SLA를 지원하는 웹 서버 부하 분산 기법
(A Web Server Load Balancing Mechanism for Supporting Service Level Agreement)

고 현 주 † 박 기 진 †† 박 미 선 †††
(Hyeonjoo Go) (Kiejin Park) (Misun Park)

요 약 클라이언트와 서비스 제공자간의 서비스 수준 계약인 SLA(Service Level Agreement)를 만족시키기 위해서는 클라이언트 요청을 우선순위 계층으로 구분하여, 낮은 수준의 서비스를 요청하는 클라이언트 보다는 고수준의 서비스를 요구하는 클라이언트에게 우선적으로 서비스를 제공할 수 있는 기술이 필요하다. 본 논문에서는 서비스 제공자의 웹 서버 노드를 우선 순위에 따라 정적·동적으로 분할하는 방법 및 다중계층(Multiclass)에서의 승인 제어(Admission Control) 기법을 연구하였으며, 시뮬레이션을 통해 SLA를 고려한 웹 서버 처리율 및 응답시간 성능을 분석하였다.

키워드 : SLA(Service Level Agreement), 웹 서버, 성능분할, 승인제어

Abstract To satisfy SLA(Service Level Agreement) contract between a client and a service provider, the client requests are classified and processed by priorities. In providing differentiated service, a request from a client who has low priority can be dealt with less important. In this paper, we study static and dynamic partitioning mechanism of web servers and also admission control policy for multiclass request distribution. Through simulation experiments, we analyze web server throughput and response time considering SLA.

Key words : SLA, Web Server, Performance Isolation, Admission Control

1. 서 론

인터넷 비즈니스와 관련되어 신뢰성 있는 고품질의 서비스를 제공하기 위해서는, 웹 서버에서 서비스 수준(Service Level)을 고려할 수 있어야 한다. 현재까지 서비스 수준을 제공하기 위한 연구는 대부분 네트워크 단계를 대상으로 이루어 졌으나, 웹 서버에서는 클라이언트 요청을 우선 순위가 아닌 선입 선출(FIFO) 방식[1]으로 처리하기 때문에, 시스템의 과부하 상황에서는 우선 순위가 높은 클라이언트 요청을 거절하는 경우가 자주 발생한다. 따라서, 네트워크 단계의 QoS(Quality of Services) 만으로는 인터넷 비즈니스 양단간 QoS를 지원할 수 없는 문제점을 해결하기 위하여, 웹 서버 단계에서 클라이언트 요청의 우선 순위에 따라 차별화된 서비스(Differentiated Service)[2,3,4]를 제공할 수 있는 기법에 대한 연구가 요청되고 있다.

기존에는 클러스터 웹 서버의 부하 조절(Load Balancing)을 위해서 Layer-4[5]기반 스위치가 사용되었으나, 최근 내용기반 분배가 가능한 Layer-7[6]스위치로 급속히 대체되고 있다. Layer-4부하 분배 방식은 TCP/IP 레벨에서 동작하며, HTTP요구가 보내지기 전에 TCP/IP연결 설정이 이루어지기 때문에 서비스 노드에 대한 선택이 요구된 내용에 의해 결정될 수 없다. 이에 비하여 Layer-7 부하 분배 알고리즘은 서비스 노드를 결정하기 전에 HTTP요구를 분석할 수 있기 때문에 클라이언트 요구 내용에 대하여 상세한 정보를 고려한 분배가 가능하다[7,8]. 내용 기반 분배의 장점으로는 서비스 노드의 메인 메모리 캐쉬에서 향상된 적중률을 가질 수 있는 것과 디스크같은 기억 장치의 확장가능성 그리고 특정 목적의 서비스 노드 운영등을 들 수 있다. 한편, 웹 서버의 성능을 향상시키는 방법으로 과거에는 고성능의 프로세서(Processor)를 여러 개 장착하는 병렬 처리(Parallel Processing)기법을 사용하였으나, 물

· 본 연구는 한국학술진흥재단 선도연구자 지원사업(KRF-2005-041-D00630) 지원으로 수행되었음
† 정 회 원 : 디오텔㈜ 소프트웨어팀 연구원
gohj82@dotel.co.kr
†† 종신회원 : 아주대학교 산업정보시스템공학부 교수
kiejin@ajou.ac.kr
††† 정 회 원 : Nextouch(주) 소프트웨어팀
mspark@nextouch.biz
논문접수 : 2004년 9월 16일
심사완료 : 2005년 12월 1일

리적인 확장의 한계와 성능 대비 투자 비용의 증가로 효율성이 매우 떨어지므로, 웹 서비스 시스템을 구성하는 웹 서버를 클러스터링 하는 기법이 주로 채택되고 있다. 클러스터 시스템은 대량 생산되는 제품을 이용하기 때문에 가격대 성능비가 우수하며, 리눅스같은 무료 운영체제를 이용할 경우 기존의 동급 고성능 컴퓨터에 비해 구축 비용을 낮출 수 있다. 클러스터 시스템은 고성능과 고가용성 컴퓨팅 분야에서 널리 응용되고 있으며, 최근에는 웹 서비스 관련 연구가 많이 수행되고 있다[9,10].

웹 상에서 차별화된 서비스를 제공하기 위하여 웹 서버 단계의 QoWS(Quality of Web Services)에 관한 분석이 필요하다[11,12]. QoWS는 사용자 또는 어플리케이션에 대해 중요도에 따라 서비스 수준을 차등화하여 한정된 WAN대역폭에서 트래픽과 대역폭을 정책적으로 관리하는 기술이다. 더 간단히 말하면, 사용자가 원하는 특정 네트워크의 안정된 서비스 보장을 위해 대역폭을 미리 예약하는 네트워크 기술을 뜻한다. SLA (Service Level Agreement)[13,14]는 서비스 제공자와 대상 고객간 계약으로, 제공되어 질 서비스와 그와 연관된 조건들을 사전에 약속하여 그 만큼의 서비스를 제공하는 것이며, 차별화된 서비스를 통해 고객과 서비스 제공자간의 SLA 보장이 가능하다. 클러스터링된 여러 대의 서버 노드로 구성된 웹 서버에서는 클라이언트의 요청 수준에 따라 그에 알맞은 특정 서버 노드들을 할당함으로써, 효율적인 차별화 서비스를 구현할 수 있다. 즉, 이와같은 서버 할당 기법은 QoWS를 보장해주는 중요한 요소 중 하나인 성능 분리(Performance Isolation)[15]에 해당하며, 분할된 서버마다 서로 다른 계층의 서비스를 담당하게 하고, 상위 계층의 클라이언트 요청일수록 더 많은 서버를 할당해 줌으로써, 클라이언트 계층별로 SLA를 보장할 수 있게 된다. 여기서 성능 분리는 웹 서버에 들어오는 클라이언트 요청을 분류하여 계층별 큐(Queue)로 보내고, 웹 서버의 과부하 발생시 상위 계층의 클라이언트 요청을 받아들이고 하위 계층의 크라이언트 요청을 거절(Admission Control)하여 차별화된 서비스를 제공하는 기본적 방식을 의미한다[16]. 또한, 서버의 부하 상태와 사용자 요청 도착률의 변동을 고려하지 않은 클러스터 부하 분배는 자원 낭비가 발생하기 때문에, 사용자 계층별 요청 도착률에 따라 계층별 요청을 전담 처리하는 서버 노드 성능 분리 개념을 도입할 경우, 효율적인 서버 노드 자원 활용이 가능하다.

본 논문의 2장에서는 관련 연구를 언급하고, 3장에서 웹 서버의 구조에 대해 알아보고, 4장에서는 웹 서버 분할 기법을 제시 하였으며, 5장에서는 제안한 방법의 성능 평가를 수행 하고, 6장에는 결론을 내렸다.

2. 관련 연구

클러스터 웹 서버 구조에서 우선 순위별로 클라이언트 요청을 처리하기 위해, 서버 노드를 정적·동적으로 분할하는 기법이 연구되고 있다. 전자는 클라이언트의 계층별 요청 수준에 따라 고정된 수의 서버 노드를 계층별로 할당하는 방법으로 클라이언트의 계층별 요청 수가 자주 변하는 웹 환경에는 적당하지 않다[17]. 이는 특정 계층에 할당된 서버 노드가 과부하 상태인 반면에 다른 계층의 서버 노드는 유휴 상태에 있을 수 있기 때문이다.

동적 분할 기법은 클라이언트의 계층별 요청 수준 혹은 필요에 따라서 동적으로 서버 노드를 할당함으로써, 높은 자원 이용률을 제공한다. 동적 분할 기법에는 DDSD(Demand-driven Service Differentiation)[18], Dynamic Partitioning[19] 기법등이 있으며, DDSD기법은 클라이언트 요청량에 따라 CPU와 디스크 I/O용량을 할당함으로써 차별화된 서비스를 제공하고, e-비즈니스와 관련된 웹 사이트 컨텐츠처럼 CPU와 디스크 I/O의 처리를 많이 요구하는 동적 요구가 많은 상황에 적합하다. Dynamic Partitioning기법은 SLA를 만족하는 서비스를 상위 계층 사용자에게 제공하기 위해 서버의 부하량에 따라 서버 노드를 동적으로 분할하는 기법이다. 하지만 이들은 정적 요구를 전혀 고려하지 않고 동적 요구만 고려했기 때문에 정적 요구가 많아질 경우 성능 저하가 발생하는 문제점이 있다. 대체적으로 동적 분할 기법은 정적 분할 기법 보다 높은 자원 이용률을 제공하지만, 정적 요구가 많아질 경우 정적 분할 기법이 더 좋은 성능을 제공한다.

LARD(Locality-Aware Request Distribution)[20,21]는 웹 서버 클러스터에서 내용 기반 라우팅에 대한 초기의 연구로써, LARD에서 부하 분배기는 각 웹 문서에 대하여 서비스 노드와 일대일 대응을 유지하며 주어진 문서에 대하여 첫 번째 요구가 도달하면 가장 적은 부하가 걸려 있는 서비스 노드에 우선적으로 할당한다. 그 후 이 문서에 대한 요구가 도착되면 이전에 할당되었던 서비스 노드로 보내지나 만일 그 노드에 임계값을 넘는 과부하가 걸려 있으면 다른 노드 중 부하가 가장 작게 걸린 노드를 선택하여 그 문서에 대한 새로운 서비스 노드로 할당한다. 즉 기존의 동적 부하 알고리즘은 사용자 요구가 들어왔을 때 무조건 가장 적은 부하를 가진 노드를 할당하는데 비해 LARD는 캐쉬 적중률 향상을 위해 지나친 과부하가 걸리지 않는 이상, 이전의 서비스 노드를 할당하는 정책을 취한다. 하지만 정적 요청만을 고려한 점과 작업 부하에 가중치를 설정해야 하는 문제점이 있다.

클라이언트의 요청을 2개의 계층(High Set, Low Set)으로 분리하여, 차별화된 서비스를 제공하려는 시도가 있었으며[19], 본 논문에서는 클라이언트 계층을 단순히 2개로만 구별 하지 않고, 이를 일반화시킨 개념인 다중 계층(Multiclass) 문제를 다루었다. 즉, 기존의 연구 기법에서 고수준의 요청은 High 클래스, 저수준의 요청은 Low클래스의 서버가 서비스를 제공하는 반면, 본 논문의 다중 계층기법에서는 High 클래스, Low 클래스 외에도 계층을 더 분류함으로써 폭넓은 계층에게 SLA를 만족시키면서 각 계층마다 차별화된 서비스를 제공할 수 있다. 다중 계층은 클라이언트의 계층을 3개 이상으로 구분하여 서비스하는 것으로, 전체 서버를 3개 이상의 그룹으로 구분하여, 클라이언트의 요청 수준에 맞는 적합한 서버그룹으로 요청을 보내어 처리하며, 계층별 요청은 이렇게 구분되어진 각 계층으로 들어오는 클라이언트의 요청을 의미한다. 웹 서버 클라이언트의 계층을 일반화 시킨, 다중 계층 기법의 요청 분류는 서비스 제공자와 클라이언트간의 서비스 수준 계약인 SLA에 의해 이루어진다.

예를 들어, 서비스 제공자에게 비용을 지불한 클라이언트는 그렇지 않은 클라이언트보다 더 좋은 서비스를 제공받기를 원하며, 이 경우 클라이언트 요청의 계층을 판단하여 비용을 지불한 클라이언트는 High 계층에서 요청을 처리하며, 그렇지 않은 클라이언트는 Low 계층에서 처리하게 된다. 정리하면, 서버 노드가 클라이언트의 계층별 우선 순위에 따라 분할되며, 클라이언트 계층별 요청률에 따라 분할된 서버 노드들을 정적·동적으로 관리하는 기법 및 다중 계층에서의 승인 제어(Admission Control)에 관하여 분석하였다.

3. 웹 서버의 구조

그림 1은 본 논문에서 고려하고 있는 웹 서버 클러스터의 연결 구조이다. 로드발런서는 서비스 노드(Real Server)들 간에 작업들을 고루 분배하는 기능을 수행하며, 과중하게 부하를 받고 있는 노드들이 다른 노드들에게 작업을 전달하거나, 유휴상태에 있는 노드의 작업분배 요청을 처리한다. 이러한 웹 서버 구성 방식은 웹 클라이언트로 하여금 하나의 컴퓨터 즉, 로드발런서로 부터 서비스를 제공받는 것으로 여겨지게 하지만, 실제 서비스 제공은 로드발런서에 연결되어 있는 서비스 노드들에 의해서 이루어진다.

로드발런서는 지속적으로 부하 분배 서비스를 제공해야 하므로, 로드발런서의 과도한 집중이나 시스템 이상으로 다운되었을 때 대기하고 있던 다른 서버(Backup)가 로드발런서 역할을 대신 수행시킬 수 있어야 한다. 백업 서버는 로드발런서가 작동하지 않으면 대체되는 서버로써, 로드발런서와 연결되어 각각 Heartbeat데몬을 가동시켜 주기적으로 서로 Heartbeat을 주고 받는다. 백업 서버는 정의된 시간동안 로드발런서에서 Heartbeat이 도착하지 않으면 로드발런서가 고장난 것으로 간주하고, 로드발런서의 기능을 대신 수행한다. 이와같은 구조의 고가용성 웹 서버는 대규모 서비스를 제공하기 위한 목적으로 제작되며 주로 인터넷 비즈니스등에 활용가치가 높다[22,23].

본 논문에서는 사용자의 요청 내용을 계층별로 분류하여 처리할 수 있는 Layer-7 스위치를 이용하며, 요청처리 결과를 해당 서버에서 사용자에게 직접 전달하는 One-way 웹 서버 구조를 고려한다. One-way 구조에서는 들어오는(Inbound) 패킷은 웹 스위치를 거쳐 웹 서버로 전달되는 반면에 나가는(Outbound) 패킷은 웹 스위치를 거치지 않고 직접적으로 사용자에게 전달된다. 이에 비해 Two-way 구조는 클러스터 안의 각각의 서버는 유일한 하나의 IP 주소로 설정되어 들어오는 패킷과 나가는 패킷은 웹 스위치에 의해 재작성되기 때문에 One-way 구조에 비해 스위치의 오버헤드가 크다[25]. 웹 서버의 서비스 전제조건은 외부의 혼란이나 방해없이 정책들의 공정한 비교를 위해, 네트워크에 병목현상이 없다고 가정한다.

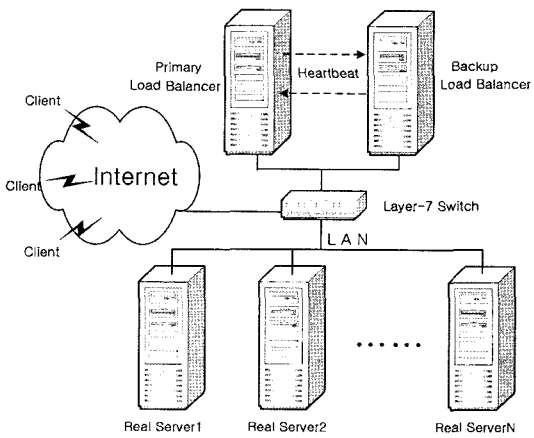


그림 1 웹 서버 클러스터 구조

4. 웹 서버 분할

본 장에서는 n개의 우선순위 계층으로 분류된 클라이언트 요청을 처리하기 위해, 정적·동적으로 서버를 분할하는 기법 및 승인 제어에 관한 내용을 기술하고자 한다.

4.1 서비스 분류(Service Classification)

서비스 분류는 로드발런서에 입력되는 클라이언트의 요청을 분류하여 계층별 큐(Queue)로 보내는 작업을 의미하며, 다양한 클라이언트 계층에 대한 차별화된 서비

스를 제공하기 위해 다중계층 개념을 도입하였다. 클라이언트 요청은 우선 순위에 따라 n개의 계층으로 구분된 후에, 그 요청을 처리할 수 있는 서버 노드가 결정된다. 전체 서버 수가 N대인, 서버 할당 방식을 수식으로 나타내면 식 (1)과 같으며, 여기서, $CS_i(t)$는 임의의 시간 t에서 계층 i에 할당되는 서버 수이며, $CS_{i-1}(t)$는 $CS_i(t)$보다 우선순위가 높다. 또한 $us_i(t)$는 임의의 시간 t에서 계층 i에 속하는 클라이언트 요청을 처리하는 서버를 구분하는 경계값이다. $us_1(t)$는 시간 t에서 최상위 계층을 서비스하는 최대 서버 노드의 수이다. 예를 들어 계층을 3개($CS_1(t)$, $CS_2(t)$, $CS_3(t)$)로 나누고 각 계층에 서버가 차례대로 2대, 3대, 5대가 할당된다고 가정하면, (1)식에 따라 $CS_1(t)$={1,2}, $CS_2(t)$={3,4,5}, $CS_3(t)$={6,7,8,9,10}의 식별번호로 구성된 서버 집합을 가지며, 집합의 원소의 개수가 클수록 해당 계층을 서비스하는 서버가 많다는 것을 의미한다. 이때 $us_1(t)=2$, $us_2(t)=5$는 서버를 구분하는 경계값으로 항상 정수값을 가진다고 가정(즉, 특정 서버는 특정 계층만을 담당)한다.

$$
\begin{aligned}
&CS_1(t) = \{1, \cdots, us_1(t)\},\\
&CS_2(t) = \{us_1(t)+1, \cdots, us_2(t)\},\\
&\qquad\vdots\\
&CS_n(t) = \{us_{n-1}(t)+1, \cdots, N\},\\
&us_i(t) \in \{1, \cdots, N\},\ us_1(t) \geq 1,\ us_{n-1}(t) \leq N-1,\\
&us_i(t) \leq us_{i+1}(t)
\end{aligned}
\tag{1}
$$

이와 같이 서버 노드들은 클라이언트 요청 우선 순위에 따라 n개의 계층으로 구분함으로 써, SLA를 만족하는 클라이언트의 계층별 차별화 서비스를 제공하고자 한다.

4.2 정적 분할(Static Partition)

정적 분할 기법은 기존의 계층별 클라이언트의 요청률만을 가지고, 서버 구분 경계값(e.g. $us_i(t)$)을 구하는 방식으로, 시스템의 최초 가동시에 각각의 클라이언트 계층별로 고정적인 서버 수가 할당되며, 한번 할당된 서버 노드는 클라이언트의 계층별 요청률이 변해도 수정되지 않는다. 서버 구분 경계값은 클라이언트 요청을 처리하는 각 계층의 서버를 구분하는 기준값이다. 즉, $us_1(t)$는 상위 계층($CS_1(t)$) 서버와 차상위 계층($CS_2(t)$) 서버를 구분해 줄 수 있는 서버 식별번호이며, 클라이언트가 서비스를 요청하면 서비스의 수준에 따라 그에 해당하는 계층으로 보내져서 서비스를 제공받게 된다. 식 (2)는 정적 분할 방식을 나타내고 있으며, $us_i(0)$는 시스템 최초 가동시에 정해지는 값으로, 최초 웹 서버를 가동시켰을 때 정해지는 i번째 계층을 서비스 하는 서버 경계 값이다(식 (1)참조). 여기서 ρ_i는 전체 요청 수와 i 계층에 속하는 클라이 언트의 요청 수의 비율이고, T_N은 N대의 서버에서 허용 가능한 최대 지연 시간이다. 또한, $MaxConn(T_N)$은 지연 시간 T_N을 만족시키는 최대 접속 수이며, T_i는 i 번째 계층의 SLA를 고려한 임계 시간(Threshold : $T_i < T_N$) 이다.

$$
us_i(0) = \lceil \rho_i \cdot N \rceil \cdot \frac{MaxConn(T_N)}{MaxConn(T_i)},\ \mathrm{i} = 1, \cdots, \mathrm{n}-1 \tag{2}
$$

일반적인 상황에서는 항상 T_N에서의 최대 접속 수를 만족하지 않기 때문에, T_N에서의 이상적인 최대 접속 수를 T_i에서의 실제 최대 접속 수로 나누어, 최대 접속 비율을 구한다. 기본적으로 서버 수를 구하기 위해서 ρ_i와 전체 서버 수를 곱하며, 최대 접속 비율를 곱하여 더 정확한 값을 끌어낸다. 식 (2)와 같은 정적 분할 방식은 특정 웹 서버 노드에 과부하가 걸렸다는 것을 웹 관리자가 인지하기 전까지 $us_i(0)$ 값을 계속 그대로 유지하는 문제점을 가지고 있다.

4.3 동적 분할(Dynamic Partition)

동적 분할 기법은 상위 계층의 사용자의 SLA를 만족하는 서비스를 제공하기 위해, 계층별 부하량에 따라 서버 노드를 동적으로 분할하는 기법으로, 특정 계층을 서비스하는 노드에 과부하가 걸렸을 경우, 이보다는 더 낮은 계층을 서비스하고 있는 서버를 추가적으로 이용하거나, 과부하 상황이 해소되면 다시 서버를 반납하는 것을 기본 개념으로 하고있다. 동적 분할 개념은 부하 변화에 맞게 대처할 수 있어 여러 사용 계층자들에게 효율적이면서도 질 높은 서비스를 제공할 수 있다.

그림 2는 계층 i를 서비스 하는 서버의 동적 분할 방식을 표시하고 있으며, 부하의 변동 비율(α)에 따라 클라이언트 계층별 서버 노드의 경계값이 계산된다. 즉, 실제 부하량과 처리 용량과의 차이를 구하여, α%이내의 변동은 서버 분할에 영향을 미치지 않도록 하였다. 여기서 $SumLoad_{CS_i}(t)$는 $CS_i(t-1)$에서의 서버 부하(Load)의 합이며, 부하는 클라이언트의 최대 접속 수

($MaxConn$)를 의미한다. 해당 계층의 부하합이 작업 능력을 넘어서면, 과부하 상태로 판단하며 기존에 할당받은 서버로는 효율적으로 일을 처리할 수 없게 된다.

$$Diff = SumLoad_{CS_i}(t) - (us_i(t-1) - us_{i-1}(t-1)) \cdot MaxConn(T_i);$$

if(| $Diff$ | > $SumLoad_{CS_i}(t) \cdot \alpha$){

If ($SumLoad_{CS_i}(t) > (us_i(t-1) - us_{i-1}(t-1)) \cdot MaxConn(T_i)$)

$$us_i(t) = us_i(t-1) + 1;$$

else if($SumLoad_{CS_i}(t) < (us_i(t-1) - us_{i-1}(t-1)) \cdot MaxConn(T_i)$)

$$us_i(t) = us_i(t-1) - 1;$$

}

그림 2 웹 서버 동적 분할 알고리즘

이와 같이 동적으로 서버 노드를 분할하려면, 과부하 상태시 어떤 계층에서 서버를 제공받을 것인지, 혹은 반대의 경우 누구에게 돌려줄 것 인지도 고려해야 하며, 본 논문에서는 특정 계층의 클라이언트 요청률이 높아져 과부하가 걸릴 경우, 과부하가 걸린 서버보다 한 단계 낮은 우선 순위의 계층에서 서버를 가져가고, 클라이언트의 요청이 적어지면 다시 한 단계 낮은 우선 순위의 계층으로 반납하는 방식을 채택하였다(그림 3참조).

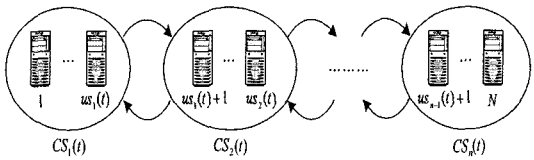


그림 3 서버 노드의 획득과 반납

4.4 승인 제어(Admission Control)

클라이언트 요구 승인은 차별화된 서비스를 제공하기 위한 가장 기본적인 기법으로써 웹 서버의 과부하 발생시 상위 계층의 클라이언트 요청은 받아들이고 하위 계층의 클라 이언트 요청은 거절하는 방식을 따른다. i번째 계층 웹 서버의 과부하시 클라이언트 요구를 거절하는 기준은 식 (3)에 나타나 있다.

$$SumLoad_{CS_i}(t) > \gamma \cdot (us_i(t) - us_{i-1}(t)) \cdot MaxConn(T_i), \quad i = 2,3,\cdots,n \tag{3}$$

파라미터 γ 은 $CS_i(t)$ 에서 서버의 부하상태와 탈락된 요청 수 사이에서 거절 여부를 조절해주는 비율이다. 식 (3)를 만족하면, i번째 계층 웹 서버의 과부하 상태가 되어, i+1 번째 계층에 서버가 있는지 없는지 확인을 한 뒤, 서버가 있으면 i번째 계층은 i+1번째 계층에서 서버를 가지고와 요청을 처리하며, 서버가 없다고 판단되면 들어온 요청을 거절한다.

5. 성능 평가

실제적인 웹 서버 클러스터 시스템의 성능 분할을 원활히 시뮬레이션하기 위해, 클라이언트의 도착률에 따른 응답시간 및 승인 거절된 요청 수에 관하여 분석하였으며, 95-percentile은 SLA를 고려하기 위해 사용된 척도로써, 들어온 요청중 95% 이상은 서비스 제공자와 고객간의 계약된 시간 안에 응답시간을 만족한다는 것을 의미한다. 웹 서버는 각 클라이언트 계층으로부터 정적 요청과 동적 요청을 받아들이며, 시뮬레이션에 사용된 웹 서버 클러스터 시스템의 운영 파라미터는 표 1, 2와 같다[17,25]. 시뮬레이션은 PC 환경(Pentium 4)에서 C 언어를 사용하여 수행하였으며, 서버내 각 자원으로의 클라이언트 요청 도착 시간 간격을 지수(Exponential) 분포를 사용하여 표현하였다. 클라이언트의 서비스 요청 사건(Event)이 발생되면, 이는 다시 서비스 시간이 상이한 동적 요청과 정적 요청을 랜덤(Random)한 비율로 발생시킨다. 기본적으로 클라이언트 요청은 동적 요청 20%, 정적 요청 80% 비율로 구성되어 있다.

동적 요청의 평균 서비스 시간 간격은 700 msec., 정적 요청의 평균 서비스 시간 간격은 100 msec. 로 하였으며, 발생된 요청은 서버 내 각 자원의 점유를 위해 해당 대기큐에 입력되고, 이미 해당 자원이 서비스를 수행하고 있으면 큐에 대기하게 된다. 요청 패킷의 경로는 요청의 계층별 우선순위에 분리되어 그에 상응하는 서버 집합(Set)이 결정된다. 각 자원의 사용 시간과 대기시간으로부터 시스템의 응답 시간이 계산되며, 그 외에 0~100% 구간의 다양한 구성 비율의 조합으로 실험을 수행하였다. 또한, 서버 수는 10~30대 구간의 경우 수를 모두 적용하였으며, 클라이언트 계층은 High · Medium · Low 3개의 수준으로 나누어 성능을 평가했다.

그림 4에서는 3개의 계층(High, Medium, Low)을 대상으로, 정적 · 동적 분할에 따른 응답 시간을 클라이언트의 도착률의 변동에 따라 표시하였다. 정적 분할보다

표 1 정적·동적 요청 부하 모델

파라미터 / 요청타입	서비스 시간	구성 비율
Dynamic Requests	700 msec.	0~1.0
Static Requests	100 msec.	0~1.0

표 2 시스템 파라미터(단위 : second)

서비스 요청 도착 시간 간격	Exp(평균 서비스 시간)
서버 수	10~30 대
계층별 요청 도착 비율 $[class_H, class_M, class_L]$	[0.2, 0.3, 0.5], [0.2, 0.5, 0.3], [0.3, 0.2, 0.5], [0.3, 0.5, 0.2], [0.5, 0.2, 0.3], [0.5, 0.3, 0.2]

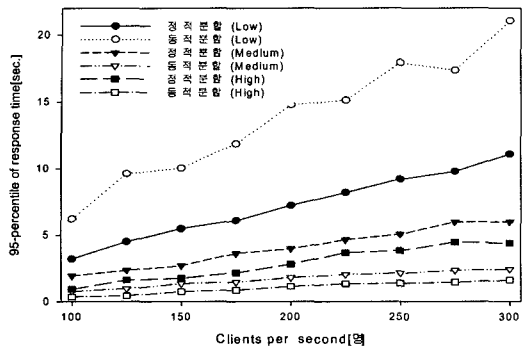


그림 4 도착률에 따른 분할기법 간의 응답시간 비교

는 동적 분할 기법의 응답시간이 High·Medium 계층에서 작게 나왔으며, 이는 동적 분할 기법이 사용자 요청을 고려하여 부하 변화에 맞게 대처하였기 때문이라 판단한다. 하지만 동적 분할 기법의 Low 계층은 과부하 상태시 상위 계층에게 서버를 먼저 주어야 하기 때문에, 정적 분할 기법에 비해 응답시간이 떨어지는 것을 알 수 있다.

그림 5에서는 Low 계층을 대상으로 정적·동적 분할 기법에 따른 거절된 요청 수를 클라이언트의 도착률의 변동에 따라 표시하였다. 동적 분할 기법은 정적 분할 기법보다 더 좋은 응답시간 성능을 나타내지만, 승인 제어를 고려할 경우, 동적 분할 기법에서 거절된 요청 수가 정적 분할 기법에서 거절된 요청 수 보다 많으며, 이는 동적 분할에서는 승인 제어를 통하여, 주로 Low 계층 요청을 거절하였기 때문이다.

그림 6은 High·Medium·Low 계층별 정적·동적 요청에 따른 응답시간을 서버 분할 방식에 따라 표시하였다. 동적 요청의 계층별 응답시간을 비교해보면, 동적 분할이 정적 분할에 비해 High 계층과 Medium 계층 응답시간이 상당히 좋은 반면, Low 계층의 요청은 정적 분할이 동적 분할에 비해 응답시간 성능이 우수했다. 이는 동적 분할에서 상위 계층이 과부하 상태일 경우

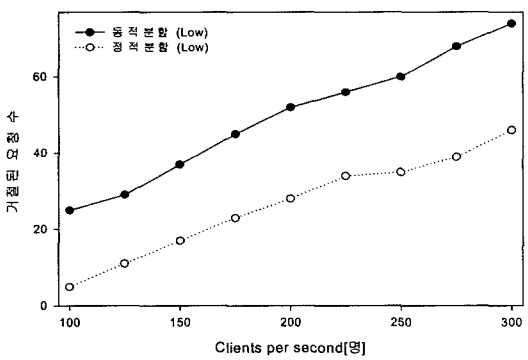


그림 5 도착률에 따른 거절된 요청의 수 비교

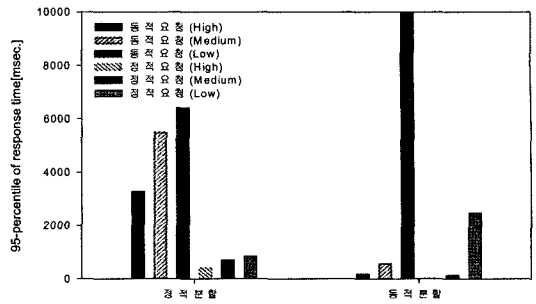


그림 6 분할 기법에 따른 정적·동적 요청의 응답시간 비교

Low 계층을 서비스하는 서버를 High·Medium 계층에 제공해주기 때문이다. 한편, 정적 요청은 변하지 않는 문서와 같은 형식의 파일이고, 동적 요청은 자주 변하면서 처리하는데 시간이 많이 걸리는 VOD와 같은 동영상 파일이라 대체적으로 정적 요청이 동적 요청에 비해 월등히 응답시간이 좋게 나왔다.

그림 7에는 High 계층을 대상으로한 정적·동적 분할 기법에 따른 서버 수 변화(서버 수 30대일 경우)를 요청률에 따라 표시하였다. 정적 분할했을 때의 서버 수는 고정적이라 처음 할당받은 서버 6대에서 변동이 없지만, 동적 분할했을 때는 요청률이 높아질수록 High 계층을 서비스하는 서버 수가 점점 증가하는 것을 알 수 있다. 이는 high 계층의 처리 용량이 부족해질 경우, 하위 계층을 서비스하고 있는 서버를 강제적으로 빌려오기 때문이다.

그림 8에서는 동적 분할에서 95-percentile 응답시간을 서버 수와 동적 요청 비율의 변동에 따라 표시하였다. 각 계층에서 서버 수가 증가할수록 응답시간이 작게 나왔으며, 동적 요청 비율이 증가할수록 처리시간이 길어져 응답시간 성능이 떨어지는 것을 알 수 있다. 한편, 서버 수 변화 보다는 동적 요청 비율 변화에 따라 시스

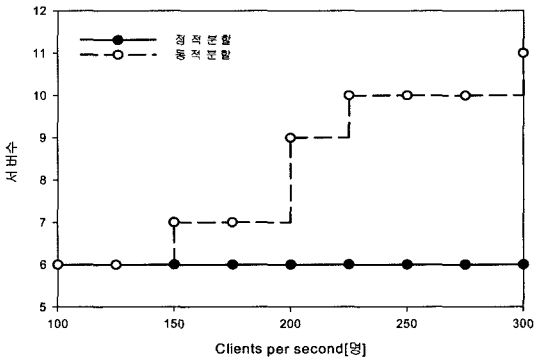
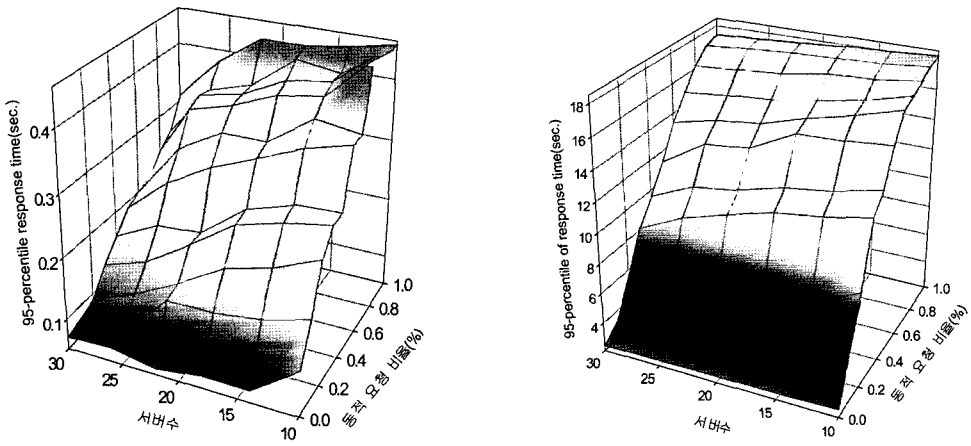


그림 7 도착률에 대한 High 계층을 서비스하는 서버 수 변화



(a) High 계층 (b) Low 계층

그림 8 서버 수와 동적 요청 구성 비율 변화에 따른 응답시간 비교

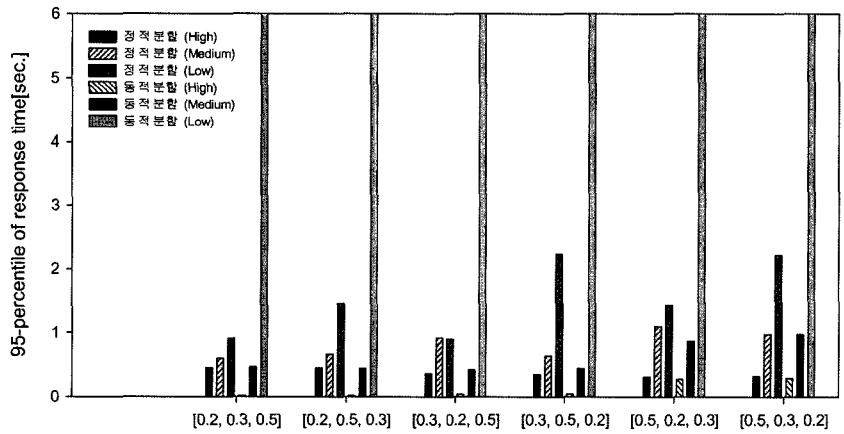


그림 9 계층별 요청 도착 비율 변화에 따른 응답시간 비교

템 성능 저하가 급격히 발생하므로, 동적 요청의 구성 비율이 중요한 성능 평가 요소임을 나타내고 있다.

그림 9는 High·Medium·Low 계층별 분할 방식에 따른 응답시간을 계층별 요청 도착 비율의 변동에 따라 표시하였다. Low 계층 비율이 높은 경우([0.3, 0.2, 0.5])와 Medium 계층 비율이 높은 경우([0.3, 0.5, 0.2])를 비교해 보면, 후자의 정적 분할(Low) 방식이 전자에 비해 응답시간 성능이 떨어지는데, 이는 정적 분할의 경우, 계층별 요청 비율의 변화를 반영하지 못하기 때문에 나타난 현상이라고 판단된다. 대체적으로, 동적 분할은 각 계층의 도착률에 따라 서버 수가 조절 되기 때문에, 정적 분할보다 응답시간 변동이 크지는 않았다. 하지만 High 계층 비율이 높을 때([0.5, 0.2, 0.3], [0.5, 0.3, 0.2])는 동적 분할(High)의 응답시간이 비교적 높게 나타나고 있는데, 이는 High 계층의 요청 비율이 0.5로 다른 경우에 비해 높았기 때문에 나타난 현상으로 판단된다.

6. 결 론

본 연구에서는 클라이언트 계층별 요청률에 따라 분할된 서버 노드들을 정적·동적으로 관리하고 다중 계층에서의 승인 제어를 함으로써, 웹 서비스의 응답시간 성능을 분석하였다. 이를 통해 클라이언트와 서비스 제공자간에 SLA를 고려한 서비스가 이루어질 수 있도록 하였으며, SLA를 만족시키기 위해 클라이언트 요청을 우선순위 계층으로 구분하여 고수준의 서비스를 요구하는 클라이언트에게 더 많은 컴퓨팅 자원을 할당하는 방식의 적용 가능성을 확인하였다. 추후에는 서비스 요청 시

간대에 따라 부하를 적절하게 조절할 수 있는 동적 분할 기법 및 로드발란서의 결함 허용도(e.g. 신뢰도, 가용도)에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] N. Bhatti and R. Friedrich, "Web server support for tiered services," IEEE Network, 13(5):pp. 64-71, 1999.

[2] Nikolaos Vasiliou, Hanan Lutfiyya, "Providing a differentiated quality of service in a World Wide Web server," SIGMETRICS Performance Evaluation Review 28(2): pp. 22-28, 2000.

[3] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

[4] V. P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet," IEEE Commun. Mag., vol. 36, pp. 152-64, May 1998.

[5] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Salvatore Tucci: "Mechanisms for quality of service in Web clusters," Computer Networks 37(6): pp. 761-771, 2001.

[6] E. Casalicchio and M. Colajanni, "A Client-aware Dispatching Algorithm for Web Clusters Providing Multiple Services," In Proceedings of the 10th International World Wide Web Conference, pp. 535-544, 2001.

[7] Cohen, A., S. Rangarajan, and H. Slye, "On the Performance of TCP Splicing for URL-aware Redirection," In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 117-125, 1999.

[8] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable Content-aware Request Distribution in Cluster-based Network Servers," Proceedings of the 2000 Annual Usenix Technical Conference, 2000.

[9] LVS Project, www.linuxvirtualserver.org

[10] http://www.alphal1.com/cluster/Cluster_Overview.html

[11] V. Cardellini, E. Casalicchio, M. Colajanni, and M. Mambelli, "Enhancing a Web-server Cluster with Quality of Service Mechanisms," Proceedings of 21st IEEE International Performance, Computing, and Communications Conference, pp. 205-212, 2002.

[12] V. Cardellini, E. Casalicchio, M. Colajanni, "A performance study of distributed architectures for the quality of Web services," Proc. of Hawaii Int'l Conf. on System Sciences (HICSS-34), Software Technology Track, Maui, Hawaii, IEEE Computer Society, pp. 3551-3560, 2001.

[13] Akhil Sahai, Vijay Machiraju, Mehmet Sayal, Aad P. A. van Moorsel, Fabio Casati: "Automated SLA Monitoring for Web Services," DSOM: pp. 28-41, 2002.

[14] Jian Pu, M. Mostofa Akbar, Eric Gowland, Gholamali C. Shoja, Eric G. Manning: "SLA Admission Controller for Reliable MPLS Networks," Applied Informatics : pp. 630-635, 2003.

[15] M. Aron, P. Druschel, and W. Zwaenepoel. "Cluster reserves: A mechanism forresource management in cluster-based network servers," In Proc. of ACM Sigmetrics 2000, pp. 90-101, 2000.

[16] X. Chen, P. Mohapatra, H. Chen, "An Admission Control Scheme for Predictable Server Response Time for Web Accesses," Proceedings of the 10th World Wide Web Conference, PP. 545-554, May 2001.

[17] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Marco Mambelli: "Web switch support for differentiated services," SIGMETRICS Performance Evaluation Review 29(2): pp. 14-19, 2001.

[18] H. Zhu, H. Tang, and T. Yang, "Demand-driven service differentiation in cluster-based network servers," In Proc. of IEEE Infocom 2001, pp. 679-688, 2001.

[19] M. Andreolini, E. Casalicchio, M. Colajanni and M. Mambelli, "A Cluster-Based Web System Providing Differentiated and Guaranteed Services," Cluster Computing, 7(1): pp. 7-19, 2004.

[20] L. Cherkasova and M. Karlsson, "Scalable web server cluster design with workload-aware request distribution strategy WARD," 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, WECWIS 2001, pp. 212 -221, 2001.

[21] 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 8th ACM Conference on Architecture Support for Programming Languages, pp. 205-216, 1998.

[22] http://www.clusterkorea.org/

[23] http://www.superuser.co.kr/

[24] 고현주외 2인, "SLA를 고려한 웹 서버 성능 분할 기법", 한국정보과학회, 제 31회 추계학술발표회, Vol. 31, No. 2, pp. 652-654, 2004.

[25] 황미선외 2인, "이질 웹 서비스 환경에서 차별화 서비스를 위한 부하분산 기법", 한국정보과학회, 제 31회 추계학술발표회, Vol. 31, No. 2, pp. 367-369, 2004.

고 현 주

2005년 안양대학교 컴퓨터학과(공학사)
2005년~현재 디오텔㈜ 소프트웨어팀 연구원. 관심분야는 Embedded System, Cluster Computing

박 기 진

1989년 한양대학교 산업공학과(공학사) 1991년 포항공과대학교 산업공학과(공학석사). 1991년~1997년 삼성종합기술원, 삼성전자(주) 소프트웨어센터 선임연구원. 1997년~2001년 아주대학교 컴퓨터공학과(공학박사). 2001년~2002년 한국전자통신연구원 네트워크장비시험센터 선임연구원. 2002년~2004년 안양대학교 컴퓨터학과 전임강사. 2004년~현재 아주대학교 공과대학 산업정보시스템공학부 조교수. 관심분야는 Dependable Embedded Computing, Intrusion Tolerance Systems, Cluster Computing

박 미 선

2005년 안양대학교 컴퓨터학과(공학사) 2005년~현재 GMD Technology, Nextouch (주) 소프트웨어 연구원. 관심분야는 Grid Computing, Global System of Mobile Communication