Sorvisor - ARM 기반의 마이크로 하이퍼바이저

조영필*, Ali Almokhtar *, 백윤흥*
*서울대학교 전기정보공학부
e-mail : ypcho@sor.snu.ac.kr

Sorvisor - The ARM based Micro Hypervisor

Yeong-pil Cho*, Ali Almokhtar*, Yun-heung Paek*
*Dept. of Electrical and Computer Engineering, Seoul National University

요 약

최근 컴퓨터 시스템에 탑재되는 프로세서에는 하드웨어적으로 가상화를 지원하기 위하여 커널보다 상위의 권한을 가진 특별한 모드를 도입하고 있다. 그러나, 가상화를 요하지 않는 시스템이 대다수인 상황에서 해당 모드가 가진 특별한 권한은 시스템 로깅 및 보안에 유용하에 활용될 여지가 있다. 본 논문은 이를 위해 Sorvisor 라는 마이크로 하이퍼바이저를 개발함으로써 해당 모드를 활용할 수 있는 환경을 제공할 수 있도록 하고 있다.

1. 서론

현대 사회에서 사람들은 컴퓨터 시스템을 통해 다양한 작업을 수행하고 있다. 문서편집과 같은 각종 사무작업, 음악 및 게임과 같은 미디어 작업, SNS 와 채팅과 같은 커뮤니케이션 작업등이 대표적인 컴퓨터 시스템인 데스크탑과 스마트폰을 통해서 이루어지고 있다. 이들 시스템은 운영체제를 탑재하고 있다. 운영체제는 하드웨어의 자원을 효과적으로 분배 및 사용할 수 있는 환경을 제공함으로써, 응용프로그램에서 손쉽게 하드웨어 자원에 접근할 수 있도록 함과 동시에, 이들 응용프로그램들이 실행되는 시간을 적절히 분배 및 조절함으로써 사용자가 위에서 언급한 내용을 포함한 다양한 작업을 동시에 처리할 수 있도록 한다.

운영체제가 이러한 역할을 수행할 수 있는 까닭은 응용프로그램과는 다른 시스템 권한과 모드를 기반으로 동작하기 때문이다. 프로세서에는 시스템 자원을 효과적으로 사용할 수 있도록 하기 위해 다양한 모드를 제공하고 있으며, 시스템 자원을 안전하게 사용할 수 있도록 하기 위해 이들 모드에 차등적인 권한을 부여하고 있다. 따라서 응용프로그램은 프로세서의 모드 중 가장 기본적인 모드를 기반으로 동작하고, 운영체제는 이보다 높은 권한 과 기능을 가진 모드들을 독점적으로 활용함으로써, 응용프로그램에 방해받지 않고, 다수의 응용프로그램들에 효과적으로 하드웨어 자원을 분배할 수 있게 된다.

최근에는 클라우드를 비롯하한 거대 시스템의 등장과 홈 오피스와 모바일 오피스와 같이 업무 환경을 가정 및 개인으로까지 확장하는 BYOD (Bring Your Own Device) 서비스가 출현하면서 이들을 효과적이고 안전하게 제공하기 위해 하나의 컴퓨터 시스템 상에서 단지 하나의 운영체제를 돌리는 것이 아닌, 다른 성격을 가진 다수의 운영체제를 돌리는 모델이 도입되고 있으며, 이를 가상화라는 이름으로 부르고 있으며, 이를 제공하는 소프트웨어 플랫폼을 VMM (Virtual Machine Monitor) 혹은 하이퍼바이저 (Hypervisor) [1]라고 부른다.

과거에는 하이퍼바이저를 개발하기 위해 동적 코드 변환과 같은 다양한 소프트웨어 테크닉을 활용하였으나, 이는 필연적으로 많은 성능저하를 일으켰고, 가상화 기술의 확산에 큰 걸림돌이 되었다. 이에 따라 인텔 및 AMD 와 같은 프로세서 개발업체는 2007 년여 경부터 프로세서 수준에서 가상화를 위한 별도의 모드를 제공하고 있으며, 최근에는 ARM 에서도 Cortex-A/R 12 시리즈 이후부터 Hyp 모드라는 가상화 제공을 위한 별도의 모드를 추가하게 되었다.

이를 기반으로 하이퍼바이저에서 가상화를 위해 도입한 많은 기술이 소프트웨어가 아닌 하드웨어를 통해 제공할 수 있게 되었으며, 전반적인 성능의 증가와 더불어 안정성이 향상되는 효과가 있었다.

지금까지 기술을 내용을 정리하면, 프로세서에는 차등된 권한을 가진 다양한 모드가 존재하며, 각각 응용프로그램, 운영체제, 하이퍼바이저 순으로 높은 권한을 가진 모드 상에서 실행이 된다. 그리고, 운영체제는 응용프로그램들에게 하드웨어 자원을 제공하며, 하이퍼바이저는 운영체제들에게 하드웨어 자원을 제공하는 역할을 하게 된다.

그러나 아무리 클라우드의 영향력이 커지고 BYOD 가 확장된다고 하더라도, 하이퍼바이저는 가상화가 필요하지 않은 대다수의 시스템에서 운영체제에 비해 그 활용도가 떨어질 수 밖에 없다. 따라서 하이퍼바이저로 하여금 가상화가 아닌 보안, 시스템 로깅 등의 작업을 수행하도록 하는 움직임이 있다. 하이퍼바이저는 운영체제보다 높은 권한에서 실행되기 때문에, 이러한 작업을 수행할 때, 운영체제의 간섭에서 자유

롭고 안전한 환경을 제공할 수 있게 된다. 이러한 하이퍼바이저의 경우 가상화에 대한 기능을 제공하지 않아도 되기 때문에 매우 간소화 되는 경향이 있으며, 따라서 마이크로 하이퍼바이저 (μ- Hypervisor) 라고 부른다.

본 논문에서는 마이크로 하이퍼바이저를 개발하는 과정을 소개하고자 한다. Sorvisor 라고 명명한 마이크로 하이퍼바이저는 가상화를 지원하지 않기 때문에, 운영체제 스케쥴링, 메모리 및 I/O 공유 등의 기능을 제공하지 않는다. 다만, 운영체제 보다 높은 권한에서 임의의 코드를 수행할 수 있는 환경을 제공함으로써 추후 시스템 로깅 및 운영체제 보안 등의 기능을 제공할 수 있도록 하는 기반 플랫폼의 성격 가지게 된다.

2. Sorvisor 의 구현

Sorvisor 는 ARM Cortex A-15 를 타겟으로 개발되었다. ARM 프로세서에는 표 1 과 같이 총 9 가지 모드가 3 가지 단계의 권한을 기반으로 동작한다. 각각의 모드에 대한 자세한 소개는 ARM 레퍼런스 매뉴얼 [2] 에 소개가 되어 있으며, Secure / Non Secure State 는 ARM 프로세서가 보안을 목적으로 제공하는 기술인 Trustzone 을 제공하기 위해 도입된 것으로, 본 논문에서는 이에 대한 자세한 소개를 생략하고자 한다. ARM 에서 제공하는 다양한 프로세서 모드 중, 사용자가 사용하는 일반적인 응용프로그램들은 User 모드를 기반으로 동작하며, 운영체제의 커널은 Supervisor 모드를 기반으로 동작하고, Sorvisor 는 Hyp 모드를 기반으로 동작하게 된다. 이들 User, Supervisor, Hyp 모드는 각각 PL (Privilege Level) 0, 1, 2 의 계층적 권한을 가지게 되어 각각의 모드는 서로 접근할 수 있는 시스템 자원에 차이게 생기게 된다. 가령 운영체제 커널은 응용프로그램에서 예외상황이 발생하거나, 파일

표 1 ARM 프로세서의 모드와 권한

모드	권한	State
User	PL 0	Secure or Non Secure
System	PL 1	Secure or Non Secure
Hyp	PL 2	Non Secure
Supervisor	PL 1	Secure or Non Secure
Abort	PL 1	Secure or Non Secure
Undefined	PL 1	Secure or Non Secure
Monitor	PL 1	Secure or Non Secure
IRQ	PL 1	Secure or Non Secure
FIQ	PL 1	Secure or Non Secure

등의 시스템 자원에 접근하기 위하여 의도적으로 시스템 콜 (System Call)을 호출하는 경우, 그리고 외부에서 I/O 입력에 의한 인터럽트가 발생할 경우 등에 대해 실행해야 하는 핸들러 (Handler)를 지정하기 위해 vector 테이블을 도입하고 있다. Vector 테이블은 메모리 상에 위치하며, 일반적인 32-bit ARM 프로세서의 경우 0xffff0000 혹은 0x00000000 에 vector 테이블이 위치하게 된다. 하지만, 때에 따라 VBAR 레지스터를 활용하여 vector 테이블의 위치를 지정할 수 있는데, 이러한 VBAR 레지스터는 PL 1 에서만 접근이 가능하기 때문에 응용프로그램에서는 이를 조작할 수 없게 된다. 이는 Sorvisor 에서도 마찬가지인데, 커널을 포함한 시스템 전반에서 발생하는 예외처리 및 인터럽트와 커널에서 의도적으로 하이퍼바이저 서비스를 사용하기 위해 호출하는 하이퍼 콜 (Hyper call) 에 대응되는 vector 테이블의 위치는 HVBAR 레지스터에 저장이 되며, 이 레지스터는 PL 2 에서만 접근이 가능하기 때문에, 응용프로그램 뿐 아니라 커널에서는 접근이 불가능하다. 따라서 Sorvisor 는 Hyp 모드를 통해 가지는 베타적인 권한을 바탕으로 커널 등의 간섭없이 독자적인 실행 환경을 구축할 수 있게 된다.

ARM 프로세서에 전원이 인가되면, Reset 예외 상황이 발생하면서, Supervisor 모드로 설정이 된다. 다만, 이 때는 Secure State 로서, Non-Secure State 에 대한 일방적인 접근 및 수정 권한을 가지게 된다. Hyp 모드는 Non-Secure State 이기 때문에,이 시점에서는 Supervisor 모드가 비록 PL 1 이라고 하더라도 PL 2 인 Hyp 모드의 레지스터 등을 자유롭게 수정 및 설정할 수 있게 된다. 이 시점에서 수행해야 할 것은 다음과 같다.

a. 하이퍼 콜 설정
   - 최초에 설정해야 할 것은 시스템에서 하이퍼 콜을 활성화 하는 것이다. 이를 통해 커널을 포함한 PL 1 을 가진 모드들에서 Sorvisor 의 서비스를 사용할 수 있게 된다. 하이퍼 콜은 Secure PL 1 에서만 접근할 수 있는 SCR 레지스터에서 설정이 가능하다.
b. HVBAR 설정
   - 하이퍼 콜 이후에, 하이퍼 콜을 포함하여 Sorvisor 에서 처리해야 할 인터럽트 및 예외처리의 vector 테이블을 지정해야 한다.

여기서 주의 할 점은 HVBAR 를 유연하게 변경할 수 있도록 구현해야 한다는 점이다. Trustzone 을 사용하지 않는 많은 시스템에서 Secure State 는 시스템이 부팅된 최초에만 존재하게 되는데, 이 시점에 HVBAR 가 고정이 되면, 추후 시스템의 확장함에 따라 유연하게 대응할 수 없게 된다. 따라서 이 시점에서 하이퍼 콜을 설정하고, HVBAR 를 설정하는 것은 Sorvisor 의 기능을 구현하는 것 보다, 추후 Sorvisor 를 온전히 활성화 하기 위해 Non-Secure State 에서 자체적으로 HVBAR 를 수정할 수 있는 인터페이스를 제공하는데 의의를 두어야 한다.

Hyp 모드를 자유롭게 사용할 수 있게 되면, 본격적으로 Sorvisor 가 시스템 자원을 관리할 수 있도록 아래와 같이 해야 한다

c. HCR 설정
   - HCR 레지스터는 Hyp 모드에 대한 각종 설정을 포함한다. 대표적으로 MMU (Memory Management Unit)에 의한 2 단계의 메모리 변환

을 활성화 함으로써 하위 모드의 메모리 접근을 관리할 수 있게 된다. 이 외에도 모든 시스템 인터럽트 및 예외처리에 대하여 IRQ, FIQ 모드 등이 아닌 Hyp 모드로 전달 되도록 함으로써 시스템 자원을 관리할 수 있도록 한다.

이 부분에서 활성화 하는 MMU 에 의한 2 단계의 메모리 변환은 Hyp 모드에서 제공하는 가장 핵심 기능이다. 이를 통해 Sorvisor 는 커널 및 응용프로그램의 메모리를 감시하는 것 뿐 아니라, 그들이 Sorvisor 에 접근하는 것을 원천적으로 방지할 수 있게 된다. 본래 MMU 는 운영체제에서 응용프로그램들에 동일한 메모리 환경을 가상적으로 제공하기 위해 도입 되었다. 하이퍼바이저의 원 목적 또한 운영체제를 가상화하는 것에 있으므로, 프로세서는 MMU 의 기능을 총 2 단계로 구분하여, 1 단계는 운영체제에서 응용프로그램에 가상 메모리를 제공하기 위하여, 그리고 2 단계는 하이퍼바이저에서 운영체제에 가상 메모리를 제공하기 위하여 사용하도록 한다. 그러나 Sorvisor 는 운영체제 가상화를 지원하지 않기 때문에 2 단계 MMU 가 별반 필요 없다고 생각될 수 있다. 하지만, 커널이 Sorvisor 의 코드 및 데이터를 악의적으로 수정하는 것을 방지하기 위해서라도 2 단계 MMU 는 활성화할 필요가 있으며, 다만, MMU 에 의한 2 단계 메모리 변환에 따른 오버헤드를 최소화 하기 위하여 운영체제가 관리하는 메모리 공간과 Sorvisor 가 관리하는 메모리 공간은 기본적으로 1:1 로 동일하게 일치시킴으로써 TLB (Translation Lookaside Buffer)에서 발생할 수 있는 cache miss 를 최소화 하도록 한다.

Sorvisor 를 활성화 하기 위해 마지막으로 수행해야 할 작업은 운영체제의 메모리를 조정하는 작업이다. 2 단계 MMU 를 통해서 운영체제에서 Sorvisor 의 메모리 영역을 악의적 혹은 실수로 수정 및 접근하는 것은 방지할 수 있다. 그러나 운영체제가 Sorvisor 의 메모리 공간을 인지하지 못하고, 자신이 관리해야 하는 메모리 영역으로 여길 경우, 이를 사용하기 위한 시도와 이에 따른 예외 상황 발생 등으로 인하여 시스템의 성능이 저하될 수 있다. 이를 해결하기 위하여 운영체제 부팅 시, 운영체제에 전달되는 시스템 정보인 device tree 를 수정하거나 커널 명령어를 활용하여 운영체제가 관리하는 메모리를 제한 한 뒤, 잔여 메모리에 Sorvisor 를 탑재할 수 있다. 가령, 2048 MB 의 메모리가 있는 시스템의 경우 앞선 1920 MB 는 운영체제에서 관리하며, 나머지 128 MB 의 경우 Sorvisor 에 전용하도록 하게 된다.

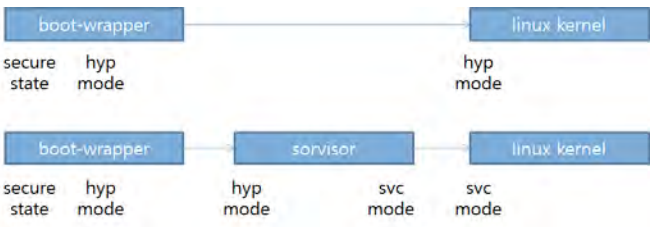

그림 1 Sorvisor 가 탑재된 시스템의 부팅 과정

3. Sorvisor 의 실험

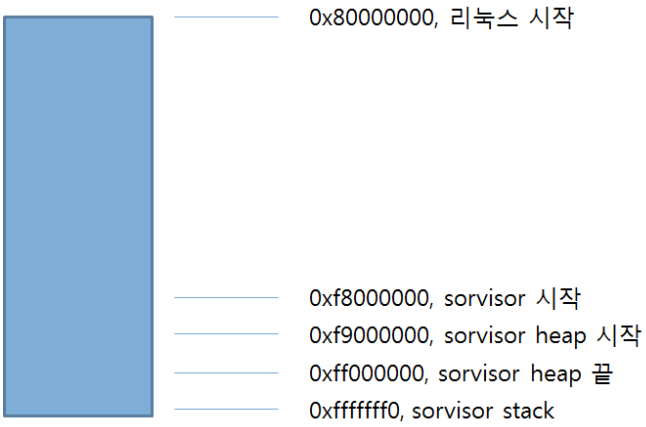

그림 2 Sorvisor 탑재 후 메모리 맵

Sorvisor 는 ARM 에서 제공하는 소프트웨어 기반 ARM 프로세서 시뮬레이터인 Fast Model 을 기반으로 개발하였으며 타겟 운영체제는 리눅스를 선택하였다. Fast Model 이 시작되면 간단한 부트로더인 boot-wraper 가 실행되며, boot-wraper 는 시스템에 대한 기본적인 설정을 한 후, 리눅스 커널로 실행 흐름을 넘기게 된다. 앞서 언급한 바와 같이 시스템이 시작한 직후에는 Secure 모드를 기반으로 동작하기 때문에, 그림 1 과 같이 boot-wraper 는 최초에 Secure state 에 속하지만, 리눅스 커널로 실행 흐름이 전환되는 시점에서는 Hyp 모드로 전환이 되게 된다. 그러나 이렇게 될 경우 커널 자체에서 Hyp 모드를 설정할 수 있게 되므로 Sorvisor 에 대한 무결성이 훼손될 수 있다. 따라서 boot-wraper 와 리눅스 커널 사이에 Sorvisor 를 설정하는 별도의 부트로더를 삽입하였다. 이 곳에서는 Sorvisor 의 기본적인 설정을 실행함과 동시에 Hyp 모드를 Supervisor 모드로 전환한 후, 리눅스 커널로 실행 흐름을 넘김으로써 리눅스 커널이 Sorvisor 영역에 접근할 수 없도록 하였으며, 추가적으로 리눅스 커널에 전달되는 device tree 를 수정하여 리눅스와 Sorvisor 가 분리된 메모리 공간을 가지도록 함으로써, 메모리 공간이 겹침으로써 발생할 수 있는 성능 저하를 최소화 하였다. 결과적으로 Sorvisor 가 탑재된 이후 시스템의 메모리 맵은 그림 2 와 같게 된다.

마지막으로 그림 3 은 실제로 Sorvisor 가 부팅되는 과정에서 발생하는 로그를 보인 것이며, 그림 4 는 Sorvisor 를 통해 리눅스가 부팅되는 과정으로, device tree 수정에 의해 리눅스의 메모리가 1920 으로 축소된 것을 확인할 수 있다.

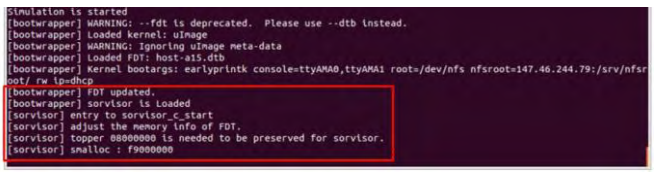

그림 3 Sorvisor 구동 로그

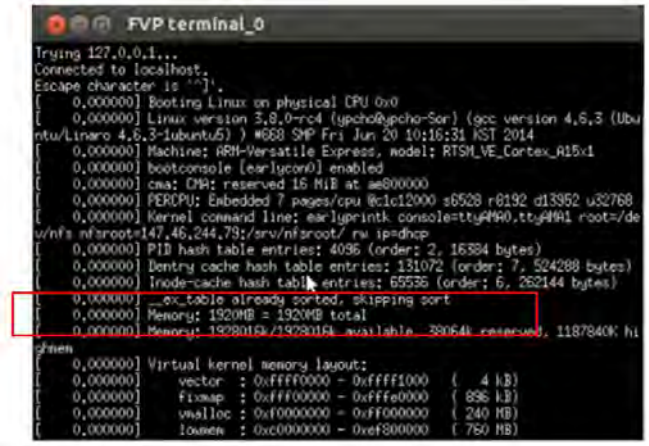

그림 4 Sorvisor 를 통한 리눅스 부팅

서사문

본 연구는 미래창조과학부/한국연구재단 우수연구센터 육성사업(과제번호 NRF-2008-0062609), 중소기업청에서 지원하는 2014 년도 산학연협력 기술개발사업(No. C0218072), 2014 년도 두뇌한국 21 플러스사업, 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신) [No. 10047212, 1kB 이하 암호문 간의 연산을 지원하는 동형 암호 원천 기술 개발 및 응용 연구] 및 IDEC 의 EDA Tool 의 지원을 받아 수행하였습니다.

참고문헌

[1] en.wikipedia.org/wiki/Hyperviso
[2] www.arm.com