Xen 가상 머신에서 QoS를 고려한 실시간 자원 할당 기법

김병기*, 장재혁, 허경우, 고영웅
*한림대학교 컴퓨터공학과
e-mail:{bkkim, jaehyok2, rapperkw, yuko}@hallym.ac.kr

Realtime Resource Allocation Scheme Considering QoS on Xen Virtual machine

Byung Ki Kim*, Jae Hyeok Jang, Kyung Woo Hur, Young Woong Ko
*Dept of Computer Engineering, Hallym University

요 약

Xen과 같은 가상 머신에서 각 게스트 운영체제가 필요로 하는 CPU 요구량을 정확하게 측정하기는 어렵다. SEDF 스케줄러는 사용자가 각 게스트 운영체제의 CPU 할당량을 직접 입력하고 있다. 따라서 가변적인 부하를 가지고 있는 상황에서 게스트 운영체제의 스케줄링이 어렵다. 본 논문에서는 작업량이 가변적으로 변화하는 시스템의 QoS를 고려하여 실시간 태스크가 필요로 하는 CPU 자원을 효율적으로 할당하는 방법을 제안하였다. 실험을 통하여 제안한 방식이 가변적인 작업량에 대해서 효율적으로 동작됨을 보였다.

1. 서론

Xen에서는 두 단계의 스케줄링을 통해 게스트 도메인의 태스크를 수행시킨다. VMM 스케줄러는 VCPU 단위로 도메인을 스케줄링하며, VMM 스케줄러에 의해 스케줄링된 도메인은 각 도메인의 고유한 스케줄링 정책에 따라서 각 태스크들을 스케줄링한다. VMM 스케줄러는 관리자가 지정해준 정책에 의해서 각 도메인이 필요로 하는 자원량을 확인할 수 없기 때문에 사용자가 직접 각 도메인들이 필요로 하는 자원을 할당해야 한다. 기존의 연구들은 하이퍼바이저의 모니터링 툴을 이용하여 도메인의 상황을 감시하거나 장치 접근을 감지하여 자원 사용을 예측하고 할당하는 방법들을 제안하였다[2,3,4]. 하지만, 대부분의 접근 방법들은 워크로드가 가변적인 환경에서 도메인이 필요로 하는 CPU 요구량이 얼마인지 정확하게 구분할 수 있는 방법이 없다. 따라서 가상화로 구축된 서버에서 실시간으로 변화하는 작업량을 효율적으로 처리할 수 있는 피드백 스케줄링 메커니즘이 요구된다.

본 논문에서는 오픈소스 기반의 Xen[1] 하이퍼바이저(Hypervisor)에서 연성 실시간 태스크의 QoS를 만족시킬 수 있는 스케줄링 방안을 제안한다. 작업량이 가변적인 환경에서 게스트 도메인의 QoS를 모니터링하는 모듈을 설계하고 QoS가 일정 수준 이하로 떨어지는 경우 하이퍼콜(Hypercall)을 이용해 스케줄러에서 가상머신들의 CPU 할당량을 재조정함으로써 태스크의 QoS를 만족시킬 수 있다. 본 연구에서는 동적으로 CPU 할당량을 재조정하는 방식으로 멀티미디어 스트리밍과 같은 연성 실시간 서비스들이 제한 시간 내에 잘 처리가 됨을 목표로 한다.

이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2010-0005442)

2. 시스템 설계 및 구현

본 연구에서 제안하는 방법은 게스트 도메인을 실시간 게스트 도메인(RT Guest Domain)과 비실시간 게스트 도메인(NRT Guest Domain)으로 구분하여 스케줄링한다. Xen 하이퍼바이저의 SEDF(Simple Earliest Deadline First) 스케줄러는 게스트 도메인들이 생성되면 기본적으로 BestEffort 모드(500 micro second, round robin)로 동작되어 게스트 도메인이 필요로 하는 CPU 자원을 효율적으로 지원하기 어렵다[5].

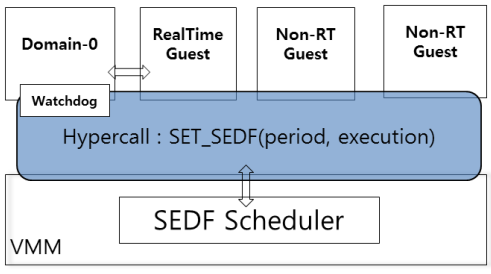

그림 1. 피드백 기반 동적 스케줄링 개념도

그림 1과 같이 RT-Domain에서 실시간 태스크의 QoS를 측정할 수 있는 모듈이 실행되어 일정한 주기마다 Domain-0에 있는 스케줄링 모듈에게 스케줄링을 요청하게 된다.

```
Input: Dom_id, period
CO_SEDF_Schedule()
begin
    while TRUE do
        watchdog(SET_SEDF_SLICE.Dom_id);
        if SET_SEDF_SLICE == TRUE then
            slice++;
            set_sched_sedf_domain_set(Dom_id, slice);
            gettimeofday(start);
        else
            gettimeofday(end);
            if (end - start) >= period then
                if slice > MIN_SLICE then
                    slice--;
                    set_sched_sedf_domain_set(Dom_id, slice);
                    gettimeofday(start);
                end
            end
        end
    end
end
```

그림 2. 피드백 스케줄링 알고리즘

```
Input: period, workload, exec_rate
RT_Guest_CPU_Bound()
begin
    while TRUE do
        gettimeofday(start);
        for i=0; i < exec_rate; i++ do
            Do_hash(workloads);
            count++;
            gettimeofday(end);
            /* every second */
            if end - start >= 1 then
                break;
            end
        end
        /* Deadline missed */
        if count < exec_rate then
            xenstore_write(SET_SEDF_SLICE);
        end
    end
end
```

그림 3. 스케줄링 요청 알고리즘

게스트 도메인에서 동작하는 실시간 태스크의 QoS가 떨어지는 상황이 발생하면 게스트 도메인에서 이벤트 채널을 통해 Domain-0에 현재 RT-Domain의 QoS 상태를 보낸다. Domain-0에 설치된 Watchdog을 통해 이벤트를 감지하면 SEDF의 스케줄러 옵션을 조정할 수 있는 하이퍼콜을 발생시킨다. 하이퍼바이저에서 Domain-0로부터 하이퍼콜을 받으면 set_sched_sedf_domain_set 하이퍼콜을 통해 RT-Domain의 slice값을 늘려주거나 감소시켜줄 수 있다[6].

그림 2는 피드백 스케줄링 알고리즘을 보이고 있다. 게스트 도메인의 QoS 모듈에서 스케줄링이 필요한 경우에 xenstore_write(SET_SEDF_SLICE) 함수를 호출하여 스케줄링이 필요함을 이벤트를 통하여 알리게 된다. 이벤트를 전달하는 방식은 xenbus에 게스트 도메인의 요청을 위한 노드를 생성하고 데드라인을 놓치는 경우가 발생하면 이 노드를 통해 Domain-0에 이벤트를 보낸다. Domain-0에서는 watchdog을 이용해 이벤트 발생 여부를 모니터링하며, 게스트 도메인으로부터 스케줄링 요청 이벤트가 발생하면 Domain-0에서 하이퍼콜을 이용해 slice를 증가시킬 수 있다. 피드백 스케줄링 알고리즘에서는 watchdog을 이용하여 게스트 도메인의 요청을 감지하고 slice를 증가시켜서 게스트 도메인의 CPU 용량을 늘려준다.

3. 성능평가 및 분석

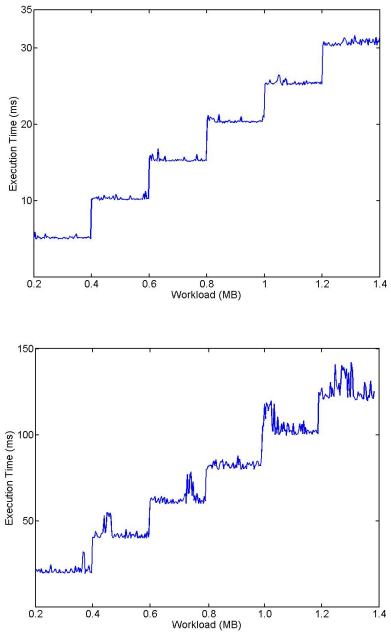

그림 4. 실시간 태스크 수행시간 측정

그림 4-(상)은 RT-Guest 도메인에 이벤트 기반의 실시간 태스크를 동작시키고 나머지 NRT-Guest들은 동작하지 않는 상황이다. 태스크는 1분 단위로 작업량을 0.2MB씩 늘리면서 실험하였다. 각 1분 동안 1초 단위로 30 회의 해시를 계산하는 한다고 했을 때 한번 해시를 할 때 처리시간을 측정하였다. 데이터 블록의 크기를 늘려 작업량을 증가시켰을 때 평균적으로 해시를 처리하는 시간이 33ms를 넘지 않으면 제한시간 내에 모두 처리되었다고 볼 수 있다. 그림 4-(하) 그래프는 과부하를 발생시키기 위해서 NRT-Guest가 CPU를 소모하는 경우, 실시간 태스크의 수행 결과를 보인다. 이때 해시를 계산하는 시간이 증가하게

되므로 대부분 데드라인 내에 작업을 처리할 수 없는 것을 확인할 수 있다. NRT-Guest가 수행되지 않을 때와 비교했을 때 80% 가까이 데드라인을 놓치는 것을 확인하였다.

-주기적 실시간 태스크(CPU intensive periodic realtime task): 순수하게 CPU의 처리 성능만을 측정하기 위한 태스크로 주기적으로 동작하여 MD5 해시를 계산하는 태스크이다. 본 실험에서는 동영상 디코딩과 같은 방식으로 1초 동안 30번의 해시를 계산하는 태스크를 생성하였다. 이때 1분 단위로 해시를 계산하는 데이터의 용량을 늘려서 워크로드를 가변적으로 늘려주었다. 이때, 초당 해시를 계산한 횟수를 기록하여 1초 동안 30번의 해시를 계산하지 못하면 데드라인을 놓친 것으로 판단할 수 있다. 본 실험에서도 3개의 NRT-Guest가 CPU를 소모하고 있는 상황에서 Co-SEDF의 RT-Guest의 처리 성능을 측정하였다.

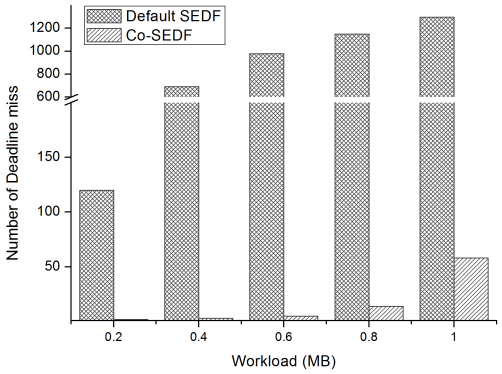

그림 5. RT 태스크의 제한시간 실패 결과

그림 5의 Y축은 워크로드가 변하는 동안 데드라인을 놓친 개수를 측정한 것이다. 본 실험은 1초에 30회의 해시를 계산하는 테스크가 1분 동안 데드라인을 놓친 개수를 측정한 것이다. Default- SEDF는 작업량이 늘어날수록 초당 처리하는 횟수가 줄어드는 것을 볼 수 있다. 초당 처리 횟수가 줄어든다는 것은 데드라인을 놓치고 있는 상황이 증가하고 있다는 것이다. 해시 처리 시간이 늘어남에 따라 데드라인을 놓치는 경우가 많이 발생하는 것을 알 수 있다.

반면, Co-SEDF는 작업량을 증가시키면서 실시간 태스크를 수행시키더라도 데드라인을 놓치는 경우가 줄어드는 것을 알 수 있다. 데드라인을 놓치는 상황이 발생하면 짧은 시간 이내에 RT 게스트의 CPU slice가 증가하여 데드라인 내에 처리가 완료되도록 하고 있다. Co-SEDF에서는 데드라인을 놓치는 구간이 발생할 때 마다 VMM 스케줄러에게 피드백을 주어 스케줄링을 요청함으로써 slice를 더 할당할 수 있기 때문에 데드라인 미스가 크게 줄어드는 것을 확인할 수 있다.

4. 결론 및 향후연구

본 논문에서는 가상머신에서 멀티미디어 스트림 서비스와 같은 연성 실시간 태스크의 QoS를 모니터링하고 실시간 태스크가 제한시간을 놓치는 경우 동적으로 CPU 자원을 조정하여 QoS를 만족시킬 수 있는 방법에 대해 설명하였다. 게스트 도메인의 작업량이 가변적인 환경에서는 도메인의 CPU 자원을 런타임에 변경하기 어렵기 때문에 데드라인을 놓치는 경우가 종종 발생한다. 본 논문에서 제안하는 방법은 연성 실시간 태스크들이 데드라인을 놓칠 때마다 Xen 하이퍼바이저에 피드백을 주어서 SEDF의 slice를 증가/감소함으로 데드라인 내에 처리가 가능하게 하였다. 실험을 통하여 워크로드가 가변적인 상황에서 제안하는 방식이 효과적으로 동작이 됨을 보였다. 향후, 실시간 도메인에 실시간 태스크와 비실시간 태스크가 혼재해 있는 상황에서 효율적으로 자원을 스케줄링 하는 방법에 대한 연구를 진행할 계획이다.

참고문헌

[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. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164 - 177, New York, NY, USA, 2003. ACM.

[2] Govindan S., R. Nath A., Das A., Urgaonkar B., and Sivasubramaniam A. Xen and Co.: Communication-aware CPU Scheduling for Consolidated Xen-based Hosting Platforms. In Proceedings of the Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE), June 2007 San Diego, CA.

[3] Kim H., Lim H., Jeong J., Jo H., and Lee J. Task-aware virtual machine scheduling for I/O performance. In VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (2009), pp. 101-110.

[4] Gupta D., Cherkasova L., Gardner R., and Vahdat A. Enforcing performance isolation across virtual machines in Xen. In Proceedings of the Seventh International Middleware Conference, Melbourne, Australia, Nov-Dec 2006.

[5] Cherkasova L., Gupta D., and Vahdat A. Comparison of the three CPU schedulers in Xen. SIGMETRICS Perform. Eval. Rev., 35(2):42 - 51, 2007.

[6] Gazagnaire T., and Hanquez V. Oxenstored: an ef ficient hierarchical and transactional database using functional programming with reference cell comparisons. In ICFP '09: Proceedings of the 14th ACM SIGPLAN international conference on Functional programming, pages 203 - 214, New York, NY, USA, 2009. ACM.