Kernel-based Virtual Machine 메모리 관리 분석+

남현우*, 박능수*, 이강우**
*건국대학교 컴퓨터공학과
**동국대학교 정보통신공학과
e-mail:namhw@konkuk.ac.kr

Memory Management Analysis in Kernel-based Virtual Machine

Hyunwoo Nam*, Neungsoo Park*, Kangwoo Lee**
*Dept of Computer Science & Engineering, Konkuk University
**Dept of Information & Communication, Dongguk University

요 약

리눅스 커널을 VMM(Virtual Machine Monitor)로 만들어 주는 KVM의 메모리 관리 기법을 분석한다. Xen과의 차이점과 KVM의 구조를 알아보고 KVM에서의 메모리 관리 기법에 대해 분석하였다. 또한 CPU의 가상화 기능인 Intel VT-x가 어떻게 적용되었는지 분석한다.

1. 서론

가상화(Virtualization) 기술은 물리적인 하드웨어 자원을 논리적인 자원 풀(pool)로서 이용할 수 있게하는 기술이다. 서버의 활용도를 높이기 위한 방안으로 시작되어 현재 서버 가상화, 애플리케이션 가상화, 네트워크 가상화, 스토리지 가상화등 많은 분야로 범위를 넓혀가고 있다.

본 논문은 리눅스 커널에 포함되어 성장 가능성이 커지고 있는 KVM(Kernel-based Virtual Machine)의 메모리 관리 기법에 대해 분석한다. 현재 KVM은 개발 된지가 오래되지 않아 관련 자료가 충분하지 않으며 KVM을 활용한 응용 시스템 및 관련 연구가 시작 단계 이다. 분석 연구를 통해 향후 KVM에서 가상화 머신 모니터링을 위한 library 구현에 기반자료로서 사용될 수 있을 것이다.

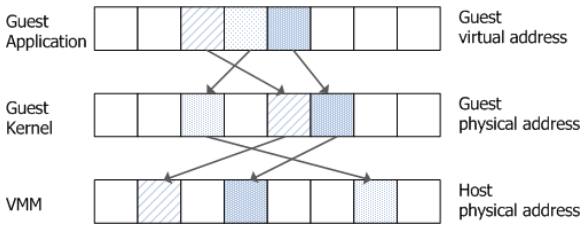
2. 관련연구

2-1 가상화 구조

VMM(Virtual Machine Monitor)은 Host 하드웨어와 Guest 운영체제 사이에 있다. 따라서 Guest 운영체제에서 하드웨어에 접근하기 위해서는 반드시 VMM을 거쳐야만 한다. CPU의 경우 OS에서 각각의 프로세스들에게 스케쥴링을 통해 CPU를 할당해주듯 각 가상머신들은 VMM이 스케쥴링 과정을 통해 CPU를 할당받는다. 메모리 또한 VMM에 의해서 할당되며 가상 머신의 가상 주소를 실제 물리주소로 변환할 때도 VMM을 거쳐야만 한다.

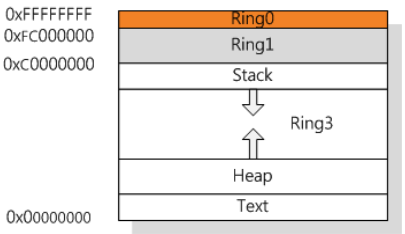
2-2 XEN의 메모리 관리

Xen[1]은 반가상화 방식으로 동작되는 대표적인 VMM으로 가상머신에서 메모리 할당이나 해제 같은 작업들은 담당하고 있다. 그림1에서 가상머신의 주소변환 과정을 살펴보면 먼저 Guest 가상 주소를 Guest 물리 주소로 변환한다. 다음으로 Guest 물리 주소는 VMM에 의해 Host 물리주소로 변환한다. Guest의 가상 주소에서 물리 주소까지 변환과정에는 VMM에서 주소 변환 과정이 추가적으로 필요하므로 오버헤드가 발생한다.



(그림 1) Xen 메모리 레이어간의 관계

아래의 그림2는 x86 시스템 메모리의 레이아웃을 보여주고 있다. Xen은 Ring0 영역에서 수행되고 있으며 메모리의 끝번지부터 64MB의 용량이 할당되어 위치하고 있다. 나머지 영역들은 리눅스 커널의 메모리 구조와 같이 0-3GB 영역은 응용 프로그램, 3-4GB 영역은 커널이 사용한다. Xen에서 커널은 Ring0이 아니라 Ring1영역을 사용하여 특권 명령어들을 사용할 수 없게 된다.



(그림 2) XEN 메모리맵

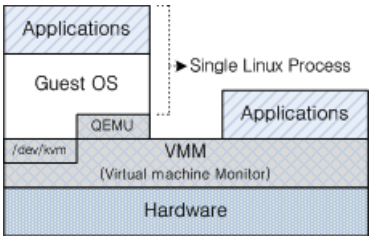
3. KVM 메모리 관리 구조

3-1 KVM의 구조

KVM은 리눅스 커널의 모듈 형태로 제작되었으며

+ 서울시 산학연 협력사업(10581)의 지원에 의한 논문임.

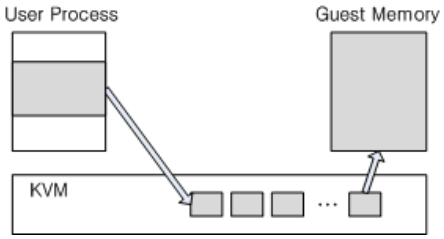
INTEL VT-x와 AMD-V 기능을 사용하는 VMM이다.[3] 그림과 같이 KVM 모듈이 올라가면 리눅스 커널은 VMM으로 전환된다. /dev/kvm을 사용하여 ioctl 시스템콜을 통해 가상머신에게 CPU나 메모리 할당을 하며 I/O의 가상화 처리는 QEMU 에뮬레이터가 담당하고 있다.



(그림 3) KVM의 구조

3-2 메모리 할당

분석에 사용된 KVM의 버전은 KVM-74 버전이며 프로세서는 인텔 제온을 사용하고 있다. 그림4는 가상머신에서 사용되는 메모리가 어떻게 할당되는지를 보여준다.[2]

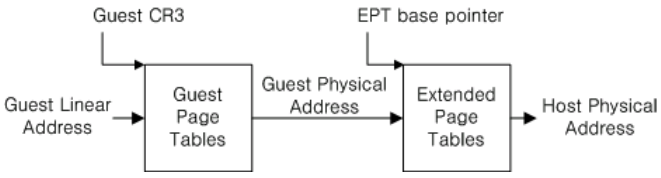


(그림 4) 메모리 할당 구조

우선 가용 메모리는 /dev/kvm 파일을 열고 가상 머신을 생성한 후 응용프로그램 계층에서 malloc() 시스템콜을 사용하여 마련하게 된다. 그리고 가상머신에서 필요한 만큼의 메모리를 사용자 프로그램에서 확보한 메모리 풀(pool)을 이용해 할당해준다. KVM에서 메모리의 할당은 리눅스 커널 메모리 할당 메커니즘을 사용한다는 것이 특징이다. 이 구조에서 응용프로그램은 mmap 시스템콜을 사용하여 가상머신의 메모리로 직접 접근할 수도 있다.

3-3 EPT(Extended Page Table)를 이용한 페이징 기법

KVM은 가상 메모리 주소의 변환 과정을 처리하기 위해 Shadow Paging을 사용하거나 CPU에서 제공하는 EPT(Extended Page Table)를 사용하고 있다.[4] EPT 테이블은 cr3 레지스터가 수정되거나 Page Fault가 발생했을 때 INVLPG 명령어가 수행되었을 때 제어할 수 있다.



(그림 5) Extended Page Table의 주소변환 과정

그림은 EPT를 이용한 주소 변화 과정을 보여주고 있다. 전체적인 구조는 Shadow Paging과 유사하다. 하지만 Guest에서의 가상주소가 Guest의 물리주소로 변환되고 난 후 Host의 물리 주소가 CPU에 의해 EPT 테이블을 참조하여 변환과정을 자동으로 수행해준다. 이는 추가적인 S/W에서의 변환과정을 생략하게 해주어 변환 과정시 trap에 의해 발생된 cpu 사용 시간을 줄일 수 있다. 즉 가상머신과 VMM에서의 전환 과정이 필요하지 않게 되어 스위칭 비용이 사라지게 되는 것이다.

3-4 Virtual TLB(translation lookaside buffer)

KVM 초기버전에서 사용되었던 Shadow Paging 기법에서는 가상머신들 간의 스케쥴링이 발생할 때 마다 CPU의 TLB를 비워줘야만 해서 성능을 크게 떨어뜨려다. TLB를 매번 비우지 않기 위해 가상머신마다 VPID (Virtual Processor Identification)을 기준으로 TLB를 개별적으로 관리한다. 물론 TLB의 크기가 더 커져야 한다는 문제가 발생하지만 TLB를 비우지 않게 되어 성능적인 측면에서는 이득을 얻을 수 있게 되었다.

3-5 비교 분석

표를 보면 KVM은 XEN과 같이 커널의 수정이나 재부팅을 하지 않고도 모듈을 올리는 것만으로 바로 시스템에 적용이 가능하다는 장점을 알 수 있다. 또한 CPU에서 지원하고 있는 가상화 시스템의 메모리 관리 기법을 사용하고 있어 성능 면에서 우월함을 알 수 있다.

<표 1> XEN과 KVM의 비교 평가

항목 \ 종류	XEN	KVM
가상화 방식	반가상화	전가상화
Guest 커널	수정된 커널 필요	수정 필요 없음
실행 조건	수정된 커널로 부팅 과정이 필요함	부팅 없이 바로 실행 가능
페이징 방법	Shadow Paging	EPT, NPT
기타	리눅스 스케줄러, 메모리 관리 기법 이용	독립적인 스케줄러, 메모리 관리 기법 사용

4. 결론 및 향후과제

리눅스 모듈 형태로 동작되는 KVM은 많은 부분에서 리눅스 커널 코드를 사용하고 있다. 그리고 INTEL VT-x와 AMD-V를 이용한 전가상화 방식으로 개발되어 성능적인 향상이 있음이 분석되었다. 논문의 분석 결과는 향후 가상머신의 상태 및 자원 모니터링을 위한 접근 라이브러리를 구현하고자 할 때 기초 자료로서 사용될 것이다.

참고문헌

[1] David Chisnall "The Definitive Guide to the Xen Hypervisor" Prentice Hall

[2] K. Avi, et al, "kvm: the Linux Virtual Machine Monitor", Procesdins of the Linux Symposium, Jun, 2007

[3] M. Tim Jones, "리눅스 커널 가상 기계 탐험", IBM developerworks, may, 2008

[4] Y. Sheng, "Extending KVM with new INTEL Virtualizaion technology", KVM Fourm 2008, Jun, 2008