크레딧-스케줄러의 불공평적 IO 작업 분배에 관한 연구

최재원*, 이재학*, 유헌창*, 이은영**
*고려대학교 정보대학 컴퓨터학과
**동덕여자대학교 컴퓨터학과
e-mail:{dps2, smreodmlvl, yuhc}@korea.ac.kr, elee@dongduk.ac.kr

A Study on Unfairness for I/O Intensive Works on Credit-Scheduler

JaeWon Choi*, JaeHak Lee*, Heonchang Yu*, EunYoung Lee**
*Dept. of Computer Science and Engineering, Korea University
**Dept. of Computer Science, Dongduk Women's University

요 약

하이퍼바이저 Xen 은 처음 개발되었을 때부터 현재까지 크레딧-스케줄러를 이용하고 있다. 크레딧-스케줄러는 CPU 이용률을 높이기 위해 디자인되어 있으며 효율적인 I/O 처리를 위해 부스트라는 개념을 도입하였지만, 부스트로 인하여 공정성이 깨질 수 있는 문제를 가지고 있다. 이 논문에서는 I/O 중심 처리 환경에서 크레딧-스케줄러의 불공정성을 증명하였으며, 실험을 통해 네트워크 성능에서 최대 약 60%의 차이를 보임을 확인할 수 있었다.

1. 서론

컴퓨터의 발명이래 컴퓨터의 자원을 보다 효율적으로 사용하기 위한 연구는 계속 진행되었다. 그 중에서도 서버의 경우 대부분 최대 성능의 극히 일부분만 쓰므로 효율성 증대가 중요한 연구과제가 되었다. 예를 들어 수강신청 서버의 경우, 수강신청 기간 외에는 처리량이 없으며, 게임 서버의 경우 주말이나 퇴근 이후 시간외에는 처리량이 거의 없다. 최대 처리량이 몰리는 시점을 기준으로 서버를 구축해야 하니 평소에는 성능을 낭비하고 있는 셈이다. 최근에 주목받고 있는 가상화 기술은 이러한 문제를 해결하기 위해 이용되고 있다. 하나의 서버에서 다수의 OS 를 작동시켜 여러 작업을 동시에 수행시킬 수 있다.

기존 시스템과는 다르게 OS 하나가 하드웨어를 독점하지 않으므로 자원을 OS 별로 나누어 주어야 한다. 이에 기존의 OS 가 가지고 있던 하드웨어 관리 권한을 다른 프로그램이 가져야 할 필요성이 생겼다. 하드웨어 관리 권한을 가진 프로그램을 하이퍼바이저라 부른다. 하이퍼바이저는 OS 가 프로세스들에게 자원을 분배하듯 자원들을 OS 들에게 분배해 준다.

Xen 은 이러한 기능을 수행하는 하이퍼바이저 중 하나이다. 케임브릿지 대학에서 처음 개발되었으며 이제는 리눅스 재단에서 오픈소스로 개발을 진행하고 있다. 2003 년 처음 1.0 버전이 나온 이후 현재 4.12 버전까지 출시 되었으며, 현재 많은 클라우드 서버의 하이퍼바이저로 채택하였다.

이 논문은 2019 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구결과임(NRF-2019R1A2C1006754).

Xen 은 타입 1 하이퍼바이저, 즉 하드웨어를 바로 제어하는 하이퍼바이저이다. 덕분에 오버헤드가 적어 다른 하이퍼바이저보다 많은 수의 가상머신을 Xen 위에서 작동시킬 수 있다.

당연하게도 Xen 위에서는 다수의 가상머신이 동시에 실행되는 상황을 가정하고 개발되었다. 따라서 Xen 은 다수의 가상머신에게 CPU 시간을 분배해야 했다. 처음 개발될 때부터 디폴트-스케줄러로 크레딧-스케줄러를 채택하였다. 하지만 크레딧-스케줄러는 가상머신에 대한 CPU 점유 정도를 기반으로 스케줄링을 하기 때문에 I/O 작업적인 가상머신들에게는 불공정한 I/O 성능을 제공하고 있다.

본 논문에서는 Xen 에서 사용중인 크레딧-스케줄러가 I/O 작업에 관하여 공평성을 유지하는지를 실험 및 평가한다. 2 장은 Xen 아키텍처에 대한 간략한 설명을 다룬다. 3 장은 가상머신들의 I/O 작업에 대한 공정성 실험 수행 및 결과를 분석하며, 마지막 4 장은 실험 결과에 대한 결론으로 논문을 마무리한다.

2. Xen 아키텍처

Xen 은 하드웨어 바로 위에서 작동하여 CPU, 메모리, 인터럽트 제어를 지원하는 반가상화를 사용하여 기존 OS 가상화 환경에서의 I/O 지연시간을 단축시켰다. 또한 Xen 은 마이크로커널 디자인을 사용하기 때문에 굉장히 작고 가볍다. Xen 의 가상머신들은 두 가지 타입으로 분류할 수 있다. 사용자들에게 제공하는 도메인-U 와 모든 도메인-U 들의 입출력 작업을 제어하는 도메인-0 이다. 도메인-0 가 모든 도메인-U 의 입출력 작업을 제어하는 이유는 아직 많은 하드웨어가 가상화를 지원하지 않기 때문이다. 조금 더 자세

히 기술하자면, 이 하드웨어들 중에 다이렉트 메모리 접근이 가능한 하드웨어가 있고, 이 하드웨어를 이용하면 도메인-U 가 자신의 영역 외의 메모리를 침범할 수 있기 때문에 이를 막기 위함이다.

이러한 분리 드라이버 모델을 채택한 Xen 의 도메인-0 는 컨트롤-스택과 시스템 서비스를 수행하고, 공유 자원을 관리하며 실질적인 모든 도메인-U 들의 I/O 작업 요청을 수행하므로, 드라이버 도메인이라고 불리기도 한다.

2.1 크레딧-스케줄러

크레딧-스케줄러는 우선순위 기반 라운드 로빈 스케줄러에 기반하여 만들어진 스케줄러이다. 가상 머신들은 크레딧을 받아 CPU 를 사용할 때 마다 크레딧을 소모한다. 가상머신의 크레딧이 0 보다 작으면 오버, 0 보다 크면 더 높은 우선순위인 언더 상태를 가지게 된다. 오버 상태의 가상머신은 CPU 의 레디-큐에서 맨 뒤에 배치가 되고 언더 상태의 가상머신들이 다 처리되어야 비로소 CPU 를 점유할 수 있다. 각각의 가상 머신은 가중치를 부여받는다. 가중치는 일종의 중요도로서 가중치가 클수록 더 많은 크레딧을 받는다.

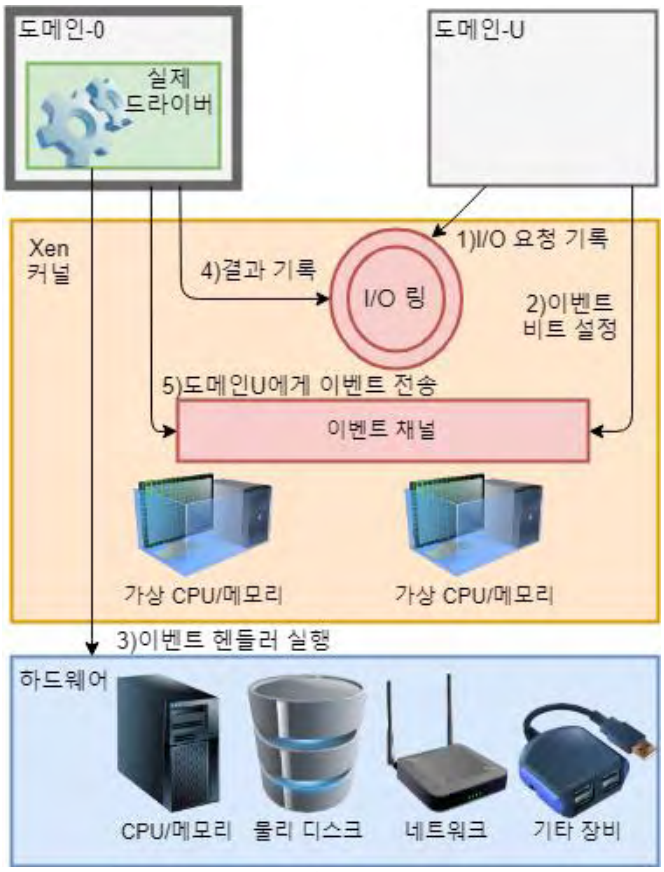
주목해야 할 점은 크레딧-스케줄러의 타임 슬라이스가 30ms 이라는 것이다. 리눅스의 CPU 스케줄러의 기본 타임 슬라이스 값은 100ms 이다. 여러 가상 머신이 Xen 위에서 작동되는 점을 고려하면 30ms 는 굉장히 큰 편이다. 이는 컨텍스트 스위치를 적게 발생시켜 CPU 처리량을 높인다는 전략이다.

큰 타임 슬라이스는 CPU 처리량은 증가시키지만, I/O 작업에는 효과적이지 못하다. 왜냐하면 타임 슬라이스가 크면 한번 작업을 처리한 이후 다시 작업을 처리하기까지 오랜 시간을 기다려야하기 때문이다. 대부분의 I/O 는 즉각적인 처리가 필요하다는 점을 주목해야 한다. 만일 처리에 지연이 생긴다면 음악이 뚝뚝 끊겨 들리고, 마우스가 느려 마우스의 움직임을 예상하면서 움직여야 하는 상황이 발생할 것이다.

2.2 Xen 의 I/O 작업

앞서 말한 상황을 벗어나기 위하여 Xen 은 가상머신의 I/O 작업 지연 현상이 발생하지 않도록 성능을 개선시켜야 했다. 이를 위해 언더 상태의 우선순위보다 높은 부스트라는 우선순위를 도입하였다[1]. 부스트 상태가 된 가상머신은 자기보다 낮은 우선순위의 가상머신을 선점하여 레디-큐의 맨 앞에 배치되어 10ms 동안 CPU 를 선점한다.

Xen 의 I/O 작업은 공유 링 버퍼, 이벤트 채널로 설명할 수 있다. 공유 링 버퍼는 도메인-U 가 물리 머신의 공유 자원에 직접 접근하지 못하므로, 공유 링 버퍼에 도메인-0 로의 I/O 작업 요청에 대한 요청 정보 및 응답 정보를 기록하는 공유 메모리 메커니즘으로 사용된다[1]. 이벤트 채널은 도메인-0 를 포함한 각 가상머신들이 자신과 관여된 I/O 요청이 존재하는지에 대한 알림 기능을 수행한다.



(그림 1) Xen 에서 I/O 처리과정

설명을 쉽게 따라갈 수 있도록 (그림 1)에서의 부여된 번호를 본문에도 표시했다. 1)도메인-U 에서 입출력 작업이 발생하면 공유 링 버퍼에 자신의 ID, I/O 요청 정보를 기록한다. 2)그 후, 도메인-0 로 이벤트를 보내 도메인-0 의 이벤트 채널에 이벤트 비트를 설정한다. 도메인-0 가 실행될 때 이벤트 채널에 이벤트 요청 비트가 설정되어 있다면, 부스트 상태가 되어 공유 링 버퍼에서 해당 도메인-U 의 이벤트 정보를 확인하여 3)실제 하드웨어로의 이벤트 핸들러를 수행한다. 4)도메인-0 는 다시 공유 링 버퍼에 I/O 작업 완료 이벤트에 대한 정보를 기입하고 5)도메인-U 의 이벤트 채널에 이벤트를 보낸다. 도메인-U 가 CPU 를 선점하고, 도메인-U 이벤트 채널을 확인 후 이벤트 요청 비트가 설정되어 있다면 부스트 상태가 되어 공유 링 버퍼를 통해 I/O 작업 요청 정보를 확인하여 그에 맞는 이벤트 핸들러를 수행한다.

2.3 문제점

스케줄러는 공평성, 처리량, 지연시간, 응답시간과 같은 여러 사항을 고려해야 한다. 그 중 이번 논문에서는 가상머신들 사이에서의 I/O 작업에 대한 공평성에 대해 다루려고 한다. Xen 에서는 부스트 상태를 도입하여 I/O 작업 처리량을 높였지만 공평성은 고려하지 않았다[2][3]. 이에 따라, 크레딧-스케줄러의 공평성이 어느 정도 어긋났는지를 I/O 작업에 대한 벤치마킹을 통해 확인하도록 한다.

3. 성능 평가 및 분석

<표 1> 실험 환경 비교표

	도메인-0	도메인-U	AWS 인스턴스
운영체제	우분투 14.04	우분투 14.04	우분투 18.04
vCPU 개수	2	2	1
메모리	1024MB	256MB	1024MB

실험 환경은 <표 1>과 같으며, 호스트 머신은 i5-4590T(2.00GHz) CPU 에 7GB 의 메모리를 가지고 있다.

3.1 입출력 작업 성능 평가 방법

Xen 에서는 가상화 기술을 하드웨어로부터 지원받는 HVM 과 커널과 드라이버로부터 가상화 기술을 지원받는 반가상화 환경인 PVM 환경이 있다. 본 벤치마킹은 반가상화 환경에서 실험을 진행하였다. 벤치마킹을 위해 iPerf3 를 활용하였다. iPerf3 는 네트워크 성능 테스트에 쓰이는 프로그램이다. 호스트의 PVM 과 AWS 인스턴스에서 각각 iPerf3 를 실행시켜 1 분동안 PVM 에서 다운로드 받은 양(그림 2)과 업로드 한 양(그림 3)을 각각 측정하였다.

동시에 수행시키기 위해 매 3 분마다 iPerf3 를 수행하도록 Bash 스크립트를 작성하여 작동시켰다. 다운로드와 업로드를 각각 10 번 수행하였다.

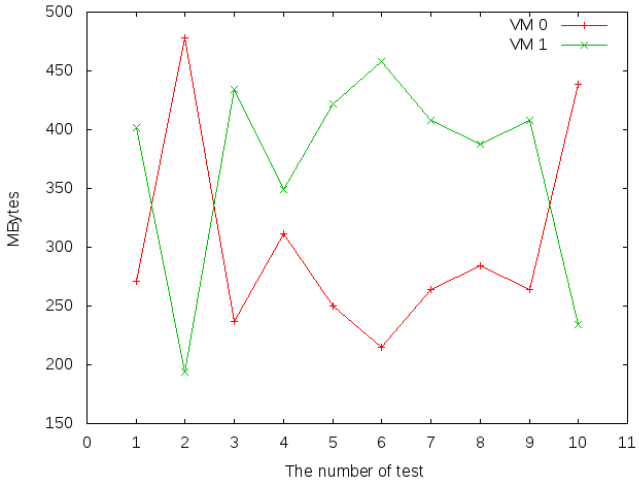
PVM 의 수만큼 AWS 인스턴스를 만들어서 일대일로 연결하였다. 하나의 인스턴스로 실험을 진행하지 않은 이유는, AWS 인스턴스에서도 불공평성이 존재할 가능성을 배재하기 힘들기 때문이었다. 하나의 인스턴스 안에서 발생하는 불공평성은 실험에 크게 영향을 미치지만, AWS 의 인프라는 굉장히 크므로, 이번 실험의 인스턴스들이 다른 인스턴스에 영향을 받을 것이라 생각하기 어렵다. 따라서 AWS 인스턴스 내부의 불공평성을 배제하기 위해 PVM 의 수만큼 AWS 인스턴스를 만들었다.

3.2 성능 불균형 현상 분석

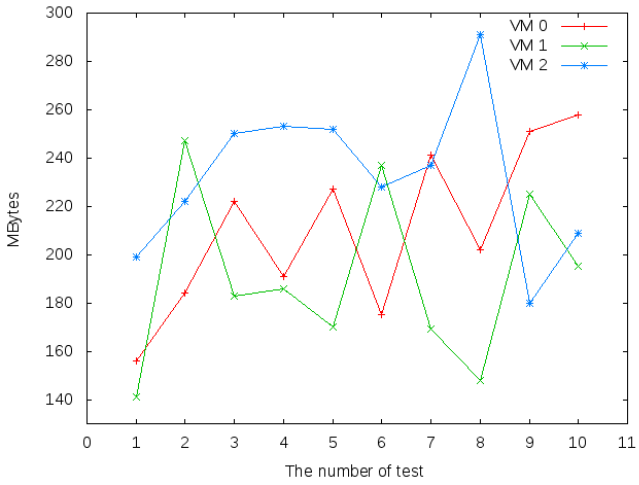
각 그래프의 세로축은 다운로드/업로드 한 양이며 단위는 메가바이트이다. 가로축은 몇 번째 실험인지를 나타낸다.

(그림 2)와 (그림 3)의 (a)를 보면 주로 특정 PVM 이 자원을 많이 할당받는 경향을 확인할 수 있다. 다른 그래프에서도 그러한 경향을 확인할 수 있지만 PVM 수가 적을 때만큼 두드러지게 나타나지는 않았다. (그림 2)와 (그림 3)을 보면 다운로드에서 자원이 불공평하게 배분된 정도와 업로드에서의 정도가 비슷함을 알 수 있다.

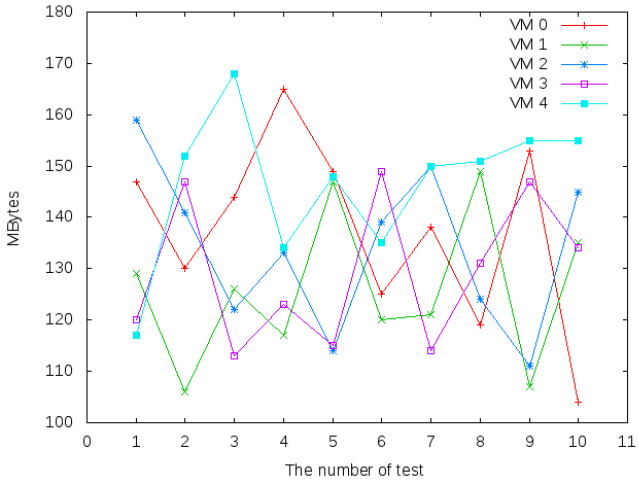
PVM 의 개수가 적을수록 자원 분배의 불공평성이 커졌다. PVM 이 5 개 있을 때 다운로드 업로드 처리량이 최대 33%, 47%, 3 개 있을 경우에는 49%, 48%, 2 개 있을 경우에는 59%, 58%의 차이가 발생하였다.



(a) PVM 이 2 개일 때 다운로드 벤치마크

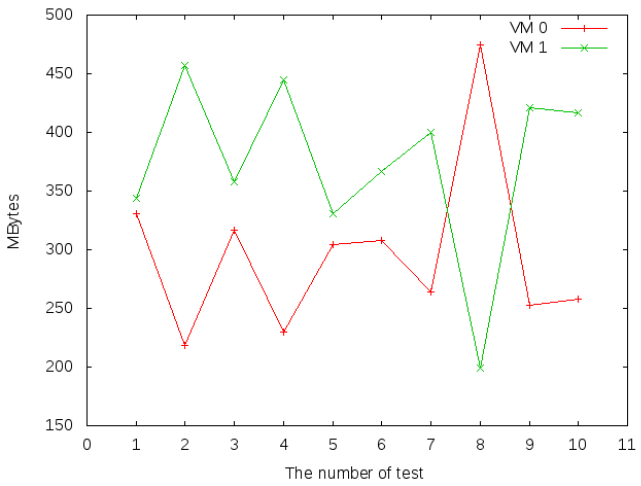


(b) PVM 이 3 개일 때 다운로드 벤치마크

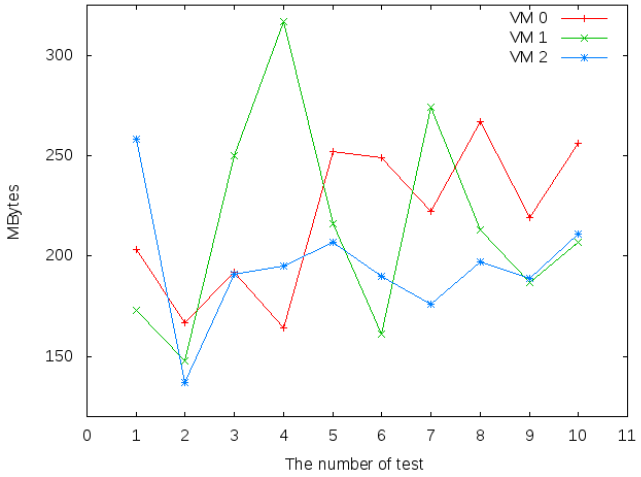


(c) PVM 이 5 개일 때 다운로드 벤치마크

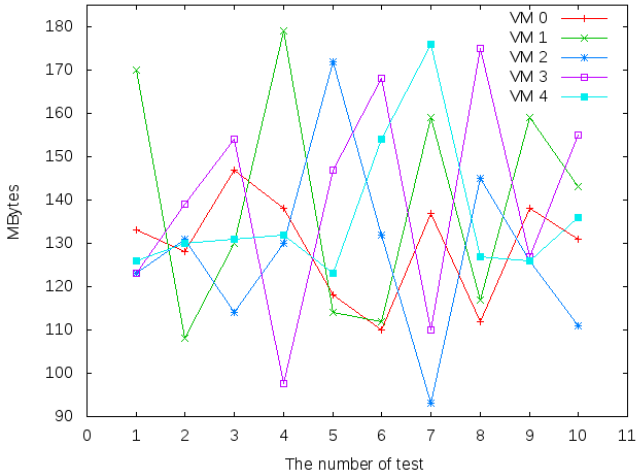
(그림 2) PVM 수에 따른 다운로드 벤치마크 비교



(a) PVM 이 2 개일 때 업로드 벤치마크



(b) PVM 이 3 개일 때 업로드 벤치마크



(c) PVM 이 5 개일 때 업로드 벤치마크

(그림 3) PVM 수에 따른 업로드 벤치마크 비교

(그림 2)와 (그림 3)를 보면, 공통적으로 네트워크 자원을 균등하게 배분받지 못함을 알 수 있다. 이는 가상머신이 어느 물리 CPU 에 할당되는지에 따라 I/O 성능이 달라지기 때문이다. 우선 앞서 언급한 내용을 다시 한번 짚을 필요가 있다. 2.1 에서 언급했듯이 크레딧-스케줄러는 작업량을 늘리기 위해 부스트 우선순위를 도입하여 I/O 작업을 먼저 처리하도록 했다. 그러나 여러 도메인들이 부스트 상태라면 무엇이 우선 처리되어야 하는지에 같은 세밀한 정의는 아직 없다. 그러므로 Xen 은 자원 배분을 적게 받은 가상머신을 찾아 자원을 나눠주는 대신, 물리 CPU 의 런큐의 제일 앞 가상머신의 일을 수행한다.

따라서 다음과 같은 상황을 예시로 위 결과를 설명할 수 있다. 1 번 물리 CPU 에 I/O 작업적인 가상머신이 많이 할당되어 있고, 2 번 물리 CPU 에 CPU 작업적인 가상머신이 많이 할당되어 있다고 하자. 동시에 1 번, 2 번 물리 CPU 에 I/O 작업적인 가상머신이 추가되었을 때, 1 번 물리 CPU 에 배정된 가상머신은 이미 존재하는 다른 부스트 상태인 가상머신들과 경쟁해야 하지만, 2 번 물리 CPU 에 배정된 가상머신은 즉시 CPU 를 점유할 수 있다.

4. 결론

Xen 에서는 CPU 와 I/O 처리량을 동시에 높이기 위하여 부스트라는 우선순위가 있는 크레딧-스케줄러를 도입하였다. 그러나 실험 데이터가 보여주듯, I/O 특히, 네트워크 자원은 공평하게 배분되지 않음을 확인할 수 있었다. 스케줄러의 주요 목적 중 하나가 자원을 공평하게 분배함으로써 기아현상을 방지하는 것인 만큼 이 문제는 반드시 해결되어야 한다.

도메인-0 도 다른 도메인들과 동일하게 스케줄링 된다는 점도 주목해야 한다. 만일 도메인-0 가 스케줄링에서 밀리는 상황이 일어난다면, 전체 처리량도 감소하게 된다. 이러한 상황이 온다면 처리량을 끌어올리려 한 처음 의도조차 달성하지 못한다.

현재 부스트 우선순위를 도입한 이유가 타임 슬라이스가 크기 때문이다. 타임 슬라이스를 유동적으로 조절하여 I/O 인터럽트를 받아야 하는 작업이 전체 작업의 일정 수 이상이 되면 타임 슬라이스를 줄이는 것도 한가지 해결책이 될 수 있다. 다만 이렇게 타임 슬라이스가 줄어들 때는 CPU 의 처리량이 다소 감소한다. 결국 처리량과 공평성 사이에서 선택을 해야 할 것으로 판단된다.

참고문헌

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, “Xen and the Art of Virtualization”, SIGOPS Operating syst. Rev., Vol. 37 No. 5, pp 164-177, 2003.

[2] E. Asyabi, S. SanaeeKohroudi, M. Sharifi and A. Bestavros, “Mitigating Tail Latency of Cloud Virtual Machines”, IEEE Transactions on Parallel and Dist. Syst., Vol. 29, No. 10, pp 2346~2359, 2018.

[3] L. Zeng, Y. Wang, D. Feng and K. B. Kent, “A Novel Improvement of Network Virtualizations in Xen for I/O-Latency Sensitive Applications on Multicores”, IEEE Transactions on Network and Service Management, Vol. 12, No. 2, pp 163~175, 2015.