Unit-I(2nd Part): Virtual Machines and Virtualization of Clusters and datacenters

From book: Distributed and Cloud Computing

K. Hwang, G. Fox and J. Dongarra

Virtualization for Datacenter Automation to serve millions of clients, simultaneously

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

Implementation Levels of Virtualization

- Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.
- The idea of VMs can be dated back to the 1960s.
- The purpose of a VM is to enhance resource ***sharing by many users and improve computer performance in terms of resource utilization and application flexibility.***
- Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.
- This ***virtualization technology has been revitalized as the demand for distributed and cloud computing increased sharply in recent years.***

Levels of Virtualization Implementation

- A traditional computer runs with a ***host operating system specially*** tailored for its hardware architecture.
- After virtualization, different user applications managed by their own operating systems ***(guest OS) can run on the same hardware, independent of the host OS***.
- This is often done by adding additional software, called a ***virtualization layer.***
- This virtualization layer is known as a **hypervisor or virtual machine monitor (VMM).**

Levels of Virtualization Implementation

- The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.
- The main function of the software layer for **virtualization** ***is to virtualize the physical hardware*** of a host machine into virtual resources to be used by the VMs, exclusively.
- This can be implemented at various operational levels. The virtualization software creates the abstraction of **VMs by interposing a virtualization layer at various levels of a computer system**.
- ***Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level).***

Instruction set

An **instruction set**, or **instruction set architecture** (ISA), is the part of the computer **architecture** related to programming, including the native data types,**instructions**, registers, addressing modes, memory **architecture**, interrupt and exception handling, and external I/O.

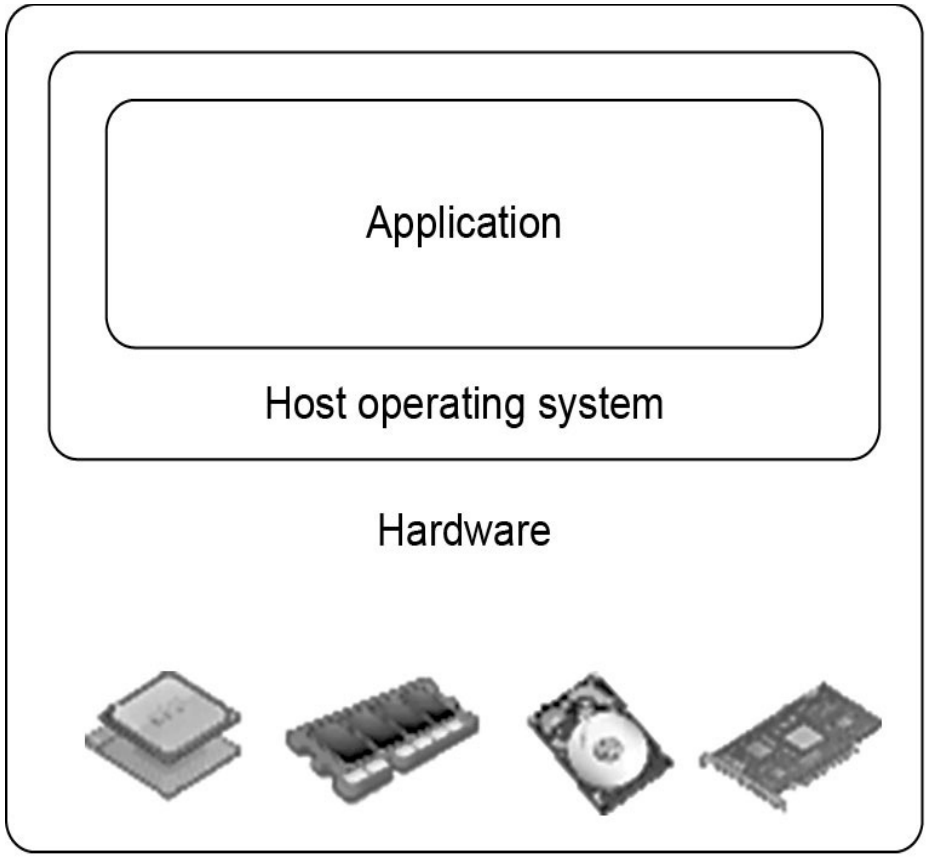
X86 processor : **x86** is a family of backward compatible instruction set architectures [a] based on the Intel 8086 **CPU** and its Intel 8088 variant.

Want's to know more about the X86 processor-

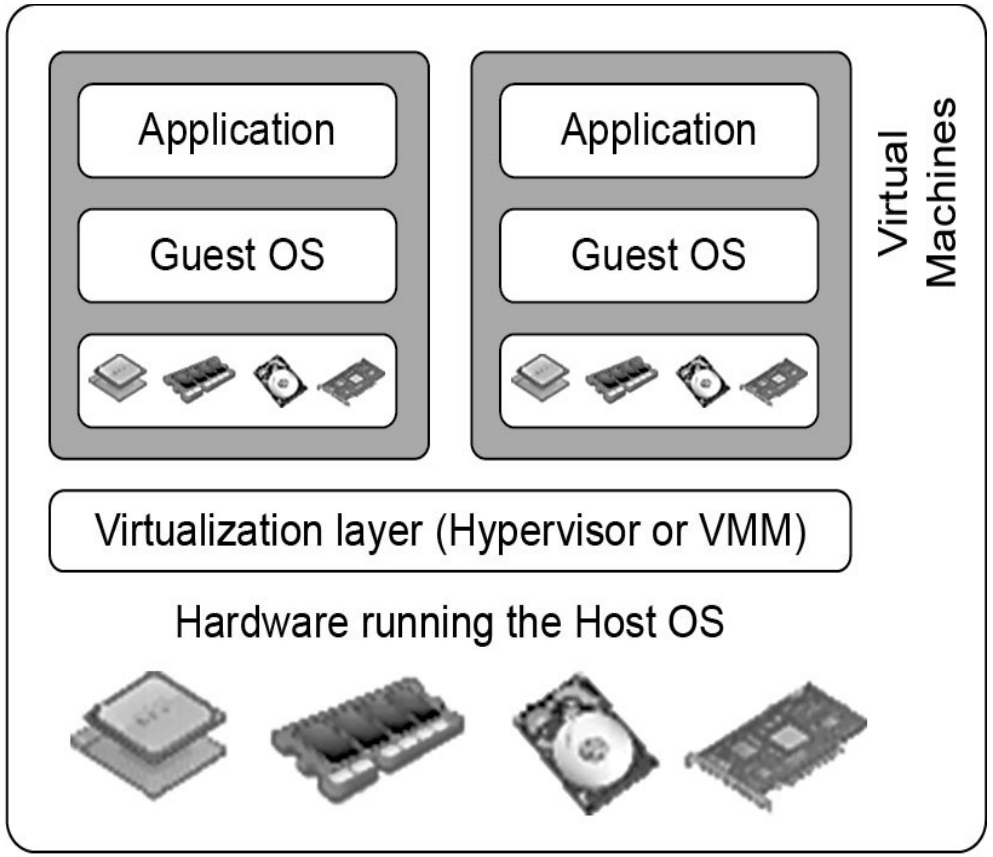
https://www.cis.upenn.edu/~cdmurphy/cit593/fall2012/pdf/lecture20-x86.pdf

> In industry terms, where major paradigms tend to shift every five or 10 years, the x86 is a true dinosaur.

Difference between Traditional Computer and Virtual machines



(a) Traditional computer



(b) After virtualization

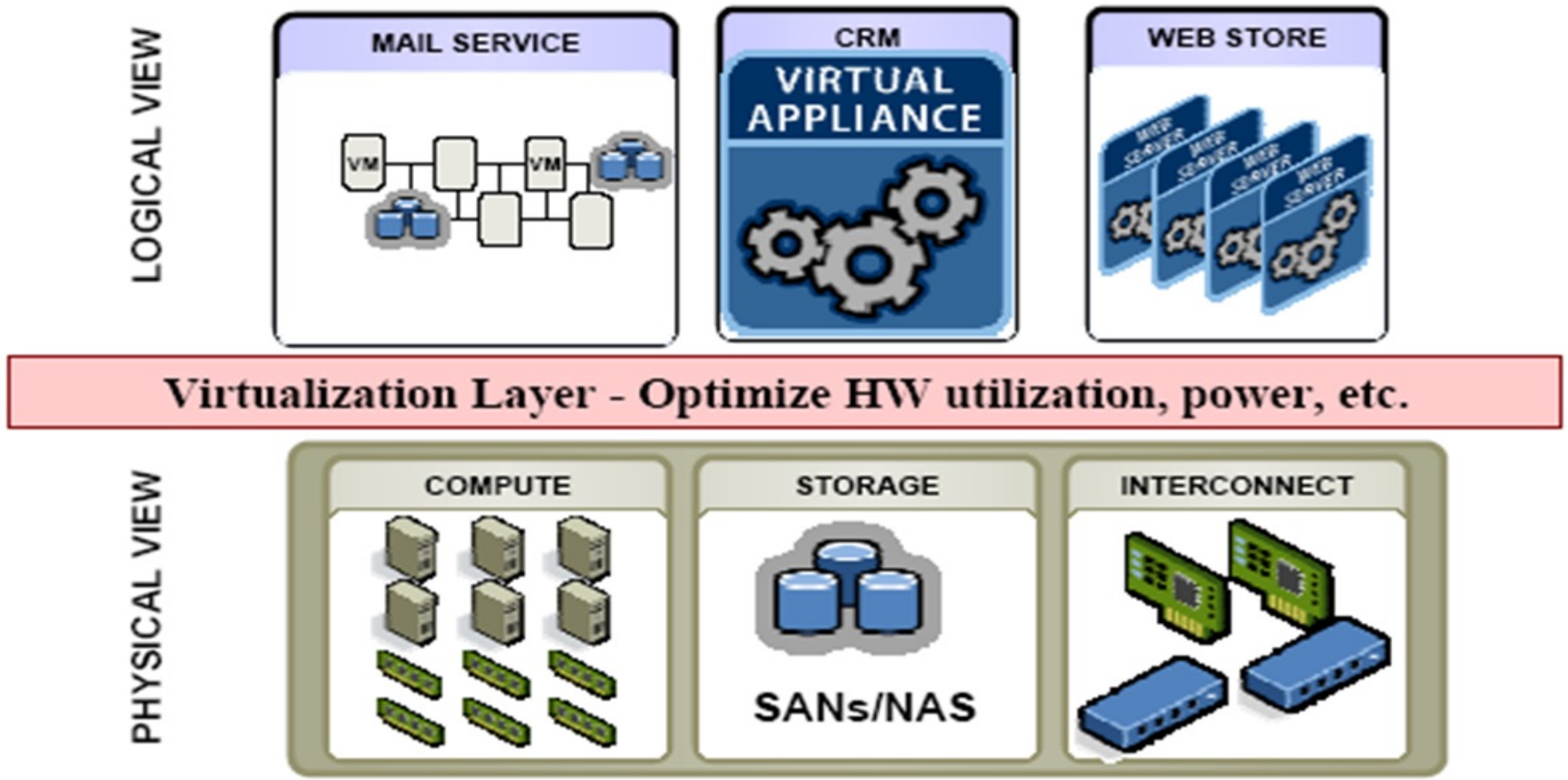
Virtual Machine, Guest Operating System, and VMM (Virtual Machine Monitor) :

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

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.

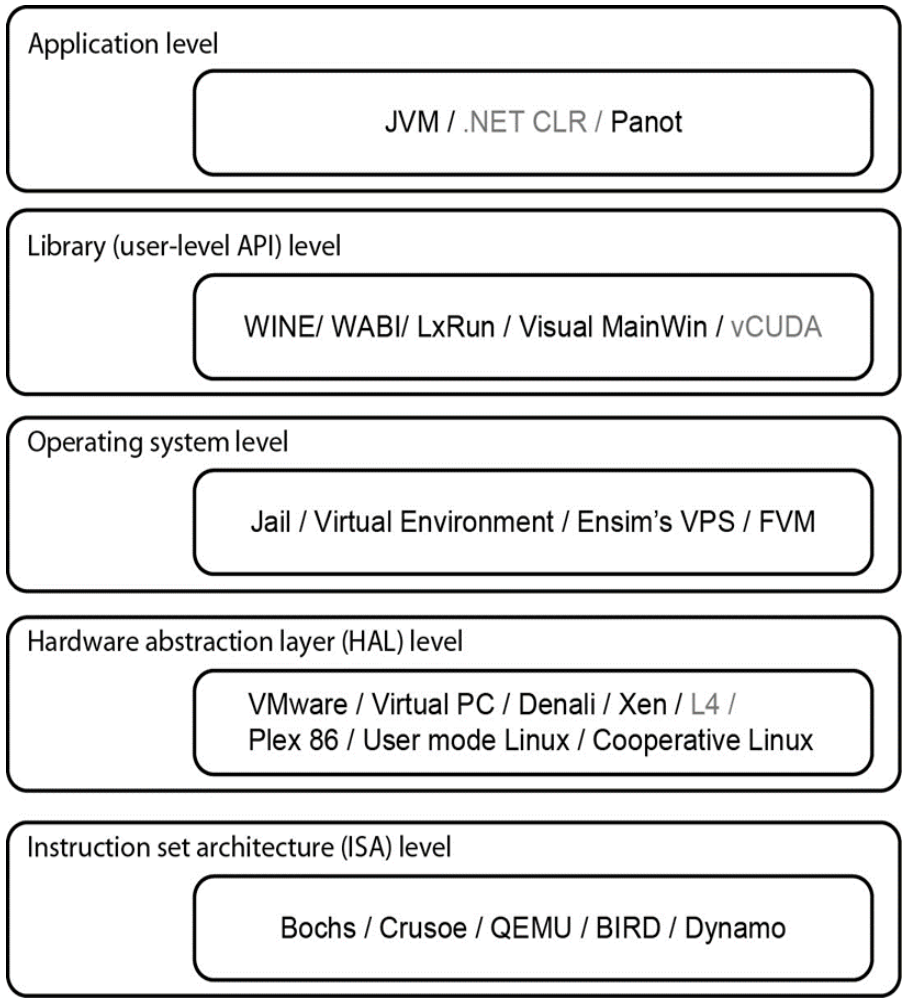
User's view of virtualization



Virtualization Ranging from Hardware to Applications in Five Abstraction Levels

A Java virtual machine (JVM) is **a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode**.

The Common Language Runtime, the virtual machine component of Microsoft .NET Framework, manages the execution of .NET programs. Just-in-time compilation converts the managed code into machine instructions which are then executed on the CPU of the computer.



User level program,behave like a real machine,I/O map, I/O mem,-mapping with the H/W.

Programming API's, application binary interfaces (binary format)→ every one can use.

Host and guest comm, Vir OS level, Win in host and win in guest→ redundant case→ OS level will identify

Trap-Privilege Instruction and send to VMM, native level instruction. Higher to lower level language

Micro-control level, Instruction set→ H/w interaction, convert to native language

Level of virtualization implementation

1) Application level : Process level virtualization (HLL) high level language .
2) Library level : Virtual hosting environment for larger API hooks
3) OS level:- allocate large numbers of users
4) HAL (H/w abstraction layer level): to virtualize the computer resources like its processor/memory and I/o devices.
5) ISA (Inst. set arch.): legacy binary code can be used.

Virtualization at ISA (Instruction Set Architecture) level:

- At the ISA level, virtualization is performed ***by emulating a given ISA by the ISA of the host machine.***
- e.g, MIPS binary code can run on **an x-86-based host machine with the help of ISA emulation.**
- With this approach, it is possible **to run a large amount of legacy binary code written for various processors on any given new hardware host machine**.
- **Instruction set emulation leads to virtual ISAs created on any hardware machine**.
- The **basic emulation method is through code interpretation**.
- An interpreter program interprets the **source instructions to target instructions** one by one.
- One **source instruction may require tens or hundreds of native target instructions** to perform its function.

Virtualization at ISA (Instruction Set Architecture) level

- Obviously, this process is relatively slow.
- For better performance, **dynamic binary translation is desired**.
- This approach translates basic blocks of dynamic source instructions to target instructions.
- The basic blocks can also be extended to program traces or super blocks to increase translation efficiency.
- **Instruction set emulation requires binary translation and optimization**.
- A **virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.**

- **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 (skip):

- 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
- The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices.
- The intention is to upgrade the hardware utilization rate by multiple users concurrently.

Advantage:

- Has higher performance and good application isolation

Shortcoming & limitation:

- Very expensive to implement (complexity)

Virtualization at Operating System (OS) level:

- This refers to an abstraction layer between traditional OS and user applications.
- This **virtualization creates isolated containers on a single physical server and the OS-instance to utilize the hardware and software in datacenters.**
- The containers behave like real servers.
- **OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.**
- It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.
- Typical systems: Jail / Virtual Environment / Ensim's VPS / FVM

Advantage:

- **Has 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.--→ this has been overcome and will be discussed shortly.**

Virtualization at OS Level

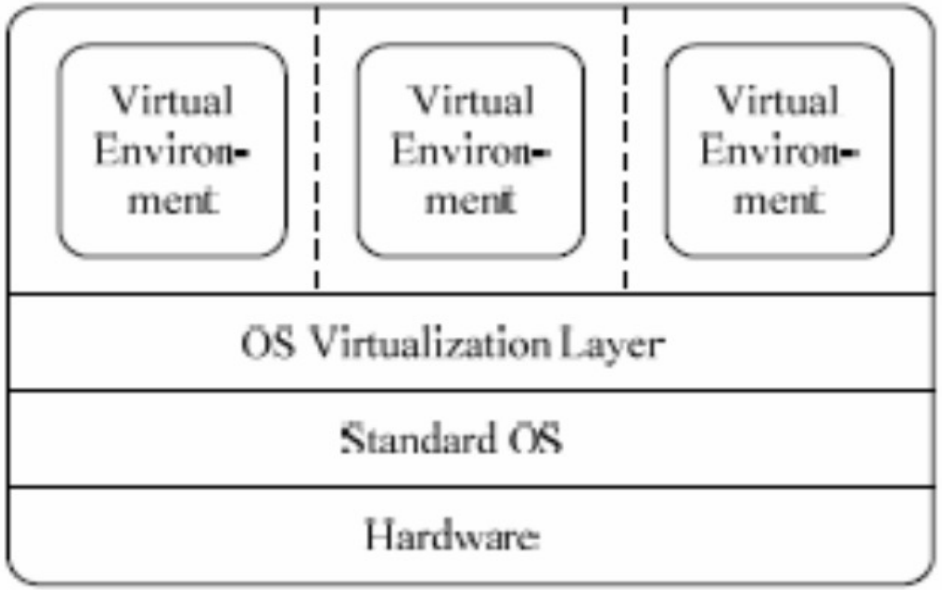
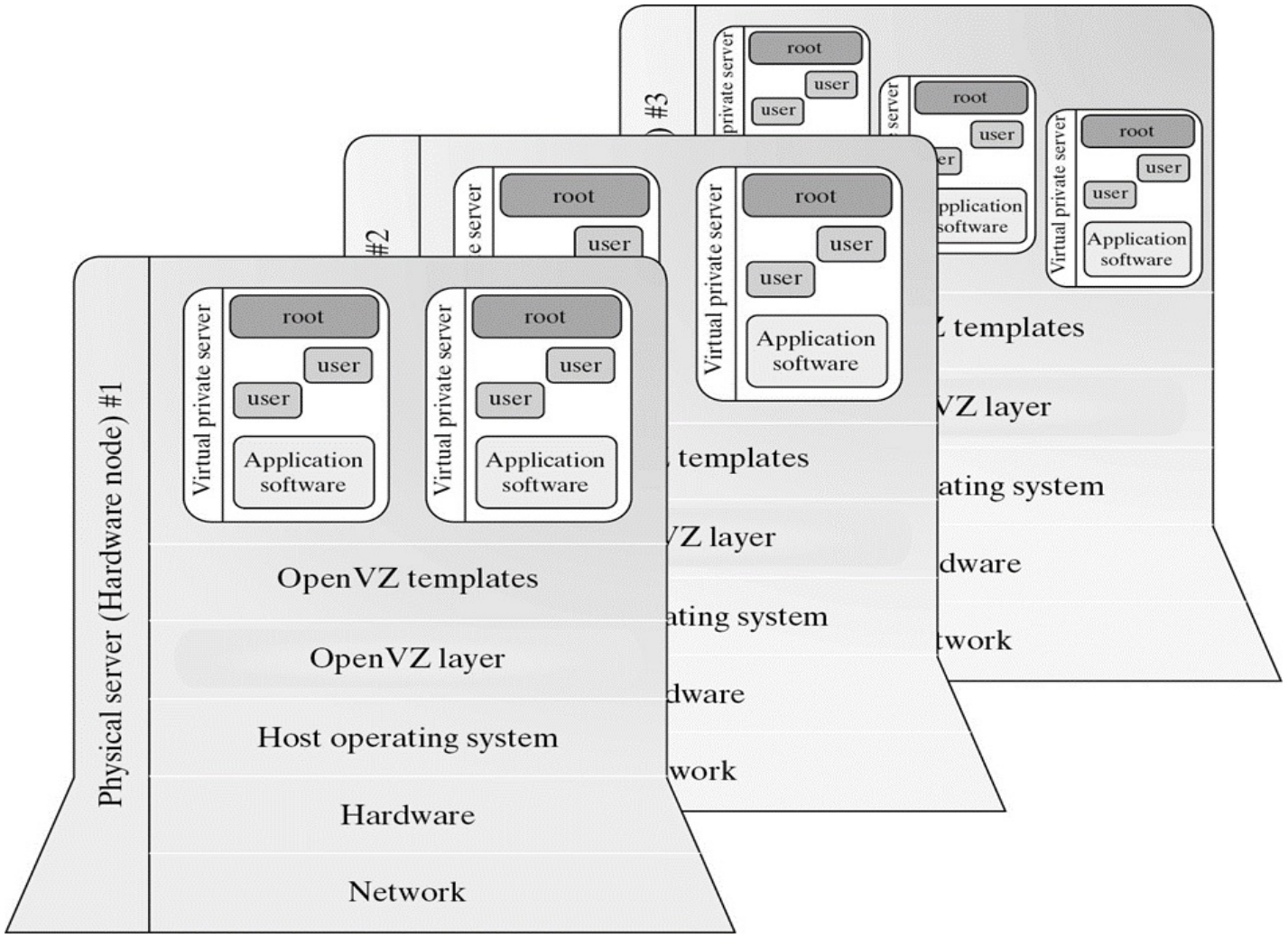


Figure 6.3 The virtualization layer is inserted inside an OS to partition the hardware resources for multiple VMs to run their applications in virtual environments

Virtualization for Linux and Windows NT Platforms



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.

Case study-SKIP

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

Advantages of OS Extension for Virtualization

- VMs at OS level has minimum startup/shutdown costs
- 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:

- Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS.
- Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization.
- Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks.
- The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts.
- Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

- 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

The vCUBE for Virtualization of GPGPU

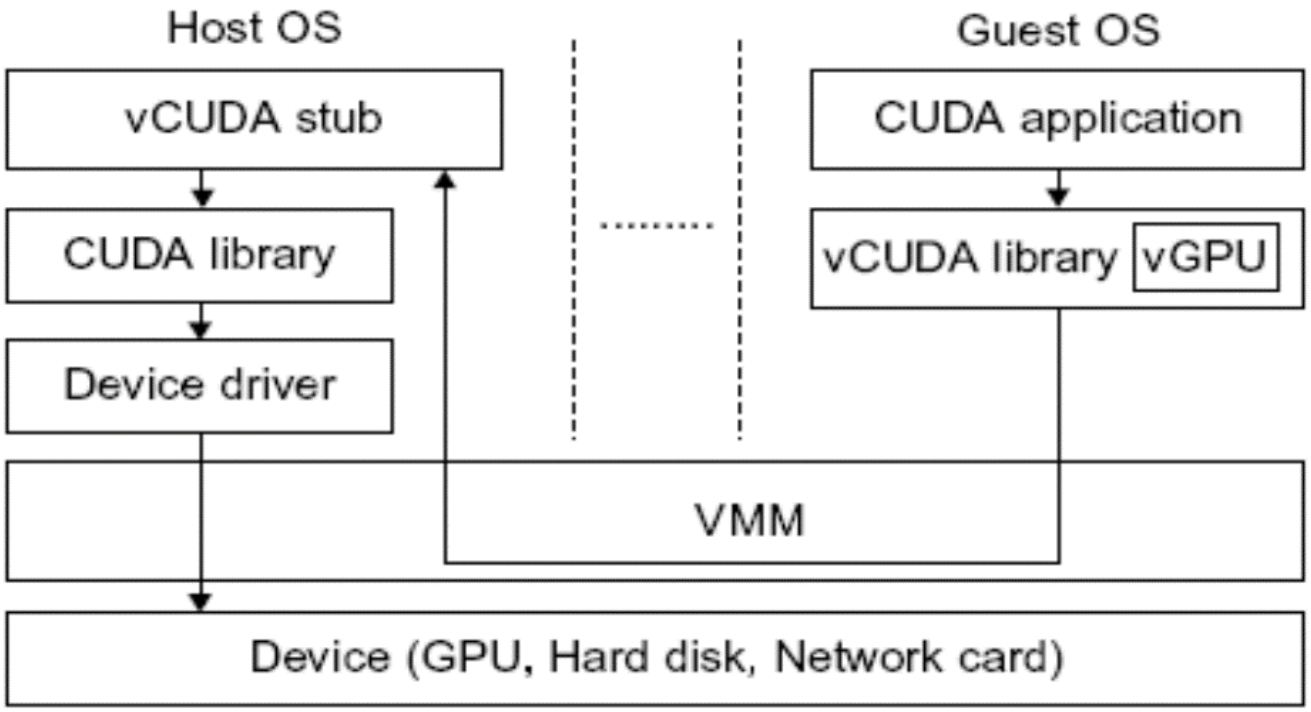
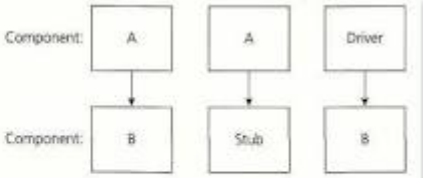


FIGURE 3.4

Basic concept of the vCUDA architecture.

These dummy pieces of code are the **stubs**. On the other hand, **Drivers** are the ones, which are the "calling" programs. **Drivers** are used in bottom up testing approach. **Drivers** are dummy code, which is used when the sub modules are ready but the main module is still not ready.



User-Application level:

- Virtualization at the application level virtualizes an application as a VM.
- On a traditional OS, an application often runs as a process.
- Therefore, **application-level virtualization** is also known as **process-level virtualization.**
- The most popular approach is to **deploy high level language (HLL) VMs**.
- Any program written in the HLL and compiled for this VM will be able to run on it.

- The **Microsoft .NET CLR(Common Language Runtime) and Java Virtual Machine (JVM: converting bytecode** to the **machine** specific **code)** are two good examples of this class of VM.
- https://en.wikipedia.org/wiki/List_of_Java_virtual_machines

Advantage:

- ***Has the best application isolation***

Shortcoming & limitation:

- ***Low performance, low application flexibility and high implementation complexity.***

Relative merits of virtualization at various levels (More X's means higher merit ,with a maximum of 5 X's)

Table 3.1 Relative Merits of Virtualization at Various Levels

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

More Xs mean higher merit

Relative Merits of Different Approaches

- The column headings correspond to four technical merits. "Higher Performance" and "**Application Flexibility**" are self-explanatory.
- "**Implementation Complexity**" implies the cost to implement that particular virtualization level.
- "**Application Isolation**" refers to the effort required to isolate resources committed to different VMs. Each row corresponds to a particular level of virtualization.
- However, the hardware and application levels are also the most expensive to implement.
- User isolation is the most difficult to achieve.
- **ISA implementation** offers the best application flexibility.

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).

Major VMM and Hypervisor Providers (self study)

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 XEN Architecture

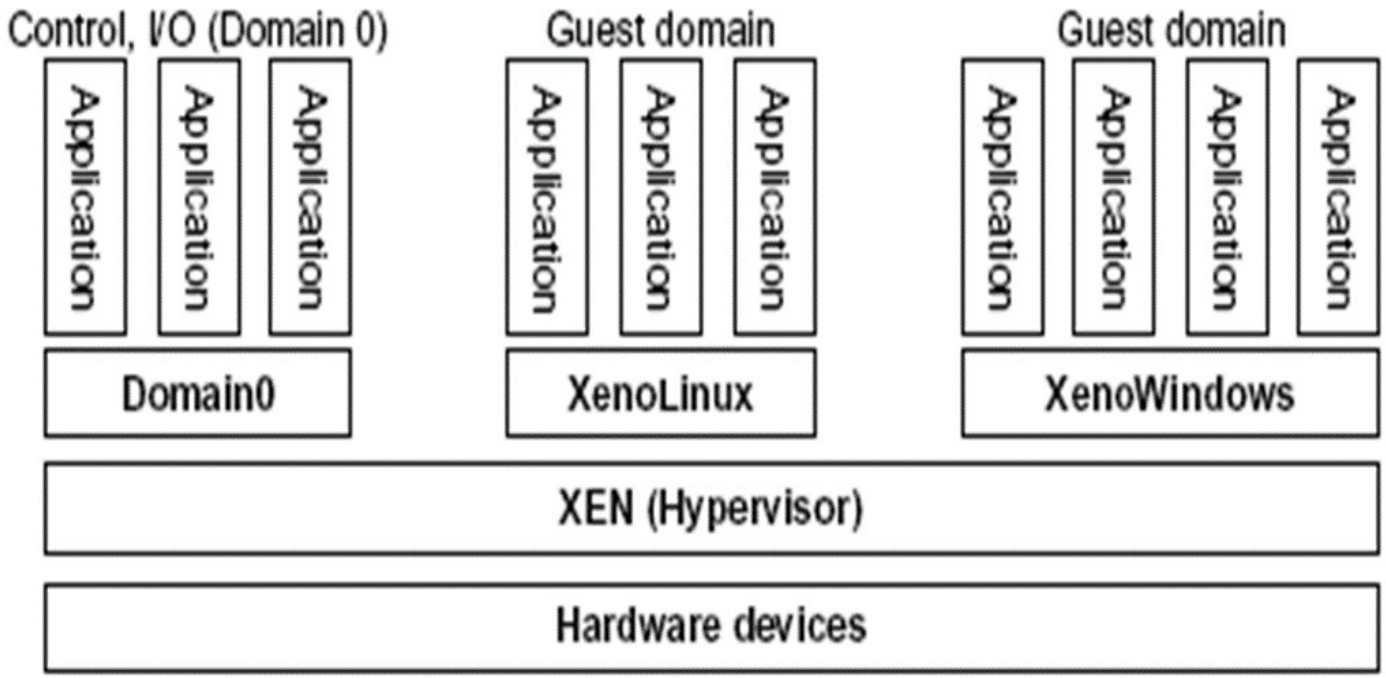


FIGURE 3.5

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

The XEN Architecture

- 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.**
- Xen does not include any device drivers natively.
- 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 and Oracle VM.

- For example, Xen is based on Linux and its security level is ***C2***. (U.S. National Computer **Security**Center (NCSC) and granted to products that pass Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) tests)
- 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.

Full Virtualization vs. Para-Virtualization

Full virtualization: https://www.youtube.com/watch?v=CLR0pq9dy4g&t=2s
Para virtualization: https://www.youtube.com/watch?v=_Tltue_pa-o

Full virtualization

- Does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation.
- VMware Workstation applies full virtualization, which uses binary translation to automatically modify x86 software on-the-fly to replace critical instructions.
- Advantage: no need to modify OS.
- Disadvantage: binary translation slows down the performance.

Para virtualization

- Reduces the overhead, but cost of maintaining a paravirtualized OS is high.
- The improvement depends on the workload.
- Para virtualization must modify guest OS, non-virtualizable instructions are replaced by hypercalls that communicate directly with the hypervisor or VMM.
- Para virtualization is supported by Xen, Denali and VMware ESX.

Some more examples of full and para virtualization>>>> next few slides…

Full Virtualization

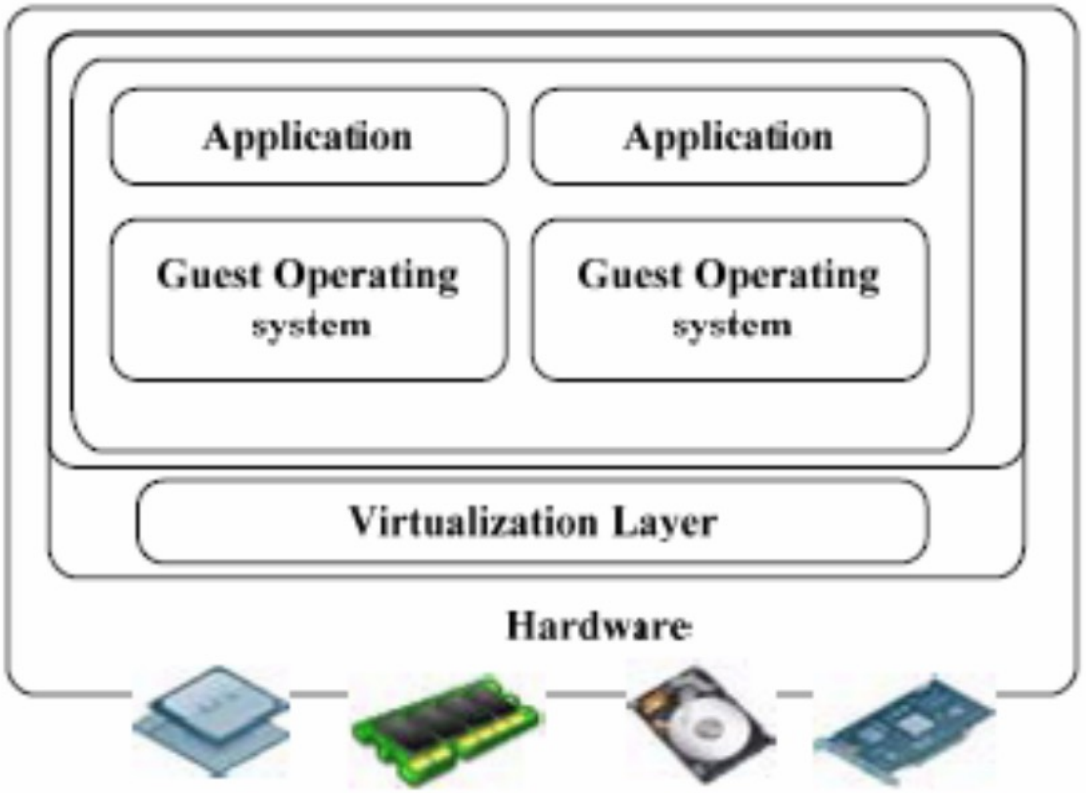


Figure 6.9 The concept of full virtualization using a hypervisor or a VMM directly sitting on top of the bare hardware devices. Note that no host OS is used here as in Figure 6.11.

Binary Translation of Guest OS Requests using a VMM:

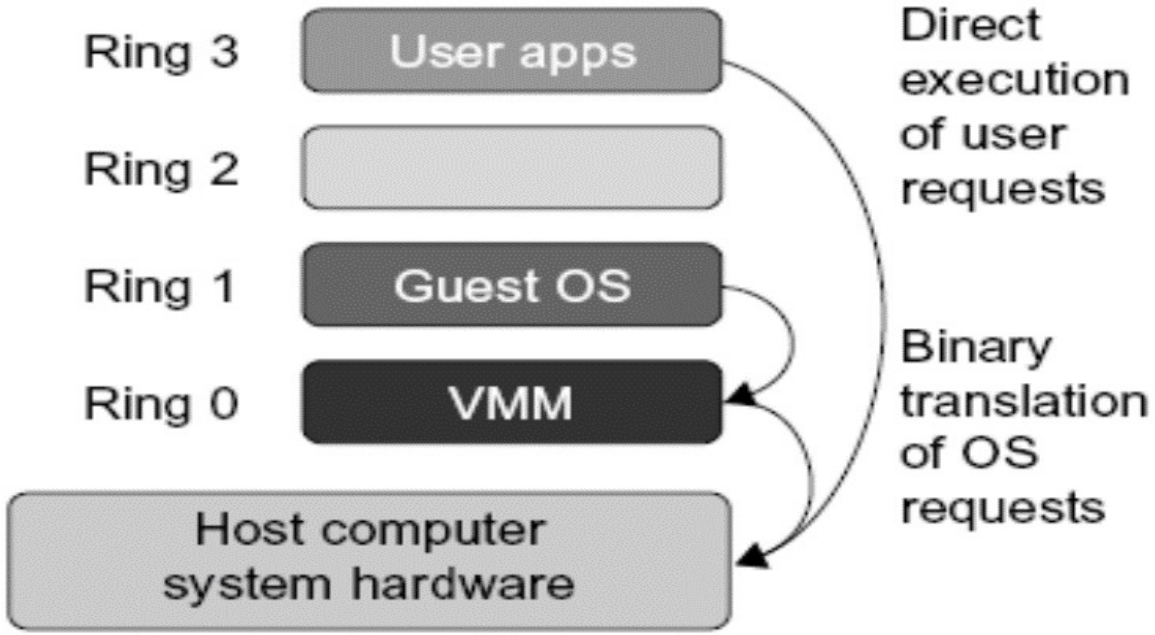
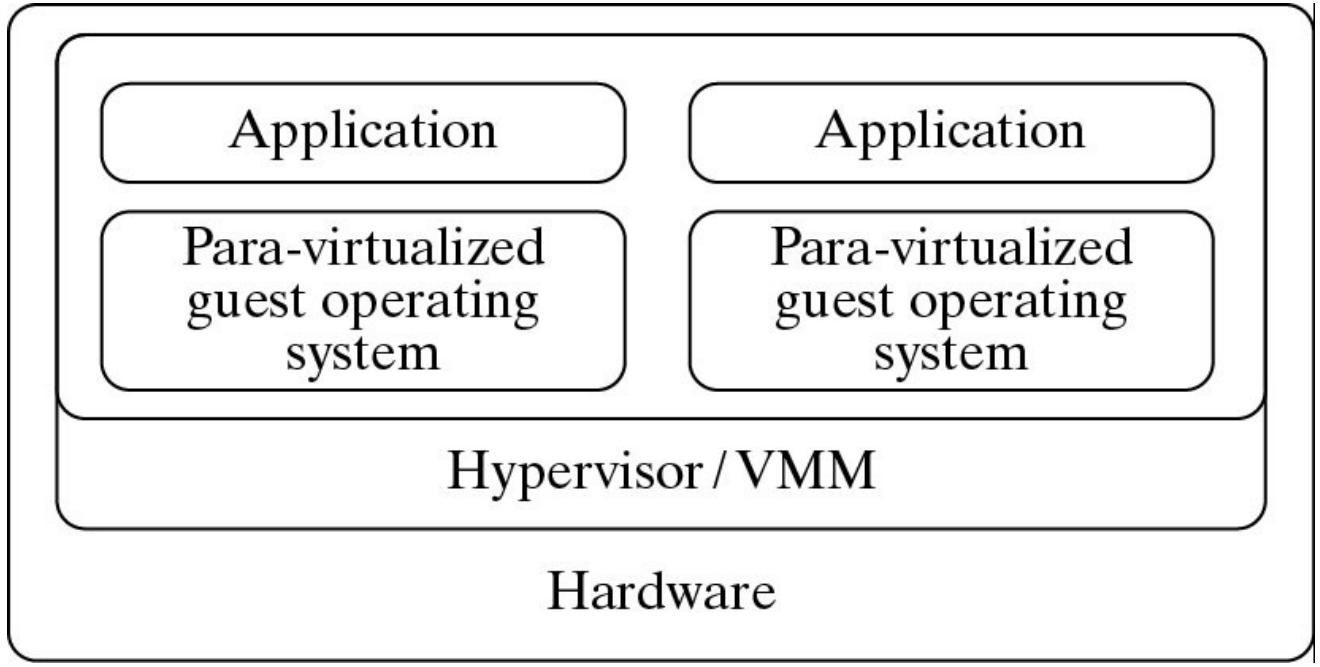


FIGURE 3.6

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.

Para-Virtualization with Compiler Support.



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

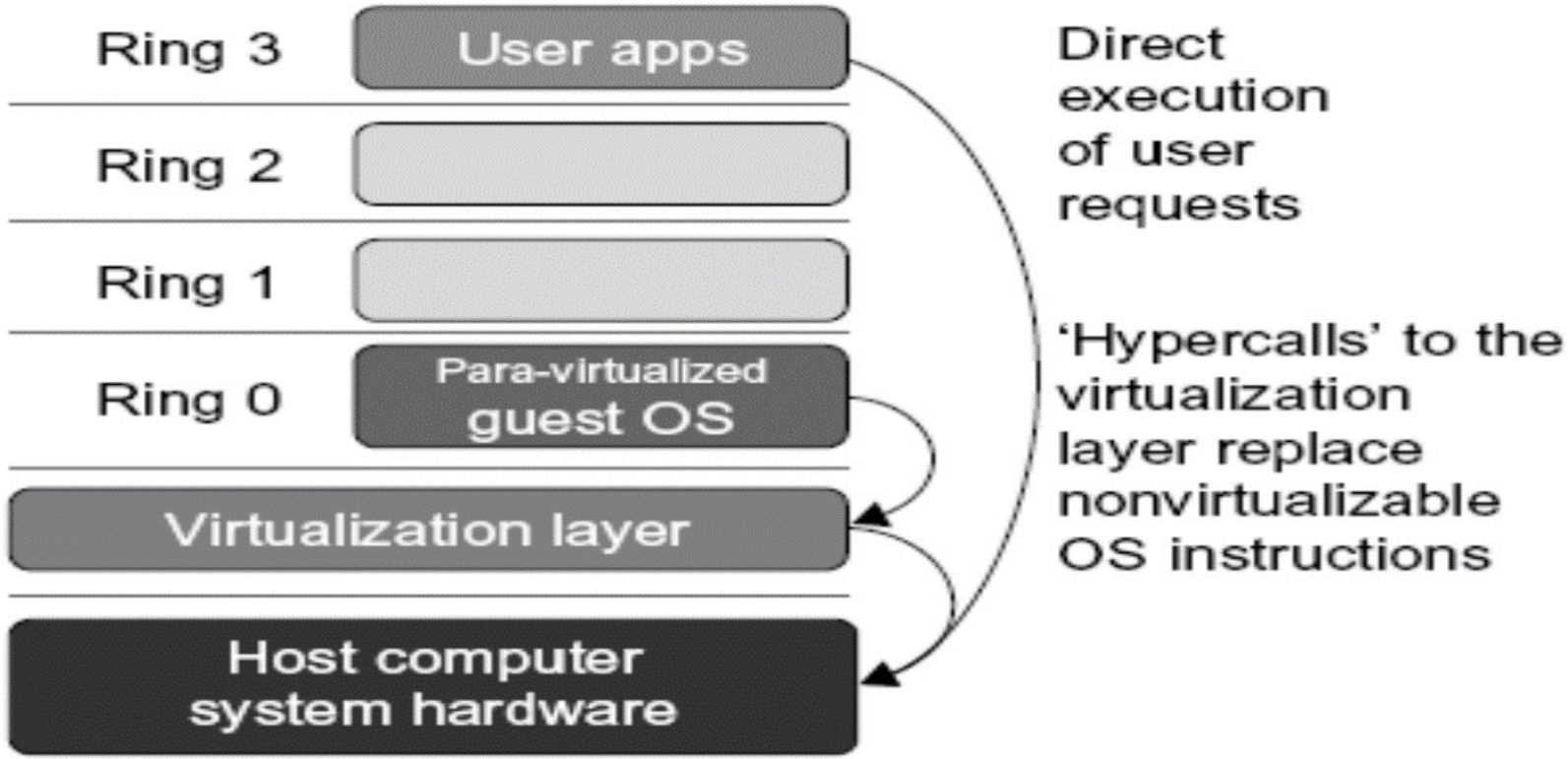


FIGURE 3.8

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

Summary Comparison of the Current State of x86 Virtualization Techniques

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	Yes	Yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

Full Virtualization vs. Paravirtualization

- Paravirtualization is different from full virtualization, where the unmodified OS does not know it is virtualized and sensitive OS calls are trapped using binary translation at run time. In paravirtualization, these instructions are handled at compile time when the non-virtualizable OS instructions are replaced with hypercalls.

- The advantage of paravirtualization is lower virtualization overhead, but the performance advantage of paravirtualization over full virtualization can vary greatly depending on the workload. Most user space workloads gain very little, and near native performance is not achieved for all workloads.

- As paravirtualization cannot support unmodified operating systems (e.g. Windows 2000/XP), its compatibility and portability is poor.

Full Virtualization vs. Paravirtualization

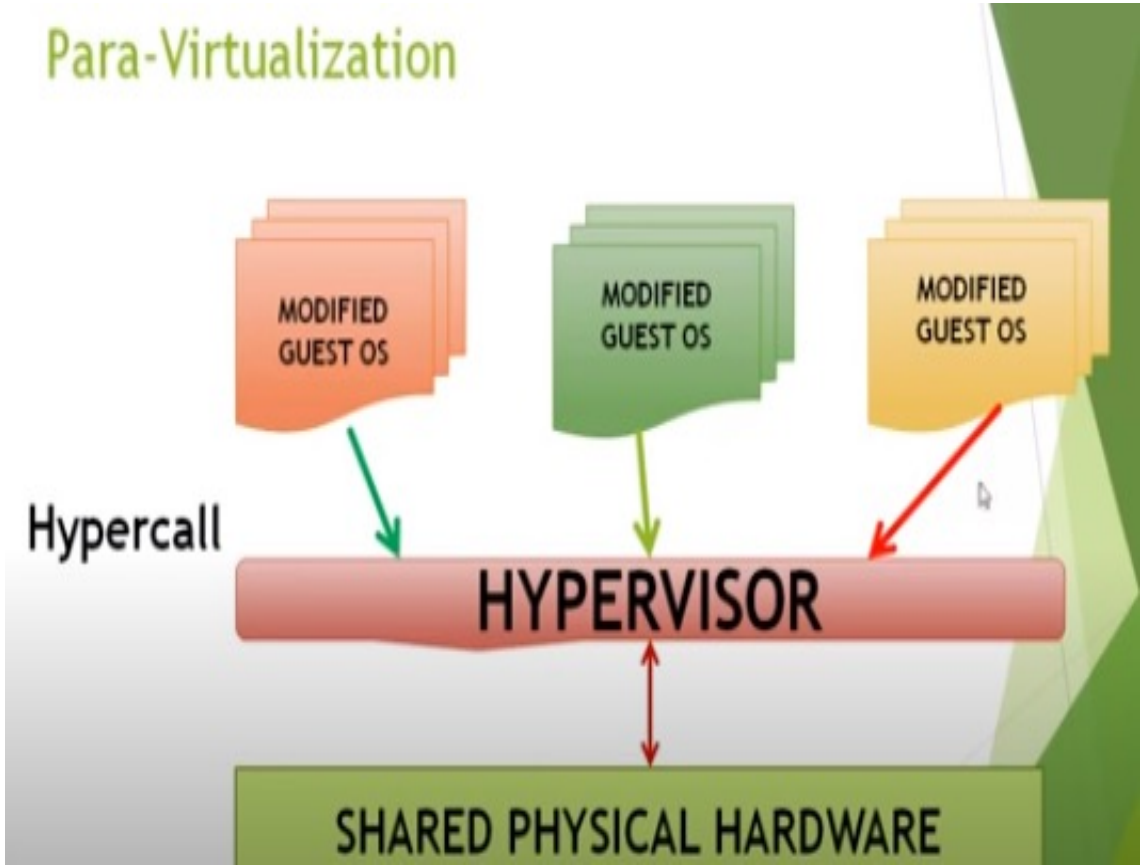
- Paravirtualization can also introduce significant support and maintainability issues in production environments as it requires deep OS kernel modifications. The invasive kernel modifications tightly couple the guest OS to the hypervisor with data structure dependencies, preventing the modified guest OS from running on other hypervisors or native hardware.

- The open source Xen project is an example of paravirtualization that virtualizes the processor and memory using a modified Linux kernel and virtualizes the I/O using custom guest OS device drivers.

PARAVIRTUALIZATION

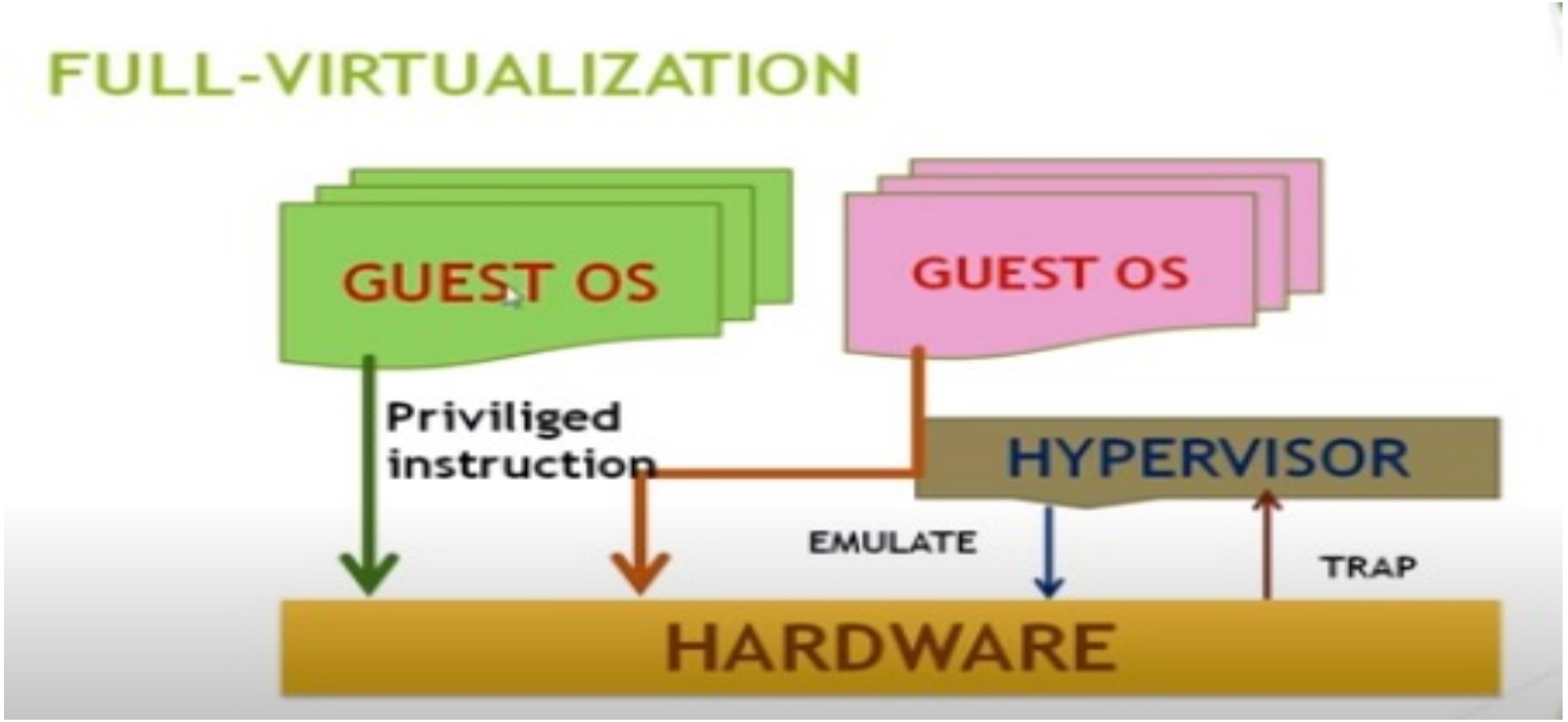
- Paravirtualization is virtualization in which the guest operating system (the one being virtualized) is aware that it is a guest .
- Accordingly has drivers that, instead of issuing hardware commands, simply issue commands directly to the host operating system.
- This also includes memory and thread management as well, which usually require unavailable privileged instructions in the processor.

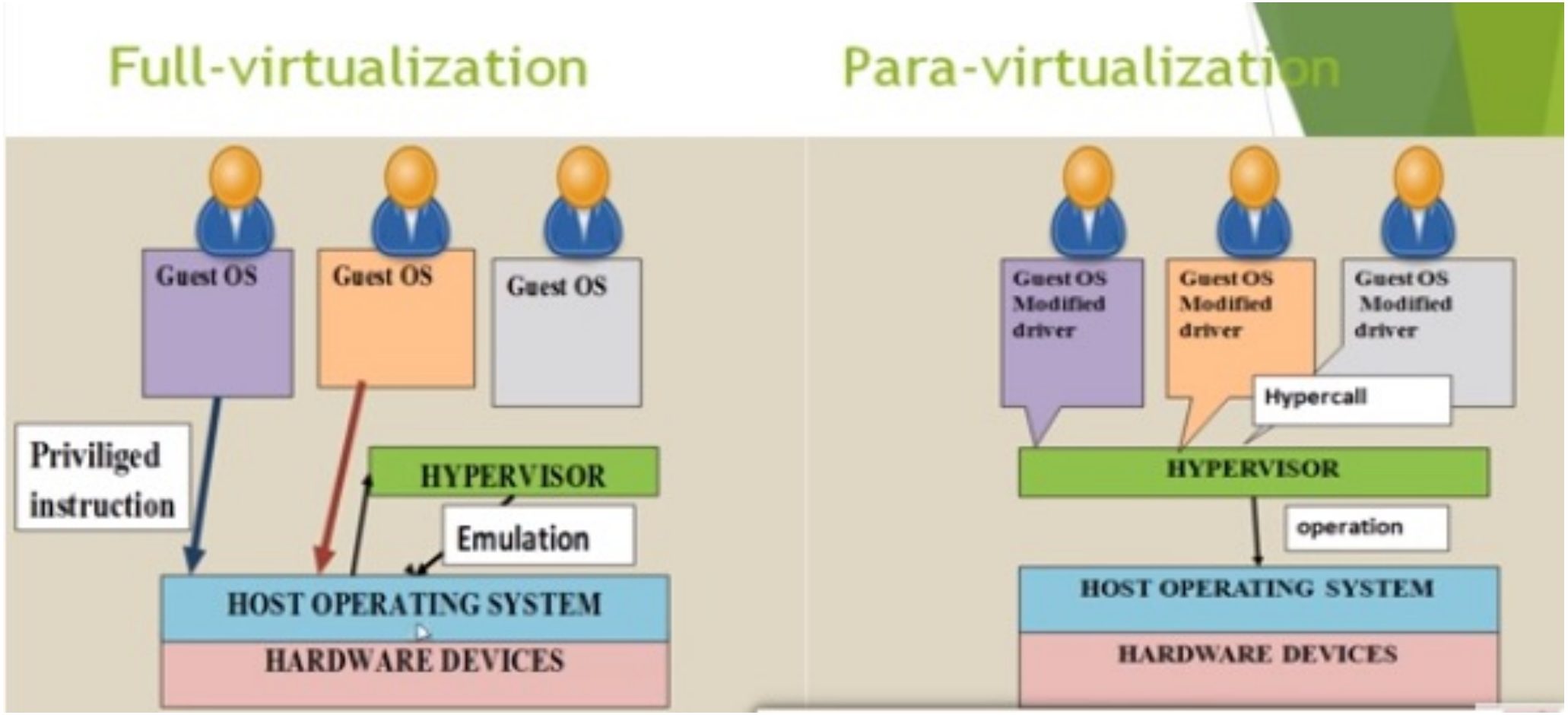
Para-Virtualization



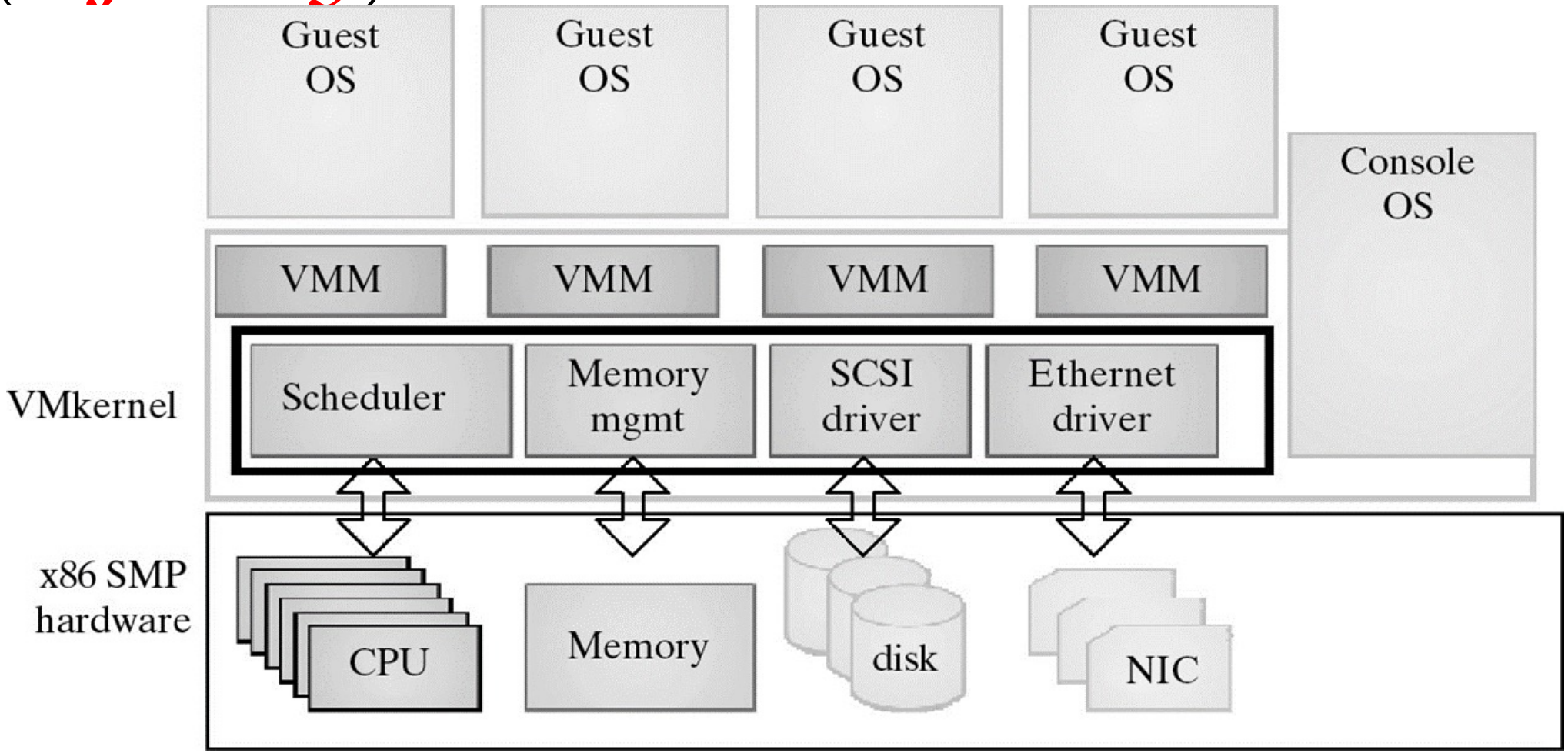
- In paravirtualization, OS hypercalls instructions are handled at compile time when the non-virtualizable OS instructions are replaced with hypercalls.
- The advantage of paravirtualization is lower virtualization overhead,.
- The performance advantage of paravirtualization over full virtualization can vary greatly depending on the workload.

Full Virtualization




Full-virtualization
Guest OS
Guest OS
Guest OS
Priviliged instruction
HYPERVISOR
Emulation
HOST OPERATING SYSTEM
HARDWARE DEVICES
Para-virtualization
Guest OS Modified driver
Guest OS Modified driver
Guest OS Modified driver
Hypercall
HYPERVISOR
operation
HOST OPERATING SYSTEM
HARDWARE DEVICES

VMWare ESX Server for Para-Virtualization (*self study*)→ESX→ VMM



Memory Virtualization Challenges

Address Translation

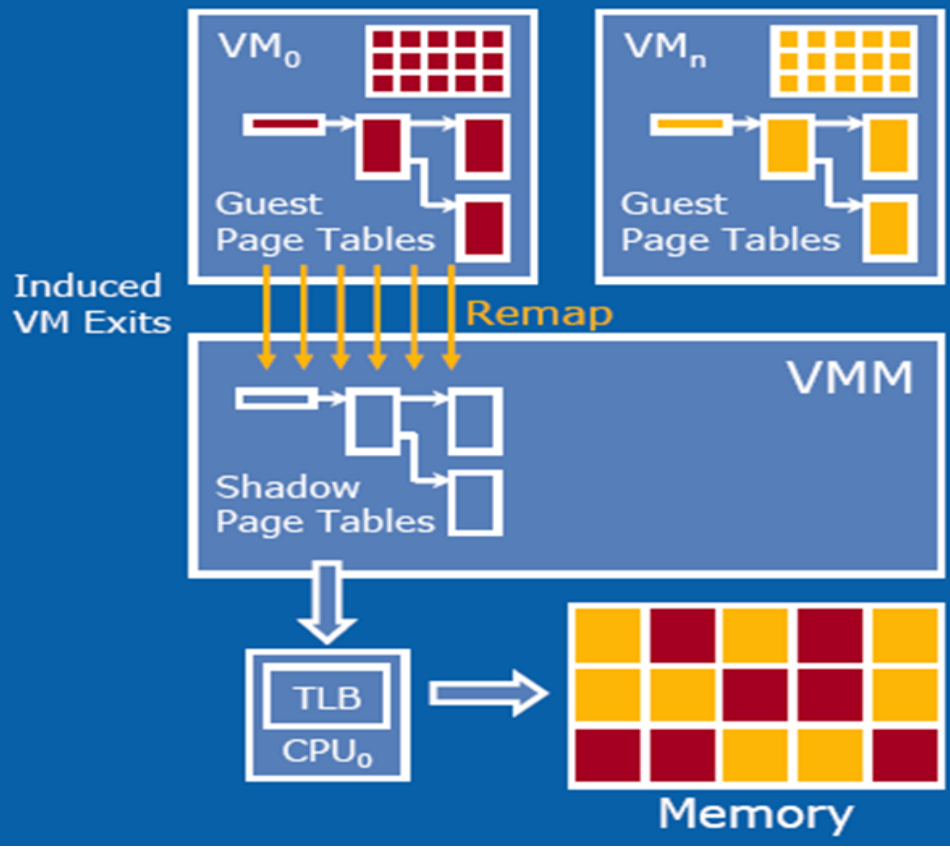
- Guest OS expects contiguous, zero-based physical memory
- VMM must preserve this illusion

Page-table Shadowing

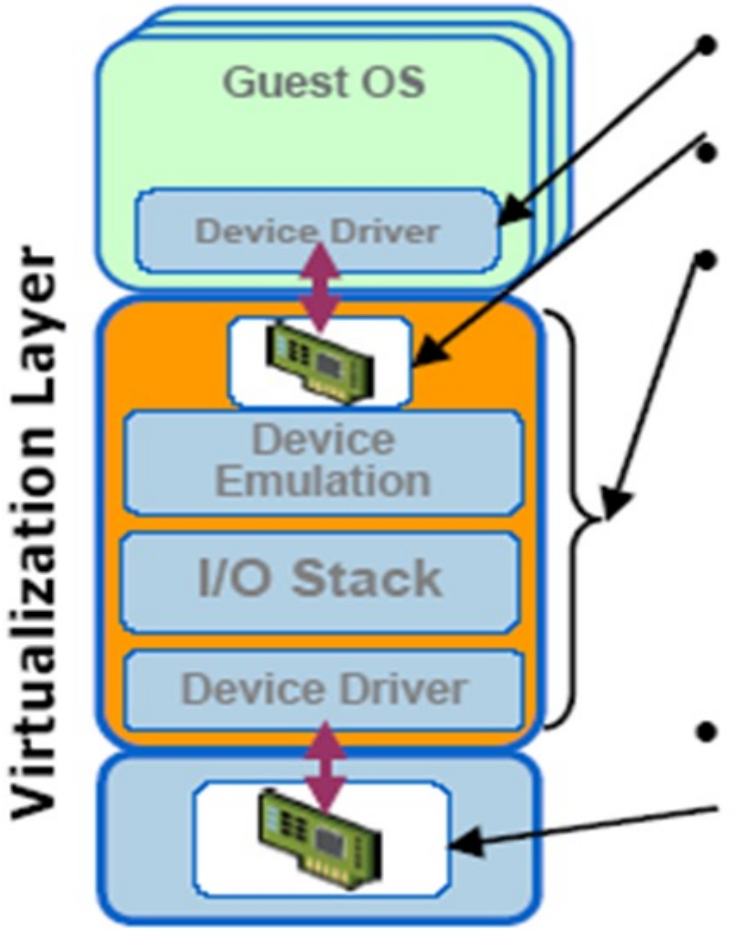
- VMM intercepts paging operations
- Constructs copy of page tables

Overheads

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



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

Copy on write disks

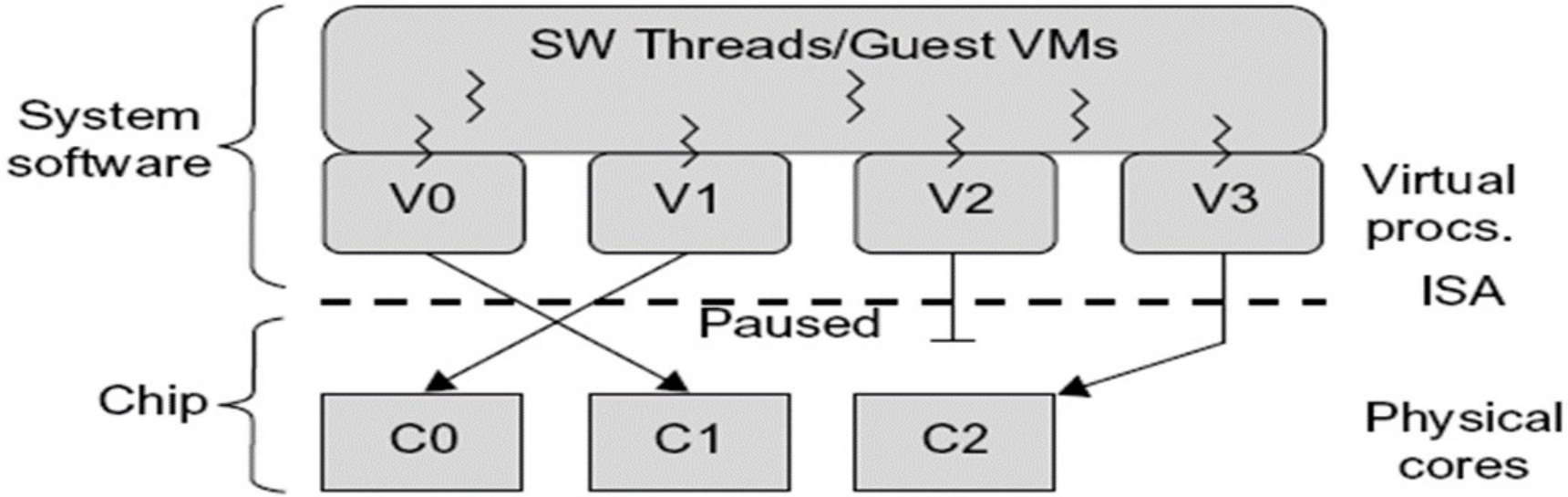
Conclusions on CPU, Memory and I/O Virtualization:

- **CPU virtualization demands hardware-assisted traps of sensitive instructions by the VMM**
- Memory virtualization demands special hardware support (**shadow page tables by VMW are 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.

What is shadowing in memory?

In computing, shadow memory is **a technique used to track and store information on computer memory used by a program during its execution**. Shadow memory consists of shadow bytes that map to individual bits or one or more bytes in main memory.

Multi-Core Virtualization: VCPU vs. traditional CPU

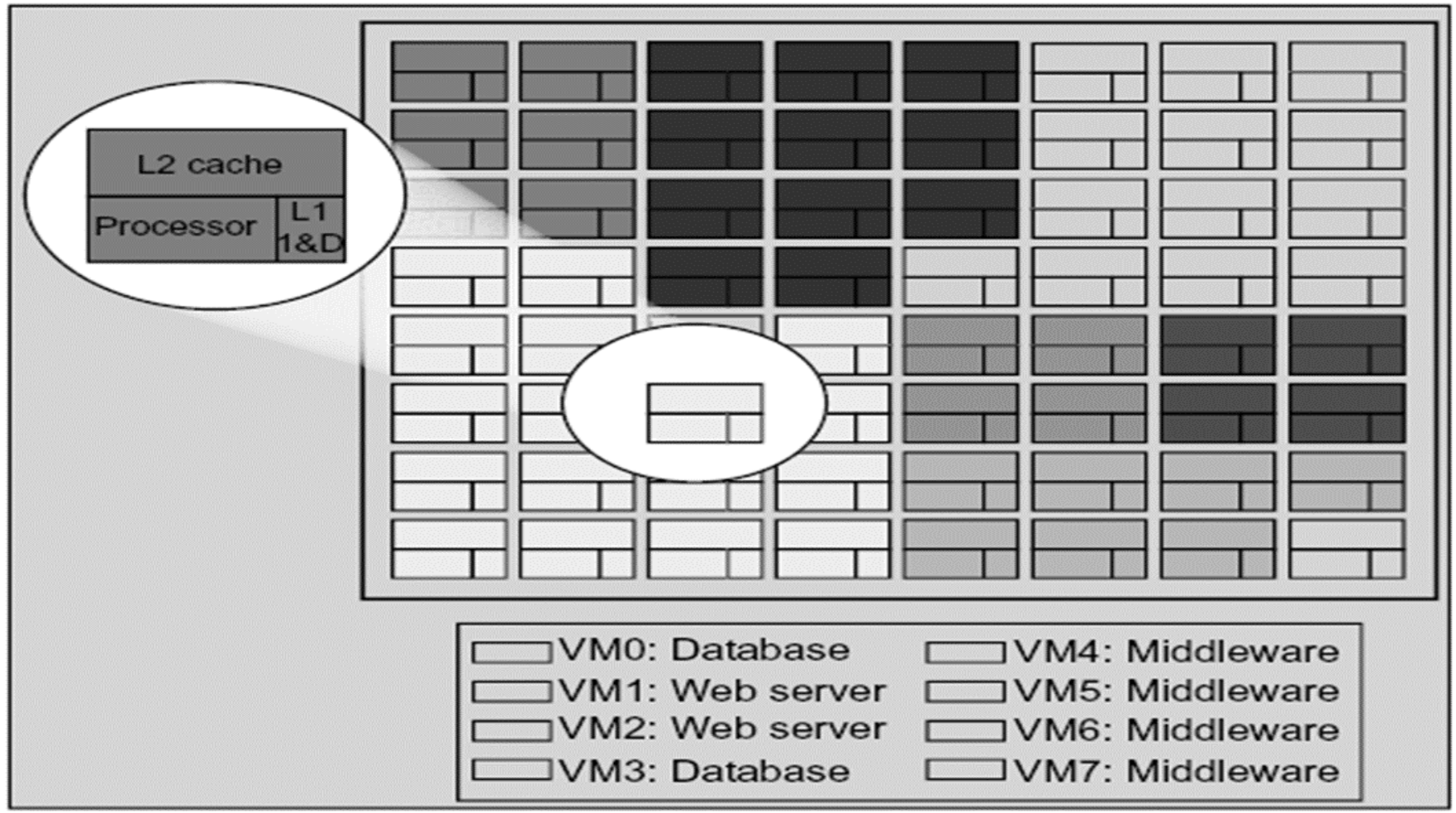


- Figure 3.16 Four VCPUs are exposed to the software, only three cores are actually present. VCPUs V0, V1, and V3 have been transparently migrated, while VCPU V2 has been transparently suspended. (Courtesy of Wells, et al., "Dynamic Heterogeneity and the Need for Multicore Virtualization", *ACM SIGOPS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

Software visible VCPU moving from one core to another and temporarily suspending execution of a VCPU when there are no appropriate cores in which it can run.

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.



(a) Mapping of VMs into adjacent cores

What is virtual cluster in CC

- Virtual cluster are built with the VM's installed at distributed servers from one or more physical clusters. The VM's in a virtual cluster are interconnected logically by a virtual network across several physical networks.
- Each virtual cluster is formed with physical cluster .It is formed with physical machines or the VM hosted by multiple physical clusters.
- (diagram……next page…….)

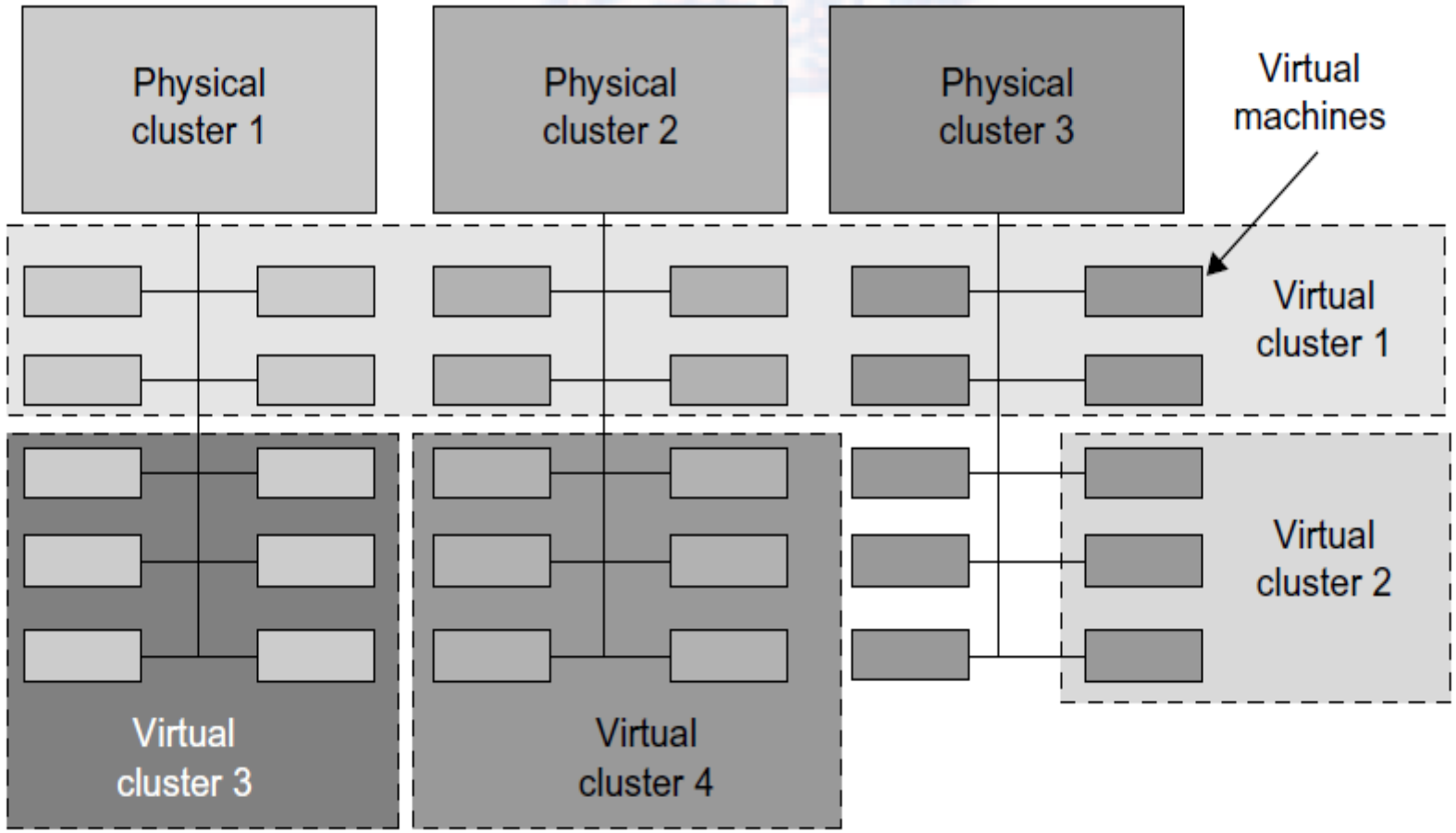
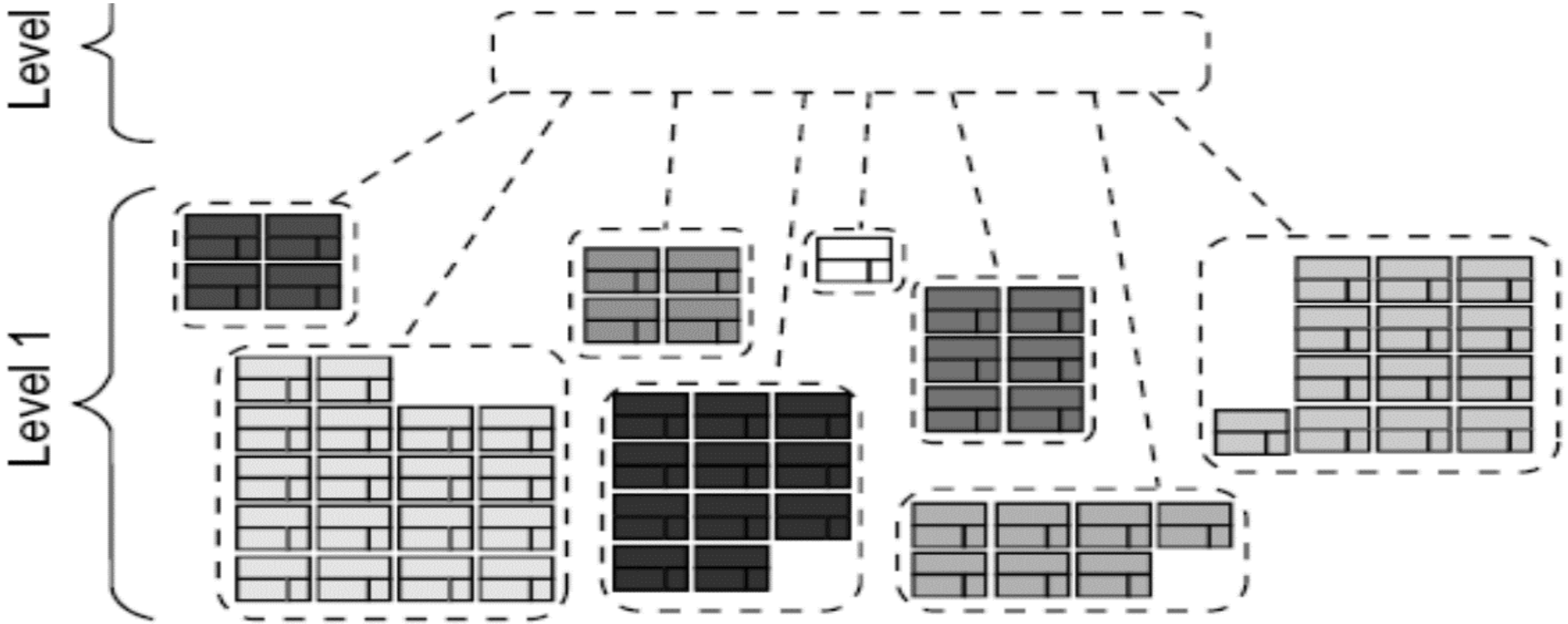


FIGURE 3.18

A cloud platform with four virtual clusters over three physical clusters shaded differently.

Virtual Clusters in Many Cores

Space Sharing of VMs -- Virtual Hierarchy



(b) Multiple virtual clusters assigned to various workloads

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

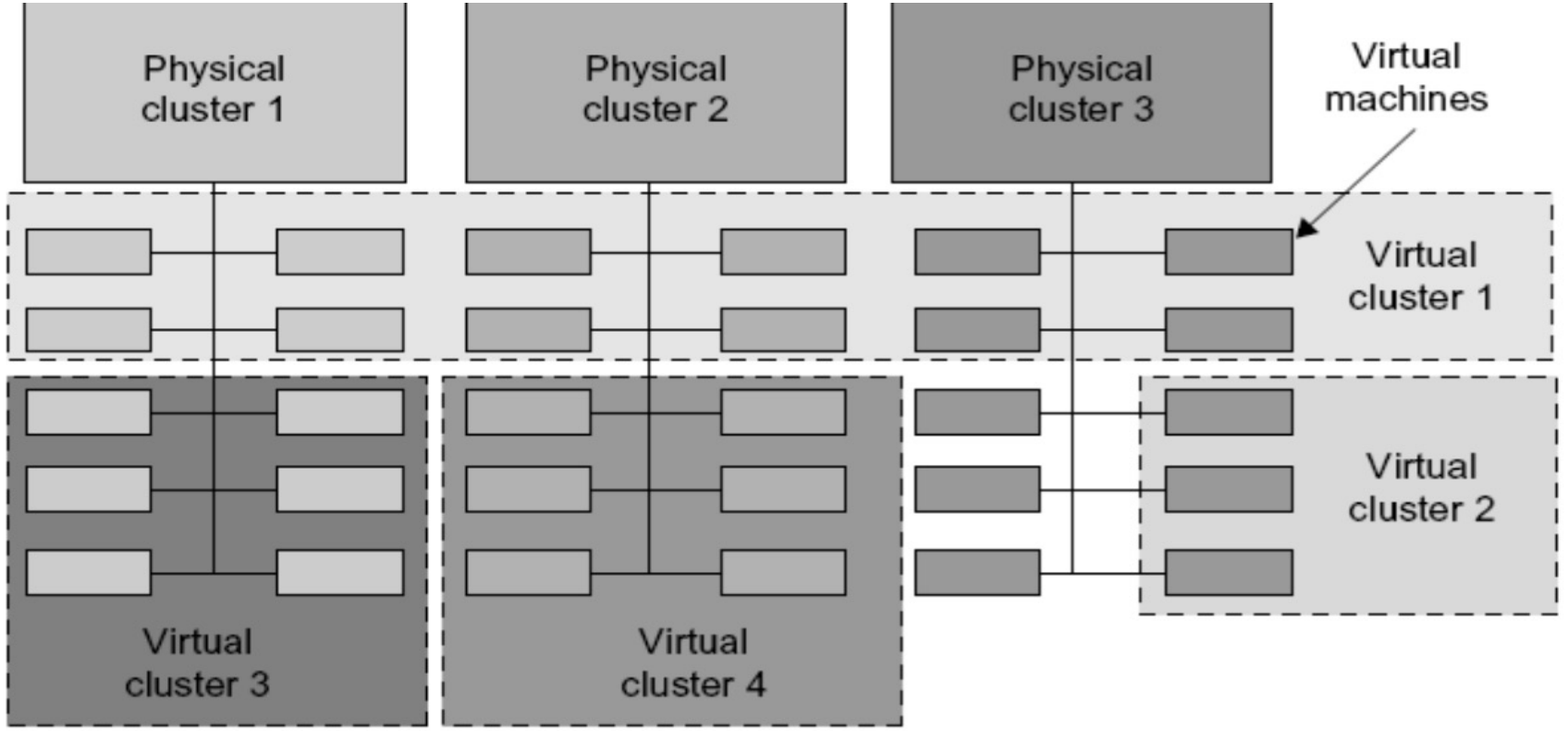


FIGURE 3.18

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

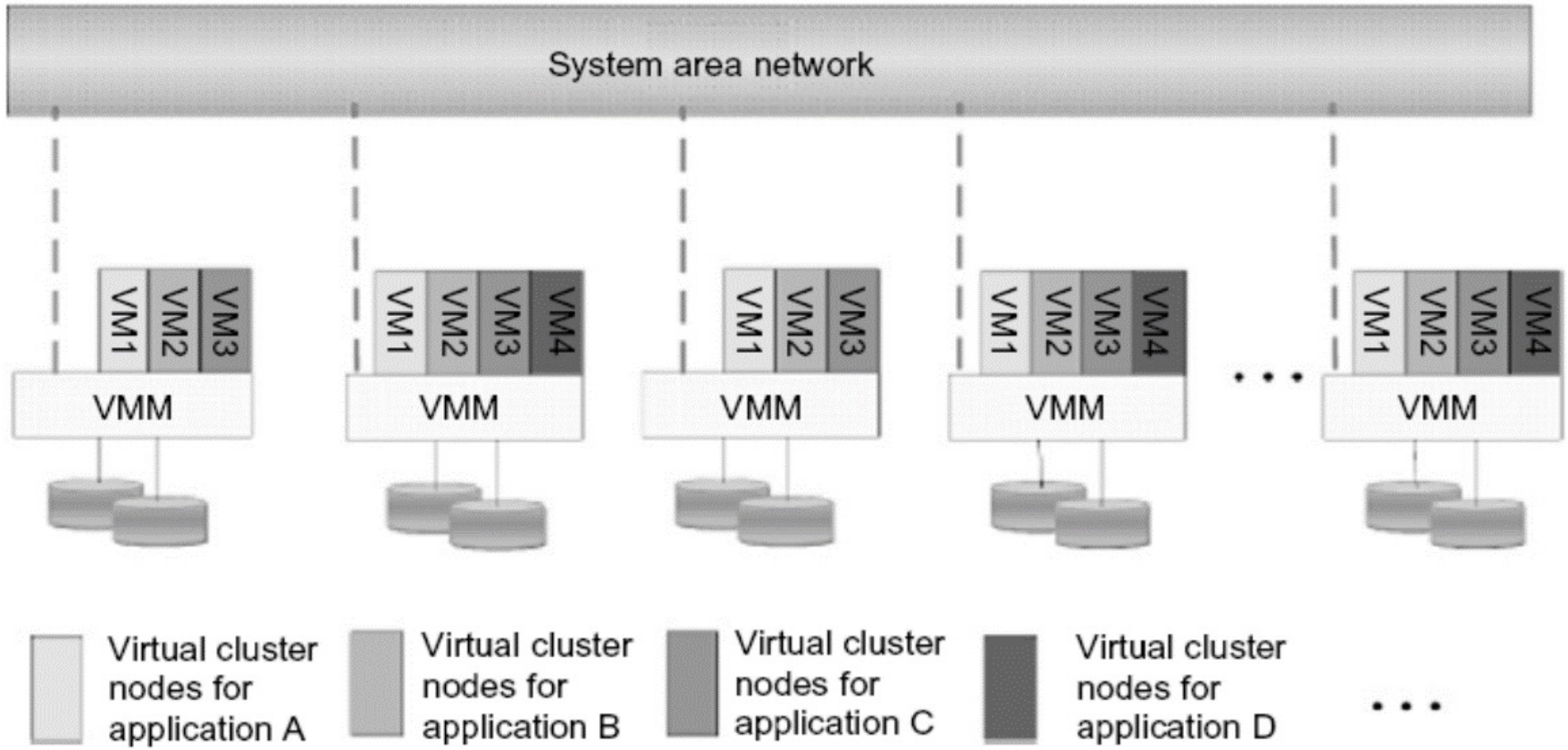


FIGURE 3.19

The concept of a virtual cluster based on application partitioning.

(*Courtesy of Kang, Chen, Tsinghua University 2008*)

Live Migration of Virtual Machines

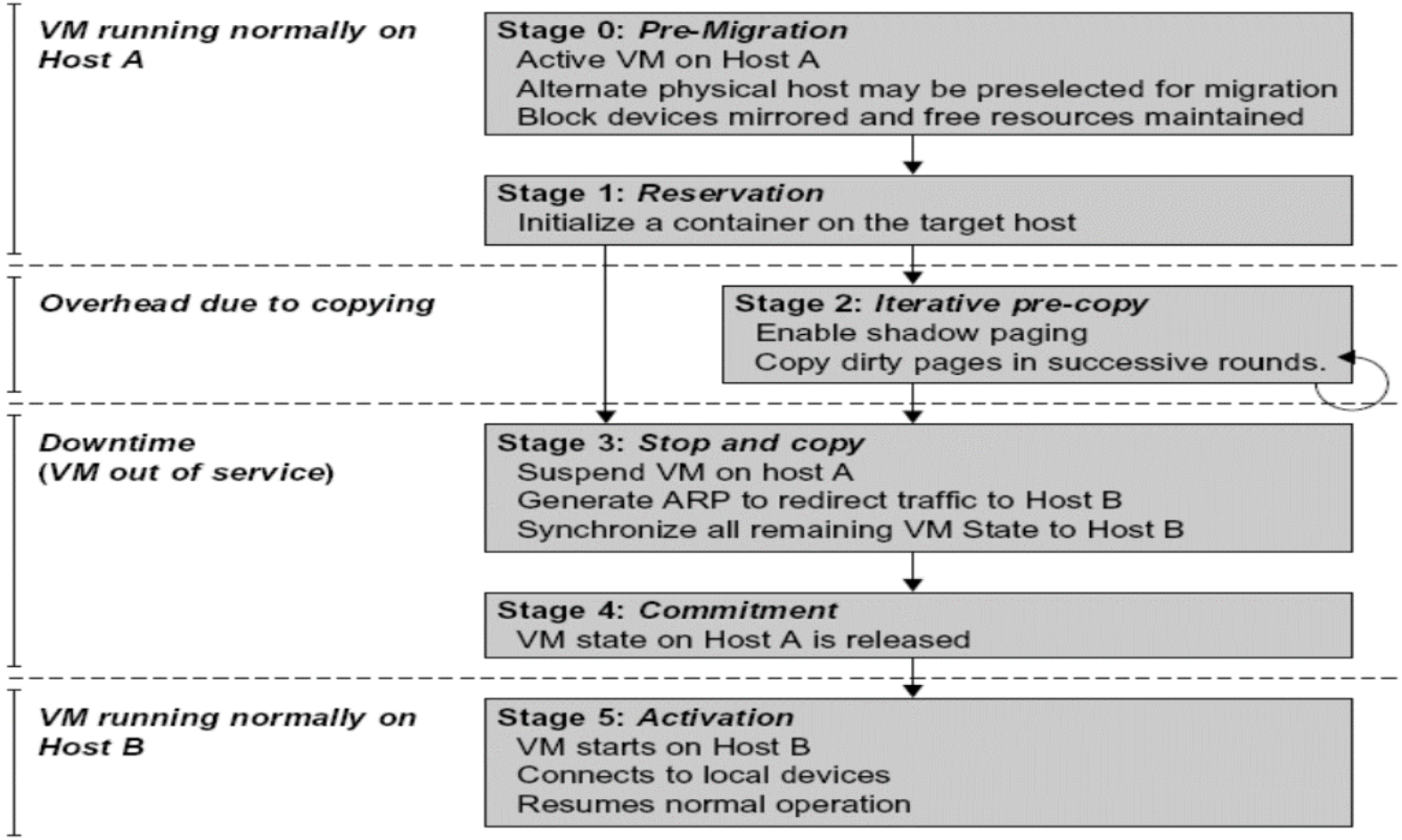


FIGURE 3.20

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

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

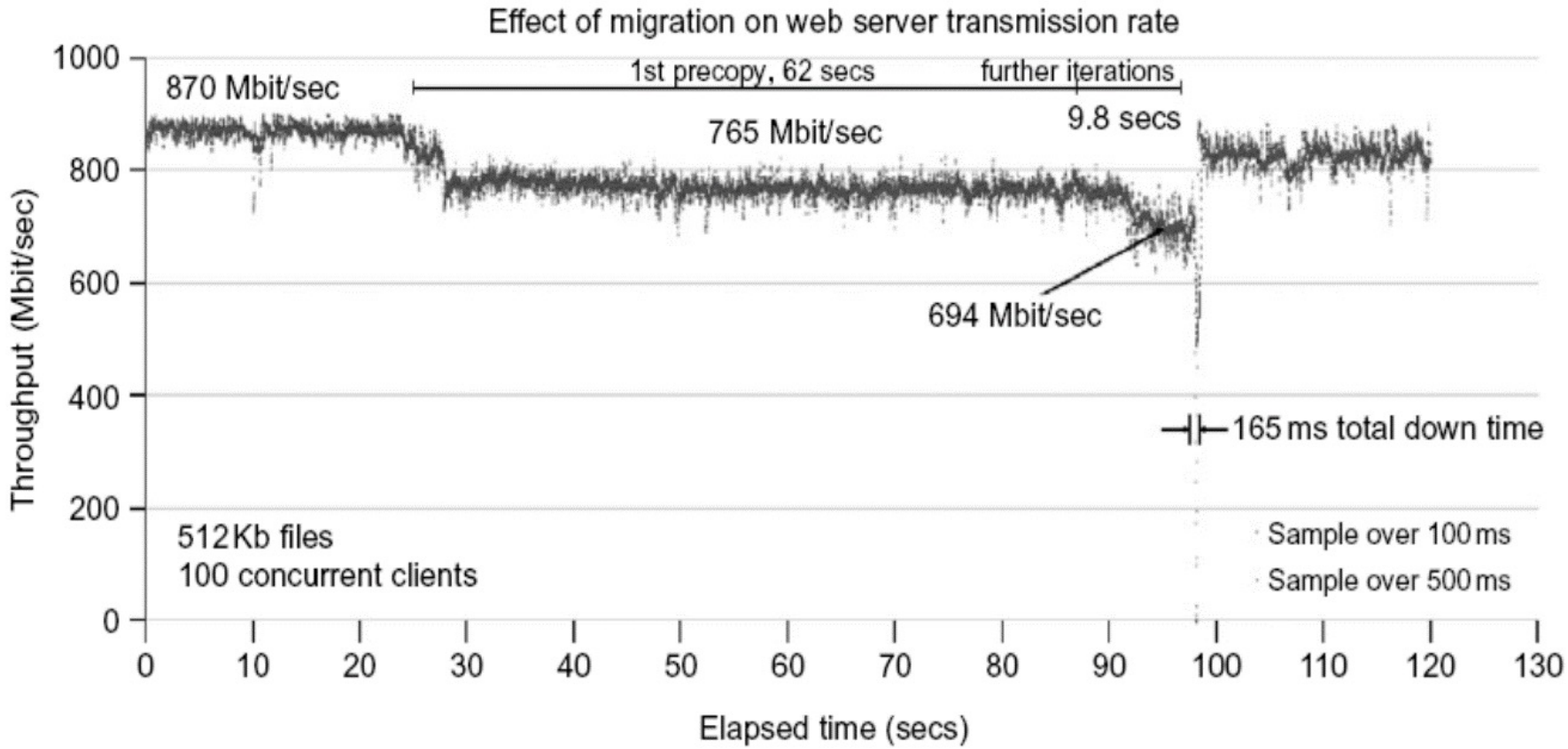


FIGURE 3.21

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

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