Xen 환경에서 스케줄링 지연을 고려한 가상머신 우선순위 할당 기법

(A Priority Allocation Scheme Considering Virtual Machine Scheduling Delays in Xen Environments)

양 은 지 † 최 현 식 †† 한 세 영 ††† 박 성 용 ††††
(Eunji Yang) (Hyunsik Choi) (Saeyoung Han) (Sungyong Park)

요 약 CPU 자원이 다수의 가상머신에 의해 공유되는 Xen 가상화 환경에서는, CPU가 하나의 가상머신의 요청을 처리하는 동안 다른 가상머신은 CPU를 기다려야 하는 가상머신 스케줄링 지연이 존재한다. 가상화 환경에서 응용프로그램의 QoS 요구사항을 만족시키기 위하여 자원을 관리하는 대부분의 시스템은 가상머신의 자원 사용률과 가상머신에서 운영하는 응용프로그램의 성능을 모니터링하고 분석하여 자원을 재할당한다. 이 때 응용프로그램의 성능 분석을 위해 큐잉 모델 등과 같은 수학적인 모델링 기법이 사용되지만 비가상화 환경에서 사용되던 모델은 가상머신 스케줄링 지연을 고려하지 않으므로, 가상화 환경에서는 정확한 분석과 예측이 어렵고, 따라서 이를 기반으로 자원을 관리하는 시스템은 요구되는 응용프로그램의 성능을 제공하지 못할 수 있다. 따라서 본 논문에서는 Xen 가상화 환경에서 가상머신 스케줄링 지연을 반영하여 응용프로그램의 성능을 측정하고, 모든 가상머신이 일으킬 수 있는 스케줄링 지연을 최소화하는 방향으로 CPU 사용 우선순위를 설정하는 기법을 제안하고, 제안한 기법이 스케줄링을 고려하지 않은 방법에 비해 응용 프로그램의 성능을 향상시킴을 보인다.

키워드 : 가상화, 자원 관리, 응용프로그램 서비스 품질

Abstract There exist virtual machine scheduling delays in virtualized environment in which virtual machines share physical resources. Many resource management systems have been proposed to provide better application QoS through monitoring and analyzing application performance and resource utilization of virtual machines. However, those management systems don't consider virtual machine scheduling delays, result in incorrect application performance evaluation and QoS violations In this paper, we propose an application behavior analysis considering the scheduling delays, and a virtual machine priority allocation scheme based on the analysis to improve the application response time by minimizing the overall virtual machine scheduling delays.

Key words : Virtualization, Resource Management, Application QoS

· 본 연구는 지식경제부 및 한국산업기술평가관리원의 IT산업원천기술개발사업의 일환으로 수행하였음[2010-KI002090, 신뢰성 컴퓨팅(Trustworthy Computing) 기반 기술 개발]

† 정 회 원 : 삼성전자 생산기술연구소 연구원
ejyang85@gmail.com
†† 정 회 원 : 한국산업은행 Core 뱅킹 전산실
hschoi@sogang.ac.kr
††† 학생회원 : 서강대학교 컴퓨터공학과
syhan@sogang.ac.kr
†††† 종신회원 : 서강대학교 컴퓨터공학과 교수
parksy@sogang.ac.kr

논문접수 : 2009년 4월 17일
심사완료 : 2010년 4월 22일

정보과학회논문지 : 시스템 및 이론 제37권 제4호(2010.8)

1. 서 론

서버 가상화는 다수의 가상서버가 하드웨어를 공유하여, 한 대의 물리서버에서 동시에 실행될 수 있도록 해주는 소프트웨어 기술로, 공간 및 컴퓨팅 자원 사용의 효율을 높여준다. 또한 가상머신 단위의 자원 할당, 이주 기능 등 시스템 관리에도 이점을 제공한다. 이러한 장점들을 바탕으로, 최근 데이터센터에서는 가상화 기술을 이용해 다수의 서버들을 통합시켜 사용하는 경우가 증가하고 있다.

그러나 가상화된 환경에서는 하드웨어 자원이 다수의 가상서버들에게 공유되어야 하기 때문에, 가상서버에서 수행되는 응용프로그램의 QoS 요구사항을 충족시킬 수

있도록 자원을 할당하는 자원관리 문제가 단일 서버 환경보다 더욱 복잡해지므로, 관리자의 개입 없이 자동으로 자원을 할당하고 조정하는 자동화된 자원관리 시스템을 필요로 하게 된다. 따라서 가상화 환경에서는 일반적으로 자원 관리 시스템을 통해 서버의 자원 사용률과 가상머신에서 운영하는 응용프로그램의 성능을 모니터링 및 분석하고, 그 결과에 따라 동적으로 자원을 할당하여, 가상서버에서 운영되는 응용프로그램의 QoS 만족도를 높이는 여러 연구들이 진행되어 왔다.

그러나 가상화 환경에서는 하나의 자원을 여러 가상머신이 공유하므로, 하나의 가상머신이 그 자원을 사용하는 동안 다른 가상머신은 그 자원을 사용하지 못하고 기다려야 하는 가상머신 스케줄링 지연이 존재하게 되는데, 기존 연구들에서는 이를 반영하지 못하고 있다. 그림 1은 균일한 요청 도착 간격 분포를 갖는 동일한 워크로드를 Xen 가상화 환경과 단일 서버 환경의 웹서버에 발생시켰을 때 실제 요청이 도착하는 사이 간격을 나타내고 있다. 단일 서버 환경의 경우 실제 발생한 워크로드와 동일하게 일정한 간격으로 요청이 도착하지만 가상화 환경의 경우는 불규칙한 요청 간격을 보인다. 이는 단일 서버 환경에서 요청은 네트워크 장치에 도착된 후 인터럽트에 의해 직접 처리되지만, 가상화 환경에서는 I/O 도메인인 Domain-0에 의해 요청이 전달된 후, 목적지 가상머신에 CPU가 스케줄링 되어야 CPU가 요청을 처리하므로, Domain-0와 목적지 가상머신이 순차적으로 CPU를 스케줄링 받도록 기다리는 스케줄링 지연이 일어나게 되어 응용프로그램의 요청 도착 간격이 일정하지 않게 된다.

따라서 본 논문에서는 Xen 가상화 시스템에서 가상머신들의 CPU 스케줄링 지연이 응용프로그램의 성능에 미치는 영향을 고려하여 응용프로그램의 성능을 추정하고, 추정한 결과를 바탕으로 각 가상머신의 우선순위 값을 조정하여 CPU 자원을 할당하는 관리 기법을 제안하고, 이를 Xen 가상화 환경에서 구현한다 [1]. Xen 가상화 환경에서 자율적인 자원 관리를 제안하는 연구 중 하나인 샌드파이퍼에서는 가상머신의 자원 사용 정보와 각 가상머신에서 동작하는 응용프로그램의 로그 정보를 활용하여 응용프로그램의 성능을 추정하고 G/G/1 큐잉 모델을 이용하여 Xen Credit 스케줄러의 파라미터인 Weight 값을 조정한다 [2]. 그러나 샌드파이퍼에서도 가상머신의 CPU 스케줄링 지연에 따르는 성능 저하를 고려하지 않고 있으므로, 본 논문에서는 CPU 스케줄링 지연을 고려하는 방법이 샌드파이퍼와 비교하여 응용프로그램의 성능을 향상시킴을 보이고자 한다.

논문의 구성은 다음과 같다. 1장에서는 연구의 배경을 제시하고, 2장에서는 본 연구와 관련이 있는 기존 연구들을 살펴본 후, 기존 연구의 문제점과 본 연구의 필요성을 제기한다. 3장에서는 Xen 가상화 환경에서 가상머신의 스케줄링 지연을 고려한 어플리케이션 성능 측정 방법 및, CPU 자원 할당 기법을 제안한다. 4장에서는 실험을 통해 제안한 기법의 성능을 측정하고, 마지막 5장에서는 본 논문의 결론과 향후 과제에 대해 논의한다.

2. 관련 연구

서비스를 제공하는 서버의 입장에서 응용프로그램이 요구하는 QoS를 지원하는 것은 매우 중요하다. 특히 다수의 자원을 공유하는 리소스 풀 환경이나, 한 대의 물리 서버가 다수의 가상 서버를 지원해야 하는 가상화

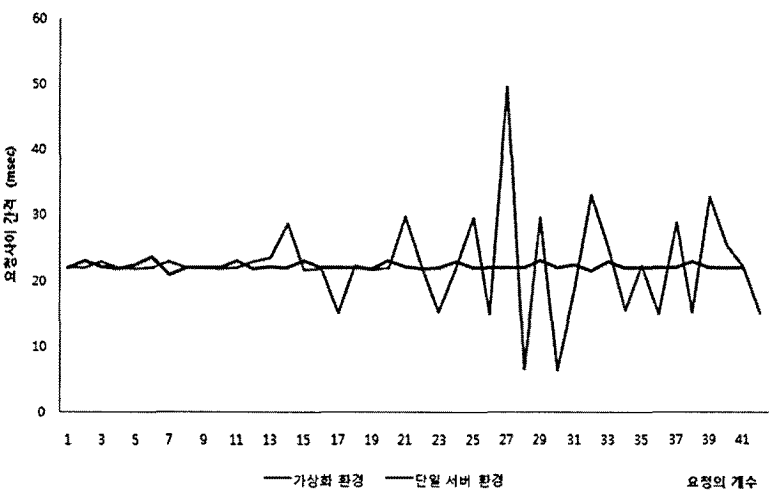

그림 1 요청 도착 시간 간격: 가상화 환경과 단일서버 환경 비교

환경에서는 그것이 더욱 중요하고 복잡하다[3]. 따라서 이에 관련된 다양한 연구가 진행되었다.

Océano 시스템은 서버 풀 환경에서의 자원관리 시스템으로, 지속적으로 서버의 부하와 성능 지표를 모니터링하여, 그 값이 지정 임계값을 넘으면 이벤트를 발생시켜 자원 할당량을 늘려주게 하여, 사용자의 서비스 레벨 요구사항(SLA)를 만족시킬 수 있도록 한다[4]. 이 연구에서는 응용프로그램에 따르는 SLA와 모니터할 성능 지표들, 그리고 SLA를 위반하지 않도록 미리 자원할당을 요구하는 이벤트를 발생시킬 수 있도록 하는 임계값 등을 정의하고, 이를 수행 할 시스템을 제안하고 있다. 그러나 Océano는 가상화 환경이 아닌 서버 풀 환경에서 제안된 시스템으로 자원을 서버 단위로 할당하고 네트워크 대역폭을 제한함으로써 자원을 관리한다. 따라서 가상화 환경에서 바로 적용하기는 어렵고, 또한 공유 자원에 대한 스케줄링 지연에 대한 고려도 하지 않고 있다.

공유된 자원 풀 환경에서 응용 프로그램의 QoS를 보장하도록 동적으로 자원을 할당 하기위해 응용 프로그램 별로 스케줄링 파라미터 값을 계산하는 연구가 진행되었다[5]. 이 연구에서는 응용 프로그램의 자원 요구량의 특성과 응용 프로그램의 QoS 요구사항, 그리고 공유된 자원을 접근하는 특성 등을 모니터 하고, 이를 기반으로 응용프로그램의 우선순위를 결정하여 관련 자원의 스케줄러 파라미터를 설정함으로써 응용 프로그램의 자원 요구사항을 만족시킨다. 그러나 이 연구에서는 trace 기반의 접근방법을 사용하여 장기간의 즉 여러 주나 여러 달동안의 응용 프로그램의 자원 요구 특성을 파악하고 자원의 할당을 결정하게 되므로, 가상화 환경에서의 실시간 동적 자원 관리에는 적합하지 않고, CPU의 스케줄링 지연을 고려하지고 않는다.

한편, 샌드파이퍼에서는 Xen 가상화 환경에서 각 가상 서버에서 서비스하는 응용 프로그램의 QoS 요구사항을 지켜줄 수 있도록 실시간으로 가상 서버의 자원 사용 현황과 응용 프로그램의 성능, 그리고 워크로드의 현황을 모니터하여 자원을 재할당 하거나 가상 서버를 자원이 충분히 남아있는 서버로 이전시키는 방법을 제안하고 구현하였다[1]. 즉, 웹 서버의 로그를 활용하여 서버의 서비스 응답시간을 모니터하고, 목표 서비스 응답시간을 만족할 수 있도록 해당 가상머신에 CPU를 할당하는데, 이 때 Xen의 CPU 스케줄러인 Credit 스케줄러의 우선순위 값인 Weight 값을 계산하여 적용함으로써 각 가상머신의 CPU할당에 변화를 주게 된다. 그러나 이 연구에서도 CPU 스케줄링 지연에 대한 고려가 없으므로, 전체적으로 시스템의 부하가 큰 상황에서는 응용 프로그램의 성능을 정확하게 예측하지 못하게 되고, 따라서 적절한 weight 값을 설정하는데 한계를 갖게 된다.

한편, 가상화 환경에서 각 가상머신에 동적으로 자원을 할당하기 위하여 두 계층의 자원 관리 컨트롤러를 제안하는 연구들이 진행되고 있다[6,7]. 이 연구들에서는 각 서버 단위의 지역 컨트롤러와 전체 자원 풀 단위의 전역 컨트롤러를 사용할 것을 제안하고 있는데, 이를 통해 데이터 센터의 각 응용프로그램의 다양한 목적에 맞게 자원을 관리할 수 있도록 한다. 즉, 지역 컨트롤러의 경우 각 서버 위에서 각 가상머신의 자원 사용량과 응용 프로그램 별 자원 요구량을 파악하여 전역 컨트롤러에게 요청하고, 전역 컨트롤러는 전체 서버 풀의 이익을 극대화 하는 방향으로 모든 서버에서의 요청들을 조정한다. 이 때 각 서버의 다양한 응용프로그램이나 서버 플랫폼에 따라 다양한 지역 컨트롤러를 구현할 수 있는데, 퍼지 로직을 기반으로 한 지역 컨트롤러[6]와 simulated annealing을 이용하여 자원 요구량을 산출하는 지역 컨트롤러[7] 등이 제안되었다. 그러나 이들 연구에서도 공유되는 자원, 특히 CPU스케줄링에 의한 지연을 따로 고려하지 않고 있으므로, 각 서버의 해당 플랫폼에서 CPU 스케줄링 특성에 따라 CPU 스케줄링 지연을 고려하여 자원을 할당하도록 지역 컨트롤러를 보완한다면 더 좋은 성능을 나타낼 수 있을 것으로 기대된다.

지금까지의 가상화 환경에서 자원관리 시스템에 대한 연구들과 달리, Xen 가상화 환경에서의 I/O 처리과정을 고려한 가상머신 스케줄러를 제안하는 연구도 진행되었다[8]. 가상화 기술을 이용한 서버들의 통합으로 물리 서버의 활용률은 높아졌으나, 클라이언트에서의 응답 시간은 감소하였음을 문제로 제기하고, 응답 시간 성능이 감소한 원인을 가상머신의 CPU 스케줄링 지연에서 찾고, 가상머신 간 스케줄링 지연 시간을 줄이는 스케줄러를 제안하고 있다. 즉, Xen 가상화 환경에서는 드라이버 도메인인 Domain-0이 실질적인 I/O 작업을 수행하므로, 요청을 보내는 가상머신이 수행된 후 실제 패킷을 전송하는 Domain-0이 스케줄링 되지 않으면 요청에 지연이 발생하고, 요청을 받는 경우에도 Domain-0에서 수신한 패킷의 목적지 가상머신이 Domain-0 다음에 바로 스케줄링되지 않을 경우 지연이 발생한다(그림 2). 따라서 이 연구에서는 Xen의 SEDF(Simple Early Deadline First) 스케줄러를 기반으로 가상머신 스케줄링이 야기하는 요청의 지연을 감소시키기 위해 송신 또는 수신할 패킷이 많은 가상머신을 마감시간이 허용하는 한도에서 먼저 스케줄링 하는, 수정된 SEDF CPU 스케줄러를 제안하고 있다. 그러나 최근에는 Symmetric Multi-Processing(SMP) 환경이 대부분이므로, CPU 간 부하분산을 지원하는, 즉, 하나의 CPU가 할당된 가상머신의 작업을 끝낸 후 idle 상태로 가는 대신 다른 CPU를 기다리는 가상머신이 있을 경우 대신 처리하여, 가상

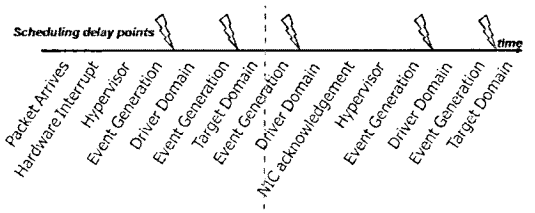

그림 2 가상머신 스케줄링 지연 발생 시점

머신이 실제로 자신에게 할당된 비율 이상의 CPU 시간을 사용할 수 있는 work-conserving 방식의 스케줄러인 Credit 스케줄러를 주로 사용하는 추세이다[9,10].

따라서 본 연구에서는 Credit 스케줄러를 기반으로 CPU 스케줄링 지연을 고려하여 가상머신의 CPU 사용 우선순위 값을 결정하여 설정함으로써 가상머신이 요구하는 CPU 자원을 할당하고 요구하는 QoS를 만족시키도록 하는 자원관리 시스템을 제안하고자 한다. 즉, 자원을 공유하는 환경에서 발생하는 CPU 스케줄링 지연을 고려하여 정확한 응용 프로그램의 성능을 측정하고 예측한 후, Credit 스케줄러의 파라미터인 Weight 값을 계산하여 좀 더 정확하고 효율적으로 가상머신의 자원을 관리하는 방안을 제안하고, 기존 연구 중 Xen 가상화 환경에서 CPU 스케줄링 지원을 고려하지 않고 같은 Credit 스케줄러의 가상머신의 CPU 사용 우선순위를 결정하는 샌드파이퍼와 비교하여, 응용 프로그램의 성능이 좋아짐을 보이고, 제안한 기법이 보다 효율적으로 자원을 관리함을 보이고자 한다.

3. 스케줄링 지연을 고려한 우선순위 할당

가상머신 스케줄링 지연은 가상머신에서 운영되는 서버 응용프로그램의 성능에 영향을 준다. 본 장에서는 Credit 스케줄러를 사용하는 Xen 가상화 시스템을 대상으로 가상머신 간 스케줄링 지연을 고려한 성능 추정 방법을 제안하며, 그 결과를 이용하여 동적으로 CPU 자원을 할당하여 자원을 관리하는 기법을 제시한다.

3.1 가상머신 스케줄링 지연을 고려한 성능 추정 방법

3.1.1 가상머신 스케줄링 지연

가상머신 스케줄링에 의해 Xen 가상화 서버에 도달한 요청들이 지연되는 경우는 Domain-0에 의한 지연과 다른 가상머신의 CPU 사용에 의한 지연, 두 가지 경우가 있다. Xen에서 가상머신의 I/O 작업은 드라이버 도메인인 Domain-0을 거쳐서 이루어지는데, 가상머신의 응용프로그램에 대한 요청이 실제 네트워크 디바이스에 도착한 시점이나 가상머신에서 보내는 응답이 Domain-0에 전달된 시점에 Domain-0이 스케줄링 되지 않을 경우 해당 요청에 지연이 발생한다. 또한 요청이 Domain-0을 거쳐 목적 가상머신에 도달해야하는 시점에서, 해당 가상머신이 스케줄링되지 않으면 지연이 발생한다. 이 때, 처리되지 않은 요청을 가진 가상머신이 CPU를 사용할 때까지 대기하는 지연 시간은 가상머신의 CPU 우선순위에 영향을 받는다.

3.1.2 성능 추정 방법

본 논문에서 제안하는 가상머신 스케줄링 지연을 고려한 큐잉 시스템은, 요청을 처리하는 서버에 해당하는 물리 CPU와 가상머신에 도착하는 서비스 요청들로 구성된다(그림 3). 이 때, 서비스 요청들은 목적지 가상머신에 따라 클래스로 분류되며, 같은 클래스에 속한 서비스 요청들, 즉 한 가상머신의 서비스 요청들은 FCFS (First Come First Served)로 처리된다.

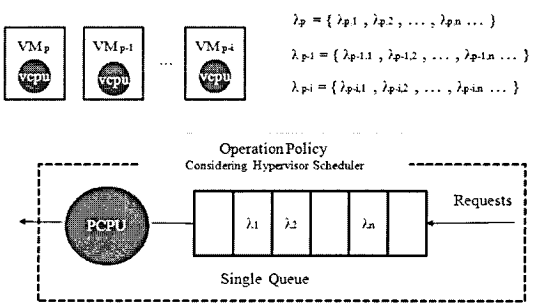

그림 3 제안한 큐잉시스템 구성

일정 시간 구간에서 한 가상머신(VM_p)에 대한 요청들의 평균 응답 시간 R_p는, 가상머신 VM_p에 대한 서비스 요청들이 실제로 CPU에서 처리되는 시간인 평균 서비스 시간 X_p, 서비스 요청이 서버에 도착했을 때 이미 다른 가상머신이 스케줄링 되어 처리되고 있거나 다른 가상머신에 의해 선점됨으로써 발생하는 지연시간 D_{VMp}, 그리고 드라이버 도메인인 Domain-0이 CPU를 기다리는 스케줄링 지연시간 D_{0p}의 합으로 식 (1)과 같이 나타낼 수 있다.

$$R_p = X_p + D_{VMp} + D_{0p} \quad (1)$$

이 때, R_p는 논리적인 평균 전체 응답시간을 의미하므로, 요청의 서비스 시간이 가상서버에 할당된 CPU 점유시간 보다 커서 응답시간이 커지는 경우도 표현할 수 있다. n개의 게스트 가상머신을 운영하는 가상화 서버에서 Xen Credit 스케줄러는 기본적으로 Round-Robin 방식으로 가상머신들을 스케줄링하므로, 다른 n-1개의 가상머신들이 발생시키는 스케줄링 지연(D_{VMp})은 식 (2)와 같이 Domain-0의 평균 요청 처리시간($\omega_0 X_0$)과 다른 가상머신의 처리로부터 발생하는 지연의 합으로 정의할 수 있다. Credit 스케줄러에서 각 가상머신의

스케줄링 횟수는 모든 가상머신의 상대적 우선순위 (w_i/w_p)에 대해 의존적이다. 따라서 다른 가상머신의 서비스 시간(X_i)도 상대적 요청률(λ_i/λ_p)을 고려하야 정의하였다. 즉, λ_p를 가상머신(VM_p)에 도달하는 초당 요청의 개수, 즉, 요청 도착률이라고 하면, VM_p의 요청하나당 다른 가상머신 VM_i의 평균 서비스 시간은 (w_iX_i/w_p)로 나타낼 수 있으므로, 이들의 합을 지연시간으로 정의한다.

$$D_{VMp} = \omega_0 X_0 + \sum_{i=1, i\neq p}^{n} \frac{\omega_i}{\omega_p} X_i, where\, X_i = \frac{\lambda_i}{\lambda_p} \overline{x_i} \quad (2)$$

한편, Domain-0은 Credit 스케줄러의 BOOST 기법을 통해 우선적으로 스케줄링 될 수 있으므로, 식 (2)와 같이 긴 스케줄링 지연을 겪지 않는다. 그에 따라 Domain-0의 지연시간(D_{0p})은 물리서버가 처리하는 I/O 작업량에 비례하게 된다. 또한 드라이버 도메인은 결국 해당 가상머신의 I/O 작업을 처리하는 역할을 하므로, 드라이버 도메인의 스케줄링 빈도는 가상머신(VM_p)의 스케줄링 빈도 및 처리하는 요청의 양과 직접적인 관련이 있다. 따라서 $\omega_p\lambda_p$로 나누어, 식 (3)과 같이 D_{0p}를 정의한다.

$$D_{0p} = \frac{IO}{\omega_p \lambda_p} D_0 \quad (3)$$

이 때, Xen Credit 스케줄러는 각각의 가상머신이 보유한 Credit 값에 따라 비율적으로 CPU 자원을 사용하도록 하므로, 가상머신(VM_p)의 우선순위 값(w_p)은 식 (4)처럼, 가상머신 VM_p의 Credit 값을 전체 Credit 값에 대해 정규화한 것으로 나타낼 수 있다. 따라서 가상머신(VM_p)의 우선순위 값(w_p)은 0보다 크거나 같고, 1보다는 작거나 같으며, 물리서버의 다수의 가상머신들 중에서 가상머신(VM_p)이 스케줄링 되는 상대적인 빈도를 나타내낸다. 같은 방법으로 드라이버 도메인의 우선순위 값인 w_0도 표현할 수 있다.

$$\omega_p = \frac{credit_p}{\sum_{i=1}^{n} credit_i}, where\, 0 \le \omega_p \le 1 \quad (4)$$

3.2 가상머신 우선순위 할당 기법

본 장에서는 가상머신에서 운영하는 웹서버의 요구성능을 만족하도록 CPU 자원을 할당하는 방법을 제안한다. 일반적으로 웹서버의 성능 측정 지표로는 초당 처리한 요청의 개수와 요청의 서버 응답 시간(ms)이 사용된다. 제안하는 CPU 자원 할당 기법에서도 단위시간당 처리되는 요청의 개수(N_p)와 서버 응답 시간(R_p)을 성능의 지표로 사용한다. 가상머신(VM_p)의 웹서버에서 요구하는 요청의 처리율(N_p)과 서버 응답 시간(R_p)을 모두 충족하도록 CPU 자원을 할당하는 것은, N_p을 서비스 할 수 있는 CPU 사용 시간의 보장과 동시에 R_p을 초과하지 않는 CPU 응답 시간을 보장하는 CPU 우선순위 값(weight)을 가상머신(VM_p)에 할당해야 함을 의미한다. 만약 우선순위 값 할당으로 요구되는 성능을 만족시킬 수 없다면, 이주 기능을 사용한 자원 할당을 수행해야 한다.

3.2.1 할당 시점의 추정

사용자가 명시한 요청 처리율($N_{p,SLA}$)과, 서버 응답 시간($R_{p,SLA}$)이 지켜지지 않을 경우를 자원 할당이 필요한 시점으로 추정한다. 일반적으로 $N_{p,SLA}$을 충족하지 못하는 상황에서는 $R_{p,SLA}$ 또한 충족하지 못한다. 하지만, 가상머신(VM_p)에 $N_{p,SLA}$개의 요청이 처리될 만큼의 CPU 시간이 할당 되더라도, 해당 가상머신에 대한 워크로드 성질에 관계없이 다른 가상머신에 의한 스케줄링 지연에 의해 $R_{p,SLA}$을 어기게 되는 상황은 발생 가능하므로, 그 역은 성립하지 않는다. 따라서 식 (5)와 같이 $N_{p,SLA}$을 충족하지 못하는 경우와 $N_{p,SLA}$은 충족하더라도 $R_{p,SLA}$을 어기는 상황이 발생할 경우를 CPU 할당의 시점으로 판단한다.

$$N_{p,SLA} \ge N_{p,Peak} \text{ or } R_{p,SLA} \le R_{p,Peak}$$
$$where \sum_{i=1}^{n} R_{i,SLA} \ge \sum_{i=1}^{n} R_{i,Peak} \quad (5)$$

$$R_{SLA}(=\sum_{i=1}^{n} R_{i,SLA}) \le R_{Peak}(=\sum_{i=1}^{n} R_{i,Peak}) \quad (6)$$

Xen의 Credit 스케줄러와 같은 작업 보존 방식 스케줄러를 사용하는 경우는, $N_{p,SLA}$을 처리하는 CPU 시간이 부족할 때, 다른 가상머신에서 남는 CPU 시간을 사용할 수 있다. 따라서 다른 가상머신의 CPU 시간까지 이용하더라도 $N_{p,SLA}$을 충족하지 못하는 상황이라면 가상머신의 자원 사용 추이를 관찰하여 과부하 시점을 감지할 수 있다.

과부하가 발생한 시스템에서의 $N_{p,Peak}$, $R_{p,Peak}$은 어플리케이션 로그로부터 프로파일링 한 데이터의 95번째 백분위수로 추정한다. 이 후, 식 (5)의 두 번째 조건과 같이 추정한 $R_{p,peak}$이 $R_{p,SLA}$를 어기는 가상머신이 발생하는 경우를 자원 할당이 필요한 시점으로 판단한다. 할당의 시점이 판단되면, 현재 물리 서버 CPU의 상황이 우선순위 값 할당으로 가상머신들의 요구사항을 만족시켜줄 수 있는가를 판단해야한다. 만약 식 (6)처럼, 시스템 내 가상머신들의 추정 응답 시간의 총 합 R_{Peak}이 가상머신에서 요구하는 응답 시간의 총 합 R_{SLA}보다 큰 경우, 현재 물리 서버에서는 CPU 우선순위 값 할당

으로 QoS를 보장해 줄 수 없다고 판단하고, Xen의 이주 기능을 이용해 자원을 관리한다.

3.2.2 CPU 우선순위 값 할당

Credit 스케줄러는 다수의 가상머신이 공평하게 CPU 자원을 사용하도록 설계된 것으로, 가상머신(VM_p)의 스케줄링 우선순위 값, Credit 값을 높이는 것이 가상머신 상의 어플리케이션 응답 시간을 직접적으로 향상시킴을 의미하지는 않는다. 하지만, 우선순위를 높이면 점진적으로 요청의 응답 시간을 향상시킬 수 있다. 따라서 가상머신(VM_p)의 어플리케이션 응답 시간($R_{p,peak}$)이 요구되는 시간($R_{p,SLA}$)보다 늦은 경우, 가상머신의 스케줄링 우선순위 재할당을 통해, 성능을 향상시킬 수 있도록 한다. 만약 물리 서버가 모든 가상머신의 부하를 처리하기에 충분한 자원을 가지고 있다면, 위의 재할당 과정은 상대적으로 쉽게 이루어질 수 있다. 반면에, 물리 서버의 통합률이 높아 부하도 높은 상황이라면, 재할당량 선정은 신중히 고려되어야 한다. 이는 임의의 가상머신(VM_p)에 대한 우선순위를 높이는 것은 다른 가상머신의 우선순위를 상대적으로 낮추어 성능에 영향을 미치기 때문이다. 따라서 재할당은 한 대의 가상머신의 스케줄링 지연을 최소화하는 방법이 아닌 물리서버 내의 모든 가상머신의 스케줄링 지연을 최소화하는 방향으로 이루어져야 하며, 변형된 배낭 문제를 통해 해를 구할 수 있다. 한편, Xen의 특성상, I/O 작업은 Domain-0에 의해 먼저 처리된 후, 가상머신에서 수행되므로, Domain-0의 우선순위 값은 변경하지 않도록 한다.

$$D_{Total} = const_1\omega_1 + const_2\omega_2 + \cdots + const_n\omega_n \quad (7)$$

할당이 이루어지는 시점에서, 식 (2)를 구성하는 우선순위 값(w_i)을 제외한 요청 평균 처리 시간($\overline{x_i}$), 초당 요청수(λ_i) 등은 관찰된 결과를 통해 상수($const_i$)로 주어지며, 결국 전체 지연 시간(D_{Total})은 식 (7)과 같이 우선순위 값을 변수로 하는 다항식으로 표현될 수 있다. 따라서 주어진 다항식의 결과를 최소화 하는 우선순위 값들의 집합을 찾아서 그를 Credit에 반영하여, 가상머신 스케줄링 지연을 최소화한다. 이 때, 위의 문제는 변형된 배낭 문제를 이용해 풀 수 있으며, 우선순위 값의 변화폭이 작으면 큰 오버헤드를 초래할 수 있으므로, 우선순위 값은 0.05 단위로 변경된다.

4. 성능 평가

본 장에서는 제안한 CPU 자원 할당 기법의 성능을 샌드파이퍼의 성능과 비교하여 평가한다. 우선 제안한 기법의 기반이 되는 스케줄링 지연을 고려한 성능 추정 방법의 정확성을 평가한다. 그리고 제안한 기법이 실제 가상머신에서 요구하는 QoS를 만족시킴을 확인한다.

4.1 실험 환경 및 가정

실험환경은 Xen 3.1.0을 사용한 가상화 서버와 서버에서 동작하는 각 가상머신에 대한 클라이언트들로 구성하였다. 가상머신에서는 아파치 2.0 웹서버 어플리케이션을 운영하며, 클라이언트에서는 httperf 벤치마크[11]를 사용하여 요청을 보낸다. 클라이언트에서는 주로 CPU 작업을 하는 php 파일을 요청하도록 하였다. 그리고 각 가상머신에 하나의 VCPU를 할당하고, VCPU들이 가상화 서버의 단일 물리 CPU를 공유하도록 하였다. 할당 기법에 사용되는 CPU 자원 사용률 데이터는 xenmon[12,13]을 사용하여 수집되며, 성능 정보 데이터 - 처리된 요청의 개수, 요청의 시스템 시간 는 아파치 로그로부터 수집된다.

4.2 가상머신 스케줄링 지연을 고려한 성능 추정 방법의 정확성

제안한 어플리케이션 성능 추정 기법의 정확성을 보이기 위해, 실제 요청을 보내는 클라이언트가 측정한 응답 시간과 제안한 기법이 추정하는 응답 시간 그리고 샌드파이퍼가 추정하는 응답 시간을 비교한다.

표 1은 실험에서 각 가상머신에 발생시킨 부하를 보여준다. 구간 1에서 240초 동안 가상머신 1, 2, 3에 대해 초당 15개, 15개, 70개의 요청을 발생시킨 후, 구간 2에서 60초 동안 초당 45개, 45개, 170개의 요청을 보낸다. 구간 3에서는 240초 동안 초당 10개, 10개, 50개의 요청을 전송하고, 마지막 구간 4에서는 120초 동안 초당 45개, 45개, 170개의 요청을 보낸다. 표의 마지막 열에 나타낸 서비스 시간/요청 값은 시나리오 전체 구간의 시간을 전체 요청의 개수로 나눈 값으로 요청의 강도를 보여준다.

다음 그림 4, 그림 5, 그림 6은 가상머신1, 가상머신2, 가상머신3에서 실제 측정된 클라이언트의 응답 시간을 제안한 방법에서 추정하는 응답 시간과 샌드파이퍼에서 추정하는 응답시간과 비교한 결과를 나타낸다. 전체적으로 샌드파이퍼는 가상머신들에 대한 부하가 높은 두 번째, 네 번째 구간에서 실제 응답 시간 보다 높은 응답 시간을 추정함을 볼 수 있는데, 이는 샌드파이퍼에서는 가상머신 스케줄링 지연을 고려하지 않고 어플리케이션의 성능을 추정하기 때문에 부하가 심한 구간에서는 측

표 1 워크로드 시나리오 1

	각 가상머신의 부하(초당 평균 요청수)				서비스시간/요청(ms)
구간 (s)	1	2	3	4	
	240	60	240	120	
가상머신1	15	45	10	45	6.6
가상머신2	15	45	10	45	6.6
가상머신3	70	170	50	170	1.7

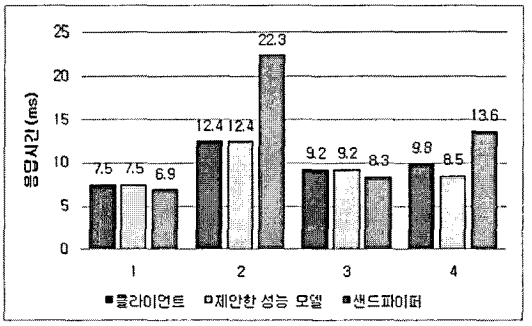

그림 4 웹서버 평균 응답 시간: 가상머신1

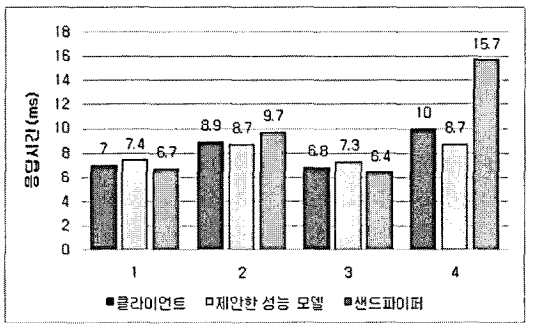

그림 5 웹서버 평균 응답 시간: 가상머신2

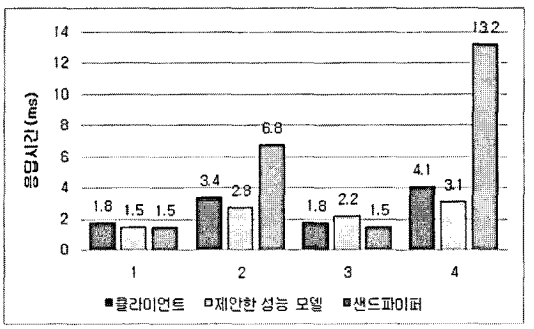

그림 6 웹서버 평균 응답 시간: 가상머신3

정한 서비스 시간이 실제보다 크게 반영되므로 이러한 부정확한 추정을 하게 된다. 반면에 제안한 시스템은 스케줄링 지연을 고려하여, 실제 클라이언트의 응답 시간과 유사한 값을 추정함을 확인할 수 있다.

한편, 본 실험에서는 가상머신 스케줄링 지연을 더욱 잘 이해할 수 있도록, 가상머신1과 가상머신2에는 동일한 양의 워크로드를 생성시키고, 가상머신 3에는 다른 양의 워크로드를 생성시켰다. 그림 4와 그림 5에서는 가상머신1과 가상머신2에 같은 양의 워크로드가 도달했음에도 불구하고 샌드파이퍼가 추정한 응답시간이 많이 차이가 나는데, 이는 가상머신 스케줄링 지연이 다르기 때문이다. 하지만 제안한 시스템은 두 경우 모두 적절한 응답시간을 추정하고 있음을 확인할 수 있다. 그림 6에서는 요청 부하가 더 큰 가상머신3에 대한 응답시간 추정 결과를 보여주고 있는데, 제안하는 시스템이 요청의 강도에 큰 영향을 받지 않고, 올바른 결과를 도출함을 확인할 수 있다.

4.3 CPU 자원 할당 기법의 QoS 만족도

본 장에서는 가상머신들의 CPU 사용률을 급격히 증가시키는 워크로드 시나리오에서, 본 논문에서의 CPU 자원 할당 기법과 샌드파이퍼의 방법을 비교하고자 하는데, 할당 기법의 수행 결과가 실제 가상머신 QoS에 미치는 영향을 평가의 기준으로 한다.

그림 7은 샌드파이퍼 환경의 가상머신 1, 2, 3에 표 2의 워크로드 시나리오를 적용한 후, 워크로드를 보낸 클라이언트에서 응답시간을 측정한 결과이다. 워크로드 시나리오에 의하면 세 번째 구간에서 60초 동안 과부하가 발생한다. 정확성 평가에서 확인했듯이, 샌드파이퍼의 어플리케이션 성능 추정 결과는 가상머신 스케줄링 지연을 반영하지 않기 때문에, 부하의 양이 커질수록 실제 응답 시간 보다 큰 응답 시간을 추정한다. 그에 따라 과부하 상황인 세 번째 구간에서도, 샌드파이퍼는 가상머신 1, 2, 3 모두가 현재 상태에서 워크로드를 모두 수용할 수 있다고 판단하고 자원 할당을 수행하지 않게 된다. 그렇지만 그림에서 보이듯, 클라이언트에서 측정한 응답시간은 SLA를 위반한다. 반면에 제안한 기법에서는 어플리케이션 성능 분석을 통해, 요청 응답 시간의

표 2 워크로드 시나리오 2

	각 가상머신의 부하 (초당 평균 요청수)			서비스시간 /요청(ms)
구간 (s)	1	2	3	
	240	240	60	
가상머신1	30	35	45	6.6
가상머신2	30	35	45	6.6
가상머신3	120	130	180	1.7

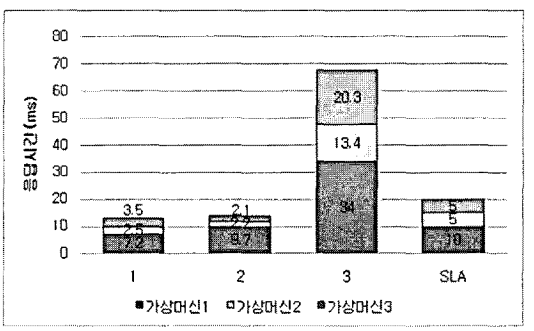

그림 7 웹서버 평균 응답 시간: 샌드파이퍼

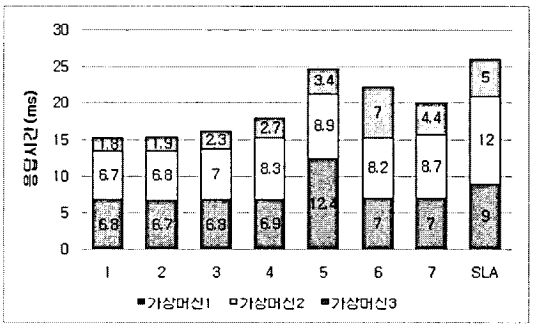

그림 8 웹서버 평균 응답 시간: 제안한 CPU 할당 기법

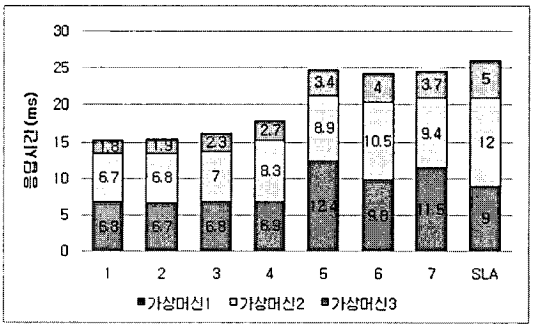

그림 9 웹서버 평균 응답 시간: 샌드파이퍼

총합이 요구되는 응답 시간의 총합보다 커서, CPU 우선순위 값의 조정을 통해 처리할 수 없음을 판단하고, 마이그레이션 결정을 내려서 SLA를 위반하지 않도록 한다.

표 3은 5-7번째 구간에서 가상화 서버에 과부하를 발생시키는 시나리오이다. 그림 8과 그림 9는 각각 제안한 CPU 할당 기법을 적용할 때와 샌드파이퍼를 적용할 때, 클라이언트에서 측정한 요청 응답 시간을 나타낸다.

그림 8의 제안한 기법에서는 5번째 구간에서 가상머신1에서 요구하는 응답 시간(9 ms)이 충족되지 않음을 감지하고, 응답 시간의 95번째 백분위수 총합(27.4 ms)이 요구하는 응답 시간 합(26 ms)보다 작으므로, 이주 결정이 아닌 CPU 우선순위 값을 조정한다. 제한한 우선순위 값 할당 기법에 의해, 다른 가상머신의 부하를 모두 처리하면서, 전체 스케줄링 지연을 최소화하도록 Credit 값을 재계산하므로, 가상머신2와 가상머신3은 기존의 Credit을 유지하고, 가상머신1은 기존의 값보다 50%가 증가된 Credit을 부여받는다, 이를 통해 6번째 구간부터는 모든 가상머신이 SLA를 위반하지 않고 어플리케이션을 처리할 수 있도록 한다. 이때, 동일한 워크로드가 지속되는 상황에서 전체 응답 시간이 줄어든 이유는, 가상머신1에 대해 가상머신2와 가상머신3의 CPU사용에 인한 스케줄링 지연뿐만 아니라 Domain-0의 패킷처리로 인해 발생되는 스케줄링 지연 또한 감소하였기 때문이다. 우선순위 값 할당에 의해 소유한 Credit이 많아진 가상머신1은, Domain-0으로부터 패킷이 전달되었을 때 우선적으로 스케줄링 될 확률이 높아지므로 가상머신2나 가상머신3의 요청 처리에 의한 지연이 줄어들게 된다. 이에 따라 Domain-0에서 상대적으로 I/O작업이 많은 가상머신3의 패킷을 처리함으로써 발생되는 지연 또한 감소하게 된다. 반면 샌드파이퍼는 모든 가상머신에 발생한 부하 λ_{peak}를 수용할 수 있다고 판단하고 할당을 수행하지 않는데, 그림 9에서 CPU 우선순위 값을 할당하지 않은 상태의 가상머신들의 응답 시간 변화를 볼 수 있다. 즉, 5번째 구간에서 발생한 과부하가 6번째와 7번째 구간에서도 지속됨에 따라, 가상머신1의 요구 응답 시간이 지켜지지 않음을 확인할 수 있다.

4.4 CPU 우선순위 할당 기법의 오버헤드

샌드파이퍼와 논문에서 제안하는 할당 기법은 CPU 자원 할당량 추정을 위해 동일한 방법으로 데이터를 수집한다. Xenmon을 사용하여 가상머신들의 CPU 활용 정보를 모니터링하고, 성능을 추정하기 위해 각 가상머신의 어플리케이션 로그를 모니터링 한다. 따라서 동일한 시간 동안, 제안한 CPU 우선순위 할당 기법과 샌드파이퍼의 CPU 할당량 추정 연산에서 발생하는 오버헤드를 측정하고 비교하였다.

그림 10은 실험의 결과를 보여준다. 제안한 기법은 샌드파이퍼와 달리 각 가상머신의 성능을 추정하기 위해 다른 가상머신의 상황을 고려하기 때문에, 시스템 내의 가상머신 수가 증가할수록 연산 오버헤드가 증가한다. 그러나 Xen 가상화 서버에서 운영할 수 있는 가상머신의 최대 대수는 32대로, 발생 가능한 최대 오버헤드는 60ms에 불과하다. 또한 위의 연산은 임의의 가상머신에

표 3 워크로드 시나리오 3

구간 (s)	각 가상머신의 부하 (초당 평균 요청수)							서비스시간/요청 (ms)
	1	2	3	4	5	6	7	
	60	60	60	60	60	60	60	
가상머신1	10	15	30	35	45	45	45	6.6
가상머신2	10	15	30	35	45	45	45	6.6
가상머신3	50	50	120	130	170	170	170	1.7

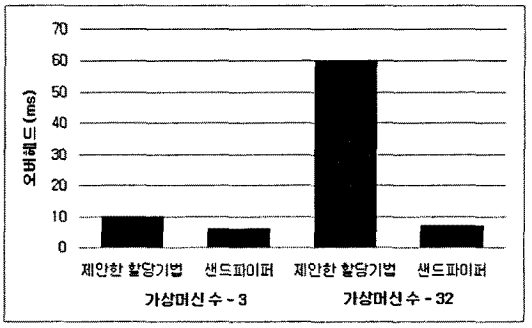

그림 10 CPU 자원 할당 기법의 오버헤드 비교

서 수행되는 어플리케이션의 응답 시간이 요구되는 값보다 커지는 경우에만 이루어지므로 연산 오버헤드는 무시할 수 있다. 따라서 서버가 장시간 운영되는 환경에서도 제안한 기법이 문제없이 사용될 수 있다.

5. 결론 및 향후 과제

본 논문에서는 다수의 가상서버가 하드웨어 자원을 공유하는 가상화 환경에서 발생하는 가상머신 스케줄링 지연에 의해 어플리케이션 QoS가 만족되지 못할 수 있다는 사실에 주목하고, 스케줄링 지연을 반영한 어플리케이션 성능 추정 방법과 성능 추정 결과를 바탕으로 한 가상머신의 우선순위 값 조정을 통한 CPU 자원 관리 기법을 제안했다. 또한 성능 분석을 통해, 제안한 성능 추정 방법이 기존 데이터센터에서 사용했던 큐잉 모델에 의한 분석보다 가상머신의 웹 서버 어플리케이션의 성능을 정확히 평가함을 확인했다. 그리고 제안한 성능 모델에 기반을 둔 CPU 할당 기법 또한 기존 샌드파이퍼의 방법보다 QoS만족도를 높여줄 수 있음을 확인하였다.

그러나 논문에서 제안한 성능 추정방법은 웹서버 어플리케이션을 운영하는 가상머신의 성능 분석에 제한되어있다. 우선 웹서버 어플리케이션 외의 스트리밍 서버와 같은 I/O 작업 중심 어플리케이션에 대해서는 Domain-0의 CPU 우선순위 값의 조정이 고려될 필요가 있다. 그리고 가상머신의 어플리케이션 성능을 보다 정확히 추정하기 위해서는 비단 CPU만이 아닌 메모리, 네트워크 자원 관리를 위한 성능 모델링 또한 필요하다. 현재 Xen의 Credit 스케줄러에서 제공하는 스케줄링 파라미터 값 조정을 통한 관리는 실시간 어플리케이션의 실시간성을 보장해 주기 어렵다. 따라서 스케줄러 수준에서 보다 나은 QoS를 제공해줄 수 있도록, 가상머신 스케줄러의 확장이 필요하다.

참 고 문 헌

[1] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A., "Xen and the Art of Virtualization," *Proceedings of the 19th ACM symposium on Operating Systems Principles*, pp.164-177, Oct. 2003.

[2] Wood, T., Shenoy, P., Venkataramani, A., and Yousif, M., "Black-box and gray-box strategies for virtual machine migration," *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation*, pp.229-242, April 2007.

[3] Chen, Y., Iyer, S., Liu, X., Milojicic, D., and Sahai. A., "SLA Decomposition: Translating Service Level Objectives to System Level Thresholds," *Proceedings of the 4th IEEE International Conference on Autonomic Computing*, p.3, June 2007.

[4] Appleby, K., Fakhouri, S., Fong, L., Goldszmidt, G., Kalantar, M., Krishnakumar, S., Pazel, D.P., Pershing, J., and Rochwerger, B., "Océano - SLA Based Management of a Computing Utility," *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management*, pp.855-868, May 2001.

[5] Rolia, J., Cherkasova, L., Arlitt, M., and Machiraju, V., "An automated approach for supporting application QoS in shared resource pools," *Proceedings of the 1st International Workshop on Self-Managed Systems and Services*, May 2005.

[6] Xu, J., Zhao, M., Fortes, J., Carpenter, R., and Yousif, M., "Autonomic Resource Management in Virtualized Data Centers Using Fuzzy Logic-based Approaches," *Proceedings of Cluster Computing*, vol.11 no.3, pp.213-227, Sep. 2008.

[7] X. Zhu, D. Young, B. J. Watson, Z. Wang, J, Rolia, S. Sighal, B. McKee, C. Hyser, D. Gmach, R. Gardner, T. Christian, and L. Cherkasova, "1000 Islands: Integrated Capacity and Workload Management for the Next Generation Data Center," *Proceedings of International Conference on Autonomic Computing (ICAC '08)*, 2008.

[8] Govindan, S., Nath, Arjun R., Das, A., Urgaonkar, B., and Sivasubramaniam, A., "Xen and co.: communication-aware CPU scheduling for consolidated xen-based hosting platforms," *Proceedings of 3rd International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, pp.126-136, June 2007.

[9] Ongaro, D., Cox, A. L., and Rixner, S., "Scheduling I/O in virtual machine monitors," *Proceedings of ACM/Usenix International Conference on Virtual Execution Environments*, pp. 1-10, Mar. 2008.

[10] Credit Scheduler, http://wiki.xensource.com/xenwiki/CreditScheduler

[11] Mosberger, D., and Jin, T., "httperf: A Tool for Measuring Web Server Performance," In *Proceedings of 1th Workshop on Internet Server Performance*, pp.59-67, June 1998.
[12] Gupta, D., Gardner, R., and Cherkasova, L., "XenMon: QoS Monitoring and Performance Profiling Tool," *Technical Report HPL-2005-187*, 2005.
[13] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing Performance Isolation Across Virtual Machines in Xen," *In Proceedings of ACM/IFIP/USENIX 7th International Middleware Conference (Middleware '06)*, 2006.

양 은 지
2007년 서강대학교 컴퓨터공학과(공학사). 2009년 서강대학교 컴퓨터공학과 대학원(공학석사). 2009년~현재 삼성전자 생산기술연구소 연구원. 관심분야는 시스템 가상화, 임베디드 시스템

최 현 식
2007년 서강대학교 컴퓨터공학과(공학사). 2010년 서강대학교 컴퓨터공학과 대학원(공학석사). 2010년~현재 한국산업은행 Core 뱅킹 전산실. 관심분야는 시스템 가상화, 임베디드 시스템

한 세 영
1991년 포항공과대학교 수학과(이학사) 2003년 서강대학교 정보통신대학원(공학석사). 2004년~현재 서강대학교 컴퓨터공학과 대학원 박사과정. 관심분야는 클라우드 컴퓨팅, 시스템 가상화, 자율적인 자원관리

박 성 용
정보과학회논문지 : 시스템 및 이론
제 37 권 제 3 호 참조