Virtualization

ACKNOWLEDGEMENTS

- This presentation has been made from various sources with minimum modifications from the presenter.
- The presenter is grateful to the authors of those various sources.
- The presenter acknowledge the efforts of those authors and thank them wholeheartedly.

Virtualization

Virtualization is a way to abstract applications and their underlying components away from the hardware supporting them,

& present a logical or virtual view of these resources.

Virtualization

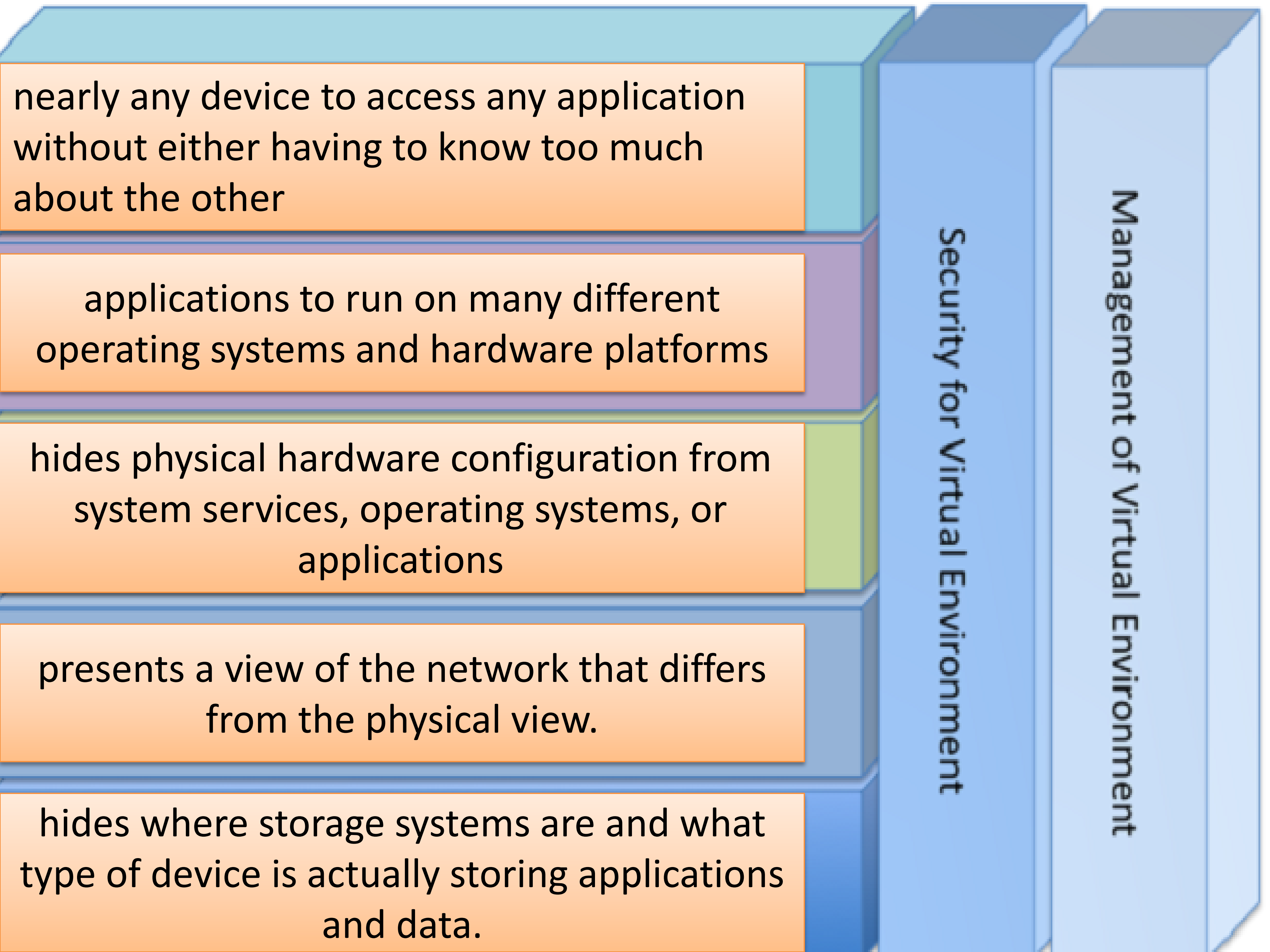
- Abstracts the underlying resources and simplifies their use
- Isolates users from one another,
- Supports replication
  - increases the elasticity
- Is a critical aspect of cloud computing
- Virtualization allows users to operate in environments they are familiar with,
- Virtualization has been used successfully since the late 1950s;
  - virtual memory based on paging was first implemented on the Atlas computer at University of Manchester in the United Kingdom, in 1959

Cloud Resource Virtualization

- Virtualization plays an important role for:
  - System security,
    - as it allows isolation of services running on the same hardware;
  - Performance and reliability,
    - as it allows applications to migrate from one platform to another;
  - The development and management of services offered by a provider;
  - Performance isolation.

Virtualization

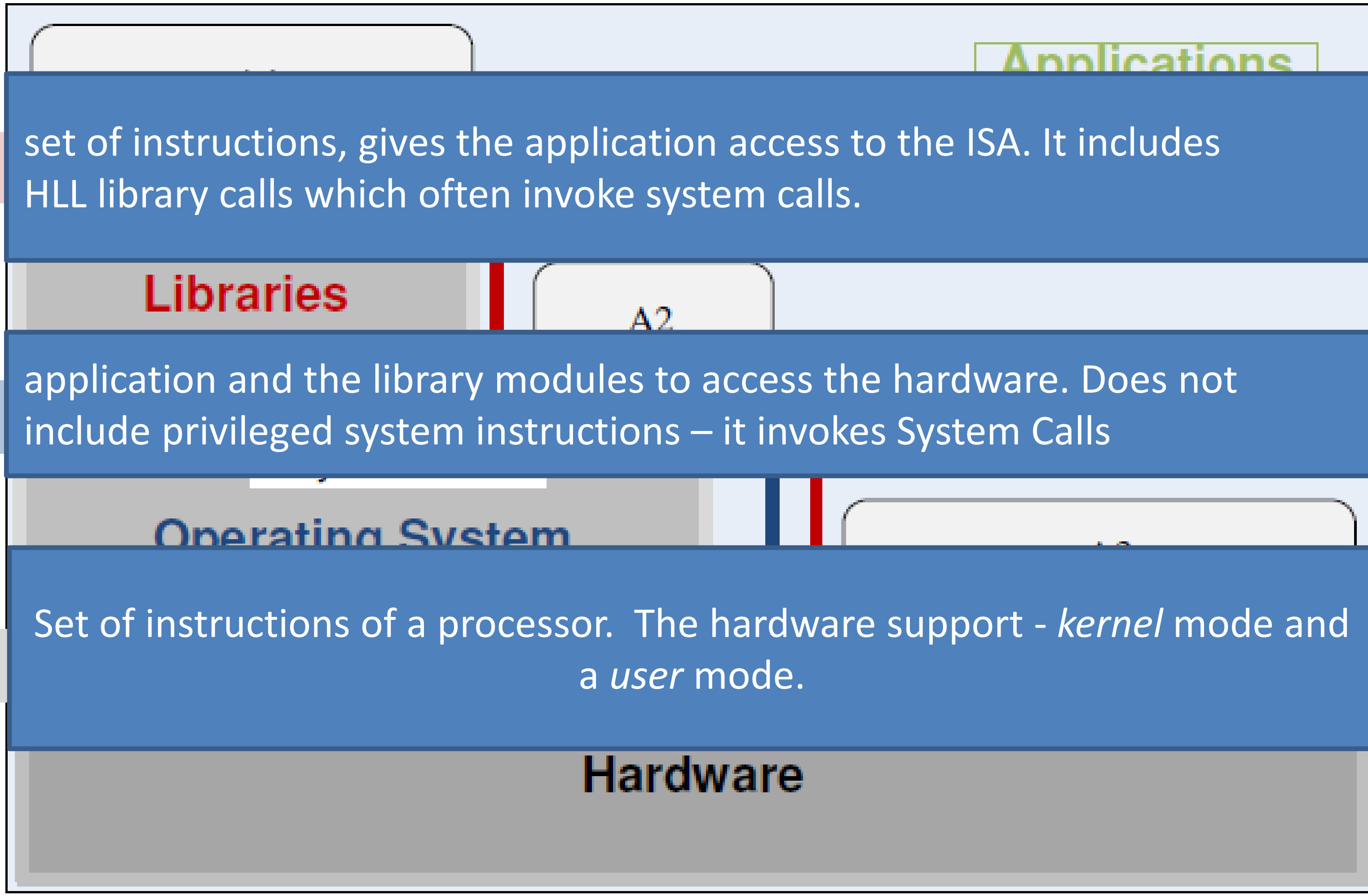
- Virtualization simulates the interface to a physical object by :
  - Multiplexing [Multiple from one]
    - Ex: processor is multiplexed among a number of processes or threads
  - Aggregation [one from multiple]
    - Ex: A number of physical disks are aggregated into a RAID disk
  - Emulation [one to one – different type]
    - Ex: A physical disk emulates a Random Access Memory
  - Multiplexing and emulation
    - Ex: virtual memory with paging multiplexes real memory and disk and a virtual address emulates a real address


nearly any device to access any application without either having to know too much about the other
applications to run on many different operating systems and hardware platforms
hides physical hardware configuration from system services, operating systems, or applications
presents a view of the network that differs from the physical view.
hides where storage systems are and what type of device is actually storing applications and data.
Security for Virtual Environment
Management of Virtual Environment

Virtualization in Cloud

- In Cloud Environment
  - virtual-machine monitor runs on the physical hardware and exports hardware-level abstractions to one or more guest operating systems
- *User convenience is a major advantage of a Virtual Machine architecture versus a traditional operating system*
- There are side effects of virtualization, notably the *performance penalty* and the *hardware costs*.

Layering and virtualization



API

set of instructions, gives the application access to the ISA. It includes HLL library calls which often invoke system calls.

ABI

application and the library modules to access the hardware. Does not include privileged system instructions – it invokes System Calls

ISA

Set of instructions of a processor. The hardware support - *kernel* mode and a *user* mode.

Ways to Virtualize

- Hardware Emulation
- Para-Virtualization
- Virtualization on the OS level
- Multi-server virtualization

Hardware Emulation

a.k.a. VM (Virtual Machine)

- VMware
- QEmu
- Bochs

Cons:

- Low density/scalability
- Slow/complex management
- Low performance

Pros:

- Can run arbitrary OS, unmodified

Para-virtualization

- Xen
- UML
  (User Mode Linux)

Multiple (modified) OSs run under a hypervisor (a.k.a. Virtual Machine Monitor), which shares the hardware resources between guests.

Pros:

- Better performance

Cons:

- Needs modified guest OS
- Static resource allocation, bad scalability, bad manageability

OS Level Virtualization

(OS == kernel)

- OpenVZ
- FreeBSD jails
- Linux-VServer
- Solaris Zones

Most applications running on a server can easily share a machine with others, if they could be isolated and secured. OS Virtualization provides the required isolation and security to run multiple applications or copies of the same OS on the same server.

Pros:

- Native performance
- Dynamic resource allocation, best scalability

Cons:

- Single (same) kernel per physical server

HYPERVISORS

Hypervisors

- *Also called as Virtual Machine Monitor (VMM), a* software that
  - securely partitions the resources of computer system into one or more virtual machines
  - Allows several operating systems to run concurrently on a single hardware platform.
- A *guest operating system* is an operating system that runs under the control of a VMM rather than directly on the hardware

Hypervisors

- Hypervisor enables:
  - Multiple services to share the same platform.
  - The movement of a server from one platform to another, the so-called live migration.
  - System modification while maintaining backward compatibility with the original system.

Hypervisor

- virtualizes the CPU and the memory.
- traps interrupts and dispatches them to the individual guest operating systems
- Traps the privileged instructions executed by a guest OS and enforces the correctness and safety of the operation.
- Controls the virtual memory management.
- maintains a *shadow page table* for each guest OS
  - replicates any modification made by the guest OS in its own shadow page table;
  - shadow page table points to the actual page frame and it is used by the hardware component called the *Memory Management Unit (MMU)* for dynamic address translation.
- Monitors the system performance and takes corrective actions to avoid performance degradation

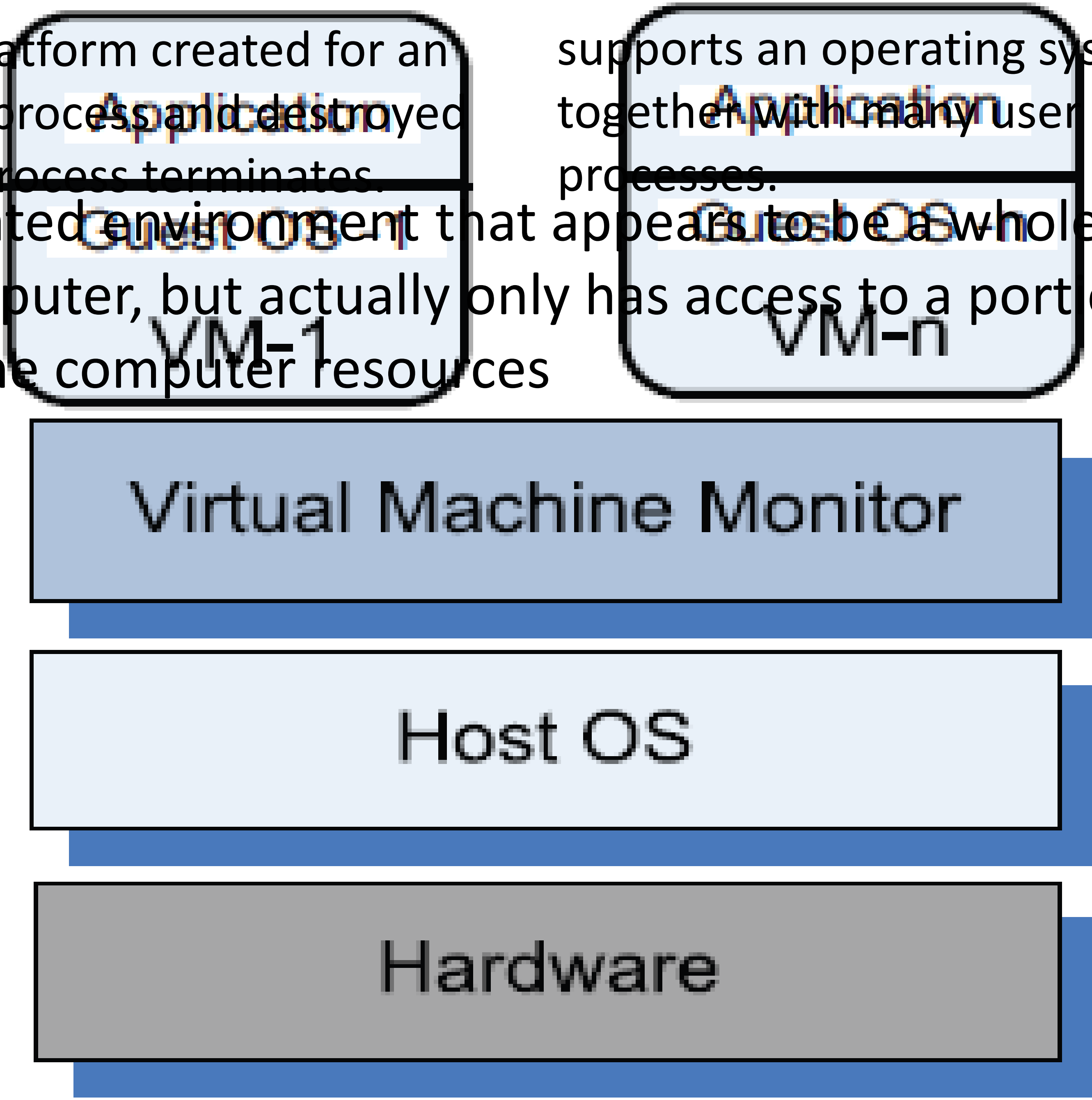
Memory virtualization

- Has important implications on the performance.
- VMMs use a range of optimization techniques
  - *VMware* systems avoid page duplication among different virtual machines
    - maintain only one copy of a shared page and use copy-on-write policies
  - Xen imposes total isolation of the VM and does not allow page sharing

a virtual platform created for an individual process and destroyed once the process terminates.

supports an operating system together with many user processes.

isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources



Name	Host ISA	Guest ISA	Host OS	guest OS	Company
Integrity VM	*x86*-64	*x86*-64	HP-Unix	Linux,Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Lynx Secure	*x86*	*x86*	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	*x86*-64	*x86*-64	Windows	Windows	Microsoft
Oracle VM	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows	Oracle
RTS Hypervisor	*x86*	*x86*	No host OS	Linux, Windows	Real Time Systems
SUN xVM	*x86*, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware EX Server	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	*x86*, *x86*-64	*x86*, *x86*-64	MAC OS *x86*	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	*x86*, *x86*-64	*x86*, *x86*-64	Linux Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	*x86*	*x86*	Denali	ILVACO, NetBSD	University of Washington
Xen	*x86*, *x86*-64	*x86*, *x86*-64	Linux Solaris	Linux, Solaris NetBSD	University of Cambridge

VMs

- Traditional
  - *VMWare ESX, ESXi* Servers, *Xen*, *OS370*, and *Denali*.
- Hybrid
  - *VMWare Workstation*.
- Hosted
  - User-mode Linux.

Performance and security isolation

- *Performance isolation* is a critical condition for Quality of Service (QoS) guarantees in shared computing environments
  - run-time behavior of an application is affected by other applications running concurrently.
- A VMM is a much simpler and better specified system than a traditional operating system.
- The security vulnerability of VMMs is considerably reduced as the systems expose a much smaller number of privileged functions.
- *Processor virtualization*
  - code is executed directly by the hardware
- *processor emulation*
  - presents a model of another hardware system. Instructions are “emulated” in software

Performance and security isolation

- Traditional operating systems multiplex multiple processes or threads
- VMM multiplexes full operating systems
  - Executes directly on the hardware a subset of frequently used machine instructions generated by the application and
  - emulates privileged instructions including device I/O requests.

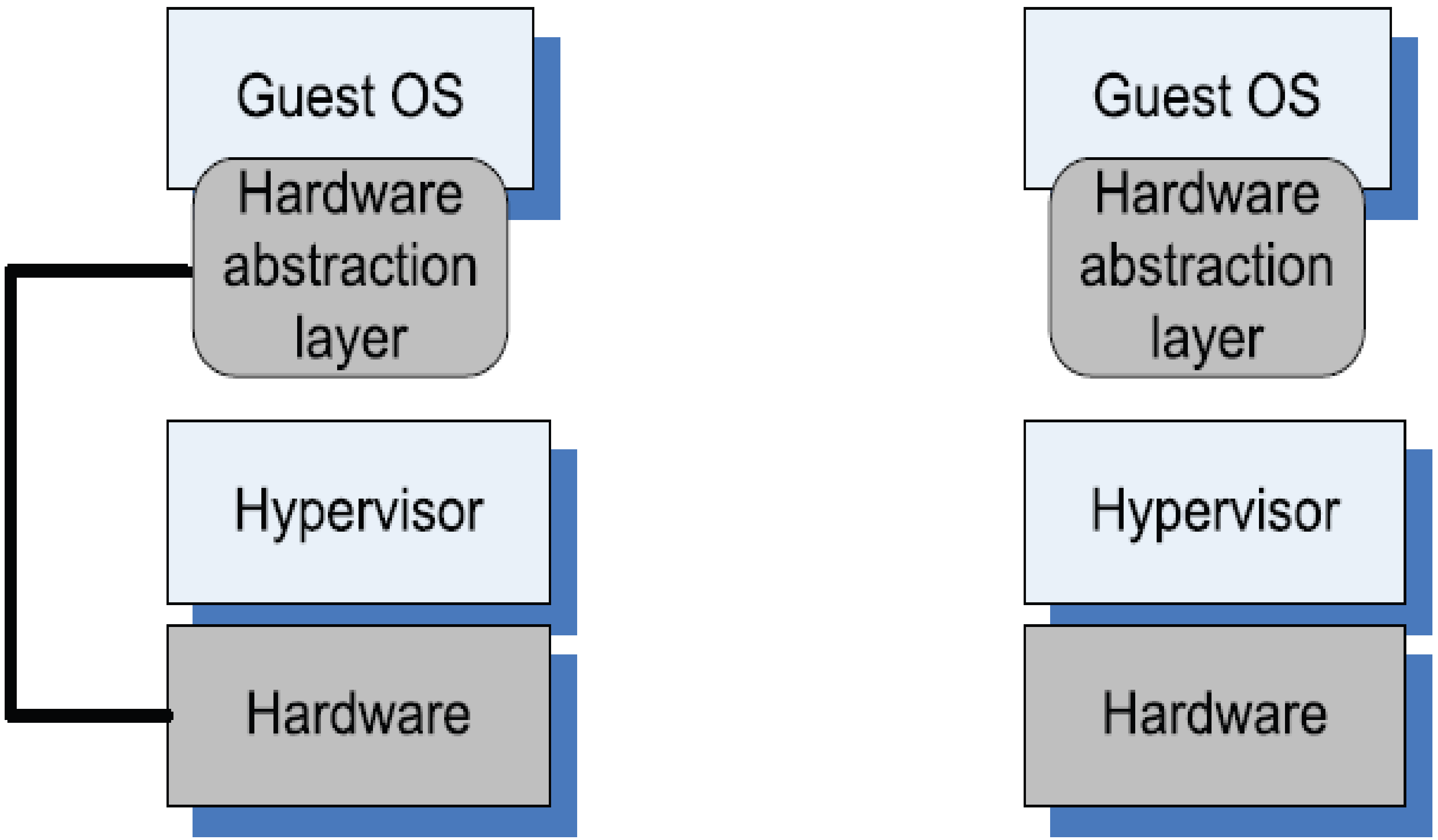
Performance and security isolation

- Operating systems use the process abstraction not only for resource sharing but also to support isolation.
  - this is not sufficient from a security perspective
- the software running on a virtual machine has the constraints of its own dedicated hardware
  - VMMs have potential to provide a level of isolation nearly equivalent to the isolation presented by two different physical systems

Computer architecture and virtualization

- Conditions for efficient virtualization:
  - Programs running under VMM should exhibit Identical Behaviror
  - The VMM should be in complete control of the virtualized resources
  - significant fraction of machine instructions must be executed without the intervention of the VMM
- Two classes of machine instructions
  - Sensitive [Control sensitive and mode sensitive]
  - Non-sensitive

Full virtualization and paravirtualization



(a) Full virtualization

(b) Paravirtualization

Full and Para virtualization

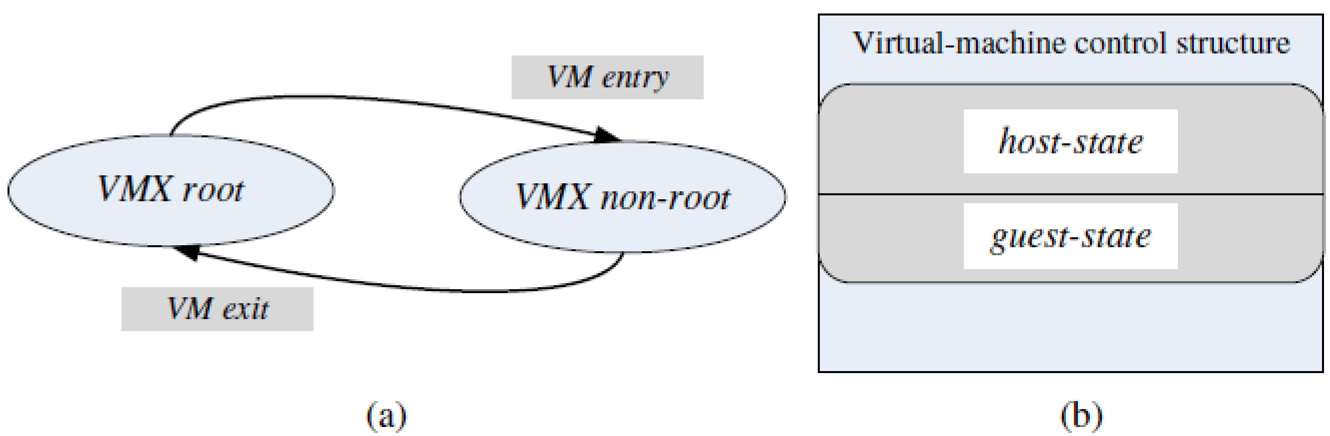
- Full
  - Requires a virtualizable architecture
  - Hardware is fully exposed to the guest OS which runs unchanged.
- Para
  - because some architectures such as *x86* are not easily virtualizable.
  - guest OS be modified to run under the VMM
  - guest OS code must be ported for individual hardware platforms.

Hardware support for virtualization

- Problems faced by virtualization of the *x86* architecture:
  - *Ring deprivileging:*
    - VMMs forces the guest software to run at privilege level > 0
  - *Ring aliasing*
    - problems created because of the previous. A guest OS is forced to run at undesired level
  - *Address space compression*
    - VMM uses parts of the guest address space to store several system data structures.
  - *Non-faulting access to privileged state*
    - OS executing one of the 'privileged' instructions does not realize that the instruction has failed
  - *Guest system calls*;
    - VMM must then emulate every guest execution of two instructions SYSENTER and SYSEXIT. (transitions to/from privilege level 0)
  - *Interrupt virtualization*
    - VMM generates a "virtual interrupt" – create overhead.
  - *Access to hidden state*
    - elements of the system state are hidden. Mechanism for saving and restoring difficult.
  - *Ring compression*
    - levels 1 and 2 cannot be used (64-bit mode can only use paging which needs 0/1/2)
  - *Frequent access to privileged resources increases VMM overhead*
    - task-priority register

Hardware support for virtualization

- Architectural enhancement provided by the *VT-x.*
  - Support for two modes of operations and
  - New data structure called the Virtual Machine Control Structure (VMCS) including *host-state* and *guest-state* areas



Hardware support for virtualization

- Processors based on two new virtualization architectures, *VT-d* and *VT-c* were developed.
  – *VT-d*: I/O Memory Management Unit (I/O MMU)
    - *gives VMs direct access to peripheral devices.*
    - *PCI pass-through*
  – *VT-c:* network virtualization

VT-d

- DMA address remapping,
  - address translation for device DMA transfers
- Interrupt remapping
  - isolation of device interrupts and VM routing
- I/O device assignment
  - devices can be assigned by an administrator to a VM in any configurations
- Reliability features,
  - Recording and reporting above.

XEN

Introduction

- Challenges to build virtual machines
  - Performance isolation
    - Scheduling priority
    - Memory demand
    - Network traffic
    - Disk accesses
  - Support for various OS platforms
  - Small performance overhead

Xen

- Multiplexes resources at the granularity of an entire OS
  - As opposed to process-level multiplexing
  - Price: higher overhead
- Target: 100 virtual OSes per machine

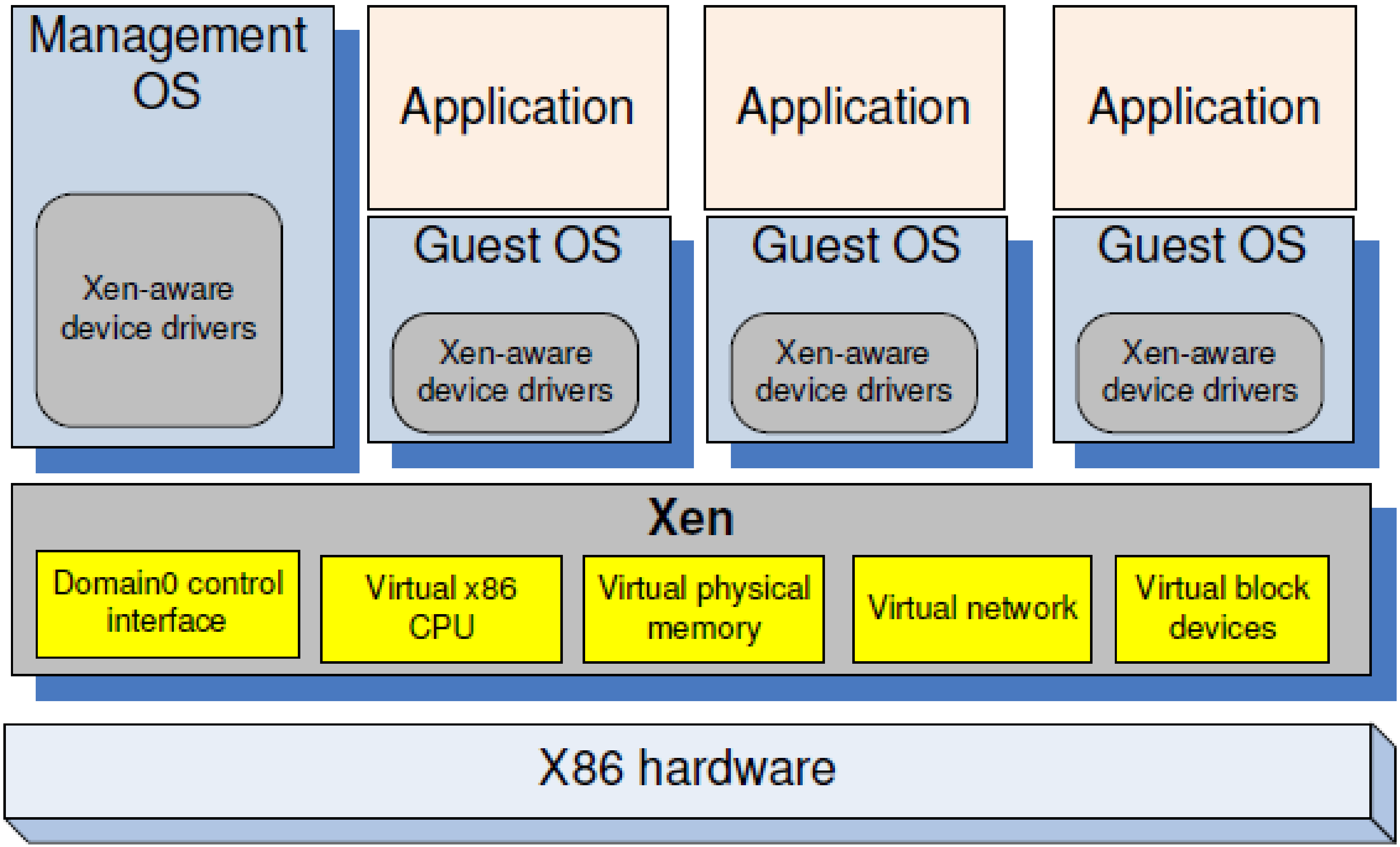
Xen: Approach and Overview

- Conventional approach
  - Full virtualization
    - Cannot access the hardware
    - Problematic for certain privileged instructions (e.g., traps)
    - No real-time guarantees

Xen: Approach and Overview

- Xen: paravirtualization
  - Provides some exposures to the underlying HW
    - Better performance
    - Need modifications to the OS
    - No modifications to applications

Xen Architecture



Dom0 components

- XenStore – a Dom0 process.
  - Supports a system-wide registry and naming service.
  - Implemented as a hierarchical key-value storage.
  - A watch function informs listeners of changes of the key in storage they have subscribed to.
  - Communicates with guest VMs via shared memory using Dom0 privileges.
- Toolstack - responsible for creating, destroying, and managing the resources and privileges of VMs.
  - To create a new VM, a user provides a configuration file describing memory and CPU allocations and device configurations.
  - Toolstack parses this file and writes this information in XenStore.
  - Takes advantage of Dom0 privileges to map guest memory, to load a kernel and virtual BIOS and to set up initial communication channels with XenStore and with the virtual console when a new VM is created.

FEATURES

Function	Strategy
Paging	A domain may be allocated discontinuous pages. A guest OS has direct access to page tables and handles pages faults directly for efficiency; page table updates are batched for performance and validated by *Xen* for safety.
Memory	Memory is statically partitioned between domains to provide strong isolation. `XenoLinux` implements a *balloon driver* to adjust domain memory.
Protection	A guest OS runs at a lower priority level, in ring 1, while *Xen* runs in ring 0.
Exceptions	A guest OS must register with *Xen* a description table with the addresses of exception handlers previously validated; exception handlers other than the page fault handler are identical with *x86* native exception handlers.
System calls	To increase efficiency, a guest OS must install a "fast" handler to allow system calls from an application to the guest OS and avoid indirection through *Xen*.
Interrupts	A lightweight event system replaces hardware interrupts; synchronous system calls from a domain to *Xen* use *hypercalls* and notifications are delivered using the asynchronous event system.
Multiplexing	A guest OS may run multiple applications.
Time	Each guest OS has a timer interface and is aware of "real" and "virtual" time.
Network and I/O devices	Data is transferred using asynchronous I/O rings; a ring is a circular queue of descriptors allocated by a domain and accessible within *Xen*.
Disk access	Only *Dom0* has direct access to IDE and SCSI disks; all other domains access persistent storage through the Virtual Block Device (VBD) abstraction.

Memory Management

- Depending on the hardware, supports
  - Software managed TLB (Translation Look-aside Buffer)
    - A cache for page table entries
    - Associate address space IDs with TLB tags
    - Allow coexistence of OSes
    - Avoid TLB flushing across OS boundaries

Memory Management

- X86 does not have software managed TLB
  - Xen exists at the top 64MB of every address space
  - Avoid TLB flushing when an guest OS enter/exist Xen
  - Each OS can only map to memory it owns
  - Writes are validated by Xen

CPU

- X86 supports 4 levels of privileges
  - 0 for OS, and 3 for applications
  - Xen downgrades the privilege of OSes
  - System-call and page-fault handlers registered to Xen
  - "fast handlers" for most exceptions, Xen isn't involved

Device I/O

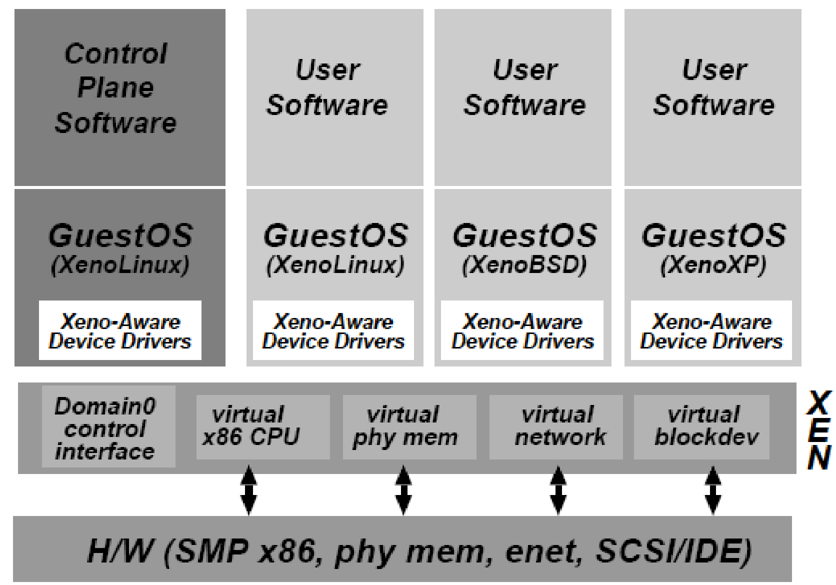
- Xen exposes a set of simple device abstractions

The Cost of Porting an OS to Xen

- Privileged instructions
- Page table access
- Network driver
- Block device driver
- <2% of code-base

Control Management

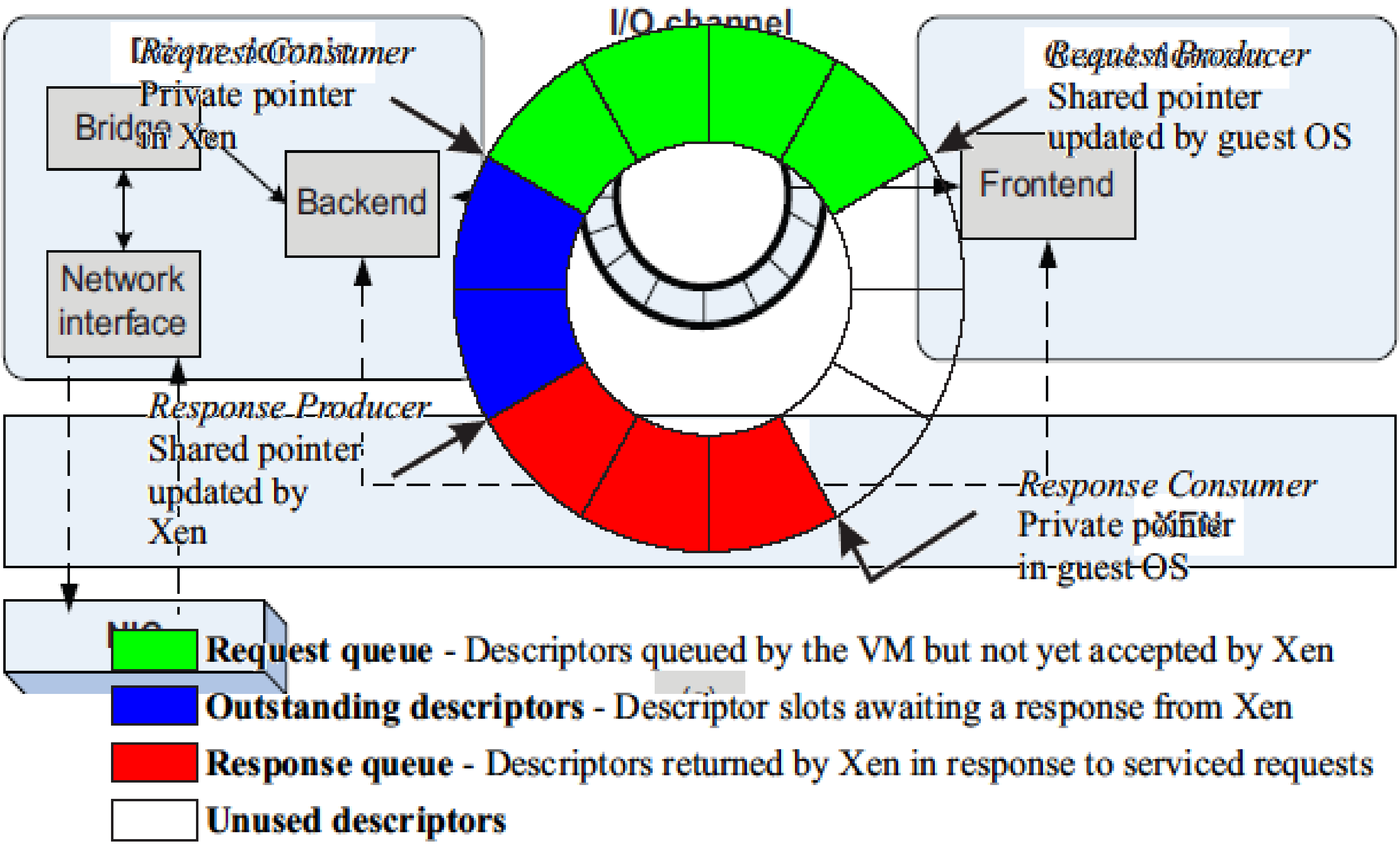
- Separation of policy and mechanism
- Domain0 hosts the application-level management software
  - Creation and deletion of virtual network interfaces and block devices



Control Transfer: Hypercalls and Events

- Hypercall: synchronous calls from a domain to Xen
  - Analogous to system calls
- Events: asynchronous notifications from Xen to domains
  - Replace device interrupts

Data Transfer: I/O Rings



CPU Scheduling

- Borrowed virtual time scheduling
  - Allows temporary violations of fair sharing to favor recently-woken domains
  - Goal: reduce wake-up latency

Time and Timers

- Xen provides each guest OS with
  - Real time (since machine boot)
  - Virtual time (time spent for execution)
  - Wall-clock time
- Each guest OS can program a pair of alarm timers
  - Real time
  - Virtual time

Virtual Address Translation

- No shadow pages (VMWare)
- Xen provides constrained but direct MMU updates
- All guest OSes have read-only accesses to page tables
- Updates are batched into a single hypercall

Physical Memory

- Reserved at domain creation times
- Memory statically partitioned among domains

Network

- Virtual firewall-router attached to all domains
- Round-robin packet scheduler
- To send a packet, enqueue a buffer descriptor into the transmit rang
- Use scatter-gather DMA (no packet copying)
  - A domain needs to exchange page frame to avoid copying
  - Page-aligned buffering

Disk

- Only Domain0 has direct access to disks
- Other domains need to use virtual block devices
  - Use the I/O ring
  - Reorder requests prior to enqueuing them on the ring
  - If permitted, Xen will also reorder requests to improve performance
- Use DMA (zero copy)

A performance comparison of virtual machines: Xen Vs OpenVZ

- VMM such as *Xen* introduces additional overhead and affects negatively the performance.
- compare the performance of two virtualization techniques with a standard operating system

OPENVZ

OSs evolution

- **Multi**task
  many processes

- **Multi**user

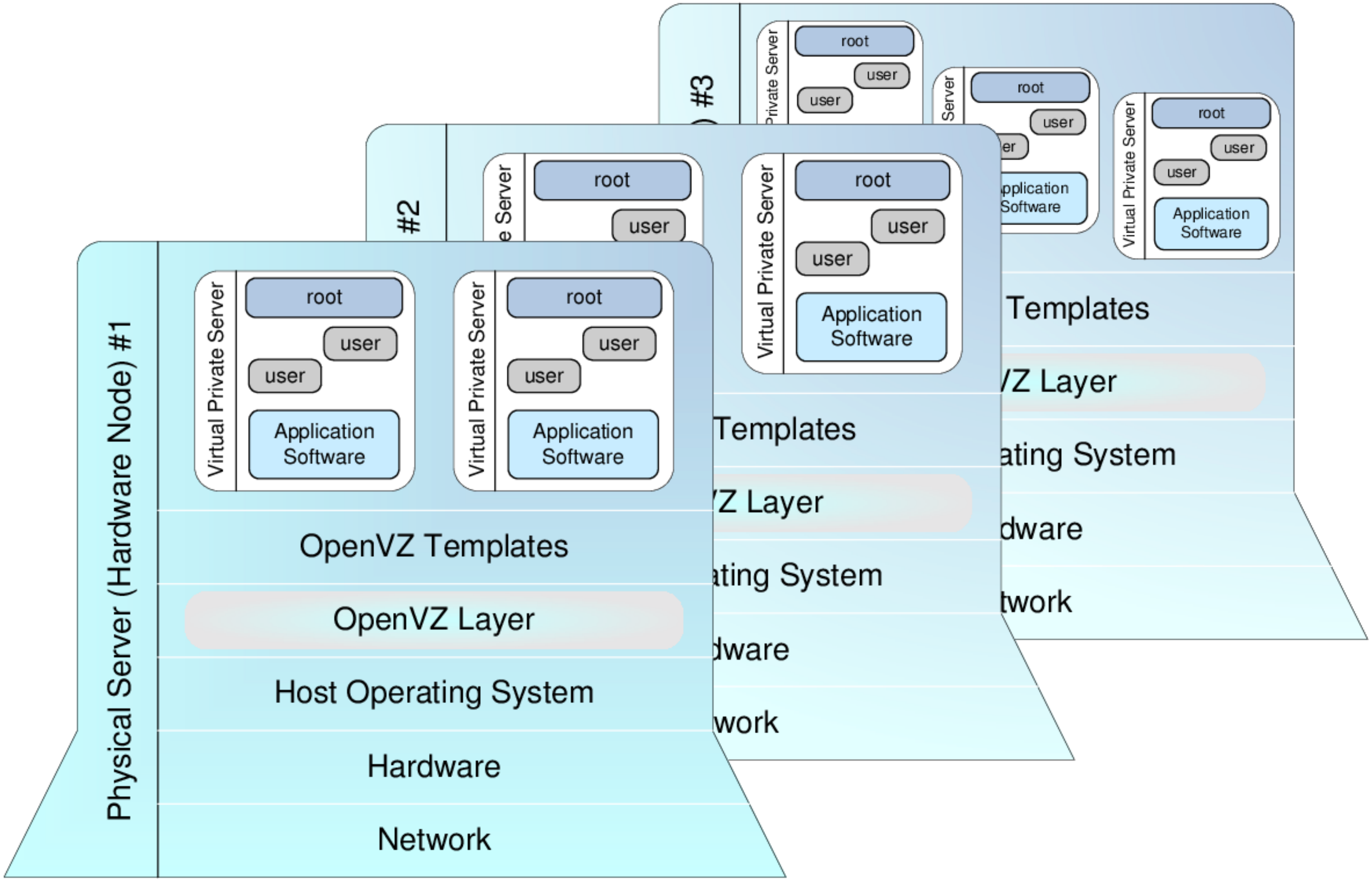
  many users

- **Multi**ple execution environments

  many Virtual Private Servers

  (VPSs, containers, guests, partitions...)

OpenVZ design approach



OpenVZ: components

- Kernel
  - Isolation
  - Virtualization
  - Resource Management
- Tools
  - vzctl: Virtual Private Server (VPS) control utility
  - vzpkg: VPS software package management
- Templates
  - precreated VPS images for fast VPS creation

Kernel: Virtualization & Isolation

Each VPS has its own

- Files
  System libraries, applications, virtualized /proc and /sys, virtualized locks etc.
- Process tree
  Featuring virtualized PIDs, so that the init PID is 1
- Network
  Virtual network device, its own IP addresses, set of netfilter and routing rules
- Devices
  If needed, any VPS can be granted access to real devices like network interfaces, serial ports, disk partitions, etc.
- IPC objects
  shared memory, semaphores, messages
- …

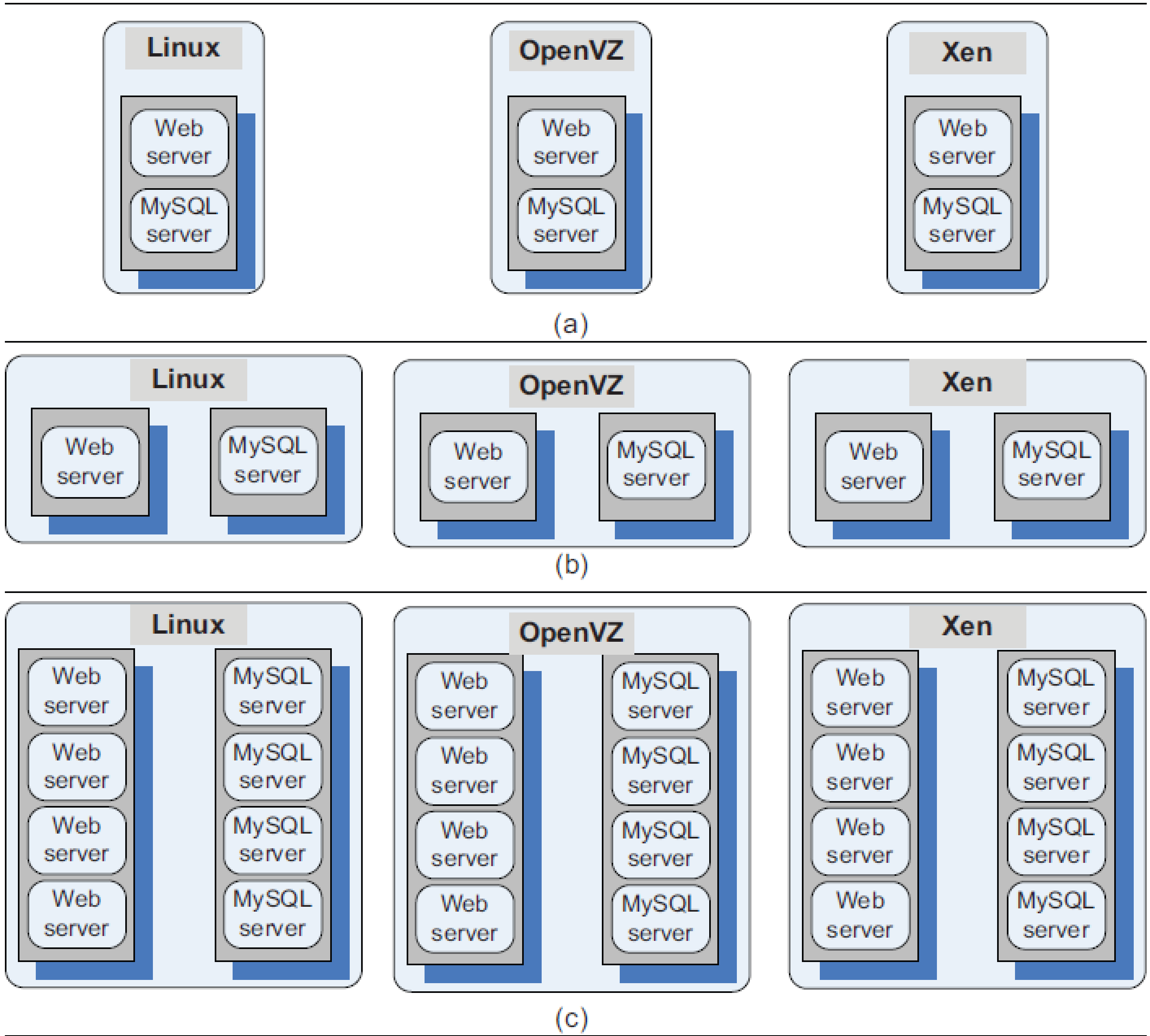
Kernel: Resource Management

Managed resource sharing and limiting.

- **User Beancounters** is a set of per-VPS resource counters, limits, and guarantees (kernel memory, network buffers, phys pages, etc.)
- **Fair CPU scheduler** (SFQ with shares and hard limits)
- **Two-level disk quota** (first-level: per-VPS quota; second-level: ordinary user/group quota inside a VPS)

Resource management is what makes OpenVZ different from other technologies.

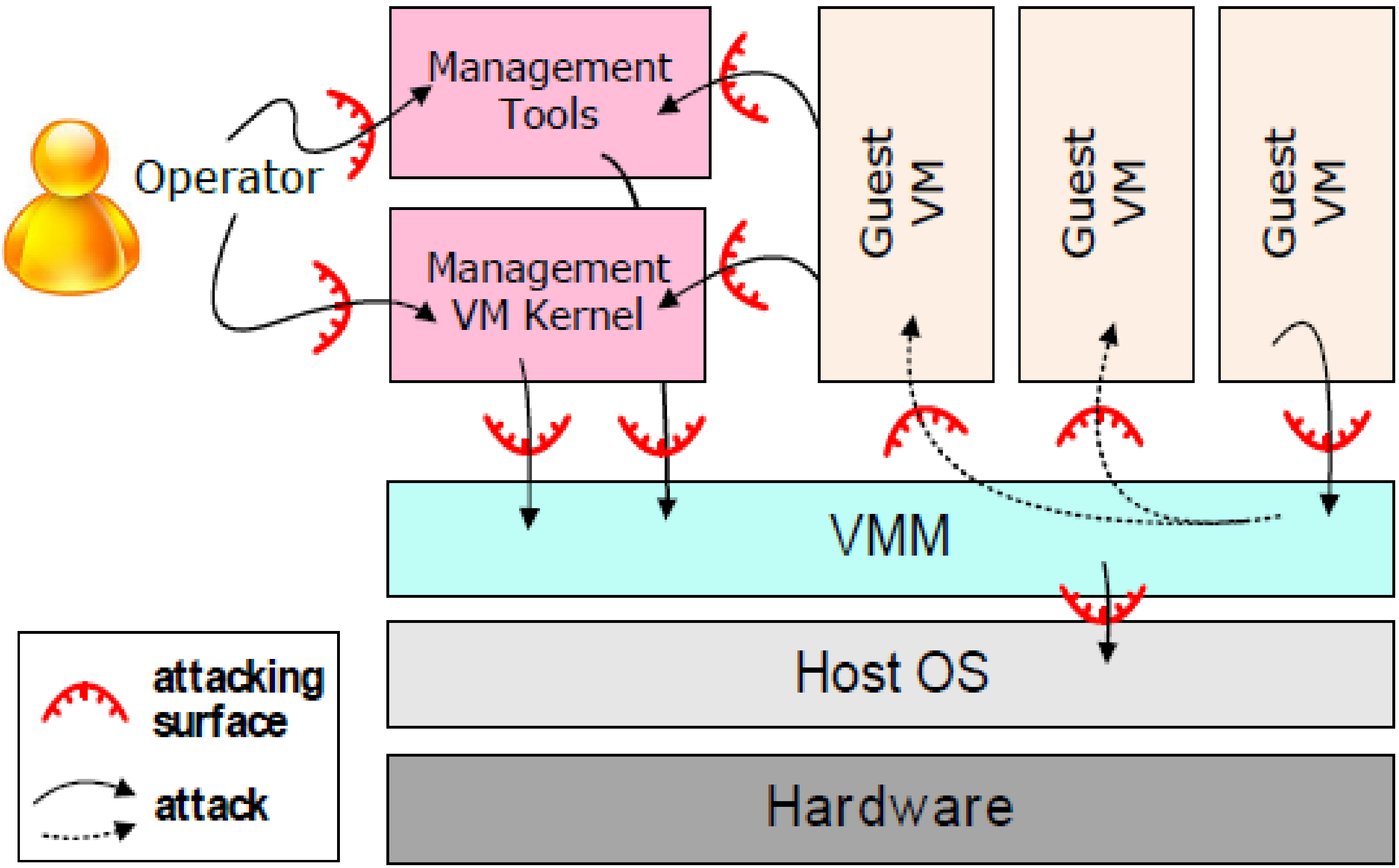
COMPARISON


Linux
Web server
MySQL server
OpenVZ
Web server
MySQL server
Xen
Web server
MySQL server
(a)
Linux
Web server
MySQL server
OpenVZ
Web server
MySQL server
Xen
Web server
MySQL server
(b)
Linux
Web server
Web server
Web server
Web server
MySQL server
MySQL server
MySQL server
MySQL server
OpenVZ
Web server
Web server
Web server
Web server
MySQL server
MySQL server
MySQL server
MySQL server
Xen
Web server
Web server
Web server
Web server
MySQL server
MySQL server
MySQL server
MySQL server
(c)

Main Conclusions

- overhead of *Xen* is considerably higher than that of *OpenVZ* and that this is due primarily to L2-cache misses.
- The performance degradation when the workload increases is also noticeable for *Xen*.
- hosting multiple tiers of the same application on the same server is not an optimal solution.

VMM VULNERABILITIES



Operator
Management Tools
Management VM Kernel
Guest VM
Guest VM
Guest VM
VMM
Host OS
Hardware
attacking surface
attack

Virtualization and Vulnerabilities

- Detecting a virtualized environment.
- Identifying the hypervisor.
- Breach in the isolation.
  - Denial of service:
  - System halt:
  - VM escape:
- The concept of the network perimeter evaporates
  - no physical segregation across VMs

Virtualization and Vulnerabilities

- The public cloud provides user access via the Internet
  - Cloud subscribers conduct administrative activities
- Cyber attacker or malware can exploit the vulnerabilities remotely throughout physical and virtual enterprise
- Virtual Machine based rootkits
  - Blue Pill, subVert

Virtualization and Vulnerabilities

- Increases the risk of VM-to-VM vulnerability exploitation
  - Colocation of VMs
  - Remote user on one VM can access another dormant VM if both reside on the same physical server
    - Malware attacks can be generated as malware scans are not done on dormant machines
- Easy reconfiguration
  - Creates an environment to propagate vulnerabilities and unknown configuration errors
- These attacks can also affect other physical devices in the cloud

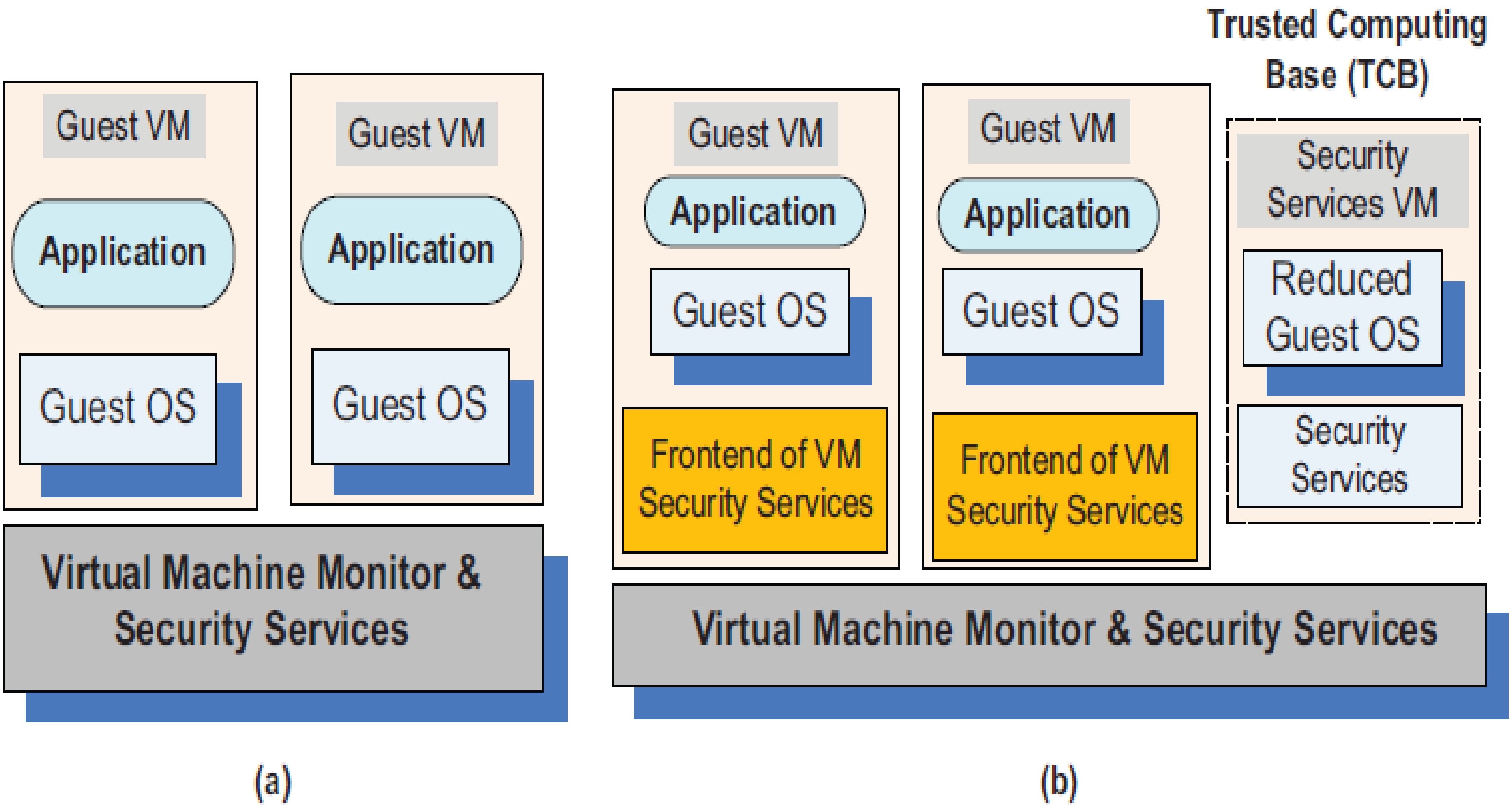
Virtual machine security

- Restricted to the traditional system VM model
  - Virtual Machine Monitor controls the access to the hardware
- Virtual security services are typically provided by the VMM

OR

- have a dedicated security services VM

Virtual machine security



VMM-based threats

- Starvation of resources and denial of service for some VMs.
  - *badly configured resource limits for some VMs;*
  - *a rogue VM with the capability to bypass resource limits set in VMM.*
- VM side-channel attacks [malicious attack on one or more VMs by a rogue VM]
  - *lack of proper isolation of inter-VM - due to misconfiguration of the virtual network*
  - *limitation of packet inspection devices*
  - *presence of VM instances built from insecure VM images*
- Buffer overflow attacks

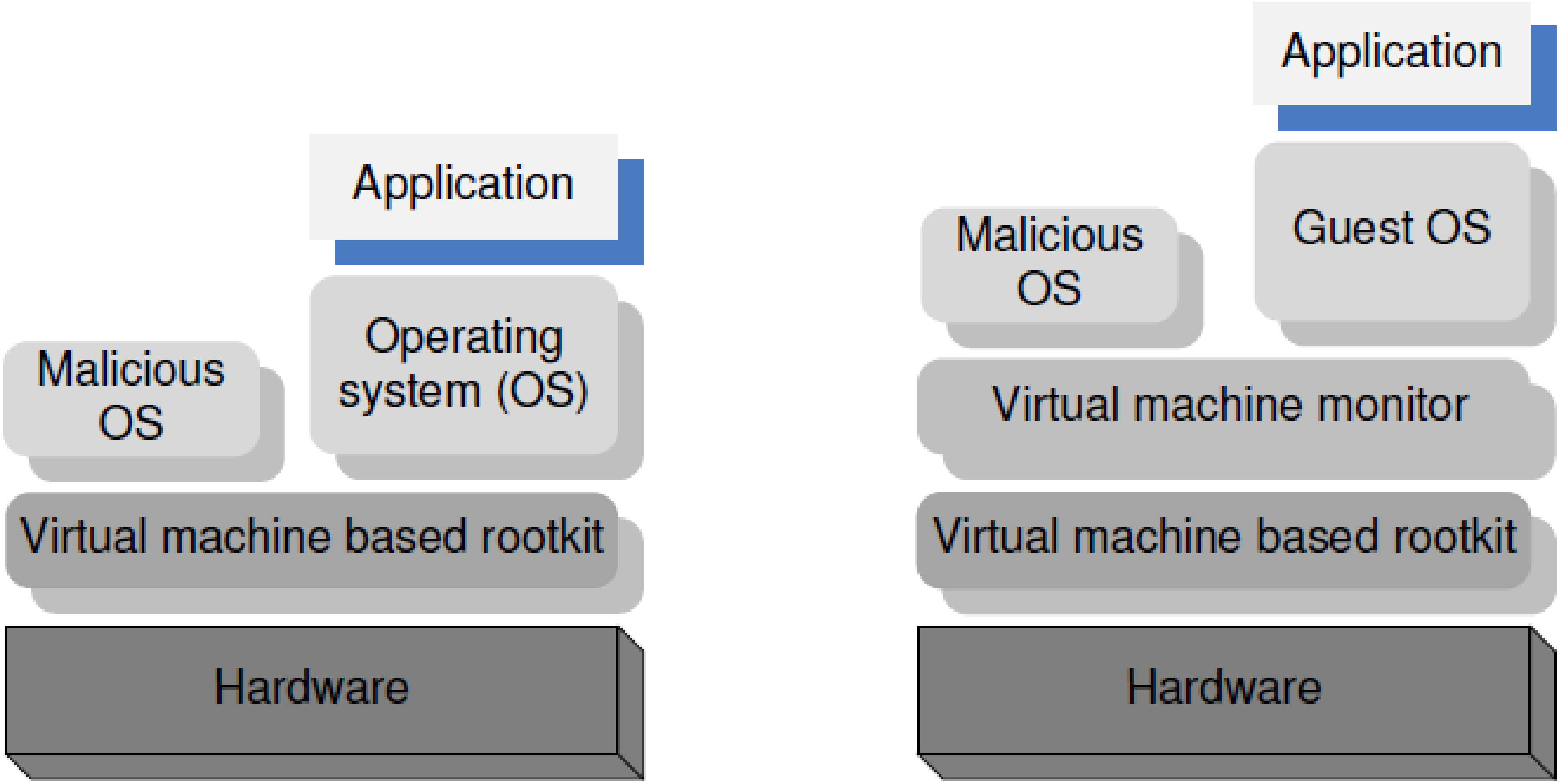
VM-based threats

- Deployment of rogue or insecure VM;
  - *improper configuration of access controls on VM administrative tasks such as instance creation, launching, suspension*
- Presence of insecure and tampered VM images in the VM image repository
  - *lack of access control to the VM image repository*
  - *Lack of mechanisms to verify the integrity of the images,*

The darker side of virtualization

- virtualization empower the creators of malware
  - in a layered structure a defense mechanism at some layer can be disabled by malware running at a layer below it
  - Focus: lowest layer of the software stack, the one which controls the hardware
- insert a “rogue VMM” between the physical hardware and an operating system

The darker side of virtualization



VMBR

- The term *rootkit* refers to malware with a privileged access to a system;
- insert the VMBK between the physical hardware and a “legitimate VMM.”
- malware runs either inside a VMM or with the support of a VMM;
  - VMM is a very potent engine for the malware
- only way for a VMBR to take control of a system is to modify the boot sequence and to first load the malware and only then load the legitimate VMM,

THANKS