6

보안과 신뢰성있는 컴퓨팅을 위한 가상화 기술

국민대학교 ▌홍대영 · 고원석 · 임성수*

1. 서 론

가상화(Virtualization)는 컴퓨터 환경에서 컴퓨터 자원의 추상화를 일컫는 범용 용어로서 실제 존재하는 물리적 자원들의 특징을 숨기고 논리적인 자원의 집합을 구성하여 이러한 자원과 상호작용하는 다른 시스템, 응용 프로그램, 사용자에게 제공하는 기술로 정의할 수 있다. 이는 서버, 운영체제, 응용 프로그램, 저장장치 등과 같은 물리적인 자원을 여러 개의 논리적인 자원으로 보이도록 할 수 있으며 여러 개의 물리적인 자원을 하나의 논리적인 자원으로 나타낼 수도 있다. 그림 1과 같이 가상화 기술은 이들 논리적 자원들과 실제 물리적 자원들에 대한 연결을 담당해 줌으로써, 가상화 자원을 이용하는 사용자는 더 이상 어떤 자원들이 사용되는지를 구체적으로 알 필요가 없어지게 된다.

가상화에 대한 연구는 1960년대부터 시작되었으며 다양한 컴퓨팅 영역에 적용되어 왔다. 모든 가상화 기술의 공통 주제는 캡슐화(Encapsulation)를 통하여 기술적으로 자세한 부분을 숨기는 것이며 이를 위한 외부 인터페이스를 지원한다. 가상화는 주어진 하드웨어 플랫폼 위에서 제어 프로그램인 가상 머신 모니터(VMM) 혹은 하이퍼바이저(hypervisor)를 통해 실행된다. 가상머신 모니터는 애플리케이션과 서비스를 실제적인 자원들과 분리하는 역할을 하며 사용자로 하여금 동일한 자원을 공유하게 해주고, 여러 자원들을 개별 자원이라기 보다는 논리적인 자원 풀로서 사용하고 다루게 해준다. 즉, VMM은 위에서 동작하는 게스트 소프트웨어(Guest Software)에 대하여 가상 기계(Virtual machine)라는 시뮬레이션된 컴퓨팅 환경을 제공한다. 게스트 소프트웨어는 대부분 완전한 운영체제 형태로 구성되며, 독립된 하드웨어 플랫폼에 설치된 것처럼 실행된다.

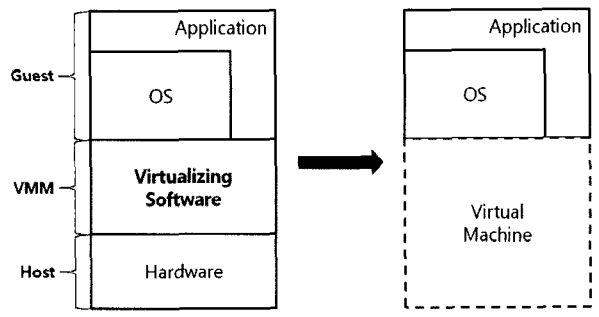

그림 1 가상화 기술의 정의

가상화 기술은 하나 혹은 여러 개의 물리적 시스템에서 실제 물리적 시스템과 독립적으로 복수 개의 논리적 시스템을 제공하는 기술로서 초기에 대규모 시스템을 개별 논리적 독립 시스템으로 보이게 하여 시스템의 개별화를 목적으로 사용된 이후 최근에는 시스템의 신뢰성, 안정성, 확장성을 보장하기 위해 가상화 기술이 활용되기 시작하였다. 본 글에서는 보안 분야와 신뢰성 있는 컴퓨팅 분야에서 가상화 기술의 필요성을 논의하고 특히, 임베디드 시스템 분야에서 가상화 기술 적용 방법으로 주목받는 마이크로 커널을 이용한 가상화 기술에 대해 파악한다.

2. 배 경

가상화 기술은 가상화를 적용하는 대상 자원에 따라 CPU 가상화, 메모리 가상화, I/O 가상화로 구분한다. 이 중에서 가장 중요한 가상화 대상 자원은 CPU이다. CPU의 가상화는 다음과 같이 전가상화, 부분 가상화, 그리고 하드웨어의 도움에 의한 가상화로 구분한다[8].

1) 전가상화(Full Virtualization): 전가상화는 게스트 OS를 가상화 계층에 의해 밑에 있는 하드웨어로부터 완전히 추상화시키는 기법이다. 게스트 OS는 수정되지 않으며 자신이 가상화되었는지 인지하지 못한다. 전가상화는 사용자 수준에서 실행되는 게스트 OS 내에서 시스템의 중요한 자원을 접근하는 특권 모드(privileged mode) 명령어들이 실행될 때 트랩(trap)이 발생하

* 종신회원

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학IT연구센터 지원사업의 연구결과로 수행되었음(IITA-2008-C1090-0701-0045)

도록 하는 하드웨어의 지원을 받아 VMM이 트랩 핸들러를 통해 대신 실행하는 방법으로 구현된다. 그러나, 최근 많이 사용되는 프로세서에는 이러한 트랩 기능이 구현되어 있지 않아서 사용자 수준에서 특권 모드의 명령어를 실행해도 시스템의 특권 모드를 진입할 수 없어 시스템의 중요한 자원을 접근할 수 없다. 즉, 게스트 OS 하위 수준의 VMM 및 하드웨어는 게스트 OS 내에서 특권 모드 명령어가 실행되는 지 전혀 파악할 수 없다. 이러한 최근 프로세서에서 전가상화를 구현하기 위해서 이진 변환(binary translation) 기법을 활용한 가상화 기술이 개발되어 VMWare 등의 가상화 소프트웨어에서 활용하고 있다. 이진 변환 기법이란 VMM에 의해 특권 모드 진입이 필요한 게스트 OS의 명령어 순차들이 동적으로 모두 비 특권 모드의 명령어들로 변환되고 특권 모드로서 시스템 자원 접근이 필요한 부분은 VMM의 특정 함수를 호출하는 방법으로 트랩 기능이 없는 프로세서 상에서 전가상화를 구현하는 방법이다.

프로세서의 트랩 기능을 활용한 전가상화 기술이나 이진 변환 기법을 활용한 전가상화 기술 모두 가상 기계들에게 최적의 고립화와 보안 수준을 제공한다. 이는 동일한 게스트 OS의 구체적 실행 형태가 가상화된 하드웨어와 기존의 물리적 하드웨어 모두에서 실행될 수 있기 때문이다. 대표적인 전가상화 소프트웨어로는 VMWare Workstation, Virtual PC, VirtualBox 등이 있다.

2) 반가상화(Paravirtualiztion): 최근의 전가상화 기법은 동적인 이진 변환을 활용하기 때문에 물리적 하드웨어 상에서 직접 실행되는 방법에 비해 성능과 효율성이 떨어진다. 반가상화는 이러한 전가상화의 성능상의 단점을 극복하기 위해 동적인 이진 변환 기법 대신 특권 모드로서의 실행이 필요한 명령어들을 사전에 하위 VMM 내의 함수를 호출하는 하이퍼콜(hypercall)로 변환한 후 가상 기계 위에서 실행하는 방법이다. 반가상화는 가상화될 수 없는 커널 명령어들을 하이퍼콜(hypercall)로 대체하기 위하여 OS 커널을 수정한다. 하이퍼콜은 하이퍼바이저와 직접적으로 통신하는 인터페이스로서 메모리 관리, 인터럽트 핸들링 등과 같은 중요한 커널 연산들을 게스트 OS가 아닌 하이퍼바이저가 실행하도록 인터페이스를 제공한다. 반가상화는 일반적으로 전가상화보다 나은 성능을 보이지만 가상화하기 위해 사전에 게스트 OS를 수정해야 하는 추가 비용이 생긴다. 이로 인해 전가상화에 비해 호환성과 이식성은 떨어진다. 대표적인 반가상화 소프트웨어로는 오픈 소스 프로젝트로 진행된 Xen 프로젝트가 있다. Xen은 수정된 리눅스 커널을 이용하여 프로세서와 메모리를 가상화하고 게스트 OS 디바이스 드라이버를 이용하여 I/O를 가상화하였다.

3) 하드웨어 지원 가상화(Hardware Assisted Virtualization): 하드웨어 제조사들은 급속히 가상화를 채택하고 있고 가상화 기술을 간단화하기 위한 새로운 특징들을 개발하고 있다. 하드웨어 지원 가상화의 대표적인 예로는 Intel사의 Intel Virtualization Technology (VT-x)와 AMD사의 AMD-V가 있다. Intel의 VT-x와 AMD-V는 기존의 특권 모드 아래에 추가적인 새로운 특권 모드로 동작하는 VMM에 대한 인터페이스를 제공하며 게스트 OS의 특권 명령은 이러한 새로운 CPU 실행 특성을 통해 실행된다. 하드웨어 지원 가상화는 하드웨어 지원을 통해 전가상화의 이진 변환(binary translation)이나 반가상화의 게스트 OS 수정 없이 VMM이 게스트 OS의 특권 명령을 자동으로 트랩할 수 있게 한다. 또한 게스트 머신의 상태는 VT-x의 Virtual Machine Control Structures나 AMD-V의 Virtual Machine Control Blocks 안에 저장된다. 2006년부터 Intel VT나AMD-V를 지원하는 프로세서가 등장하였으며 최근에 출시되는 프로세서들에는 이러한 하드웨어 지원 특성이 포함되어 있다.

둘째, 메모리 가상화는 현대의 운영체제에서 지원되는 가상 메모리 시스템과 매우 유사하다. 가상 메모리 시스템을 지원하는 운영체제는 가상 메모리와 물리 메모리간의 사상 정보를 가지고 있다. 단일 시스템 상에서 여러 개의 가상 기계를 실행시키면서 게스트 OS에 가상 메모리를 지원하기 위하여 MMU를 가상화하여야 한다. 게스트 OS는 가상 주소를 논리적인 가상 기계 내의 물리 주소에 맵핑하는 것을 제어하지만 게스트 OS는 실제 물리 주소에 직접 접근을 할 수 없다. 따라서, VMM은 실제 물리 주소와 가상 기계의 물리 주소 사이의 연관을 책임져야 하며 이러한 연관 관계는 쉐도우 페이지 테이블이라는 자료 구조를 통해 관리된다.

마지막으로 I/O 가상화는 가상 장치와 공유된 물리 하드웨어 간의 I/O 요청의 전달을 관리하는 것을 포함한다. 하이퍼바이저는 물리적인 하드웨어를 가상화하고 각 가상 장치의 표준화된 집합을 가지는 가상 기계를 나타낸다. 이러한 가상 장치는 잘 알려진 하드웨어를 효율적으로 에뮬레이션하고 가상 기계의 요청을 시스템 하드웨어에 맞게 변환한다.

가상화 기술은 VMM이 어떻게 구현되는가에 따라 호스트 운영체제 기반 가상화 기술과 호스트 운영체제가 없는 가상화 기술로 분류한다. 호스트 운영체제

기반 가상화 기술은 이미 시스템에 운용되고 있는 호스트 운영체제가 있고 그 위에 하이퍼바이저가 실행되어 가상화를 이루는 기술이며. 호스트 운영체제가 없는 가상화 기술은 하드웨어 상에 운영체제 없이 하이퍼바이저 역할을 하는 소프트웨어 계층만 존재하는 가상화 기술이다. 데스크탑 컴퓨터나 서버 컴퓨터 상에서는 주로 호스트 운영체제 상에서 구현하는 하이퍼바이저에 의한 가상화 기술을 적용하며 임베디드 실시간 시스템에서는 호스트 운영체제가 없이 실시간 처리 성능을 보장하는 작은 크기의 하이퍼바이저에 의한 가상화 기술을 적용한다.

3. 가상화 기술과 보안 문제

가상화 기술이 보안 문제와 관련한 기술로 대두된 것은 2006년 Subvirt[9]와 Bluepill[10]이라고 하는 루트킷(rootkit)이 발표된 이후이다. 루트킷이라 함은 시스템의 수퍼유저 권한을 정상적이지 않은 방법으로 취득하는 도구 및 방법을 말하는 것으로 모든 바이러스 및 해킹의 근본적인 접근 방법이다. 루트킷에서 수퍼유저의 권한을 취득하여 시스템에 해를 끼치는 소프트웨어 요소를 멀웨어(Malware)라고 부른다. 이러한 멀웨어(Malware)를 은폐하기 위하여 가상화 기술을 사용하는 사례가 발표되면서 가상화 기술을 이용한 보안 문제가 주목을 끌기 시작했다. 멀웨어는 시스템 구조에서 호스트 OS로 실행 중인 운영체제 아래쪽에 하이퍼바이저를 삽입하여 호스트 OS를 게스트 OS로 전환시키는 기술을 사용하며 이에 대한 그림은 그림 2와 같다. 그림 2와 같은 구조를 지닌 루트킷(Rootkit)이 가지는 장점으로 여러 가지가 있는데 가장 큰 장점은 멀웨어의 서비스를 대상 OS에 적용하기 위하여 대상 OS, 대상 응용 소프트웨어를 수정할 필요가 없다는 것이다. 또한, 멀웨어가 대상 OS 밑에서 동작하기 때문에 대상 OS 내에서 동작하는 보안 프로그램이 멀웨어의 존재를 알기 어렵다.

이러한 가상화 기술을 이용한 루트킷의 대표적인 예로는 SubVirt와 BluePill이 있다. SubVirt는 Microsoft Research와 University of Michigan이 공동 연구하여 개발한 소프트웨어 기반 가상화를 이용한 루트킷이다. SubVirt는 시스템의 저장장치에 루트킷의 실행 이미지를 저장하고 부팅 절차(Booting Sequence)를 변경하여 다음 부팅부터 SubVirt가 대상 OS를 제어하는 방식으로 설치된다. 이와 같은 방식은 오프라인이나 다른 미디어를 통한 부팅시 루트킷의 존재를 감지할 수 있다. 또한 SubVirt는 VMWare, Virtual PC와 같은 상용 VMM에 기반한 기술이다. 이와 같은 상용 VMM은 제공하는 기능이 많지만 하드웨어 에뮬레이션 오버헤드가 존재하기 때문에 대상 OS 내의 디텍터가 VMM의 존재를 감지할 수 있다.

BluePill은 SubVirt의 단점을 보완하여 Invisible Things Lab에서 개발한 하드웨어 기반 가상화를 이용한 루트킷이다. BluePill은 AMD-V 기술을 이용하여 시스템의 작동 중에 시스템 구조를 변화시킨다. 이는 SubVirt와 달리 재부팅(Reboot)이나 바이오스(BIOS) 또는 부트 섹터(Boot Sector)의 변경을 수행하지 않고서 루트킷을 설치할 수 있다. 저장 장치에 루트킷 이미지를 저장하지 않았기 때문에 오프라인으로 루트킷을 존재를 감지할 수 없다. 또한 BluePill은 매우 간단한 하이퍼바이저를 사용하므로 모든 하드웨어는 성능 저하 없이 접근이 가능하다. 이와 같은 이유로 대상 OS 내의 루트킷을 감지하기 위한 소프트웨어가 루트킷의 존재를 감지할 수 없다. BluePill의 단점으로는 하드웨어 지원 가상화를 지원하지 않는 플랫폼에는 설치가 불가능하고 현재 AMD-V만을 지원한다.

일반적인 멀웨어는 사용자 모드에서 동작하는 멀웨어와 커널 모드에서 동작하는 멀웨어로 구분된다. 사용자 모드 멀웨어는 다양한 프로그램 언어로 개발할 수 있기 때문에 구현이 용이하다. 또한 제공되는 라이브러리와 OS 수준 자원 접근을 통하여 강력한 기능을 수행하는 멀웨어를 제작할 수 있다. 하지만 사용자 모드 멀웨어는 OS에 의해 감지되기 쉽다. 반면에 커널 모드 멀웨어는 OS에서 사용하는 데이터 구조체의 내용을 변경하여 동작한다. 이는 OS 내의 감지기에 의해 감지되기는 어렵지만 수행할 수 있는 기능은 제한적이고 OS의 수정이 필요하다. SubVirt와 BluePill과 같은 가상화 기술을 이용한 보안 침입 기술은 사용자 모드 멀웨어와 커널 모드 멀웨어의 장점을 모두 가진다. 이러한 기술에서 멀웨어는 호스트 OS에서 동작하는 사용자 어플리케이션의 형태로 제작되기 때문에 사용자 모드 멀웨어처럼 구현의 용이하며 강력한 기능을 가진다. 또한 멀웨어는 게스트 OS 밑의 계층에 존재하는 VMM에 의해 게스트 OS로 서비스되기 때문에 게스트 OS 내의 감지기가 멀웨어를 감지할 수 없을 뿐 아니라 게스트 OS에 멀웨어 서비스를 적용하기 위하여 게스트 OS를 수정할 필요가 없다.

현재 연구된 가상화를 이용한 보안 기술들은 가상화를 이용한 공격, 가상화를 이용한 보안 향상, 가상 기계가 설치된 것을 탐지하는 분야, 하드웨어와 운영체제 사이에 특정 소프트웨어 계층을 삽입하는 분야

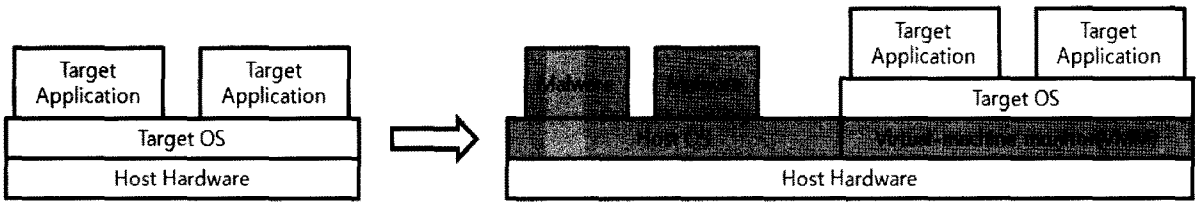

그림 2 하이퍼바이저를 이용한 보안 침입 기술

에서 다양하게 응용될 수 있다. 예를 들어 하드웨어와 운영체제 사이에 삽입되어 시스템의 보안을 위협하는 가상화된 소프트웨어 계층이 설치되었다면, 가상기계가 설치된 것을 탐지하는 기술을 이용하여 시스템을 통해 멀웨어가 수행되는 것을 방지할 수 있어야 한다. 가상화 기술을 이용한 보안 기술들은 보안 소프트웨어가 동작하는 소프트웨어 계층이 멀웨어가 실행될 수 있도록 시스템을 제어하는 루트킷의 하부에서 동작하느냐, 상부에서 동작하느냐에 따라서 구분할 수 있다.

첫 번째로, 루트킷을 탐지하기 위한 가장 좋은 형태는 루트킷의 동작 여부를 탐지하기 위한 보안 소프트웨어가 루트킷의 하위 계층에서 동작하는 것이다. 이 경우에는 보안 소프트웨어가 루트킷에 종속되어 수행되지 않기 때문에, 루트킷의 존재의 유무를 감지할 수 있게 된다. 루트킷의 존재를 확인하기 위해서 루트킷이 동작하는 중에 나타나는 증상들을 메모리나 디스크 등에서 찾아낼 수 있다. 루트킷이 기존의 운영체제보다 먼저 수행되도록 부팅 과정이 변경되는 경우가 그러한 예이다. 루트킷 하위 계층에서 루트킷의 동작 여부를 탐지하기 위해서 사용되는 다른 방법들로는 안전한 외부 장치(USB, CD-ROM)를 사용하여 그 안의 부트코드가 루트킷이 수행되기 전에 먼저 수행되고, 따라서 그 후에 루트킷이 수행되는 것을 확인할 수 있도록 하는 기술들이 있고, 운영체제 하부에 시스템 전체를 제어할 수 있는 VMM을 두어, 보안 침입을 위해 루트킷이 삽입되더라도, VMM과 운영체제 사이에 삽입되어, 여전히 시스템이 안전하게 동작할 수 있게 하는 방법도 있다.

두 번째로 루트킷을 탐지 하기 위한 기술은 보안 소프트웨어가 루트킷의 상부에서 동작하는 형태이다. 이러한 경우에는 보안 소프트웨어의 수행이 루트킷에 종속적이므로 보안 소프트웨어가 탐지한 결과를 루트킷이 조작할 수 있고, 보안 소프트웨어가 침입 흔적을 스캔(scan)하기 위해서 접근 할 수 있는 영역도 루트킷이 사용할 수 있는 메모리 영역에 비해 한정적이므로, 탐지가 불가능하지만, CPU 수행시간, 메모리 사용양과 같은 자원의 활용 상태를 분석하여 이를 탐지 해내기도 한다. 예를 들어 특정 벤치마크 셋의 루트킷이 설치되기 전의 정상적인 상태에서의 수행시간과, 루트킷이 설치되어 동작하기 시작한 후의 수행시간에 차이가 있다면, 이를 루트킷이 설치되었다고 판단할 수 있는 근거자료로 사용할 수 있다.

4. 가상화 기술과 신뢰성 있는 컴퓨팅

4.1 신뢰성 있는 컴퓨팅

컴퓨터 시스템의 신뢰 수준에 대한 기준은 여러 가지 이론적, 실제적인 방법이 있다. 본 글에서는 일부 컴퓨터 시스템 내부의 오류가 전체 시스템의 동작에 영향을 미치는 수준으로 신뢰 수준의 기준을 삼는다고 가정한다. 즉, 운영체제 내의 작은 오류가 전체 시스템을 멈추거나 오동작을 하게 만들 확률을 신뢰 수준의 척도로 삼는다. 최근 많이 사용되는 일반 운영체제의 예에서 보면, 운영체제 내의 디바이스 드라이버 내의 작은 오류가 전체 시스템 오류로 파급될 확률이 크기 때문에 일반 운영체제를 기반으로 한 시스템은 전반적으로 신뢰 수준이 낮다고 본다.

컴퓨터 시스템의 신뢰 수준을 가늠하기 위해 TCB (Trusted Computing Base)[5-7]라는 단위를 사용한다. TCB는 컴퓨터 시스템을 이루는 요소로서 이 요소의 안정성과 신뢰성을 보장하는 경우 전체 시스템의 안정성 및 신뢰성을 보장할 수 있다는 단위 요소를 말한다. TCB는 컴퓨터 시스템 전체에 가장 큰 영향을 주는 단위 요소이기 때문에 통상적으로 운영체제 커널을 가리킨다. 즉, 운영체제 커널의 안정성 및 신뢰성을 보장할 수 있는 경우 전체 시스템의 안정성 및 신뢰성을 보장할 수 있다. 일반 운영체제를 사용하는 경우에는 일반 운영체제 커널 전체가 TCB가 된다. 따라서, TCB의 안정성 및 신뢰성을 보장하기 위해서는 운영체제 커널 전체의 안정성 및 신뢰성을 보장해야 하는데 코드의 크기 및 메모리 사용량이 큰 현대 운영체제 커널 전체의 안정성 및 신뢰성을 100% 보장한다는 것은 상당히 어려운 일이다. 결국 전체 시스템의 안정성 및 신뢰성을 보장하기 위해서는 될 수 있으면 작은 크기의 TCB로 이루어진 시스템을 구성해야 한다는 논리가 만들어진다[7].

작은 TCB로 이루어진 시스템을 구성하기 위해서는 일반 운영체제 커널을 사용해서는 불가능하므로 마이크로커널을 사용해야 한다는 결론이 도출된다. 따라서, 안정성 및 신뢰성을 보장하기 위한 컴퓨터 시스템을 마이크로커널로 구축하는 연구가 활발하게 진행중이다. 최근에는 호주의 NICTA에서 개발중인 마이크로커널인 L4를 중심으로 안정성 및 신뢰성을 보장하는 가상화 기반 시스템 구축에 대한 연구 및 상용화 결과가 주목을 받고 있다[2].

4.2 신뢰성 있는 컴퓨팅을 위한 마이크로커널의 예: L4

마이크로커널은 그 이름에서부터 알 수 있듯이 커널을 경량화 하여 커널의 복잡성을 줄인 커널을 말한다. 그 중에서 L4 마이크로커널[3,4]은 프로세스끼리의 통신을 위한 IPC(Inter Process Communication), 주소 공간(Address Space), 쓰레드(Thread) 그리고 UID (Unique Identification) 만을 커널에서 제공해 주기 때문에 그 커널의 크기 또한 다른 커널에 비해 매우 작으며 복잡하지 않다. 1세대 마이크로커널과 L4 마이크로커널의 가장 큰 차이점은 처리 성능이다. 이는 마이크로커널의 구조 특성상 IPC의 성능이 전체 시스템의 성능을 좌우하게 되는데 L4 마이크로커널의 IPC 성능은 이전 마이크로커널에 비해 매우 많이 향상되었기 때문에 더 좋은 성능을 보이고 있다. 최근 발표된 OKL4의 커널에서 구동하는 Wombat 리눅스[1]는 일반 임베디드 리눅스와 그 성능의 차이가 거의 없거나 주소공간을 공유하는 프로세스 사이의 문맥교환시 일반 리눅스보다 더 좋은 결과를 보이고 있다고 한다. L4는 커널의 이름이 아니라 커널의 특징을 나타낸 명세에 가깝다고 한다. 이 L4 커널을 구현하기 위해 여러 연구 단체들이 L4 커널을 구현 하였는데 처음으로 만들어진 L4 커널은 L4Ka팀의 Pistachio였다. 이후에 NICTA에서는 임베디드 환경에서 구동하기 위한 NICTA: Pistachio-embedded를 개발하였고 최근에는 Open Kernel Labs를 통해 OKL4라는 이름으로 배포되었다. OKL4는 L4Ka팀의 Pistachio L4 커널과 Linux나 WinCE와 같은 범용 운영체제를 L4위에 구동 할 수 있도록 도와주는 Iguana 서버를 제공한다.

Iguana는 Guest OS를 지원하기 위한 L4의 확장 라이브러리이다. Iguana는 단일 주소 공간만을 사용하기 때문에 별도의 메모리 관리 메카니즘이 필요하다. Iguana는 PD(Protection Domain)를 통해 메모리를 보호 한다. 그리고 L4 API를 통해 Iguana를 기반으로 하는 Iguana 쓰레드를 만들 수 있으며 각각의 쓰레드

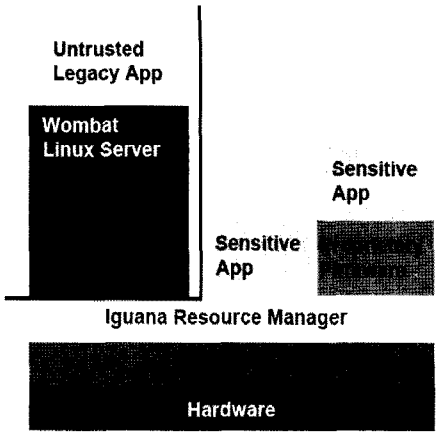

그림 3 L4 마이크로커널을 기반으로 한 시스템 구조

는 메모리 접근을 위한 Capability를 가지고 있다. 또한 쉽게 드라이버 인터페이스를 만들기 위한 IDL4 컴파일러를 지원한다. Wombat의 라이선스는 GPL을 따르지만 L4와 Iguana는 BSD 라이선스를 따르기 때문에 상업용 모듈을 만들 경우 Iguana로 만들면 좀 더 유리하다. 그림 3은 L4 마이크로 커널을 기반으로 한 시스템 구성의 예를 보이고 있다.

L4 마이크로 커널의 핵심적인 특징 중 가상화 기술과 관련 있는 특징으로는 재귀적으로 구성되는 시스템을 구축할 수 있다는 데에 있다. 재귀적으로 구성된다는 것은 가장 하위 루트 태스크의 주소 공간 및 메모리 관리 태스크를 기준으로 하여 상위에 생성되는 모든 태스크들이 각각 자신의 주소 공간 및 메모리 관리 태스크, 그리고 예외 상황 처리기를 가질 수 있다는 것이다. 즉, 상위 태스크들은 모두 자신만의 주소 공간, 메모리 관리기, 예외 상황 처리기를 갖추고 독립적으로 동작하는 하나의 시스템으로 인식되어 수행된다는 것이다. 그림 4는 L4 마이크로 커널을 기반으로 한 시스템의 재귀적 시스템 구성을 보인다. 그림에서 보이는 바와 같이 초기 주소 공간에서 각 응용 태스크를 위한 개별 주소 공간 및 메모리 관리기(pager)가 재귀적으로 생성된다.

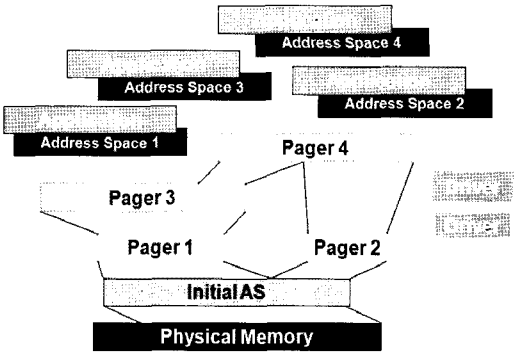

그림 4 L4 마이크로커널의 재귀적 구성 특성

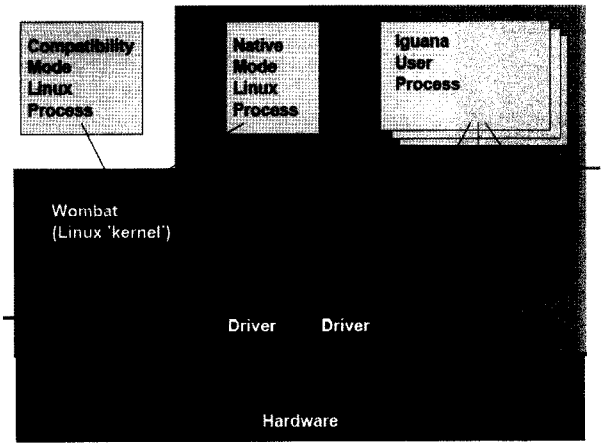

그림 5 L4 마이크로커널을 기반으로 한 리눅스 가상화

이를 가상화의 관점에서 설명하면, 최하위 주소 공간을 기반으로 하여 VMM 내지 하이퍼바이저가 실행되며 상위 태스크로 각 개별 가상 기계들이 실행될 수 있도록 시스템 구성을 할 수 있는 기능이 제공된다.

4.2 마이크로커널 기반 가상화 구축의 예

위의 그림은 L4 마이크로 커널을 기반으로 구축한 리눅스 운영체제의 가상화 사례를 보인다. 가상화된 리눅스인 Wombat의 주소체계를 보면, L4 위의 공간은 Iguana에서 관리하는 주소 공간의 영역이며 그 안의 작은 사각형 공간은 보호 도메인을 나타낸다. 왼쪽 위 사각형은 Iguana의 주소 공간과 분리된 주소 공간을 나타낸다. Wombat은 Iguana의 공유 주소 공간에서 보호 도메인으로 분리가 되어 동작하는 리눅스 서버다. 이러한 Iguana 주소 공간에서 동작하므로 빠른 문맥 교환과 데이터 공유 지원을 받을 수 있다. MyOS Server라 명칭이 되어 있는 Iguana서버는 Iguana에서 기본적인 서비스를 지원해주기 위한 추가적인 서비스를 지원한다. 이 영역은 드라이버와 함께 시스템의 TCB 영역에 존재한다. Iguana 유저 프로세스는 시스템의 기본적인 운영체제 서비스와 다른 응용프로그램 서버들이다.

주소 공간의 가상화와 더불어 가상화 시스템을 구축하기 위해 필수적인 요소는 I/O 가상화이다. 하드웨어에 장착된 I/O 장치들에 대해 각 가상 기계가 통일된 인터페이스를 통해 접근할 수 있어야 하며 이러한 접근을 통해 각 가상 기계에서 요구하는 성능 조건을 최대한 만족시킬 수 있어야 한다. I/O 가상화와 관련하여 L4 마이크로 커널 및 Iguana 기반 시스템에서는 통일된 디바이스 드라이버 인터페이스를 제공한다.

Iguana의 인터페이스는 IDL4 컴파일러를 통해 생성되는 Stub 코드를 이용한다. 생성된 Stub 코드는 서비스 하려는 함수의 프로토타입을 선언하고 사용자 쪽

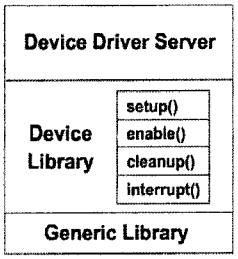

그림 6 Iguana 디바이스 드라이버 구조

에서 간단히 호출하는 정도로 쉽게 Iguana 서버를 구현 할 수 있도록 도와준다. Iguana 디바이스 드라이버는 그림 6과 같이 표준 라이브러리와 디바이스 드라이버의 기능을 하는 디바이스 라이브러리 그리고 Iguana 서버로 동작을 할 수 있도록 디바이스 서버 라이브러리로 구성이 된다.

그림 7은 OKL4 v1.4.1에 포함되어있는 Wombat 시리얼 드라이버의 전체 구성을 나타낸 것이다. 서버와 클라이언트는 IPC를 이용하여 데이터를 주고받는다. Iguana에서 제공하는 인터페이스를 통해 I/O 가상화를 위한 API를 제공하고 있다.

L4 마이크로 커널을 사용한 가상화 시스템 구축 시 I/O 가상화의 구현을 위해서는 몇 가지 이슈를 해결해야 한다. 가상화된 리눅스 커널인 Wombat의 디바이스 드라이버가 직접 장치를 제어하기 위해서는 하드웨어 메모리 접근이 가능해야 한다. 그렇기 때문에 주소공간을 관리하는 Sigma0를 통해 해당 장치의 레지스터 영역을 할당을 받아 드라이버를 통해 장치를 직접 제어한다. 그러나, 사용자 수준의 가상화된 계층에서 직접 하드웨어 레지스터를 접근하는 것은 I/O 가상화 관점에서 맞지 않다. 따라서, 이러한 하드웨어 접근을 위한 통일된 API를 각 가상 기계에 마련해 주고 해당 API를 통해서만 하드웨어 자원 접근이 가

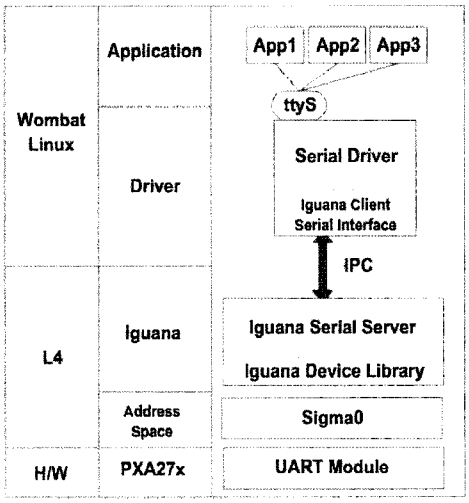

그림 7 L4 IPC를 이용한 UART 드라이버

능하도록 한다. 두 번째 이슈는 인터럽트 문제이다. 모든 인터럽트의 감지는 L4 커널에 의해 이루어지고 감지된 인터럽트 처리는 인터럽트 테이블에 등록된 Iguana 쓰레드에서 수행한다. 즉, 모든 하드웨어 인터럽트는 L4에 의해 처리되고 해당 인터럽트 처리 결과가 필요한 가상 기계에 결과를 전달해주기 위한 인터페이스가 제공된다. 마지막으로 해결해야 하는 문제는 DMA이다. Wombat의 디바이스 드라이버가 DMA를 사용할 경우 해당 사용되는 주소가 올바른 물리 주소이어야 한다. 따라서, L4의 초기 주소공간에서 두 번 주소 변환이 이루어지는 동안에도 올바른 DMA용 물리주소가 전달될 수 있도록 주소 변환에 주의를 기울여야 한다. 이렇게 I/O 가상화를 통해 구현하는 디바이스 드라이버와 기존 방식의 디바이스 드라이버의 성능을 비교하면 직접적으로 하드웨어 자원을 접근하는 기존 디바이스 드라이버가 다소 우수한 성능을 보인다. 이러한 성능 차이를 줄이는 것이 가상화된 시스템의 활용도를 높이는 중요한 요소가 될 것이다. 그림 8은 성능평가를 위해 구성한 L4 IPC를 이용한 디바이스 드라이버의 모습니다. 이렇게 작성한 가상화된 디바이스 드라이버와 리눅스 커널에서 직접 하드웨어 자원을 접근하는 디바이스 드라이버의 성능 평가를 수행한 결과는 그림 8에 보인다.

아래의 그림 9는 Iguana와 Wombat에서 특정 메모리 영역에 특정 값을 쓰는 드라이버를 제작하여 그

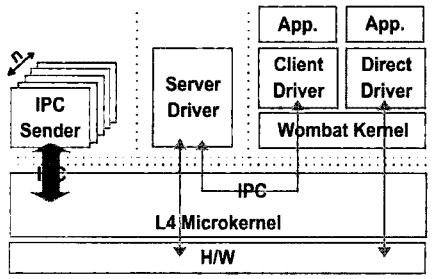

그림 8 L4 IPC를 이용한 디바이스 드라이버

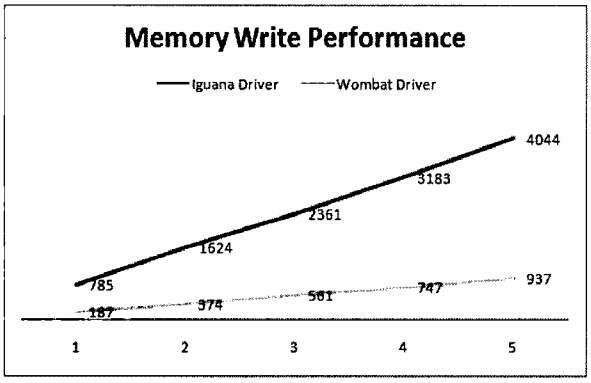

그림 9 Iguana/Wombat 디바이스 드라이버 성능 비교

성능을 비교한 그림이다. 그림의 가로축은 쓰려는 메모리의 크기(MB)를 나타내며 세로축은 전체를 쓰는데 소요된 시간을 나타낸다. 그림에서와 같이 Wombat 디바이스 드라이버는 IPC를 통해 전송을 할 필요가 없으므로 Iguana 드라이버보다 빠른 성능을 보인다. 그리고 메모리의 쓰는 시간은 그 크기에 비례하기 때문에 그림처럼 소요 시간이 증가한다.

5. 결 론

본 글에서는 가상화 기술과 보안 문제와의 연관성 및 가상화 기술을 활용한 신뢰성있는 시스템 구축 방법 및 사례에 대해 논의하였다. 가상화 기술을 고신뢰성 및 실시간 성능이 필요한 시스템에 적용하는 경우 아직 해결되지 않은 이슈들이 존재한다. 가장 큰 이슈는 실시간성에 대한 이슈이다. 즉, 현재까지 가상화 기술이 자원의 효율적인 활용 및 신뢰성있는 컴퓨팅을 위한 기술에 초점이 맞추어져 있었다면 이를 실제 산업 현장을 비롯한 응용 분야에 활용하려면 실시간 성능에 대한 고려가 필요하다. 즉, 가상 기계의 각 자원 접근 API를 사용할 때의 실시간 성능 기준이 필요하고 가상 기계 사이의 전환의 경우에도 실시간 성능 자료가 필요하다. 이러한 가상화 기술을 적용한 시스템에서의 실시간 성능에 대해서는 아직 연구된 바가 없으며 향후 가상화 기술을 임베디드 실시간 시스템에 적용해야 하는 상황에서는 정밀한 분석 도구 및 확정정(deterministic) 성능 지표를 나타낼 수 있는 커널 내 기술이 필요하다.

본 글에서의 보안 문제와 신뢰성있는 컴퓨팅이 가상화 기술과 가지는 지 논의하였다. 연관성에 대한 논의 및 문제 제시가 실시간 성능이 필요한 시스템에서 가상화 기술을 활용하는 데 도움이 되는 연구 및 결과물 도출로 이어지기를 기대한다.

참고문헌

[1] Ben Leslie, Carl van Schaik and Gernot Heiser, "Wombat A portable User-Mode Linux for Embedded Systems", National ICT Australia, 2004.

[2] Peter Chubb, "Get More Device Drivers out of the Kernel!", National ICT Australia.

[3] Ihor Kuz, "L4 User Manual NICTA L4-embedded API", Embedded, Real-Time and Operating Systems Program, 2005.

[4] Gernot Heiser, "The OKL4 Microkernel, A High-Performance Virtualisation Platform", Open Kernel Labs, 2007.

[5] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, D. Boneh, "Terra: a virtual machine-based platform for trusted computing", ACM SIGOPS Operating Systems Review, pages 193-206, Vol. 37, Issues 5, Dec. 2003.

[6] B. Lampson, M. Abadi, M. Burrows and E. Wobber, "Authentication in Distributed Systems: Theory and Practice", ACM Transactions on Computer Systems, pages 265-310, Nov. 1992

[7] Heiser, Gernot; Elphinstone, Kevin; Kuz, Ihor; Klein, Gerwin, Petters, Stefan M. "Towards Trustworthy Computing Systems: Taking Microkernels to the Next Level", Operating Systems Review 41 (3): 3-11, July 2007.

[8] VMware, Inc., "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", White Paper, Nov. 2007.

[9] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch, "SubVirt: Implementing malware with virtual machines", Proceedings of the 2006 IEEE Symposium on Security and Privacy , May 2006.

[10] Bluepill Project, http://bluepillproject.org

홍 대 영

2001~2005 국민대학교 컴퓨터과학 학사
2005~2008 국민대학교 전산과학 석사
2008~현재 국민대학교 컴퓨터공학 박사
관심분야 : 임베디드 시스템, 이동통신 단말, 컴퓨터 구조, 실시간 시스템, 시스템 성능 평가, 저전력 시스템
E-mail : 14vicente@gmail.com

고 원 석

2001~2007 국민대학교 컴퓨터과학 학사
2007~현재 국민대학교 전산과학 석사
관심분야 : 임베디드 시스템, 마이크로 커널, 컴퓨터 구조, 실시간 시스템
E-mail : magicyaba@gmail.com

임 성 수

1993 서울대학교 컴퓨터공학 학사
1995 서울대학교 컴퓨터공학 석사
2002 서울대학교 전기컴퓨터공학 박사
2000~2004 팜팜테크㈜ 기술총괄이사
2004~현재 국민대학교 컴퓨터학부 조교수
관심분야 : 임베디드 시스템, 이동통단말, 컴퓨터 구조, 실시간 시스템신
E-mail : sslim@kookmin.ac.kr