임베디드 환경에서의 가상머신 스케줄러

(Virtual machine scheduler in embedded environment)

박 찬 웅, 유 혁*
(Chan-Woong Park, Chuck Yoo)

Abstract : As more computer resources are available, we can efficiently use them through the virtualization. In addition, the virtualization can improve security and mobility. That's why virtualization has been studied in server or desktop environment, Now it is studied in embedded environment, too. So, many effort is tried to porting XEN to embedded environment.

XEN is focused on network server virtualization, its aim is to increase throughput and decrease latency of network. It may be inadaptable in embedded environment which has relatively insufficient resources. Therefore, this paper focuses on the problem when XEN virtual machine scheduler is used in embedded environment. And we present the improvement of it.

Keywords : virtualization, hypervisor, virtual machine, scheduler

Ⅰ. 서 론

하드웨어의 발전에 따라 단일 운영체제로서는 가용한 자원을 효율적으로 모두 사용할 수 없게 되었다. 또한, 프로그램 코드의 크기와 복잡도의 증가는 하나의 운영체제에서 많은 수의 응용 프로그램의 관리를 어렵게 만들었다. 이러한 문제의 해결 방법으로써 플랫폼 가상화에 대한 요구가 증대되어 왔다.

이런 현상은 임베디드 시스템에서도 나타난다. 임베디드 소프트웨어의 복잡도는 지난 10년간 두 배씩 증가되어 왔다. 더구나 응답성에 민감한 멀티 쓰레드 임베디드 프로그램은 운영체제가 기본적으로 복잡성을 내재하게 한다. 따라서 이러한 내재된 문제점을 해결하고, 보안성과 이동성을 증대 시키기 위해 임베디드 환경에서도 가상화 플랫폼에 대한 요구가 늘어나고 있다.

하지만, 현재 임베디드 환경에서 가상화 기법에 대한 연구는 기존의 기법을 임베디드 환경으로 옮겨오는데 집중되어 있다. 즉, 상대적으로 제한된 자원을 가지는 임베디드 환경에 대한 고려가 되지 않고 있다. 그러나, 가상화란 하드웨어적인 자원을 추상화 하여 각각의 운영체제들이 이를 자신의 독립적인 자원으로 인식하여 사용할 수 있게 해주는 기법이다. 그러므로 하드웨어적인 특성이 고려되지 않은 가상화 정책은 문제점을 가지게 된다.

현재 임베디드 환경으로 포팅이 이루어지고 있는 Xen은 네트워크 서버들을 효율적으로 사용할 수 있도록 하는 것이 목적이다. 따라서 기본 스케줄러인 Credit 스케줄러는 각 도메인들이 물리 자원을 공평하게 분배 받을 수 있도록 해준다. 하지만, 서버 환경과 달리 임베디드 시스템에서 모든 가상머신의 소유자는 한 명이기 때문에, 이런 정책은 임베디드 환경에서 맞지 않다.

가상머신 스케줄링 정책은 가상머신 모니터의 성능을 결정짓는 중요 요소 중 하나가 된다. 이 정책에 의해 하나의 물리적인 자원에서 운용되는 여러 가상머신들이 실제 자원을 점유할 수 있게 되기 때문이다. 따라서 본 논문에서는 가상머신 모니터의 여러 정책 중 가상머신 스케줄러에 그 초점을 두고 있다. 현재 Xen[1]에서 기본 스케줄러로 사용되고 있는 Credit 스케줄러를 기반으로 하여 이것이 임베디드 환경에서 그대로 사용했을 때 발생할 수 있는 문제점을 제시한다. 또한 제시된 문제점들을 바탕으로 하여 이를 개선시킬 수 있는 방법을 제안하고 있다.

Ⅱ. 본 론

* 교신저자(Corresponding Author)
박찬웅, 유혁 : 고려대학교 컴퓨터학과

1. Credit 스케줄러

현재 Xen의 기본 스케줄러로 등록되어 있는 Credit 스케줄러[2]는 UNDER와 OVER의 두 가지 우선순위만을 갖는다. 이는 Credit 스케줄러가 빠른 선택과 공평성을 목표로 하기 때문이다. 스케줄링의 대상이 되는 VCPU는 일정 시간(30ms)마다 credit을 받는다. UNDER 우선순위를 갖던 VCPU는 자신의 credit을 모두 소모하면 OVER 우선순위가 된다. 스케줄러는 일반적으로 큐에서 UNDER 우선순위를 갖는 VCPU만을 선택하면 되기 때문에 일반적으로 O(1)의 시간 복잡도로 스케줄링이 가능하다. 또한 할당 받은 credit에 따라 스케줄링이 결정되기 때문에 공평성을 보장받게 된다.

하지만, 이 UNDER와 OVER 우선순위는 물리 CPU 점유에 대한 우선순위이다. 즉, IO에 대한 우선순위는 고려되지 않고 있다. 그러므로 분리된 드라이버(split driver)모델을 사용하는 Xen(그림 1)에서는 IO에 대한 응답성이 떨어지게 된다. IO 처리를 담당하는 드라이버 도메인도 CPU점유에 대한 우선순위로서만 고려되기 때문이다.

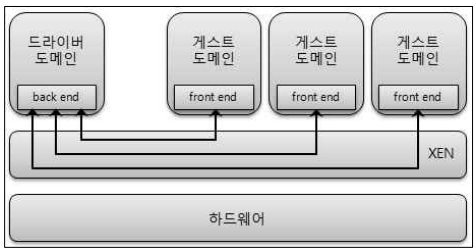

그림 1. 분리된 드라이버 모델

이 문제를 해결하기 위해, 특정 도메인이 유휴상태에서 wake함수에 의해 깨어날 때 BOOST우선순위를 주어 IO 응답성을 높일 수 있도록 해주었다[3]. 드라이버 도메인은 back-end 드라이버에 의한 IO 처리가 주 작업이기 때문에 대부분 BOOST 우선순위를 갖게 된다. BOOST 우선순위는 모든 우선순위보다 높기 때문에, 인터럽트에 의해 항상 드라이버 도메인이 스케줄링 되게 된다. 이것은 범용 커널에서 인터럽트 처리를 위해 ISR이 실행되는 루틴과 유사한 형태를 취하게 된다.

그러나 문맥 전환의 관점에서 중요한 차이점이 있다. 범용 커널에서의 디바이스 드라이버는 커널과 같은 공간을 사용하기 때문에 ISR이 실행될 때 문맥 전환이 일어날 필요가 없다 (그림 2). 하지만, 드라이버 도메인은 하나의 도메인 객체로서 다른 주소 공간을 사용하기 때문에 문맥 전환을 필요로 한다는 것이다(그림 3).

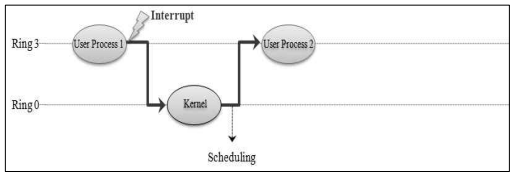

그림 2. 범용 커널에서의 인터럽트 처리

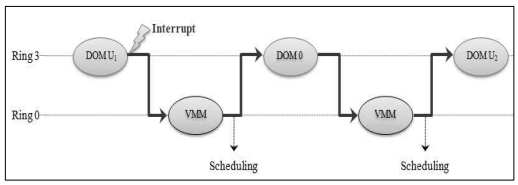

그림 3. 가상머신 모니터에서의 인터럽트 처리

따라서 가상머신 모니터는, 인터럽트 처리 후 다른 도메인(프로세스)으로의 전환까지 범용 커널에 비해 두 배의 문맥 전환을 필요로 하게 된다. 문맥 전환에 드는 비용은 CPU자원이 충분하다면 문제가 되지 않을 수도 있다. Xen은 CPU자원이 충분한 네트워크 서버가 대상이므로, 문맥전환에 드는 비용을 희생함으로써 얻을 수 있는 네트워크 성능 증가가 상대적으로 더 클 수 있다.

하지만 일반적으로 문맥 전환은 TLB flush를 동반하기 때문에, 잦은 문맥 전환은 상대적으로 CPU 자원이 부족할 수 있는 임베디드 시스템에서 성능 저하의 원인이 될 수 있다. 그러므로 임베디드 시스템에서는, IO의 응답성만을 고려하기 보다는 CPU의 처리량에 따라 인터럽트의 처리 속도를 조절 할 수 있어야 한다.

2. 이벤트 채널

Xen에서는 분리된 드라이버 모델을 사용하기 때문에 비동기화 메커니즘으로써 이벤트 채널을 사용한다[2]. 게스트 도메인에서는 이벤트 채널에 IO에 대한 요청을 하게 되며, 드라이버 도메인에서는 펜딩된 이벤트를 보고 해당 인터럽트를 처리하게 된다. 인터럽트가 비동기적으로 처리될 수 있기 때문에, Tx 인터럽트는 즉시 처리되어야 할 필요가 없다. 이상적인 경우 모든 게스트 도메인의 Tx 요청을 기록해 놓을 수 있다면 한번의 드라이버 도메인 전환으로 모든 요청을 처리할 수 있다. (그림 4)와 같은 경우, (그림 5)와 같은 수의 인터럽트를 처리

하면서 4번의 문맥전환을 덜 한다.

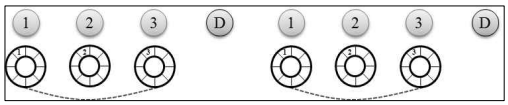

그림 4. 이상적인 경우의 드라이버 도메인 스위칭

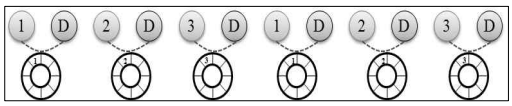

그림 5. BOOST 상태의 드라이버 도메인 스위칭

하지만, Credit 스케줄러에서는 이렇게 스케줄링이 되기가 힘들다. 일반적으로 다른 도메인으로의 스위칭 간격보다 Rx 인터럽트가 들어오는 시간 간격이 더 짧기 때문에, Rx 처리를 위해 드라이버 도메인으로 전환이 일어나기 때문이다.

하나의 IO에 대한 응답 시간은 (그림 5)와 같은 방식이 좋지만, 다음 IO를 요청하기 위한 대기 시간이 더 길며 더 많은 문맥 전환을 요구한다. 그러므로 임베디드 환경에서는 IO의 종류와 성질에 따라, 드라이버 도메인이 어떻게 스케줄링 되어야 할지에 대한 고려가 필요하다.

3. 문맥 전환 오버헤드

BOOST 우선순위는 드라이버 도메인과 게스트 도메인 사이의 명시적인 문맥 전환을 유발한다. 또한, 한 번에 한 도메인의 이벤트만을 처리할 수 밖에 없게 한다. 두 가지 문제 모두 문맥 전환이 얼마나 자주 일어나느냐가 문제가 될 수 있게 된다. 즉, 일정 시간 동안의 문맥 전환의 횟수가 다른 작업에 영향을 미치지 않는 수준이라면, 문맥 전환에 따른 부작용은 고려할 필요가 없을 수도 있다. 따라서, 일반적인 환경에서 문맥 전환이 얼마나 많이 일어나는가에 대한 실험이 필요하다.

(그림 6은)은 실제 수행 시간중 문맥 전환이 차지하는 비율인 문맥 전환 오버헤드를 나타낸 것이다. 100Mb/s의 NIC이 설치된 2.0GHz Dual core에서 측정되었고, 드라이버 도메인을 제외한 4개의 게스트 도메인에서 Iperf을 실행하였다. 문맥 전환 오버헤드는 [문맥 전환 시간 / (문맥 전환 시간 + 문맥 전환 이후 도메인 실행 시간) * 100] 으로 계산하였다. 이때, 문맥 전환 오버헤드는 약 12.13%로 측정되었고, 약 0.03ms마다 도메인간의 문맥 전환이 일어났다.

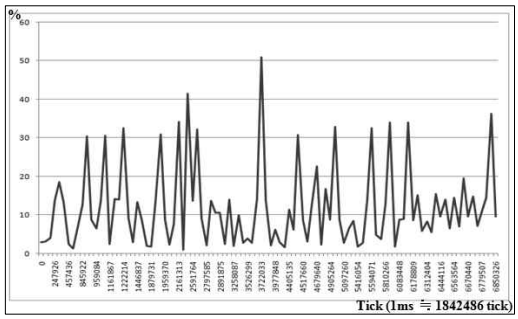

그림 6. 컨텍스트 스위칭 오버헤드

인터럽트에 의해 즉시 드라이버 도메인으로 전환되는 것이 응답성에 미치는 영향을 알아보기 위해 도메인의 개수를 늘려가며 ping에 대한 Latency를 측정 하였다. (표 1)에서 Latency1은 모든 도메인에서 ping만을 수행한 것이며, Latency2는 Iperf을 수행하면서 ping을 같이 수행하여 측정한 것이다. 도메인에서 다른 작업을 동반하는 경우, 도메인의 개수가 늘어남에 따라 응답성이 현격히 떨어지는 것을 Latency2에서 확인할 수 있다.

표 1. 도메인 개수에 따른 Throughput과 Latency

개수	Latency 1(ms)	Iperf (Mb/s)	Latency 2(ms)
1개	0.31	92.4	8.4
2개	0.34	88.3	25.6
3개	0.38	90.0	84.7
4개	0.41	91.3	148.2

하지만, 처리량은 도메인의 개수와 상관없이 일정한 것을 볼 수 있다. 이것은 비록 문맥전환에 드는 비용이 12%로 비교적 높지만, 처리량은 네트워크 성능에 의존된다는 것을 보여준다.

그러나 임베디드 환경은 네트워크 성능의 증대가 목적이 아니다. 따라서 만약 반복되는 IO가 많을 경우, 이에 따른 문맥전환에 드는 비용은 다른 CPU작업의 효율성을 떨어뜨릴 수 있다.

Ⅲ. 개선 방향

첫째, 가상머신 스케줄링에 있어서 IO에 대한 우선순위가 고려 되어야 한다. 현재는 드라이버 도메인이 언제나 최상위 우선순위(BOOST)를 가진다. 하지만 이것은 IO에 대한 우선순위이고, OVER와 UNDER는 CPU 사용에 대한 우선순위이다. 따라서 서로 다른 위상을 갖는 우선순위를 같이 비교하기

때문에 비효율적인 문맥전환이 이루어 지게 된다. 따라서, 게스트 도메인의 IO횟수와 이벤트 채널의 사용빈도 등을 고려하여 도메인의 우선순위를 설정할 수 있도록 해 주어야 한다.

둘째, 가상머신 모니터에서 인터럽트를 받은 후 가능한 범위 내에서 이전의 도메인이 다시 실행 될 수 있도록 해 주어야 한다. 도메인과 가상머신 모니터는 같은 주소 공간을 사용하기 때문에, 이전의 도메인으로의 스위칭은 문맥 전환이 필요 없다. 따라서 TLB 소모와 문맥 전환에 따른 비용이 들지 않게 된다.

기존의 Credit 스케줄러에서는 (그림 7)과 같은 전환이 이루어 지며, 총 7번의 스위칭이 일어난다. 하지만, (그림 8)과 같이 스위칭이 이루어 진다면 3번의 문맥 전환만을 필요로 한다. M은 가상머신 모니터에서 인터럽트를 펜딩 시키는 것을 나타낸다.

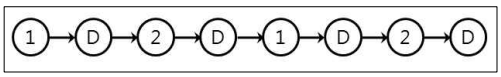

그림 7. 현재의 드라이버 도메인 스위칭

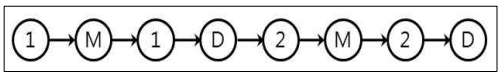

그림 8. 문맥 전환을 최소화한 스위칭

하지만, 즉시 드라이버 도메인으로 전환이 이루어 지지 않을 경우, 그 인터럽트는 처리된 것이 아니라 펜딩된 상태로 남아 있게 된다.그러므로, 인터럽트 지연이 가능한 범위 내에서 이루어 져야 한다.

IV. 결론 및 향후 연구 방향

IO처리를 우선시한 기존 Xen Credit 스케줄러를, 문맥전환 비용 증가에 따른 CPU 비효율성에 초점을 두었다. 이때 상대적으로 자원이 부족한 임베디드 환경에서 문제가 될 수 있는 부분을 설명하였으며 그 개선방향을 제시하였다.

향후 연구방향은 다음의 2가지 관점에서 접근하고자 한다. 첫째, 실제 임베디드 장비에서 문맥전환에 드는 비용을 측정함으로써, 인터럽트 처리가 CPU 자원 소모에 미치는 영향을 알아본다. 둘째, 드라이버 도메인의 스케줄링 비율이 인터럽트 처리 지연에 미치는 영향에 대해서 분석한다. 그럼으로써 지연 시간에 미치는 영향을 최소화 하고, CPU 이용률을 최대화 할 수 있는 스위칭 알고리즘에 대한 연구가 필요하다.

참 고 문 헌

[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. In Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003.

[2] I. Pratt, K. Fraser, S. Hand, C. Limpach, A.Warfield, D. Magenheimer, J. Nakajima, and A. Malick. Xen 3.0 and the art of virtualization. In Proc. of the 2005 Ottawa Linux Symposium.

[3] Diego Ongaro, Alan L. Cox, Scott Rixner. Scheduling I/O in virtual machine monitors. ACM/Usenix International Conference On VEE Ottawa, Canada, July 2005.

저 자 소 개

박 찬 웅(Chanwoong Park)
2007년 2월 : 고려대학교 컴퓨터학 학사
2007년~현재 : 고려대학교 컴퓨터학 석사과정
관심분야 : 가상머신, 스케줄러, 멀티 코어 플랫폼
Email : cwpark@os.korea.ac.kr

유 혁(Chuck Yoo)
1982년 2월 : 서울대학교 전자공학과 학사
1983년 2월 : 서울대학교 전자공학과 석사
1986년 8월 : Master of Computer Science in University of Michigan
1990년 8월 : Ph.D of Computer Science in University of Michigan
1990년~1995년 : Sun Microsystems Lab.Researcher
1995년~현재 : 고려대학교 컴퓨터학과 교수
관심분야 : 시스템 가상화, 멀티 코어 플랫폼, 커널 네트워킹, 센서 네트워킹, 멀티미디어 스트리밍
Email : hxy@os.korea.ac.kr