2014년 9월 전자공학회 논문지 제 51권 제 9 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 9, September 2014 http://dx.doi.org/10.5573/ieie.2014.51.9.103

논문 2014-51-9-12

# 멀티 프로세스를 사용한 가상 머신에서의 소프트웨어 로드 밸런서의 효율적인 물리 자원 활용 연구

## ( Improving Hardware Resource Utilization for Software Load Balancer using Multiprocess in Virtual Machine )

김 민 수*, 김 승 훈*, 이 상 민**, 노 원 우***

( Minsu Kim, Seung Hun Kim, Sang-Min Lee, and Won Woo Ro© )

요 약

클라우드 컴퓨팅 서비스 환경에서 가상화 기술은 클라우드 컴퓨팅을 위한 필수 요소로 자리잡고 있다. 가상화는 한정된 물리 자원을 공유하므로 가상 머신에 대한 자원 할당 관리는 중요하다. 일련의 작업은 하이퍼바이저에 존재하는 스케줄러에 의해 이루어지는데 특정 가상 머신에 I/O 요청이 집중되는 경우, 기존의 스케줄러는 이에 대한 처리가 미흡하다. 이는 특히, 가상 머신 상에서 소프트웨어 로드 밸런서를 구동시킬 때 두드러진다. 본 논문에서는, 이를 해결하기 위해 가상화 환경에서 동작하는 소프트웨어 로드 밸런서의 성능을 향상시킬 수 있는 구조를 제안한다. 가용 유휴 자원이 존재할 경우, 스케줄러와 소프트웨어 로드 밸런서 간의 통신을 통해 멀티 프로세스로 동작함으로써 유휴 자원을 활용할 수 있도록 한다. 이를 통해 가상 머신에서 할당하는 자원 변경에 의한 오버 헤드 없이 로드 밸런서의 성능을 향상시킬 수 있음을 보인다.

Abstract

In the virtualized server systems, a scheduler in a hypervisor is responsible to assign physical resources for virtual machines. However, the traditional scheduler is hard to provide optimized resource allocation considering the amount of I/O requests. Especially, the drawback hinders performance of software load balancer which runs on virtual machines to distribute I/O requests from the clients. In this paper, we propose a new architecture to improve the performance of software load balancer using multiprocess. Our architecture aims to improve hardware resource utilization and overall performance of the server systems which utilize virtualization technology. Experimental results show the effectiveness of the proposed architecture for the various cases.

**Keywords** : Virtualizastion, Software load balancer, I/O, Multiprocess

* 학생회원, *** 정회원, 연세대학교 전기전자공학부
(School of Electrical and Electronic Engineering, Yonsei University)
** 정회원, 한국전자통신연구원
(Electronics and Telecommunications Research Institute)
© Corresponding Author(E-mail: wro@yonsei.ac.kr)
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발 사업의 일환으로 수행되었음 [14-000-05-001, 스마트 네트워킹 핵심기술 개발]
접수일자: 2014년8월14일, 수정일자: 2014년8월28일, 수정완료: 2014년9월11일

## Ⅰ. 서 론

클라우드 컴퓨팅 서비스 환경에서 클라이언트는 언제 어디서든 필요로 하는 어플리케이션을 사용할 수 있다. 이를 위해 서비스 제공자는 거대 규모의 데이터센터 및 다수의 컴퓨팅 자원을 운영해야 할 필요가 있으며 관련 기능의 중요성은 더욱 증가하고 있다[6]. 이에 따라, 서비스 제공을 위한 인프라 구축 및 유지비용 절감에 대한 연구가 활발히 진행되었다. 특히 서버 가상

화 기술은 효율적인 물리 자원 사용을 가능하게 함으로써 클라우드 컴퓨팅 인프라를 위한 필수 요소로 자리 잡게 되었다.

가상화는 한정된 물리 자원을 바탕으로 구현되기 때문에 가상 머신의 자원 사용에 대한 스케줄링이 중요하다. 일련의 작업은 일반적으로 하이퍼바이저(Hypervisor)에 존재하는 스케줄러에 의하여 이루어지며, Xen에서 사용하는 Credit 스케줄러의 경우 각각의 가상머신에 할당된 Credit 을 바탕으로 물리 자원 사용 시간을 제한 한다[2~3]. 다른 예로써, VMware 사의 ESXi CPU 스케줄러는 스케줄링 요소 중 하나로 I/O 요청을 고려하여 자원을 할당 한다[4]. 그러나 특정 가상 머신에 I/O 요청이 집중되는 경우, Credit 스케줄러와 ESXi CPU 스케줄러는 이에 대한 처리가 미흡하다[1, 5].

기존의 하이퍼바이저가 갖는 이러한 한계점은 특히 가상화 환경에서 소프트웨어 로드 밸런서를 구동할 때 두드러진다. 소프트웨어 로드 밸런서가 가상 머신에서 구동 될 때 로드 밸런서가 처리해야 하는 I/O 에 대한 고려가 없는 물리 자원 스케줄링은 물리 자원 사용의 불균형 및 원활하지 못한 패킷 처리를 야기하여 서버 가상화의 장점을 저해하는 요인이 된다.

가상화 환경에서의 자원 활용 및 I/O 에 대한 연구는 다양하게 진행되고 있다[9~10]. 이 연구들의 경우, 사용자를 위한 정보 제공 연구 및 추가적인 하드웨어 자원 확장 시에 발생하는 I/O 와 관련된 연구만 언급되어 있으며 가상 머신으로의 직접적인 I/O 처리와 관련된 언급은 없다.

본 논문에서는, 가상화 환경에서 동작하는 소프트웨어 로드 밸런서의 성능을 향상할 수 있는 구조를 제안한다. 본 논문에서 제안하는 하이퍼바이저는 로드 밸런서와의 통신을 통해 가용 유휴 자원이 존재할 경우, 소프트웨어 로드 밸런서의 멀티 프로세스 실행을 통해 유휴 자원을 활용할 수 있도록 한다. 이를 통해 가상 머신에 할당하는 자원 변경에 의한 오버 헤드 없이 로드 밸런서의 성능을 향상할 수 있음을 보인다.

본 논문의 구성은 다음과 같다. Ⅱ장절에서는 가상화와 I/O 로드 밸런싱의 특징 및 제안하는 구조에 대해 기술한다. Ⅲ장에서는 실험을 통해 제안하는 구조의 타당성을 검증한다. 마지막으로 Ⅳ장에서는 결론 및 향후 연구 방향에 대해 서술한다.

## Ⅱ. 배경 및 제안하는 구조

### 1. 가상화

가상화 기술이 적용되지 않은 서버 환경에서는 서비스 제공을 위해 개별적인 하드웨어가 필수적이다. 즉, 개별의 목적을 가진 서버가 독립적으로 작동하며, 서비스 제공을 위한 서버 환경이 다르다면 이를 위한 별도의 장치가 모두 갖추어져야 한다. 이는 서버 시스템 구축 및 유지비용 증가의 문제로 이어진다. 가상화 기술은 이러한 정적인 방식의 서버 운용으로 인한 문제를 해결하기 위해, 물리 자원의 추상화를 바탕으로 서비스 제공을 위한 독립적인 가상 머신을 제공한다.

하이퍼바이저를 통해 추상화된 가상 머신 (Virtual Machine) 은 하이퍼바이저 내부의 스케줄러에 의해 물리 자원을 할당 받는다. 이를 통해, 제한된 물리 자원 환경에서도 복수의 가상 머신을 운용 하며 복합적인 서비스를 제공하고 서버 시스템의 구축 및 유지비용을 절감할 수 있다.

### 2. 로드 밸런싱

로드 밸런싱은 하나의 서버에 집중된 서비스 요청을 분산시켜 이를 처리하기 위한 부하를 줄이고 서버 인프라 전체의 자원을 효율적으로 사용할 수 있도록 지원하는 기술이다. 하나의 서버에 있어 한정된 물리 자원를 활용해야 하는 가상화 기술의 특성을 고려할 때, 로드 밸런싱은 가상화 기술의 효과를 극대화 할 수 있는 주요 요소 중 하나이다. 본 논문에서 제안하는 구조는 상황에 따라 소프트웨어 로드 밸런서를 병렬 실행함으로써, 클라이언트의 I/O 처리 요청에 효과적으로 대응할 수 있는 가상화 환경을 제공한다.

### 3. 제안하는 구조

기존의 하이퍼바이저 스케줄러는 가상 머신의 CPU 사용 요청을 주로 고려하여 물리 자원을 할당한다. 따라서 특정 가상 머신에 집중된 I/O 요청이 발생할 경우 효율적인 물리 자원 활용을 기대하기 어렵다.

이를 해결하기 위해 본 논문에서는, 로드 밸런서를 구동하는 가상머신과 하이퍼바이저의 통신 모듈을 제안한다. 통신 모듈은 집중된 I/O 요청이 발생할 경우 스케줄러에게 이에 대한 정보를 전달하고, 스케줄러는 가용 유휴 자원의 유무를 로드 밸런서에게 응답한다. 만

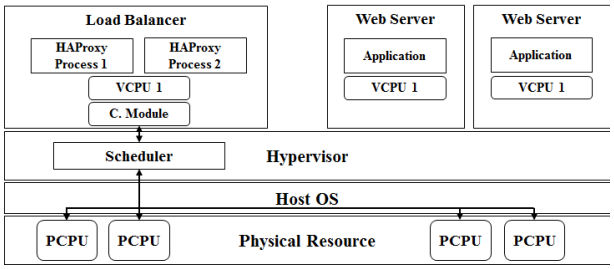



그림 1. 통신 모듈이 적용된 제안 구조
Fig. 1. A proposed architecture with communication module.

약, 가용 유휴 자원이 존재하는 경우 로드 밸런서는 멀티 프로세스로 동작하여 I/O 로드 밸런싱의 성능을 높인다.

그림 1은 도식화 된 제안구조이다. 위의 구조에서 물리 자원 (Physical CPU) 은 4개의 코어가 존재하며, 가상 머신들에게 할당된 가상 자원 (Virtual CPU)은 총 3개로 1 개의 물리 자원이 가용 유휴 자원으로 존재한다. 집중된 I/O 요청이 발생하는 경우, 통신을 통해 로드 밸런서는 1 개의 가용 유휴 자원을 파악하고 멀티 프로세스를 실행한다. 가용 유휴 자원을 사용함에 따라 물리 자원 사용률은 높아지고 가상 머신에서 동작하는 어플리케이션의 I/O 응답시간은 짧아진다.

## Ⅲ. 실험 환경 및 결과

### 1. 실험 환경 구성

본 연구에서는 제안하는 구조의 유효성을 입증하기 위하여 VMware 사의 VMware Workstation을 활용하여 가상화 환경을 구성하였다. 2개의 가상 머신 위에 웹 서버를 구현하였고 각각의 웹 서버로 포워딩되는 I/O 요청들의 시간을 측정하기 위해 httperf 툴을 사용하였다[7]. 다른 가상 머신 위에는 소프트웨어 로드 밸런서로 HAProxy를 사용하였으며 호스트 머신은 4 코어 Intel i5-2300 의 CPU 와 8 GByte 의 메모리로 구성되어 있다[8]. 호스트 머신의 운영체제는 윈도우 7, 가상 머신의 운영체제는 Ubuntu 12.04 를 사용하였다.

주어진 물리 자원의 양과 로드 밸런서의 병렬 실행에 변화에 따라 3 가지의 경우를 가정하여 실험을 진행하였다. 먼저, HAProxy 프로세스와 가상 프로세서의 개수 변화를 보기 위해 고정된 물리 자원 환경에서의 멀티 프로세스 실행 효과를 보았다. 두 번째 경우로 물리 자원 양과 HAProxy 멀티 프로세스 개수 변화를 보기 위해 물리 자원 증가에 따른 멀티 프로세스 실행 효과를 확인하였다. 마지막으로 물리 자원 경쟁 상황에서의 멀티 프로세스 실행 효과를 확인하기 위해 가상 자원이 물리 자원보다 많은 상황에서의 소프트웨어 로드 밸런서 병렬 실행 효과를 검증하였다. 평가 항목은 두 가지로써, 첫 번째는 250명의 동시 접속자가 생성하는 60,000개의 연결 요청에 대한 완료 시간을, 두 번째는 250명의 동시 접속자가 5 초 동안 보내는 I/O 요청의 수에 대한 응답 시간을 측정한다.

### 2. 응답 완료 시간 비교

그림 2는 첫 번째 평가 항목에 대한 결과 그래프를 나타낸다. 그래프 2(a)는 가용 유휴 자원이 존재하는 경우로서, 로드 밸런서의 자원은 적지만 프로세스를 2개로 증가시킨 결과 I/O 요청을 더 빠른 시간에 처리할 수 있음을 보여 준다. 그러나 4 개의 프로세스로 증가

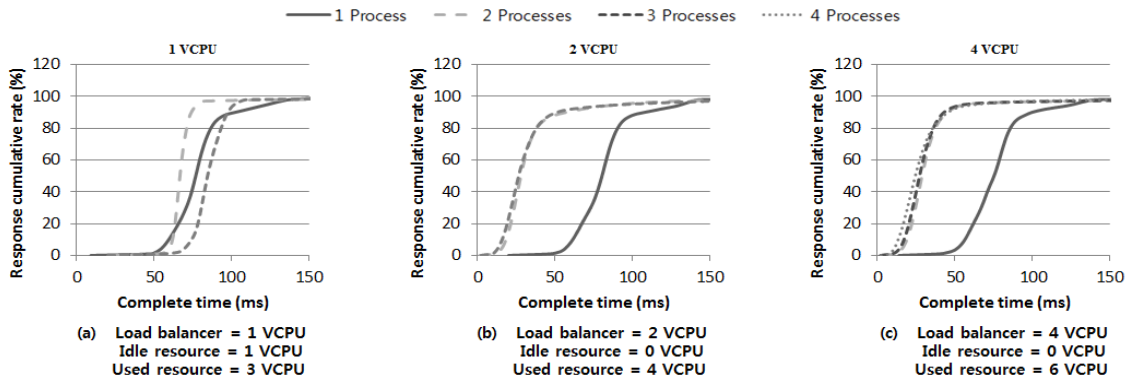



그림 2. 가용 유휴 자원 유무에 따른 I/O 응답 완료 시간 누적 분포
Fig. 2. A cumulative distributions of I/O response time in resources available.

한 경우 응답 완료 시간의 분포가 더 느려지는 것을 관찰할 수 있는데, 이는 처리해야 할 패킷을 저장하는 공유 큐에서 병렬화로 인해 혼잡 현상이 발생했기 때문으로 유추할 수 있다. 즉, 가용 유휴 자원이 존재 하는 경우, 제한적인 멀티 프로세스 실행은 I/O 처리율을 증가시킬 수 있음을 알 수 있다.

그래프 2(b)는 할당 받은 가상 자원이 2(a) 의 경우보다 많고, 물리 자원 사용에 대해 웹 서버 및 로드 밸런서의 가상 머신 간 경쟁 상황이 존재하지 않은 경우의 실험 결과이다. 로드 밸런서를 구동하는 가상 머신으로의 할당 자원이 증가함에 따라 멀티 프로세스를 통해 첫 I/O 응답이 빨라졌음을 알 수 있다. 그러나 가용 유휴 자원은 존재하지 않으므로 HAProxy 의 프로세스 수를 3 개 이상으로 증가시켜도 응답 시간의 변화폭은 크지 않다.

그래프 2(c)는 로드 밸런서에 물리 자원보다 많은 양의 가상 자원을 할당함으로써 자원 경쟁 상황이 발생하는 경우를 나타낸다. 비록 로드 밸런서가 4 개의 가상 자원을 할당 받았지만 물리 자원 수의 제약에 의해 성능 증가에 한계가 발생한다. 그러나 그래프 2(a)의 2 개 프로세스 실행 상황과 마찬가지로 프로세스 수 증가에 의한 전체 응답 속도는 향상함을 관찰할 수 있다. 이는 자원 경쟁 상황에서도 멀티 프로세스를 통해 추가로 성능이 향상될 수 있음을 의미 한다.

## 3. I/O 응답 완료 비교

두 번째 실험에 대한 결과 그래프는 그림 3에 나타나 있다. 로드 밸런서가 1개, 2개, 4개의 가상 프로세서를 할당 받았을 때의 제한된 시간 내에 응답 가능한 패킷의 수를 프로세스 수에 따라 보여 준다.

할당받은 가상 프로세서의 수가 1개인 경우 로드 밸런서의 병렬 실행에 관계없이 제한된 시간 내에 응답 가능한 I/O 요청 수는 큰 변화가 없다. 반면에, 로드 밸런서의 가상 머신에 할당된 가상 프로세서가 2 개인 경우 응답 가능한 클라이언트의 요청 수는 약 2 배로 증가함을 관찰할 수 있다. 즉, 짧은 시간 내에 높은 성능을 내기 위해서는 병렬 실행 뿐 아니라 실제 가상 자원의 충분한 할당이 필요함을 알 수 있다.

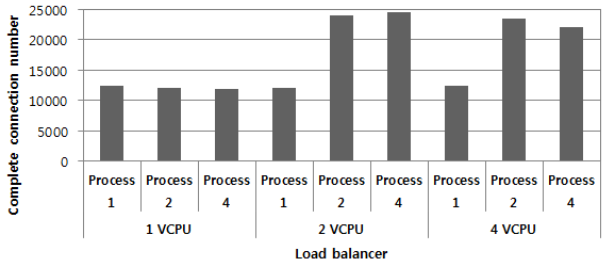



그림 3. 제한 시간 내에 처리 완료된 I/O 요청 개수
Fig. 3. The number of complete I/O requests within limited time.

4 개의 가상 프로세서를 로드 밸런서의 가상 머신이 할당받은 경우에도 유사한 경향을 확인할 수 있다. 로드 밸런서의 프로세스 수를 늘려가도 사용할 수 있는 물리 자원의 한계에 의해 성능은 2 개의 프로세스를 실행할 때 이상으로 증가하지 않음을 알 수 있다.

## 4. 제안 구조 적용

그림 4는 제안 구조를 바탕으로 자원 할당을 통한 멀티 프로세스의 효과를 나타낸 그래프이다. 추가적인 가상 머신을 구성하여 할당된 가상 자원은 총 10 개 인 환경에서 실험하였다. Base는 기본 스케줄링, L/A는 로드 밸런서와 추가 가상 머신에 물리 자원 집중 할당, L은 로드 밸런서에 물리 자원 집중 할당이며, L/W는 로드 밸런서와 웹 서버 가상 머신에 물리 자원을 집중 할당한 것이다.

그래프는 가용 유휴 자원을 로드 밸런서에 할당함에 따라 응답 시간이 짧아지는 것을 보인다. 반면, I/O 요청이 적은 추가적인 가상 머신에 자원을 할당한 경우 응답 성능은 감소하였다. 이를 통해 로드 밸런서에 자원을 할당하여 멀티 프로세스로 동작하는 것이 I/O 처리율을 향상시킨다는 것을 관찰할 수 있다.

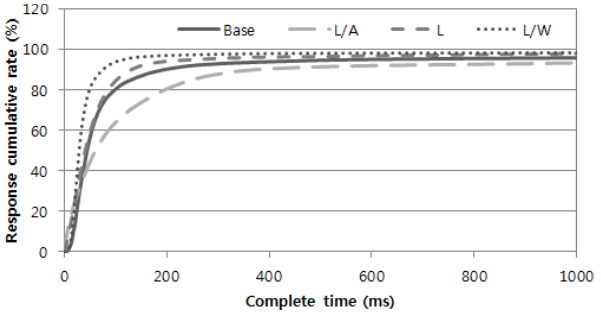



그림 4. 제안하는 구조적용에 따른 응답 시간 분포
Fig. 4. A distributions of response time for the resource allocation.

## Ⅳ. 결 론

본 논문에서는 가상화 환경에서의 가용 유휴 자원을 이용하여 I/O 처리율을 향상시키는 구조를 제안하였다. 실험을 통해 가용 유휴 자원이 존재할 경우 멀티 프로세스를 통해 I/O 처리 성능을 향상시킬 수 있음을 보였다. 로드 밸런서는 멀티 프로세스를 실행하기 위해 하이퍼바이저 내부의 스케줄러와 통신한다. 스케줄러는 가상 자원의 변경을 통해 로드 밸런서에 자원을 할당할 수 있지만, 이는 가상화 환경에서 오버헤드로 동작한다. 그러므로 할당 자원 변경보다 멀티 프로세스로 동작하는 것이 오버헤드를 발생시키지 않고 효율적으로 동작하며 실험을 통해 이를 증명하였다.

더 나아가 본 연구결과를 바탕으로 가상화 환경에서 동작하는 로드 밸런싱의 자원 할당 알고리즘의 개선 방안을 연구하고 로드 밸런서 내부의 통신 모듈과 하이퍼바이저 간의 프로토콜을 바탕으로 한 자원 변경의 오버헤드 감소 방안에 대한 연구를 진행할 계획이다.

## REFERENCES

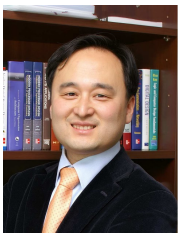
[1] S. Jeffrey, “I/O Virtualization Bottlenecks in Cloud Computing Today”, in Proceedings of the 2nd conference on I/O virtualization, USENIX Association, pp. 5-11, March 2010.
[2] C. Ludmila and R. Gardner, “Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor”, in Proceedings of USENIX Annual Technical Conference, General Track, 2005.
[3] B. Paul, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization”, in Proceedings of the 9th ACM sysposium on Operating systems principles, ACM, pp. 164-177, October 2003.
[4] VMware, Inc., The CPU Scheduler in VMware vSphere 5.1, http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-CPU-Sched-Perf.pdf
[5] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, “Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments”, in Proceedings of the 3rd International Conference on Cloud Computing (CLOUD), IEEE, July 2010.
[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A View of Cloud Computing”, Communication of the ACM 53.4, pp. 50-58, March 2010.
[7] www.hpl.hp.com/research/linux/httperf
[8] haproxy.1wt.eu
[9] Ji-Eun Lee, Tae-Hoon Kim, and Jaechun No, “VDI Platform Design for providing the Optimized VM Assignment”, Summer Conference, IEIE, pp. 1317-1320, July 2013.
[10] Woong Shin and H.Y. Yeom, “Evaluating Virtual I/O overhead when using Low Latency Storage devices for Virtual Machines”, CEIC, IEIE, pp. 75-78, December 2011.

## 저 자 소 개


**김 민 수**(학생회원)
2013년 아주대학교 전자공학과 졸업 (학사)
2013년 연세대학교 전기전자 공학과 입학 (석사과정)
<주관심분야 : 가상화, 임베디드 시스템, 컴퓨터 시스템 등>

**김 승 훈**(학생회원)
2009년 연세대학교 전기전자 공학과 졸업 (학사)
2011년 연세대학교 전기전자 공학과 졸업 (석사)
2011년 연세대학교 전기전자 공학과 입학 (박사과정)
<주관심분야 : 컴퓨터 시스템, 트랜잭셔널 메모리 등>

**이 상 민**(정회원)
1994년 경북대학교 전자공학과 졸업 (학사)
1996년 경북대학교 전자공학과 졸업 (석사)
2000년~현재 ETRI 책임 연구원
<주관심분야 : 패킷 전송 네트워크, 스마트 인터넷 등>



**노 원 우**(정회원)-교신저자
1996년 연세대학교 전기공학과 졸업(학사)
1999년 University of Southern California 졸업(석사)
2004년 University of Southern California 졸업(공학박사)
2003년~2004년 Apple Computer Inc. 인턴 연구원
2004년~2007년 California State University 전기 및 컴퓨터공학과 조교수
2006년~2007년 ARM Inc. 소프트웨어 엔지니어
2007년~현재 연세대학교 전기전자공학과 부교수
<주관심분야 : 고성능 마이크로프로세서 디자인, 컴파일러 최적화, 임베디드 시스템 디자인 등>