VELLORE INSTITUTE OF TECHNOLOGY
WINTER SEMESTER 2021

CSE2005 OPERATING SYSTEMS

VIRTUALIZATION

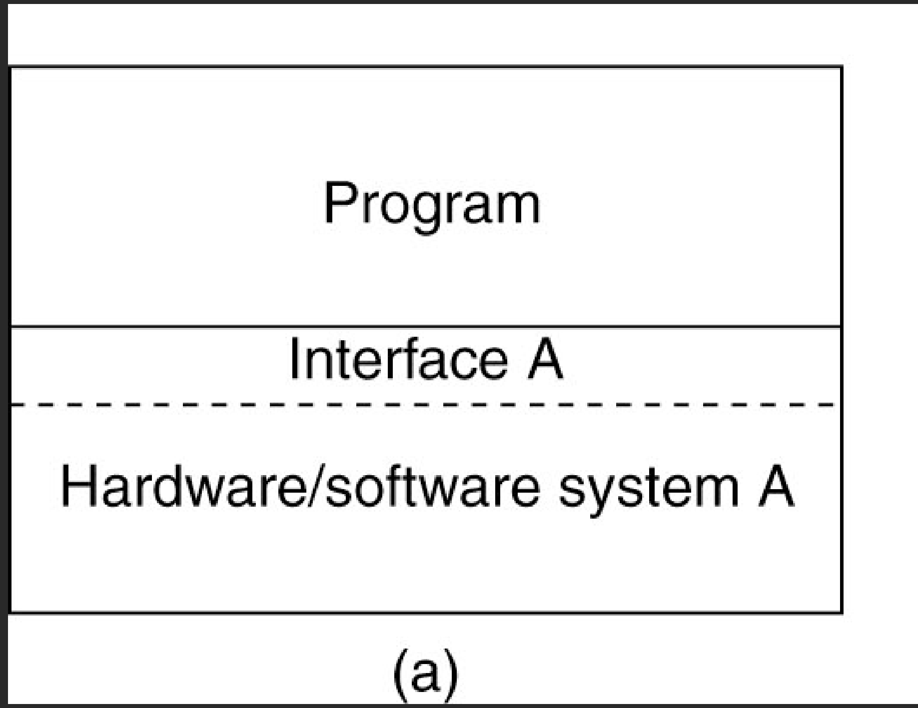
DIGITAL ASSIGNMENT 03

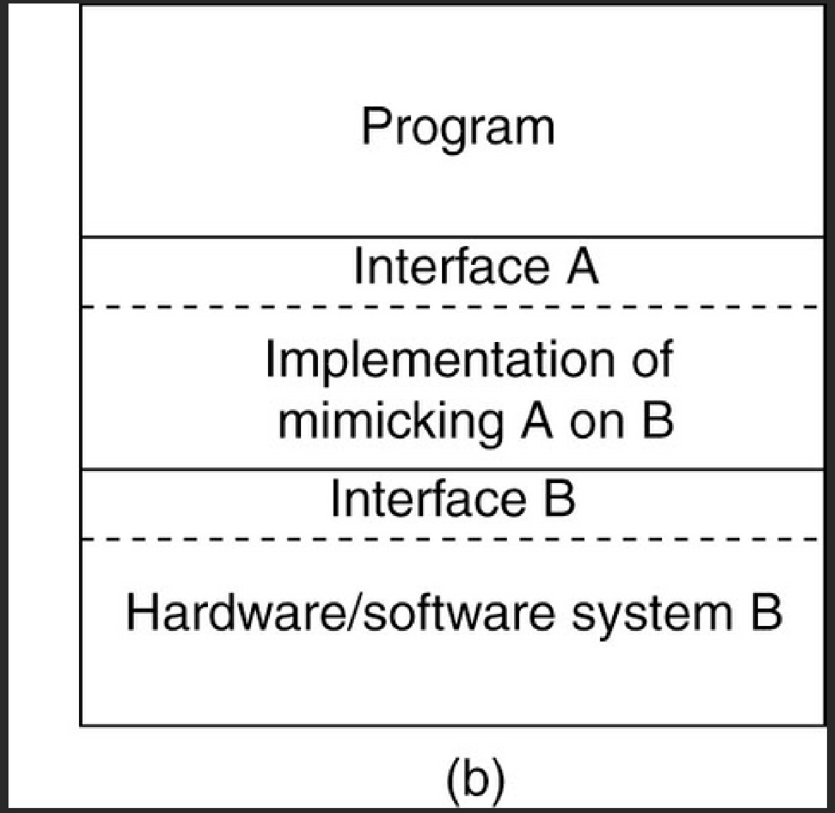
REG NO: 19BCE1521

NAME :PRADUSH J N

VIRTUALIZATION

- Virtualization: extend or replace an existing interface to mimic the behavior of another system.
  - – Introduced in 1970s: run legacy software on newer mainframe hardware
- Handle platform diversity by running apps in VMs
  - – Portability and flexibility





VIRTUALIZATION MACHINES

- Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS
- A machine with virtualization software can host numerous applications, including those that run on different operating systems, on a single platform
- The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS
- The solution that enables virtualization is a virtual machine monitor (VMM), or hypervisor

APPROACHES TO VIRTUALIZATION

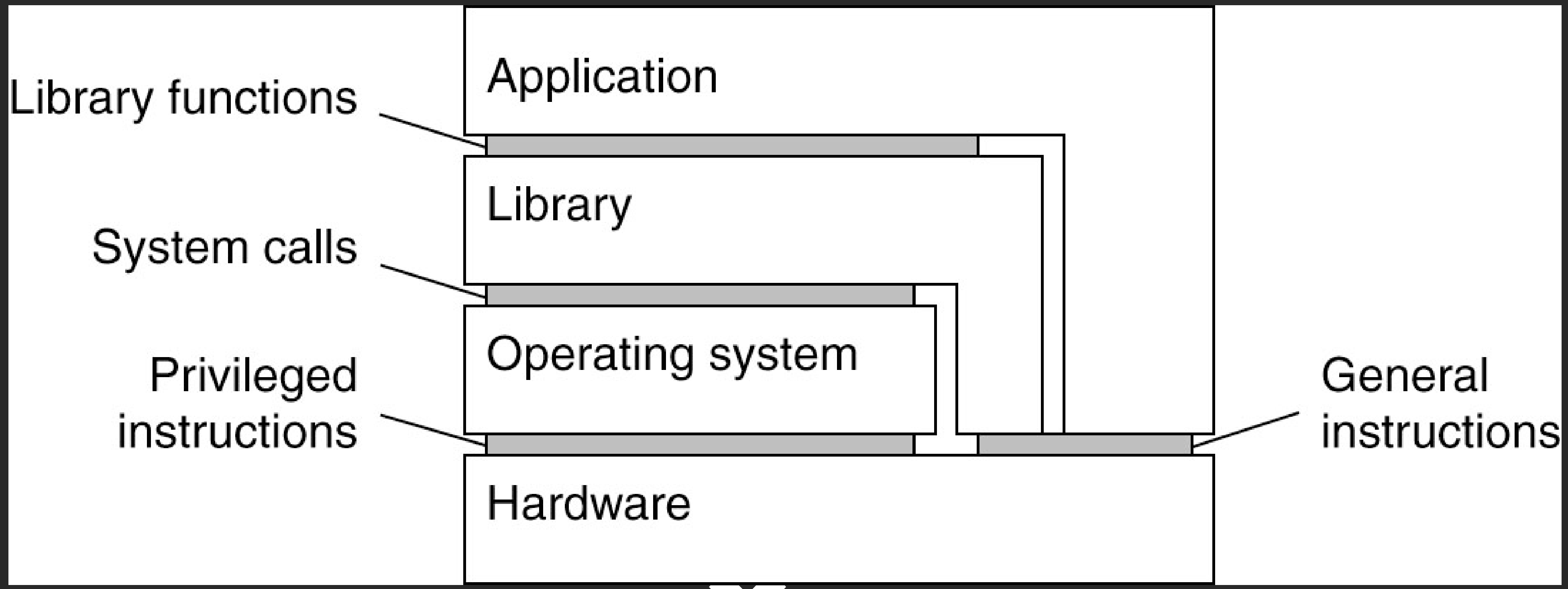
- it is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports
- once the VM is created it can be powered on like a physical server, loaded with an operating system and software solutions, and utilized in the manner of a physical server
- unlike a physical server, this virtual server only sees the resources it has been configured with, not all of the resources of the physical host itself

VIRTUAL MACHINES ARE MADE UP OF FILES:

- configuration file describes the attributes of the virtual machine
- it contains the server definition, how many virtual processors (vCPUs) are allocated to this virtual machine, how much RAM is allocated, which I/O devices the VM has access to, how many network interface cards (NICs) are in the virtual server, and more
- it also describes the storage that the VM can access
- when a virtual machine is powered on, or instantiated, additional files are created for logging, for memory paging, and other functions
- since VMs are already files, copying them produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself

TYPES OF INTERFACES

- Different types of interfaces
  - Assembly instructions
  - System calls
  - APIs
- Depending on what is replaced /mimiced, we obtain different forms of virtualization



TYPES OF VIRTUALIZATION

- Emulation
  - – VM emulates/simulates complete hardware
  - – Unmodified guest OS for a different PC can be run
- Bochs, VirtualPC for Mac, QEMU
- Full/native Virtualization
  - – VM simulates "enough" hardware to allow an unmodified guest OS to be run in isolation
- Same hardware CPU
  - – IBM VM family, VMWare Workstation, Parallels, VirtualBox

TYPES OF VIRTUALIZATION

- Para-virtualization
  - – VM does not simulate hardware
  - – Use special API that a modified guest OS must use
  - – Hypercalls trapped by the Hypervisor and serviced
  - – Xen, VMWare ESX Server
- OS-level virtualization
  - – OS allows multiple secure virtual servers to be run
  - – Guest OS is the same as the host OS, but appears isolated
- apps see an isolated OS
  - – Solaris Containers, BSD Jails, Linux Vserver, Linux containers, Docker
- Application level virtualization
  - – Application is gives its own copy of components that are not shared
- (E.g., own registry files, global objects) - VE prevents conflicts
  - – JVM, Rosetta on Mac (also emulation), WINE

TYPES OF HYPERVISORS

- Hypervisor/VMM: virtualization layer

– resource management, isolation, scheduling, …

- Type 1: hypervisor runs on “bare metal”
- Type 2: hypervisor runs on a host OS

– Guest OS runs inside hypervisor

- Both VM types act like real hardware

HOW VIRTUALIZATION WORKS?

- CPU supports kernel and user mode (ring0, ring3)
  - – Set of instructions that can only be executed in kernel mode
- I/O, change MMU settings etc -- sensitive instructions
  - – Privileged instructions: cause a trap when executed in user mode
- Result: type 1 virtualization feasible if sensitive instruction subset of privileged instructions
- Intel 386: ignores sensitive instructions in user mode
  - – Can not support type 1 virtualization
- Recent Intel/AMD CPUs have hardware support
  - – Intel VT, AMD SVM
- Create containers where a VM and guest can run
- Hypervisor uses hardware bitmap to specify which inst should trap
- Sensitive inst in guest traps to hypervisor

TYPE I HYPERVISOR

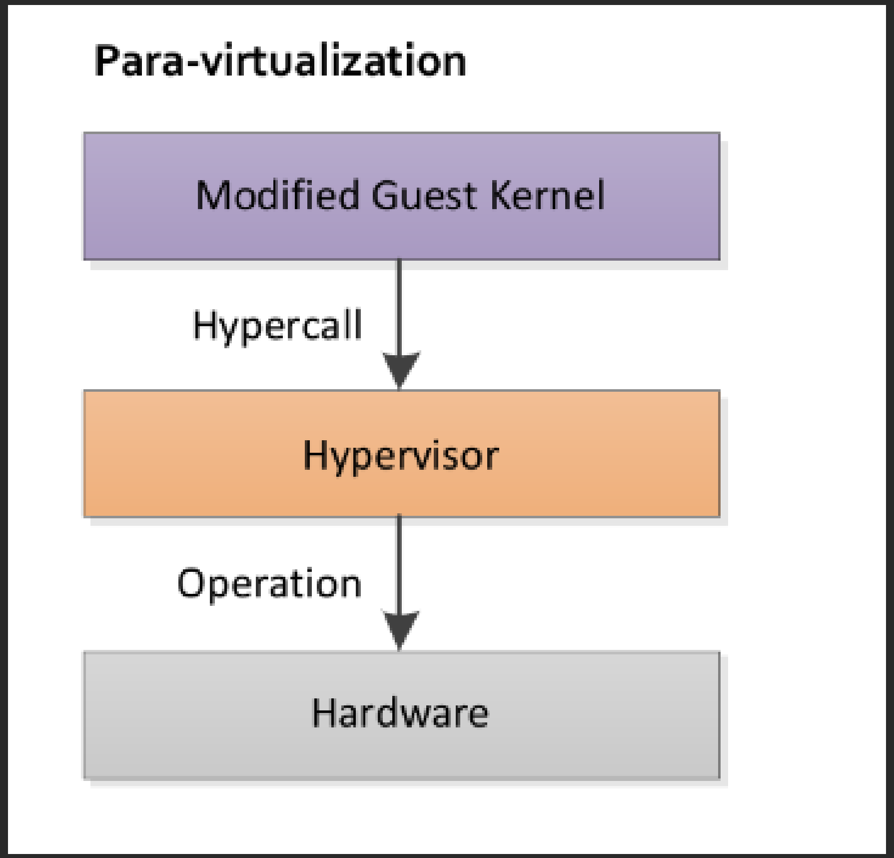
- Unmodified OS is running in user mode (or ring 1)
  - – But it thinks it is running in kernel mode (virtual kernel mode)
  - – privileged instructions trap; sensitive inst-> use VT to trap
  - – Hypervisor is the “real kernel”
- Upon trap, executes privileged operations
- Or emulates what the hardware would do

TYPE 2 HYPERVISOR

- VMWare example
  - Upon loading program: scans code for basic blocks
  - If sensitive instructions, replace by Vmware procedure
- Binary translation
  - Cache modified basic block in VMWare cache
- Execute; load next basic block etc.
- Type 2 hypervisors work without VT support
  - Sensitive instructions replaced by procedures that emulate them.

PARAVIRTUALIZATION

- Both type 1 and 2 hypervisors work on unmodified OS
- Paravirtualization: modify OS kernel to replace all sensitive instructions with hypercalls
  - – OS behaves like a user program making system calls
  - – Hypervisor executes the privileged operation invoked by hypercall.



MEMORY VIRTUALIZATION

- OS manages page tables
  - Create new pagetable is sensitive -> traps to hypervisor
- hypervisor manages multiple OS
  - Need a second shadow page table
  - OS: VM virtual pages to VM's physical pages
  - Hypervisor maps to actual page in shadow page table
  - Two level mapping
  - Need to catch changes to page table (not privileged)
- Change PT to read-only - page fault
- Paravirtualized - use hypercalls to inform

I/O VIRTUALIZATION

- Each guest OS thinks it “owns” the disk
- Hypervisor creates “virtual disks”
  - Large empty files on the physical disk that appear as “disks” to the guest OS
- Hypervisor converts block # to file offset for I/O
  - DMA need physical addresses
- Hypervisor needs to translate
- NIC Virtualization

VIRTUAL APPLIANCES & MULTI-CORE

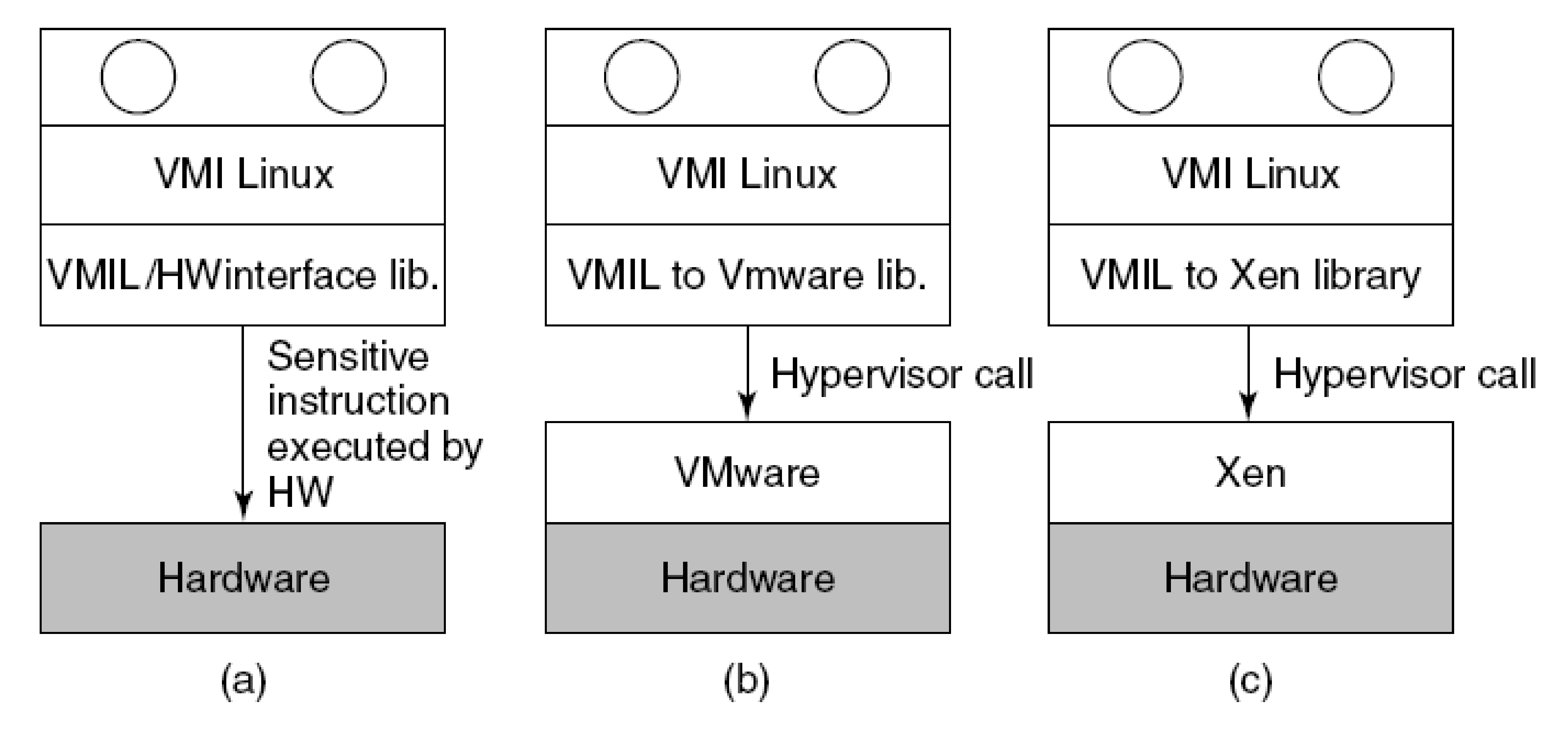
- Virtual appliance: pre-configured VM with OS/ apps pre-installed
  - Just download and run (no need to install/configure)
  - Software distribution using appliances
- Multi-core CPUs
  - Run multiple VMs on multi-core systems
  - Each VM assigned one or more vCPU
  - Mapping from vCPUs to physical CPUs
- Today: Virtual appliances have evolved into docker containers

USE OF VIRTUALIZATION TODAY

- Data centers:
  – server consolidation: pack multiple virtual servers onto a smaller number of physical server
- saves hardware costs, power and cooling costs
- Cloud computing: rent virtual servers
  – cloud provider controls physical machines and mapping of virtual servers to physical hosts
  – User gets root access on virtual server
- Desktop computing:
  –Multi-platform software development
  – Testing machines
  – Run apps from another platform

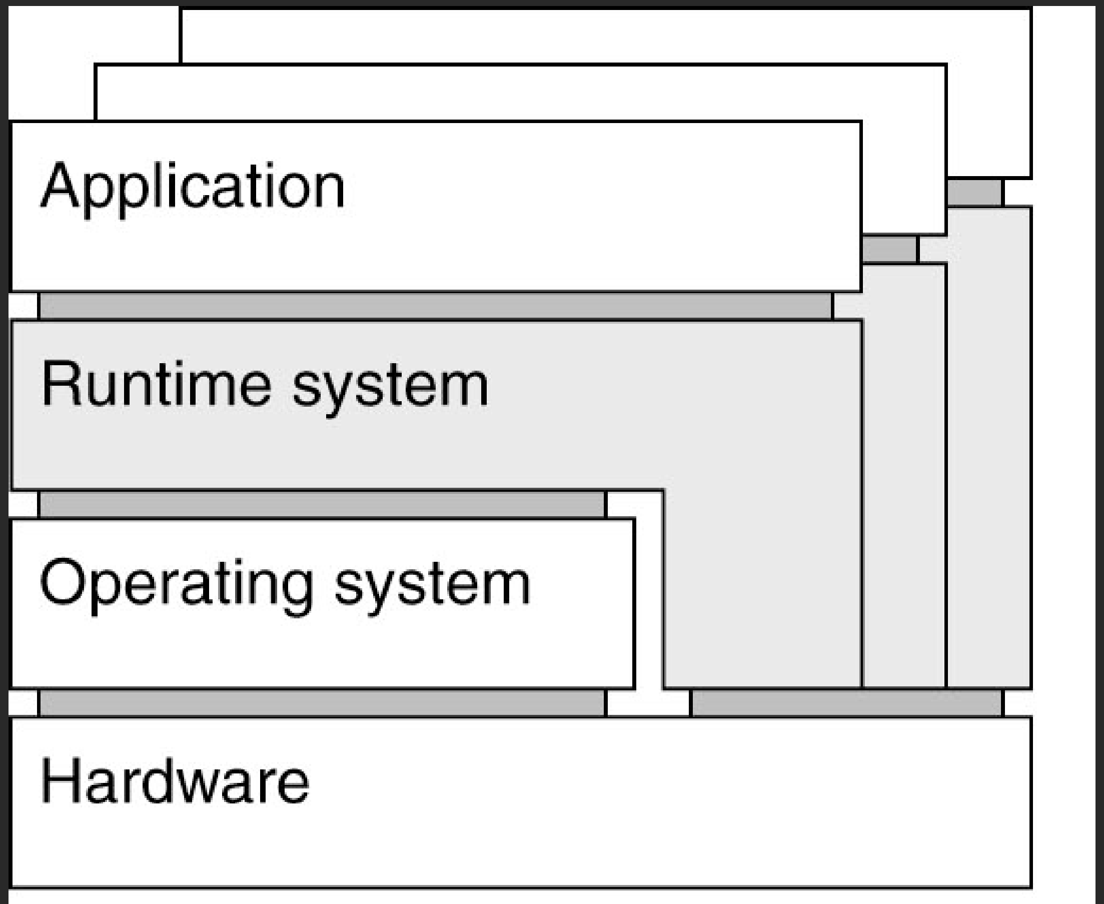
VIRTUAL MACHINE INTERFACE

- Standardize the VM interface so kernel can run on bare hardware or any hypervisor

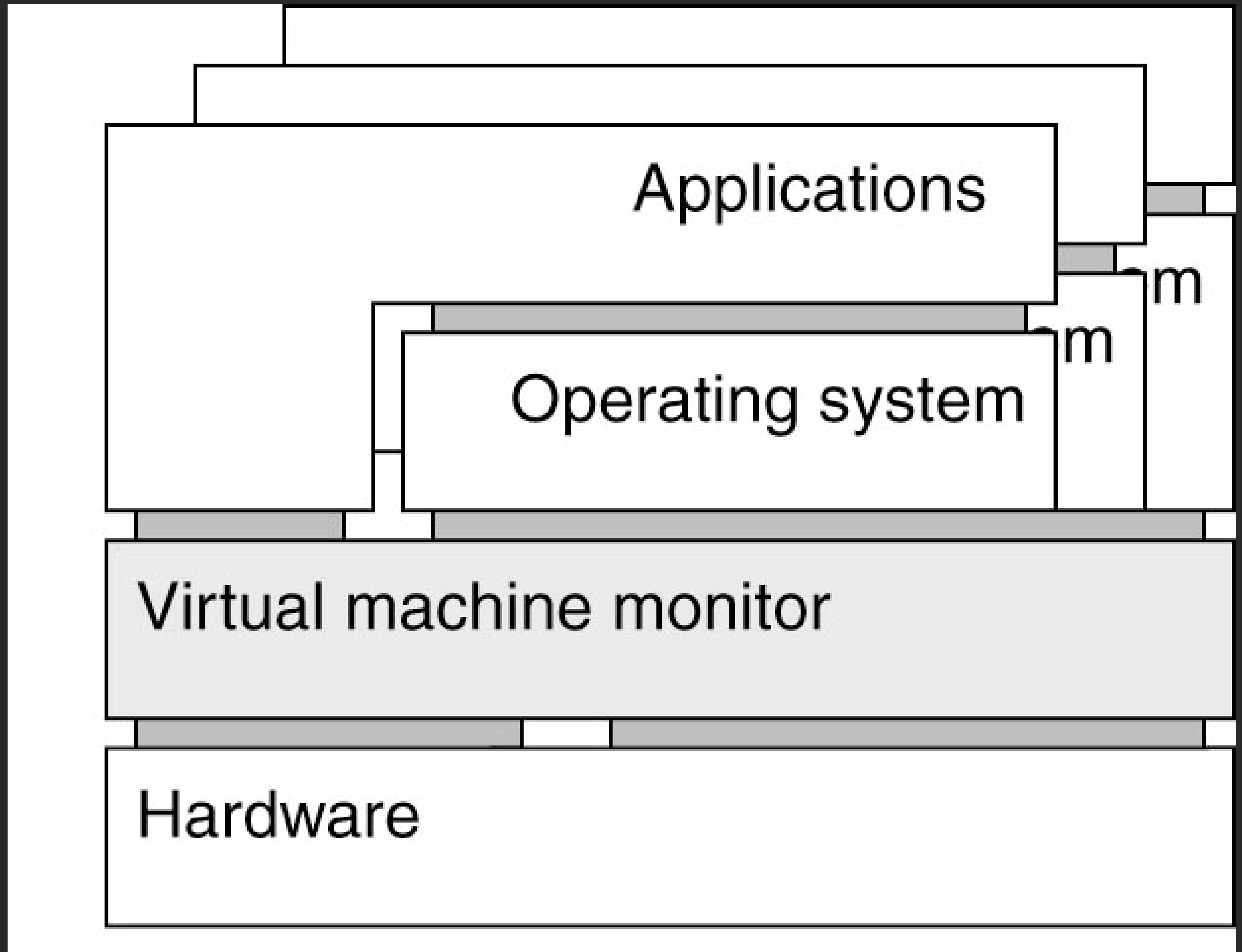


EXAMPLES

- Application-level virtualization: “process virtual machine”
- VMM /hypervisor



(a)



(b)

JAVA VM

- **The goal of a Java Virtual Machine (JVM) is to provide a runtime space for a set of Java code to run on any operating system staged on any hardware platform without needing to make code changes to accommodate the different operating systems or hardware**
- The JVM can support multiple threads
- Promises “Write Once, Run Anywhere”
- The JVM is described as being an abstract computing machine consisting of:
- an instruction set
- a program counter register
- a stack to hold variables and results
- a heap for runtime data and garage collection
- a method area for code and constants

LINUX VSERVER

- Linux VServer is an open-source, fast, lightweight approach to implementing virtual machines on a Linux server
- Only a single copy of the Linux kernel is involved
- VServer consists of a relatively modest modification to the kernel plus a small set of OS userland tools
- The VServer Linux kernel supports a number of separate virtual servers
- The kernel manages all system resources and tasks, including process scheduling, memory, disk space, and processor time

thank you