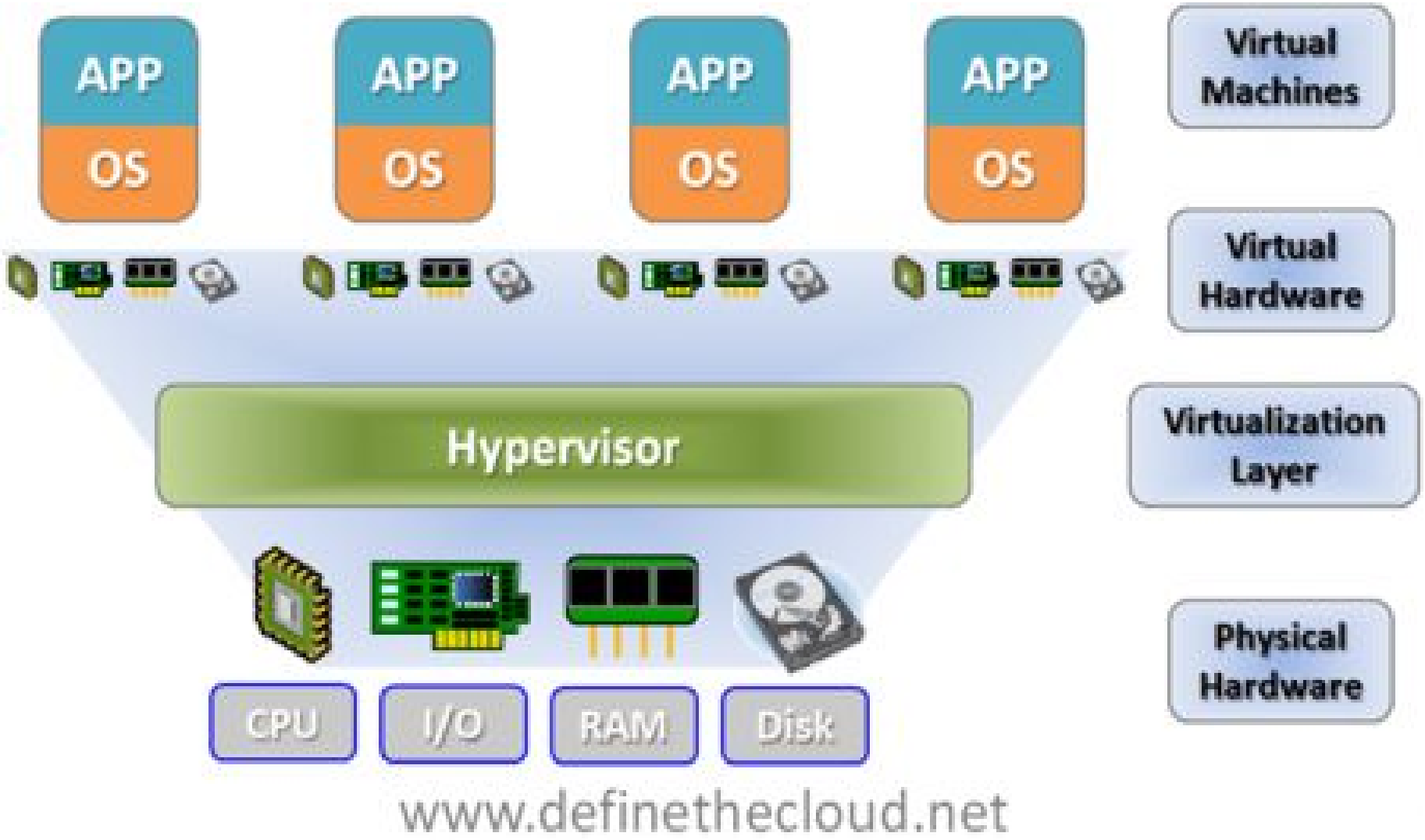
Cloud Resource Virtualization



CHAPTER 04

Contents

- Virtualization.
- Layering and virtualization.
- Virtual machine monitor.
- Virtual machine.
- Performance and security isolation.
- Full and para virtualization.
- H/W support for virtualization.
- x86 support for virtualization.
- Case study: Xen 2.0.
- Performance comparison of virtual machine monitors.
- Software fault isolation.

Motivation

- Three classes of fundamental abstractions – interpreters, memory, and communications links are necessary to describe the operation of a computing system.
- The physical realization of each one of these abstractions, such as processors that transform information, primary and secondary memory for storing information, and communication channels that allow different systems to communicate with one another, can vary in terms of bandwidth, latency, reliability, and other physical characteristics

Motivation

- Software systems such as operating systems are responsible for the management of the system resources – the physical implementations of the three abstractions.
- The main task is resource management and resource management grows increasingly complex as the scale of a system as well as the number of users and the diversity of applications using the system increase.

Motivation

- The traditional solution for a data center is to install standard operating systems on individual systems and rely on conventional OS techniques to ensure resource sharing, application protection, and performance isolation.
- System administration, accounting, security, and resource management are very challenging for the providers of service in this setup

Virtualization (alternative solution)

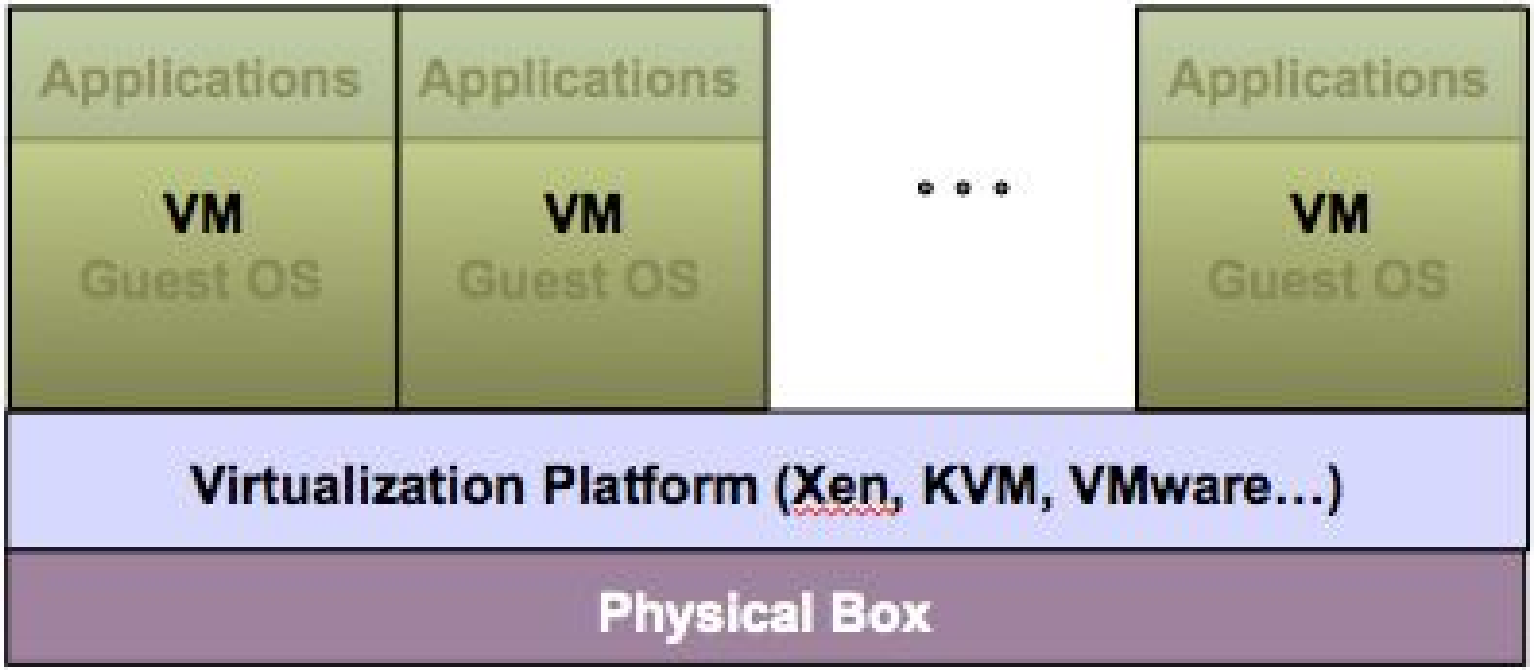
- **Virtualization** is the creation of a virtual -- rather than actual -- version of something, such as an operating system, a server, a storage device or network resources.
- Virtualization is a basic tenet of cloud computing – that simplifies some of the resource management tasks.
- For example, the state of a virtual machine (VM) running under a virtual machine monitor (VMM) can be saved and migrated to another server to balance the load.

Virtualization

- It is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources.
- It is the process by which one computer hosts the appearance of many computers.
- Virtualization is used to improve IT throughput and costs by using physical resources as a pool from which virtual resources can be allocated.

Virtualization Architecture

- A Virtual machine (VM) is an isolated runtime environment (guest OS and applications)
- Multiple virtual systems (VMs) can run on a single physical system



Resources for sharing among VMs

- CPU cycles,
- memory,
- secondary storage,
- I/O
- communication bandwidth

for each VM, resources must be shared among multiple instances of an application

Benefits of Virtualization

- Sharing of resources helps cost reduction
- Isolation: Virtual machines are isolated from each other as if they are physically separated
- Encapsulation: Virtual machines encapsulate a complete computing environment
- Hardware Independence: Virtual machines run independently of underlying hardware
- Portability: Virtual machines can be migrated between different hosts.

Virtualization in Cloud Computing

- Cloud computing takes virtualization one step further:
  - You don’t need to own the hardware
  - Resources are rented as needed from a cloud
  - Various providers allow creating virtual servers
  - You get billed only for what you used

Virtualization in CC

- Simulates the interface to a physical object by:
  - Multiplexing: creates multiple virtual objects from one instance of a physical object. Example - a processor is multiplexed among a number of processes or threads.
  - Aggregation: creates one virtual object from multiple physical objects. Example - a number of physical disks are aggregated into a RAID disk.

Virtualization in CC

- Simulates the interface to a physical object by:
  - Emulation: constructs a virtual object from a different type of a physical object. Example - a physical disk emulates a Random Access Memory (RAM).
  - Multiplexing and emulation. Examples - virtual memory with paging multiplexes real memory and disk; a virtual address emulates a real address.

Virtualization in CC

- Virtualization abstracts the underlying resources and simplifies their use, isolates users from one another, and supports replication, which, in turn, increases the elasticity of the system.
- Virtualization is a critical aspect of cloud computing, equally important to the providers and consumers of cloud services.

Virtualization plays an important role in:

- 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 in CC

- In a cloud computing environment a VMM runs on the physical hardware and exports hardware level abstractions to one or more guest operating systems.
- A guest OS interacts with the virtual hardware in the same way it would interact with the physical hardware, but under the watchful eye of the VMM which traps all privileged operations and mediates the interactions of the guest OS with the hardware.

Side effects of virtualization

- Performance penalty
- Hardware costs

} Justify your answer

Layering and virtualization

- Layering – a common approach to manage system complexity with well-defined *interfaces* among them.
  - Minimizes the interactions among the subsystems of a complex system.
  - Simplifies the description of the subsystems; each subsystem is abstracted through its interfaces with the other subsystems.
  - We are able to design, implement, and modify the individual subsystems independently.

Layering in a computer system

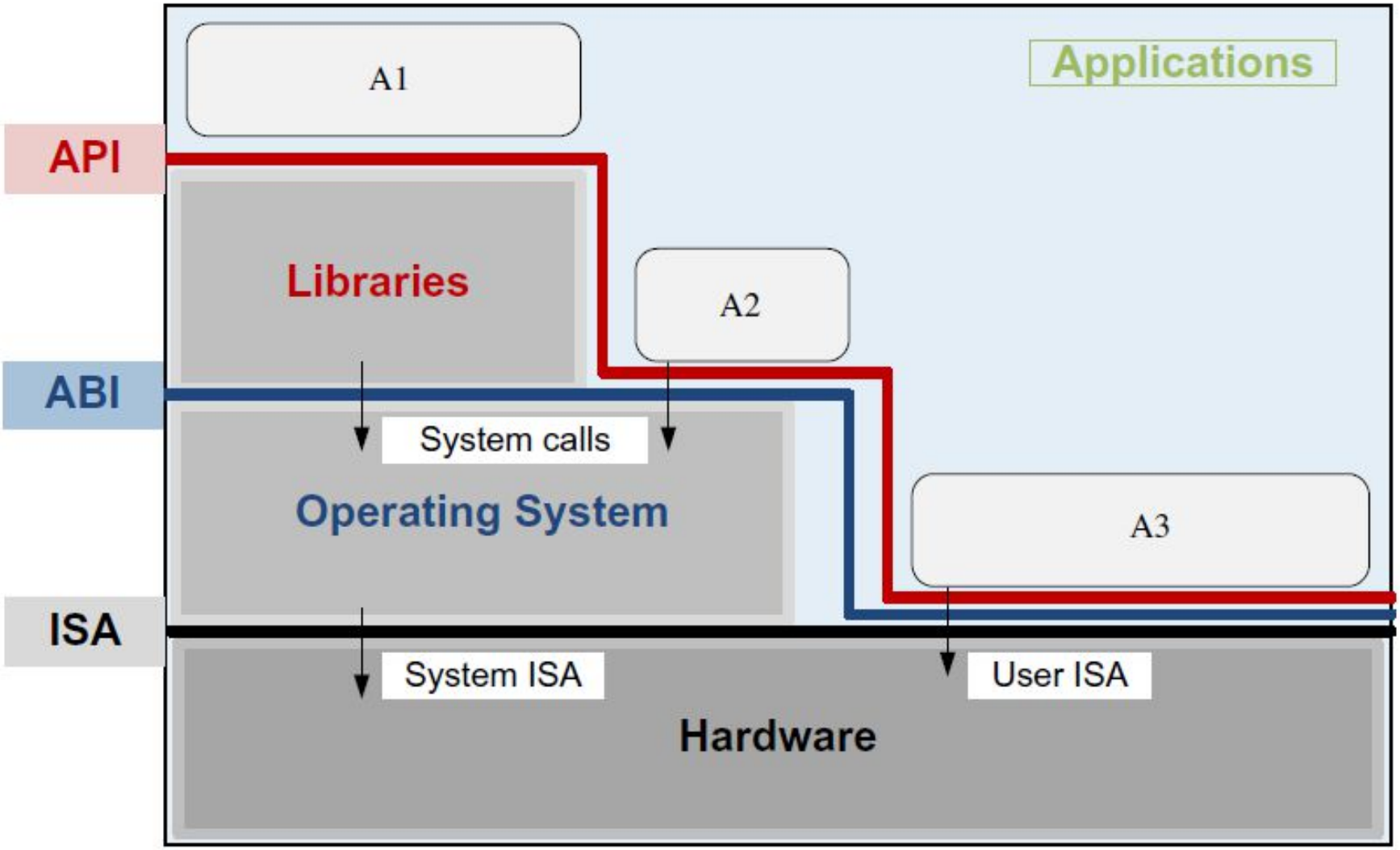
- Hardware.
- Software.
  - Operating system.
  - Libraries.
  - Applications.

Interfaces

- Instruction Set Architecture (ISA) – at the boundary between hardware and software.
- Application Binary Interface (ABI) – allows the ensemble consisting of the application and the library modules to access the hardware; the ABI does not include privileged system instructions, instead it invokes system calls.
- Application Program Interface (API) - defines the set of instructions the hardware was designed to execute and gives the application access to the ISA; it includes HLL library calls which often invoke system calls.

Note

- A process is the abstraction for the code of an application at execution time.
- A thread is a lightweight process.
- The ABI is the projection of the computer system seen by the process,
- The API is the projection of the system from the perspective of the HLL program.



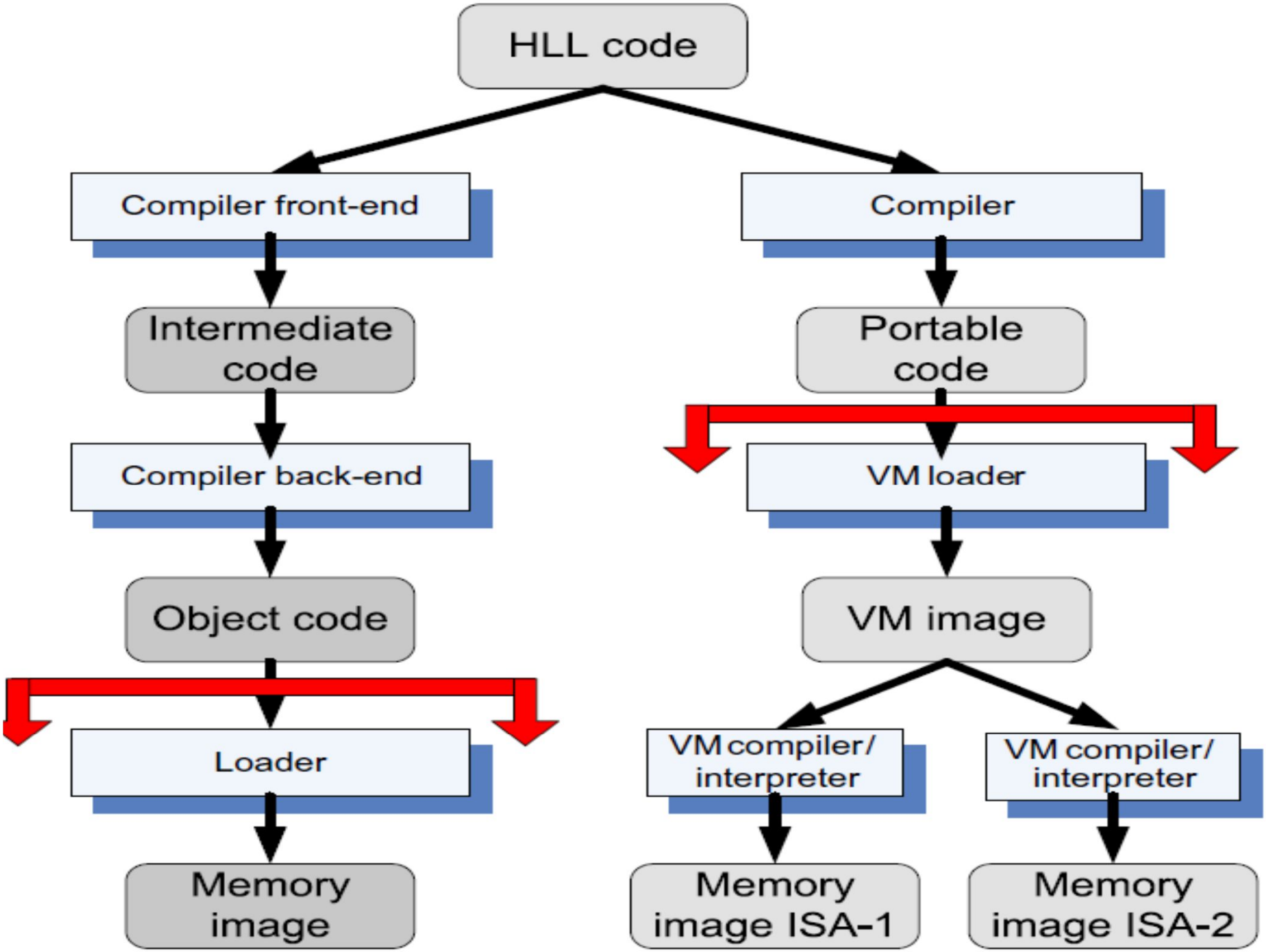
The software components, including applications, libraries, and operating system, interact with the hardware via several interfaces: API, ABI, ISA. An application uses library functions (A1), makes system calls (A2), and executes machine instructions (A3).

Code portability

- The binaries created by a compiler for a specific ISA and a specific operating system are not portable.
- Such code cannot run on a computer with a different ISA or on computers with the same ISA but different operating systems.
- However, it is possible to compile an HLL program for a VM environment, where portable code is produced and distributed and then converted by binary translators to the ISA of the host system.

Code portability

- A dynamic binary translation converts blocks of instructions from the portable code to the host instruction and leads to a significant performance improvement as such blocks are cached and reused.


HLL code
Compiler front-end
Intermediate code
Compiler back-end
Object code
Loader
Memory image
Compiler
Portable code
VM loader
VM image
VM compiler/ interpreter
VM compiler/ interpreter
Memory image ISA-1
Memory image ISA-2

Virtual machine monitors (VMM)

- It is also called a hypervisor.
- It is the software that securely partitions the resources of a computer system into one or more virtual machines.
- A guest operating system is an operating system that runs under the control of a VMM rather than directly on the hardware.
- The VMM runs in kernel mode, whereas a guest OS runs in user mode.

VMM

- VMMs allow several operating systems to run concurrently on a single hardware platform.
- VMMs enforce isolation among these systems, thus ensures security and encapsulation.
- A VMM controls how the guest operating system uses the hardware resources. The events occurring in one VM do not affect any other VM running under the same VMM.
- The VMM monitors system performance and takes corrective action to avoid performance degradation;

VMM

- The VMM monitors system performance and takes corrective action to avoid performance degradation;
  - for example, the VMM may swap out a VM(copies all pages of that VM from real memory to disk and makes the real memory frames available for paging by other VMs) to avoid thrashing.

VMM enables:

- Multiple services to share the same platform.
- Live migration - the movement of a server from one platform to another.
- System modification while maintaining backward compatibility with the original system.
- Enforces isolation among the systems, thus security.

VMM virtualizes the CPU and the memory

- Traps the privileged instructions executed by a guest OS and enforces the correctness and safety of the operation.
- The VMM traps interrupts and dispatches them to the individual guest operating systems.
- VMMs control the virtual memory management and decide what pages to swap out.

VMM virtualizes the CPU and the memory

- Maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table. This shadow page table points to the actual page frame and it is used by the Memory Management Unit (MMU) for dynamic address translation.
- Monitors the system performance and takes corrective actions to avoid performance degradation. For example, the VMM may swap out a Virtual Machine to avoid thrashing.

Virtual machines (VMs)

- VM - isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources.
- **Virtual machine** is a software implementation of a physical machine.
- Each VM appears to be running on the bare hardware, giving the appearance of multiple instances of the same computer.

Two types of VMs

- Process VM - a virtual platform created for an individual process and destroyed once the process terminates. All operating systems provide a process VM for each one of the applications execution. Process virtual machines are designed to execute computer programs in a platform-independent environment.
- System VM - It provides a a substitute for a real machine (complete system); each VM can run its own OS, which in turn can run multiple applications.

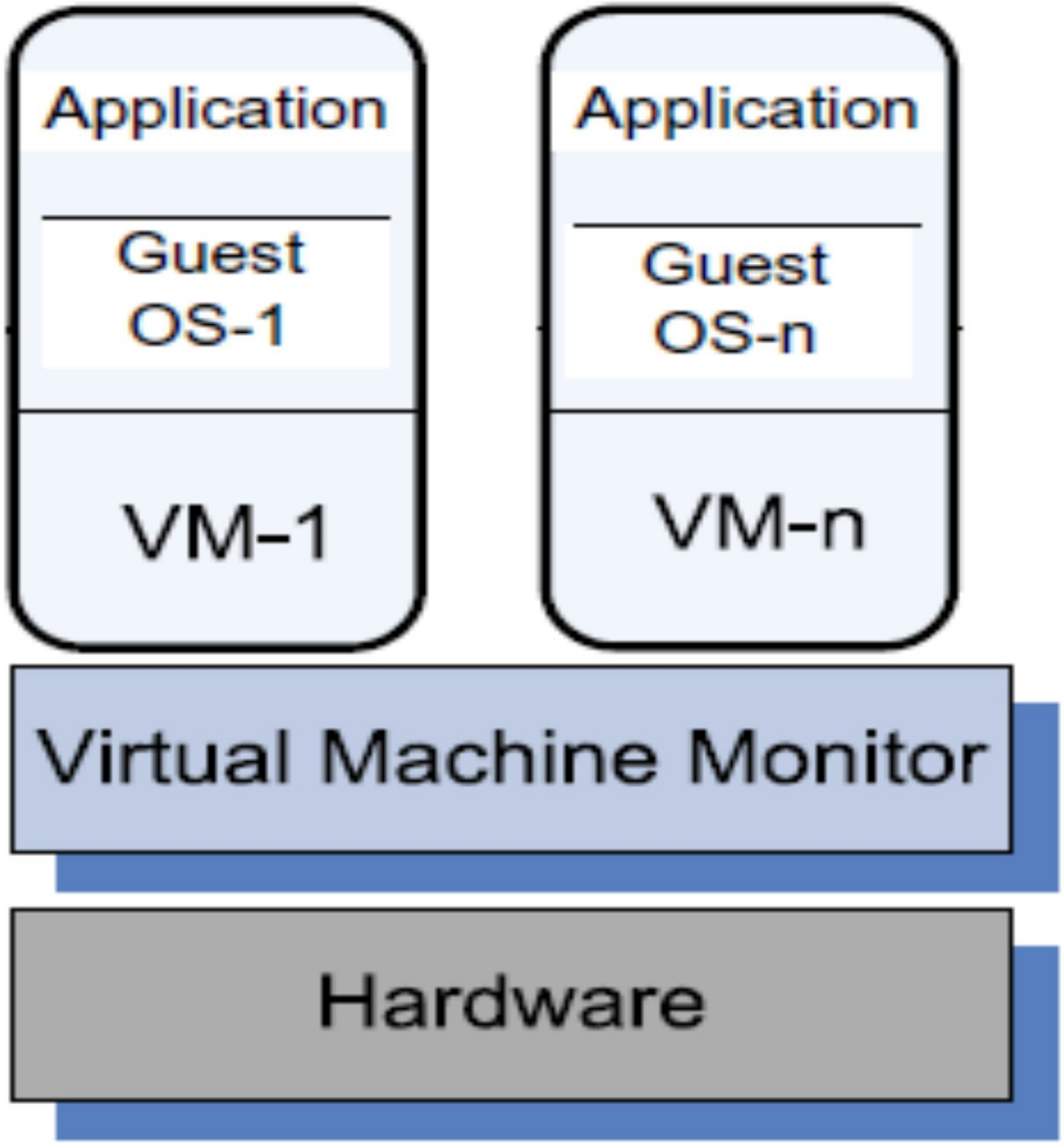
Application virtual machine

- An application virtual machine (e.g., Java Virtual Machine [JVM]) runs under the control of a normal OS and provides a platform-independent host for a single application.

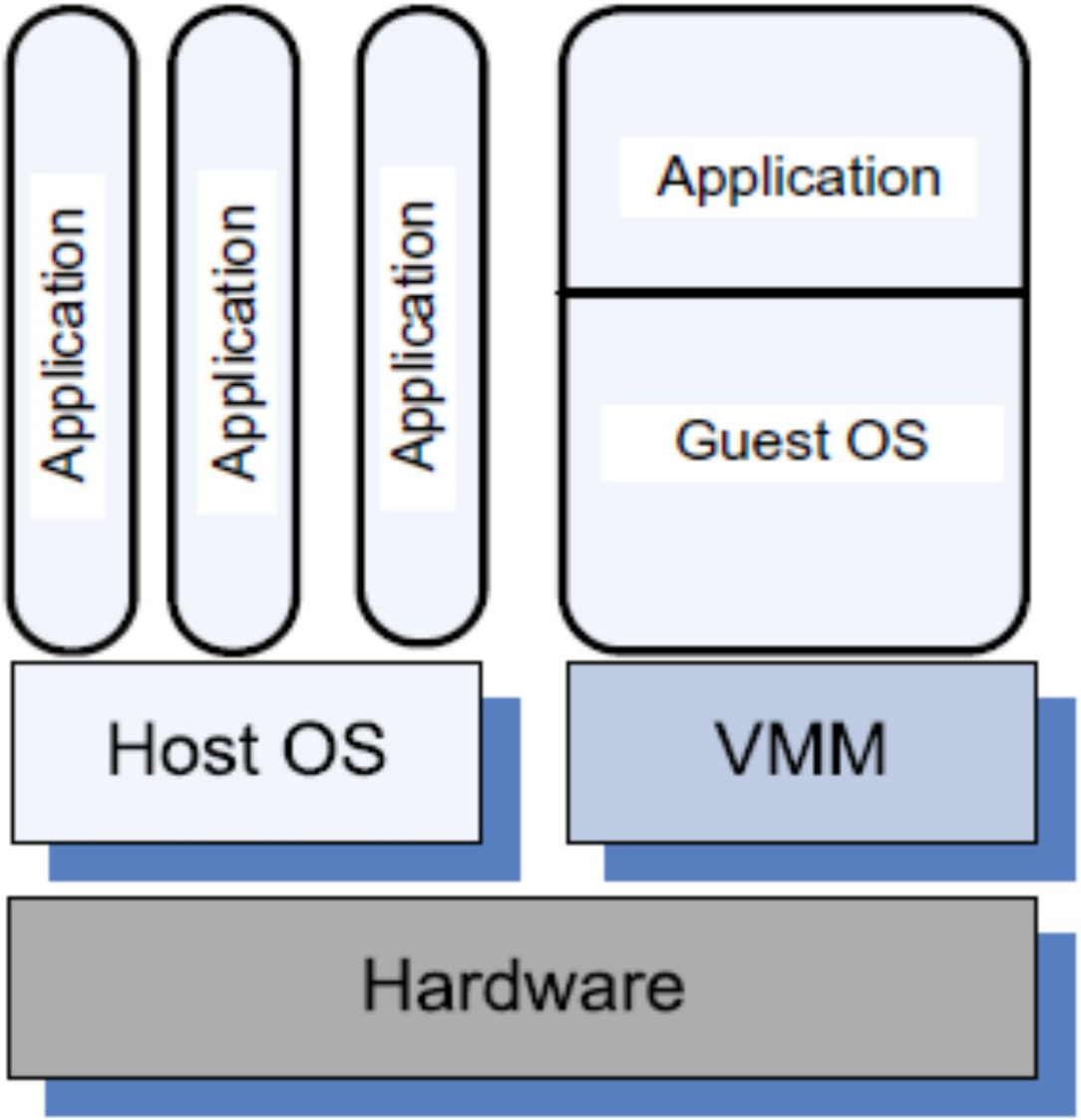
Three classes of VM for systems with the same ISA

- Traditional VM - The VMM supports multiple VMs and runs directly on the hardware. Examples: VMWare ESX, ESXi Servers, Xen, OS370, and Denali.
- Hybrid VM - The VMM shares the hardware with a host operating system and supports multiple virtual machines. Example: VMWare Workstation.
- Hosted VM – VMM runs under a host operating system. The main advantage of this approach is that the VM is easier to build and install. Example: User-mode *Linux*.

Traditional VM



Hybrid VM


Application
Application
Application
Application
Guest OS
Host OS
VMM
Hardware

Hosted VM

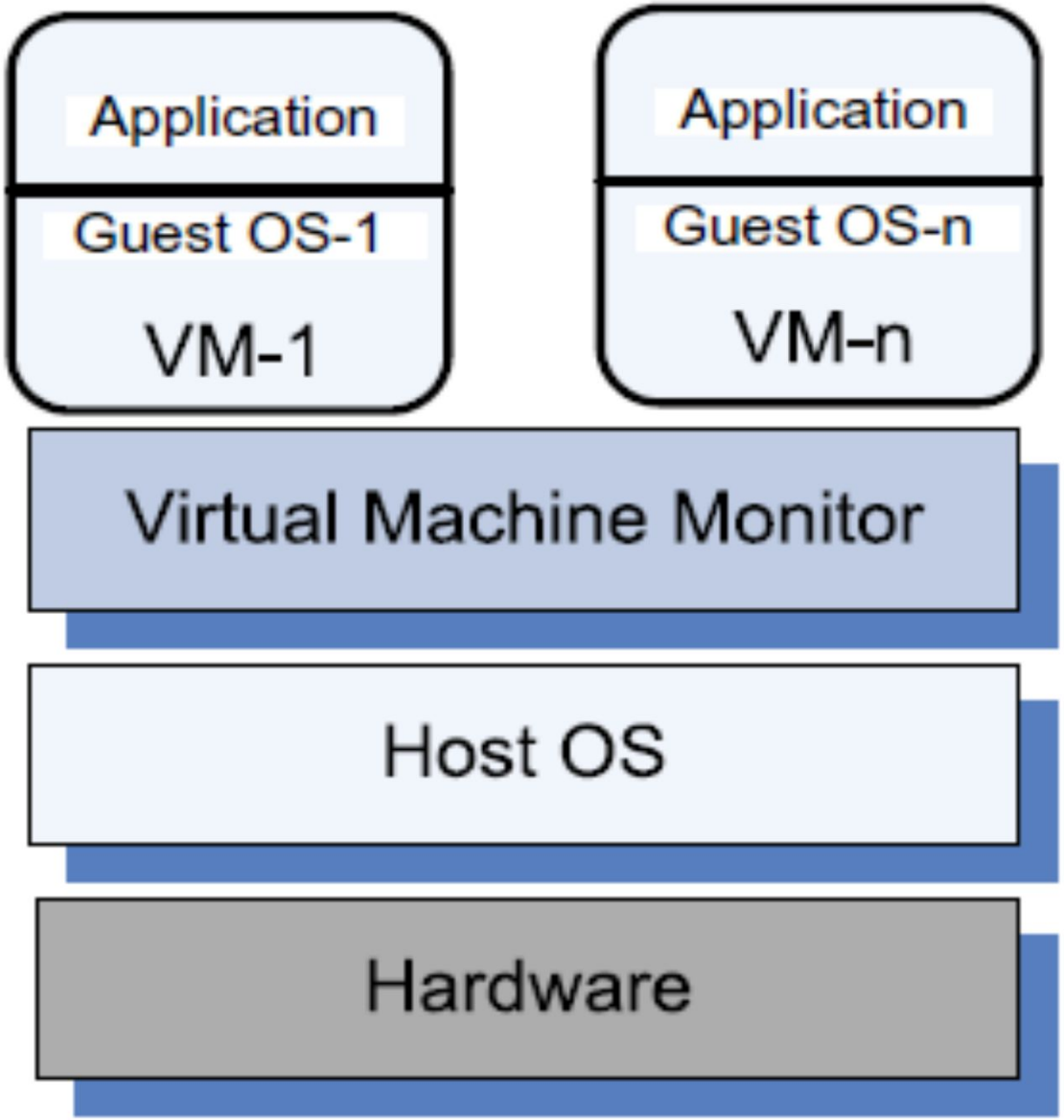

Application
Guest OS-1
VM-1
Application
Guest OS-n
VM-n
Virtual Machine Monitor
Host OS
Hardware

Table 5.1 A nonexhaustive inventory of system virtual machines. The host ISA refers to the instruction set of the hardware; the guest ISA refers to the instruction set supported by the virtual machine. The VM could run under a host OS, directly on the hardware, or under a VMM. The guest OS is the operating system running under the control of a VM, which in turn may run under the control of the VMM.

Name	Host ISA	Guest ISA	Host OS	Guest OS	Company
Integrity VM	*x86*-64	*x*86-64	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

Performance isolation

- The run-time behavior of an application is affected by other applications running concurrently on the same platform and competing for CPU cycles, cache, main memory, disk and network access. Thus, it is difficult to predict the completion time!
- Performance isolation - a critical condition for QoS guarantees in shared computing environments.

Performance isolation

- **Temporal isolation** or **performance isolation** among virtual machine (VMs) refers to the capability of isolating the temporal behavior of multiple VMs among each other, despite them running on the same physical host and sharing a set of physical resources such as processors, memory, and disks.

security isolation

- The software running on a virtual machine can only access virtual devices emulated by the software. Thus, the virtualization can be used to improve security in a cloud computing environment.
- A VMM is a much simpler and better specified system than a traditional operating system. For example, the Xen VMM has approximately 60,000 lines of code, whereas the Denali VMM has only about 30,000 lines of code.
- The security vulnerability of VMMs is considerably reduced because the systems expose a much smaller number of privileged functions.

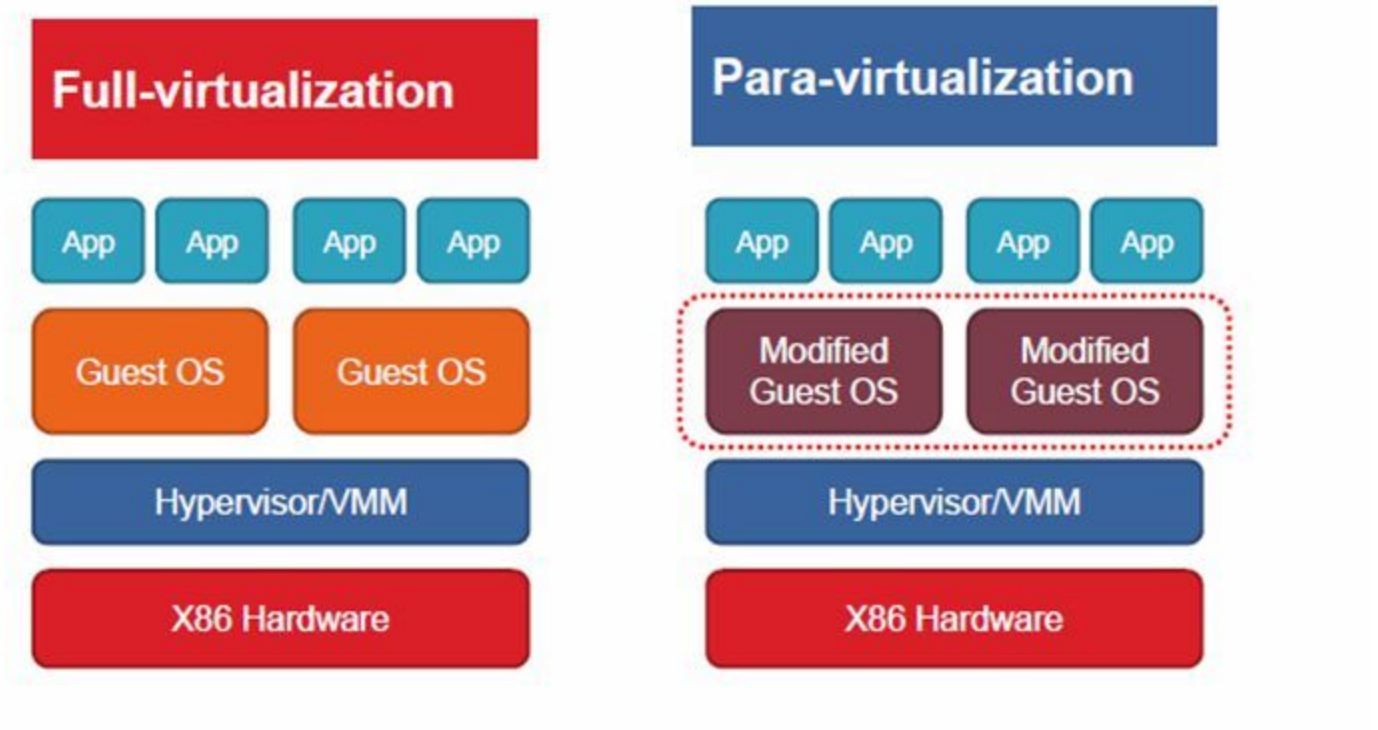
Conditions for a computer architecture to support virtualization

- A program running under the VMM should exhibit a behavior identical to that demonstrated when running on an equivalent machine directly.
- The VMM should be in complete control of the virtualized resources.
- Machine instructions must be executed without the intervention of the VMM.

Full virtualization and Paravirtualization

- Full virtualization and paravirtualization both enable hardware resource abstraction, the two technologies differ when it comes to isolation levels.
- A hypervisor is a software tool installed on the host system to provide this layer of abstraction.
- Once a hypervisor is installed, OSes and applications interact with the virtualized resources abstracted by the hypervisor -- not the physical resources of the actual host computer. Virtualization is classified based on the level of isolation provided by the hypervisor.

- Full vs. para virtualization

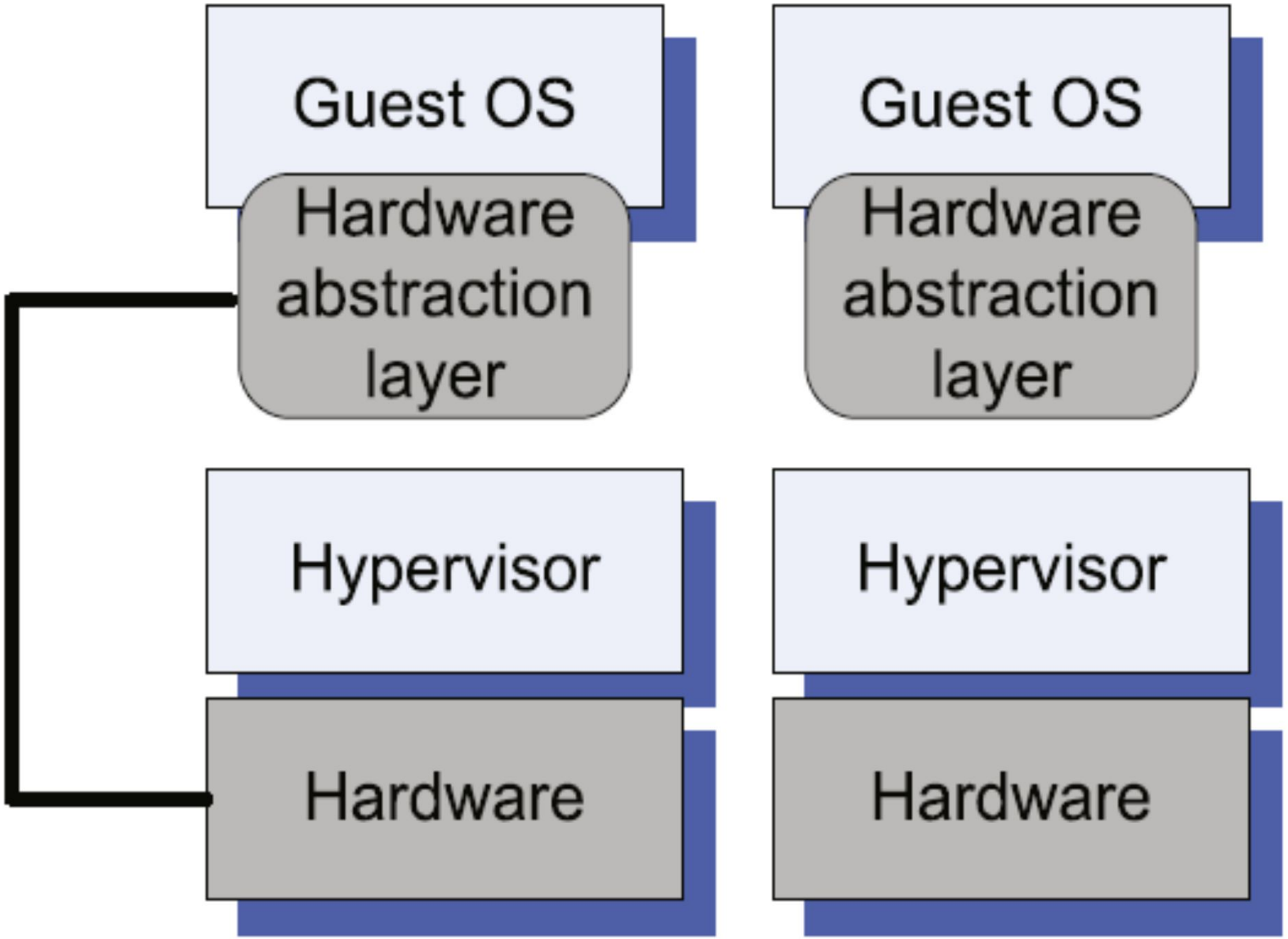


Full virtualization

- Hypervisor provides complete abstraction, and the guest OSes don't know -- or care -- about the presence of a hypervisor.
- Each VM and its guest OS works as if it was running alone on independent computers, and no special modifications or adaptations are needed in the OS.
- Machine runs on an exact copy of the actual hardware.
- Full virtualization requires a virtualizable architecture; the hardware is fully exposed to the guest OS, which runs unchanged.
- Examples: Vmware. Oracle's Virtaulbox , Microsoft Virtual PC.

Paravirtualization

- In this type, a guest operating system is modified to use only those instructions that can be virtualized to run under the VMM.
- Virtual machine runs on a slightly modified copy of the actual hardware.
- The guest OS code must be ported for individual hardware platforms.
- Reasons for paravirtualization:
  - Some aspects of the hardware cannot be virtualized.
  - Improved performance.
  - Present a simpler interface.
  - Examples: Xen, Denaly


Guest OS
Hardware abstraction layer
Hypervisor
Hardware
Guest OS
Hardware abstraction layer
Hypervisor
Hardware

FV **(a)** PV **(b)**

Hardware support for virtualization

- Hardware support enables efficient virtualization. Both Intel and AMD have incorporated explicit support for virtualization into their CPU designs.
- This can simplify the design of virtual machine monitor (VMM), techniques such as paravirtualization and hosted VMM's.
- The x86 (x86-64) architecture supports 4 privilege levels, or rings, with ring 0 being the most privileged and ring 3 the least. Operating systems run in ring 0, user applications run in ring 3, and rings 1 and 2 are not typically used.

Hardware support for virtualization

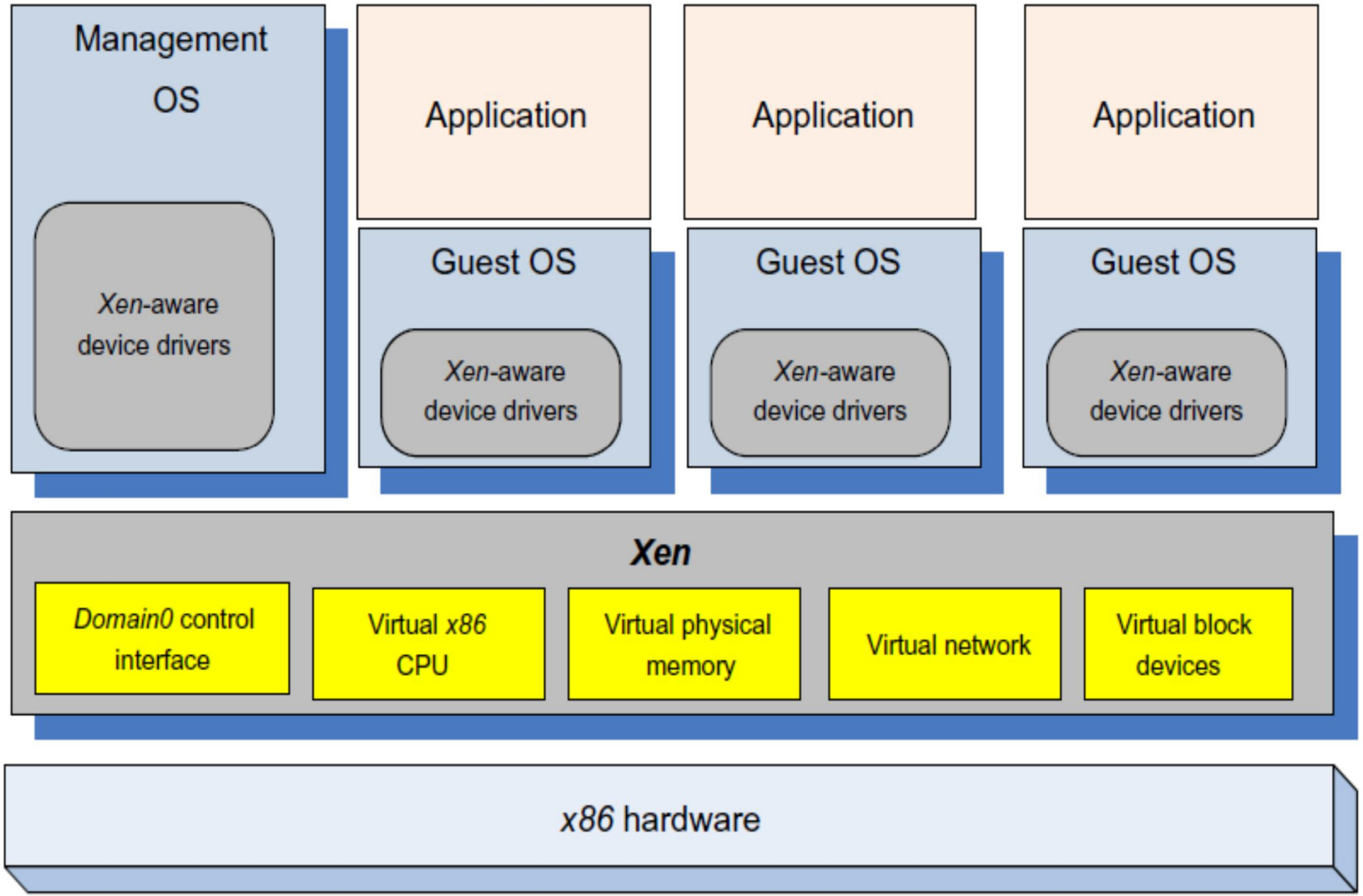
- Hardware should support
  - CPU virtualization
  - I/O virtualization
  - MMU (Memory Management Unit) virtualization

Xen

- A VMM based on paravirtualization.
- Xen is a VMM or hypervisor developed by the Computing Laboratory at the University of Cambridge, United Kingdom, in 2003. Since 2010 Xen has been free software, developed by the community of users and licensed under the GNU General Public License (GPLv2).
- Several operating systems, including Linux, Minix, NetBSD, FreeBSD, NetWare, and OZONE, can operate as paravirtualized Xen guest operating systems running on x86, x86-64, Itanium, and ARM architectures.

Xen

- The goal of the Cambridge group - design a VMM capable of scaling to about 100 VMs running standard applications and services without any modifications to the Application Binary Interface (ABI).
- Fully aware that the x86 architecture does not support efficiently full virtualization, the designers of Xen opted for paravirtualization.



Xen Architecture

Xen

- The creators of *Xen* used the concept of *domain* (*Dom*) to refer to the ensemble of address spaces hosting a guest OS and address spaces for applications running under this guest OS.
- Each domain runs on a virtual x86 CPU. Dom0 is dedicated to the execution of Xen control functions and privileged instructions, and DomU is a user domain.
- Xen provides paravirtualization for virtual memory management, CPU multiplexing, and I/O device management.

Xen

- In *Xen* the VMM runs at Level 0, the guest OS at Level 1, and applications at Level 3.
- XenStore is a Dom0 process that supports a system-wide registry and naming service. XenStore communicates with guest VMs via shared memory using Dom0 privileges.
- The Toolstack is another Dom0 component responsible for creating, destroying, and managing the resources and privileges of VMs.
- *Xen* defines abstractions for networking and I/O devices

Xen

- *Xen* enforces access control for the shared memory and passes synchronization signals.
- Access control lists (ACLs) are stored in the form of *grant tables*, with permissions set by the owner of the memory.
- Each domain has one or more virtual network interfaces (VIFs) that support the functionality of a network interface card.

Xen

- Privileged domains, called `driver' domains, use their native device drivers to access I/O devices directly, and perform I/O operations on behalf of other unprivileged domains, called guest domains.
- Guest domains use virtual I/O devices controlled by paravirtualized drivers to request the driver domain for device access.
- It is reported that a total of about 3,000 lines of Linux code, or 1.36%, had to be modified; for Windows XP this figure is 4,620, or about 0.04%

Optimization of network virtualization in *Xen* 2.0

- A virtual machine monitor introduces a significant network communication overhead.
- For example, it is reported that the CPU utilization of a VMware Workstation 2.0 system running Linux 2.2.17 was 5 to 6 times higher than that of the native system (Linux 2.2.17) in a 100 Mbps network.
- Similar overheads are reported for other VMMs and, in particular, for *Xen* 2.0.

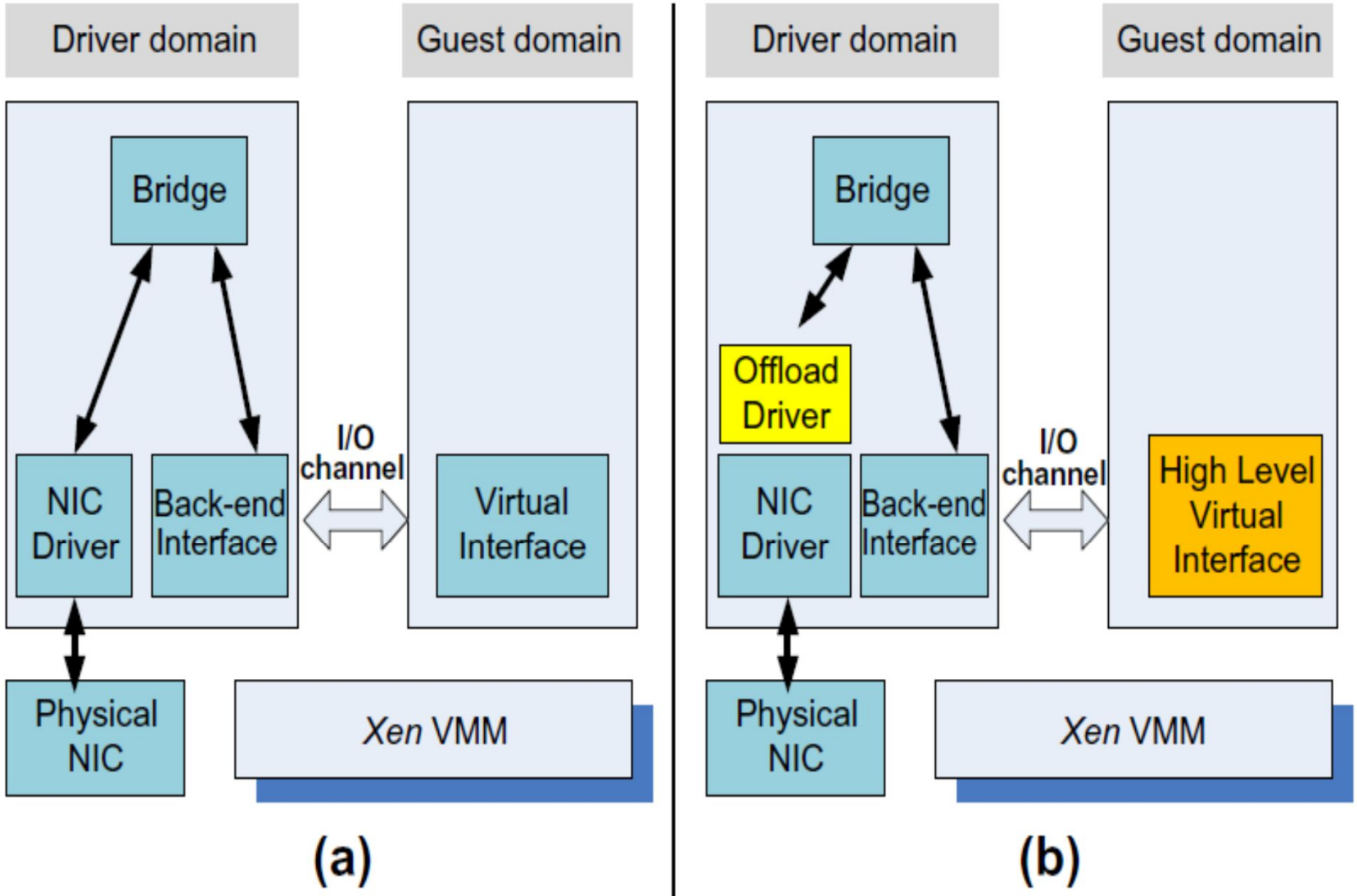


FIGURE 5.8

Xen network architecture. (a) The original architecture. (b) The optimized architecture.

Optimization of network virtualization in *Xen* 2.0

- Xen provides each guest domain with a number of virtual network interfaces, which is used by the guest domain for all its network communications.
- Corresponding to each virtual interface in a guest domain, a `backend' interface is created in the driver domain, which acts as the proxy for that virtual interface in the driver domain.
- The guest domain communicates with the driver domain through an I/O channel; the guest OS in the guest domain uses a virtual interface to send/receive data to/from the back-end interface in the driver domain.

Optimization of network virtualization in *Xen* 2.0

- All the backend interfaces in the driver domain (corresponding to the virtual interfaces) are connected to the physical NIC through a virtual network bridge.
- Since each packet transmitted or received on a guest domain's virtual interface had to pass through the I/O channel and the network bridge, a significant fraction of the network processing time was spent in the Xen VMM and the driver domain respectively.

Optimization of network virtualization in *Xen* 2.0

- For instance, 70% of the execution time for receiving a packet in the guest domain was spent in transferring the packet through the driver domain and the Xen VMM from the physical interface.
- Similarly, 60% of the processing time for a transmit operation was spent in transferring the packet from the guest's virtual interface to the physical interface.
- The breakdown of this processing overhead was roughly 40% Xen, 30% driver domain for receive traffic, and 30% Xen, 30% driver domain for transmit traffic.

Optimization of network virtualization in *Xen* 2.0

- The Xen network optimization includes the optimization of (i) the virtual interface; (ii) the I/O channel; and (iii) the virtual memory.
- The effects of these optimizations are significant for the send data rate from the optimized Xen guest domain, an increase from 750 to 3, 310 Mbps, and rather modest for the receive data rate, 970 versus 820 Mbps.

Optimization of network virtualization in *Xen* 2.0

- The architecture introduces a new component, a `software offload' driver, which sits above the network device driver in the driver domain.
- It intercepts all packets to be transmitted on the network interface. When the guest domain's packet arrives at the physical interface, the offload driver determines whether the offload requirements of the packet are compatible with the capabilities of the NIC, and takes different actions accordingly.

vBlades: paravirtualization targeting an *x86*-64 Itanium processor

- It is VMM project at HP-Laboratories.
- The goal of the *vBlades* project was to create a VMM for the *Itanium* family of *IA64* Intel processors, capable of supporting the execution of multiple operating systems in isolated protection domains with security and privacy enforced by the hardware.
- The vBlades support CPU virtualization and memory virtualization.

vBlades: paravirtualization targeting an *x86*-64 Itanium processor

- the *Itanium* processor supports four *privilege rings*, PL0, PL1, PL2, and PL3. Privileged instructions can only be executed by the kernel running at level PL0, whereas applications run at level PL3 and can only execute non-privileged instructions; PL2 and PL4 rings are generally not used.
- *Itanium* is a processor developed jointly by HP and Intel and based on a new architecture, explicitly parallel instruction computing (EPIC), that allows the processor to execute multiple instructions in each clock cycle.

Software fault isolation

- *Software fault isolation* (SFI) offers a technical solution for sandboxing (security mechanism) binary code that can affect security in cloud computing.
- *Insecure and tampered VM images are one of the security threats because binary codes of plug-ins to a Web browser can pose a security threat when Web browsers are used to access cloud services.*


P.A.I.N.S
Positive Attitude
In Negative
Situations.