VMM의 USB 2.0 API 사용을 위한 모듈제안

서영철, 이민순, 김준환, 이병수
인천대학교 일반대학원
yongzhexu@hotmail.com, zerone@incheon.ac.kr , andyhwan@incheon.ac.kr , bsl@incheon.ac.kr

Proposal Implement USB 2.0 API Module for VMM

Xu Yong Zhe, Min soon Lee, Jun hwan Kim, Byoung soo Lee
Incheon Graduate School

요 약

PC 기술의 발전으로 사용자는 하나의 물리적인 컴퓨터에서 서로 다른 운영체제를 실행하는 작업이 가능하게 되고 서버에서는 하드웨어의 교체로 인한 특정(MIPS, PowerPC)하드웨어에서 서비스 되고 있는 프로그램을 다른 하드웨어에 이식하는 경우의 재개발 비용 없이 성능을 개선할 수 있게 되었다.Virtualization 기술은 PowerPC기반의 서버 프로그램을 X86기반의 개인용 컴퓨터에 이식하는 경우와 같이 다양한 하드웨어 프로그램 개발 환경을 제공해 준다. 본 논문에서는 일반 사용자가 Virtualization을 사용하여 USB 장치를 사용할 수 있게 하는 다양한 방법들을 논한다. 이를 구현하기 위해 리눅스 커널 2.6의 특징인 모듈화를 이용하며 USB 디바이스에 대한 가상화 모듈을 구현하여 VMM (Virtual Machine Monitor)이 USB를 지원하도록 한다. 따라서 GuestOS(가상화하여 실행중인OS)에서는 별도의 개발 없이 USB를 사용할 수 있다.

1. 서 론

Virtualization 기술은 1960년 IBM에서 연구 되어왔다. 그 이후 많은 연구가 있었지만 운영체제 가상화는 시스템 성능의 제약으로 실용적이지 못했다. 그러나 2007년 하나의 CPU에 2-4개의 코어를 가진 고성능 개인용 컴퓨터의 등장으로 운영체제의 Virtualization은 점점 각광을 받고 있다. Virtualization 기술 중 VMware의 경우 운영체제의 가상화 뿐만 아니라 스토리지 Virtualization 등 연관된 제품을 판매하고 있다. Virtualization은 서버용 Application로 많이 개발 되었다. 서버용은 다양한 기능의 모듈이 추가되어 있어 일반 사용자의 경우 불필요한 기능으로 성능저하 문제점이 많다. 특히 USB 하드웨어의 Virtualization 기술은 아직 해결하고 개선해야할 점이 많다. 본 논문에서는 Virtualization 논문중에서 가장 많이 연구되고 비교적 완벽한 제품을 출시하고 있는 Full Virtualization 에서의 USB 2.0 API의 구현을 다룬다. 현시점의 문제점은 Virtualization을 위한 특권권한(접근, 실행)이 부여된 USB Switching기법이 Hardware chip으로 제작 되어 있지 않다는 점이다[3]. 따라서 Switching 기법을 S/W을 통하여 해결해야 한다. 본 논문에서는 이러한 문제점을 해결하기 위한 방법으로 USB를 선택적인 GuestOS에서만 Module화하는 방식을 제안하고자 한다. 이러한 방식은 몇 가지의 문제점은 있으나 일반 사용자일 경우 최소한의 CPU 자원으로 Virutalization을 통해 USB를 사용할 수 있게 한다. 본 논문에서는 USB를 지원하기 위한 다양한 방법들을 비교 하여 분석하고 일반 사용자와 개발자로 하여금 Virtualization에서 더욱 쉽게 USB 사용할 수 있게 한다.

본 논문에서는 VMM에 추가하는 USB API를 USB API 표준 모델로 제안 하며, VMM과 인터페이스 하는 부분도 이식성을 고려하여 개발한다.

2.1 관련연구

Virtualization, Multitasking, Hyper Threading

그림1 Virtualization 에서 Virtual CPU1, 2, 3 실제 하드웨어 CPU가 아닌 Virtualization을 사용하여 CPU 하여금 3개의 독립적인 CPU처럼 작동하게 하는 기술이다. 이 기술에 대해 특권권한[1]에 대한 이해가 요구 된다. Virtualization에서 Virtual CPU는 그에 대응하는 운영체제에 사용권한을 가능하게 해준다. 기본적인 운영체제는 Ring0에 대한 권한은 Kernel만 허가하게 하고 시스템이 실행하는 시작부터 끝나는 시점까지이다. 하지만 사용자가 Ring0권한을 사용하게 되면 사용자의 실수로 의해 실행되는 Application중 위험한 Instruction등은 시스템을 불안하게 한다. 이를 해결하기 위한 방식으로 Intel, CPU에서는 위험한 권한에 대한 해결 방법으로 위험한 Instruction 을 운영체제의 Kernel에서만 사용하게하고 사용자의 위험한 실행은 취소하게 한다. VMware의 해결책으로 Instruction을 사용하면 실행타임에서 VMM의 자체적인 허가 Instruction 변경하는 작업을 하여 문제점을 해결하였다. VMware는 사용자 매뉴얼에는 Intel VT 및 AMD V 지원한다고 있다. 운영체제는 독립적으로 실행되고 메모리는 강제적 방식으로 사용자가 처음 설정한 메모리대해 접근하여 사용한다. 사용자는 다른 GuestOS 설정한

메모리대한 권한은 없다. 이러한 Full Virtualization 방식으로 GuestOS (Virtualization 시스템)는 하나의 운영체제상에서 Application처럼 실행한다. 이러한 방식의 문제점은 GustOS 서로 독립적이므로 사용하고 있는 자원에 대한 독립적인 확보로 인하여 시스템 자원을 많이 소모한다. 하지만 서버 경우 GuestOS 문제로 인하여 다른 GustOS 및 BaseOS (GuestOS 실행하는 시스템)영향을 최소화 할 수 있다는 장점을 갖고 [1] 있다.

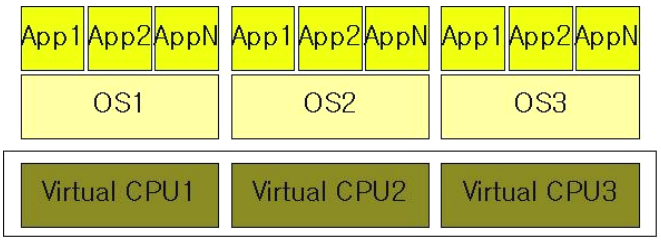


그림 1 Virtualization

그림 2 Multitasking 경우 하나의 CPU 그에 대응하는 운영체제 연산력 지원하고 여러 개의 Application Program 실행하고 있는 방식이다. Virtualization 차이점은 CPU 대한 가상화가 없다. Virtualization처럼 Hardware 가상화가 전혀 없다.

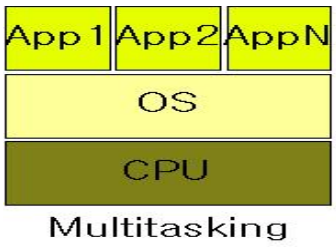


그림2 Multitasking

그림 3 Hyper Threading는 완전히 다른 방식 이다. Hyper Threading 은 CPU 업체인 Intel에서 처음 제시한 기술이다. Hyper Threading 은 SMP (Symmetric Multi Processing) 중 하나의 CPU 가 두 개의 CPU로 시뮬레이션 하여 프로그램 실행 성능을 평균화하는데 있다. 시뮬레이션 한 CPU 협동작업만 가능하다. 이를 간단하게 설명하면 프로세싱에서 더하기 빼기 연산과 floating point operation 을 두 개로 나누어 하나의 CPU 하여금 빠른 연산경우인 더하기 빼기 연산과 늦은 연산 경우인 floating point operation으로 분리하여 하나의 물리적인 CPU에서 두 개의 연산을 실행하게 하는 것이다.즉 floating point operation 연산중 나머지 CPU 자원을 더하기 빼기 연산에 지원 할뿐 서로 다른 체

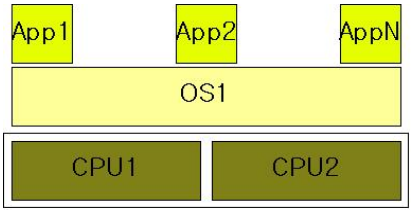


그림 3 Hyper Threading

계의 연산 체계를 가지고 있지 않다.

2.2 가상화 종류

밑에 그림 4에서 Virtualization는 현재 3가지 방식으로 되어 있다. 활발히 연구되고 있는 것은 Full_Virtualization 과 Para Virtualization 이다. Full_Virtualization는 하드웨어 컨트롤에 대한 모든 부분에서 가상화 하는 것이다. 다시 말하면 가상화 프로그램은 하드웨어 대한 Virtualzation하여 GuestOS 하여금 하드웨어 컨트롤 유일한 사용자인 착각을 하게 된다. 운영체제인 경우 하드웨어 사용에 있어서 유일성을 중요시 한다.
운영체제 구성에 있어서 하드웨어 자원에 대한 접근권한, 실행권한 진행중 다른 실행중인 프로그램 접근하면 운영체제는 굉장히 불안하게 작동되게 된다. 예를 들어 메모리에 대한 컨트롤 중에서 운영체제 커널이 잡고 있는 메모리 영역을 실행되는 다른 프로그램이 컨트롤 하면 운영체제의 커널은 그에 대한 저지가 없으면 동시접근 문제점으로 하여 커널은 기본 서비스를 사용자한테 지원 못하고 혼란스러운 작동 중 다시 커널 메모리에 올려야 하는 리셋의 현상이 나온다. 이를 해결하기 위한 방법으로 인텔등 CPU 업체에서 사용하는 방식은 접근권한 실행권한 등을 운영체제가 갖고 있는 ring0 ,ring3방식을[3] 사용한다. 즉 사용자 하여금 위험한 실행을 제한하자는 취지에서 나온 방식이다. 전체 가상화는 이러한 방식을 모든 부분에서 Virtualization 하여 구현하는 방식이다[10].

Para Virtualization 경우 하드웨어 대한 Virtualization 경우 BaseOS에서 사용하는 하드웨어 컨트롤에 의하여 GuestOS는 하나의 사용자 프로그램처럼 BaseOS에서 실행하고 있는 방식이다. 하드웨어 대한 가상화 보다는 GuestOS에 운영체제 변경을 하여 커널에서 직접 하드웨어 컨트롤에 대한 (접근권한 실행권한)을[3] GuestOS한테 제공하는 방식을 말한다. 현 시점 Para Virtualization은 가상화 제품중에서 제일 좋은 성능을 갖고 있다. Full Virtualization 경우 자원의 손실을 5%이상인데 비하면 Para Virtualization 5%미만의 손실로 측정 되였다. 초창기 Para Virtualization 경우 GuestOS 대한 커널의 변경문제점으로 되고 있었지만 (CPU 연산의 Switching) 현시점에서 Para Virtualization 제품중 Xen은 Intel VT사용하여 Virtualization 구현하고 [2]있다.

2.3 Protection Ring

그림 5 이를 설명하기 위하여 CPU Instruction System 대한 이해가 있어야 한다. 이는 특별권한이다. 이중 17개의 Instruction은 위험한 특성을 지닌다. 17개 Instruction 실행에서 문제점 있을 경우 시스템의 붕괴를 맡게 된다. 예를 들면 메모리 dump, 사용 중인 메모리 접근, 수정, 하드웨어 대한 접근 (커널을 통하지 않은 접근) 이러한 문제점으로 인해 CPU는 특권권한의 Instruction과 비 특권권한의 Instruction으로 분류 되여있다. 위험한 Instruction 경우 오직 시스템 및 대응되는

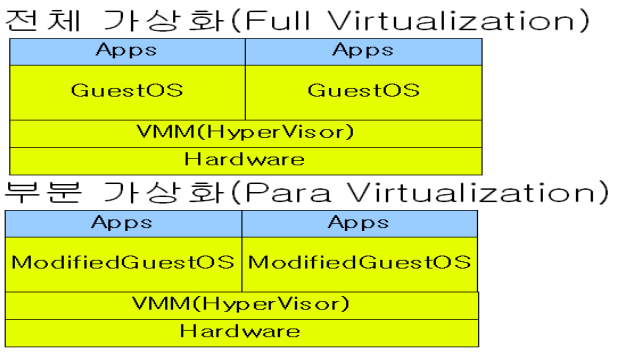


그림 4 가상화 종류

모듈만 사용하게 하고 일반 Application Program 경우 위험하지 않은 Instruction만 사용하게 설정해야 한다. Intel CPU 경우 특권 권한을 4개의 레벨로 나누었다. RING0, RING1, RING2, RING3 윈도우 시스템 경우 이중의 RING0,RING3만 사용하며 RING0 경우 시스템 커널만 사용이 가능하게 하고 RING3 경우 일반 Application 사용이 가능하게 하였다. 일반 Application Program이 Ring0의 Instruction 실행 중 윈도우 시스템에서는 "합법적이지 않은 Instruction " 에러 메시지를 출력 실행해 취소한다. [3]이러한 특성에 의하여 Virtualization 경우 BaseOS 와 GuestOS간의 Switching을 구현한다. Swithcing 없을 경우 모든 작업을 VMM에서 Instruction을 변경하는 작업을 했으며 이는 별도의 연산을 소모하게 된다. 새로 나온 Intel VT 경우 이러한 작업을 Hardware상에서 가능하게 했다. Virtualization 프로그램 제작의 난이도를 대폭 하향하는 우점을 갖게 된다.

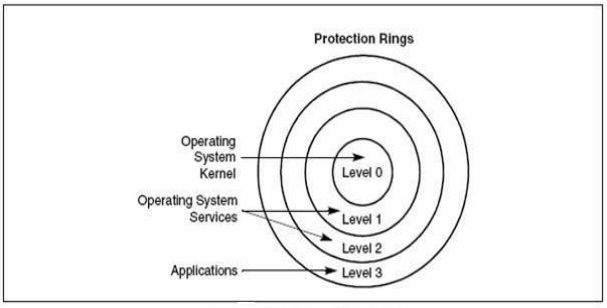


그림 5 Protection Ring

2.4 USB Linux Module

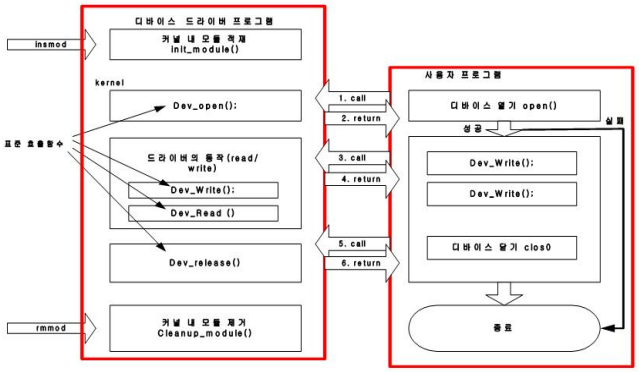
그림 6 리눅스 디바이스 드라이버

(1) 사용자 관점에서의 디바이스 드라이버

사용자는 디바이스 자체에 대한 자세한 정보를 알 필요가 없고 디바이스는 하나의 파일로 인식되고 파일에 대한 접근을 통하여 실제 디바이스 접근 가능하다.

(2)리눅스에서의 디바이스 드라이버

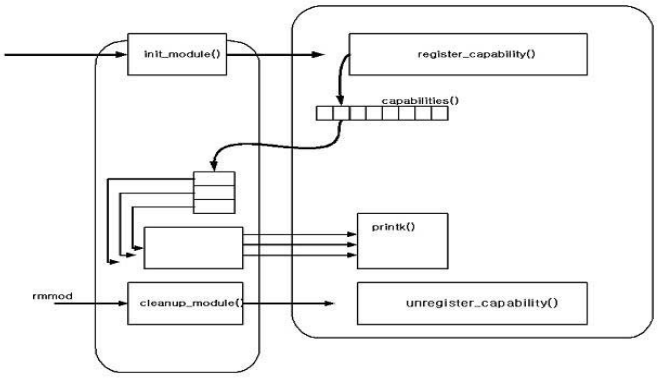
리눅스에서 디바이스는 특별한 하나의 파일처럼 취급되고 억세스가 가능하고 사용자는 파일동작을 적용할 수 있고 각 디바이스는 주 번호 (Mjaor Number) 와 부 번호 (Minor Number)를 갖는다. 디바이스 드라이버는 커널 모드에서 실행되고 메모리에 상주하며 swap 되지 [4]않는다.



리눅스 디바이스 드라이버 등록 도식도

그림 6

2.5 커널 모듈



커널에 모듈이 link되는 개념도

그림 7 Kernel Module link 개념도

그림 7의 모듈은 리눅스 시스템이 부팅 (Booting) 된 후에 동적으로 load, unload할 수 있는 커널의 구성요소이다. 이런 특징으로 커널을 다시 컴파일 하거나 시스템을 재부팅 할 필요 없이 커널의 일부분을 교체 할 수 있다. 디바이스 드라이버 , 파일시스템, 네트워크 프로토콜 등이 모듈로 만들어져 커널에 포함된다. 커널에서는 의존성이라는 또 다른 특성이 존재하는데 커널을 구성하고 있는 모든 모듈에는 컴파일한 커널의 버전 정보가 들어 가야하고 현재 실행되고 있는 커널 버전과 같아야 한다는 의미로, 만약 다르면 에러가 발생한다. 모듈의 버전 정보는 리눅스 커널소스에 정의 되어있다. insmod (install module)시에 위의 변수를 이용하여 버전을 검사하게 되는 것이다. 따라서 모듈 버전 정보는 전체 모듈에서 하나만 존재해야 한다.

2.6 USB 구조

그림 8 Linux 하위 시스템 (subsystem)에서는 단지 하나의 자료구조를 사용하는데 그것이 HRB[5]이다. 이 자료구조는 USB에서의 모든 전송을 설정하는데 필요한 매개변수를 포함하고 있다. 모든 전송요구는 USB core쪽으로 비동기적으로 전달되며 요구의 완료은 callback함수로 알려진다.

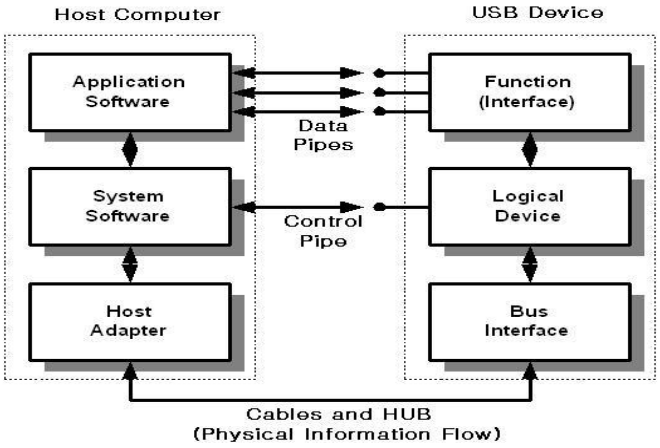


그림 8 Computer와 USB Device 구조도

2.7 USB device driver의 초기화 및 제거하기

USB device driver를 초기화는 것은 먼저 모듈(module)을 초기화 하는 것과 그와 관련된 함수부분에서 USB device를 찾아서 초기화 하는 부분으로 이루어 졌다. 모듈을 초기화 하기 위해서는 밑에 표와 같이 하면 된다.

```
int __init init_module (void)
{
        return plusb_init ();
}

void __exit cleanup_module (void)
{
        plusb_cleanup ();
}
```

3 USB API 선택적 모듈화

그림 9 USB API 모듈은 시스템에서 VMM 실행이 없는 경우에만 커널에 USB API 추가하게 된다. VMM 추가후 사용자 VMRESUME에 있는 GuestOS의 QIHotKey함수[7] 등 여러 방식으로 사용하여 사용자가 작업하고 있는 GuestOS 선별하여 Virtualization 모듈을 작업하고 있는 GuestOS에 추가한다. 이때 GuestOS는 USB 하드웨어에 대응된 모듈을 찾아내고 USB 사용이 가능하게 된다. GuestOS에서 VMMOFF 메시지 출력하면 이에 대한 모듈에 CleanUP를 실행(USB device driver에서)한다[11]. 이러한 과정으로 제작될 경우 시스템 부화를 최소화 된다. USB 구동 모듈을 최소화하고 오직 사용하는 GuestOS에서만 사용하게 하는 방식이다[8].

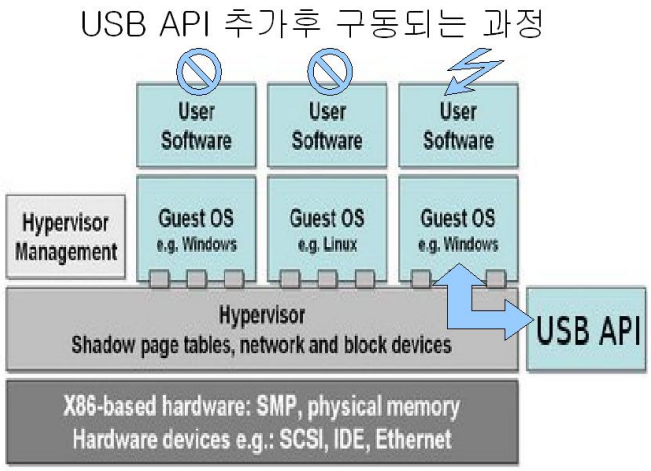


그림 9 USB API 구동되는 과정

4 결론

하지만 본 논문에서의 문제점으로는 생성된 모듈에서 사용자 작업이 끝나지 않은 상태에서 GuestOS에서 Switching하는 경우 이에 대한 보호가 전혀 없다. 이를 개선하는 방법으로 VMM에서 Manage가 해당 모듈의 사용여부 즉 쓰기, 읽기, 실행 등 조작이 있을 경우 Switching을 불가한 상태에서 사용자로 하여금 해당 작업이 끝나는 시점까지 Watting (Watting Time Schedule)하게 하는 방법이 있다[10]. 그러나 이 경우 사용자 사용에 있어서 불편함을 감수해야 한다. 다른 방법으로 USB Virtualization 경우 분산으로 가능하게 하는 방법이 있지만 문제점은 GuestOS 간의 USB 상태를 실행타임에 실시간으로 VMM Manage와 정보교환이 구현 되여야 한다. 다른 방식으론 USB완벽히 Virtualization하는 방식이 있다. 이 방식은 BaseOS USB 모듈을 추가하고 이에 대한 정보를 VMM과 실행타임에 VMM과 실시간 통신하여 모든 GuestOS에 가상화된 모듈을 지원하고 이에 대한 접근 제어 가능하고 기존 환경에서 변화가 있을 경우 이에 대해 VMM에서 관리 및 커널과 연결하여 변경된 조작을 실행하는 방식이 있다. 이 방식의 문제점은 더욱 많아진 연산으로 인하여 시스템 부화에도 영향을 준다. 특히 대용량의 Bulk 데이터의 변경사항이 있을 경우 병렬처리로 이루어 질 경우 이에 대한 최적화 작업이 있어야 한다. Switching (GuestOS & BaseOS)빈번히 이루어질 경우 이에 대한 자원의 낭비도 불가피하게 증가한다. USB의 Virtualizaiton는 자원의 낭비와 사용의 편리성에서 최적의 성능에서 문제점으로 되고 있다. 이에 대해 해결책으론 추후의 분산 병렬 실시간 등 기술로 성능과 편리성 그리고 사용자와 시

스템간의 고려하여 그 과정을 제작 하여야 한다. 모듈의 빈번한 생성과 소멸일 경우 완벽한 가상화로 해결방향을 두고 아니면 Virtualization에 모듈추가로 해결하면 성능의 저하를 줄 일 수 있다. USB 3.0 부터 Virtualization 기술을 chip에서 실현 한다고 한다. Intel VT, AMD V 등 CPU업체에서 지원하고 있는 Switching기법을 USB3.0에서 지원한다[6]

참고문헌

[1]wikipedia,http://en.wikipedia.org/wiki/Instruction_set_architecture

[1]virtualization,http://zhidao.baidu.com/question/25003268.html

[2]Intel®VirtualizationTechnologyforDirectedI/O,http://www.intel.com/technology/itj/2006/v10i3/2-io/5-platform-hardware-support.htm

[3]ProtectionRing,,http://hi.baidu.com/hurryhx/blog/item/fccb77f0ddba66c27831aa77.html Intel Develop Manual

[4] 임베디드시스템 5.1장에서 발취

[5] USB Guide for Developer

[6]http://www.extremetech.com/article2/0,1558,2184743,00.asp?kc=ETRSS02129TX1K0000532

[7]VirtualBox Virtualization User Manual

[8]Xen the art of Virtualization

[9]Robert Rose, Survey of System Virtualization Techniques

[10]Eric S. Chung, Eriko Nurvitadhi, James C. Hoe, Babak Falsafi, Ken Mai,Virtualized Full-System Emulation of Multiprocessors using FPGAs

[11]James E.Smith ,Ravi Nair ,The Architecture of Virtual Machines