가상머신 스케줄러의 I/O 성능 향상을 위한 대출/상환 기법

(Loan/Redemption Scheme for I/O performance improvement of Virtual Machine Scheduler)

김기수*, 장준혁**, 홍지만***

(Kisu Kim, Joonhyouk Jang, Jiman Hong)

요 약

가상화 기술에 의해 추상화된 자원은 하드웨어적으로 효율적으로 사용 할 수 있어 관리가 용이하며, 이로 인해 클라우드 시스템과 대형 서버 클러스터 구축 등에 가상 머신 모니터가 널리 사용되고 있다. 가상화된 시스템의 성능은 가상머신 스케줄러의 영향을 크게 받는다. 하지만, 기존의 가상 머신에서 사용하는 Credit 스케줄러는 스케줄링 지연 시간이 길어질 경우, I/O 응답성이 저하되는 문제점이 있다. 본 논문에서는 가상머신의 이벤트 응답성 저하 현상을 개선하기 위해 기존 가상머신의 Credit 스케줄러에 대출/상환 기법을 도입하였다. 제안 기법은 가상 머신에 I/O 이벤트 처리를 위한 credit을 대출해주고, 대출 credit의 소비 패턴을 분석하여 각 가상머신의 태스크 특징을 분류한다. I/O 이벤트가 도착했을 때, 분석된 태스크 특징을 기반으로 일시적으로 가상 머신의 스케줄링 우선순위를 높임으로써 시스템의 I/O 성능을 향상시킨다. 제안 기법을 가상머신 모니터에 구현하였으며, 기존 가상머신의 Credit 스케줄러 대비 제안된 기법을 적용한 가상머신의 I/O 평균 응답성과 대역폭이 각 60%, 62% 향상되었다.

■ 중심어 : 가상머신 모니터 ; Xen ; I/O 응답성 ; 스케줄링

Abstract

Virtualized hardware resources provides efficiency in use and easy of management. Based on the benefits, virtualization techniques are used to build large server clusters and cloud systems. The performance of a virtualized system is significantly affected by the virtual machine scheduler. However, the existing virtual machine scheduler have a problem in that the I/O response is reduced in accordance with the scheduling delay becomes longer. In this paper, we introduce the Loan/Redemption mechanism of a virtual machine scheduler in order to improve the responsiveness to I/O events. The proposed scheme gives additional credits for to virtual machines and classifies the task characteristics of each virtual machine by analyzing the credit consumption pattern. When an I/O event arrives, the scheduling priority of a virtual machine is temporally increased based on the analysis. The evaluation based on the implementation shows that the proposed scheme improves the I/O response 60% and bandwidth of virtual machines 62% compared to those of the existing virtual machine scheduler.

■ keywords : Virtual Machine Monitor ; Xen ; I/O performance ; Scheduling

Ⅰ. 서 론

가상화는 다수의 물리적 자원을 논리적으로 통합하거나, 단일 물리적 자원을 논리적으로 분할하여 자원을 유연하며 효율적으로 사용할 수 있게 만든다. 가상화 기술을 도입한 단일 머신에서는 가상화 계층을 통해 다수의 독립적인 시스템을 운용할 수 있다. 이러한 가상화 기술을 가상머신 모니터(Virtual Machine Monitor, VMM) 또는 하이퍼바이저라고 부른다. 가상머신 모니터는 Hyper-V, VMware, KVM, Xen, VirtualBox 등 다양하다. 가상머신 모니터 중의 하나인 Xen에서는 추상화된 가상머신을 도메인이(Domain)라고 표현한다. 또한 메모리, I/O, CPU 등을 모두 가상화하여 가상머신에게 제공한다. 도메인 내에는 물리적 CPU를 추상화한 VCPU(Virtual Central Processing Unit)를 사용한다 [2, 3, 6]. 운영체제와 프로세스의 관계 중 VCPU는 프로세스에 해당된다. 운영체제가 CPU

* 학생회원, 숭실대학교 컴퓨터학과
** 학생회원, 숭실대학교 융합소프트웨어플랫폼 연구소
*** 정회원, 숭실대학교 컴퓨터학부
이 논문은 2013년 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2013R1A1A2058154)

접수일자 : 2016년 08월 16일 게재확정일 : 2016년 12월 14일
수정일자 : 2016년 12월 08일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

자원을 사용할 프로세스를 선택하는 반면에, 가상머신 모니터에서는 CPU 자원을 사용할 타겟으로 VCPU를 선택한다. 가상머신 모니터 내의 스케줄링 정책을 통해 VCPU를 선택하며, 게스트 운영체제의 경우 자신의 본연의 스케줄러에 따라 실행될 프로세스를 결정한다.

Xen은 가상머신 스케줄링 위해 Credit 스케줄러를 사용한다. Credit 스케줄러는 VCPU에게 CPU 자원을 의미하는 credit 인자를 할당한다. VCPU는 자신이 CPU를 점유한 시간만큼 보유하고 있는 credit을 소비한다. 도메인 내에 VCPU가 실행 가능한 태스크가 없거나 credit을 소모하지 않아 특정 값 이상의 credit을 보유할 경우 해당 VCPU는 이벤트가 발생할 때까지 블록 상태가 된다. 이 상태를 가진 VCPU를 비활성화 VCPU라고 한다. 블록 되지 않은 활성화 VCPU의 I/O 이벤트 처리는 스케줄링 될 때 처리된다.

스케줄링 대상에서 제외된 비활성화 VCPU 의 I/O 이벤트 처리에 대한 응답성을 높이기 위해서 Credit 스케줄러는 boost 매커니즘을 사용한다. boost 매커니즘이란, 블록 된 VCPU에 I/O 이벤트가 지연되었을 경우, 해당 VCPU에 가장 높은 BOOST 우선순위를 주어 다른 VCPU를 선점하여 실행 가능하게 만들어준다. 하지만, CPU 집중적인 작업을 하는 VCPU는 credit을 지속적으로 소모하고, 도메인 내에 실행가능 태스크가 존재할 확률이 높기 때문에 기존 boost 매커니즘의 혜택을 받을 수 없다 [1]. 그리고 활성상태 VCPU가 다수 존재할 경우, 가상머신 스케줄링 지연시간이 증가하기 때문에 이벤트 응답성은 더욱 떨어진다. 가성머신의 I/O 과정에서 발생하는 인터럽트는 이벤트 매커니즘을 이용해 처리하기 때문에 이벤트 응답성이 떨어질 경우 전체적인 I/O 성능이 떨어진다.

본 논문에서는 CPU 집중적인 작업을 하는 도메인의 I/O 성능 저하 문제를 해결하기 위해 Credit 스케줄러에 대출/상환 기법을 도입하였다. 제안 기법은 각 도메인에 boost매커니즘를 위한 credit을 대출해 주고, 이 credit의 소비 패턴을 분석하여 각 도메인의 태스크 특징을 파악한다. 또한 각 도메인의 태스크 특징에 따라 boost 매커니즘을 차등 적용함으로써 특정 VCPU가 CPU 자원을 독점하는 것을 예방한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 제안 기법을 설명한다. 4장에서는 구현을 통한 실험 결과를 보이고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

1. 가상화 기술

가상화 계층에 따라 전가상화, 반가상화로 구분할 수 있다 [3, 6]. 그림 1(a)와 같이, 하드웨어를 모두 가상화한 것을 전가상화라고 한다. 단일 머신에서 사용되는 하드웨어들(CPU, Disk, Memory, Network, etc)을 가상화하였기 때문에 다양한 운영체제를 수정 할 필요 없이 게스트 운영체제로 설치 할 수 있다. 하지만, 이를 위해서는 CPU의 하드웨어 지원이 필요하다. CPU 제조사 별로 가상화를 위한 하드웨어 지원을 하고 있다. 인텔 사의 VT(Virtualization Technology), AMD 사의 SVM(Secure Virtual Machine)이 있다 [7]. 반면 그림1(b)와 같이 하드웨어 지원을 필요로 하지 않는 가상화 계층 모델을 반가상화라고 한다. 하지만, 전가상화는 다르게 반가상화는 게스트 운영체제는 소스 코드를 수정해서 가상화를 구현한다. 반가상화는 CPU의 하드웨어 지원이 필요 없는 장점이 있지만 게스트 운영체제를 가상머신 모니터에 맞게 수정해야하는 단점이 있다.

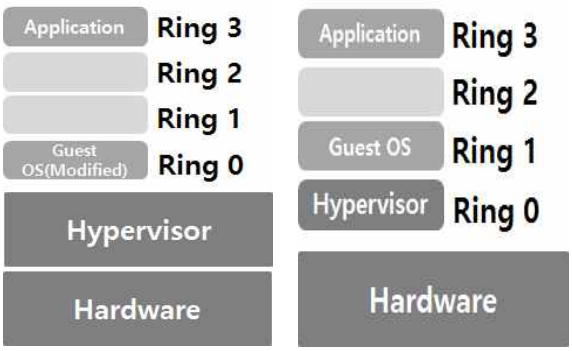

그림 (a) 반가상화 모델 그림 (b) 전가상화 모델

그림 1. 가상화 모델

2. IDD 모델 I/O 처리과정

Xen에서는 그림2와 같이 I/O 가상화를 위해 IDD(Isolated Driver Domain) 모델을 사용한다. IDD 모델은 특정 도메인에 특권을 주어 실제 디바이스 드라이버를 관리하게 한다. 기본적으로 도메인0이 IDD로 설정되어있다. 도메인U라고 불리는 특권 도메인(IDD 이외 도메인)은 실제 디바이스 드라이버 대신 프론트-엔드 드라이버(front-end driver)라고 불리는 가상 디바이스 드라이버를 갖게 된다. 프론트-엔드 드라이버는 도메인U의 I/O 요청을 도메인0이 갖고 있는 백-엔드 드라이버(back-end driver)에 전달한다. 백-엔드 드라이버는 전달받은 I/O 요청을 IDD의 실제 디바이스 드라이버에 전달하여 실제 디바이스에 I/O를 요청하여 처리한다. 과정은 아래와 같다 [4, 5].

도메인U의 어플리케이션이 I/O를 발생시키면 프론트-엔드 드라이버는 I/O 요청을 I/O 링에 기입한다. I/O 링은 도메인 간의

통신에 사용되는 공유 메모리이다. 도메인 U에서 요청하는 I/O 들을 I/O 링에 적재 후, 이벤트 채널을 통해 IDD에게 I/O 요청을 발생을 알린다. IDD의 백-엔드 드라이버는 이벤트 채널을 통해 전달받은 요청에 해당하는 I/O를 I/O 링에서 읽어 실제 디바이스 드라이버로 보낸다. 실제 디바이스 드라이버에서 요청을 처리가 완료 된 후, IDD의 백-엔드 드라이버는 완료된 I/O를 I/O 링에 적재 한 후, 도메인U의 프론트-엔드 드라이버에 이벤트를 발생시킨다. 이벤트를 수신한 도메인U의 VCPU가 스케줄링 되면 VCPU는 이벤트를 감지하고 프론트-엔드 드라이버에 등록된 이벤트 핸들러가 동작하여 I/O를 처리한다. 프론트-엔드 드라이버가 이벤트를 수신 받고 이벤트 핸들러가 호출되는 동작은 하드웨어 인터럽트와 디바이스 드라이버의 인터럽트 핸들러 처리 방식과 동일하다. 이와 같이 Xen에서는 I/O 이벤트를 가상의 인터럽트로 처리한다. I/O 이벤트는 인터럽트 달리 즉각적으로 반응하지 않는다. I/O 이벤트를 수신한 VCPU가 스케줄링 돼야만 I/O 이벤트 발생을 감지하고 처리한다 [11][12].

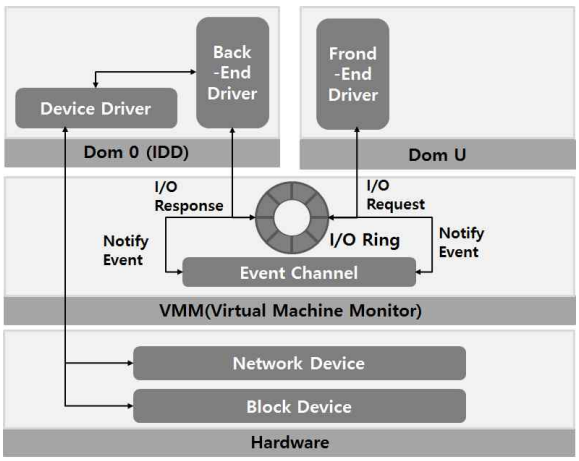

그림 2. 반가상화 I/O 처리과정

3. Credit 스케줄러

Xen 4.5 기준으로 Xen Project VMM에서 지원하는 스케줄러는 Credit, Credit2, RTDS, ARINC 653 이며, 사용분야는 아래 표 1에서 확인 할 수 있다 [11][12]. 이 중 현재 Credit 스케줄러가 기본 스케줄러로 널리 사용되고 있다. 비례공정 지분 스케줄러인 Credit 스케줄러는 도메인이 갖는 지분에 비례해서 CPU 자원을 분배한다. weight, cap, credit 은 Credit 스케줄러에서 사용되는 주요 인자들이다. weight 인자는 도메인이 할당 받은 CPU 자원의 지분을 나타낸다. cap 인자는 도메인이 사용할 수 있는 최대 CPU 사용량을 나타낸다. credit 인자는 CPU 자원을 사용할 수 있는 가용시간을 나타낸다. 스케줄링되어 CPU를 차지한 시간만큼 차감되며 30ms 마다 도메인이 갖는 weight 값을 비례한 credit을 추가로 분배받는다.

우선순위로는 BOOST, UNDER, OVER 가 있다. VCPU에 credit이 남아 있는 경우 UNDER 우선순위, credit이 음수일 때 OVER 우선순위가 부여되며 이는 VCPU가 가진 지분 이상 실행됐음을 나타낸다. BOOST 우선순위는 비활성화된 VCPU에 I/O 이벤트가 지연 된 경우에 부여된다. 물리 CPU가 가지고 있는 런큐에 BOOST > UNDER > OVER 순으로 VCPU가 정렬되어 있다. Credit 스케줄러는 다음 스케줄링 될 VCPU를 런큐의 맨앞에서 라운드-로빈(round-robin)방식으로 선택한다 [11][12].

Credit 스케줄러는 도메인을 비활성, 활성 상태를 분리하여 관리한다 [8]. 유휴한 VCPU는 비활성 상태로 만들어 런큐에서 제외시킨다. credit을 소모하지 않아 300 credit 이상을 보유하게 되거나 실행가능한 태스크가 없는 경우 VCPU는 비활성화 상태로 전환된다. CPU 집중적인 작업을 하는 도메인은 credit을 지속적으로 소모하기 때문에 VCPU는 활성상태가 유지된다. 하지만, CPU 작업이 적은 도메인의 VCPU는 credit을 소모하지 않아 credit을 축적하게 되어 비활성 상태가 될 가능성이 크다. 비활성 도메인은 I/O 이벤트를 받아 깨어날 때까지 스케줄링 되지 않는다. Credit 스케줄러는 비활성화 도메인에 이벤트가 지연 됐을 때 BOOST 우선순위를 실행하여 응답성을 향상시킨다.

표 1. Xen 가상머신 스케줄러 종류

스케줄러	사용분야
Credit	General Purpose
Credit2	General Purpose Optimized for Lower Latency
RTDS	Soft & Firm Real-time Multi-core Embedded, Automotive Low Latency Workloads
ARINC 653	Hard Real-time Single-core Avionics, Drones, Medical

기존 Credit 스케줄러의 문제점을 해결하기 위해 Zeng [10]은 새로운 I/O 처리를 위한 tasklet 와 로드 밸런싱을 변경하였다. Li[11]는 I/O 응답을 예측 및 CPU 자원 할당을 예측하는 PRGS(Predictable Resource Guarantee Scheduler)를 연구하였다.

III. 대출/상환 가상머신 스케줄링 구현

1. 태스크 특징 분류

I/O, CPU 집중 정도에 따라 태스크의 특징을 그림 3와 같이 분류 할 수 있다. 태스크가 CPU 집중적일수록 CPU 사용 시간이 길어진다. 태스트가 CPU 집중적일수록 이벤트 지연에 따라 boost 매커니즘 적용의 횟수를 적게 해줘야 CPU 자원의 독점을 피할 수 있다.

태스크가 I/O 집중적이라는 것은 I/O 이벤트가 자주 발생한다는 의미이다. I/O 집중적인 태스크가 집중적으로 I/O 이벤트가 발생할 경우 자주 boost 되어 CPU 자원을 독점하는 문제점이 발생한다. 그렇기 때문에 태스크가 I/O 집중적일수록 I/O 이벤트 지연에 따른 boost 매커니즘 적용 빈도수를 조절해줘야 한다.

그림 3은 태스크 특징 분류에 따른 그래프이다. B에 해당하는 I/O, CPU 집중적인 복합 태스크를 수행하는 도메인에 boost 매커니즘을 적용하면 해당 도메인은 CPU를 독점적으로 선점하는 문제가 발생한다. 따라서 태스크 특징에 따라 복합 태스크일수록 boost 매커니즘을 제한해야할 필요성이 있다. 따라 boost 빈도 수는 태스크 특징에 따라 C > D = B > A 와 같아야하며 도메인에 비활성화 VCPU가 많을수록 boost 되는 빈도수는 높아야한다.

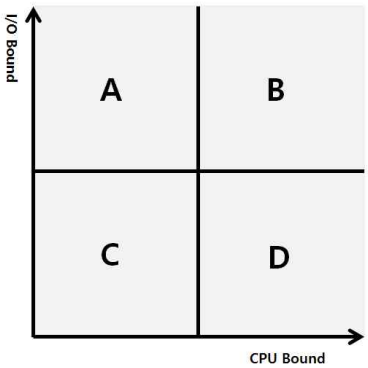

그림 3. 태스크 특징 분류

2. 대출/상환 매커니즘

본 절에서는 대출/상환 매커니즘을 통해 태스크 특징을 추정할 수 있는 방법을 기술한다. 대출/상환 매커니즘은 그림 4와 같이 도메인에 I/O 이벤트가 처음 지연됐을 경우를 기점으로 추가적인 credit을 대출해준 후, boost 매커니즘을 적용하여 I/O 응답성능을 향상시킨다. boost를 통해 CPU 점유 시간만큼 대출 credit에서 차감한다. 대출 credit을 전부 소모했을 경우, 대출받은 credit을 전부 상환하기 전까지는 boost 매커니즘에 영향을 받지 않는다. 도메인은 자신의 지분에 따라 분배받는 credit의 일정부분을 대출 credit 상환에 사용한다.

아래와 같은 과정을 통해 credit 대출과 상환 패턴을 분석하여 태스크 특징을 평가한다. 도메인이 대출 credit 소비속도가 빠를수록 태스트가 CPU, I/O 집중적인 것으로 평가하고, 상환속도가 빠를수록 유휴하다고 평가한다.

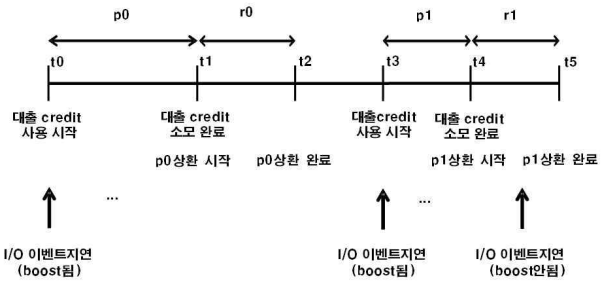

그림 4. 대출/상환 매커니즘

가. I/O 집중 태스크 평가 방법

i는 VCPU 번호를 의미하며, j는 boost 되어 실행 후 대출 credit을 모두 소비할 때 까지 걸리는 주기를 뜻한다. $VCPU_i$가 처음 boost 될 시(그림 4의 t0), $CREDIT\_LINE$(1000 credit) 상수 만큼에 해당하는 대출 credit을 빌려준다. j 주기(그림 4의 p0)동안 $VCPU_i$가 대출 credit을 전부 사용할 때 까지 걸린 시간 $p_{i,j}$와 boost된 횟수를 구한다. 수식 1을 통해 태스크의 CPU 집중 정도 $cb_{i,j}$을 구한다. $cb_{i,j}$는 $p_{i,j}$동안 $VCPU_i$가 boost되어 실행되는 동안 소모한 credit 양을 나타낸다. 값이 클수록 boost 동안 많은 대출 credit을 소모한 것을 뜻하며, I/O 처리과정 중 태스크가 CPU 집중적이라는 것을 나타낸다.

$$cb_{i,j} = \frac{CREDIT\_LINE}{n_{i,j}} credit/boost \quad (1)$$

나. CPU 집중 태스트 평가 방법

I/O 집중 태스크 평가 방법과 마찬가지로 해당 $VCPU_i$가 처음 boost가 될 시(그림 4의 t0), $CREDIT\_LINE$(1000 credit) 상수 만큼에 해당하는 대출 credit을 빌려준다. j 주기(그림 4의 p0)동안 $VCPU_i$가 대출 credit을 전부 사용할 때 까지 걸린 시간 $p_{i,j}$와 boost된 횟수를 구한다. 수식 2을 통해 태스크의 I/O 집중 정도 $bm_{i,j}$을 구한다. 값이 클수록 I/O 이벤트 발생 횟수가 많다는 것을 뜻하며, I/O 집중적인 태스크로 평가된다.

$$bm_{i,j} = \frac{n_{i,j}}{p_{i,j}} \quad boost/ms \tag{2}$$

다. 태스트 특징 종합 평가 방법

수식 3을 통해 태스크의 종합적인 특징을 평가한다. $cb_{i,j}$은 boost될 때마다 소모한 credit을 뜻하므로 CPU 집중 정도를 나타내며, $bm_{i,j}$은 ms당 boost 횟수를 뜻하므로 I/O 집중 정도를 나타내는 평가지표로 사용 할 수 있다.

$$cm_{i,j} = cb_{i,j} * bm_{i,j} = \frac{CREDIT\_LINE}{n_{i,j}} * \frac{n_{i,j}}{p_{i,j}} \quad credit/ms \tag{3}$$

$VCPU_i$가 ms당 소모한 대출 credit의 양을 나타내는 $cm_{i,j}$이 클수록 태스크 특성이 그림 3의 B에 가까우며, 작을수록 C에 가깝다. 대출/상환 매커니즘을 통한 수식 1, 2, 3의 지표를 통해, 태스크의 CPU, I/O 집중 정도를 종합적으로 평가한다.

3. 차등적 boost 매커니즘 적용 방법

수식 3을 통해 평가한 태스크 특징 $cm_{i,j}$ 지표와 상환과정을 통해 차등적으로 boost 매커니즘을 적용하는 방법을 설명한다. 태스크 특징에 따라 상환 시기 및 boost 매커니즘 적용의 빈도를 조절한다.

태스크 특징을 나타내는 $cm_{i,j}$ 값이 클수록 태스크가 그림 3의 B에 가깝다. B와 같이 I/O, CPU 집중적인 복합 태스크를 갖는 도메인의 VCPU가 boost될 경우, CPU자원을 장시간 독점 할 수 있다. 따라서 B에 가까운 경우 다른 VCPU에 비해 적은 boost 빈도수를 가져야한다. 그림 4의 t2와 같이 상환이 완료 된 후 초기 설정했던 대출 credit을 재 설정해주어 boost 빈도수를 조절해준다. 이전 태스크 특징이 그림 3의 C에 가까울수록 현재 대출 받을 수 있는 credit 양이 늘어나며, B에 가까울수록 대출 받을 수 있는 credit 양이 줄어든다. 다음 주기의 대출 credit은 수식 4를 통해 계산된다. 수식 4를 통해 계산된 다음 대출 credit인 $credit\_boost_{current+1}$ 지표를 통해 태스크 특징에 따라 차별적으로 boost 되도록 하였다.

$$credit\_boost_{current+1} = \frac{CREDIT\_LINE}{cm_{i,j}} \tag{4}$$

상환은 대출 받은 credit을 전부 소모한 후(그림 4의 r0, r1)부터 이루어지며 해당 주기에 대출 된 credit을 모두 상환하기 전까지는 boost 매커니즘에 영향을 받지 않는다. 태스크가 CPU, I/O 집중적일수록 도메인이 스케줄링 되고 남은 credit양이 적기 때문에 상환까지 걸리는 시간이 길어지며, 다른 VCPU가 스케줄링 되어 boost될 가능성이 높아진다.

제안하는 대출/상환 매커니즘을 통한 태스크 특징 평가는 유동적으로 boost 매커니즘을 적용할 수 있게 한다. 과거의 태스크 특성에 따라 현재 대출 credit의 양이 정해진다. 상환시기가 스케줄링 이 후 인 이유는, 현재 태스크의 특징을 boost 매커니즘에 반영하기 때문이다. 과거에 태스크 특징이 B에 가깝더라도 현재 태스크 특징이 C에 가깝다면 상환을 빠르게 진행하여 새로운 대출을 받아 boost 된다.

IV. 실험 및 평가

본 장에서는 다양한 환경을 기반으로 제안하는 대출/상환 매커니즘을 적용한 가상머신 스케줄링 기법을 수행하며 결과를 보였다. 제안하는 기법에 따른 I/O 응답성 향상을 보여주기 위해, 다수의 가상머신이 CPU 집중적인 태스크를 가지고 있는 환경에서 진행하였다. 아래 환경에서는 스케줄링 지연 시간이 길어지기 때문에, 활성화 VCPU의 경우 이벤트 처리 지연 시간이 길어지는 문제가 발생한다.

본 논문에서 성능의 평가를 위해 구성한 실험 환경은 아래와 같다. 서버와 클라이언트는 별도의 하드웨어로 클라이언트가 성능측에 영향을 주지 않도록 하였다.

표 2. 실험 환경

구분		설명
서버	CPU	Intel Celeron 847 (Dual core 1.1GHz)
	메모리	DDR3 8GB
	하드 디스크	SSD 256GB
	Xen 버전	Xen 4.5 (paravirtualization)
	Dom 0 커널 버전	3.16
	Dom U 커널 버전	3.13
클라이언트	CPU	AMD G-T48E 1.4GHz
	메모리	DDR3 2GB
	운영체제	CentOS 5.7
네트워크	100Mbps Ethernet	

1. 실험 방법

제안 기법과 기존 Credit 스케줄러 기법의 I/O 성능 비교 실험 및 제안 기법의 차등적 boost 매커니즘 적용여부를 측정하였다. 측정에 사용된 측정 유틸리티는 표 3과 같다.

표 3. 성능 측정 유틸리티

구분	프로그램	비고
1	iperf	클라이언트와 서버간의 네트워크 대역폭을 체크하는 프로그램
2	xentop	도메인의 CPU 사용량을 확인할 수 있는 프로그램
3	hping	클라이언트와 서버간의 TCP/UDP 응답성을 측정하는 프로그램
4	stress	CPU, I/O 워크로드 생성 프로그램

가. 제안 기법과 기존 기법의 I/O 성능비교 실험

표 4와 같이 가상머신 수를 늘려 스케줄링 지연 시간을 증가시켰으며, stress 유틸리티를 통해 CPU 집중적인 태스크를 생성해 도메인U를 활성상태로 유지시켰다. 위와 같은 환경에서 본 논문에서 제안하는 대출/상환 기법이 적용된 Credit 스케줄러와 기존 Credit 스케줄러의 성능을 측정하였다. iperf 유틸리티를 사용하여 네트워크 대역폭, hping 유틸리티를 사용하여 네트워크 응답성, xentop 유틸리티를 사용해 CPU 점유율을 측정하였다.

표 4. I/O 성능 비교 환경 설정

도메인 ID	테스크 특징	VCPU 수	Weight	Cap	stress 인자
0 (IDD)	idle	1	128	0	0
1 (성능 측정 도메인)	CPU 집중적	1	128	0	stress -c 1
2 ~12	CPU 집중적	1	128	0	stress -c 1

나. 제안 기법의 차등적 boost 매커니즘 적용여부

본 논문에서 제안하는 기법에 따라 도메인에 차별적으로 boost 매커니즘이 적용되는 지에 대한 여부를 실험을 통해 확인한다. 각 태스크의 특징에 따라 boost 되어 실행 될 수 있는 대출 credit 양을 주기적으로 변경시켜 boost 빈도를 조절한다. 실험 환경은 표 5와 같으며 도메인의 cm, credit_boost변수 값을 측정하였다.

표 5. 차등적 boost 매커니즘 적용여부 환경 설정

실험	태스크 특징	VCPU 수	Weight	Cap	stress 인자	도메인 수
1	CPU 집중 ↓ I/O 집중 ↑	1	128	0	stress -i 1	4
2	CPU 집중 ↑ I/O 집중 ↑	1	128	0	stress -i 1 -c 1	4

2. 실험 결과

초기 대출 credit 변수인 *CREDIT_LINE* 값은 1000, 대출 credit 상환비율은 20%로 고정하여 실험을 진행하였다. 상환비율은 30ms 마다 추가로 받는 credit 중 사용한 대출 credit을 갚는데 사용하는 비율을 나타낸다.

가. I/O 성능비교 실험 결과

표 6, 그림 5은 기존 Credit 스케줄러와 제안 된 기법이 적용된 Credit 스케줄러의 네트워크 응답성 측정결과를 보여준다. 네트워크 응답성 측정 유틸리티는 hping을 사용하였다. 측정결과 기존 스케줄러의 평균 응답시간은 37.2m이고 제안 된 스케줄러의 평균 응답시간은 23ms로 측정되었다. 이를 통해, 기존기법 대비 60%의 성능이 향상된 것을 확인 할 수 있다. 그림 5은 기존 Credit 스케줄러와 제안 된 기법이 적용된 Credit 스케줄러를 각각 적용 했을 때 측정된 응답시간을 누적 분포 함수 그래프로 나타낸 결과이다. x축은 응답성 시간을 나타낸다. 실험 결과, 응답시간이 10ms 이하로 측정된 실험 데이터가 기존 스케줄러에선 35%, 제안된 스케줄러에선 58%가 측정되었고 20m 이하로 측정된 데이터는 기존 스케줄러에선 41%, 제안된 스케줄러에서는 70%가 존재하는 것이 측정되었다. 이를 통해 제안된 스케줄러는 스케줄링 지연시간이 줄어 네트워크 응답성이 향상되는 것을 확인하였다.

표 6. TCP/IP 응답성 실험 결과

실험	평균 응답성
기존 기법	37.2ms
제안 기법	23ms

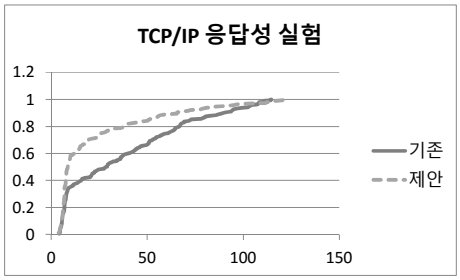

그림 5. 응답성 실험 결과

그림 6은 각 스케줄러를 적용했을 때 가상머신의 네트워크 대역폭을 나타내는 그래프이다. 네트워크 대역폭 측정 유틸리티는 iperf를 사용하였다. 기존 스케줄러 경우 평균 대역폭을 30.9Mbtis/sec, 제안된 스케줄러는 34.1Mbits/sec로 측정되었다. 기존 대비 62%의 성능 향상을 확인 할 수 있다. 실험결과를 통해 스케줄링 지연의 감소는 가상머신의 응답성뿐만 아니라 대역폭에도 영향을 주는 것을 확인했다.

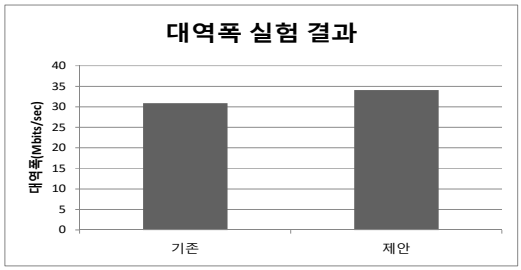

그림 6. 대역폭 실험 결과

그림 7, 8, 9은 기존 Credit 스케줄러, 제안 된 기법이 적용된 Credit 스케줄러, 무조건 boost 기법이 적용된 스케줄러를 xentop 유틸리티를 이용해 가상머신의 CPU 점유율을 측정한 그래프이다. 해당 실험에서 기존 Credit 스케줄러의 대역폭은 30.9Mbits/sec, 제안 된 기법이 적용된 Credit 스케줄러는 37.1Mbits/sec, 무조건 boost 기법의 경우 86.2Mbits/sec으로 측정되었다. 실험 결과 무조건 boost 기법을 적용 할 경우 CPU 점유율에 공정성을 해치지만, 제안된 기법을 통해 기존 Credit 스케줄러와 비슷한 CPU 점유율을 통해 공정성을 유지할 수 있다.

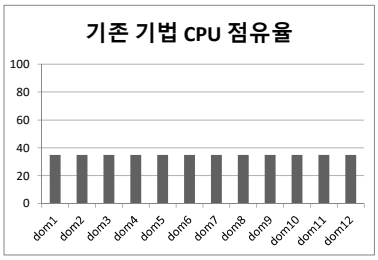

그림 7. 기존 기법 CPU 점유율 실험 결과

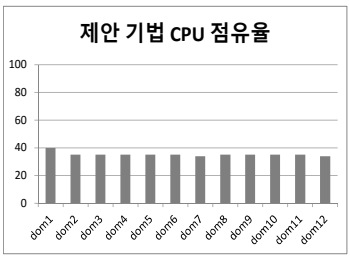

그림 8. 제안 기법 CPU 점유율 실험 결과

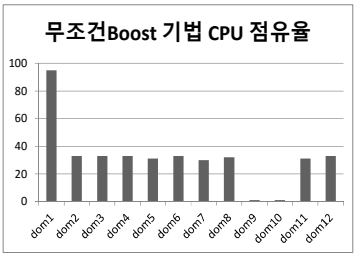

그림 9. 무조건 Boost 기법 CPU 점유율 실험 결과

나. boost 매커니즘 적용여부 실험 결과

표 7은 표5 와 같은 환경에서 대출/상환 기법이 적용된 Credit 스케줄러에 태스크 특징에 따라 도메인이 갖는 cm, credit_boost 값을 나타낸다. 실험결과에 따라, 같은 I/O 작업을 가지고 있는 도메인이라도 CPU 작업에 대한 가중이 다를 경우 credit_boost와 cm 값이 다른 것을 확인 할 수 있다. CPU 작업에 대한 집중도가 낮을수록 ms 당 소모되는 credit을 나타내는 cm 값은 작아지며, CPU 작업에 대한 집중도가 높을수록 cm 값은 커진다. cm과 credit_boost 값은 반비례 관계이므로, cm 값이 커질수록 credit_boost 값은 작아진다. 실험결과를 통해 제안하는 기법은 대출/상환 기법에 따라 평가된 태스크 특징에 따라 차별적으로 boost 매커니즘이 적용되는 것을 확인하였다.

표 7. I/O 성능 비교 환경 설정

실험	태스크 특징	credit_boost	cm	stress 인자
1	낮은 CPU 집중 높은 I/O 집중	600	5	stress -i 1 -t 60
2	높은 CPU 집중 낮은 I/O 집중	375	9.2	stress -c 1 -i 1 -t 60

V. 결 론

본 논문은 Xen Credit 스케줄러에서 발생하는 I/O 문제점을 파악하여 이를 해결하기 위한 해결방법을 제안하였다. 기존 Credit 스케줄러는 I/O 응답성 향상을 위해 boost 매커니즘을 사용한다. boost 매커니즘은 비활성화된 도메인의 I/O 이벤트가 지연될 경우, 일시적으로 비활성화 도메인 VCPU 우선순위를 높여주어 해당 도메인이 우선적으로 CPU를 점유 할 수 있게 도와준다. 기존 boost 매커니즘은 비활성 도메인에게만 적용되기 문제점이 있다. 이는 CPU 지분 공정성과 boost 매커니즘의 관계를 고려했기 때문이다. 비례 지분 공정 스케줄러인 Credit 스케줄러 최우선 중점은 CPU 지분 공정성이다. 기존 Crdit 스케줄러의 boost 매커니즘 동작 방식은 CPU 지분 공정성은 지킬 수 있지만, 활성 상태 도메인들이 많아져 가상머신 스케줄링 지연 시간이 증가 할 경우 I/O 응답성에 대한 성능이 낮아지는 문제점이 있다.

본 논문에서는 위와 같은 문제점을 해결하기 위해 대출/상환 기법을 도입하여 활성상태 도메인의 태스트 특징을 파악하여 차등적으로 boost 매커니즘을 적용하는 기법을 제안하였다. 대출/상환 매커니즘을 통해 패턴을 파악하여 각 도메인의 태스크 특징을 평가하여 boost 매커니즘의 적용 빈도를 조절한다.

제안기법의 성능을 평가하기 위해 다양환 환경 내에서 제안된 기법과 기존 Credit 스케줄러 기법을 적용한 시스템의 I/O 성능 및 가상머신의 CPU 점유율을 측정하였다. 실험결과, 제안기법은 기존 Credit 스케줄러 기법보다 I/O 성능을 향상 시키며, CPU 지분 공정성을 공평하기 유지하지는 것을 확인하였다.

그러나 본 논문에서 제안하는 기법의 주요 인자인 credit_line와 상환 비율은 고정되어 실험되었다. 두 인자는 제안하는 기법의 동작 및 성능에 많은 영향을 미칠 수 있기 때문에 효율적인 두 인자의 동적으로 적용 할 수 있는 기법 연구가 필요하다.

References

[1] 김병기, 고영웅, “Xen 가상화 환경에서 지연시간에 예민한 도메인을 지원하기 위한 스케줄링 기법”, *한국정보기술학회논문지,* 제 10권, 제11호, 2012.

[2] 박은병, 김태훈, 이상철, 문대혁, "Xen으로 배우는 가상화 기술의 이해", *한빛미디어,* 2013.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield, "Xen and the art of virtualization", *ACM SIGOPS Operating Systems Review - SOSP* 2013.

[4] Ludmila Cherkasova, Diwaker Gupta, Robert D. Gardner, "System and method for controlling aggregate CPU usage by virtual machines and driver domains over a plurality of scheduling intervals", US Patent, US20080028411, June 26, 2012.

[5] Huacai Chen, Hai Jin, Kan Hu, Minhao Yuan, "Adaptive Audio-aware Scheduling in Xen Virtual Environment", *2010 IEEE/ACS International Conference on Computer Systems and Applications(AICCSA)*, 2010.

[6] David Chisnall, *"Xen 하이퍼바이저 완벽 가이드"*, 제이펍, 김세영, 정윤선 옮김, 2012.

[7] Abel Gordon, Nadav Amit, Nadav Har'El, Muli Ben-Yehuda, Alex Landau, Assaf Schuster, Dan Tsafrir, "ELI: Bare-Metal Performance for I/O Virtualization", *The International conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[8] Ryan Hnarakis, In Perfect Xen, "A Performance Study of the Emerging Xen Scheduler", Ph. D, Dissertation, *the Faculty of California Polytechnic State University San Luis Obispo*, 2013.

[9] Xi, Sisu, et al. "Real-time multi-core virtual machine scheduling in xen." Embedded Software (EMSOFT), 2014 International Conference on. IEEE, 2014.

[10] Li, Jian, and David SL Wei. "Accurate CPU Proportional Share and Predictable I/O Responsiveness for Virtual Machine Monitor: A Case Study in Xen." (2015).

[11] The Xen Project, http://wiki.xen.org/wiki/Credit_Scheduler.

[12] http://www.xenproject.org/developers/teams/hypervisor.html.

저 자 소 개

김기수(학생회원)

2015년 숭실대학교 컴퓨터학과 학사 졸업.
2016년 숭실대학교 컴퓨터학과 석사 재학.

<주관심분야 : 시스템 소프트웨어, 운영체제>

장준혁(정회원)

2009년 서울대학교 컴퓨터공학과 학사 졸업.
2015년 서울대학교 컴퓨터공학과 학과 석사 졸업.
2015년 서울대학교 컴퓨터공학과 학과 박사 졸업.

<주관심분야 : 시스템 소프트웨어, 운영체제>

홍지만(종신회원)

1999년 서울대학교 컴퓨터학과 박사 졸업
2004년~2007년 광운대 컴퓨터학과 교수
2007년~ 숭실대학교 컴퓨터학과 교수

<주관심분야 : 시스템 소프트웨어, 운영체제>