논문 2012-07-25

짧은 지연 시간 태스크를 지원하는 타이머 기반 크레딧 스케줄러

(Timer-based Credit Scheduler for Supporting Low Latency Task)

김 병 기, 고 영 웅*
(Byung-Ki Kim, Young-Woong Ko)

Abstract : Virtualization allows multiple commodity operating systems to share on a single physical machine. Resource allocation among virtual machines is a key to determine virtual machine performance. To satisfy time-sensitive task on a domain, hypervisor needs to observe the resource requirements and allocates proper amount of CPU resources in a timely manner. In this paper, we propose a realtime credit scheduler for latency sensitive application on virtual machines. The key idea is to register a time event in the Xen hypervisor. Experiment result shows that the proposed scheme is superior to Credit scheduler.

Keywords : Xen, Credit scheduler, Realtime, Latency sensitive, Monitoring

1. 서론

최근 가상화 기술은 비용, 공간 그리고 전력 절감을 위한 목적으로 널리 사용되고 있다. 특히, 데이터 센터 등에서 서버 통합을 위한 목적으로 가상화가 사용되고 있으며 성공적인 기술로 각광받고 있다. 가상화는 최근 몇 년간 하드웨어 및 운영체제의 발전으로 성능적인 측면에서 많은 발전이 이루어졌다. 이러한 가상화 환경에서는 각 가상머신들이 필요로 하는 자원을 효율적으로 분석하고 자원을 할당하는 방법이 중요하다. 이를 위하여 가상 머신의 하드웨어 자원을 효율적으로 할당하고 스케줄링하는 부분에서 많은 연구들이 진행되는 중이다.

Xen 하이퍼바이저의 기본적인 스케줄러인 크레딧(credit) 스케줄러는 도메인들이 CPU를 공평하게 공유할 수 있으며 도메인들의 I/O 요청을 빠르게 처리할 수 있는 스케줄러로 알려져 있다 [1]. 하지만, 도메인 내부에서 짧은 지연 시간 내에 처리가 되어야 하는 작업이나 실시간적인 작업을 처리하는 경우에는 효율적이지 못한 문제를 가지고 있다. 왜냐하면 가상머신이 스케줄링 되어 동작하고 있는 상태가 아니면 하이퍼바이저에서 도메인 내에 존재하는 태스크의 긴급도를 파악할 수 없기 때문에 스케줄링에 따른 지연시간이 발생하기 때문이다. 이러한 문제점은 응답속도가 중요한 작업을 처리하는 가상 머신을 지원하는데 있어서 큰 단점으로 작용한다.

본 연구에서는 기존의 Xen 하이퍼바이저의 스케줄링 정책을 수정하여 짧은 지연 시간을 태스크를 지원할 수 있는 방안을 제시하고 있다. 주요 아이디어는 짧은 지연 시간이 필요한 태스크가 동작되는 도메인에서 타이머 이벤트를 하이퍼바이저에 등록하고 타이머 이벤트가 만료되는 경우에 해당 도메인의 우선 순위를 높이는 방법이다. 이를 위하여 기존의 크레딧 스케줄러에서 RT 우선 순위를 추가하여 타이머 이벤트 기반 처리가 될 수 있도록 하였다. 따라서 짧은 지연 시간을 가지고 처리해야 하는 작업이 도메인이 스케줄링이 될 때까지 기다리지 않고 빠른 시간 내에 제어를 얻을 수 있는 장점을 지니게 된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구 동향에 대해 설명하고, 3장에서는 제안하는 타이머 기반 스케줄링 기법에 대해서 기술한다. 4장에

* 교신저자(Corresponding Author)
논문접수 : 2012. 03. 30., 수정일: 2012. 04. 07.,
채택확정 : 2012. 06. 04.
김병기, 고영웅 : 한림대학교 컴퓨터공학과
※ 이 논문은 한국연구재단의 지원을 받아 수행된 기초연구사업임(2010-0005442).

ISSN : 1975-5066

서는 제안하는 방법에 대한 실험 결과를 보이며, 마지막으로 5장에서 결론과 향후연구에 대해 기술한다.

II. 관련연구

가상화 시스템에서 자원 활용도를 높이기 위해서는 효율적인 스케줄링 메커니즘이 필요하다. Xen에서는 두 단계의 스케줄링을 통해 게스트 도메인의 태스크를 수행시킨다. 가상머신 스케줄러는 virtual-CPU 단위로 도메인을 스케줄링하며, 가상머신 스케줄러에 의해 스케줄링 된 도메인은 각 도메인의 고유한 스케줄링 정책에 따라서 각 태스크들을 스케줄링한다. 최근 수년 동안 효율적인 자원 스케줄링을 위해서 다양한 연구들이 진행되었다. Govindan [2]은 네트워크 사용률이 높은 서버의 CPU 사용율과 다른 게스트 도메인들에 공평한 CPU를 할당하기 위해 네트워크 사용률이 높은 도메인의 우선순위를 높여 네트워크 응답속도를 개선하는 방법을 제안하였다. Kim [3]은 크레딧 스케줄러의 I/O 성능을 높이기 위해 Grey-Box 메커니즘을 이용해 I/O 요청이 많은 도메인을 구분하고 I/O를 주로 수행하는 도메인들을 부분적으로 BOOST 모드로 동작시켜 I/O 응답속도를 향상시켰다. SEDF-DC 스케줄러와 같은 경우는 Xenmon을 이용해 게스트 도메인의 I/O요청을 감시하고 I/O를 사용하는 도메인의 CPU 할당량을 감소시켜 다른 도메인들의 성능을 저하시키는 문제점을 해결할 수 있는 방법을 제안하였다 [4]. Chen [5]은 오디오 장치를 주로 이용하는 도메인의 요청이 들어올 때 QEMU를 이용해서 장치에 접근하려는 도메인의 CPU 할당량을 늘려 특정 목적에 맞게 설계된 시스템에서 활용이 가능한 방법을 제안하였다. 하지만, 기존의 I/O 요청의 응답속도를 높이는 방법들은 주로 I/O 요청을 확인한 후에 하이퍼바이저 스케줄러에서 이를 감지하고 도메인의 우선순위를 높이는 방법들이다. [6]에서는 멀티미디어와 같은 연성 실시간 태스크를 지원하기 위한 방법으로 다양한 파라미터를 이용하여 스케줄러의 동작을 제어할 수 있는 방법을 제안하고 이를 위한 툴로 Xentune을 제시하고 있다. [7]에서는 비주기와 주기적인 태스크들이 존재할 때 비주기 태스크의 응답시간의 중요도를 적용하여 응답성을 높일 수 있는 스케줄링 알고리즘을 제안하였다. [8]에서는 드라이버 도메인이 인터럽트를 처리를 마치더라도 스케줄링 대상에서 빠지지 않도록 하여 I/O 요청을 빠르게 처리 할 수 있도록 하였다. 즉 실시간 I/O 장치를 지원하기 위해 인터럽트 요청을 받은 드라이버 도메인은 항상 높은 우선순위를 가지고 스케줄링 되어 있도록 하여 I/O 요청을 빠르게 처리할 수 있다. 본 연구에서는 가상머신이 블록(block)된 상태에서 제때에 스케줄링 되어 깨어나지 못하는 상황에서 I/O 요청이 지연되는 상황에 대하여 문제점을 제시하고 해결할 수 있는 방법에 대한 해결책을 제시하는 것이다. 이외에도 프로파일링 기법을 통한 가상 머신 연구 [9] 또는 네트워크 처리량을 높이는 연구들이 최근에 많이 연구되고 있다 [10].

III. 시스템 설계 및 구현

Xen 가상머신은 드라이버 도메인이라고도 불리는 특별한 가상머신인 Domain-0가 하드웨어에 바로 접근할 수 있는 권한을 가지며 다른 게스트 도메인들은 Domain-0를 통해 하드웨어 자원을 사용할 수 있다. Xen 시스템에서 입출력 장치들은 모든 게스트 도메인에서 공유할 수 있지만, 장치 접근은 하이퍼바이저를 통해서만 가능하다. Xen은 입출력을 위해 Domain-0에서 동작하는 backend 드라이버에서 입출력 장치를 사용하는 스플릿 디바이스 드라이버 모델을 채택하였다. 게스트 도메인에서는 frontend 드라이버를 가지고 있어서 공유메모리를 통해 backend 드라이버와의 통신을 통해 입출력 처리를 할 수 있다. 모든 입출력 처리는 Domain-0에 의해 처리되기 때문에 입출력이 잦은 도메인은 Domain-0를 스케줄링 하는 정책이 성능에 큰 영향을 미친다.

Xen에서 기본적으로 사용되는 스케줄러는 크레딧 스케줄러로써 각 가상 머신들이 공평하게 CPU를 사용할 수 있도록 설계되었다. 하이퍼바이저 스케줄러는 게스트 도메인을 virtual-CPU(VCPU) 단위로 스케줄링하며 부팅시에 도메인마다 여러 개의 VCPU를 생성할 수 있다. VCPU들은 weight 과 credit을 할당 받아서 CPU 자원을 이용한다. 크레딧 스케줄러는 30ms 단위로 각 VCPU에 할당된 CPU 자원량(credit)을 계산하며 credit이 남아 있는 경우 UNDER(-1)라는 우선순위를 가지게 된다. 또한, 30ms 주기 동안 credit을 모두 소모한 VCPU는 OVER(-2) 우선순위를 가지게 된다. I/O 이벤트를 기다리며 블록 되었던 VCPU와 같은 경우는 I/O 이벤트가 발생하여 깨어나게 되는 순간

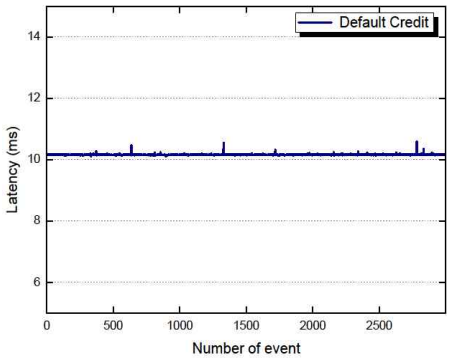

그림 1 크레딧 스케줄러에서 ping 도착 시간(외부 서버에서 가상머신)
Fig. 1. Ping arrival time in the credit scheduler(from external server to VM)

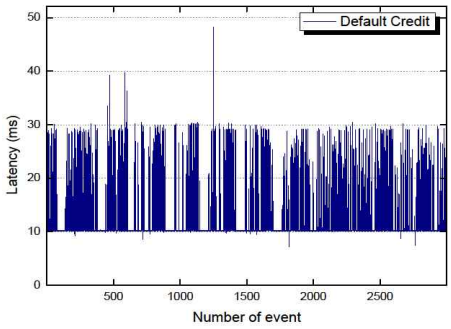

그림 2 크레딧 스케줄러에서 ping이 도착한 시간(가상머신에서 외부서버)
Fig. 2. Ping arrival time in the credit scheduler(from VM to external server)

BOOST(0)라는 우선순위를 가지며 가장 먼저 스케줄링 되어 동작이 된다. BOOST 상태로 깨어난 VCPU와 같은 경우는 짧은 시간 동안 동작되고 다시 원래의 우선순위를 가지게 된다. 이러한 BOOST 방법은 크레딧 스케줄러에서 I/O 요청이 자주 발생되는 경우 해당 도메인의 우선순위를 높여 I/O를 빠르게 처리함으로써 I/O 요청이 잦은 게스트 도메인들의 응답속도를 높여줄 수 있다. I/O에 의해 BOOST 되는 경우는 외부에서 해당 도메인에 I/O 인터럽트가 전달된 경우이다. 해당 도메인에 도착된 I/O 인터럽트가 있고, 블록된 상태에 있던 도메인은 BOOST로 가장 먼저 스케줄링 되기 때문에 크레딧 스케줄러는 I/O 요청을 빨리 처리할 수 있다. 이처럼 외부 인터럽트에 의한 가상머신의 I/O는 빠르게 처리되지만, 가상머신 내부에서 이벤트가 필요한 경우는 이를 빠르게 처리하지 못하는 것을 실험을 통해 알 수 있다.

그림1과 같은 경우는 외부 서버로부터 ping이 도착한 시간을 기록한 것이다. 이때, 세 개의 도메인들은 20%의 CPU를 점유하여 총 60%의 CPU를 점유하도록 설정하였다. 이때 10ms 마다 ping이 도착한 시간을 기록해 보면 10ms 근처에서 모든 작업이 처리되는 것을 알 수 있다. 이는 외부 인터럽트가 도착하면 해당 도메인을 BOOST로 스케줄링하여 빠르게 응답할 수 있기 때문이다. 다른 도메인들이 CPU를 사용하면서 바쁘게 동작하고 있어도 BOOST 우선순위가 가장 높기 때문에 우선적으로 스케줄링 되고 있는 것을 알 수 있다.

그림 2는 도메인이 짧은 주기로 외부에 이벤트를 발생시키는 상황으로 가상머신으로부터 ping이 도착한 시간을 외부 서버에서 기록한 것이다. 가상머신은 10ms 마다 외부서버로 ping을 보내고 있다. 가상 머신의 CPU가 idle 상태가 되면 *xen_safe_halt*가 호출되어 *sched_block* 이벤트가 발생하기 때문에 Xen 스케줄러는 다른 도메인을 스케줄링한다. 크레딧 스케줄러는 30ms 단위로 가상머신을 스케줄링 하기 때문에 외부 인터럽트에 의해 BOOST 되거나 스케줄러에서 선택될 때 까지 기다려야 한다.

따라서, 그림 2 와 같이 짧은 주기로 이벤트를 발생시키는 태스크가 있는 경우 제때에 도메인을 스케줄링하지 못하는 상황이 발생한다.

3.1 타이머 기반 스케줄링 기법

본 연구에서 제안하는 스케줄링 방법은 BOOST 보다 높은 우선순위인 RT(RealTime) 우선순위를 두어 도메인에서 RT 요청이 발생하는 경우에 빠르게 해당 도메인을 RT 상태로 변환하여 응답속도를 높일 수 있는 방법에 대하여 제안한다. 짧은 시간 동작하고 깨어나는 주기적인 작업을 처리하는 도메인들과 같은 경우 깨어날 때 BOOST 우선순위를 받기 때문에 이러한 도메인들이 많아질 경우 BOOST 우선순위를 가지는 도메인들끼리 경쟁을 할 수 있다. 제안하는 방법은 하이퍼콜(hypercall)을 이용하여 도메인이 RT 모드로 동작하기를 원할 때 하이퍼바이저로 RT 우선순위를 요구한다. 하이퍼바이저는 RT 요청을 받으면 크레딧 스케줄러에 realtime_ticker를 등록하고 스케줄링 이벤트가 발생할 때 해당 도메인의 우선순위를 높여서 먼저 스케줄링 될 수 있도록 설정한다. 해당 도메인이 주기

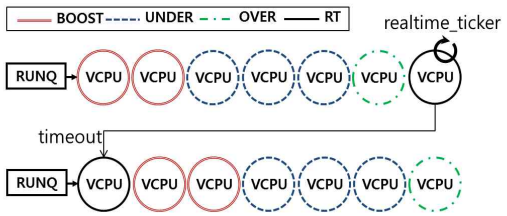

그림 3 타이머 기반 RT 스케줄링 개념도
Fig.3. Timer-based RT scheduling diagram

적인 작업을 하면서 일정시간을 sleep하게 되면 이때 sleep 만큼의 시간을 하이퍼바이저에 알려주고 잠을 잘 수 있도록 한다. 이때, 각 도메인에서 수행되는 태스크가 하이퍼콜을 통하여 sleep 시간을 지정할때에 극단적으로 짧은 시간에 재스케줄링이 되는 것을 막기 위하여 최소값을 지정하는 방법을 사용하였다. 프로토타입 시스템에서는 5ms 이상의 sleep 시간을 지정하도록 설정하였다.

하이퍼바이저에서는 RT 요청이 있고 sleep 하면서 요청한 시간을 realtime_ticker에 기록하고 해당 시간이 지나면 runq에 스케줄링 인터럽트를 보내서 runq에서 새로운 가상머신을 스케줄링 할 수 있다. 하지만 이때 우선순위가 가장 높은 도메인을 스케줄링할 확률이 높기 때문에 RT 모드로 동작하는 도메인이 더욱 자주 스케줄링 될 수 있도록 하였다. 이때, RT 우선 순위의 도메인이 지속적으로 수행됨으로 BOOST, UNDER, OVER에 있는 도메인이 기아 상태(starvation)에 빠지는 상황을 막는 방법이 고려되어야 한다. 이를 위하여 RT 우선순위도 BOOST와 같은 방법으로 10ms 마다 소모한 credit을 계산할 때 UNDER 우선순위로 전환하여 다른 도메인들이 기아 상태에 빠지는 것을 막고 있다.

RT 우선순위가 가장 높기 때문에 RT가 가장 먼저 스케줄링 되면 BOOST 도메인의 I/O 처리 성능이 떨어 질 수 있다. 따라서, 그림 3과 같이 리얼타임 타이머를 등록하여 지정된 시간이 만료되면 runq에서 가장 먼저 스케줄링 될 수 있도록 하였다.

3.2 시스템 모니터링 기법

본 연구에서 제안하는 시스템은 그림 4와 같이 Xentrace를 이용하여 Xen 스케줄러에서 발생하는 스케줄링 이벤트를 감시하여 각 도메인의 스케줄링 되는 상태를 분석할 수 있다. 그림 4에서 Xentrace Report는 trace buffer로부터 xentrace 데이터를

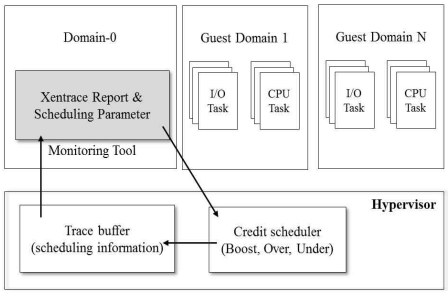

그림 4. 모니터링 시스템 아키텍처
Fig. 4. Monitoring system architecture

수집하여 분석한다. Xentrace로부터 수집한 각 도메인별 CPU 사용정보를 이용하여 각 도메인의 특성을 분석할 수 있다.

Xentrace는 Xen 하이퍼바이저에서 발생하는 이벤트들을 기록하는 도구로 Xen 패키지에 포함되었다. 본 연구에서는 Xentrace에서 발생하는 이벤트를 기록하여 각 도메인들이 BOOST, UNDER, 그리고 OVER의 우선순위 상태 별로 어느정도 시간을 사용하였는지 측정하였다. Xentrace에서 제공하는 이벤트의 포맷은 다음과 같다.

CPU(uint) TSC(u64) EVENT(u32) D1 D2 D3 D4 D5 (all u32)

CPU는 프로세서 번호를 나타내고 EVENT는 발생한 이벤트 아이디를 나타내며 D1부터 D5까지는 해당 이벤트에 관련된 데이터를 나타낸다. 바이너리 형태의 Xentrace 데이터는 xentrace_format 툴을 이용하여 해독이 가능한 형태의 ASCII 포맷으로 변경한 뒤 유저가 직접 trace 데이터를 보고 분석할 수 있다. 본 연구에서는 Xentrace에 각 도메인이 스케줄링될 때의 우선순위와 함께 해당 우선순위에서 CPU를 사용한 시간을 기록하여 스케줄링 정보를 분석하였다.

크레딧 스케줄러에서 CPU를 주로 소모하는 경우에는 UNDER 상태에서 credit을 모두 소모할 때까지 동작되며, I/O 요청을 주로 하는 도메인의 경우 해당 credit을 거의 사용하지 않고 짧은 시간동안 BOOST 상태로 스케줄링 되어 CPU를 사용한다. I/O 요청이 많은 태스크를 도메인에서 실행하면 해당 도메인은 다른 도메인에 비해 굉장히 자주 스케줄링 된 것을 알 수 있다. 해당 도메인은 자주 스케줄링 되는 반면에 CPU를 거의 소모하지 않는 것을 알 수 있다. 반면에 CPU를 주로 사용하는 도메

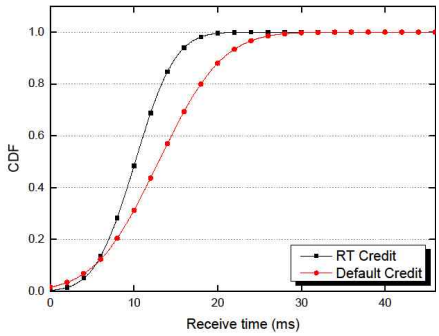

그림 5 RT-Credit에서 핑 도착 시간
Fig. 5. Ping arrival time in the RT-Credit

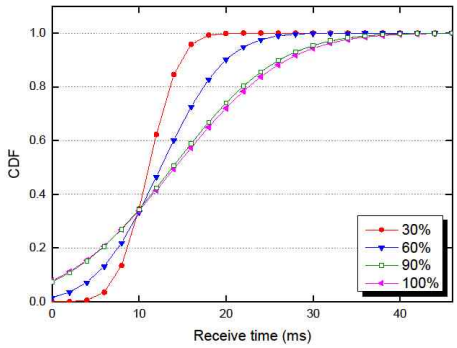

그림 6 CPU 점유율에 따른 지연시간
Fig. 6. Latency on varying CPU share

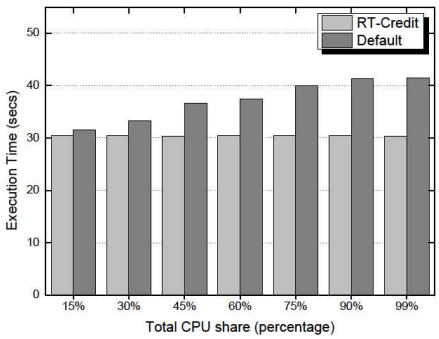

그림 7 CPU 점유율에 따른 성능 측정
Fig. 7. Performance evaluation on varying CPU share

인과 같은 경우는 UNDER 우선순위를 주로 받으면서 스케줄링 되며 문맥 전환의 횟수가 I/O 상황과 비교해 현저히 낮아진다. CPU를 주로 사용하는 도메인과 같은 경우는 스케줄링이 되면 다른 도메인으로 교체하기 전까지 주어진 시간동안 CPU를 계속 사용하기 때문에 문맥전환 비율이 낮아진다. 하지만, CPU를 주로 사용하는 도메인에서도 BOOST 우선순위가 나타나는데 이는 다른 도메인이 CPU를 선점하면서 블록상태에 있다가 깨어날 때 크레딧 스케줄러에 의해서 잠깐 BOOST 되었다가 UNDER 우선순위로 변환된다.

Ⅳ. 실험 및 성능평가

본 연구에서 제안하는 시스템의 유효성을 검증하기 위해 짧은 주기(10ms)로 깨어나서 외부 서버에 패킷을 전송하는 실험 환경을 구성하였다. 외부 서버에서는 패킷이 도착한 시간만을 기록하며, 두 대의 머신은 물리적으로 같은 스위치에 물려 있기 때문에 네트워크 전송 지연시간은 고려하지 않아도 된다. 가상 머신은 10ms 마다 패킷을 전송하고 sleep을 하도록 구현하였다. 패킷이 도착한 시간은 해당 도메인이 스케줄링 되어 패킷을 보낸 시간이라고 보면 된다. 이때 패킷 도착 시간이 10ms 가 넘으면 스케줄링에 의한 지연시간으로 볼 수 있다. 실험을 하고 있는 동안 세 개의 다른 도메인에서는 일정량의 CPU를 소모하면서 CPU를 점유하기위해 경쟁하고 있는 상황이다. 해당 도메인은 10ms 주기의 태스크를 3,000회 반복하면서 지연시간을 측정하였다.

그림 5는 RT 요청을 주기적으로 보내는 도메인과 그렇지 않은 경우 응답시간을 CDF(cumulative distribution function)을 표현한 것이다. 실험을 하는 동안 세 개의 CPU연산 집중적인 도메인 세 개가 각각 20%의 CPU를 소모하고 있는 상황이다. 지속적으로 RT 요청을 보내면서 주기적으로 동작하는 도메인이 깨어날 시간을 하이퍼바이저에 알려주어 제 시간에 해당 도메인을 스케줄링 할 수 있기 때문에 제안하는 기법(RT-Credit)과 같은 경우는 응답 시간이 대부분 10ms 근방에 기록되는 것을 알 수 있다.

세 개의 도메인이 각 10%, 20%, 30% 의 CPU를 점유하고 있는 상황에서 해당 도메인의 지연시간을 측정해 보았다. 그림 6과 같이 다른 도메인의 CPU 점유율이 높아짐에 따라 지연시간 또한 늘어나는 것을 알 수 있다.

그림 7과 같이 해당 도메인이 스케줄링 되지 못하고 지연되는 경우 최대 10초 이상 차이가 나는 것을 확인할 수 있다. 본 연구에서 제안하는 방법을

이용하면 전체 30초 근처에서 모든 처리가 완료 되는 것을 알 수 있다.

Ⅴ. 결론 및 향후 연구

본 연구에서는 응답시간에 민감한 태스크들이 동작하는 가상머신을 지원하기 위한 타이머 기반 크레딧 스케줄러를 제안하고 실험하였다. 본 연구에서 제안하는 방식은 기존의 Xen 하이퍼바이저의 스케줄링 정책을 수정하여 짧은 지연 시간을 태스크를 지원할 수 있다. 본 논문의 유용성을 보이기 위해서 Xen 가상화 환경에서 가상머신들이 스케줄링 되는 특성을 분석하고 실험을 수행하였다. 실험 결과 본 연구에서 제안하는 스케줄링 기법이 효과적으로 짧은 지연 시간을 필요로 하는 가상 머신 환경에서 사용될 수 있음을 보였다.

참고문헌

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization," Proceedings on the 19th ACM symposium on Operating systems principles, pp.164 - 177, 2003.

[2] S. Govindan, A. Nath, A. Das, N. Urgaonkar, A. Sivasubramaniam, "Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting plat forms," Proceedings on the 3rd international conference on Virtual execution environments, 2007.

[3] H. Kim, H. Lim, J. Jeong, H. Jo, J. Lee, "Task-aware virtual machine scheduling for I/O performance," Proceedings on the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp.101-110, 2009.

[4] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat, "Enforcing performance isolation across virtual machines in Xen," Proceedings on the 7th International Middleware Conference, 2006.

[5] H. Chen, H. Jin, K. Hu, M. Yuan, "Adaptive Audio-aware Scheduling in Xen Virtual Environment," Proceedings on ACS/IEEE International Conference on Computer Systems and Applications, 2010.

[6] M. Lee, A.S. Krishnakumar, P. Krishnan, N. Singh, S. Yajnik. "XenTune: Detecting Xen Scheduling Bottlenecks for Media Applications," Proceedings on GLOBECOM. 2010.

[7] 박현찬, 김세원, 유혁, "함수 단위 동적 커널 업데이트 시스템의 설계와 평가," 대한임베디드공학회논문지, Vol. 02, No. 03, pp.145-154, 2007.

[8] S. Yoo, K.-H. Kwak, J.-H. Jo and C. Yoo, "Toward Under-Millisecond I/O Latency in Xen-ARM," Proceedings on The 2nd ACM SIGOPS Asia-Pacific Workshop on Systems, Shanghai, 2011.

[9] R. Nikolaev, G. Back, "Perfctr-Xen: a framework for performance counter virtualization," Proceedings on the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, 2011.

[10] B. Gerofi, Y. Ishikawa, "Enhancing TCP throughput of highly available virtual machines via speculative communication," Proceedings on the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments, 2012.

저 자 소 개

김 병 기

2006년 한림대학교 컴퓨터공학과 학사.
2008년 한림대학교 컴퓨터공학과 석사.
2012년 9월 한림대학교 컴퓨터공학과 박사.
현재, 한림대학교 컴퓨터공학과 연구원.
관심분야: 임베디드 소프트웨어, 가상화 시스템
Email: bkkim@hallym.ac.kr

고 영 웅

1997년 고려대학교 컴퓨터학과 학사.
1999년 고려대학교 컴퓨터학과 석사.
2003년 고려대학교 컴퓨터학과 박사.
현재, 한림대학교 컴퓨터공학과 부교수.
관심분야: 임베디드 소프트웨어, 실시간 운영체제, 파일 처리, 가상화 시스템 소프트웨어
Email: yuko@hallym.ac.kr