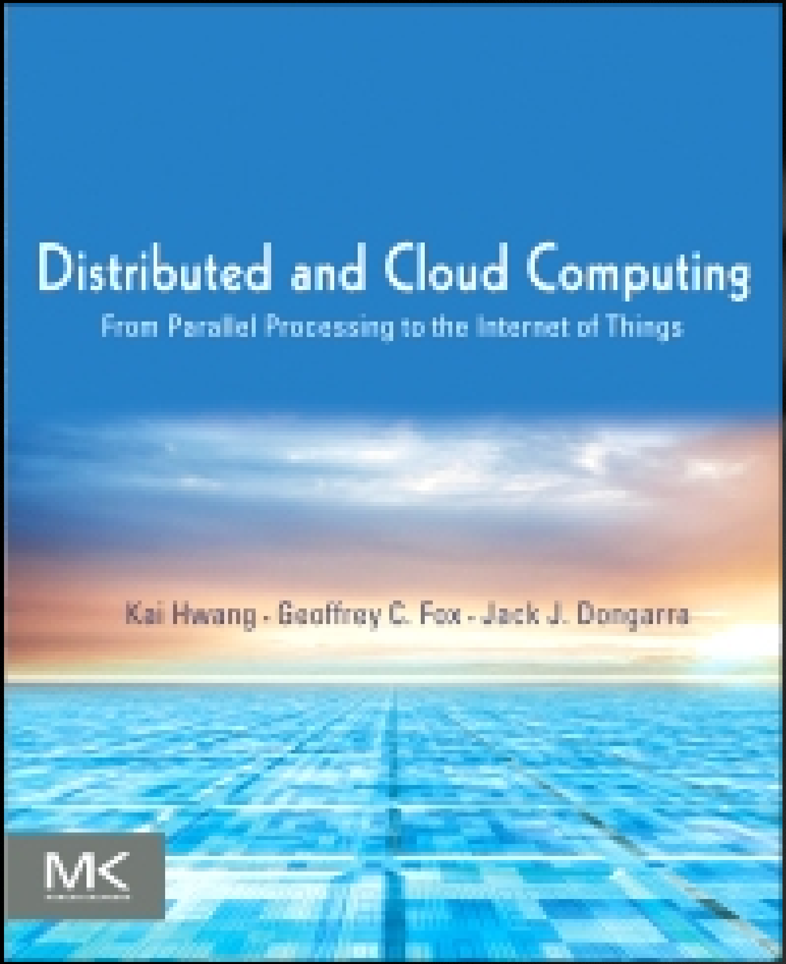
PSI5021 Tópicos de Computação em Nuvem

Aula 02 - Virtualização

Livro texto


Distributed and Cloud Computing
From Parallel Processing to the Internet of Things
Kai Hwang · Geoffrey C. Fox · Jack J. Dongarra
MK

Virtualization for Datacenter Automation

to serve millions of clients, simultaneously

- Server Consolidation in Virtualized Datacenter
- Virtual Storage Provisioning and Deprovisioning
- Cloud Operating Systems for Virtual Datacenters
- Trust Management in virtualized Datacenters

Difference between Traditional Computer and Virtual machines

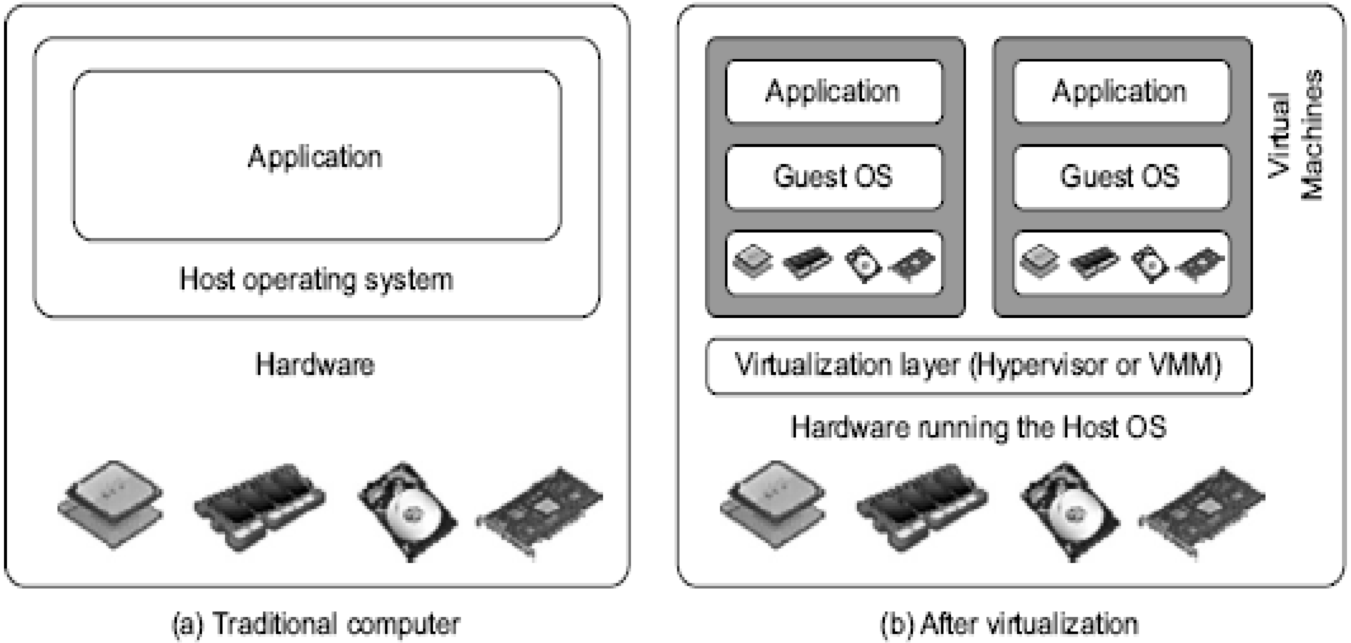
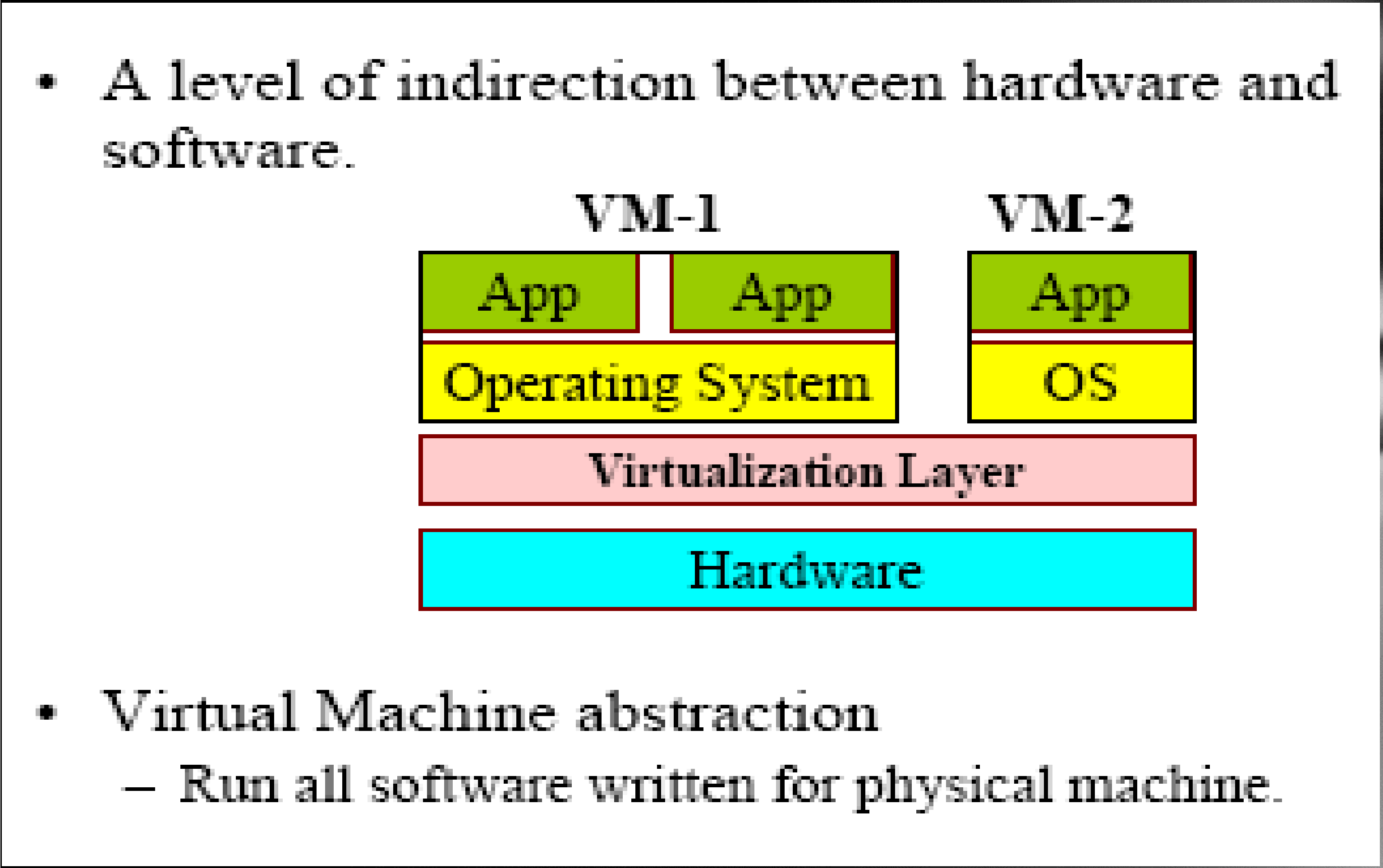


FIGURE 3.1

The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.

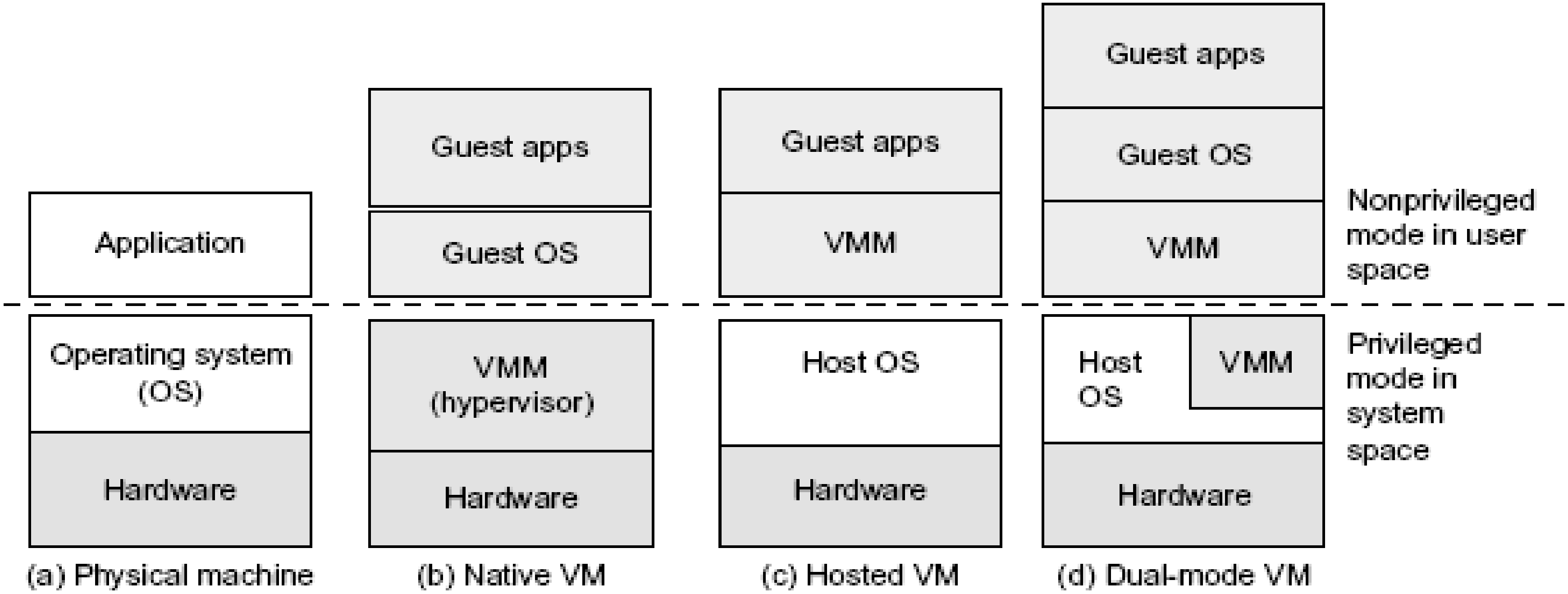
(Courtesy of VMWare, 2008)

What is Virtualization ?



- A level of indirection between hardware and software.

- Virtual Machine abstraction
  - Run all software written for physical machine.

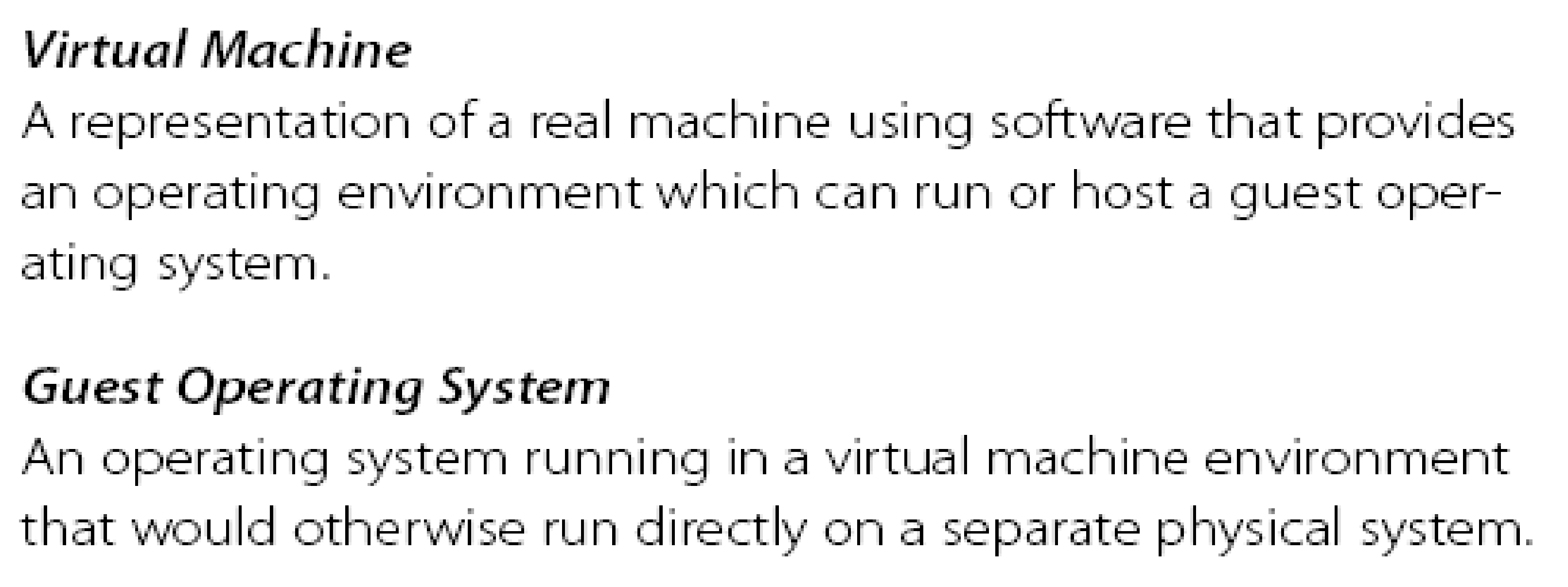


(a) Physical machine (b) Native VM (c) Hosted VM (d) Dual-mode VM

FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

Virtual Machine, Guest Operating System, and VMM (Virtual Machine Monitor) :



Virtual Machine

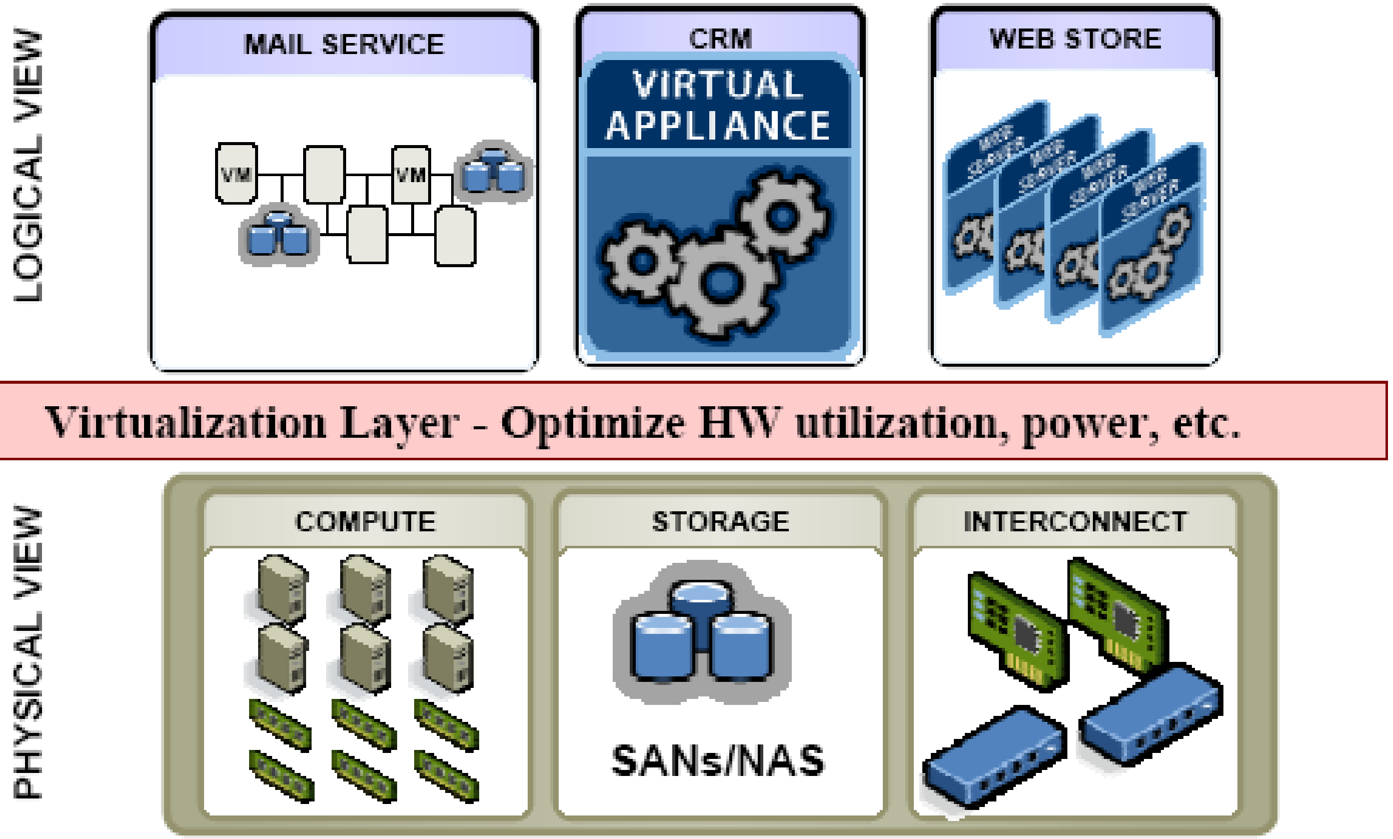
A representation of a real machine using software that provides an operating environment which can run or host a guest operating system.

Guest Operating System

An operating system running in a virtual machine environment that would otherwise run directly on a separate physical system.

The Virtualization layer is the middleware between the underlying hardware and virtual machines represented in the system, also known as *virtual machine monitor* (VMM).

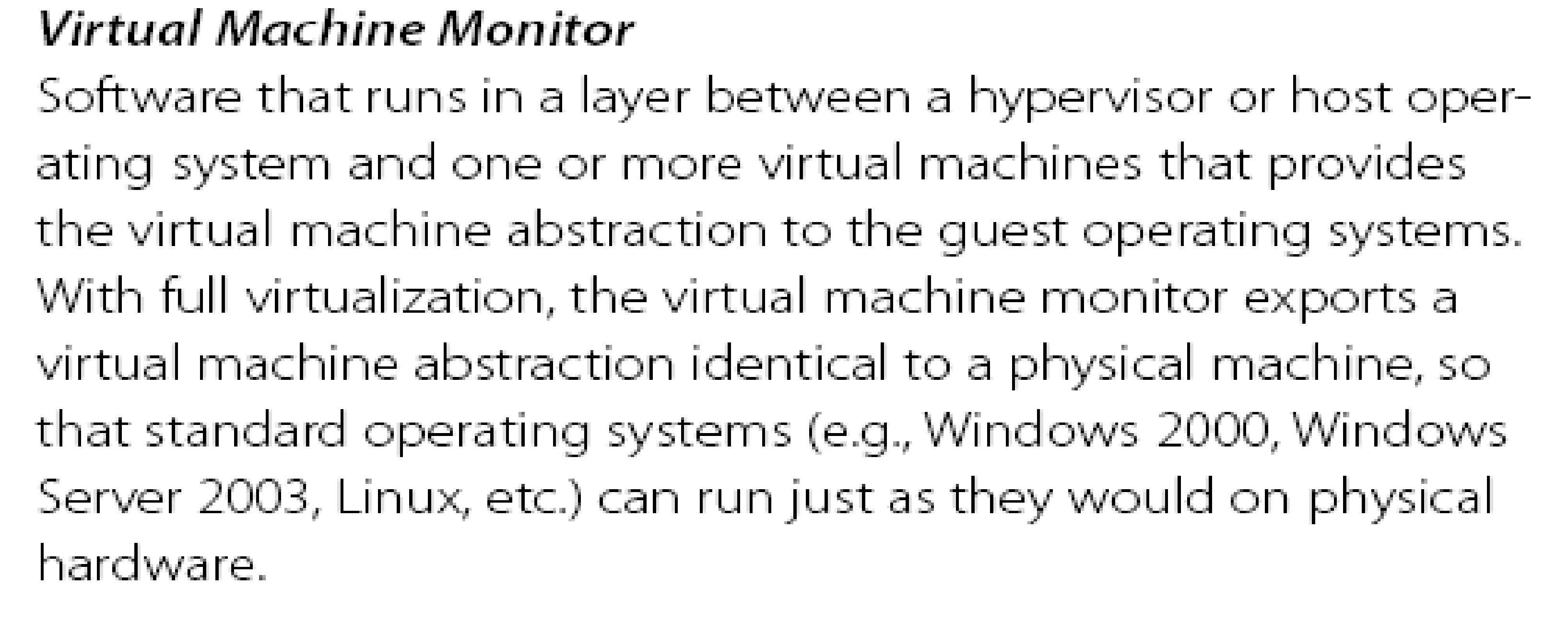
User's view of virtualization



Virtualization Layer - Optimize HW utilization, power, etc.

(Courtesy of VMWare, 2008)

VMM : Virtual Machine Monitor



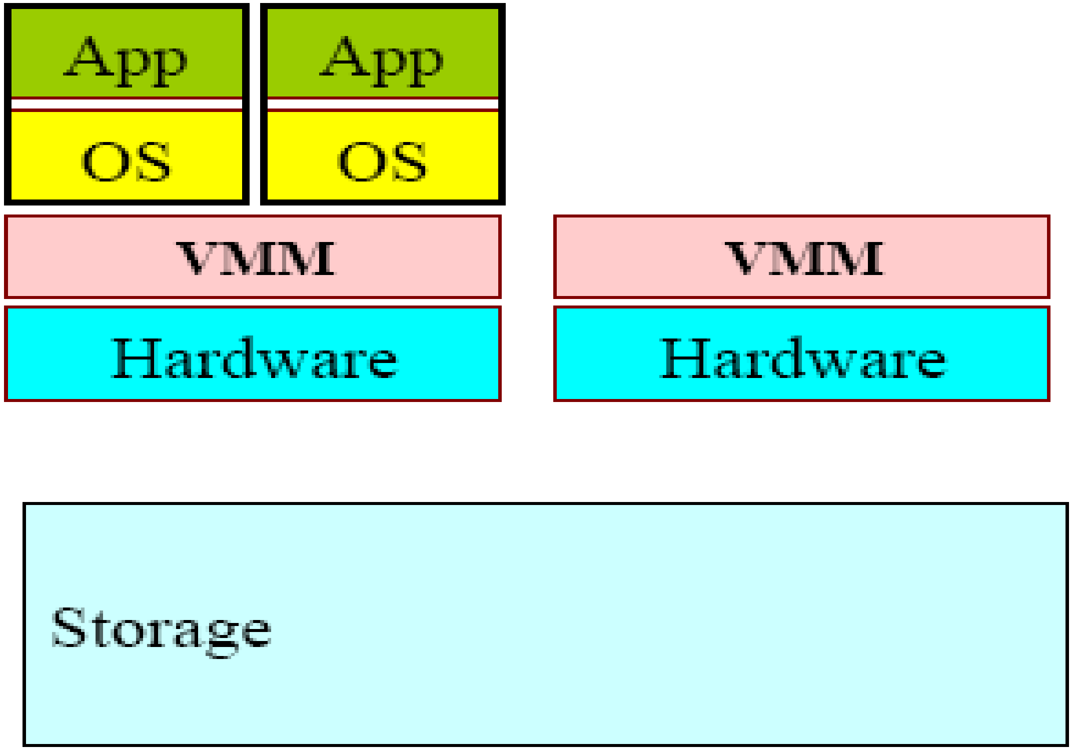
Virtual Machine Monitor

Software that runs in a layer between a hypervisor or host operating system and one or more virtual machines that provides the virtual machine abstraction to the guest operating systems. With full virtualization, the virtual machine monitor exports a virtual machine abstraction identical to a physical machine, so that standard operating systems (e.g., Windows 2000, Windows Server 2003, Linux, etc.) can run just as they would on physical hardware.

(Courtesy of Mendel Rosenblum, 2005)

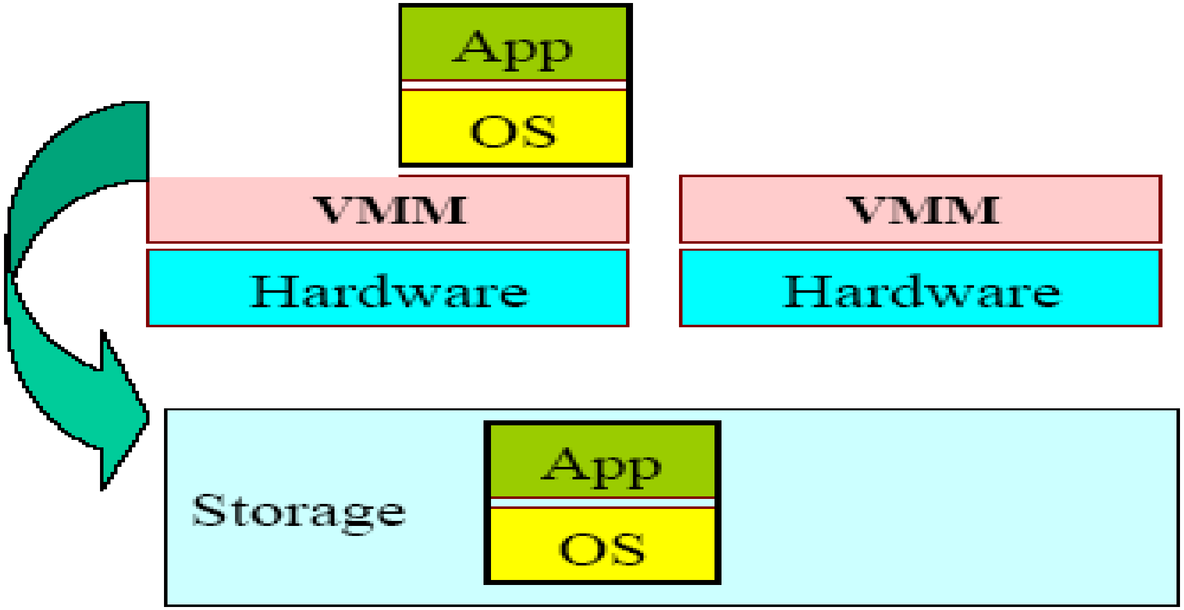
Low-Level VMM Operations (1)

- Multiplex



(Courtesy of Mendel Rosenblum, 2006)

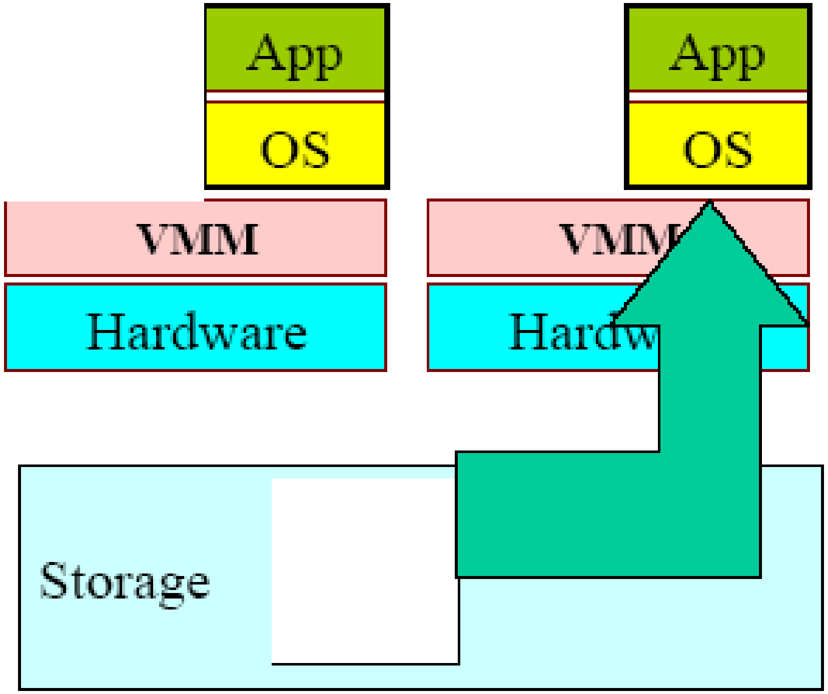
Low-Level VMM Operations (2)



- Multiplex
- Suspend

(Courtesy of Mendel Rosenblum, 2006)

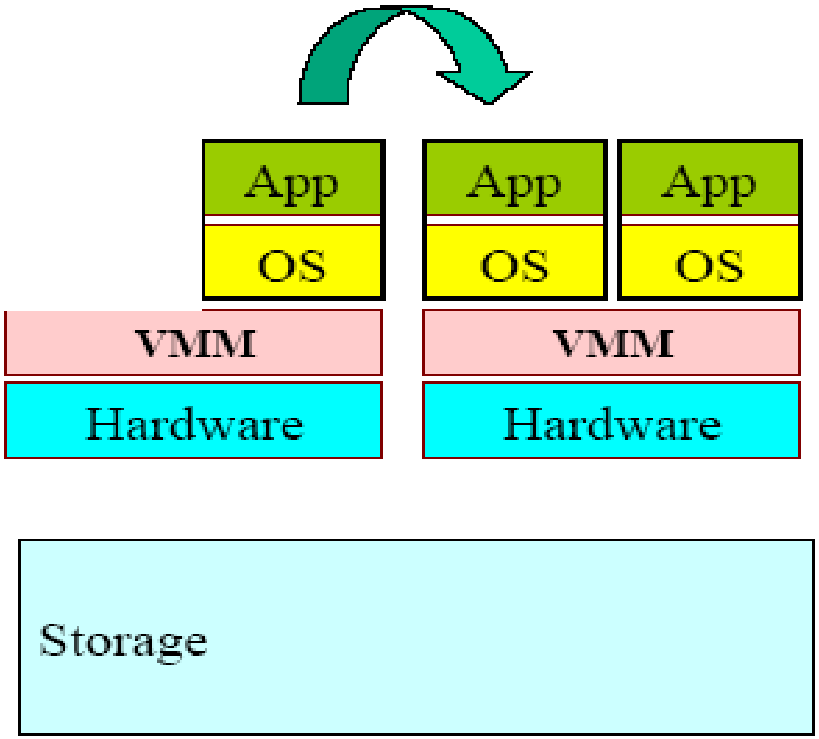
Low-Level VMM Operations (3)



- Multiplex
- Suspend
- Resume (Provision)

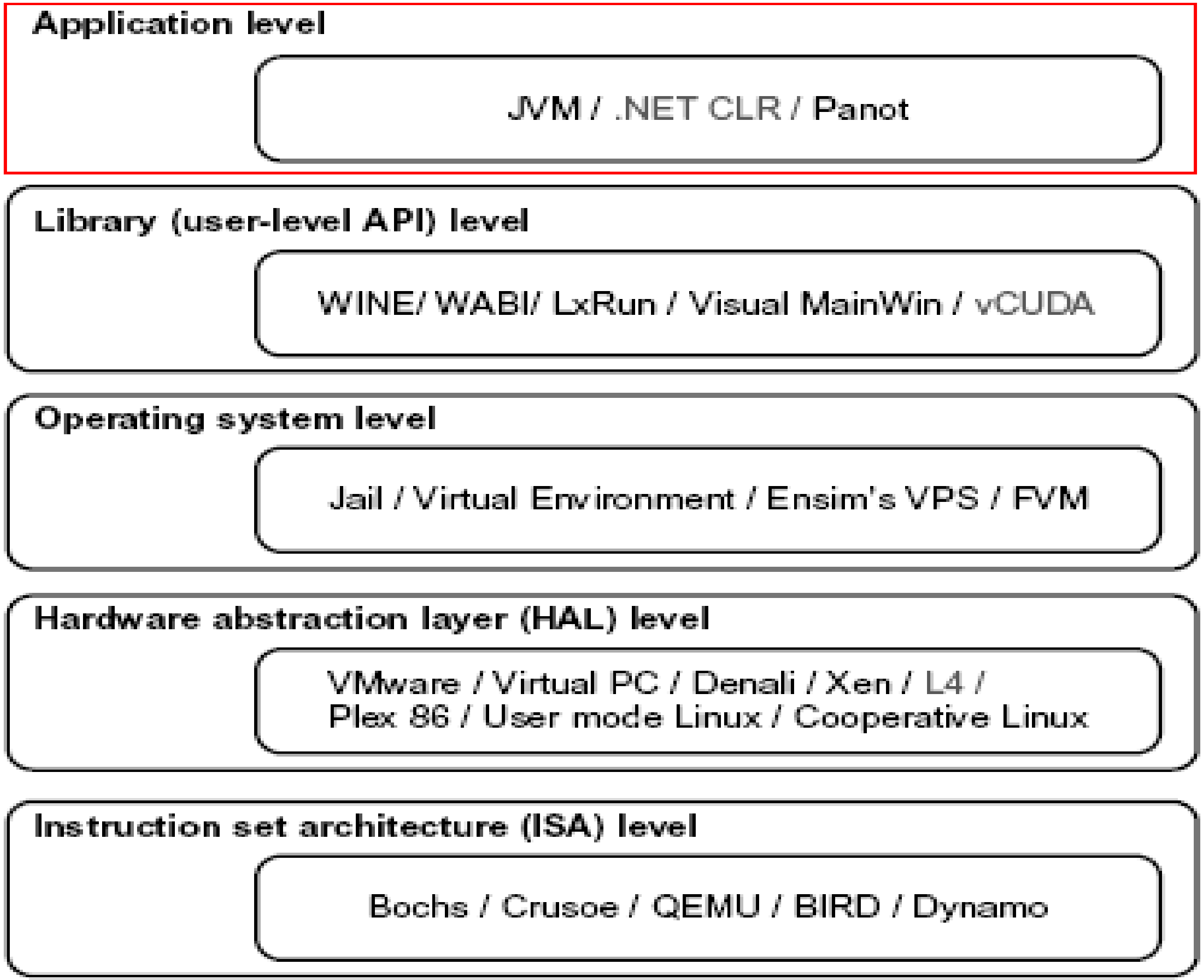
(Courtesy of Mendel Rosenblum, 2006)

Low-Level VMM Operations (4)



- Multiplex
- Suspend
- Resume (Provision)
- Migration

(Courtesy of Mendel Rosenblum, 2006)



Application level

JVM / .NET CLR / Panot

Library (user-level API) level

WINE/ WABI/ LxRun / Visual MainWin / vCUDA

Operating system level

Jail / Virtual Environment / Ensim's VPS / FVM

Hardware abstraction layer (HAL) level

VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux

Instruction set architecture (ISA) level

Bochs / Crusoe / QEMU / BIRD / Dynamo

FIGURE 3.2

Virtualization ranging from hardware to applications in five abstraction levels.

Virtualization at ISA level:

Emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x-86-based host machine with the help of ISA emulation. Typical systems: Bochs, Crusoe, Quemu, BIRD, Dynamo

Advantage: It can run a large amount of legacy binary codes written for various processors on any given new hardware host machines; best application flexibility

Shortcoming & limitation: One source instruction may require tens or hundreds of native target instructions to perform its function, which is relatively slow. V-ISA requires adding a processor-specific software translation layer in the complier.

Virtualization at Hardware Abstraction level:

Virtualization is performed right on top of the hardware. It generates virtual hardware environments for VMs, and manages the underlying hardware through virtualization. Typical systems: VMware, Virtual PC, Denali, Xen

Advantage: has higher performance and good application isolation

Shortcoming & limitation: very expensive to implement (complexity)

Virtualization at Operating System level:

It is an abstraction layer between traditional OS and user placations. This virtualization creates isolated containers on a single physical server and the OS-instance to utilize the hardware and software in datacenters. Typical systems: Jail / Virtual Environment / Ensim's VPS / FVM

Advantage: have minimal starup/shutdown cost, low resource requirement, and high scalability; synchronize VM and host state changes.

Shortcoming & limitation: all VMs at the operating system level must have the same kind of guest OS; poor application flexibility and isolation.

Virtualization at OS Level

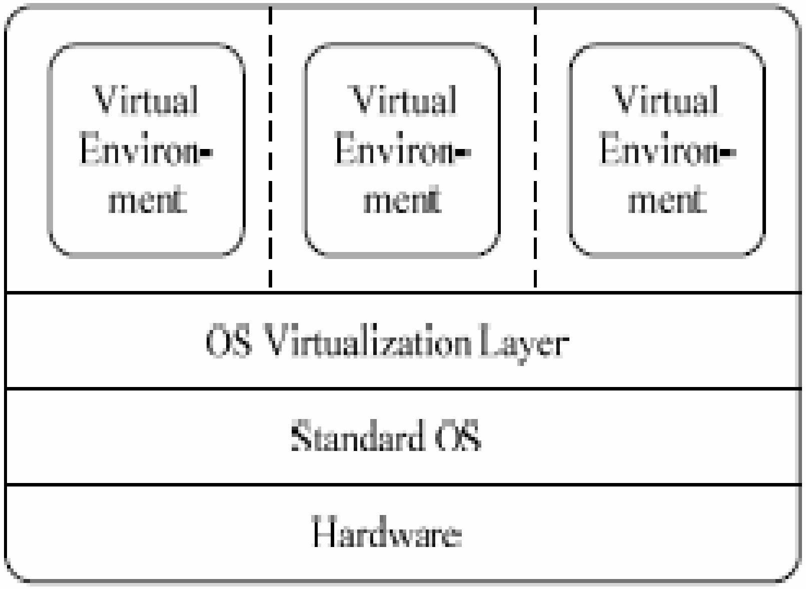


Figure 6.3 The virtualization layer is inserted inside an OS to partition the hardware resources for multiple VMs to run their applications in virtual environments

Virtualization for Linux and Windows NT Platforms

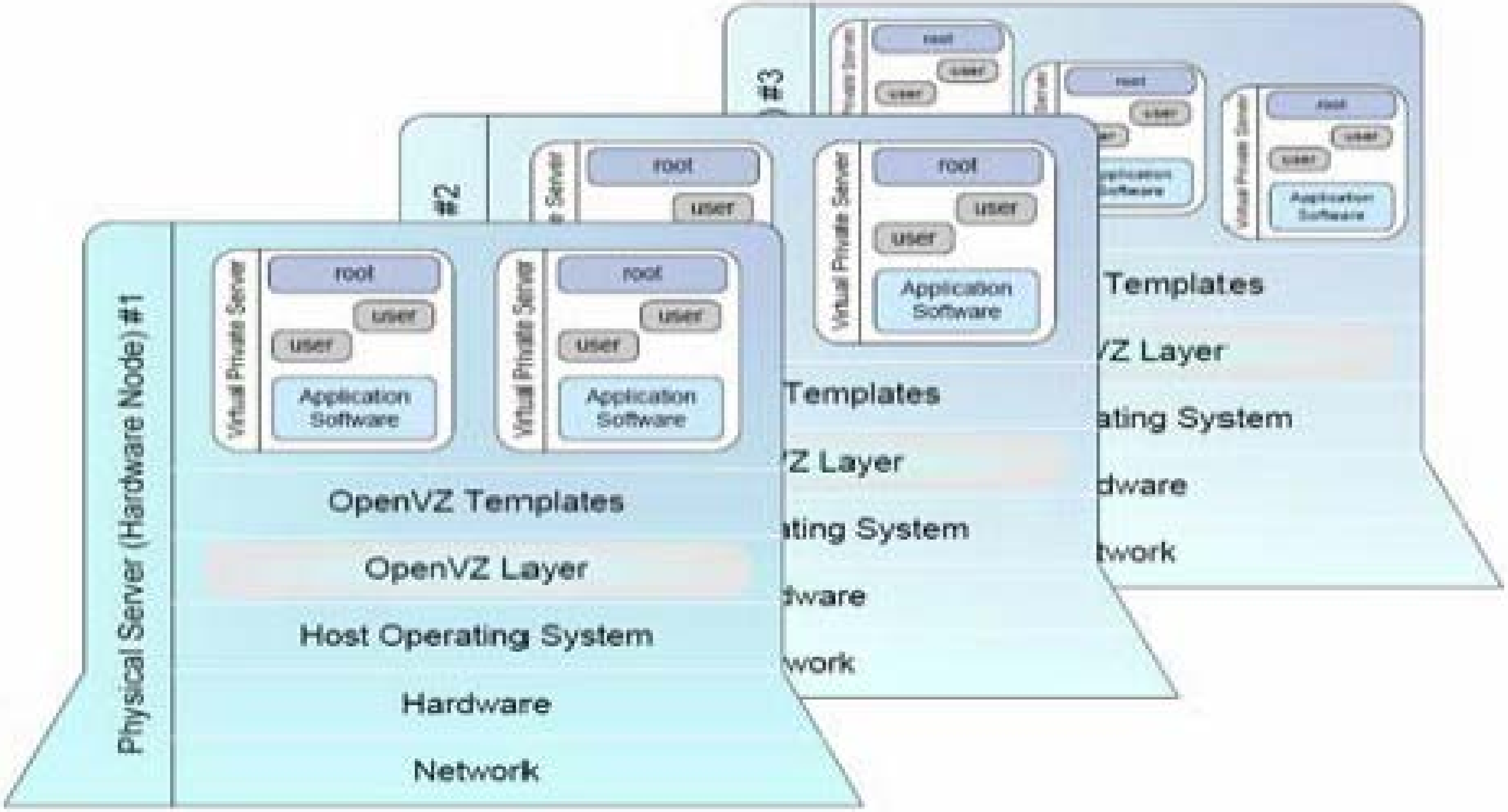


Figure 6.4 OpenVZ inserts a virtualization layer called OpenVZ inside the host OS. This layer provides some OS images to create VMs quickly (Courtesy of OpenVZ User's Guide, http://ftp.openvz.org/doc/OpenVZ-Users-Guide.pdf)

Advantages of OS Extension for Virtualization

1. VMs at OS level has minimum startup/ shutdown costs

2. OS-level VM can easily synchronize with its environment

Disadvantage of OS Extension for Virtualization

All VMs in the same OS container must have the same or similar guest OS, which restrict application flexibility of different VMs on the same physical machine.

Library Support level:

It creates execution environments for running alien programs on a platform rather than creating VM to run the entire operating system. It is done by API call interception and remapping. Typical systems: Wine, WAB, LxRun , VisualMainWin

Advantage: It has very low implementation effort

Shortcoming & limitation: poor application flexibility and isolation

Virtualization with Middleware/Library Support

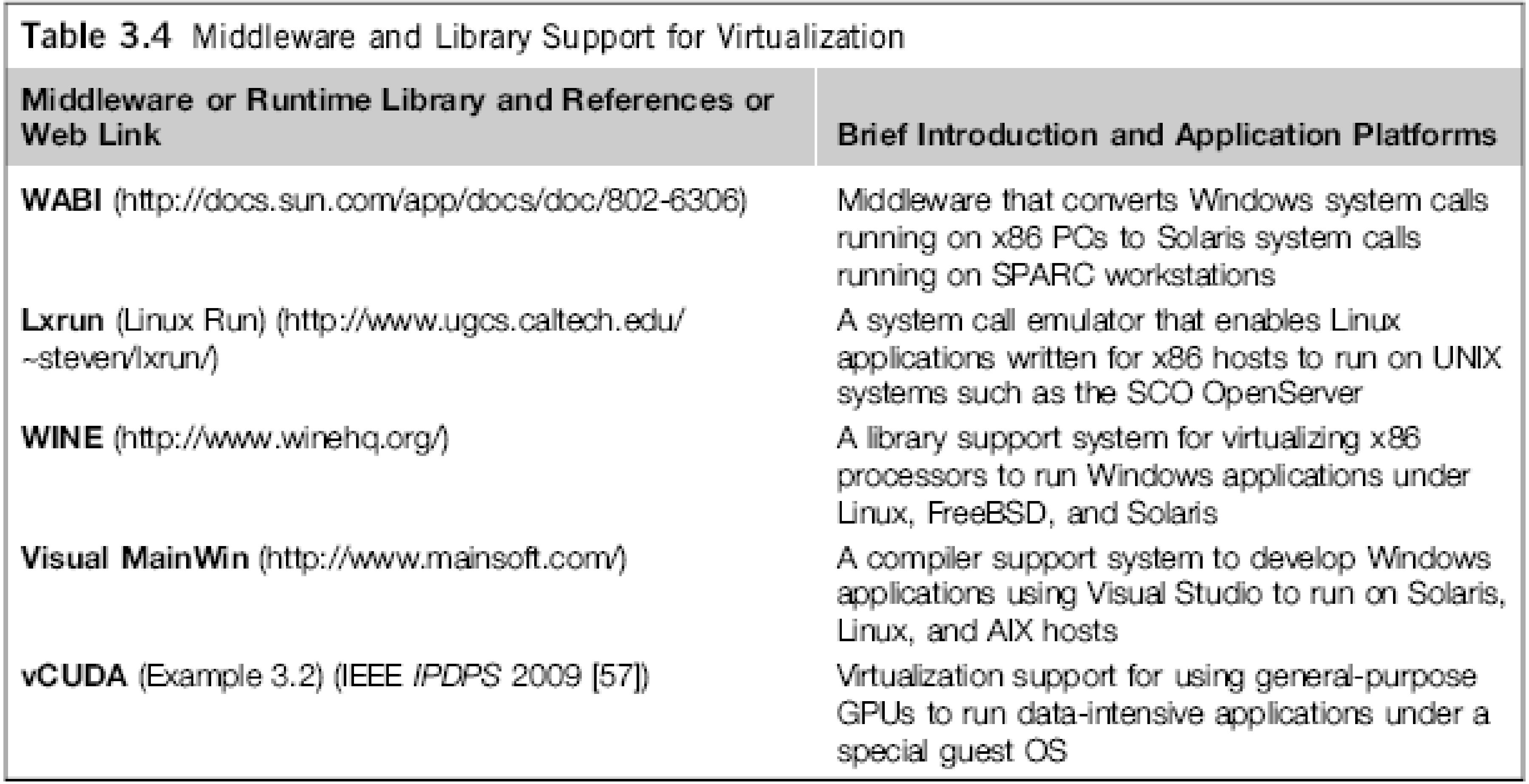


Table 3.4 Middleware and Library Support for Virtualization

Middleware or Runtime Library and References or Web Link	Brief Introduction and Application Platforms
WABI (http://docs.sun.com/app/docs/doc/802-6306)	Middleware that converts Windows system calls running on x86 PCs to Solaris system calls running on SPARC workstations
Lxrun (Linux Run) (http://www.ugcs.caltech.edu/~steven/lxrun/)	A system call emulator that enables Linux applications written for x86 hosts to run on UNIX systems such as the SCO OpenServer
WINE (http://www.winehq.org/)	A library support system for virtualizing x86 processors to run Windows applications under Linux, FreeBSD, and Solaris
Visual MainWin (http://www.mainsoft.com/)	A compiler support system to develop Windows applications using Visual Studio to run on Solaris, Linux, and AIX hosts
vCUDA (Example 3.2) (IEEE *IPDPS* 2009 [57])	Virtualization support for using general-purpose GPUs to run data-intensive applications under a special guest OS

User-Application level:

It virtualizes an application as a virtual machine. This layer sits as an application program on top of an operating system and exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Typical systems: JVM , NET CLI , Panot

Advantage: has the best application isolation

Shortcoming & limitation: low performance, low application flexibility and high implementation complexity.

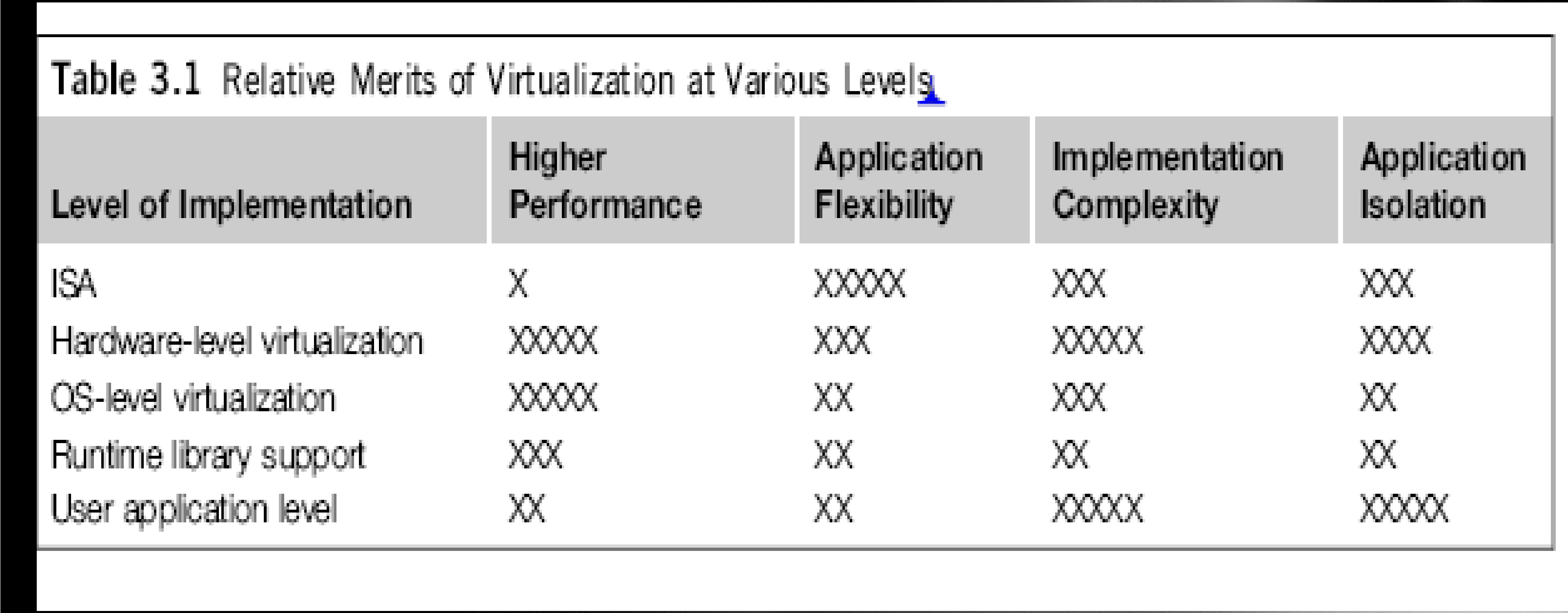


Table 3.1 Relative Merits of Virtualization at Various Levels[1]

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

Full Virtualization vs. Para-Virtualization

Full virtualization does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation. On the other hand, para virtualization needs to modify guest OS, and non-virtualizable instructions are replaced by hypercalls that communicate directly with the hypervisor or VMM. As of full virtualization, the advantage is no need to modify OS. However, this approach of binary translation slows down the performance a lot.

Para virtualization reduces the overhead, but the cost of maintaining paravirtualized OS is high. The improvement depends on the workload. VMware Workstation applies full virtualization, which uses binary translation to automatically modify x86 software on-the-fly to replace critical instructions. The para virtualization is supported by Xen, Denali and VMware ESX.

Full Virtualization

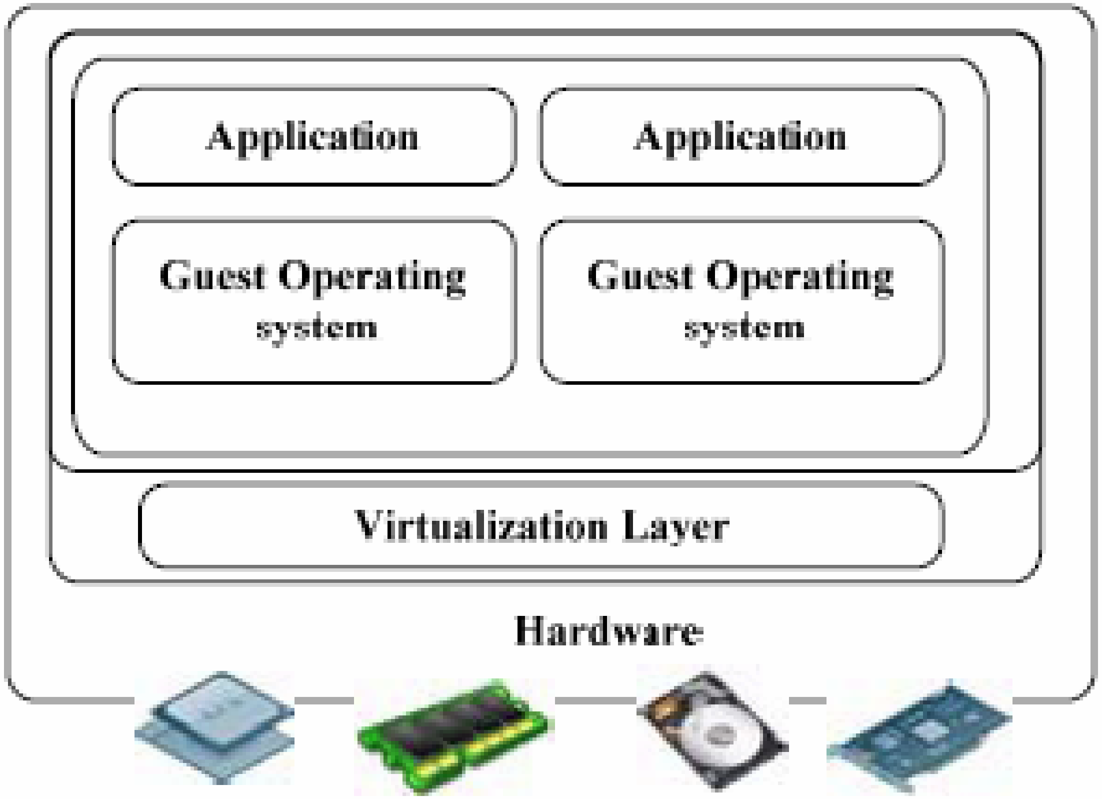
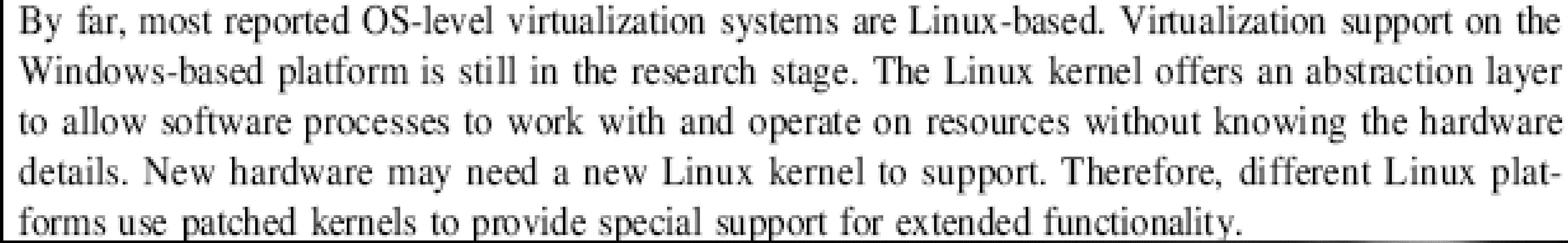


Figure 6.9 The concept of full virtualization using a hypervisor or a VMM directly sitting on top of the bare hardware devices. Note that no host OS is used here as in Figure 6.11.



By far, most reported OS-level virtualization systems are Linux-based. Virtualization support on the Windows-based platform is still in the research stage. The Linux kernel offers an abstraction layer to allow software processes to work with and operate on resources without knowing the hardware details. New hardware may need a new Linux kernel to support. Therefore, different Linux platforms use patched kernels to provide special support for extended functionality.

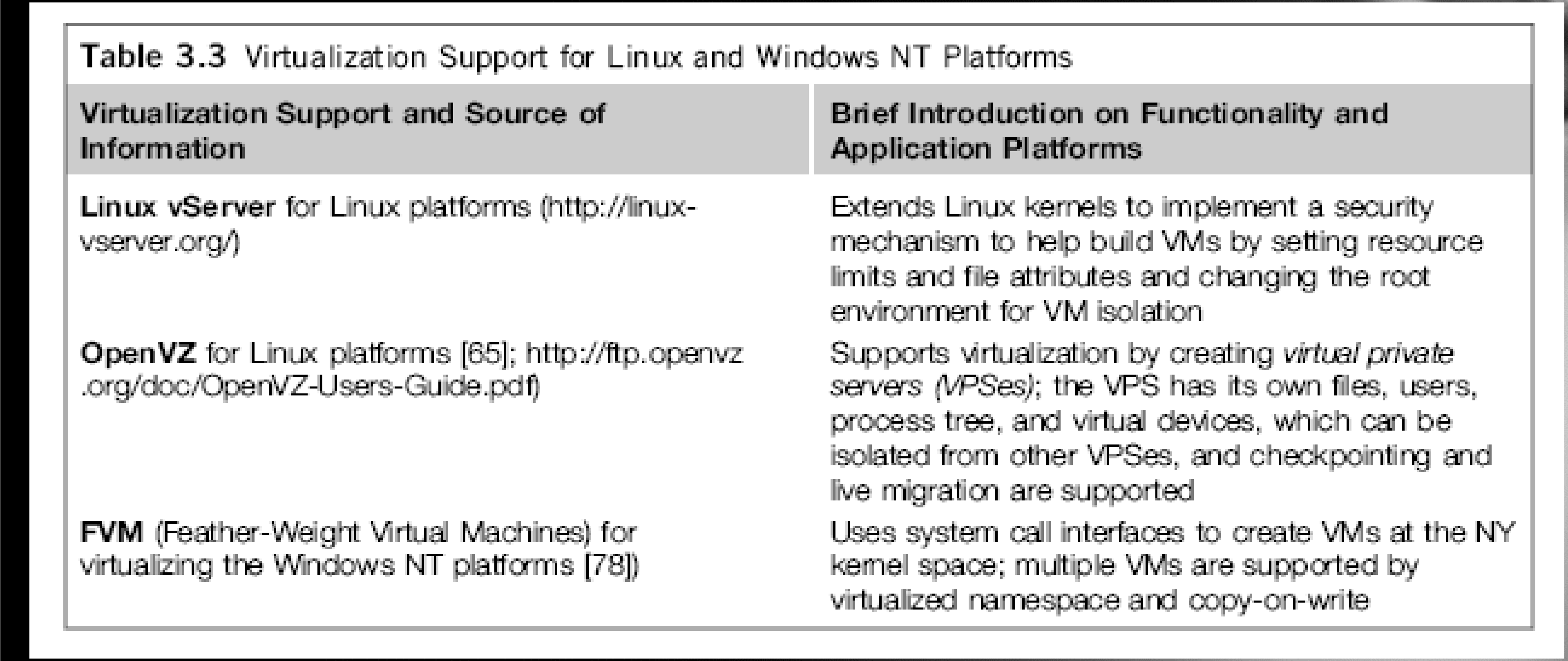


Table 3.3 Virtualization Support for Linux and Windows NT Platforms

Virtualization Support and Source of Information	Brief Introduction on Functionality and Application Platforms
Linux vServer for Linux platforms (http://linux-vserver.org/)	Extends Linux kernels to implement a security mechanism to help build VMs by setting resource limits and file attributes and changing the root environment for VM isolation
OpenVZ for Linux platforms [65]; http://ftp.openvz.org/doc/OpenVZ-Users-Guide.pdf)	Supports virtualization by creating *virtual private servers (VPSes)*; the VPS has its own files, users, process tree, and virtual devices, which can be isolated from other VPSes, and checkpointing and live migration are supported
FVM (Feather-Weight Virtual Machines) for virtualizing the Windows NT platforms [78])	Uses system call interfaces to create VMs at the NY kernel space; multiple VMs are supported by virtualized namespace and copy-on-write

Feather- Weight VM (FVM)

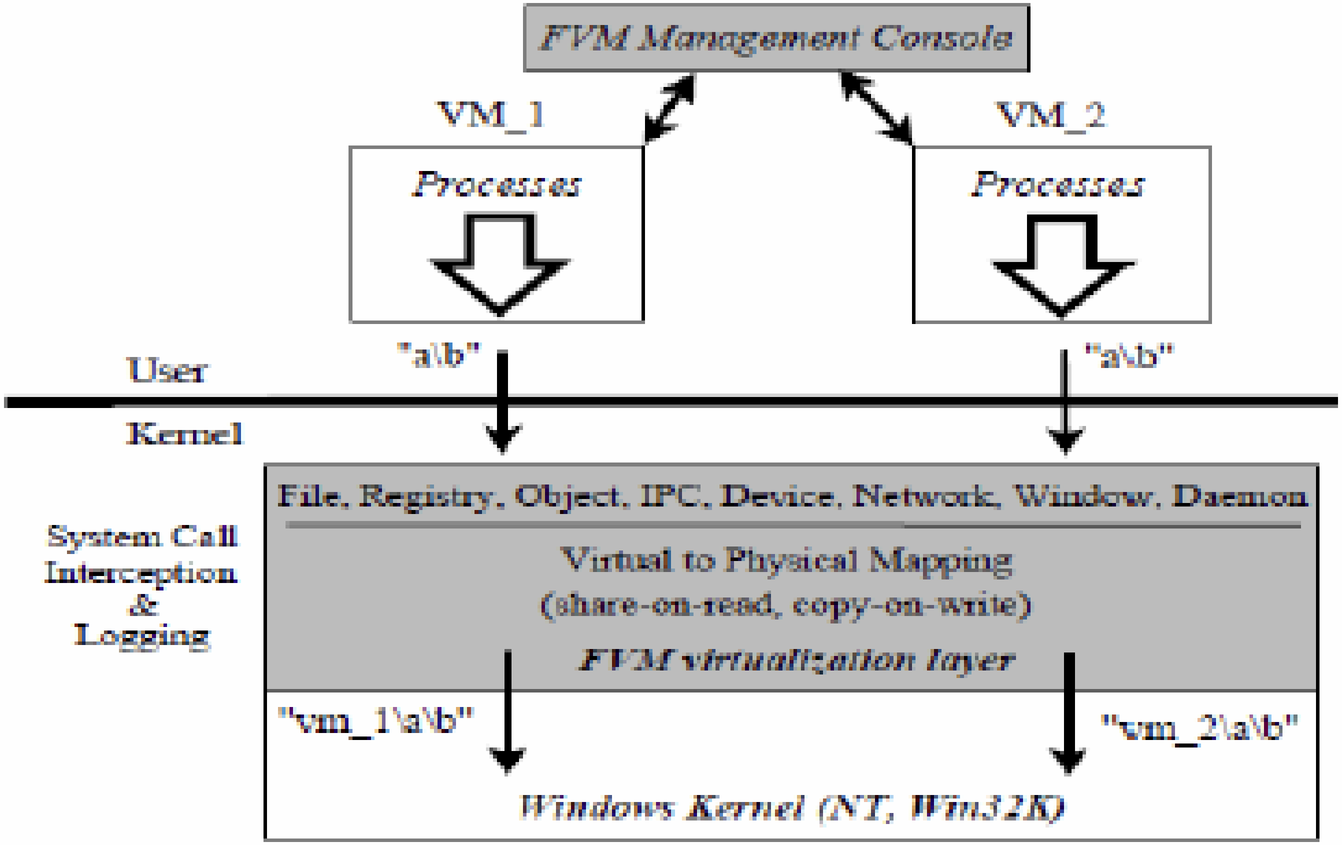
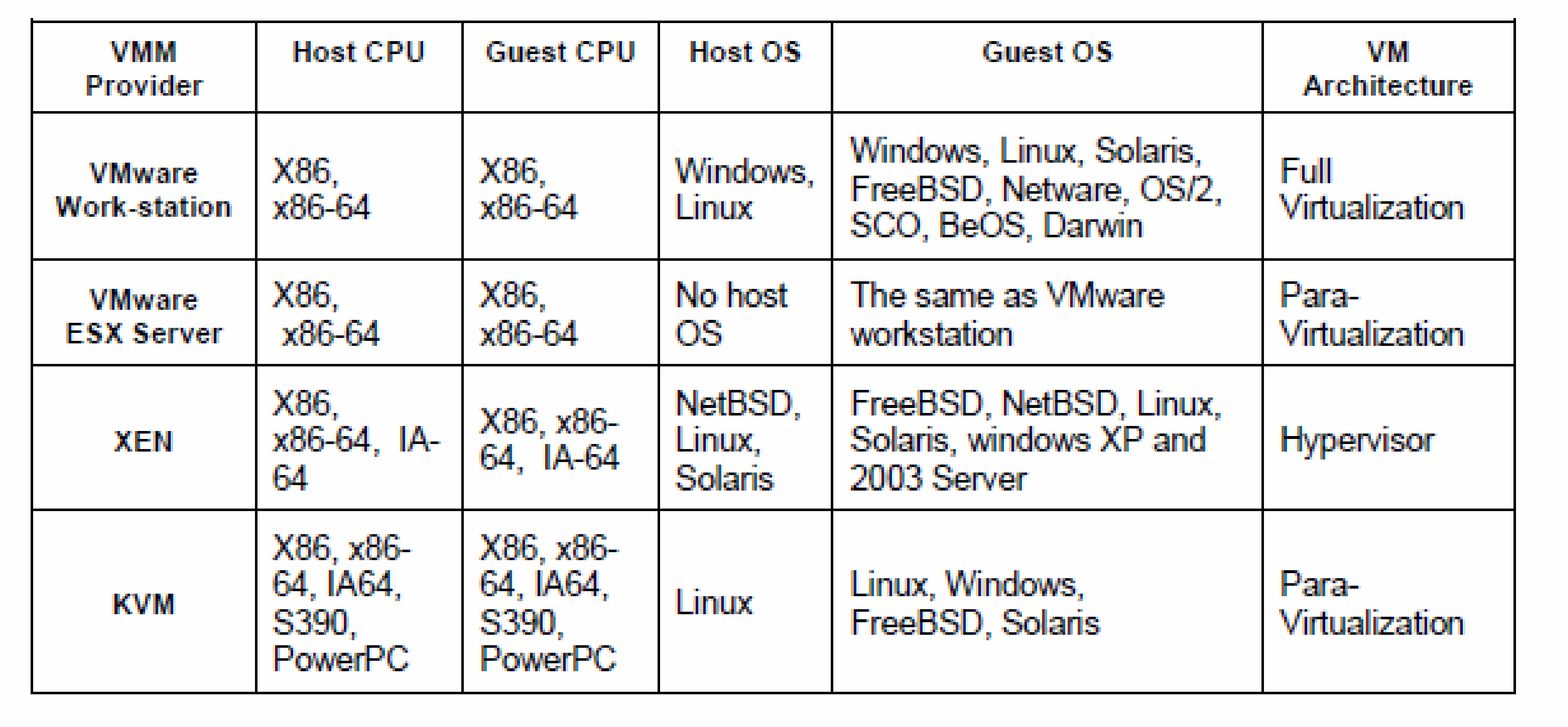

FVM Management Console
VM_1
VM_2
Processes
Processes
"a\b"
"a\b"
User
Kernel
System Call Interception & Logging
File, Registry, Object, IPC, Device, Network, Window, Daemon
Virtual to Physical Mapping
(share-on-read, copy-on-write)
FVM virtualization layer
"vm_1\a\b"
"vm_2\a\b"
Windows Kernel (NT, Win32K)

Figure 6.5 The FVM supports multiple VMs by namespace virtualization and copy-on-write. The management console allows users to perform FVM operations on specified VM. (Courtesy of Yang Yu's *Ph.D. Thesis:* OS-level Virtualization and Its Applications, December 2007, CS Dept., SUNY, Stony Brook, 2007 [71])

Major VMM and Hypervisor Providers



VMM Provider	Host CPU	Guest CPU	Host OS	Guest OS	VM Architecture
VMware Work-station	X86, x86-64	X86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server	X86, x86-64	X86, x86-64	No host OS	The same as VMware workstation	Para-Virtualization
XEN	X86, x86-64, IA-64	X86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, windows XP and 2003 Server	Hypervisor
KVM	X86, x86-64, IA64, S390, PowerPC	X86, x86-64, IA64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

The vCUBE for Virtualization of GPGPU

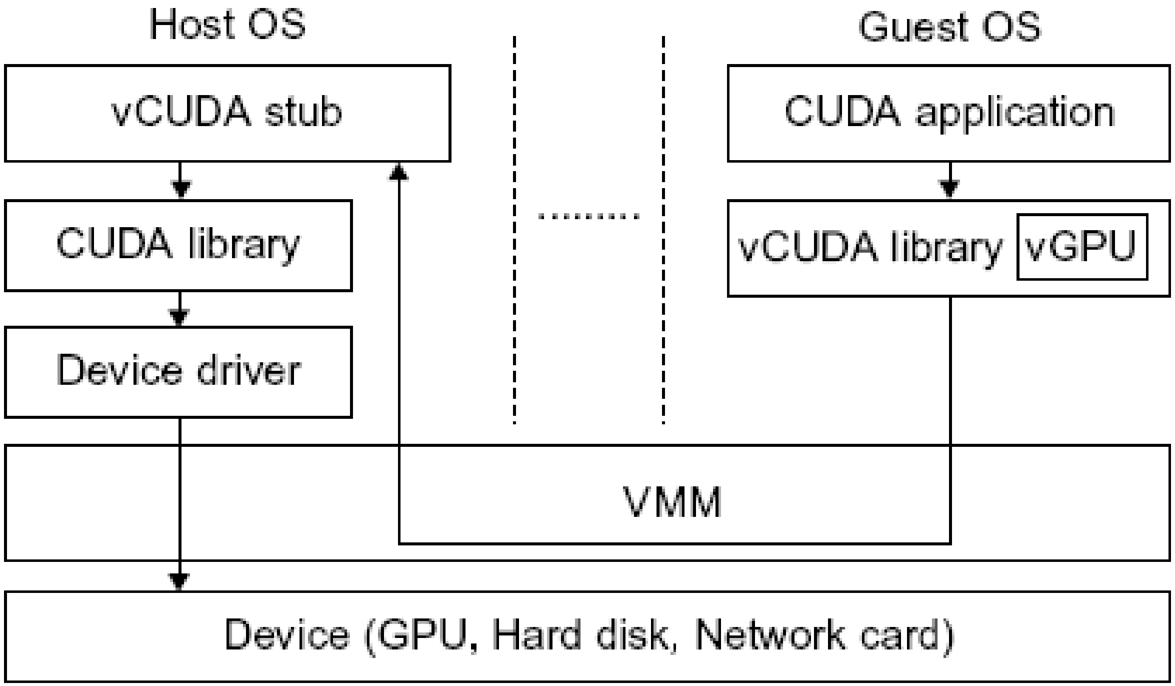


FIGURE 3.4

Basic concept of the vCUDA architecture.

(Courtesy of Lin Shi, et al. [57])

Hypervisor

A hypervisor is a hardware virtualization technique allowing multiple operating systems, called guests to run on a host machine. This is also called the Virtual Machine Monitor (VMM).

Type 1 hypervisor or the bare metal hypervisor sits on the bare metal computer hardware like the CPU, memory, etc. All the guest operating systems are a layer above the hypervisor. So the hypervisor is the first layer over the hardware. The original CP/CMS hypervisor developed by IBM was of this kind. Examples are Microsoft Hyper-V.

Type 2 or the hosted hypervisor do not run over the bare metal hardware but they run over a host operating system. The hypervisor is the second layer over the hardware. The guest operating systems run a layer over the hypervisor and so they form the third layer. Examples are FreeBSD. The operating system is usually unaware of the virtualization

Hypervisor and the XEN Architecture

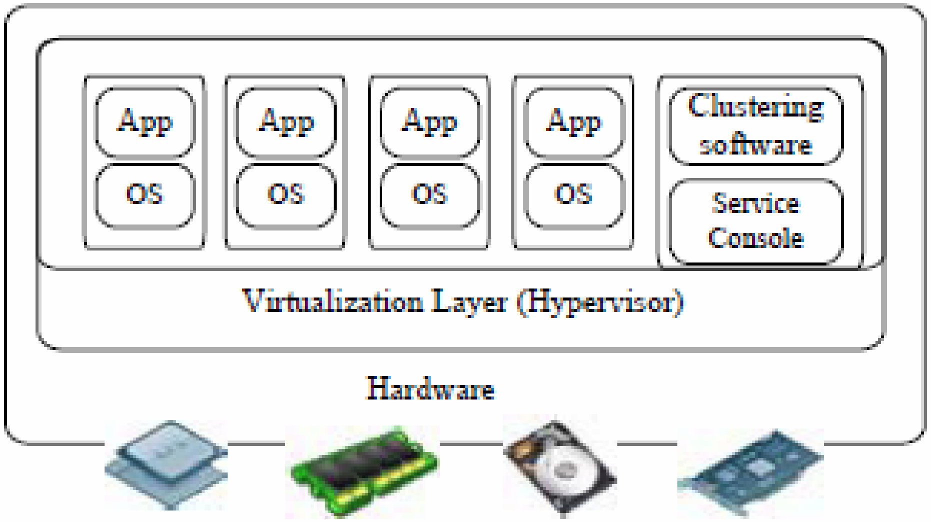
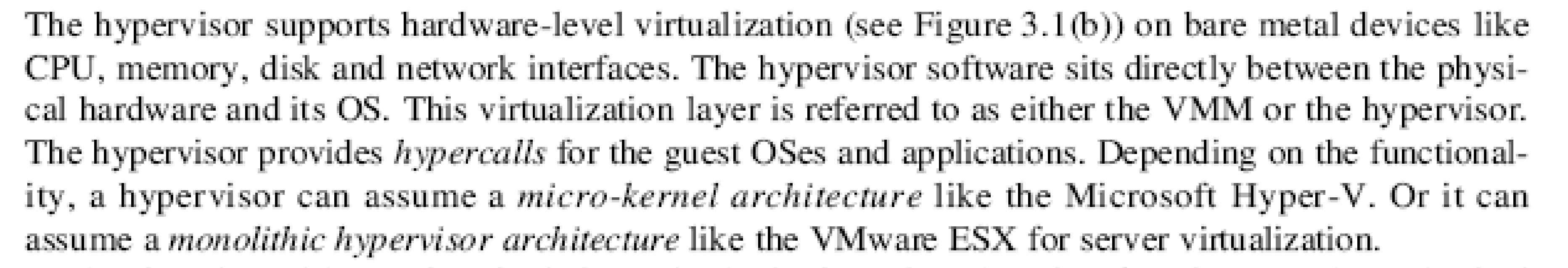


Figure 6.7 A hypervisor is the software layer for virtualization of the bare metal hardware. This layer can be implemented as a micro-kernel of the OS. It converts physical devices into virtual resources for user applications to run on the virtual machines deployed.



The hypervisor supports hardware-level virtualization (see Figure 3.1(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides *hypercalls* for the guest OSes and applications. Depending on the functionality, a hypervisor can assume a *micro-kernel architecture* like the Microsoft Hyper-V. Or it can assume a *monolithic hypervisor architecture* like the VMware ESX for server virtualization.

The XEN Architecture (1)

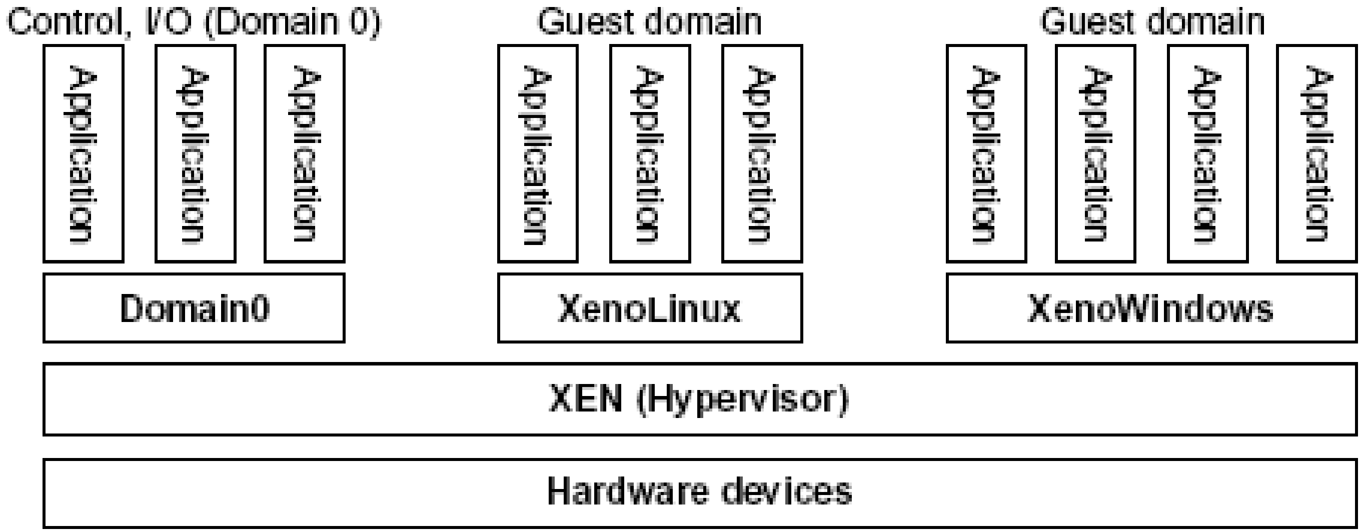
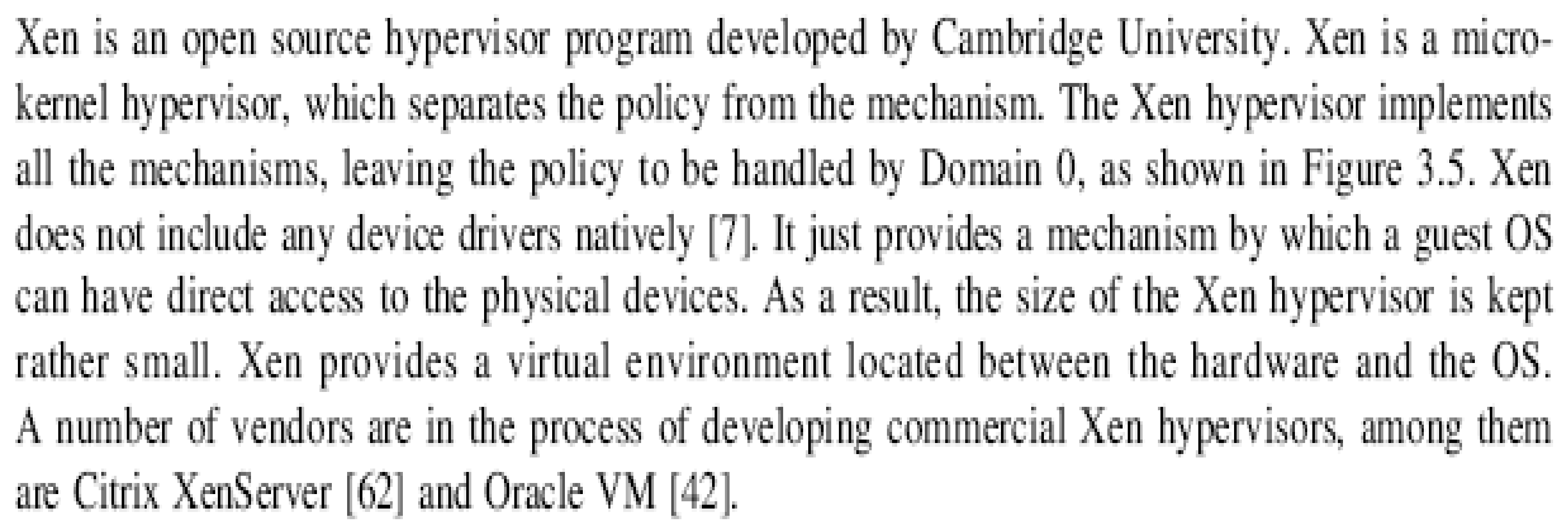


FIGURE 3.5

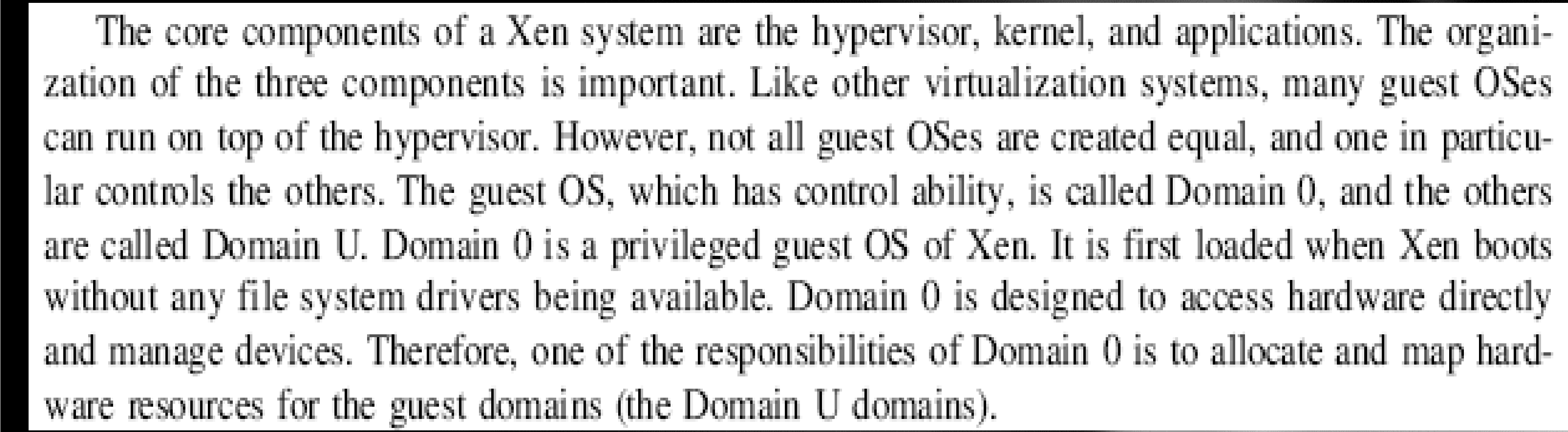
The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

The XEN Architecture (2)

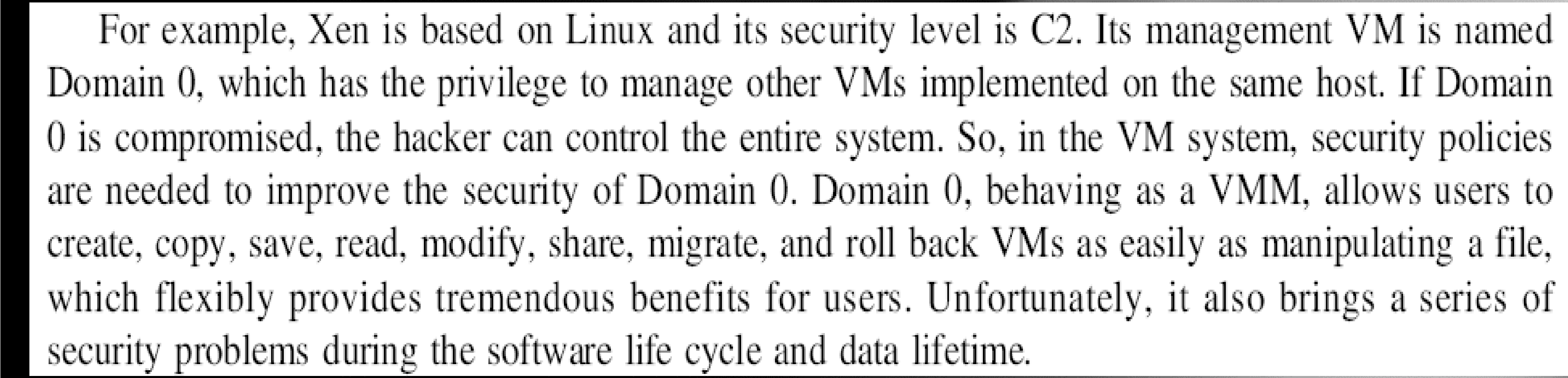


Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure 3.5. Xen does not include any device drivers natively [7]. It just provides a mechanism by which a guest OS can have direct access to the physical devices. As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS. A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer [62] and Oracle VM [42].

The XEN Architecture (3)



The core components of a Xen system are the hypervisor, kernel, and applications. The organization of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor. However, not all guest OSes are created equal, and one in particular controls the others. The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices. Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).



For example, Xen is based on Linux and its security level is C2. Its management VM is named Domain 0, which has the privilege to manage other VMs implemented on the same host. If Domain 0 is compromised, the hacker can control the entire system. So, in the VM system, security policies are needed to improve the security of Domain 0. Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate, and roll back VMs as easily as manipulating a file, which flexibly provides tremendous benefits for users. Unfortunately, it also brings a series of security problems during the software life cycle and data lifetime.

Para-Virtualization

In para-virtualization, the guest operating system has to be modified. Para-virtualization provides specially defined 'hooks' to do some tasks in the host and guest operating systems, which would otherwise have been done in a virtual environment, which is slower. The VMM in a para-virtualized platform is simpler because the critical tasks are now performed in the operating system rater than by the VMM. Since the virtualization overhead decreases the performance increases.

Some of the disadvantages are the compatibility and the portability is reduces because of the modified operating system. Also the cost of maintenance is high because of the deep OS modifications.
Some examples of Para-virtualization are KVM and the XenWindowsGplPv project.

Para-Virtualization with Compiler Support.

The KVM builds offers kernel-based VM on the Linux platform, based on para-virtualization

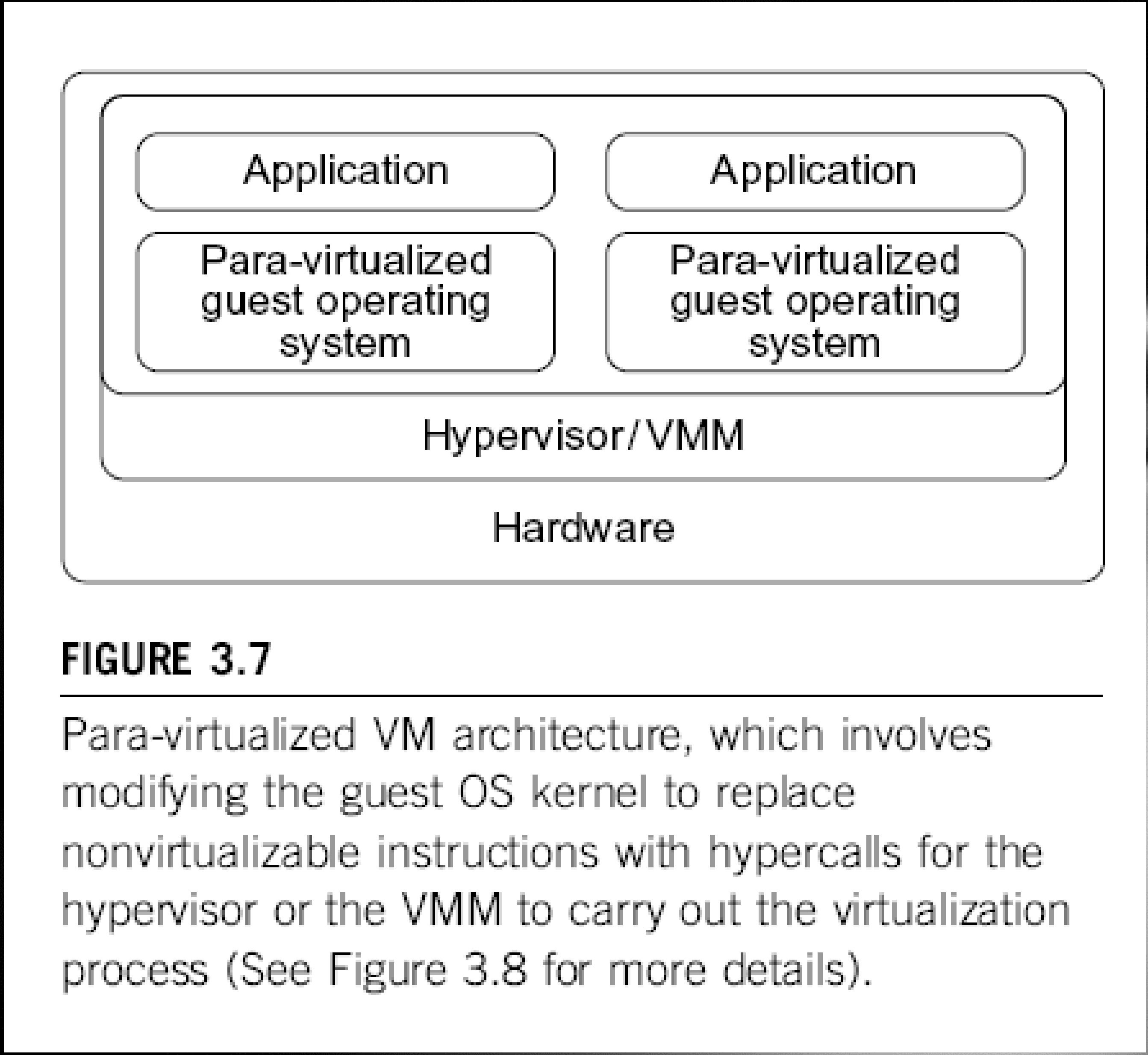

Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor/VMM
Hardware

FIGURE 3.7

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process (See Figure 3.8 for more details).

Host-based Virtualization

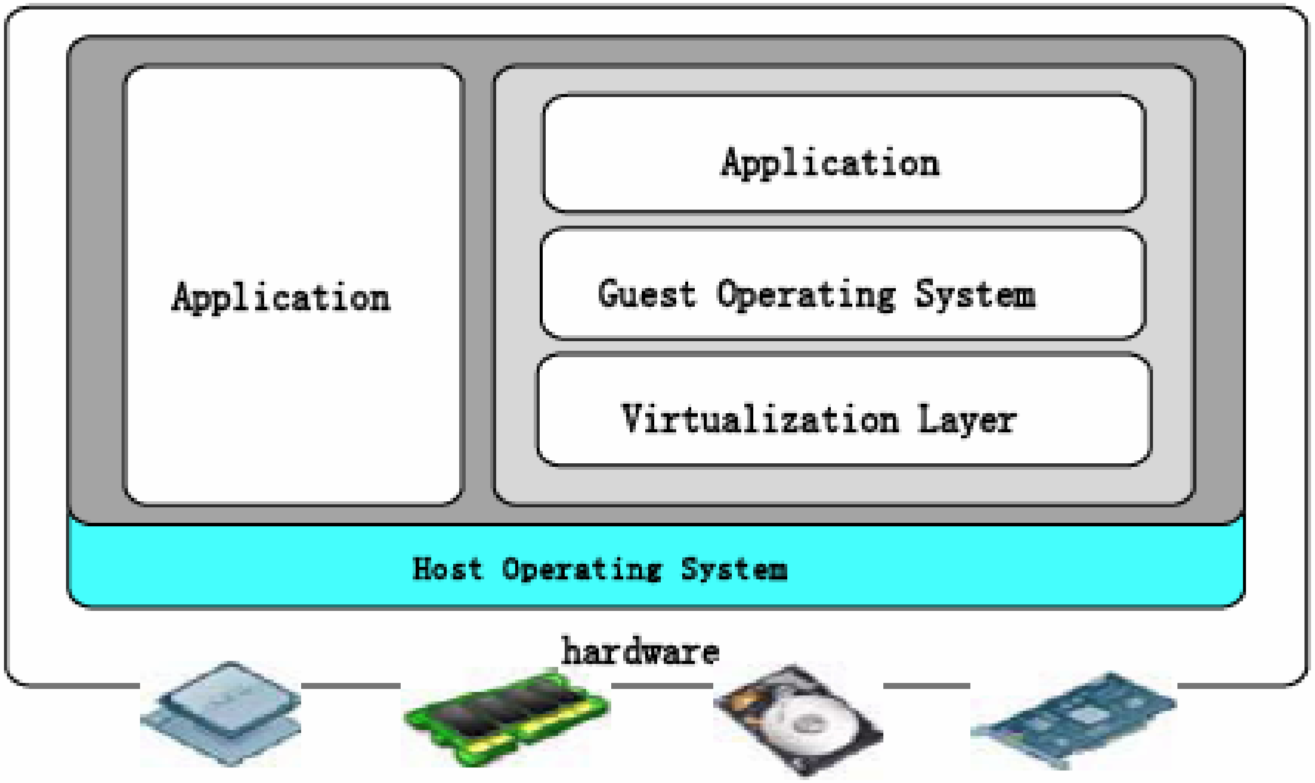


Figure 6.11 A hosted VM installs a guest OS on top of the host OS. This differs from the full virtualization architecture shown in Figure 6.9.

Virtual Cores vs. Physical Processor Cores

Physical cores	Virtual cores
The actual physical cores present in the processor.	There can be more virtual cores visible to a single OS than there are physical cores.
More burden on the software to write applications which can run directly on the cores.	Design of software becomes easier as the hardware assists the software in dynamic resource utilization.
Hardware provides no assistance to the software and is hence simpler.	Hardware provides assistance to the software and is hence more complex.
Poor resource management.	Better resource management.
The lowest level of system software has to be modified.	The lowest level of system software need not be modified.

Binary Translation of Guest OS Requests using a VMM:

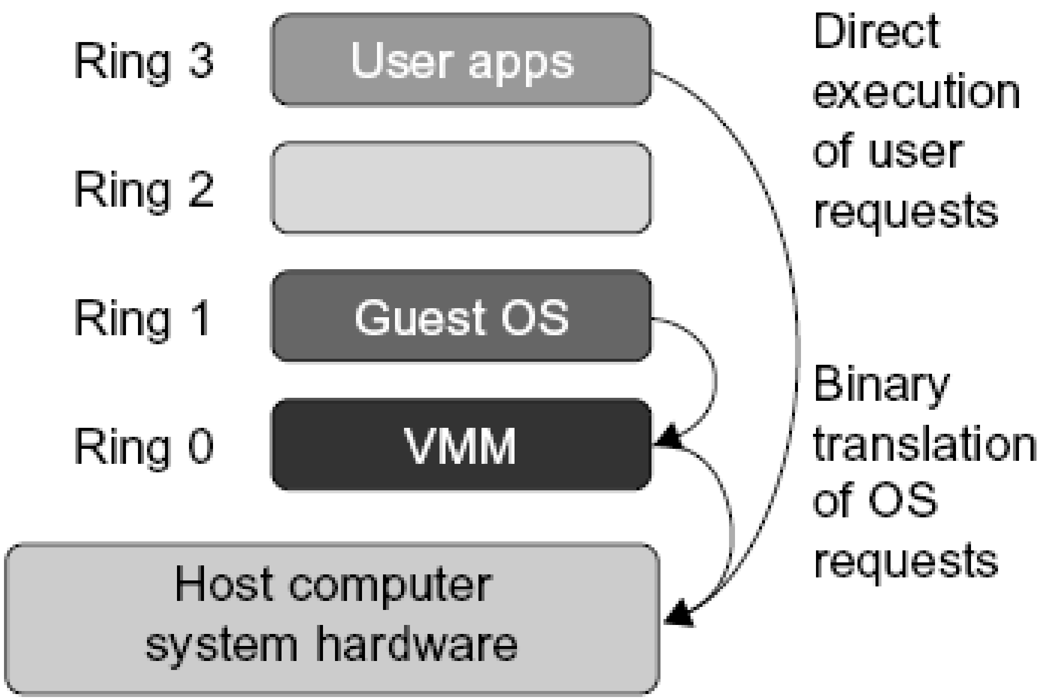

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

FIGURE 3.6

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

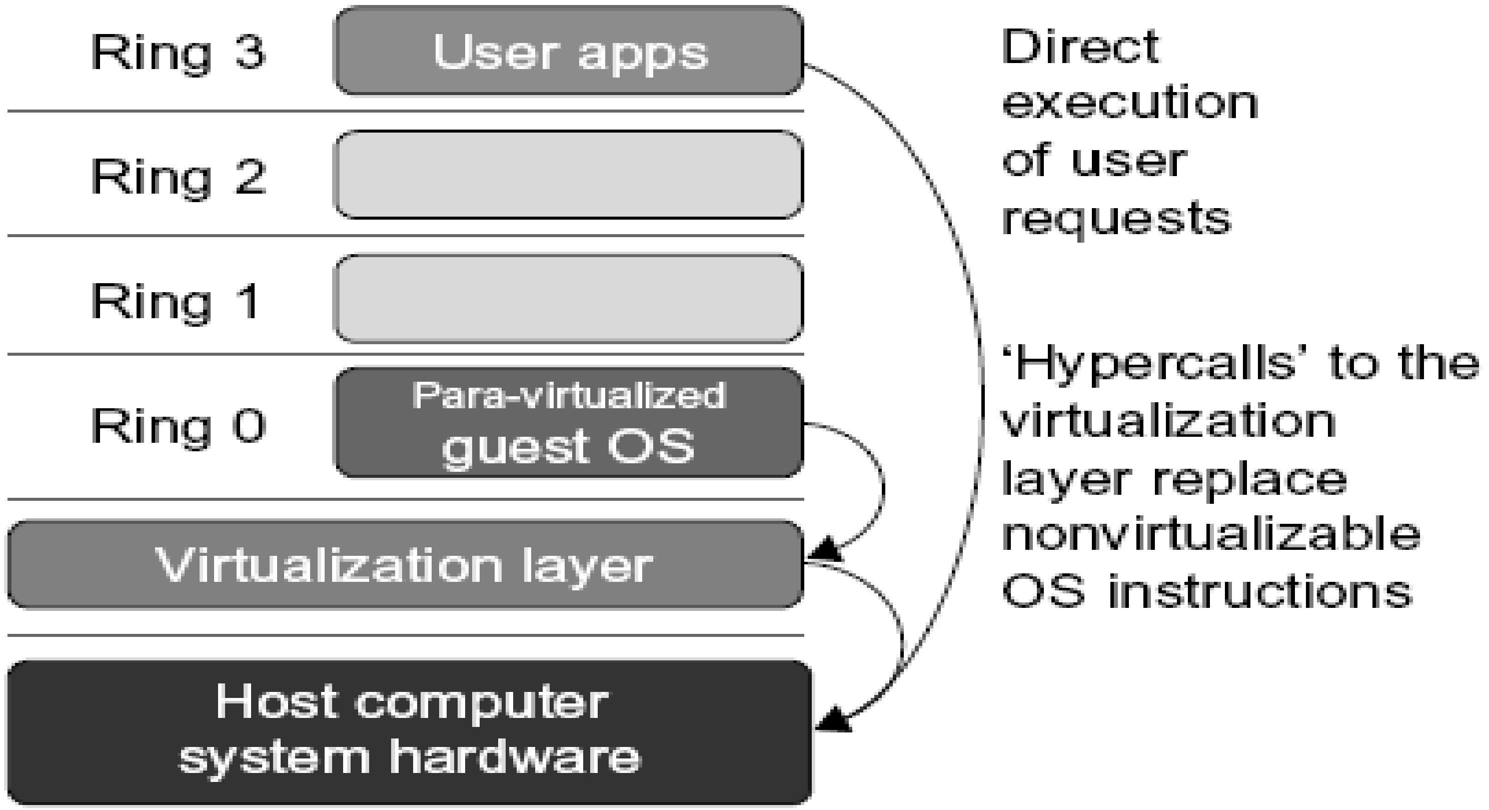

Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

FIGURE 3.8

The Use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

Hypercall Execution

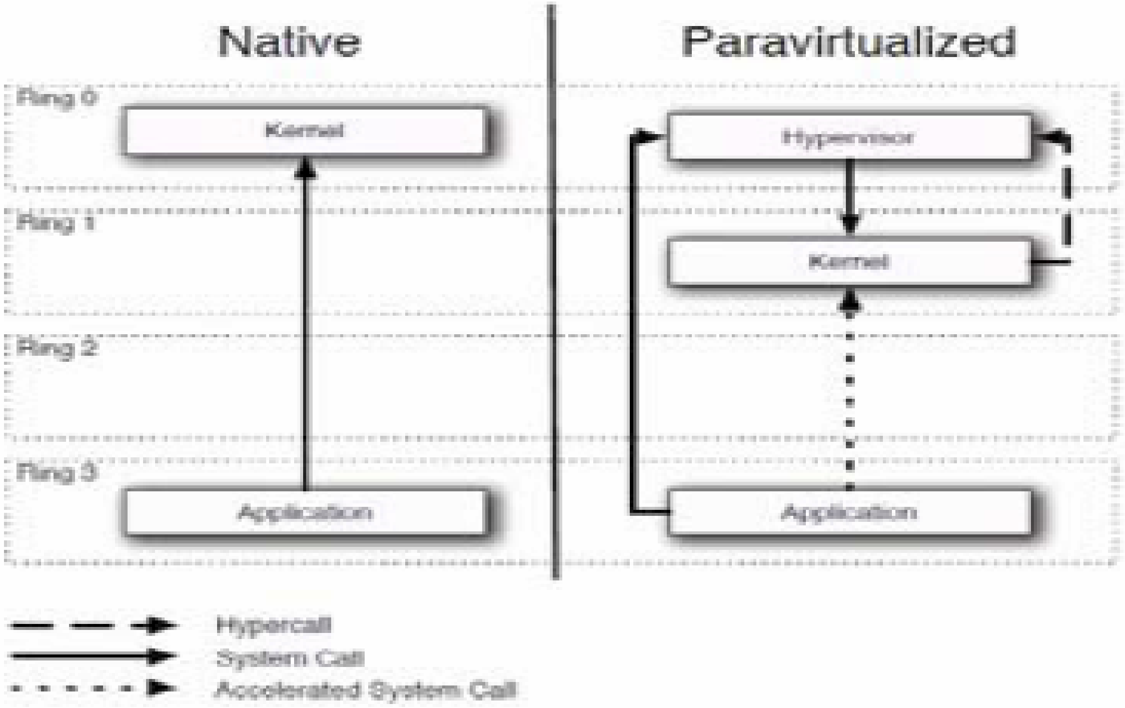


Figure 6.15. When an application VM issues a guest OS system call, it interrupts the hypervisor to handle and then pass control back to the guest OS. (Courtesy of "the definitive guide to the XEN hypervisor" by David Chisnall [13]).

VMWare ESX Server for Para-Virtualization

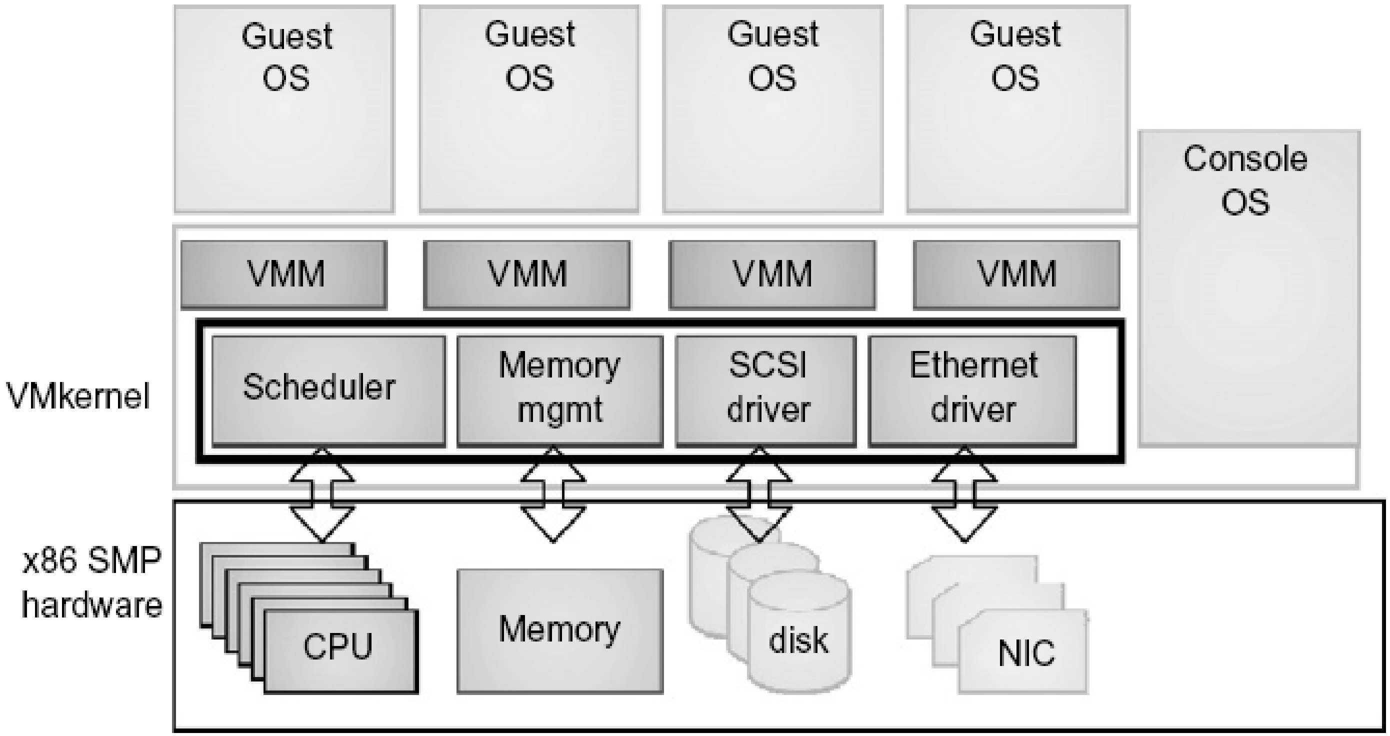


FIGURE 3.9

The VMware ESX server architecture using para-virtualization.

(Courtesy of VMware [71])

Virtualization Support at Intel

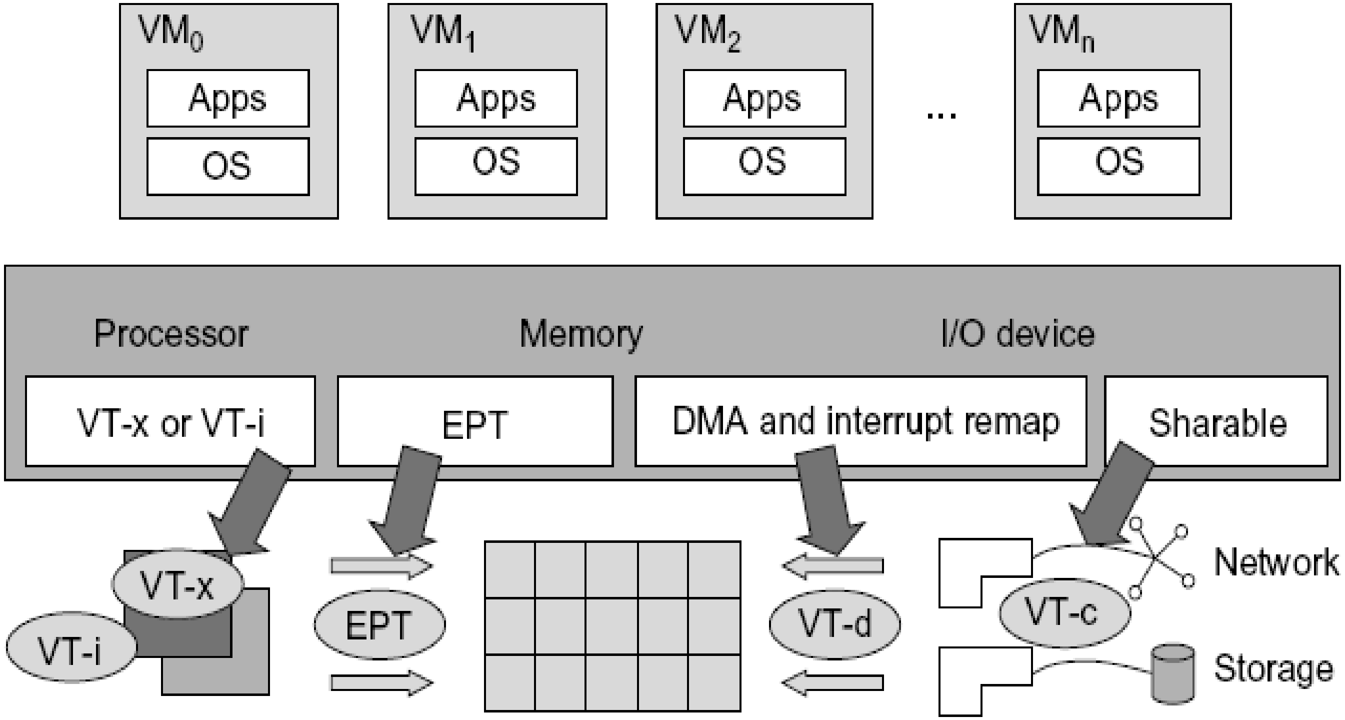
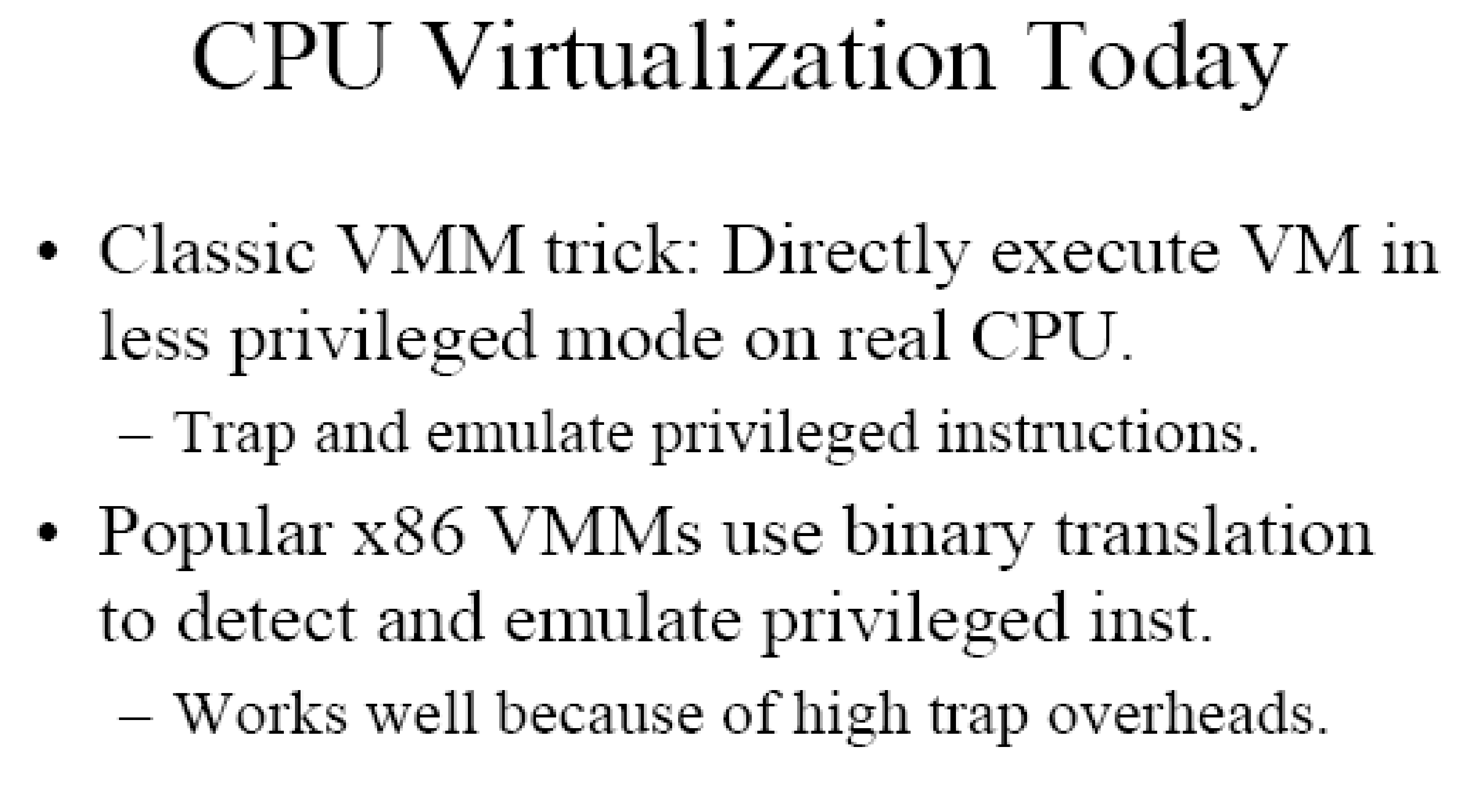


FIGURE 3.10

Intel hardware support for virtualization of processor, memory, and I/O devices.

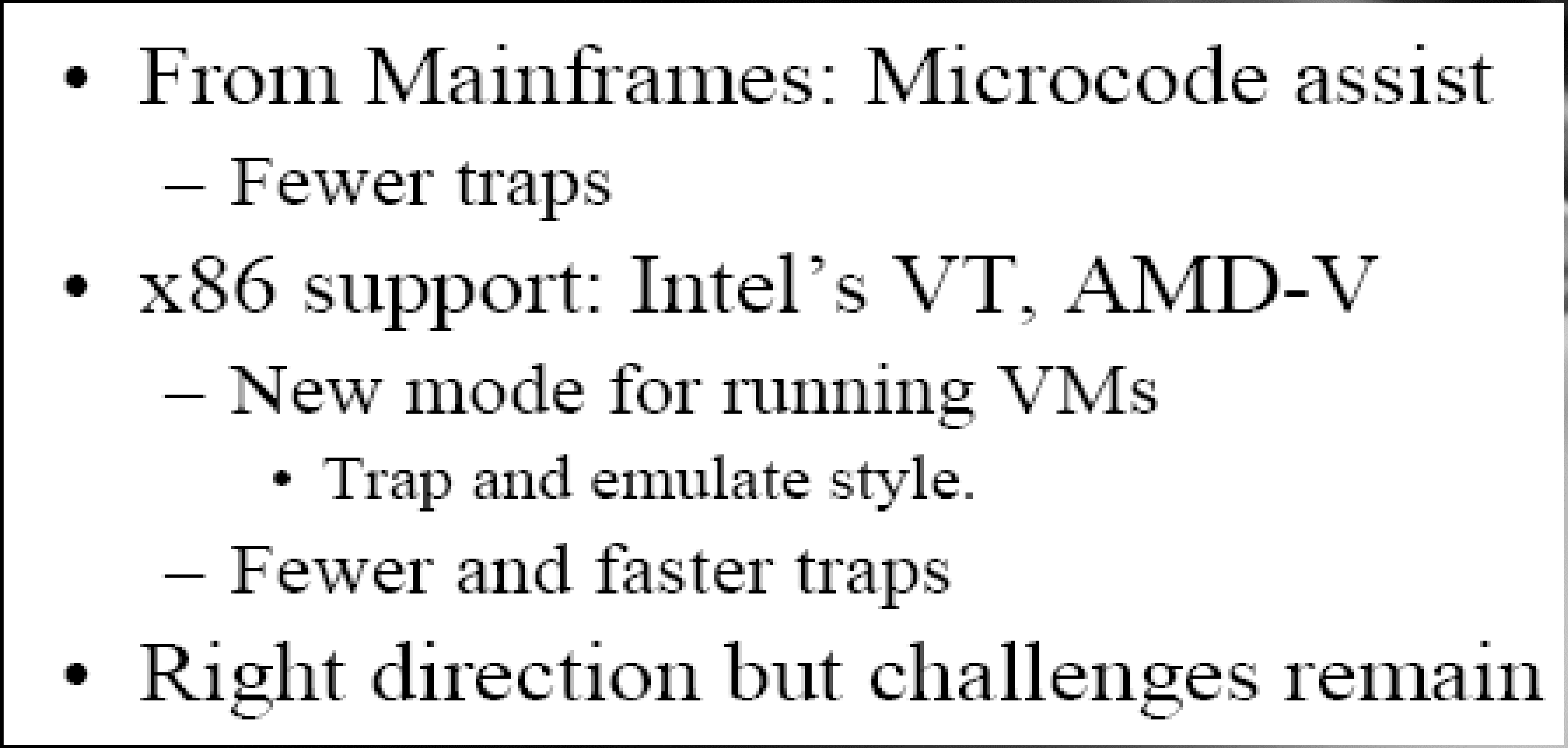
(Modified from [68], Courtesy of Lizhong Chen, USC)



CPU Virtualization Today

- Classic VMM trick: Directly execute VM in less privileged mode on real CPU.
  - Trap and emulate privileged instructions.
- Popular x86 VMMs use binary translation to detect and emulate privileged inst.
  - Works well because of high trap overheads.

(Courtesy of Mendel Rosenblum, 2006)



- From Mainframes: Microcode assist
  - Fewer traps
- x86 support: Intel’s VT, AMD-V
  - New mode for running VMs
    - Trap and emulate style.
  - Fewer and faster traps
- Right direction but challenges remain

(Courtesy of Mendel Rosenblum, 2006)

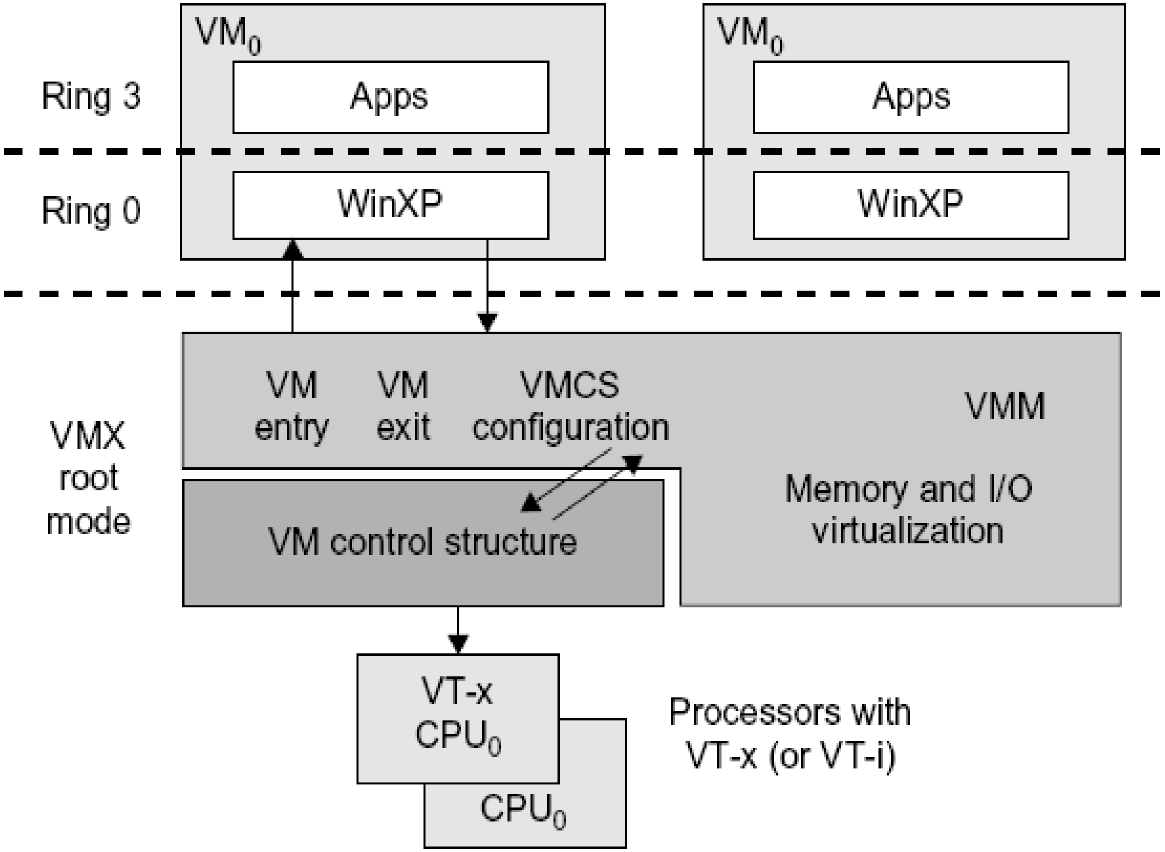


FIGURE 3.11

Intel Hardware-assisted CPU virtualization.

(Modified from [68], Courtesy of Lizhong Chen, USC)

Memory Virtualization Challenges

Address Translation

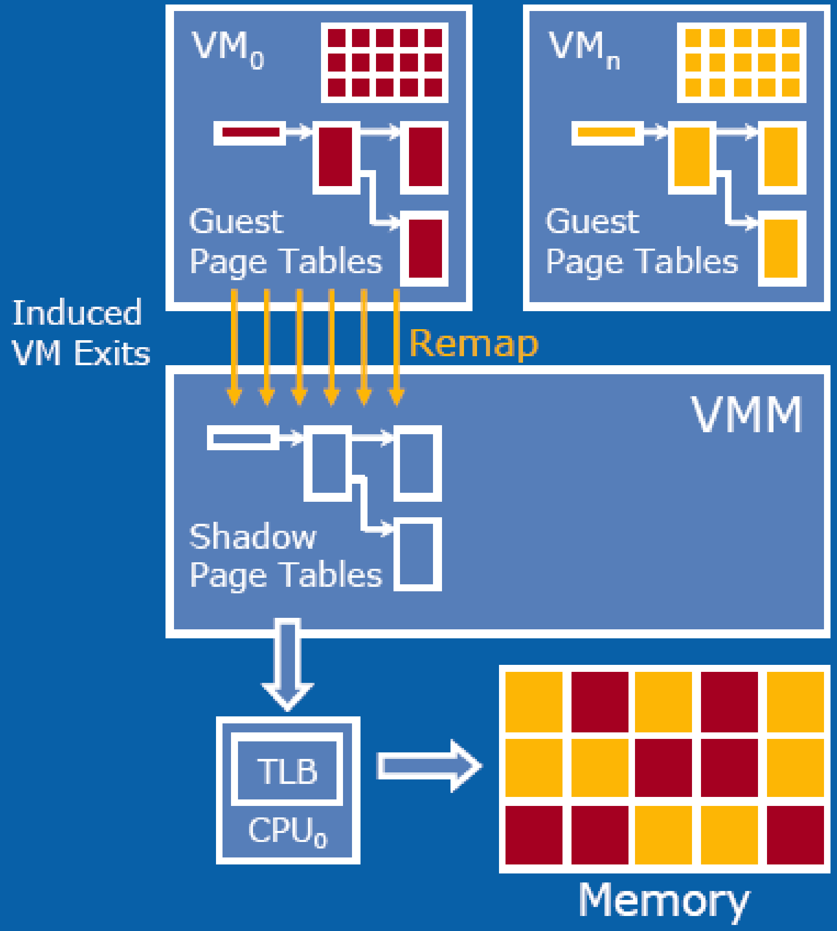
- Guest OS expects contiguous, zero-based physical memory
- VMM must preserve this illusion

Page-table Shadowing

- VMM intercepts paging operations
- Constructs copy of page tables

Overheads

- VM exits add to execution time
- Shadow page tables consume significant host memory



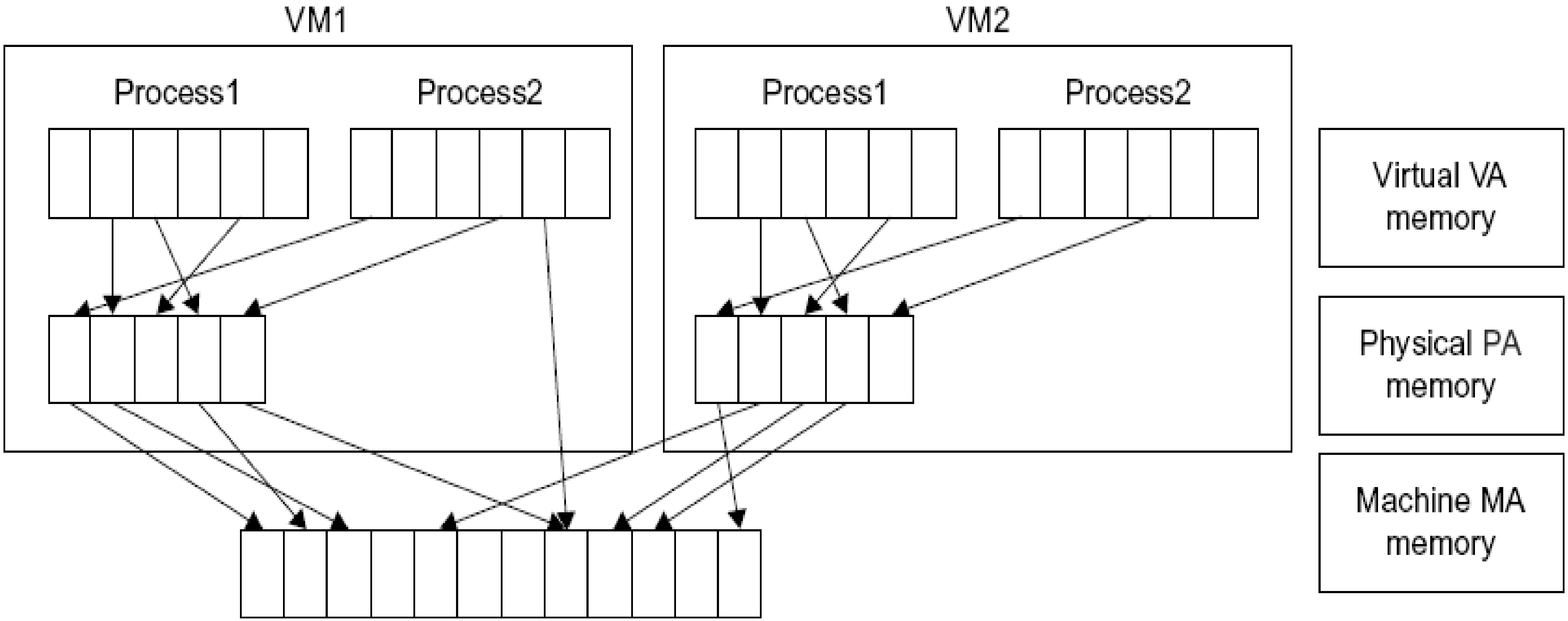
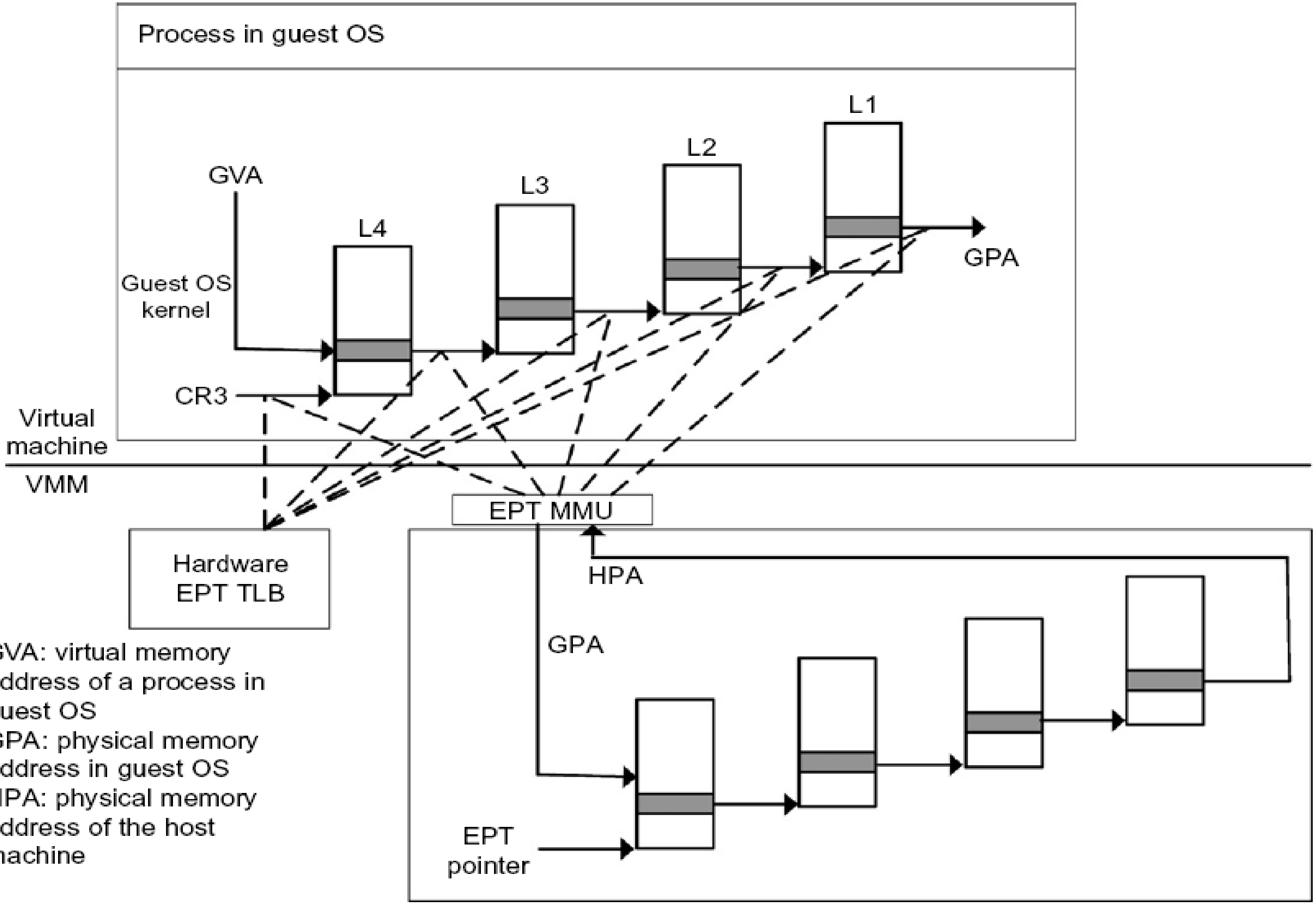


FIGURE 3.12

Two-level memory mapping procedure.

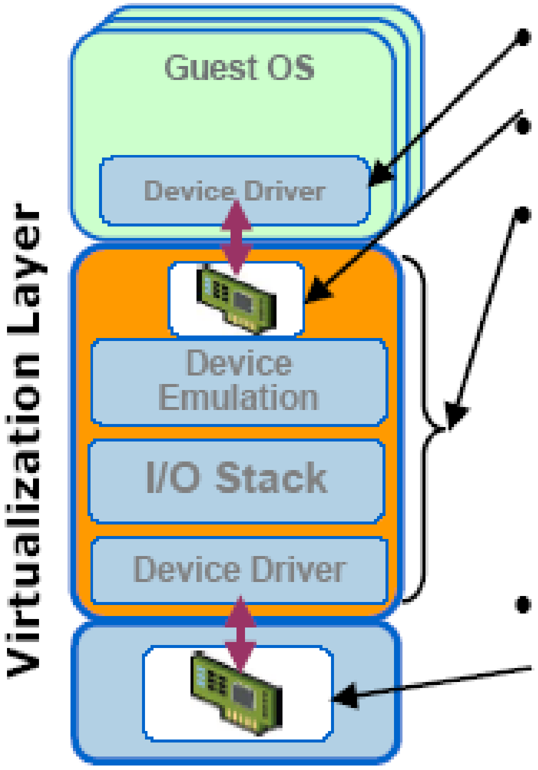


GVA: virtual memory address of a process in guest OS
GPA: physical memory address in guest OS
HPA: physical memory address of the host machine

FIGURE 3.13

Memory Virtualization Using EPT by Intel (the EPT Is Also Known As the Shadow Page Table [68]).

Current virtual I/O devices



- Guest device driver
- Virtual device
- Virtualization layer
  - emulates the virtual device
  - remaps guest and real I/O addresses
  - multiplexes and drives the physical device
  - I/O features, *e.g.*, COW disks,
- Real device
  - may be different from virtual device

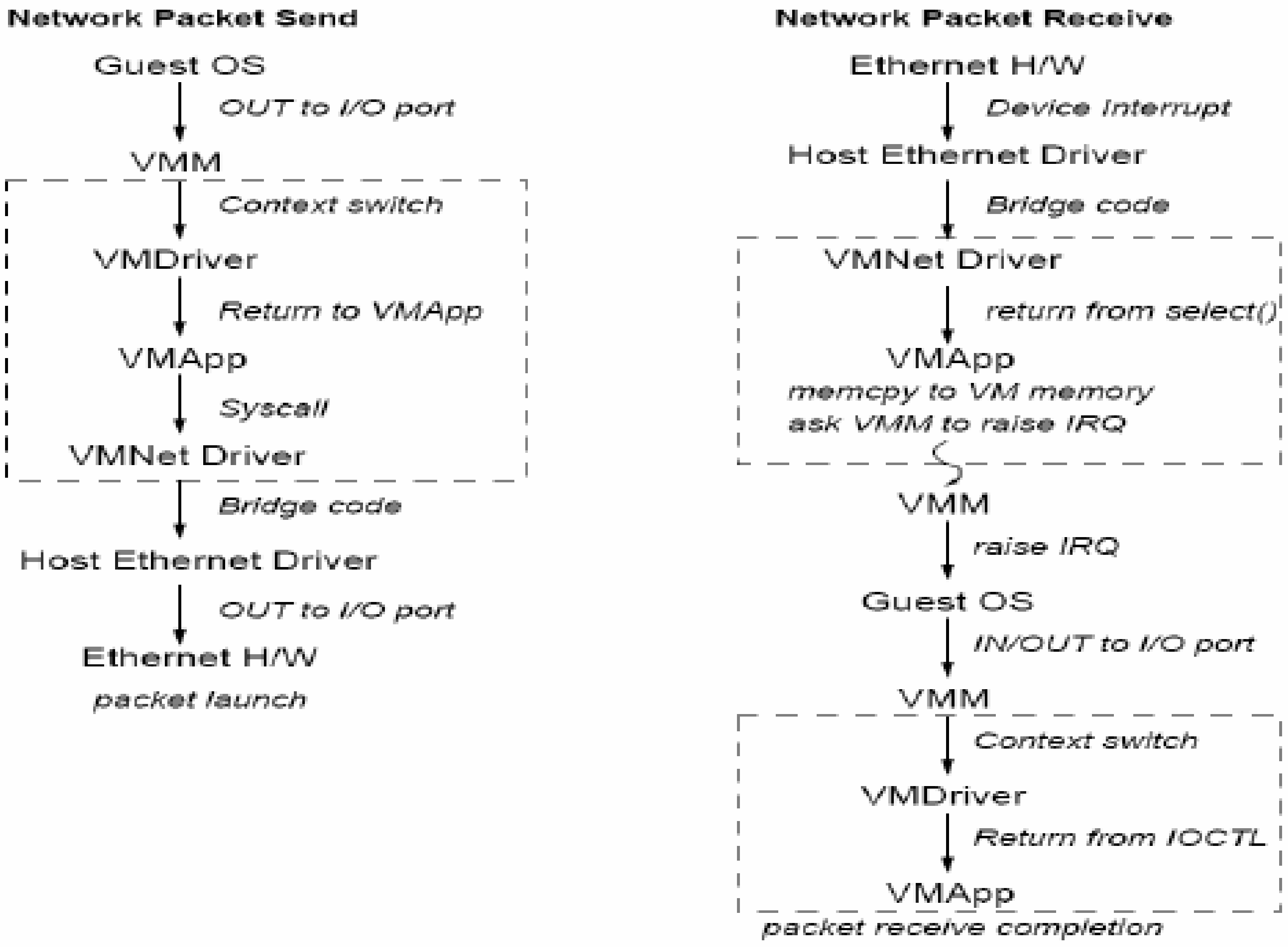


Figure 6.22 Functional blocks in the VMware workstation involved in a VM send and receive of network packets.

(Courtesy of VMWare, 2008)

Conclusions on CPU, Memory and I/O Virtualization :

- CPU virtualization demands hardware-assisted traps of sensitive instructions by the VMM
- Memory virtualization demands special hardware support (shadow page tables by VMWare or extended page table by Intel) to help translate virtual address into physical address and machine memory in two stages.
- I/O virtualization is the most difficult one to realize due to the complexity if I/O service routines and the emulation needed between the guest OS and host OS.

Multi-Core Virtualization:

VCPU vs. traditional CPU

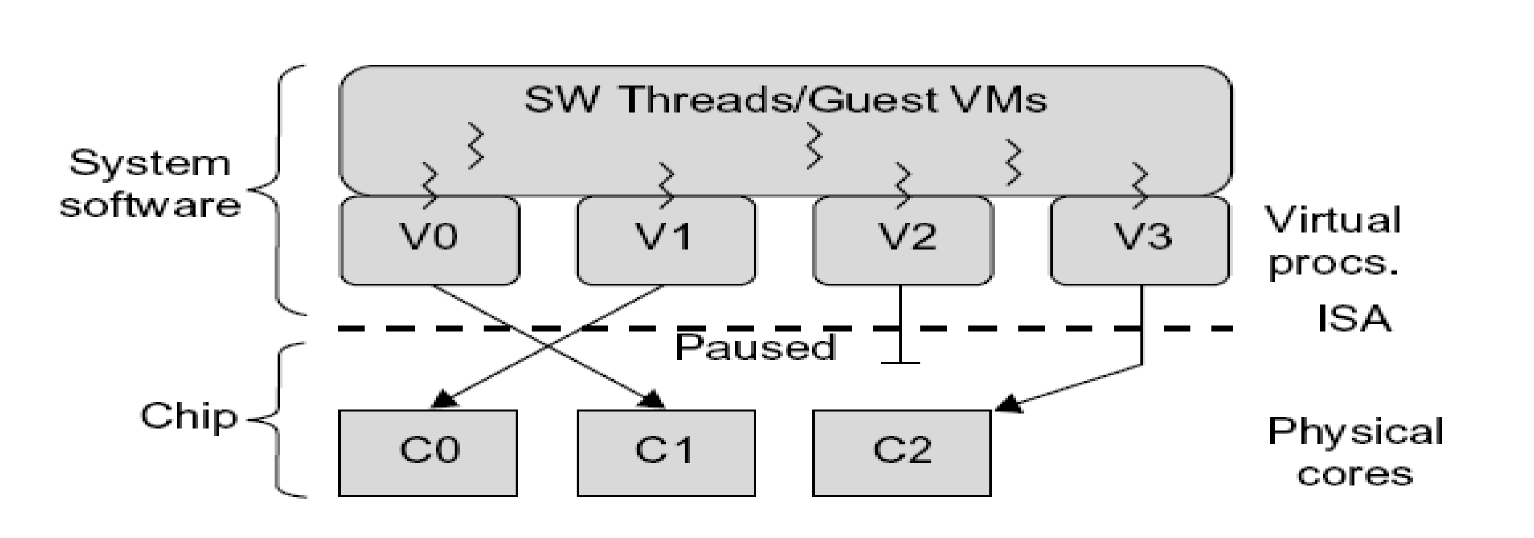
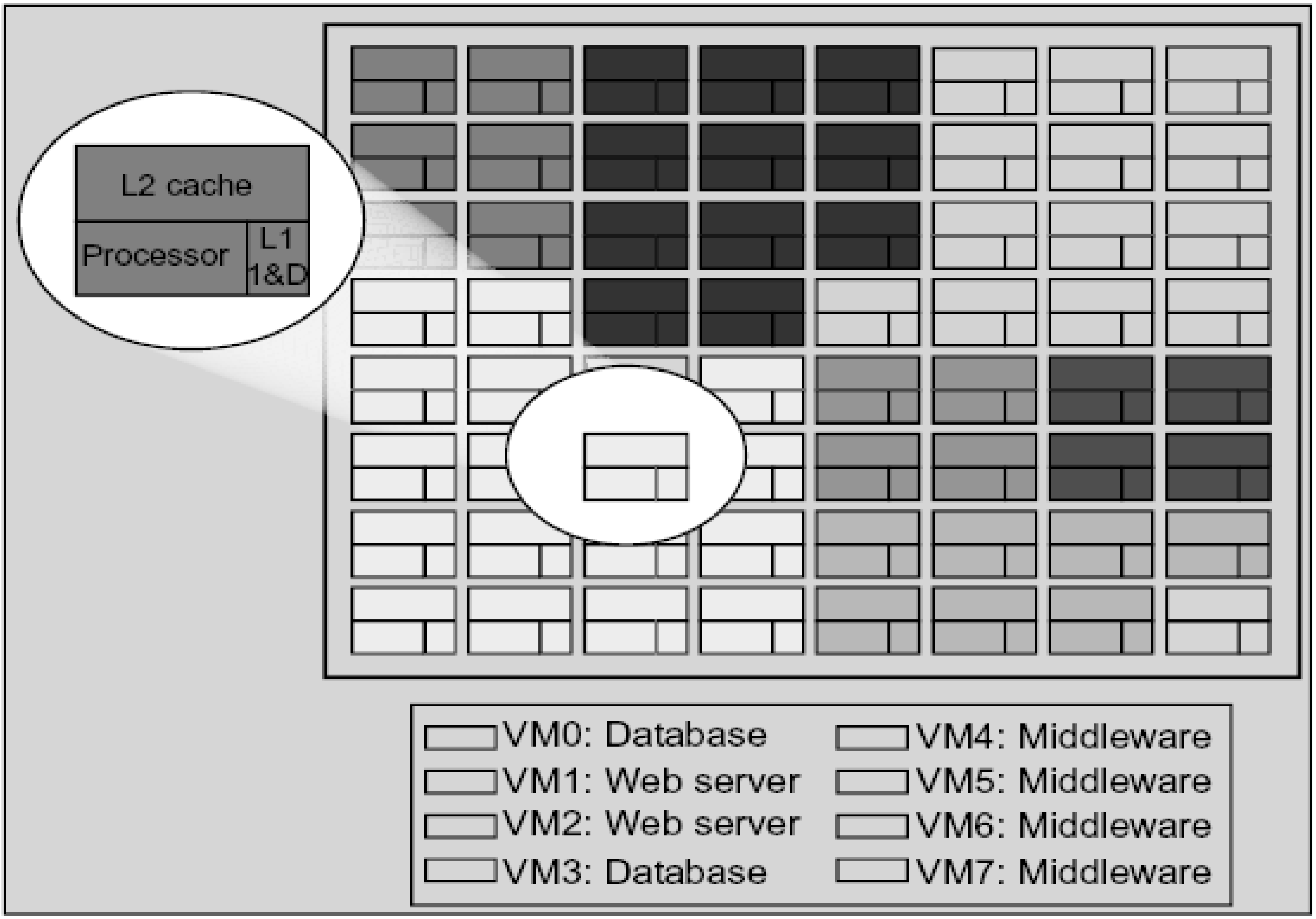


Figure 3.16 Four VCPUs are exposed the software, only three cores are actually present. VCPUs V0, V1, and V3 have been transparently migrated, while VCPU V2 has been transparently suspended. (Courtesy of Wells, et al., "Dynamic Heterogeneity and the Need for Multicore Virtualization", *ACM SIGOPS Operating Systems Review,* ACM Press, 2009 [68])

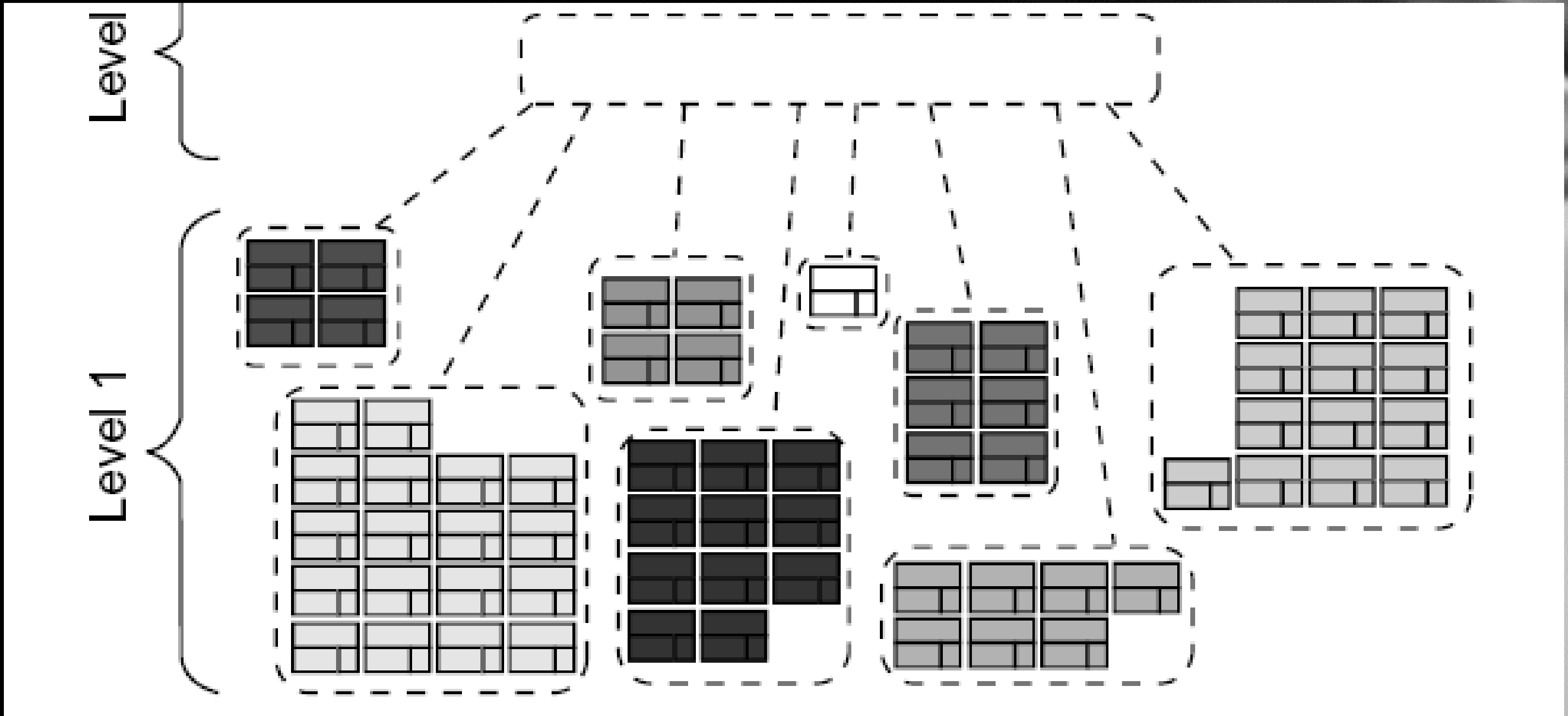


(a) Mapping of VMs into adjacent cores

(Courtesy of Marty and Hill, 2007)

Virtual Clusters in Many Cores

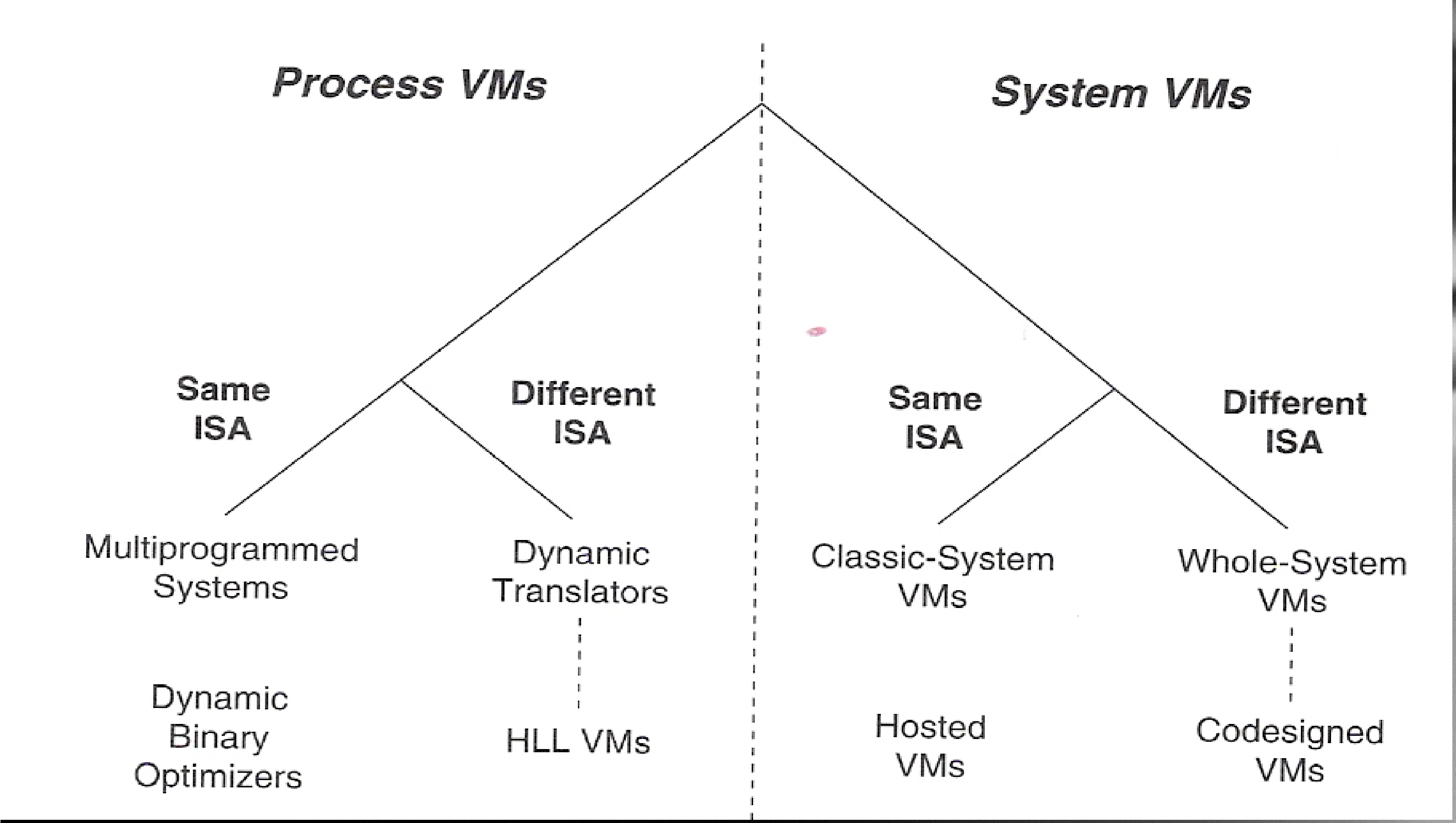
Space Sharing of VMs -- Virtual Hierarchy



(b) Multiple virtual clusters assigned to various workloads

(Courtesy of Marty and Hill, 2007)

A Taxonomy of Virtual Machines



(Courtesy of Smith and Nail, 2005)

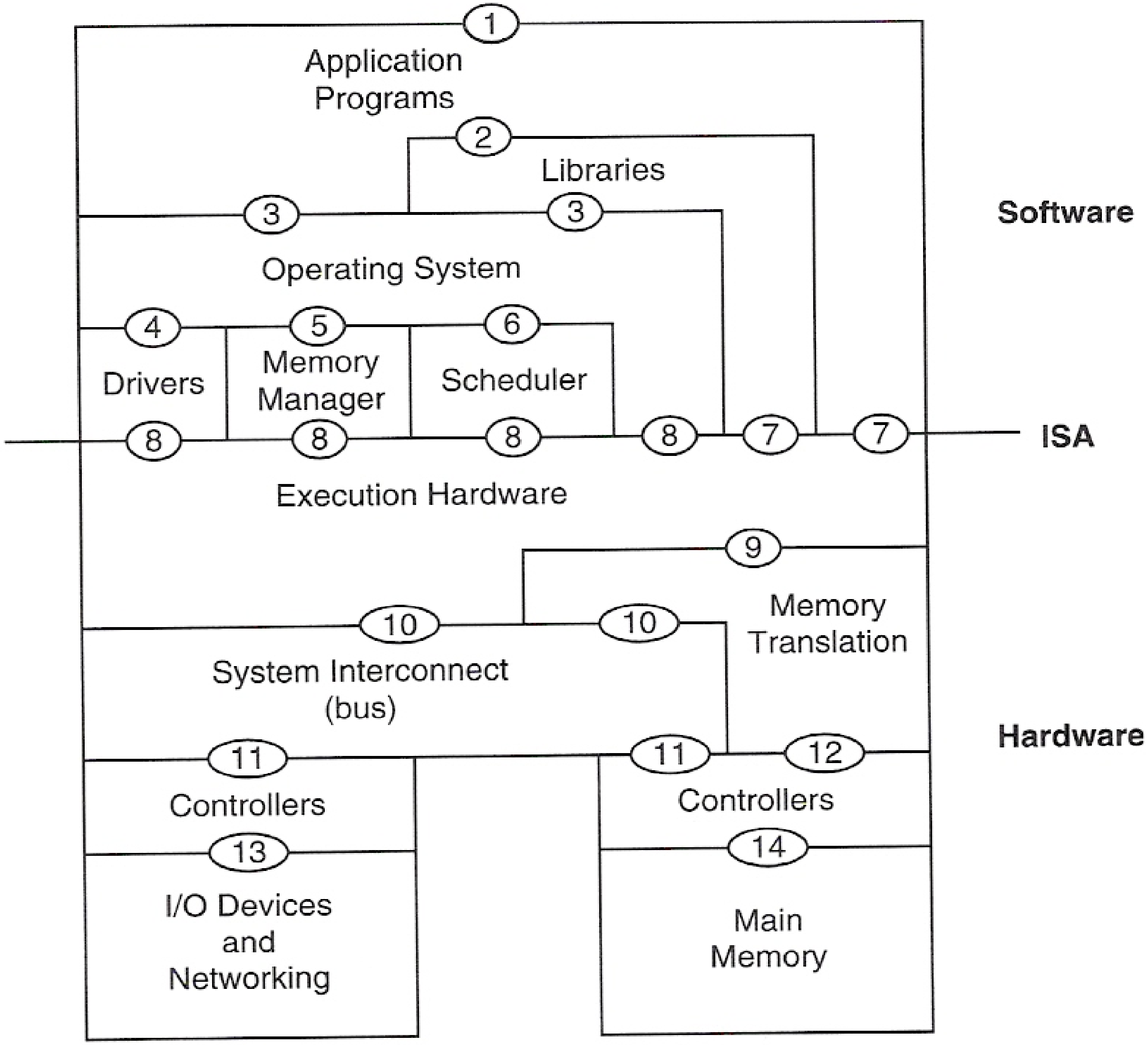
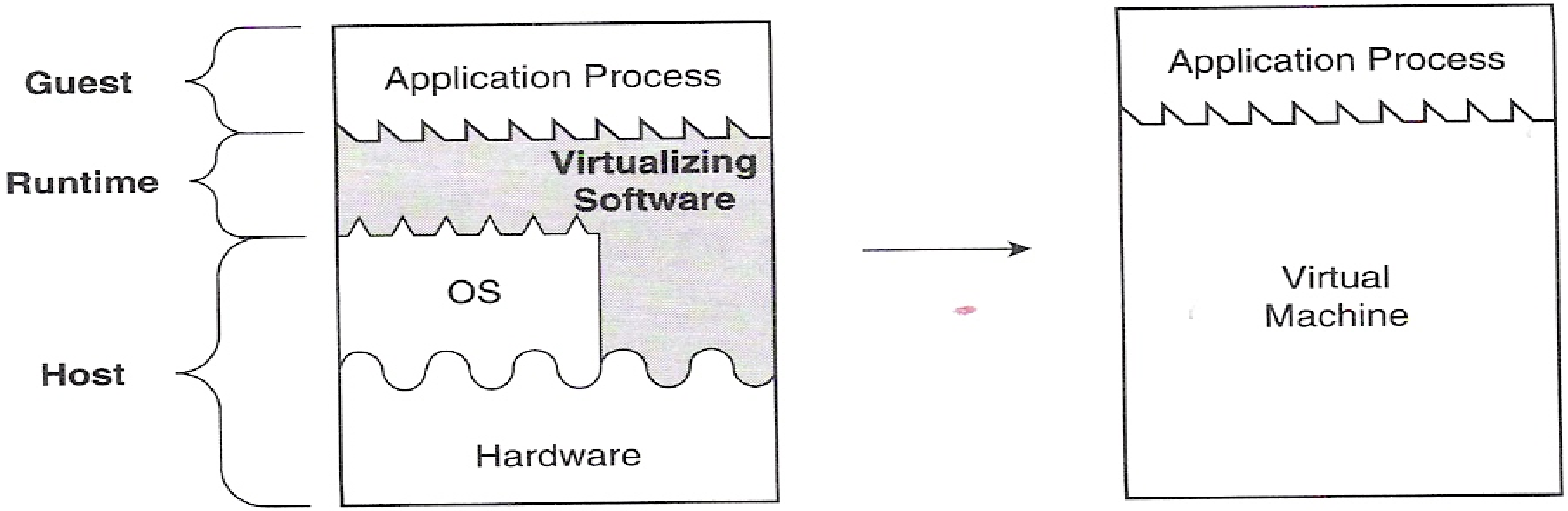


Figure 1.4 Computer System Architectures. *Implementation layers communicate vertically via the shown interfaces. This view of architecture is styled after one given by Glenford Myers (1982).*

(Courtesy of Smith and Nail, 2005)

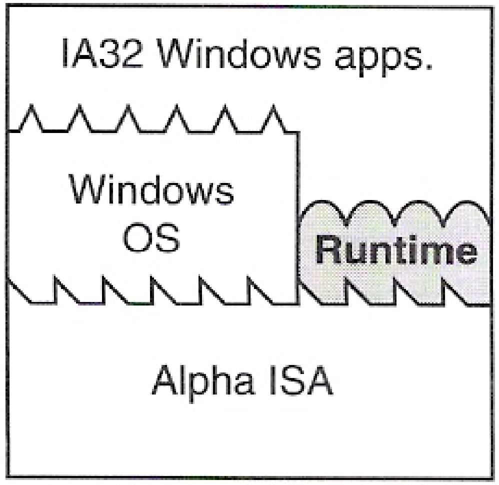


A Process Virtual Machine. *Virtualizing software translates a set of OS and user-level instructions composing one platform to another, forming a process virtual machine capable of executing programs developed for a different OS and a different ISA.*

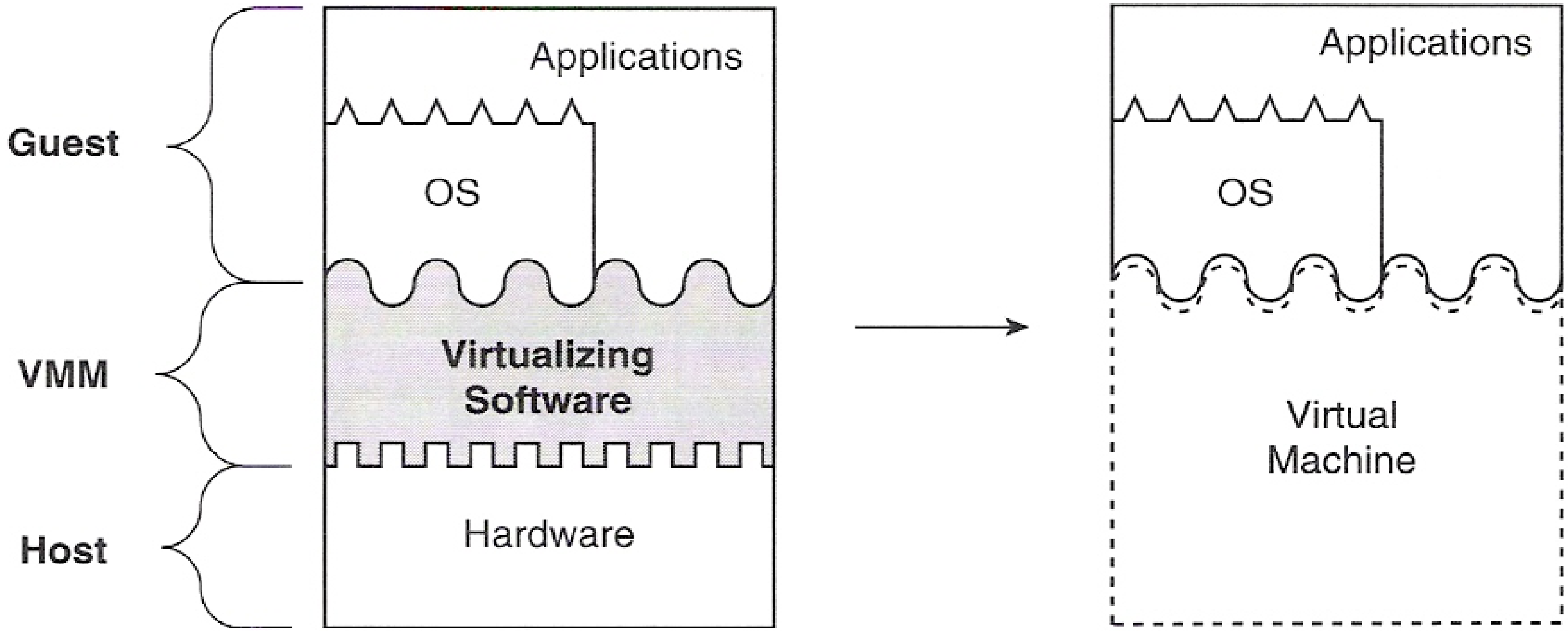
(Courtesy of Smith and Nail, 2005)

Process VM :

An example that emulates Guest IA32 Applications to run on an Alpha Windows platform



System Virtual Machine



A System Virtual Machine. *Virtualizing software translates the ISA used by one hardware platform to another, forming a system virtual machine, capable of executing a system software environment developed for a different set of hardware.*

(Courtesy of Smith and Nail, 2005)

Example of Virtual Machine Applications

Emulation **Replication** **Composition**

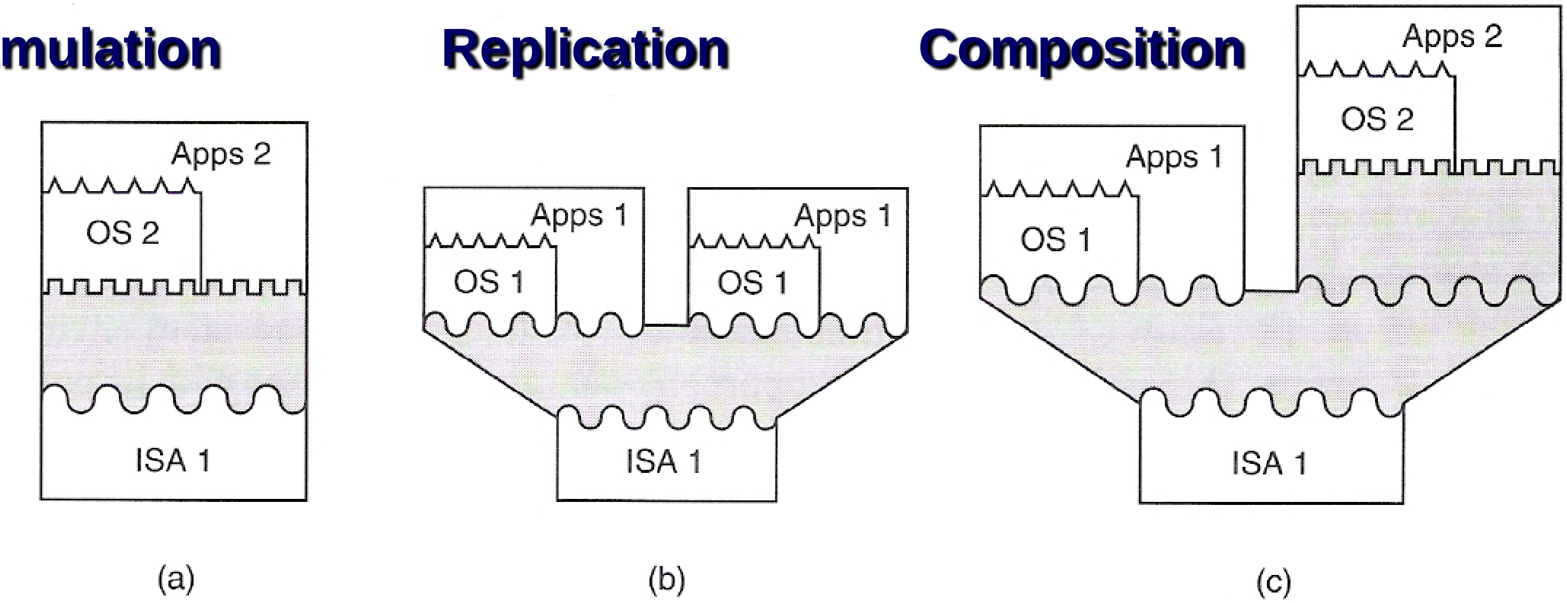


Figure 1.8 Examples of Virtual Machine Applications. *(a) Emulating one instruction set with another; (b) replicating a virtual machine so that multiple operating systems can be supported simultaneously; (c) composing virtual machine software to form a more complex, flexible system.*

(Courtesy of Smith and Nail, 2005)

Virtual Cluster Characteristics

- The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSs can be deployed on the same physical node.
- A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.
- The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance the server utilization and application flexibility.
- VMs can be colonized (replicated) in multiple servers for the purpose of promoting distributed parallelism, fault tolerance, and disaster recovery.
- The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similarly to the way an overlay network varies in size in a P2P network.
- The failure of any physical nodes may disable some VMs installed on the failing nodes. But the failure of VMs will not pull down the host system.

Virtual Clusters vs. Physical Clusters

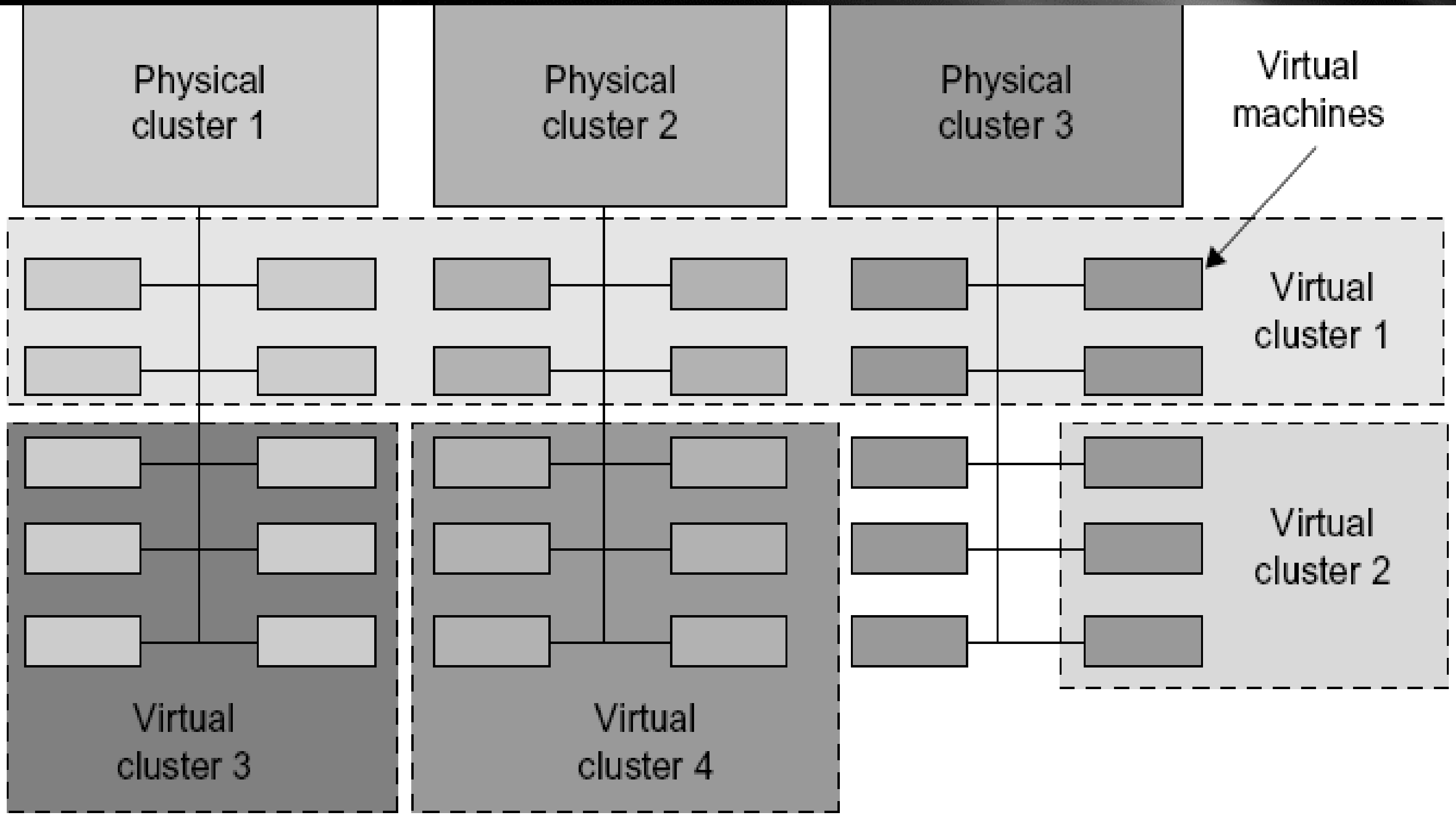


FIGURE 3.18

A cloud platform with 4 virtual clusters over 3 physical clusters shaded differently.

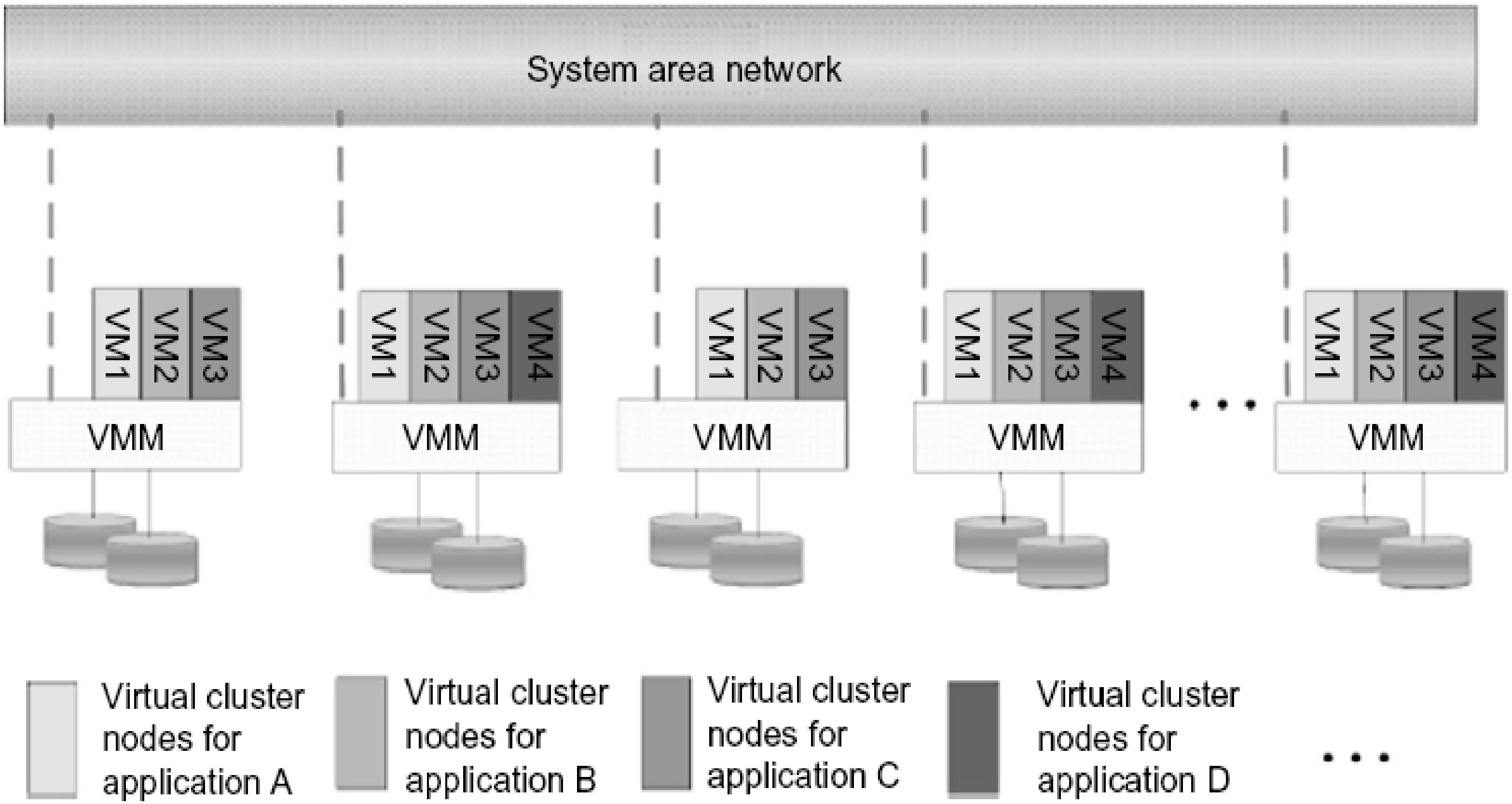

System area network
VM1
VM2
VM3
VM4
VMM
. . .
Virtual cluster nodes for application A
Virtual cluster nodes for application B
Virtual cluster nodes for application C
Virtual cluster nodes for application D
. . .

FIGURE 3.19

The concept of a virtual cluster based on application partitioning.

(Courtesy of Kang, Chen, Tsinghua University 2008)

Virtual Cluster Projects

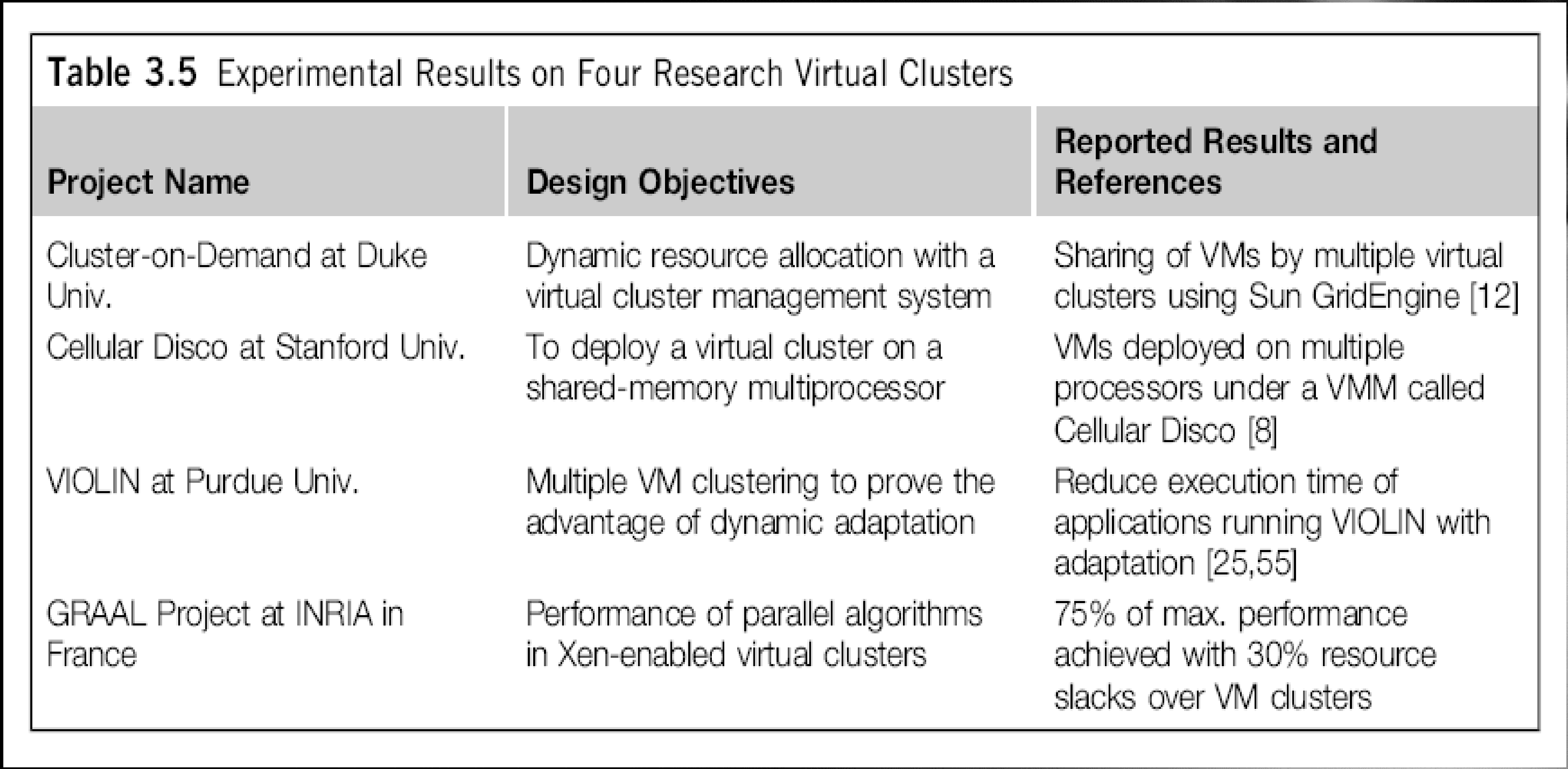


Table 3.5 Experimental Results on Four Research Virtual Clusters

Project Name	Design Objectives	Reported Results and References
Cluster-on-Demand at Duke Univ.	Dynamic resource allocation with a virtual cluster management system	Sharing of VMs by multiple virtual clusters using Sun GridEngine [12]
Cellular Disco at Stanford Univ.	To deploy a virtual cluster on a shared-memory multiprocessor	VMs deployed on multiple processors under a VMM called Cellular Disco [8]
VIOLIN at Purdue Univ.	Multiple VM clustering to prove the advantage of dynamic adaptation	Reduce execution time of applications running VIOLIN with adaptation [25,55]
GRAAL Project at INRIA in France	Performance of parallel algorithms in Xen-enabled virtual clusters	75% of max. performance achieved with 30% resource slacks over VM clusters

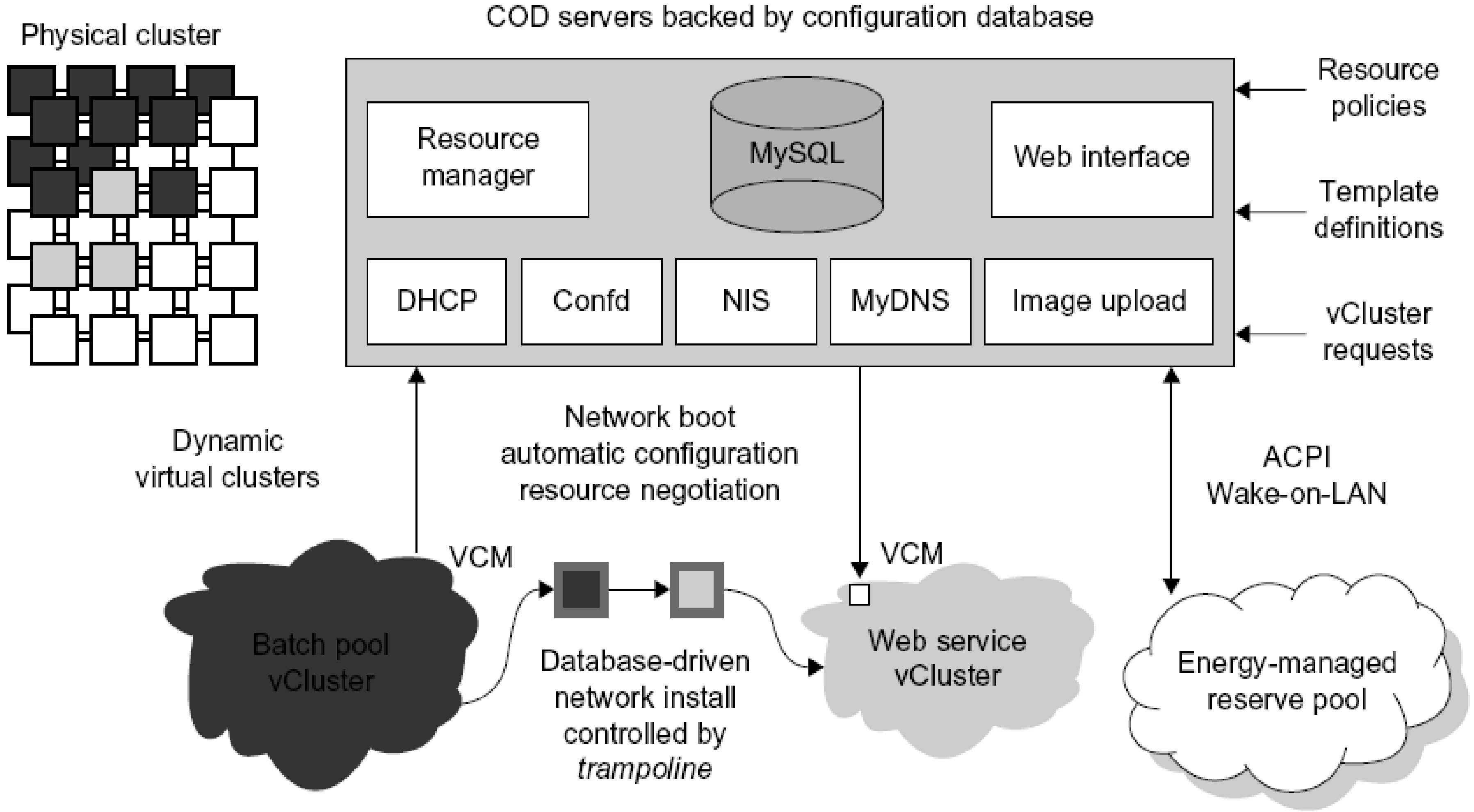
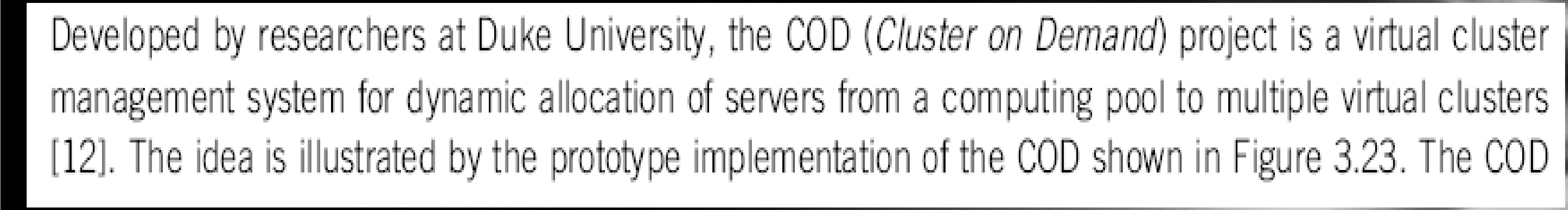


FIGURE 3.23

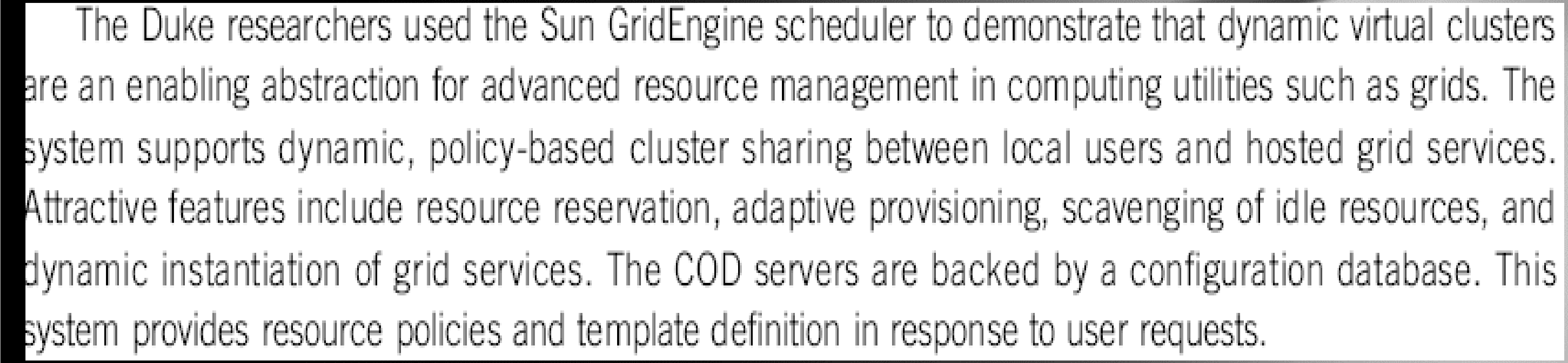
COD partitioning a physical cluster into multiple virtual clusters.

(*Courtesy of Jeff Chase, et al, HPDC-2003 [12]*)

Cluster-on-Demand (COD Project at Duke University



Developed by researchers at Duke University, the COD (*Cluster on Demand*) project is a virtual cluster management system for dynamic allocation of servers from a computing pool to multiple virtual clusters [12]. The idea is illustrated by the prototype implementation of the COD shown in Figure 3.23. The COD



The Duke researchers used the Sun GridEngine scheduler to demonstrate that dynamic virtual clusters are an enabling abstraction for advanced resource management in computing utilities such as grids. The system supports dynamic, policy-based cluster sharing between local users and hosted grid services. Attractive features include resource reservation, adaptive provisioning, scavenging of idle resources, and dynamic instantiation of grid services. The COD servers are backed by a configuration database. This system provides resource policies and template definition in response to user requests.

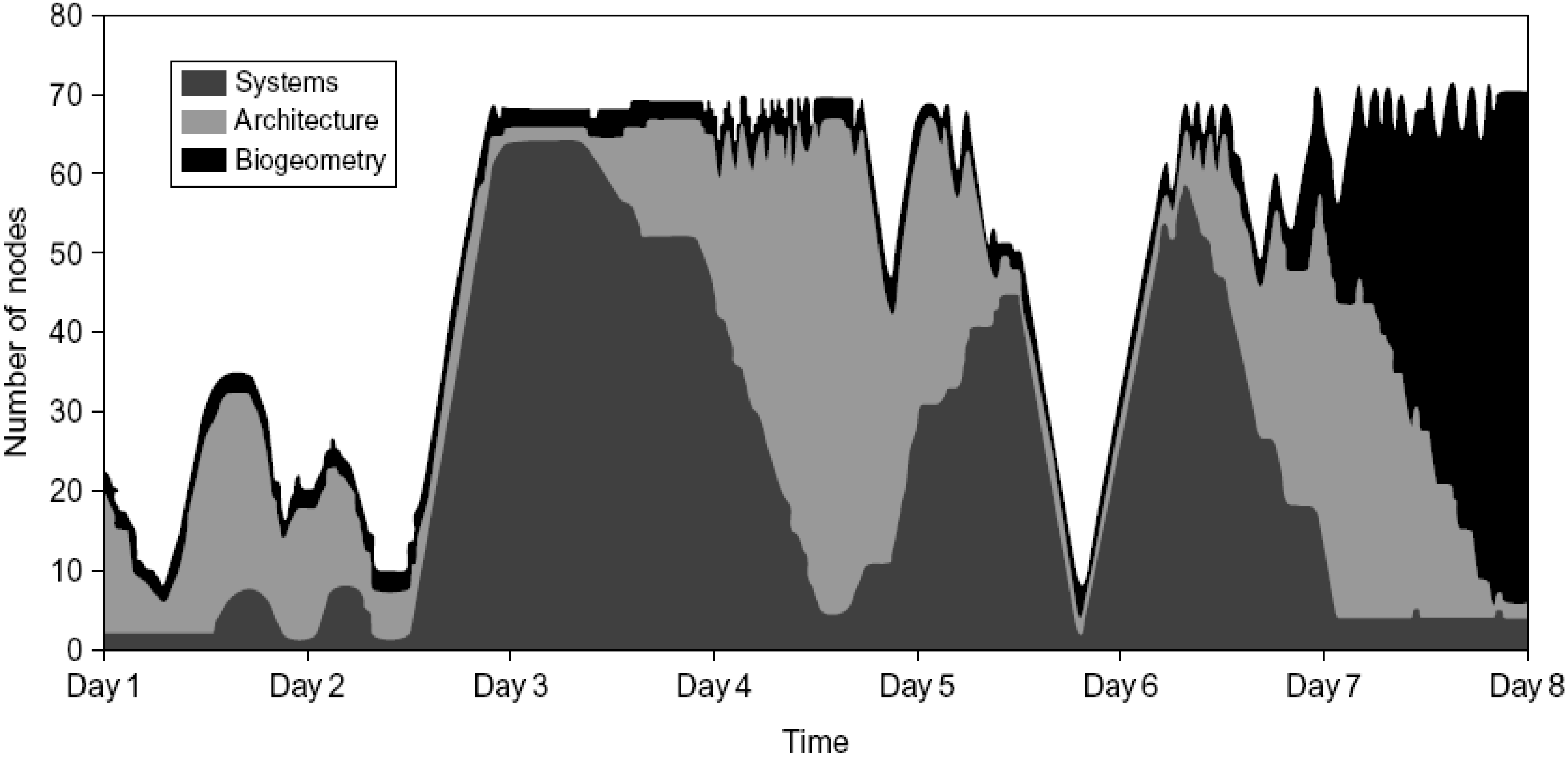
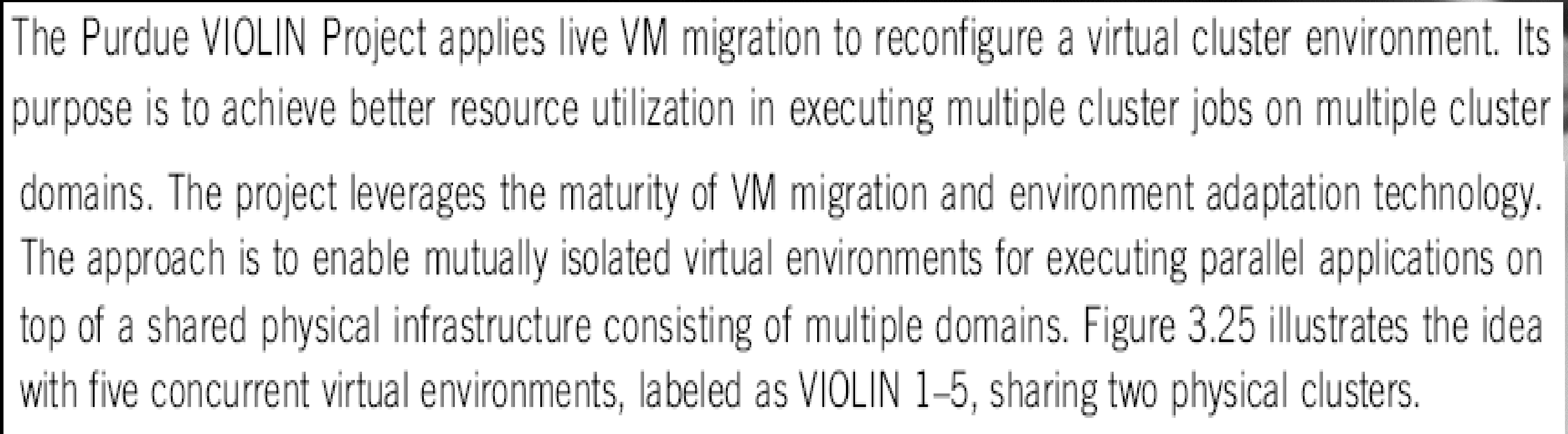


FIGURE 3.24

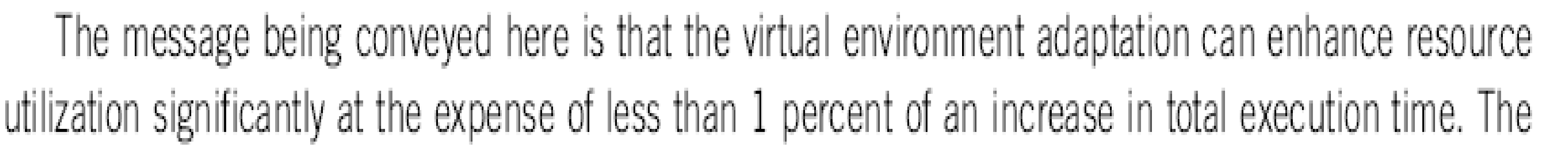
Cluster size variations in COD over eight days at Duke University.

(*Courtesy of J. Chase, et al. [12]*)

VIOLIN Project at Purdue University



The Purdue VIOLIN Project applies live VM migration to reconfigure a virtual cluster environment. Its purpose is to achieve better resource utilization in executing multiple cluster jobs on multiple cluster domains. The project leverages the maturity of VM migration and environment adaptation technology. The approach is to enable mutually isolated virtual environments for executing parallel applications on top of a shared physical infrastructure consisting of multiple domains. Figure 3.25 illustrates the idea with five concurrent virtual environments, labeled as VIOLIN 1–5, sharing two physical clusters.



The message being conveyed here is that the virtual environment adaptation can enhance resource utilization significantly at the expense of less than 1 percent of an increase in total execution time. The

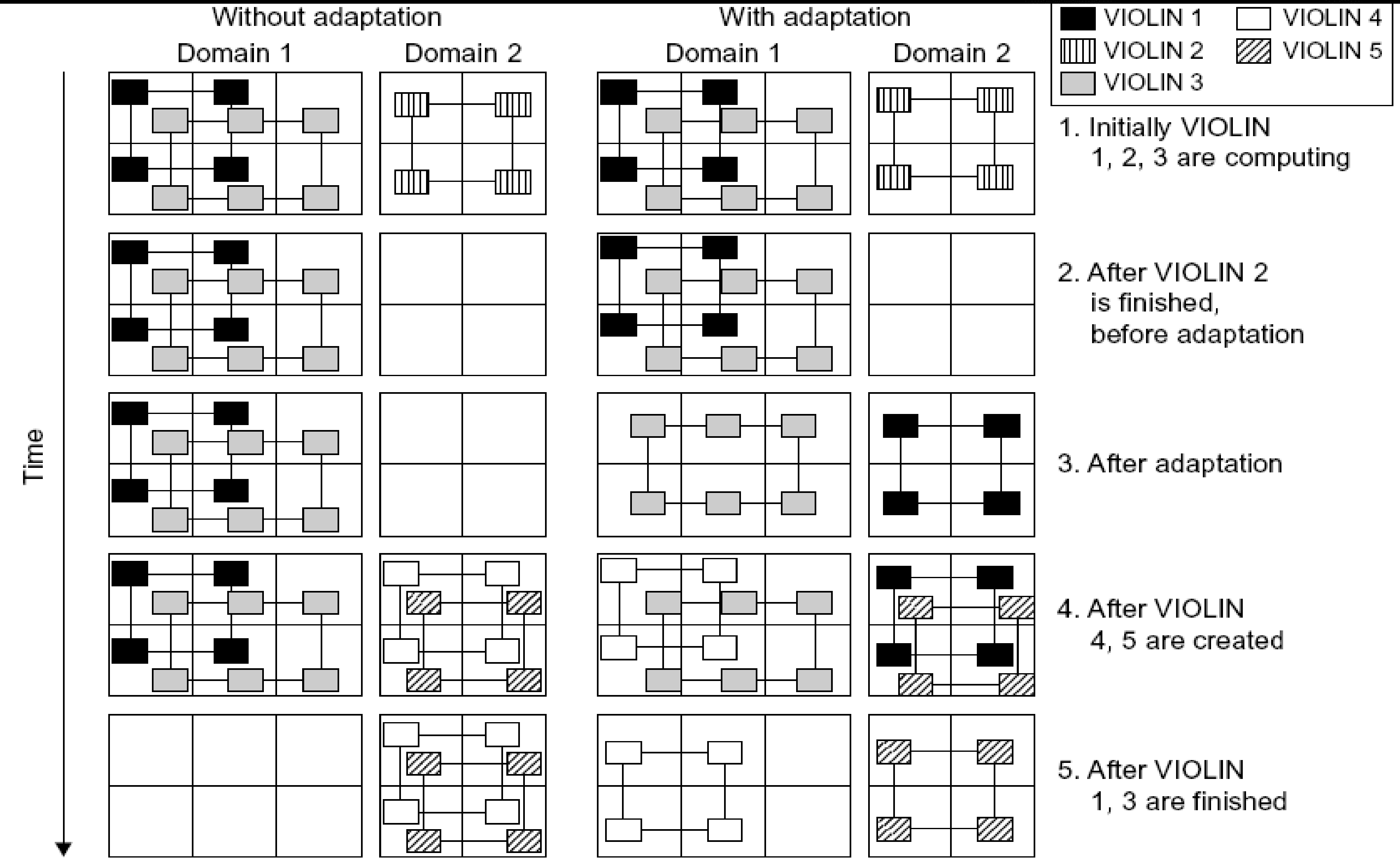
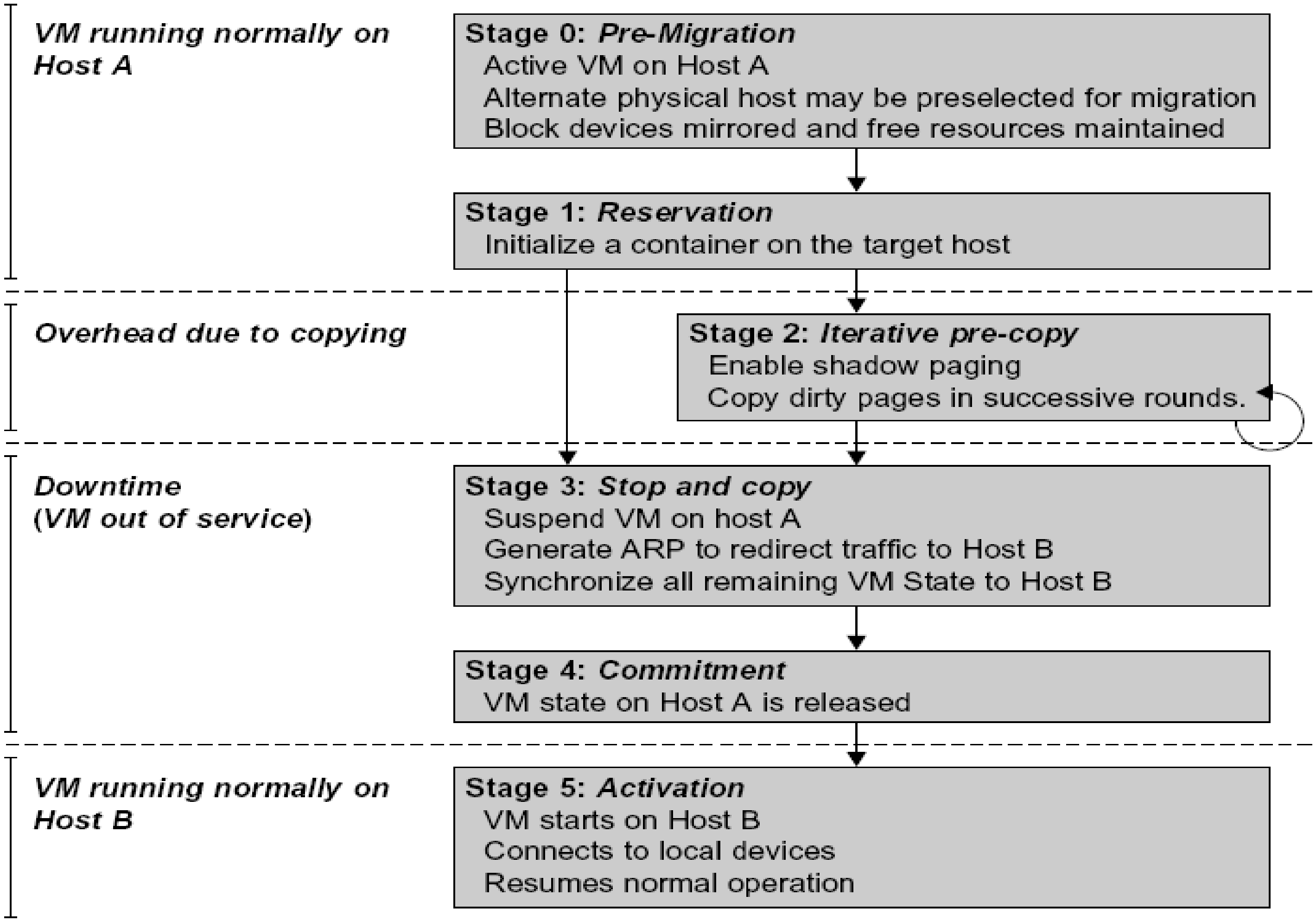

Without adaptation
With adaptation
Domain 1
Domain 2
Domain 1
Domain 2
■ VIOLIN 1
▥ VIOLIN 2
▨ VIOLIN 3
□ VIOLIN 4
▧ VIOLIN 5
Time
1. Initially VIOLIN 1, 2, 3 are computing
2. After VIOLIN 2 is finished, before adaptation
3. After adaptation
4. After VIOLIN 4, 5 are created
5. After VIOLIN 1, 3 are finished

FIGURE 3.25

VIOLIN adaptation scenario of five virtual environments sharing two hosted clusters; Note that there are more idle squares (blank nodes) before and after the adaptation.

(*Courtesy of P. Ruth, et al.* [24,51])

Live Migration of Virtual Machines



VM running normally on Host A

Stage 0: ***Pre-Migration***
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained

Stage 1: ***Reservation***
Initialize a container on the target host

Overhead due to copying

Stage 2: ***Iterative pre-copy***
Enable shadow paging
Copy dirty pages in successive rounds.

Downtime
(VM out of service)

Stage 3: ***Stop and copy***
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM State to Host B

Stage 4: ***Commitment***
VM state on Host A is released

VM running normally on Host B

Stage 5: ***Activation***
VM starts on Host B
Connects to local devices
Resumes normal operation

FIGURE 3.20

Live migration process of a VM from one host to another.

(*Courtesy of C. Clark, et al. [14]*)

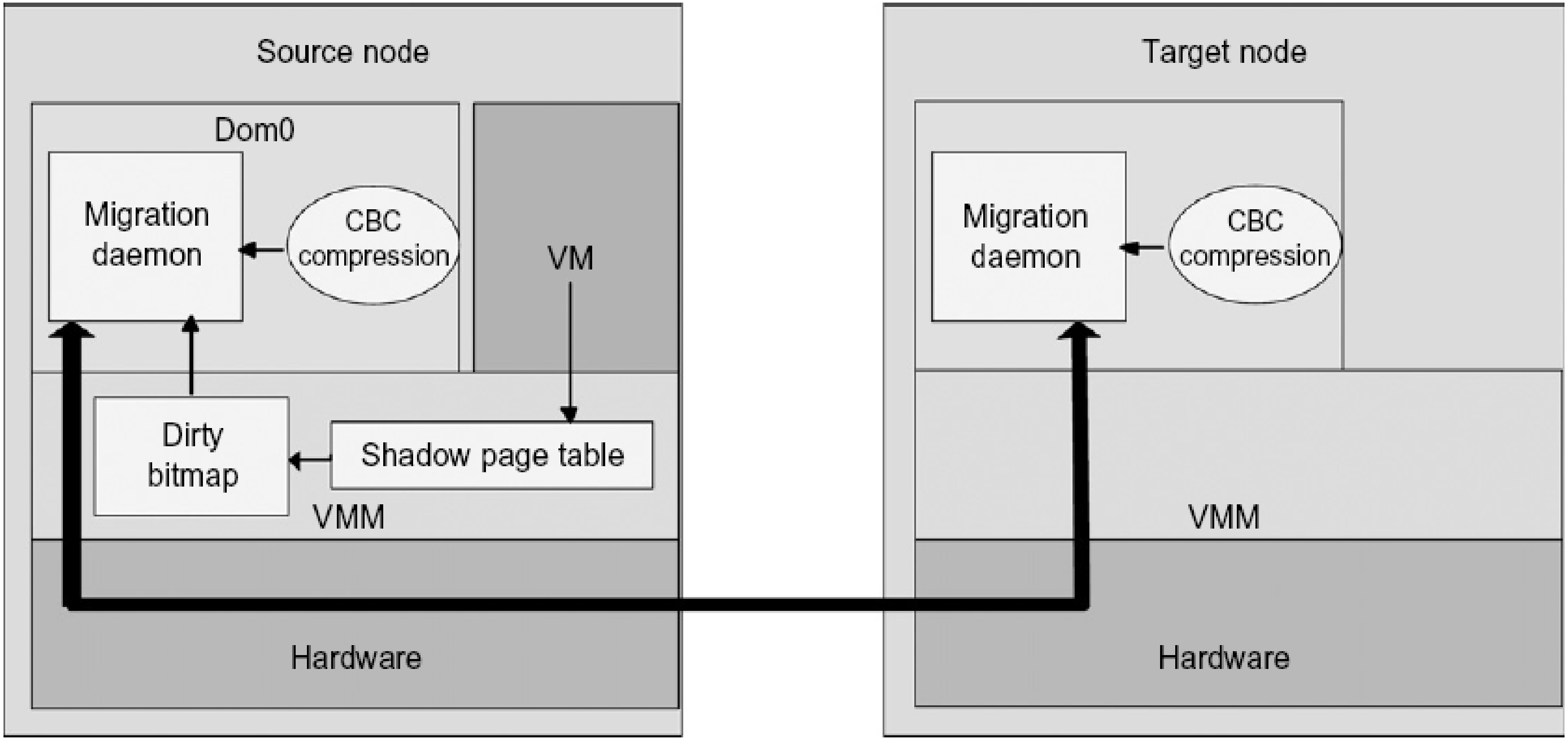


FIGURE 3.22

Live migration of VM from the Dom0 domain to a Xen-enabled target host.

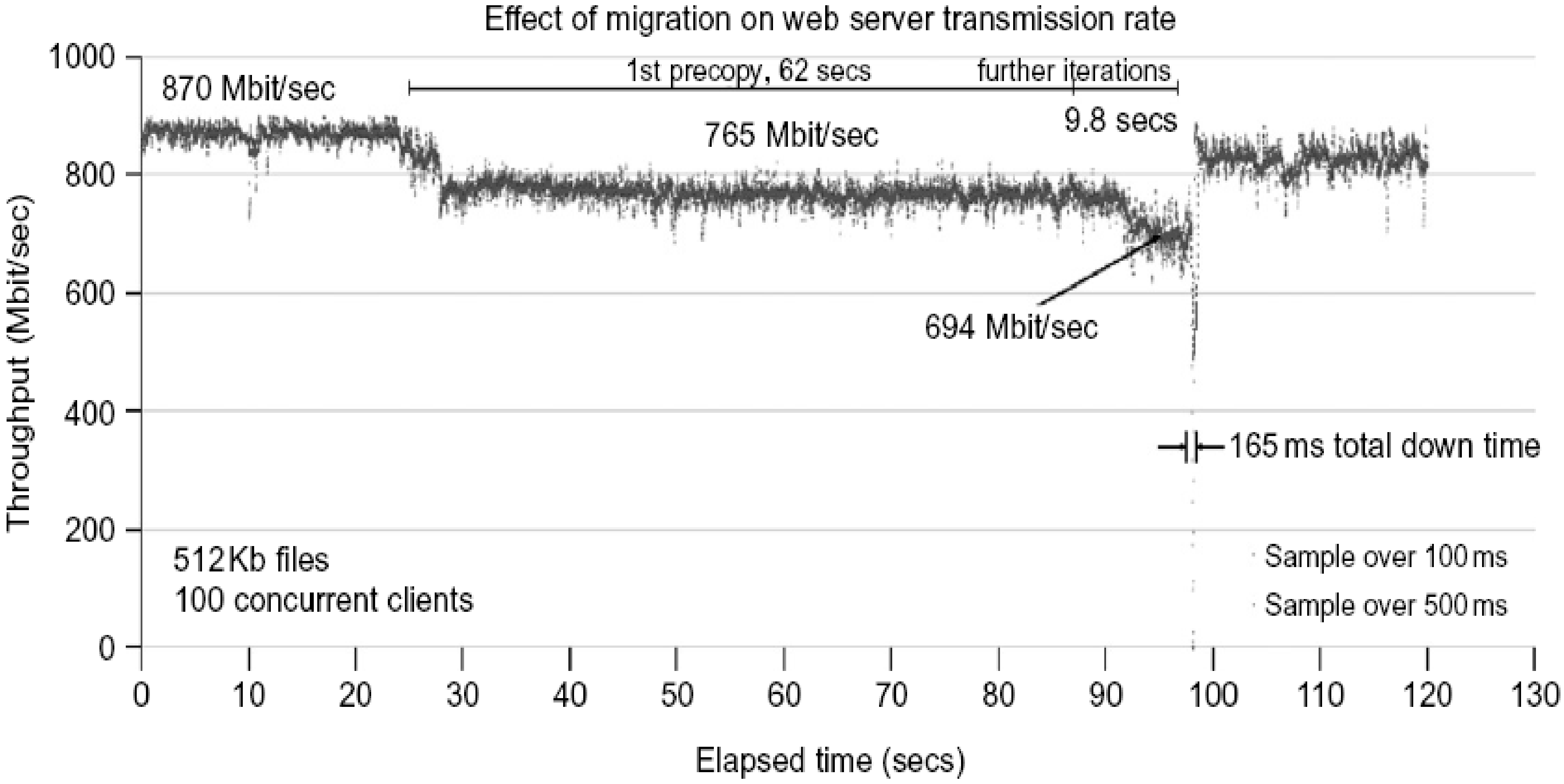


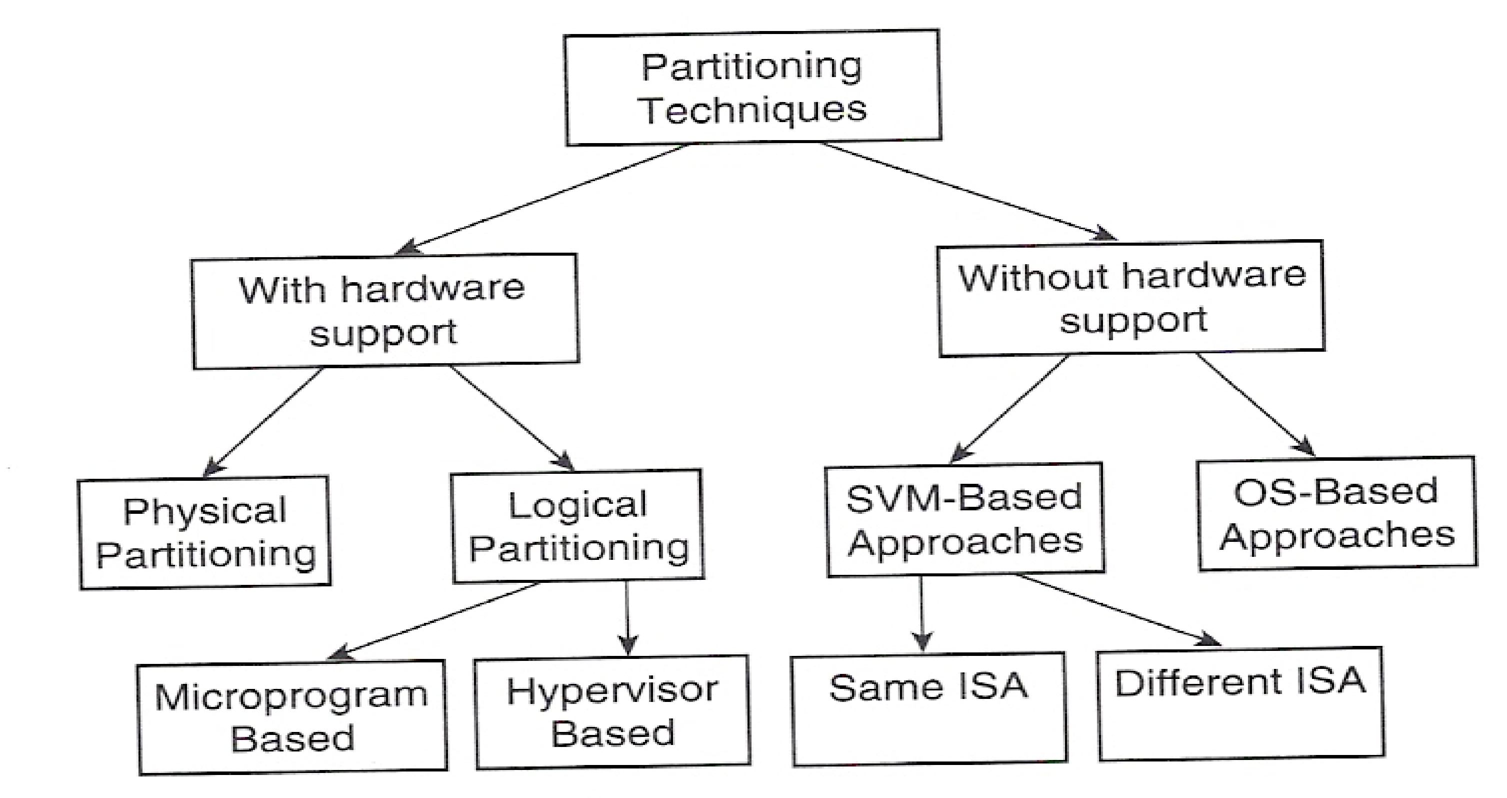
FIGURE 3.21

Effect on data transmission rate of a VM migrated from one failing web server to another.

(*Courtesy of C. Clark, et al. [14]*)

Cluster Partitioning

for VM-based Parallel Systems



(Courtesy of Smith and Nail, 2005)

System VM-based Partitioning

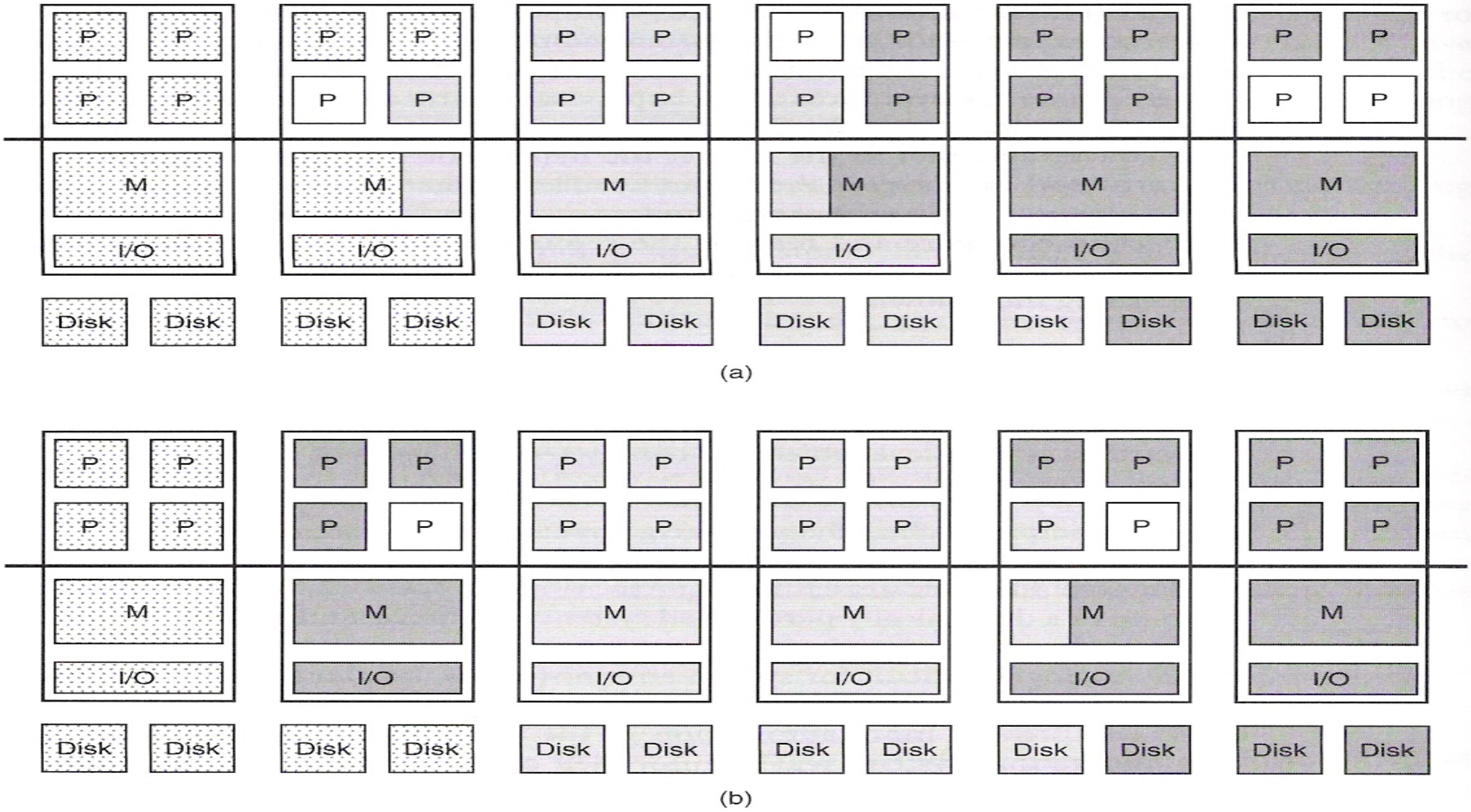


Figure 9.11 Illustration of the Effects of Dynamic Partitioning. *The figure shows the use of resources by three partitions at two different points in time. In (a) the usage of resources, for example, just after the system is initialized, it displays some degree of physical locality. However, as time progresses (b) and as the processes get migrated to achieve better load balancing, the physical range of the partitions running these processes start merging with each other.*

(Courtesy of Smith and Nail, 2005)

A Virtual Cluster on a Real Cluster

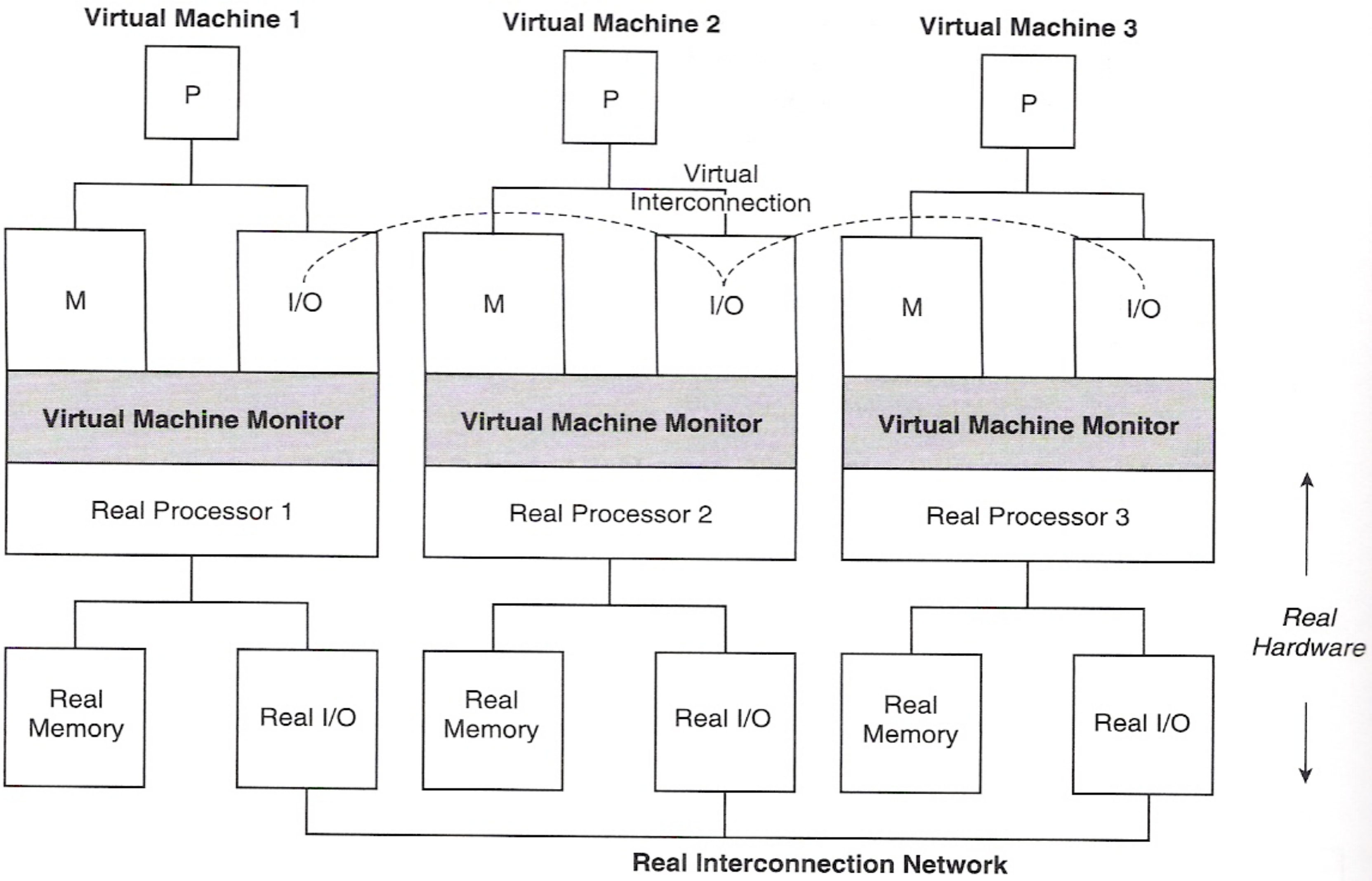


Figure 9.16 Virtual Uniprocessor Cluster on a Real Uniprocessor Cluster.

(Courtesy of Smith and Nail, 2005)

Virtualization with Different Host and Guest ISAs

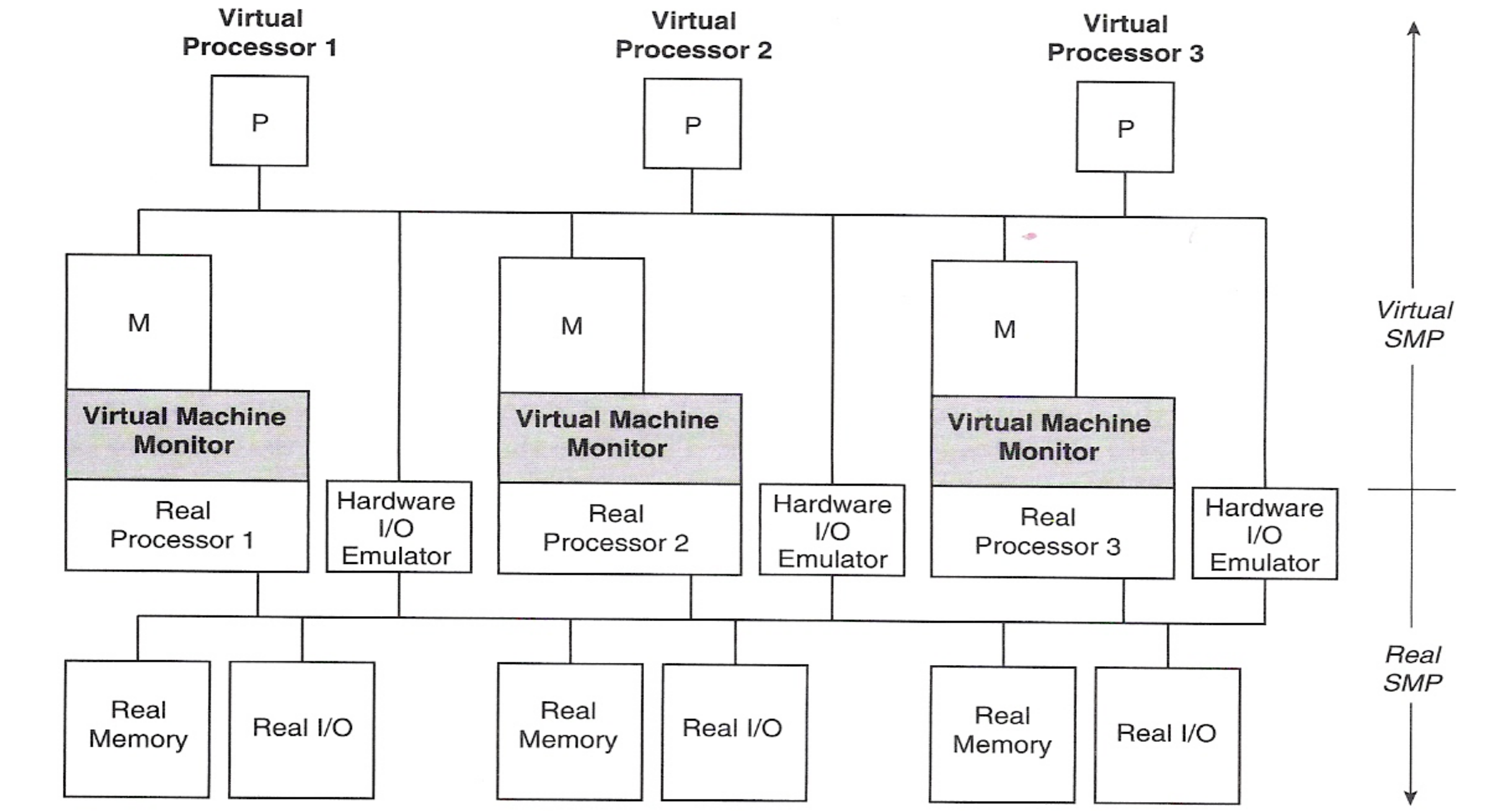
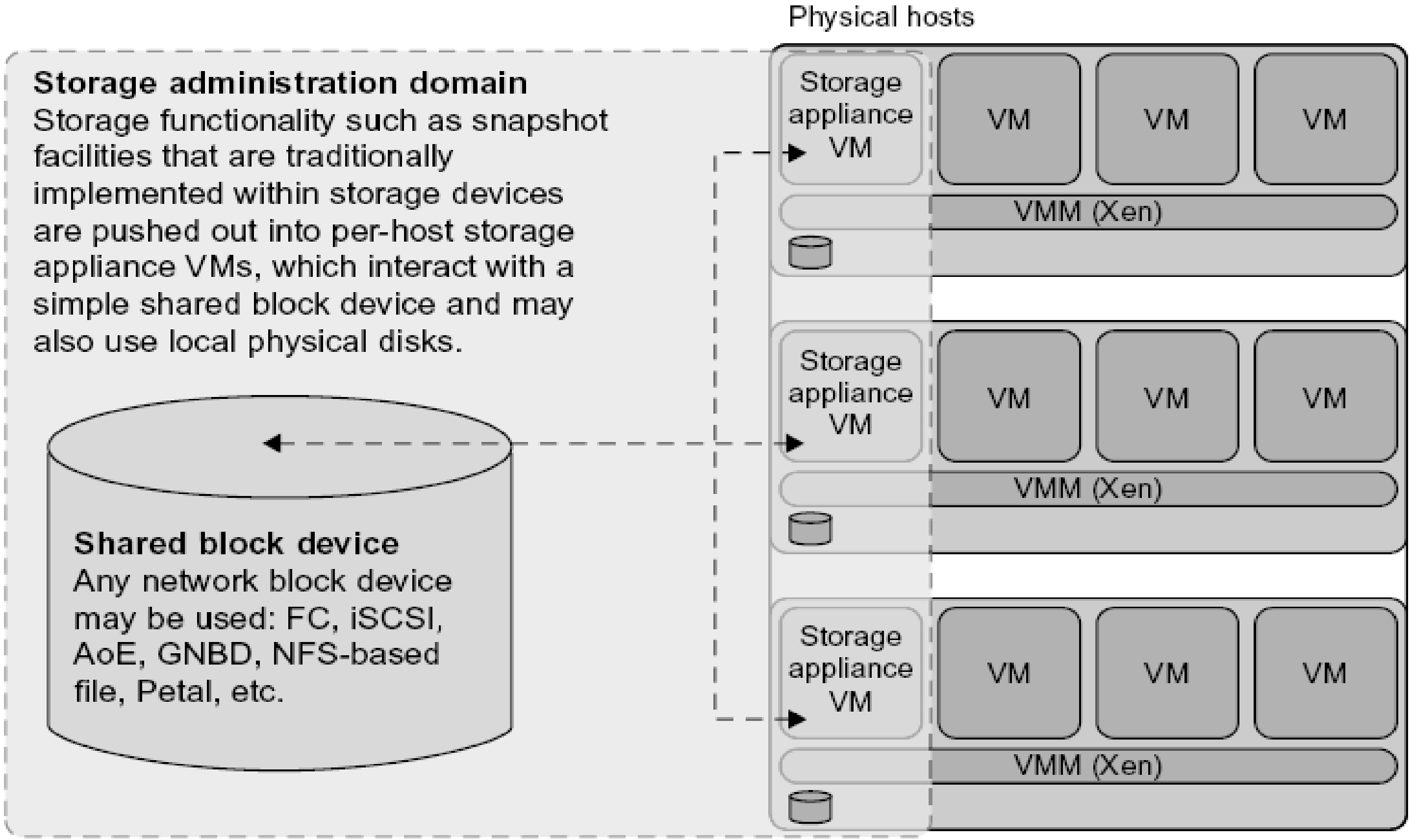


Figure 9.17 Virtual Shared-Memory System on a Real Shared-Memory System. *When there is a mismatch between the I/O of the virtual system and the real system, it may be necessary to add hardware, as shown, for efficient I/O emulation.*

(Courtesy of Smith and Nail, 2005)

Parallax for VM Storage Management



Storage administration domain
Storage functionality such as snapshot facilities that are traditionally implemented within storage devices are pushed out into per-host storage appliance VMs, which interact with a simple shared block device and may also use local physical disks.

Shared block device
Any network block device may be used: FC, iSCSI, AoE, GNBD, NFS-based file, Petal, etc.

FIGURE 3.26

Parallax is a set of per-host storage appliances that share access to a common block device and presents virtual disks to client VMs.

(*Courtesy of D. Meyer, et al. [43]*)

Cloud OS for Building Private Clouds

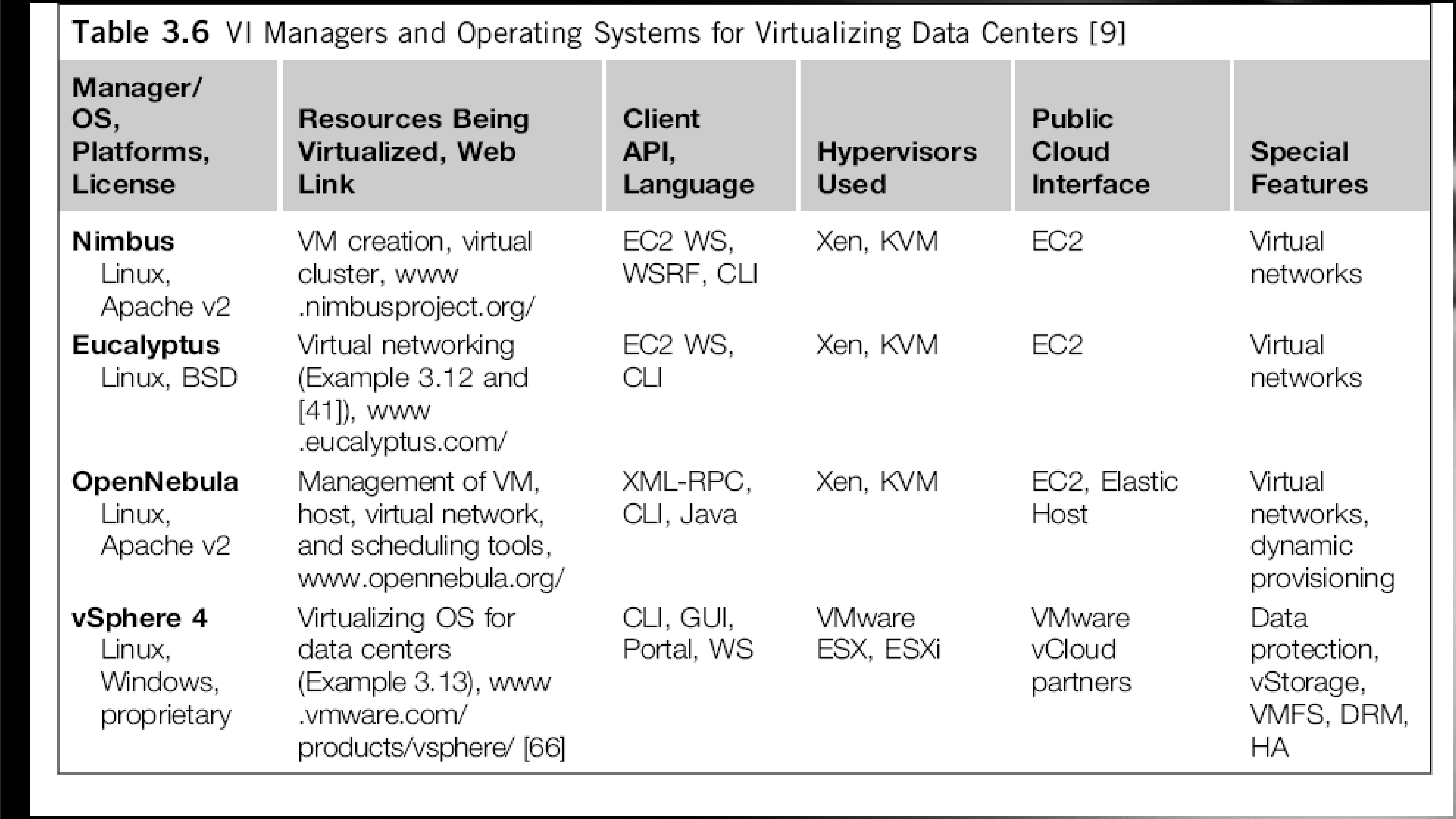
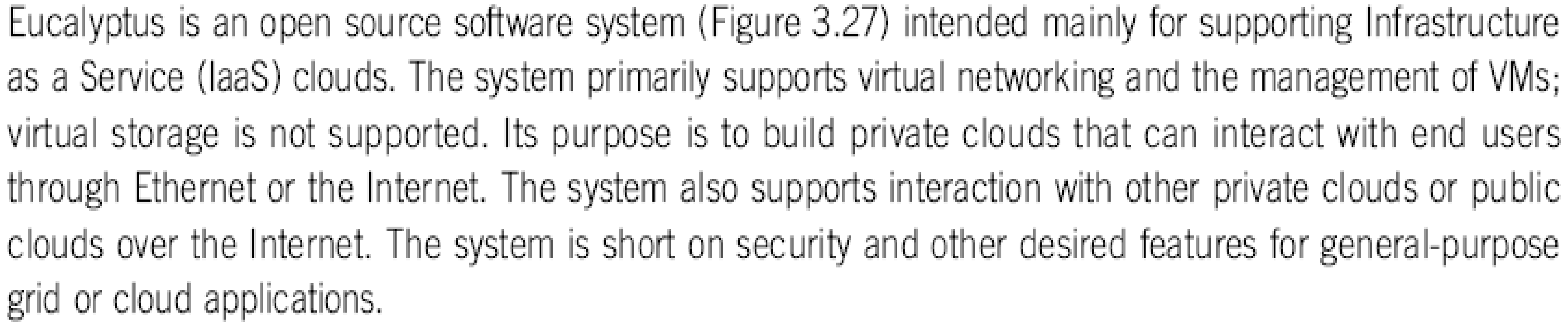


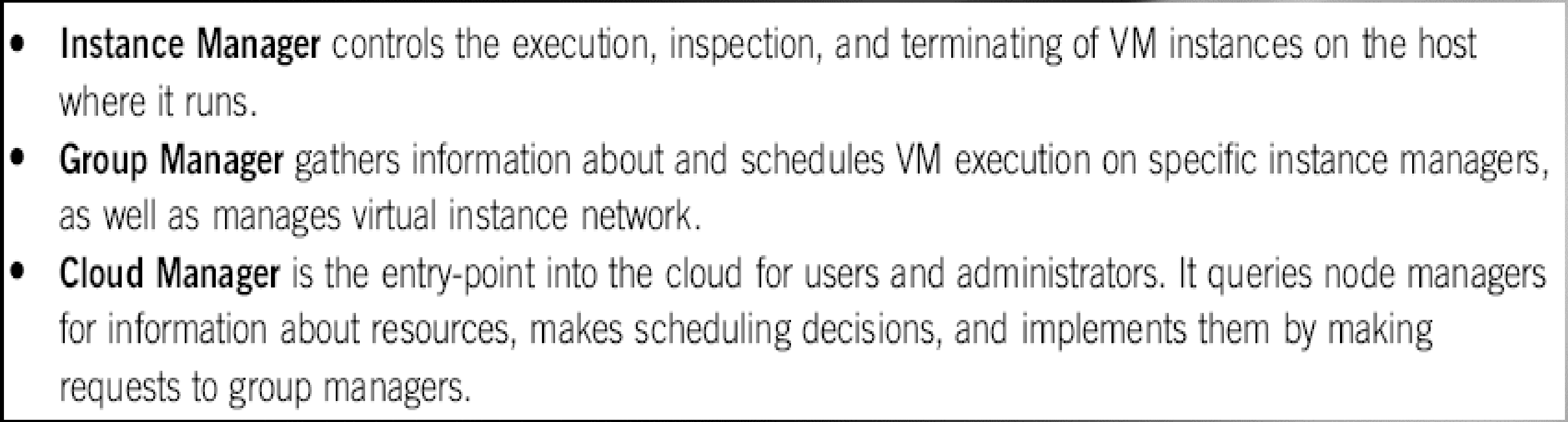
Table 3.6 VI Managers and Operating Systems for Virtualizing Data Centers [9]

Manager/ OS, Platforms, License	Resources Being Virtualized, Web Link	Client API, Language	Hypervisors Used	Public Cloud Interface	Special Features
Nimbus Linux, Apache v2	VM creation, virtual cluster, www.nimbusproject.org/	EC2 WS, WSRF, CLI	Xen, KVM	EC2	Virtual networks
Eucalyptus Linux, BSD	Virtual networking (Example 3.12 and [41]), www.eucalyptus.com/	EC2 WS, CLI	Xen, KVM	EC2	Virtual networks
OpenNebula Linux, Apache v2	Management of VM, host, virtual network, and scheduling tools, www.opennebula.org/	XML-RPC, CLI, Java	Xen, KVM	EC2, Elastic Host	Virtual networks, dynamic provisioning
vSphere 4 Linux, Windows, proprietary	Virtualizing OS for data centers (Example 3.13), www.vmware.com/products/vsphere/ [66]	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vCloud partners	Data protection, vStorage, VMFS, DRM, HA

Eucalyptus : An Open-Source OS for Setting Up and Managing Private Clouds



Eucalyptus is an open source software system (Figure 3.27) intended mainly for supporting Infrastructure as a Service (IaaS) clouds. The system primarily supports virtual networking and the management of VMs; virtual storage is not supported. Its purpose is to build private clouds that can interact with end users through Ethernet or the Internet. The system also supports interaction with other private clouds or public clouds over the Internet. The system is short on security and other desired features for general-purpose grid or cloud applications.



- **Instance Manager** controls the execution, inspection, and terminating of VM instances on the host where it runs.
- **Group Manager** gathers information about and schedules VM execution on specific instance managers, as well as manages virtual instance network.
- **Cloud Manager** is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes scheduling decisions, and implements them by making requests to group managers.

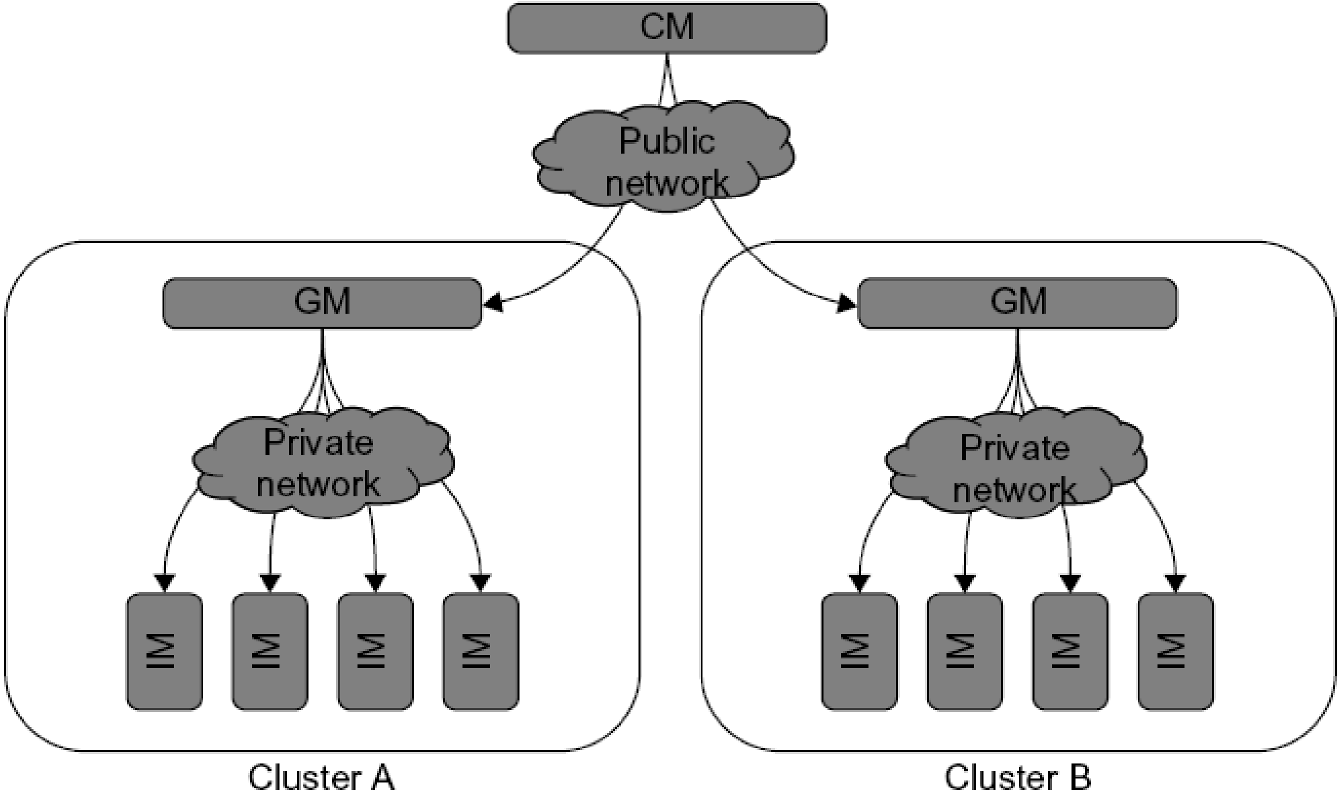


FIGURE 3.27

Eucalyptus for building private clouds by establishing virtual networks over the VMs linking through Ethernet and the Internet.

(Courtesy of D. Nurmi, et al. [45])

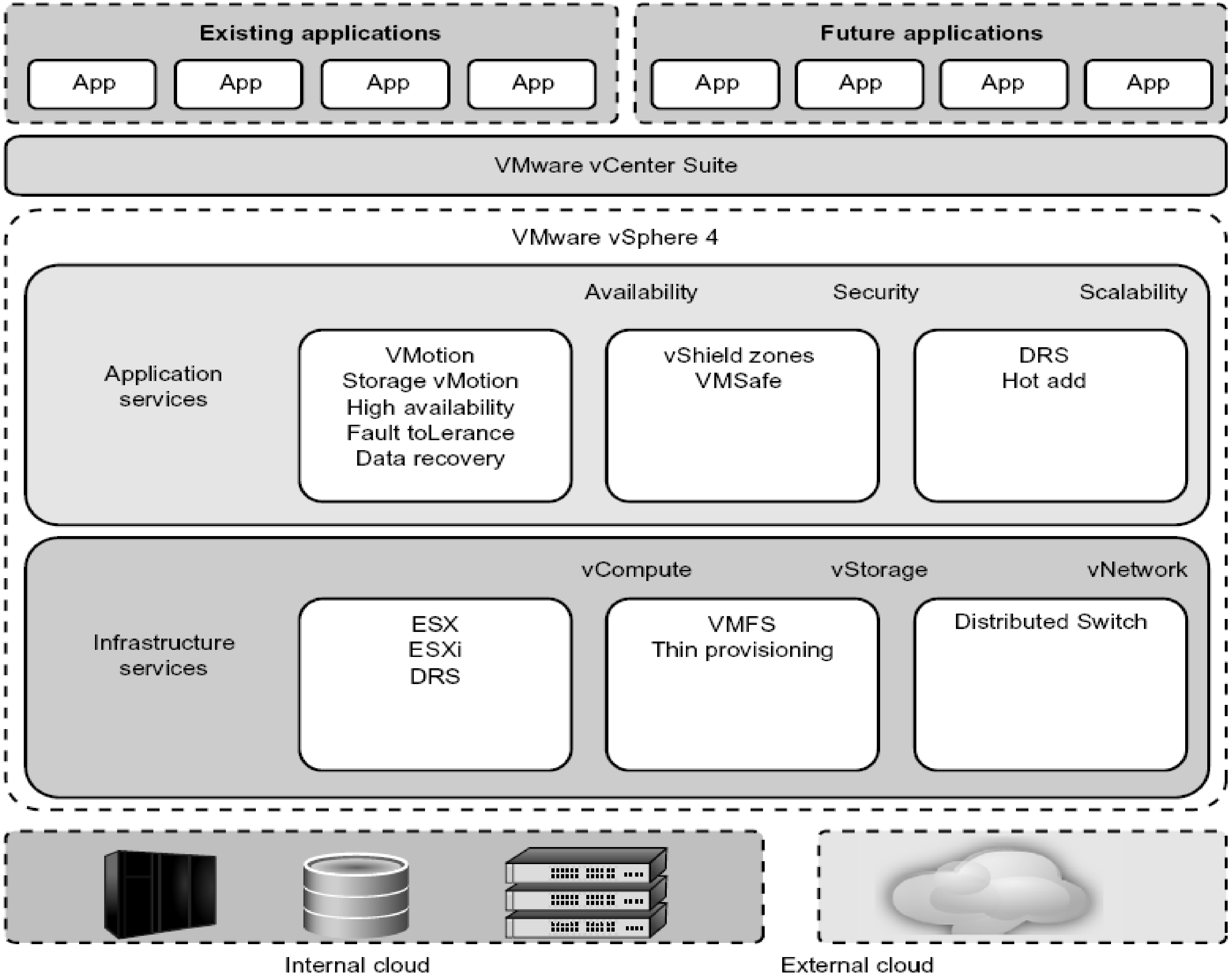
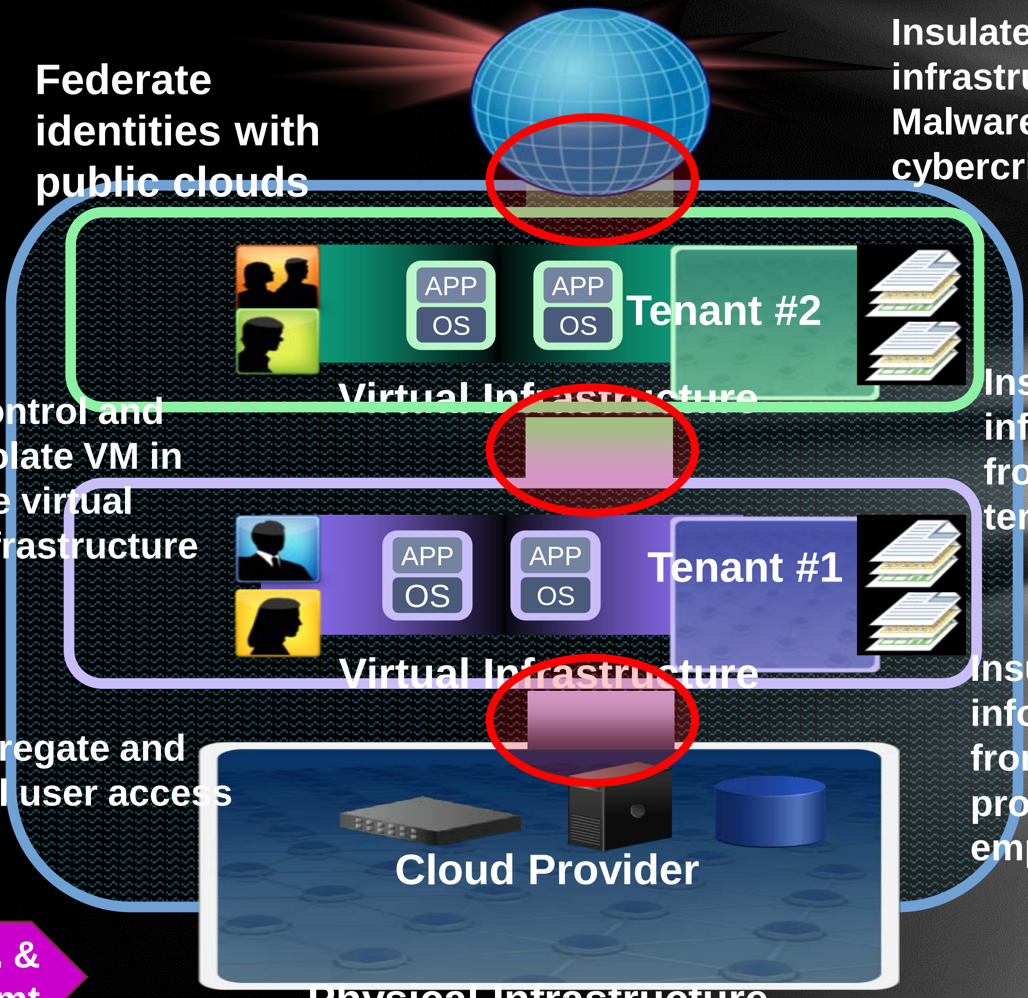


FIGURE 3.28

vSphere/4, a cloud operating system that manages compute, storage, and network resources over virtualized data centers.

(Courtesy of VMware, April 2010 [72])

Trusted Zones for VM Insulation



Identity federation

Federate identities with public clouds

Virtual network security

Control and isolate VM in the virtual infrastructure

Access Mgmt

Segregate and control user access

Security Info. & Event Mgmt

Enable end to end view of security events and compliance across infrastructures

Insulate infrastructure from Malware, Trojans a cybercriminals

Anti-malware

Cybercrime intelligence

Strong authentication

Insulate information from other tenants

Data loss prevention

Insulate information from cloud providers' employees

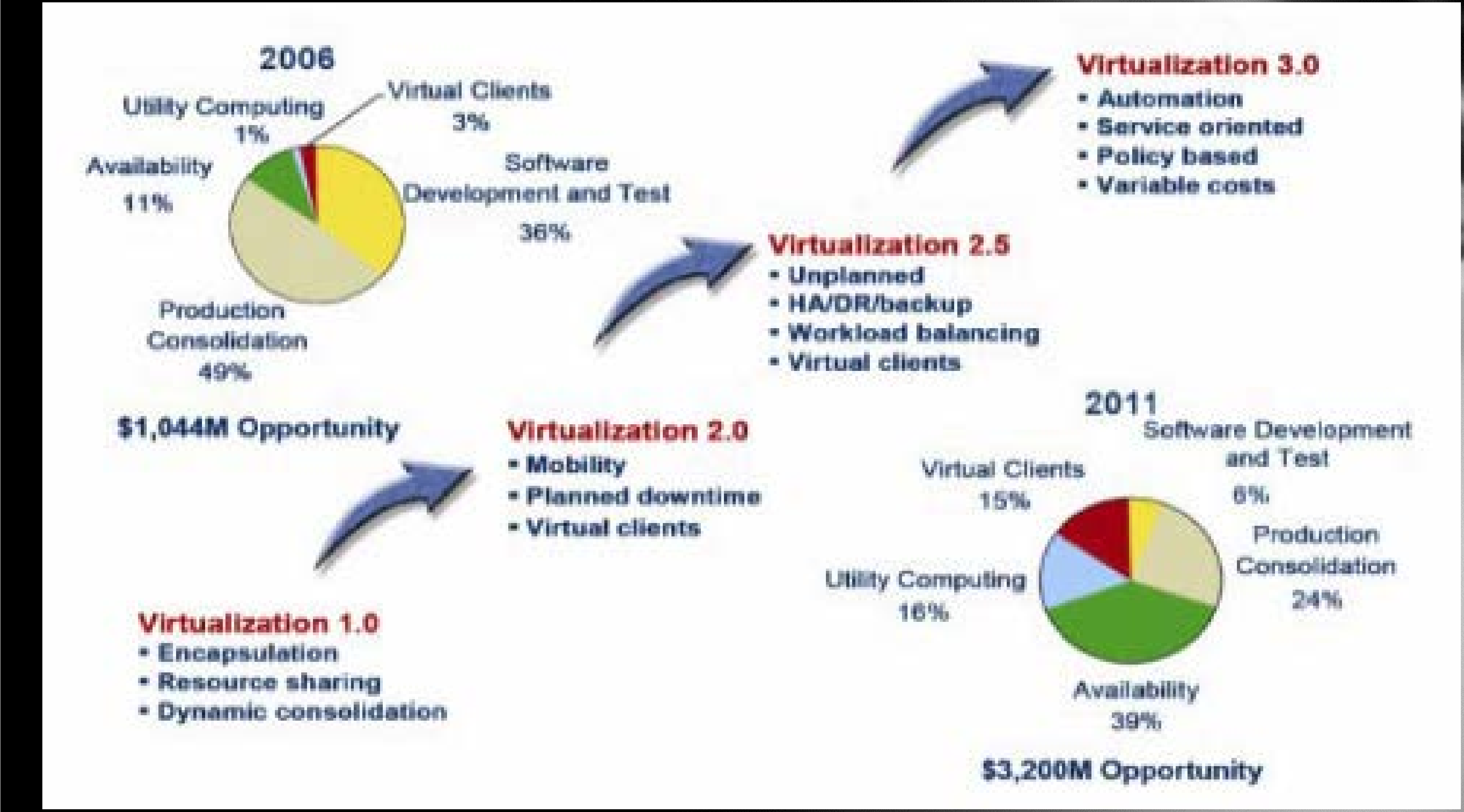
Encryption & key mgmt

Tokenization

GRC

(Courtesy of L. Nick, EMC 2008)

Projected Growth of Virtualization Market from 2006 to 2011 by IDC



Virtualization 1.0
- Encapsulation
- Resource sharing
- Dynamic consolidation

Virtualization 2.0
- Mobility
- Planned downtime
- Virtual clients

Virtualization 2.5
- Unplanned
- HA/DR/backup
- Workload balancing
- Virtual clients

Virtualization 3.0
- Automation
- Service oriented
- Policy based
- Variable costs

Reading Assignments :

1. K. Hwang, G. Fox and J. Dongarra, *Distributed Systems and Cloud Computing,* Chapter 3, 2011

2. M. Rosenblum and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends", *IEEE Computer Magazine,* May 2005, pp.39-47.

3. VM Ware, Inc., "Virtualization Overview ", White paper, http://www.vmware.com , 2006.

4. Virtual Machines by James Smith and Ravi Nair, Morgan Kaufmann, an Elesevier imprint, 2005