성공률 결과와 실패한 경우를 60초로 계산한 평균 응답시간을 제시하였다. 이 시뮬레이션 결과는 더 큰 용량의 컨텐츠(5*MB*)에서도 동일한 결과를 보였는데, 500*KB* 이하에서는 HTTP 응답시간 평균분배 알고리즘이 가장 좋은 성공률을 보였고 HW 성능분배 알고리즘이 따르고 있으나, 1*MB* 이상의 경우는 PCI 적용 분배 알고리즘이 나머지 두 알고리즘보다 좋은 결과를 나타냈고, HTTP 응답시간 평균 값 역시 가장 작았다.

〈표 6〉 분배법에 따른 요청성공률 비교

분 배 법		HTTP 응답시간 평균분배	HW 성능분배	PCI 적용분배
500*KB*	요청성공률	0.952	0.895	0.857
	평균응답시간	35.32	36.37	38.83
1*MB*	요청성공률	0.883	0.883	0.943
	평균응답시간	45.59	49.11	37.36

이는 PCI가 HTTP 응답시간의 평균과 분산 2차원 벡터 메트릭을 사용함으로써, 보다 많은 정보를 토대로 분배확률이 결정되었기 때문이다. 따라서 유사한 거리에 밀집된 클라이언트가 요청한 컨텐츠의 양이 비교적 크고, 부본서버간의 하드웨어 성능처리가 다른 경우에는 PCI를 고려한 알고리즘이 가장 효과적이라 할 수 있다.

5. 결 론

부본서버를 이용하여 인터넷 응용 서비스를 제공하는 경우 효과적인 트래픽 분배 방법이 필요하다. 대부분의 부본서버 선택 알고리즘은 클라이언트가 요청한 컨텐츠의 양과 확률분포 또는 HTTP 반응시간과 같은 메트릭을 사용하고 있다. 그러나 용량이 큰 컨텐츠의 경우는 클라이언트의 다운로딩 또는 서비스 완료가 QoS에 중요한 영향을 끼치므로, HTTP 반응시간보다는 HTTP 응답시간이 더 적당하다. 또한 인터넷 서비스를 제공하는 많은 사이트들의 부본서버들은 하드웨어의 구입시기와 예산 때문에 하드웨어간의 성능 차가 크므로 HTTP 응답시간 평균만으로는 좋은 분배를 기대할 수 없다.

따라서 본 논문에서는 비교적 큰 용량의 컨텐츠를 하드웨어 종류가 다양한 부본서버들을 통해서 인터넷 서비스 하는 경우를 위하여, 부본서버의 하드웨어 성능에 따라 분배하는 알고리즘과 부본서버의 HTTP 응답시간 평균과 분산을 동시에 고려하는 PCI 메트릭을 활용한 분배알고리즘을 제안하고 장단점을 논하였다. 먼저 하드웨어 성능분배 알고리즘은 계산이 간단하며 사전데이터 분석이 필요하지 않는 장점이 있으나, 부본서버간의 하드웨어 성능차이가 크지 않을 때는 적당하지 않다. 그리고 PCI 분배알고리즘은 부본서버의 하드웨어 성능차가 클 때와 서비스되는 컨텐츠의 용량이 클때 적당하나, 사전 분석결과가 요구되며 계산양이 많다.

제안된 두 알고리즘의 휴리스틱한 비교실험결과, 서비스되는 컨텐츠의 용량이 큰 경우 기존의 HTTP 응답시간 평균분배 알고리즘보다 클라이언트 요청 성공률이 더 높았으며, HTTP 평균 응답시간도 훨씬 짧은 시뮬레이션 결과를 보여주었다. 이로써 클라이언트의 지리적 외부 네트워크의 부하량이 일정하다고 고려했을 경우, 본 논문에서 제안하는 알고리즘이 효과적임을 보였다.

향후 연구로는 PCI를 이용한 분배알고리즘의 C_p를 붓스트랩(bootstrapping) 기법을 적용하여 추정하는 알고리즘 개발과 시뮬레이션 연구로 확장할 수 있을 것이다.

참 고 문 헌

[1] Azer Bestavros, "Demand-based Document Dissemination to Reduce Traffic and Balance Load," Proceedings of SPDP, 1995.

[2] Kevin Thompson, Gregory J. Miller, Rick Wilder, "Wide Area Internet Traffix Patterns and Caracteristics," IEEE Network, pp.10-23, Nov., 1997.

[3] M. E. Crovella, R. L. Carter, "Dynamic server Selection in the Internet," proceedings of the Third IEEE workshop on the Architecture and Implementation of High Performance Communication Subsystems, 1995.

[4] Mehmet Sayal, Yuri Breitbart, Scheuermann, "Selection Algorithms for Replicated Web servers," Proceedings of the Workshop on Internet Server Performance, 1998.

[5] Radek Vingralek, Yuri Breitbart, Mehmet Sayal, Peter Scheuermann., "Web++ : A System for Fast and Reliable Web Service," Proceedings of the USENIX Annual Technical Conference, 1999.

[6] Thanasis Loukopoulos and Ishfaq Ahmad, "Replicating the Contents of a WWW Multimedia Repository to Minimize Download Time," Proceedings of the 14th International Parallel and Distributed Processing Symposium, 2000.

[7] 이경희, 한정혜, 김동호, "효율적 웹 기반 VOD서비스를 위한 에이전트시스템", 한국디지털컨텐츠학회논문지, Vol.2, No.1, pp.73-80, 2001.

[8] 이재돈, 전길남, "도메인 네임 시스템을 이용한 복제 서비스에서의 동적 서버 선택 알고리즘", 한국정보과학회지, Vol.4, No.19, pp.65-71, 2001.

[9] SiSoftware Sandra (the System ANalyser, Diagnostic and Reporting Assistant), http://www.sisoftware.co.uk/sandra.

한 정 혜

e-mail : hanjh@cje.ac.kr

1998년 충북대학교 대학원 전자계산학과 (이학박사)

1998년~1999년 연세대학교 컴퓨터 전기전자학부 포닥연구원

1999년~2001년 행정자치부 국가 전문행정 연수원 전임교수

2001년~현재 청주교육대학교 컴퓨터 교육과 교수

관심분야 : 네트워크 관리시스템, 전자상거래, 데이터마이닝, WBI

이 경 희

e-mail : khlee@seowon.ac.kr

1999년 충북대학교 대학원 전자계산학과 (이학석사)

2001년 충북대학교 대학원 전자계산학과 박사수료

2000년~2002년 (주)엔슬래시 닷컴 선임연구원

2002년~현재 서원대학교 교양학부 교수

관심분야 : 네트워크 관리시스템, XML 문서관리 시스템