SLA를 고려한 클러스터 기반 고가용성 이질 웹 시스템의 성능 분리 기법

강 창 훈[†] · 박 기 진[††] · 김 성 수[†††]

요 약

인터넷 사용자들에게 신뢰성 있는 차별화된 고품질의 서비스를 제공하기 위해서, 클러스터 기반 웹 시스템에서 클라이언트의 계층별 요청 형태에 따라 서비스 제공자와 클라이언트간의 서비스 계약 조건(SLA: Service Level Agreement)을 만족시킬 수 있는 차별화 서비스(Differentiated Service)에 대한 연구가 활발하다. 본 논문에서는 클라이언트의 요청을 클러스터를 구성하는 각각의 웹 서버에 분배하는 이중화 웹 스위치와 서버들을 서로 다른 기능 도메인(Function Domains)으로 구분한 이질(Heterogeneous) 웹 시스템의 성능분할(Performance Isolation) 및 승인제어(Admission Control) 기법을 연구하였으며, 다양한 실험을 통해 제안된 구조와 기법이 적용된 웹 시스템의 가용도(Availability), 응답시간 및 승인제어 성능을 분석하였다.

키워드 : SLA, 이질 웹 서버, 성능 분리, 승인제어

A Performance Isolation Mechanism Considering SLA in High Available Heterogeneous Cluster Web Systems

Changhoon Kang[†] · Kiejin Park[††] · Sungsoo Kim[†††]

ABSTRACT

To provide reliable differentiated service for internet clients, many studies on differentiated service are widely done. Accordingto the types of the request of the clients in cluster-based web systems, SLA (Service Level Agreement) between service provider and the clients is considered. In this paper, we study primary-backup web switch in the web systems that distribute the request of clients to each of web servers and also study performance isolation and admission control schemes of heterogeneous web systems which distinguish the servers into each different function domains. We analyze the availability, the response time and the performance of admission control through various simulated experiments on the proposed web systems.

Key Words : SLA, Heterogeneous Web Server, Performance Isolation, Admission Control

1. 서 론

인터넷 사용자의 증가와 멀티미디어 데이터를 포함한 웹 서비스의 팽창으로 인해, 웹 시스템 성능과 가용성을 개선하려는 시도가 활발하다. 웹 시스템의 성능을 향상시키는 방법으로 과거에는 고성능의 프로세서를 여러 개 장착하는 병렬처리 기법을 사용하였으나, 물리적인 확장의 한계와 성능 대비 투자 비용의 증가로 효율성이 매우 떨어지므로, 웹 시스템을 구성하는 웹 서버를 클러스터링하는 기법이 주로 채택되고 있다[1, 2]. 클러스터 기반 웹 서버 환경에서는 사용자 요청에 대한 응답 시간을 최소화하기 위해, 클러스터를 구성하고 있는 서버의 성능분리를 통한 웹 서버 간의 부하 분산(Load Balancing) 메커니즘이 중요하다. 한편, 웹 시스템은 인터넷 사용자에게 기본적인 서비스 제공은 물론, 고품질의 차별화된 서비스를 제공해야 할 필요성이 증가하고 있으며, 현재까지는 주로 네트워크 계층을 대상으로 한 QoS(Quality of Services) 기법이 연구되었으나, 네트워크 수준의 QoS만으로는 양단간 QoS를 지원할 수 없는 문제점이 발생하므로, QoS의 기본원리를 포함한 웹 서버 단계의 QoWS(Quality of Web-based Services)에 관한 분석 및 메커니즘에 대한 연구가 요청되고 있다.

최근에는 사용자와 서비스 제공자간의 서비스 품질을 보장해 주는 정책(SLA : Service Level Agreement)을 고려한 보다 상위 계층의 QoS에 관한 연구가 활발하다[3, 4]. SLA는 서비스 제공자와 대상 고객간 계약으로, 제공될 서비스와 그와 연관된 조건들을 사전에 약속하여 그 만큼의 서비스를 제공하는 것이며, 차별화된 서비스를 통해 고객과 서비스 제공자간의 SLA 보장이

※ 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음. (KRF-2003-003-D00331)
† 정 회 원 : 극동정보대학 방송영상미디어과 부교수
†† 정 회 원 : 아주대학교 산업정보시스템공학부 조교수
††† 종신회원 : 아주대학교 정보통신전문대학원 교수
논문접수 : 2005년 3월 21일, 심사완료 : 2005년 9월 30일

가능하다[5, 6, 7]. 기본적인 서비스만을 제공하는 웹 시스템은 과부하 상황에서, 우선순위가 높은 사용자의 요청이 거절되는 문제가 발생될 수 있으며, 이를 해결하기 위해 웹 서버의 성능 분리(Performance Isolation) 개념을 적용할 수 있다. 성능 분리는 웹 시스템에 들어오는 사용자 요청을 분류하여 계층별 큐로 보내고, 웹 시스템의 과부하 발생시 상위 계층의 사용자 요청을 받아들이고 하위 계층의 사용자 요청을 거절하여 차별화된 서비스를 제공하는 기본적 방식을 의미한다[8]. 예를 들어 e-비즈니스 사이트에서는 상품 목록을 검색하는 여러 사용자의 요청 보다 결제 처리를 요청하는 한 명의 사용자의 요청 처리가 중요하며, 상위 계층 사용자를 위한 차별화된 서비스의 제공으로 리소스의 사용 효율을 높일 수 있다. 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.

일반적으로 클러스터링된 웹 서버 노드들을 기능 도메인(Function Domains)에 따라 구분할 수 있다. 특히 e-비즈니스를 처리하는 사이트는 메일, 문서, VOD(Video On Demand), 사용자 정보관리 및 결제 등의 기능 도메인으로 나눌 수 있으며, 각각의 도메인은 사이트 관리기에 의해 실질적인 특정 서버로 매핑된다. 하지만 기존의 성능 분리 개념을 도입한 부하 분산 기법들은 클러스터 웹 시스템의 모든 노드가 단일 도메인으로 이루어진 동질(Homogeneous) 웹 서버 환경을 가정한 방식이며, 서로 다른 기능 도메인을 갖는 이질(Heterogeneous) 웹 서버 환경에 대한 고려는 없었다. 그러나 인터넷에서 웹 서버의 역할이 매우 중요해지고 해당 사이트의 규모가 커지며 사용자들과의 SLA에 따라 만족할 만한 서비스를 보장하기 위해 서로 다른 컨텐츠를 제공하는 서버들을 별도로 그룹핑하여 운영하는 이질 웹 서버 환경이 증대되고 있는 현실이다. 한편, 클라이언트의 요청을 처리할 서버를 결정하는 역할은 웹 스위치(Web Switch)가 담당하게 되는데, 웹 스위치가 하드웨어/소프트웨어적인 결함으로 인한 서비스 중단 또는 외부로부터의 악의적 공격 대상이 되어 원래 제공해야 하는 정상적인 서비스를 제공하지 못할 경우, 결국에는 웹 시스템 전체의 다운을 초래할 가능성이 높다.

본 논문의 2장에서는 관련 연구를 살펴보고, 3장에서는 주-백업 웹 스위치로 구성된 웹 시스템 구조를 제시하고, 4장에서는 이질 웹 서버의 성능 분리 기법을 제시하며 5장에서는 제시한 구조의 성능을 평가하였고 6장에서는 결론을 내렸다.

2. 관련 연구

클러스터 기법은 시스템의 성능을 향상시키기 위해 여러 개의 서버를 하나로 연결하는 방법이며, 클러스터 기반 웹 시스템에서 사용자의 요청을 효율적으로 분배하기 위한 연구들은 Client-Side 접근 방식, TCP 연결 라우팅 방식 및 HTTP 리다이렉션 (Redirection) 방식, DNS(Domain Name Server) 기반의 접근 방식으로 분류할 수 있다[9]. 이러한 방식들은 Layer-4 부하 분배 방식으로 TCP/IP 레벨에서 동작하며 HTTP 요구가 보내지기 전에 TCP/IP 연결 설정이 이루어지기 때문에 서비스 노드에 대한 선택이 요구된 내용에 의해 결정될 수 없다[10]. 이에 비해 최근에 각광 받고 있는 Layer-7 부하 분배 방식은 서비스 노드를 결정하기 전에 HTTP 요구를 분석할 수 있어 사용자 요구의 내용에 대해 상세한 정보를 고려한 분배가 가능하다[11]. 즉 내용 기반 요청 분배(Content Based Request Distribution)는 기억 장치의 확장 가능성과 특정 목적의 서비스 노드 운영, 서비스 노드의 메인 메모리 캐쉬에서 향상된 적중률 등의 장점을 가진다. LARD(Locality-Aware Request Distribution)[12, 13] 기법은 사용자의 요청 내용에 따라 부하를 분배하며, 서비스 노드의 메인 메모리 캐쉬 적중률을 향상시켰으나 정적 요청만을 고려한 점과 작업 부하에 가중치를 설정해야 하는 문제점이 있다.

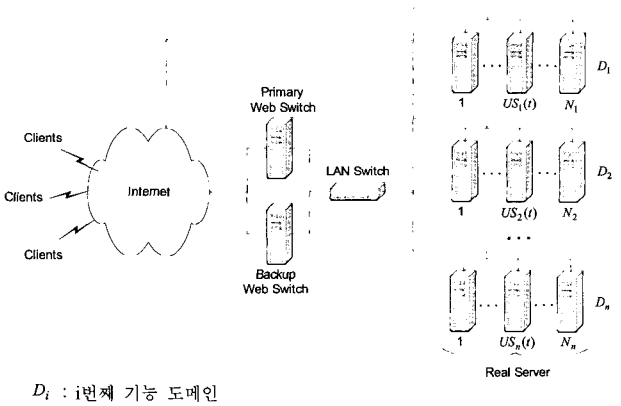
한편, 웹 시스템의 SLA를 보장하기 위해, 시스템 구조를 통한 방법과 차별화 서비스를 통한 방법이 연구되고 있으며, 전자는 시스템 관리자에 의한 서버 노드의 추가 또는 제거 작업이 이루어짐으로 인해, 구조의 변화가 쉽지 않아서 효율적이지 못하며, 후자의 방법들 중 DDSD(Demand-driven Service Differentiation)[14] 기법은 사용자의 요청에 따라 CPU와 디스크 I/O의 용량을 할당하고 동적 요구가 많은 상황에 적합한 기술이며, Dynamic Partitioning 기법[3]은 서버 부하량에 따라 서버 노드를 동적으로 분할하여 차별화된 서비스를 제공한다. 하지만 이러한 기법들은 일정한 시간 간격으로 서버 노드를 분할하기 때문에 갑작스런 과부하 상태를 처리하기에는 부적절하며, 특히 Dynamic Partitioning 기법은 정적 요청을 전혀 고려하지 않고 동적 요청만을 고려한 기법으로 정적 요구량이 많아지면 시스템의 성능 저하가 발생한다.

클라이언트의 요청을 두 개의 계층(High Set, Low Set)으로 분리하여, 차별화된 서비스를 제공하려는 시도가 있었으며[11, 15]는 이 방법을 일반화시킨 개념으로, 클라이언트의 요청을n개의 계층으로 분할하여 각 요청들을 분배하였다. 기존의 연구들은 각 요청을 해당 서버로 분배해주는 웹 스위치가 안정적일 것이라는 가정을 적용하고 있으나 웹 스위치에 결함이 발생하게 되면 클러스터의 요청을 분배할 수 없게 되어, 해당 사이트는 서비스 불능 상태가 될 가능성이 높다.

따라서 본 논문에서는 클러스터 기반 웹 시스템의 웹 스위치가 내부적 결함이나 외부적인 침입(Intrusion)에 의해 기능이 정지되는 가능성을 줄이기 위하여, 주 웹 스위치에 이상이 발생하였을 때 백업 웹 스위치로 대치하는 여분(Redundancy) 개념을 도입한 주-백업 웹 스위치 부하 분배 클러스터 기반 웹 시스템의 구조를 제안하였다. 또한 기존의 방식들은 클러스터 웹 시스템의 모든 서버가 단일 도메인으로 이루어진 동질 웹 서버 환경을 가정한 방식이었으나, 본 논문에서는 웹 서버들이 서로 다른 기능 도메인으로 구성된 이질 웹 서버 구조의 성능분할 및 승인제어(Admission Control) 기법을 제시하였으며, 다양한 실험을 통해 제안된 기법의 가용도(Availability), SLA를 고려한 응답시간 및 승인제어 성능을 분석하였다

3. 고가용성 이질 웹 시스템 구조

(그림 1)은 사용자의 요청 내용을 계층별로 분류하여 처리할 수 있는 Layer-7 스위치 기반 고가용성 이질 웹 시스템의 구조를



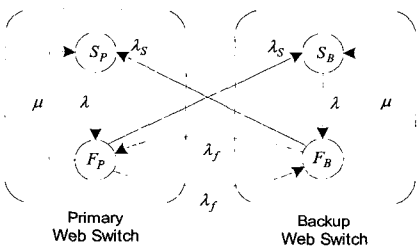
D_i : i번째 기능 도메인
$US_i(t)$: 임의의 시간 t에서 D_i의 사용자 계층별 요청을 처리하는 서버 구분 경계 값

(그림 1) 주-백업 웹 스위치로 구성된 이질 웹 시스템 구조

보여준다. 본 논문에서는 사용자의 요청처리 결과를 담당 서버에서 사용자에게 직접 전달하는 One-way 웹 서버 구조[16]를 채택하였으며, 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 재 작성되기 때문에 One-way 구조에 비해 스위치의 오버헤드가 크다.

웹 스위치는 실제 서버(Real Server)들 간에 작업들을 고루 분배하는 기능을 수행하며, 과중하게 부하를 받고 있는 서버들이 다른 서버들에게 작업을 전달하거나, 유휴 상태에 있는 서버의 작업 분배 요청을 처리한다[15]. 웹 스위치는 지속적으로 부하분배 서비스를 제공해야 하므로, 웹 스위치의 과도한 집중이나, 시스템 이상 혹은 외부의 침입으로 다운된 경우, 대기하고 있던 백업 웹 스위치가 주 웹 스위치의 역할을 대신 수행시킬 수 있어야 한다. 백업 웹 스위치는 주 웹 스위치가 작동하지 않으면 대체되는 웹 스위치로, 주 웹 스위치와 연결되어 각각 Heartbeat 데몬을 가동시켜 주기적으로 서로 Heartbeat을 주고받으며 Heartbeat을 일정 시간 내에 받지 못하면 상대편 웹 스위치가 다운된 것으로 판단한다. 이와 같은 구조의 고가용성 웹 서버는 대규모 서비스를 제공하기 위한 목적으로 제작되며 주로 인터넷 비즈니스 등에 활용 가치가 높다.

이질 웹 시스템은 주 웹 스위치, 백업 웹 스위치와 $\sum_{i=1}^{n} N_i$개의 실제 서버들로 구성되며, 이질 웹 시스템은 서버들을 기능에 따라 n개의 도메인(예 : D_1:메일서버 집합, D_2:파일서버 집합,…, D_n : VOD서버 집합)으로 나눈다. i번째 기능 도메인(D_i)은 N_i개의 서버들로 구성되며, 각 기능 도메인은 임의의 시간 t에서 $US_i(t)$개의 상위 계층(High Class) 서버와 $(N_i - US_i(t))$개의 하위 계층(Low Class) 서버로 구분되어, 해당 계층에 속하는 사용자의 요청 처리를 담당한다.



(그림 2) 주-백업 웹 스위치 상태전이도

(그림 2)는 주-백업 웹 스위치 시스템의 상태 전이도(State Transition Diagram)를 표시한다. 주-백업 웹 스위치 시스템의 주 웹 스위치와 백업 웹 스위치는 각각 S(Safe State)와 F(Failure State) 상태로 구성된다. 주 웹 스위치는 안정상태(S_P)에서 시스템 내부의 결함이나 외부의 공격 혹은 침입에 의해 λ의 고장률로 시스템이 고장 나게 된다. 또한 주 웹 스위치가 고장상태(F_P)에서 백업 웹 스위치가 안정상태(S_B)에 있다면 백업 웹 스위치로 대치되어 부하 분배의 기능을 수행하고, 자체 시스템이 수리되면 자체 시스템이 주 웹 스위치의 기능을 수행한다. 만약 백업 웹 스위치도 고장 상태에 있는 경우, 둘 중 하나 혹은 두 대 모두 수리될 때까지 부하 분배 기능을 수행하지 못함으로 인해, 해당 클러스터 웹 서버가 고장 상태로 존재하게 된다. 본 논문에서 제안한 웹 스위치 시스템에 사용된 가정은 다음과 같다.

- 주-백업 웹 스위치는 Hot Standby 방식으로 대기 상태에 있다.
- 주 웹 스위치와 백업 웹 스위치의 평균 고장($1/\lambda$) 및 수리($1/\mu$) 시간 간격은 지수 분포를 따른다.
- 주 웹 스위치에서 백업 웹 스위치로 작업전이(Switch-over) 시간은 평균값이 $1/\lambda_S$인 지수분포를 따른다.
- 임의의 웹 스위치가 고장인 상태(F_P 혹은 F_B)에서 또 다른 웹 스위치도 고장인 상태로 되는 즉, 동시에 고장 나는 시간 간격은 평균값이 $1/\lambda_f$인 지수분포를 따른다.

(그림 2)의 상태 전이에 해당하는 평형상태(Steady State)에서의 균형식(Balance Equation)은 식(1)~식(4)와 같다.

$$\lambda P_{S_P} = \mu P_{F_P} + \lambda_S P_{F_B} \tag{1}$$

$$\lambda P_{S_B} = \mu P_{F_B} + \lambda_S P_{F_P} \tag{2}$$

$$(\mu + \lambda_S + \lambda_f) P_{F_P} = \lambda P_{S_P} + \lambda_f P_{F_B} \tag{3}$$

$$(\mu + \lambda_S + \lambda_f) P_{F_B} = \lambda P_{S_B} + \lambda_f P_{F_P} \tag{4}$$

위의 균형식과 각 상태에서 머물 확률의 총합이 1이 되는 식을 결합한 $P_{S_P} + P_{F_P} + P_{S_B} + P_{F_B} = 1$ 풀면, 시스템이 평형일 때 각 상태에서 머물 확률을 다음과 같이(식(5)~식(8) 참조) 얻을 수 있다.

$$P_{S_P} = \frac{\mu + \lambda_S}{2(\mu + \lambda_S) + \lambda} \tag{5}$$

$$P_{F_P} = \frac{\lambda}{2(\mu + \lambda_S) + \lambda} \tag{6}$$

$$P_{S_B} = P_{S_P} \tag{7}$$

$$P_{F_B} = P_{F_P} \tag{8}$$

4. 이질 웹 서버의 성능 분리 기법

이질 웹 시스템((그림 1) 참조)에 입력되는 클라이언트의 요청은 웹 스위치에서 성능 분리 과정을 통해 실제 서버들로 보내지며, 성능 분리는 클라이언트 요청의 특성에 맞추어 적합한 기능 도메인의 적합한 서버 클래스에 작업을 분배하는 것을 의미한다. 즉, 웹 스위치에 입력되는 클라이언트의 요청을 분류하여 기능 도메인의 해당 서버 큐로 보내는 작업이다. 실제 서버들은 여러 개의 기능 도메인으로 분류되고, 각 기능 도메인은 상위 계층과 하위 계층으로 구분되며, 상위 계층은 동적 요청(Dynamic Request)이나 유료 사용자를 서비스하며, 하위 계층은 정적 요청(Static Request)이나 무료 사용자들을 대상으로 한다. 동적 요청은 자주 변하면서 처리 시간을 많이 요구하는 서비스로 데이터베이스 질의나 VOD 같은 동영상 파일에 대한 처리를 필요로 한다.

4.1 서비스 분류(Service Classification)

서비스 분류는 웹 스위치에 입력되는 클라이언트의 요청을 분류하여 계층별 큐로 보내는 작업을 의미하며, (그림 1)에서 i 번째 기능 도메인 웹 서버($D_i = \{1,2,...,N_n\}$)에 사용자의 요청이 들어오면, 그 요청의 우선 순위를 확인한 후에 그에 적합하게 서버를 할당한다. 이 과정을 수식으로 나타내면 식 (9)와 같다. 여기서, $HS_i(t)$와 $LS_i(t)$는 임의의 시간 t에서 각 도메인 i에서 우선 순위에 따라 각각 상위 계층과 일반 계층에 할당되는 서버 집합이며, 서버 분할 기준선을 의미하는 $US_i(t)$는 임의의 시간 t에서 도메인 i에 속하는 사용자 계층별 요청을 처리하는 서버 구분 경계 값이다. 즉, $US_i(t)$는 특정한 기능 도메인을 서비스하는 서버 노드 집합을 대상으로 계층별 요청을 우선 순위별로 처리하는 서버 성능 분리 기준을 말한다.

$$\begin{aligned} &D_1 : HS_1(t) = \{1,...,US_1(t)\},\ LS_1(t) = \{US_1(t)+1,...,N_1\} \\ &D_2 : HS_2(t) = \{1,...,US_2(t)\},\ LS_2(t) = \{US_2(t)+1,...,N_2\} \\ &\qquad\vdots \\ &D_n : HS_n(t) = \{1,...,US_n(t)\},\ LS_n(t) = \{US_n(t)+1,...,N_n\} \\ &\text{전체 서버 수} = \sum_{i=1}^{n} N_i \end{aligned} \tag{9}$$

4.2 정적 서버 성능 분리

정적 서버 성능 분리는 기존의 계층별 클라이언트의 요청률만을 가지고, 서버 구분 경계값($US_i(t)$)을 구하는 방식으로 최초 가동시에 각각의 기능 도메인별 클라이언트 계층별로 고정적인 서버 수가 할당되며, 한번 할당된 서버 노드는 클라이언트의 계층별 요청이 변해도 수정되지 않는 방식이다. 사용자 요청의 도착률과 서버 노드의 적재 상태에 따라 서버 노드를 변동할 수 없기 때문에, 한 계층에 할당된 서버 노드가 과부하 상태이더라도 다른 계층의 서버는 유휴 상태로 서버 노드의 자원 낭비가 일어나는 문제점이 있다.

$US_i(t)$를 계산하는 수식을 식(10)에 표시하였으며, 여기서 ρ_i는 전체 요청 수와 상위 계층에 속하는 사용자의 요청 수의 비율이고, T_{N_i}은 i 번째 도메인의 서버 N_i 대에서 허용 가능한 최대 지연 시간이다. 또한, $MaxConn(T_{N_i})$은 지연 시간 T_{N_i}을 만족시키는 최대 접속 수이며, T_H는 상위 계층의 SLA를 고려한 임계 시간(Threshold: $T_H < T_{N_i}$)이다. 각 도메인의 상위 계층의 서버 수를 구하기 위해서 ρ_i와 도메인 i의 전체 서버 수를 곱하며, T_{N_i}에서의 이상적인 최대 접속 수를 T_H 에서의 실제 최대 접속 수로 나누어 구한, 최대 접속 비율을 곱하여 정확한 값을 도출하고자 하였다.

$$US_i(t) = \lceil \rho_i N_i \rceil \cdot \frac{MaxConn(T_{N_i})}{MaxConn(T_H)} \tag{10}$$

4.3 동적 서버 성능 분리

동적 서버 성능 분리는 상위 계층 사용자의 SLA를 만족시켜주기 위하여, 계층별 부하량에 따라 서버 노드를 동적으로 분할하는 방법으로, 상위 계층을 서비스 하는 서버 노드에 과부하가 걸렸을 경우 하위 계층의 서버를 추가로 이용하고, 과부하 상황이 해소되면 다시 서버를 반납하는 개념으로 부하 변화에 맞게 대처할 수 있는 방법이다. 사용자 요청의 도착률이나 필요에 따라 동적으로 서버 노드를 할당하여 유휴 서버 노드를 줄여줌으로써 높은 자원 활용을 제공한다.

(그림 3)은 동적 서버 성능 분리 알고리즘을 나타내며, $SumLoad_{HS_i}(t)$는 $HS_i(t-1)$에서의 서버 부하(Load)의 합이며, 사용자의 최대 접속 수($MaxConn$)를 의미한다. 즉 상위 계층의 부하 합이 작업 능력을 넘어서면, 과부하 상태로 하위 계층의 서버를 가져오고, 부하 량이 적어질 경우 가져온 서버를 반납하는 방식을 따르며, 임의의 시간 t에서 $US_i(t)$가 가질 수 있는 최소의 서버 수는 $t=0$일 때 서버 수와 같아야 하며, 최대의 서버 수는 $N_i - 1$의 가정을 두었다. 이는 사용자 계층별로 할당된

$$\begin{aligned} &if\left(SumLoad_{HS_i}(t) > US_i(t-1)\cdot MaxConn(T_H)\right) \\ &\quad US_i(t) = US_i(t-1) + 1; \\ &else\ if\left(SumLoad_{HS_i}(t) < (US_i(t-1)-1)\cdot MaxConn(T_H)\right) \\ &\quad US_i(t) = US_i(t-1) - 1; \end{aligned}$$

(그림 3) 동적 서버 성능 분리 알고리즘

서버 수가 한 대도 없는 최악의 성능 분리를 방지하여, 최소한 한 대의 서버가 각 계층에 할당되는 것을 보장한다.

4.4 승인제어(Admission Control)

서비스 노드의 과부하 발생시 사용자의 우선순위에 따라 상위 계층의 사용자 요청을 받아들이고 하위 계층의 사용자 요청을 거절하는 방식으로 차별화된 서비스를 제공함으로 사용자의 SLA를 보장하고자 하였다. 웹 서버의 과부하시 하위 계층 요청의 거절 여부를 식 (11)에 의해 결정하며, 여기서 파라미터 γ은 $LS_i(t)$에서 서버의 부하 상태와 탈락된 요청의 수 사이에서 거절 여부를 조절해 주는 비율이다. 예를 들어 식 (11)이 만족되면, i 번째 도메인 하위 계층의 웹 서버가 과부하 상태로 하위 계층의 요청이 거절된다.

$$SumLoad_{LS_i}(t) > \gamma \cdot (N_i - US_i(t)) \cdot MaxConn(T_H),$$
$$i = 1,2,3,\cdots,n \quad (11)$$

5. 실험 및 분석

실험은 두 가지 과정으로 나누어 실시하였다. 단일 웹 스위치 구조와 이중 웹 스위치 구조인 주-백업 웹 스위치 운영 시 웹 스위치의 가용도 변화를 고장률과 수리율에 따라 실험 하였고, 이질 웹 서버 환경에서 사용자 계층에 따른 성능 분리 기법의 성능을 분석하기 위해, 웹 서버 클러스터 시스템을 구성하는 서버 대수, 사용자 계층, 작업 수행시간 등 다양한 입력 파라미터에 따른 응답 시간의 변화를 실험하여 성능을 분석하였다.

〈표 1〉 웹 스위치 운영 파라미터

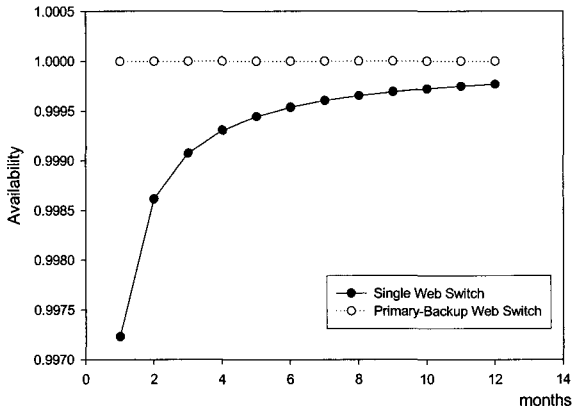
고장률(λ)	2 time / year	1/(6*30*24*3600)
수리율(μ)	1 time / 2 hours	1/(2*3600)
작업전이율(λ_S)	1 time / 10 sec.	1/(10)
동시고장률(λ_f)	1 time / 2 years	1/(24*30*24*3600)

5.1 웹 스위치의 가용도 분석

두 개의 웹 스위치가 모두 작동하거나 한 대의 웹 스위치가 작동하는 경우 웹 스위치 시스템은 가용하다고 볼 수 있다. 즉, 두 개의 웹 스위치가 모두 고장 난 경우를 제외하면 가용한 상태라고 볼 수 있다. 따라서 실험에서 제시한 웹 스위치 시스템의 가용도 정의는 다음과 같다.

$$Availability = 1 - (P_{F_P} + P_{F_B}) \quad (12)$$

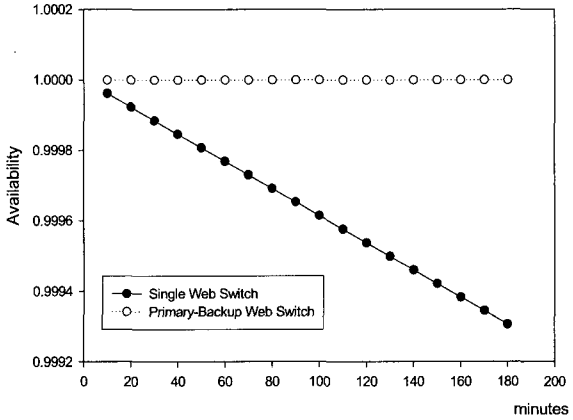
실험은 PC 환경(Pentium 4)에서 C 언어를 사용하여 수행하였으며, 또한 실험에 사용한 웹 스위치 운영 파라미터는 <표 1>과 같다[17]. 웹 스위치의 고장(λ)은 6개월에 1회. 고장 수리에 2시간이 소요되며. 웹 스위치를 대체하는 작업전이 시간은 10초가 소요된다. 또한 동시에 두 서버가 고장 나는 경우는 2년에 1회이다.



(그림 4) 고장률 변화에 따른 웹 스위치 가용도

(그림 4)는 한 개의 웹 스위치를 사용한 구조와 주-백업 웹 스위치 구조간 고장률 변화에 따라 가용도를 측정한 결과이다. 한 개의 웹 스위치 사용시 고장이 자주 일어나면(즉, 고장 시간 간격이 작을 때) 가용도가 현저히 떨어지고, 고장률이 낮아질수록 가용도가 증가하는 반면, 주-백업 웹 스위치 모델은 고장률 변화에 거의 무관하게 가용도가 1에 가까워(0.999998) 항상 가용한 상태라고 볼 수 있다. 이런 결과는 주-백업 웹 스위치 구조는 한대의 웹 스위치 고장시 다른 여분의 웹 스위치로 대치되기 때문이다.

(그림 5)는 수리율에 따른 가용도 변화로써 한 개의 웹 스위치인 경우는 수리 시간이 많이 걸릴수록 가용도는 떨어지는 반면, 주-백업 웹 스위치 구조는 수리 시간에 거의 무관하게 가용도를 유지하는데(0.9999994), 이는 한대의 웹 스위치가 수리되는 동안에 여분의 백업 스위치가 부하 분배 기능을 계속적으로 수행하기 때문이다.



(그림 5) 수리율 변화에 따른 웹 스위치 가용도

5.2 이질 웹 서버의 성능 분리 분석

이질 웹 서버 환경에서 사용자 계층에 따른 동적 서버 분할 기법의 성능을 분석하기 위해, 웹 서버 클러스터 시스템을 구성하는 서버 대수, 사용자 계층, 작업 수행시간 등 다양한 입력 파라미터에 따른 응답 시간의 변화에 대한 그래프를 작성하였으며, 성능 평가를 위한 기능 도메인 부하 파라미터는 표 2, 3과 같다[11]. 실험은 PC 환경(Pentium 4)에서 C 언어를 사용하여 수행하였으며, 시스템내 각 자원으로의 클라이언트 요청 도착 시간 간격을 지수(Exponential) 분포를 사용하여 표현하였다. 클라이언트의 서비스 요청 사건이 발생되면, 이는 다시 서비스 시간이 상이한 동적 요청과 정적 요청을 랜덤한 비율로 발생시킨다. 기본적으로 클라이언트 요청은 동적 요청 20%, 정적 요청 80% 비율로 구성되어 있다.

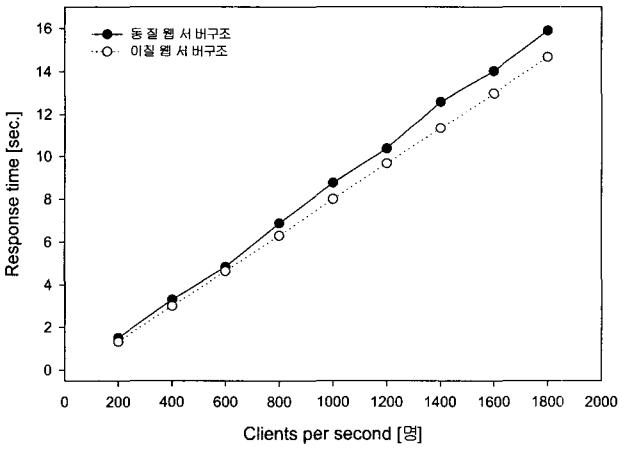
〈표 2〉 기능 도메인 부하 파라미터

기능적 도메인(D_i)	서비스 시간(S_i)	요청비율		
		Low 높을때	Low/High 비슷할때	High 높을때
D_1 : Low Intensive (e.g. Static Text)	100 msec.	0.75	0.40	0.15
D_2 : Medium Intensive (e.g. Dynamic Text)	300 msec.	0.15	0.10	0.10
D_3 : High Intensive (e.g. VOD)	600 msec.	0.10	0.50	0.75

〈표 3〉 이질 웹 시스템 파라미터

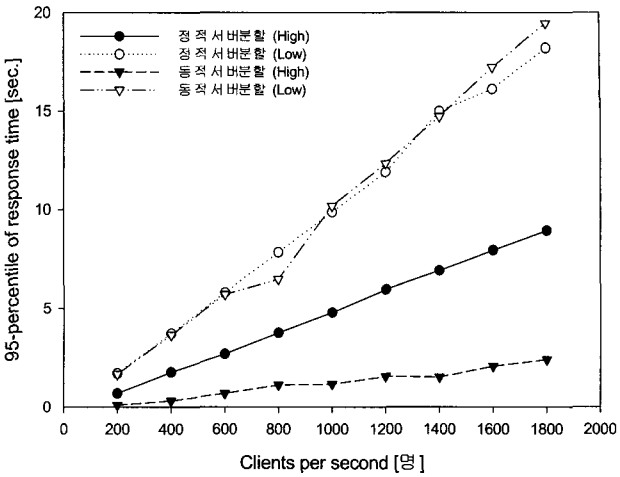
서비스 요청 도착 시간 간격	Exp(평균서비스시간)
도메인 D_i 의 서버 대 수	$Max\left(1,\left\lfloor N\times\frac{S_i}{Sum(S_i)}\right\rfloor\right),\ N=20,30,\cdots,60$
사용자 요청비율 $[class_H, class_L]$	$class_H=\{0.0\sim1.0\},\ class_L=\{0.0\sim1.0\}$
시간(t)	500 msec, 3000 msec

본 실험에서 고려한 3종류의 기능 도메인(D_1, D_2, D_3)은 Low, Medium, High Intensive로 각각 100, 300, 600 msec. 의 서비스 시간을 갖고, 사용자 요청의 빈도를 세가지로 Low Intensive의 요청이 많을 때와 Low, High Intensive가 비슷할 때, High Intensive가 많을 때로 구분하였다. 서비스 요청 도착 시간 간격은 서비스 시간(S_i)의 평균을 모수로 하는 지수 분포를 사용하였으며, 도메인(D_i)을 서비스 하는 서버 대 수는 전체 서버 수에 서비스 시간 비율을 곱하여 구하였다. 도메인은 최소 1대 이상의 서버 수를 가져야 하며, 여분 서버의 유휴 상태를 방지하기 위해 $N-\sum_{i=1}^{n}Max\left(1,\left\lfloor N\times\frac{S_i}{Sum(S_i)}\right\rfloor\right)$가 양수일 경우 여분 서버를 최상위 도메인(e.g. High Intensive)에 강제로 할당시켰다. 사용자 요청 비율은 상위 계층과 하위 계층으로 0.0~1.0 구간의 경우의 수를 모두 적용하였으며, 시간(t)는 동적 서버 분할 주기와 승인제어 검사 주기를 말하며 주기가 짧을 경우(500 msec.)와 비교적 긴 경우(3000 msec.) 두 가지를 고려하였다.



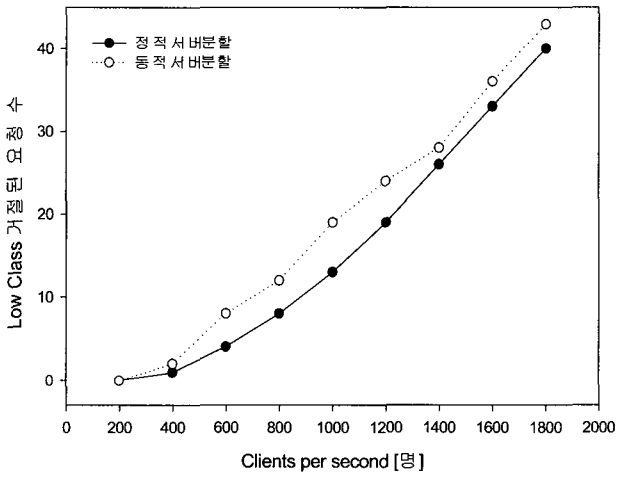
(그림 6) 사용자 도착률에 따른 동질·이질 웹 서버 구조의 응답시간비교

(그림 6)은 사용자 도착률에 따른 동질·이질 웹 서버 구조의 응답시간 변화를 표시하고 있다. 사용자 요청 수가 증가함에 따라 이질 웹 서버의 응답시간이 동질 웹 서버 일 경우보다 빠르게 나타나는 것을 알 수 있다. 이는 이질 웹 서버 구조하의 사용자 요청은 각 기능적 도메인 서버로 이동하여 기능적 도메인 간의 도착시간에 관계없이 작업을 동시에 수행하는 반면에, 동질 웹 서버 구조는 사용자 요청이 도착한 순서대로 작업을 수행하여 먼저 도착한 작업이 끝난 후 다음 요청을 수행하기 때문이다.



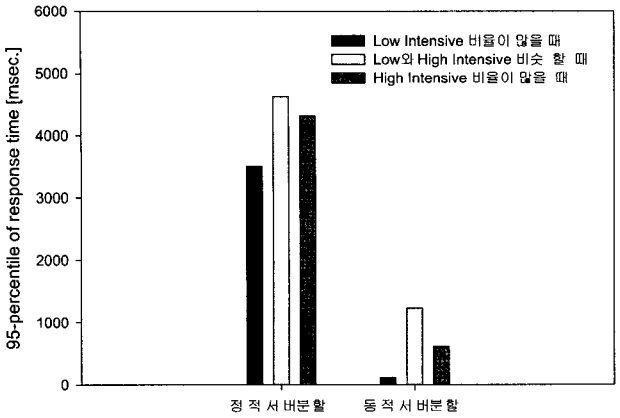
(그림 7) 사용자 도착률에 따른 정적·동적 서버 성능 분할 기법의 95-percentile 응답시간비교

(그림 7)은 이질 웹 서버 환경에서 사용자 도착률에 따른 정적·동적 서버 분할 기법간의 95-percentile 응답 시간 차이를 나타낸다. 95-percentile은 SLA를 고려하기 위해 사용된 척도로써, 들어온 요청중 95% 이상은 서비스 제공자와 사용자 간의 계약된 시간 안에 응답시간을 만족하는 것을 의미한다. 정적 서버 분할보다는 동적 서버 분할 기법의 응답시간이 대체적으로 작게 나왔으며, 이는 동적 서버 분할 기법이 사용자 요청을 고려하여 부하 변화에 맞게 대처하였기 때문이다. 하지만 동적 서버 분할 기법의 하위 계층은 과부하 상태 시 상위 계층에게 서버를 먼저 주어야 하기 때문에, 정적 서버 분할 기법에 비해 응답시간이 떨어지는 것을 알 수 있다.



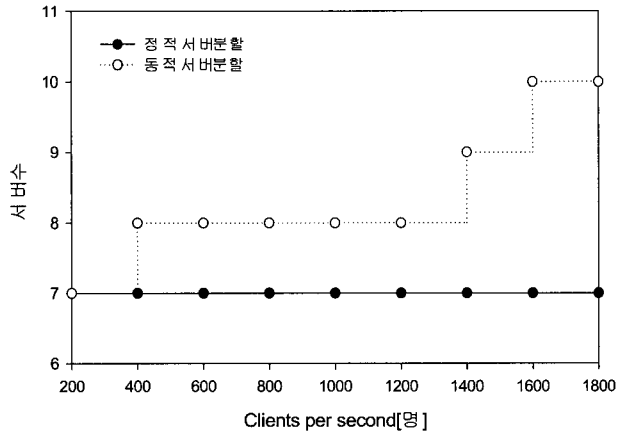
(그림 8) 사용자 도착률에 따른 거절된 하위 계층의 요청 수 비교

(그림 8)은 사용자 도착률에 따른 정적·동적 서버 성능 분할 기법간의 거절된 하위 계층의 요청 수를 비교한 것으로 동적 서버 성능 분할 기법에서 거절된 요청 수가 많을 것을 알 수 있다. 이는 동적 서버 성능 분할 기법이 승인제어를 고려하여 주로 하위 계층의 요청을 거절하였기 때문이다.



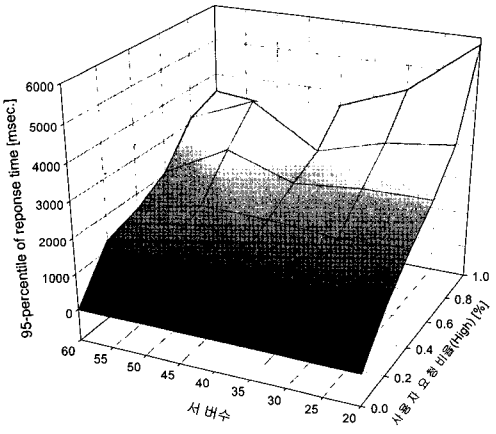
(그림 9) 사용자 요청 빈도수에 따른 정적·동적 서버 성능 분할 기법의 95-percentile 응답시간비교

(그림 9)는 사용자 요청 빈도수를 Low Intensive가 많을 때, Low Intensive와 High Intensive의 요청이 비슷할 때, High Intensive의 요청이 많을 때로 구분하여 정적·동적 서버 분할 기법의 95-percentile 응답시간을 나타낸다. Low Intensive와 High Intensive의 요청이 많을 때 보다는 이들의 요청 비율이 비슷할 때의 응답시간 성능이 저하되는 것을 볼 수 있으며, 정적 서버 분할 기법보다 동적 서버 분할 기법일 때의 응답시간이 빠른 것을 알 수 있다. 이는 동적 서버 분할 기법이 Intensive 비율의 증가로 과부하 상태인 계층에 하위 계층의 서버를 제공하기 때문이며, 또한 Low Intensive의 요청은 정적 문서와 같은 파일 형식으로 서비스 시간이 짧으며, High Intensive는 서비스 시간이 긴 VOD 동영상 파일이라 대체적으로 Low Intensive의 요청이 High Intensive 요청에 비해 응답시간이 좋게 나왔다.

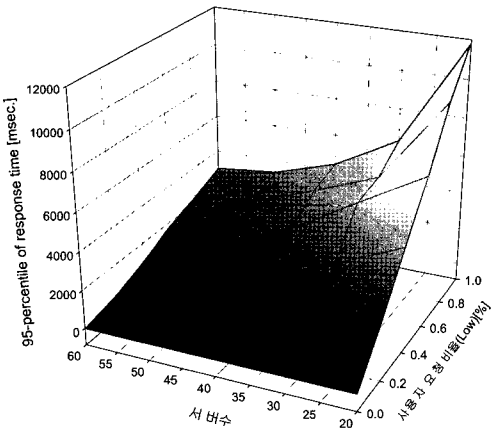


(그림 10) 사용자 도착률에 따른 상위 계층의 서버 수

(그림 10)은 사용자의 도착률이 증가함에 따라 정적·동적 서버 분할 기법을 통해 상위 계층에 할당되는 서버 수의 변화를 나타낸다. 정적 서버분할 기법은 사용자 계층별 서버 수를 고정적으로 할당하여 사용자의 도착률이 증가하여도 변함이 없으나, 동적 서버분할 기법은 사용자의 요청이 많아지면 상위 계층의 부하가 발생하여 상위 계층의 서버를 상위 계층으로 할당하고 부하를 개선하기 때문에 상위 계층의 서버 수가 증가하였다.



(a) 동적 서버 성능 분할(High)



(b) 동적 서버 성능 분할(Low)

(그림 11) 사용자 요청 비율과 서버 수의 변화에 따른 95-percentile 응답시간비교

(그림 11)은 사용자 요청 비율과 총 서버 수의 변화에 따른 상위 계층과 하위 계층의 응답시간을 표시하였다. 사용자 계층에 관계없이 서버 수가 증가할수록 응답시간이 감소하는 것을 볼 수 있으며, 또한 각 계층의 요청 비율이 증가할수록 처리시간이 길어져 응답시간 성능이 떨어지는 것을 알 수 있다. 이는 서버 수의 변화보다 사용자 요청 비율 변화에 따라 시스템의 성능 저하가 발생하므로, 사용자 요청 비율이 성능 평가 요소임을 알 수 있다.

6. 결 론

웹 스위치를 두 대로 운영하는 주-백업 웹 스위치 시스템 구조를 제시한 후, 가용도를 분석한 결과 한 개의 웹 스위치를 운영하는 것보다 주-백업 웹 스위치 구조를 채택할 경우 가용도를 매우 높이고 일정하게 유지할 수 있음을 확인하였다. 또한 웹 서버 노드를 기능적 도메인별로 구분한 이질 웹 서버 환경에서 사용자 계층별 요청률에 따라 서버 노드들을 정적·동적으로 관리하는 성능 분리 및 승인제어 기법을 제안하였으며, 실험을 통해 웹 서비스의 응답시간 성능을 평가하였다. 이로부터, 사용자와 서비스 제공자간에 SLA를 고려한 서비스가 이루어질 수

있도록 하였으며, SLA를 만족시키기 위해 고수준의 서비스를 요구하는 사용자에게 더 많은 컴퓨팅 자원을 할당하는 방식의 적용 가능성을 확인하였다. 추후에는 이질 웹 서버 환경에서 다양한 사용자들의 SLA를 보장하기 위하여 사용자 계층을 다중계층(Multiclass)로 확장한 서버 분할 기법에 대한 연구를 수행할 계획이다.

참 고 문 헌

[1] L. Aversa, et al., "Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting," Proc. of the IEEE Int. Performance, Computing, and Communication Conference (IPCCC'2000), Phoenix, AZ, pp.24-29, Feb., 2000.

[2] Microsoft Corporation, "Server Cluster Architecture," http://www.microsoft.com/windowsserver2003/techinfo/overview/servercluster.mspx., Mar., 2003.

[3] M. Andreolini, et al., "A Cluster-Based Web System Providing Differentiated and Guaranteed Services," Cluster Computing, Vol.7, No.1, pp.7-19, Jan., 2004.

[4] H. Chen, et al., "Overload Control in QoS-aware Web Servers," Computer Networks, Vol.42, No.1, pp.119-133, Dec., 2003.

[5] V. Cardellini, et al., "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.

[6] A. Sahai, et al., "Automated SLA Monitoring for Web Services," DSOM, pp.28-41, 2002.

[7] J. Pu, et al., "SLA Admission Controller for Reliable MPLS Networks," Applied Informatics, pp.630-635, 2003.

[8] X. Chen, et al., "ACES: An Efficient Admission Control Scheme for QoS-aware Web Servers," Computer Communications, Vol.26, No.14, pp.1581-1593, Sep., 2003.

[9] J. Martínez, et al., "QoS Estimators for Client-Side Dynamic Server Selection: Limitations and Keys," The Ninth IEEE Symposium on Computers and Communications. Alexandria, Egypt, pp.933-938, July, 2004.

[10] M. Aron, et al., "Scalable Content-aware Request Distribution in Cluster-based Network Servers," in Proceedings of USENIX'2000 Technical Conference, pp.323-336, June, 2000.

[11] E. Casalicchio, et al., "Content-aware dispatching algorithms for cluster-based Web servers," Cluster Computing, Kluwer Academic Publ., Vol.5, No.1, pp.67-76, January, 2002.

[12] V. Pai, et al., "Locality-aware Request Distribution in Cluster-based Network Servers," In Proceedings of 8th ACM Conference on Architecture Support for Programming Languages, pp.205-216, Oct., 1998.

[13] 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.

[14] H. Zhu, et al., "Demand-Driven Service Differentiation in Cluster-based Network Servers," Proceedings of IEEE Infocom, pp.679-688, Apr., 2001.

[15] 고현주 외 2인, "SLA를 고려한 웹 서버 부하 분산 기법", 한국정보과학회 가을학술발표논문집, Vol.31, No.2, pp.652-654, 2004.

[16] V. Cardellini, et al., "The State of the Art in Locally Distributed Web-Server Systems," ACM Computing Surveys(CSUR), Vol.34, No.2, pp.263-311, June, 2002.

[17] Y. Huang, et al., "Software Rejuvenation: Analysis, Module and Applications," IEEE Intl. Symposium on Fault Tolerant Computing, FTCS 25, pp.381-390, June, 1995.

강 창 훈

e-mail : chkang@kdc.ac.kr

1986년 충남대학교 계산통계학과(이학사)

1988년 충남대학교 계산통계학과 (이학석사)

1996년~현재 아주대학교 컴퓨터공학과 박사과정

1994년~현재 극동정보대학 방송영상미디어과 부교수

관심분야 : 클러스터컴퓨팅, 그리드컴퓨팅, 결함허용, 성능분석

박 기 진

e-mail : kiejin@ajou.ac.kr

1989년 한양대학교 산업공학과(공학사)

1991년 POSTECH 산업공학과(공학석사)

1991년~1997년 삼성전자 선임연구원

1997년~2001년 아주대학교 컴퓨터공학과 (공학박사)

2001년~2002년 한국전자통신연구원 선임연구원

2002년~2004년 안양대학교 컴퓨터학과 전임강사

2004년~현재 아주대학교 산업정보시스템공학부 조교수

관심분야 : Dependable Embedded Computing, Intrusion Tolerance Systems, Cluster Computing

김 성 수

e-mail : sskim@ajou.ac.kr

1982년 서강대학교 전자공학과(공학사)

1984년 서강대학교 전자공학과(공학석사)

1995년 Texas A&M University 전산학과 (공학박사)

1983년~1996년 삼성전자 수석연구원

2002년~2003년 Texas A&M University 교환교수

1996년~현재 아주대학교 정보통신전문대학원 정교수

관심분야 : Dependable System & Network, Autonomic Computing, Ubiquitous Computing & Network