Sri Sai Vidya Vikas Shikshana Samithi ®

SAI VIDYA INSTITUTE OF TECHNOLOGY

Approved by AICTE, New Delhi, Affiliated to VTU, Recognized by Govt. of Karnataka
Accredited by NBA
RAJANUKUNTE, BENGALURU 560 064, KARNATAKA
Phone: 080-28468191/96/97/98 ,Email: info@saividya.ac.in, URLwww.saividya.ac.in

Department of Computer Science and Engineering

CLOUD COMPUTING (BCS601)

Module-02

Virtual Machines and Virtualization of Clusters and Data Centers

- Implementation Levels of Virtualization
- Virtualization Structure/Tools and Mechanisms,
- Virtualization of CPU/Memory and I/O devices,
- Virtual Clusters and Resource Management
- Virtualization for Data Center Automation.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

Definition: Virtualization is a computer architecture technology enabling multiple virtual machines (VMs) to operate on the same hardware.

Historical Origin: The concept of VMs dates back to the 1960s.

Purpose of VMs:

- Enhances resource sharing among multiple users.
- Improves performance through efficient resource utilization and application flexibility.
- Virtualizes hardware (CPU, memory, I/O devices) or software (operating systems, libraries) in various layers.

Modern Relevance: Virtualization has gained importance recently due to rising demand for distributed and cloud computing.

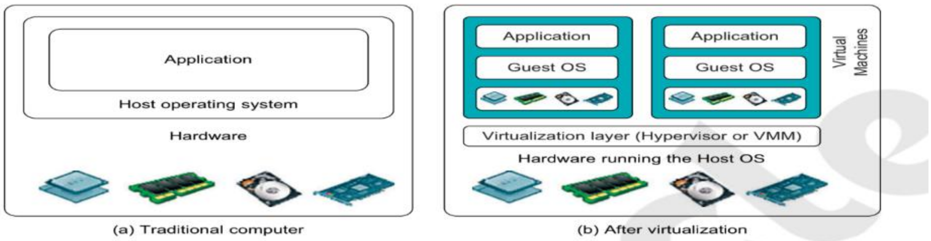
Core Idea: Separates hardware from software for better system efficiency.

Example: Virtual memory expanded users' accessible memory space.

Similar techniques optimize the use of compute engines, networks, and storage.

Levels of Virtualization Implementation

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure 3.1(a).



(a) Traditional computer

(b) After virtualization

FIGURE 3.1

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

Effect of Virtualization:

- Enables different user applications, each with their own operating system (guest OS), to run on the same hardware.
- Allows operation independently of the host OS.

Virtualization Layer:

- Achieved using additional software called a **hypervisor** or **virtual machine monitor (VMM)**.
- The virtualization layer interposes itself at various levels of the computer system.

Virtual Machines (VMs):

- Applications run in VMs with their own guest OS.
- VMs operate over virtualized CPU, memory, and I/O resources.

Common Virtualization Layers:

1. **ISA Level** (Instruction Set Architecture level).
2. **Hardware Level**.
3. **Operating System Level**.
4. **Library Support Level**.
5. **Application Level**.

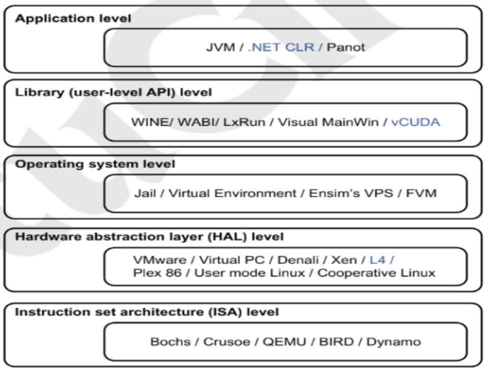

Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo

FIGURE 3.2

Virtualization ranging from hardware to applications in five abstraction levels.

Instruction Set Architecture Level

Virtualization at ISA Level:

- Emulates one ISA using the ISA of a host machine.
- Enables running legacy binary code (e.g., MIPS code on x86 hardware).

Benefits:

- Facilitates running old binary code on new hardware.
- Creates virtual ISAs on any hardware.

Emulation Methods:

1. **Code Interpretation**:
   - Interprets source instructions to target instructions one at a time.
   - Requires multiple native instructions per source instruction.
   - Slower process overall.
2. **Dynamic Binary Translation**:
   - Translates basic blocks of source instructions into target instructions dynamically.
   - Improves performance compared to interpretation.
   - Can extend to program traces or super blocks for better efficiency.

Instruction Set Emulation:

- Relies on binary translation and optimization.
- Requires a virtual ISA (V-ISA).
- Involves adding a processor-specific software translation layer to the compiler.

Hardware Abstraction Level

Definition: Hardware-level virtualization operates directly on top of the bare hardware.

Dual Functionality:

- Creates a virtual hardware environment for virtual machines (VMs).
- Manages the underlying hardware through virtualization.

Objective: Virtualizes computer resources (processors, memory, I/O devices) to enhance hardware utilization for multiple users concurrently.

Historical Development:

- Implemented in the IBM VM/370 system in the 1960s.
- Recently, the **Xen hypervisor** has been used to virtualize x86-based machines for running Linux or other guest OS applications.

Operating System Level

Definition: OS-level virtualization is an abstraction layer between the operating system (OS) and user applications.

Functionality:

- Creates isolated containers on a single physical server.
- OS instances in containers utilize hardware and software resources in data centers.

Behavior: Containers function like real servers.

Common Uses:

1. Virtual Hosting:
   - Allocates hardware resources among multiple, mutually distrusting users.
2. Server Consolidation:
   - Moves services from separate hosts into containers or virtual machines (VMs) on a single server.
   -

Library Support Level

API-Based Virtualization:

- Applications typically use APIs from user-level libraries rather than direct system calls to the OS.
- APIs serve as an interface suitable for virtualization.

Method:

- Virtualization is achieved by controlling communication between applications and the system through API hooks.

Examples:

1. WINE:
   - Supports Windows applications on UNIX systems using API-based virtualization.
2. vCUDA:
   - Enables applications in VMs to utilize GPU hardware acceleration.

User-Application Level

Definition:

- Application-level virtualization virtualizes applications as virtual machines (VMs).
- Applications typically run as processes on traditional operating systems (OS), making this also known as process-level virtualization.

Popular Approach:

- Utilizes high-level language (HLL) VMs.

- Virtualization layer acts as an application program on top of the OS, exporting a VM abstraction.
- Programs written and compiled for a specific abstract machine definition can run on this VM.
- Examples: Microsoft .NET CLR and Java Virtual Machine (JVM).

Other Forms of Application-Level Virtualization:

1. Application Isolation: Wraps applications in a layer isolated from the host OS and other applications.
2. Application Sandboxing: Restricts application interactions for enhanced security.
3. Application Streaming: Streams applications to user devices for on-demand usage.

Benefits:

- Simplifies application distribution and removal from user workstations.
- Example: LANDesk application virtualization platform, which deploys software as self-contained, executable files without requiring installation or system modifications.

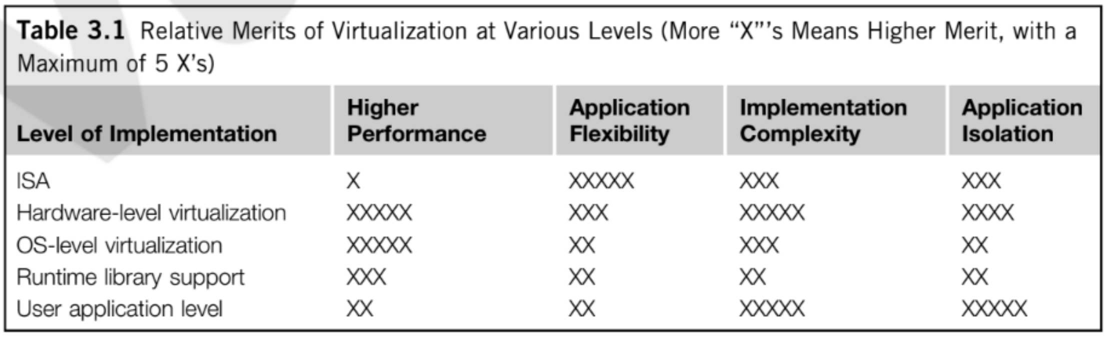


Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

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

VMM Design Requirements and Providers

Definition:

- Hardware-level virtualization inserts a layer, called the Virtual Machine Monitor (VMM), between real hardware and operating systems (OS).

Functionality:

- The VMM manages hardware resources and captures all hardware access processes.
- Acts like a traditional OS, enabling multiple traditional operating systems (same or different) to run simultaneously on the same hardware.

Key Features of a VMM:

1. Environment:
   - Provides an environment for programs that is nearly identical to the original hardware.
2. Performance:
   - Programs should show only minor speed decreases.

3. Control:
   - Maintains complete control of hardware resources.

Exceptions to Identical Functionality:

- Differences due to resource availability when multiple VMs are running.
- Timing dependencies caused by the virtualization layer or concurrent VMs.

Resource Requirements:

- Individual VM resource needs (e.g., memory) are reduced.
- Total resource usage may exceed the hardware's capacity due to multiple VMs.

Efficiency:

- Efficiency is critical; users won't prefer a VMM if performance is too low.
- A subset of virtual processor instructions must execute directly on the hardware to improve efficiency.

Traditional Emulators vs. VMM:

- Emulators and simulators offer flexibility but are too slow for practical use.
- VMMs must execute instructions without excessive software intervention to ensure speed.

Control Over Resources:

1. The VMM allocates hardware resources to programs.
2. Programs cannot access unallocated resources.
3. The VMM can regain control of allocated resources when needed.

Processor Limitations:

- Some processors, like x86, have limitations (e.g., inability to trap privileged instructions) that make VMM implementation challenging.
- Hardware-Assisted Virtualization modifies hardware to meet VMM requirements.

Virtualization Support at the OS Level

Emergence of Cloud Computing:

- Enabled by virtual machine (VM) technology.
- Shifts hardware and staffing costs of managing computational centers to third-party providers, similar to a banking model.

Two Major Challenges:

1. Variable Resource Needs:
   - Tasks may require fluctuating numbers of physical machines and VM instances.
   - Example: A task might need one CPU at some phases but hundreds of CPUs at other times.
2. Slow VM Instantiation:
   - New VMs are either fresh boots or template replicates, unaware of the current application state.
   - Results in delays and inefficiencies.

Call for Improvement:

- Significant research and development are needed to address these challenges and enhance support for cloud computing.

Why OS-Level Virtualization?

Challenges of Hardware-Level Virtualization:

1. Slow initialization due to each VM creating its image from scratch.
2. Storage issues from considerable repeated content among VM images.
3. Disadvantages include:
    - Slow performance.
    - Low density.
    - The need for para-virtualization to modify the guest OS.
4. May require hardware modifications to reduce performance overhead.

OS-Level Virtualization as a Solution:

- Inserts a virtualization layer inside the operating system to partition physical resources.
- Creates multiple isolated virtual machines (VMs) within a single OS kernel.

Characteristics of OS-Level VMs:

- Referred to as **Virtual Execution Environments (VE)**, **Virtual Private Systems (VPS)**, or **containers**.
- VEs behave like real servers with their own:
    - Processes.
    - File systems.
    - User accounts.
    - Network interfaces (IP addresses, routing tables, firewall rules, etc.).
- VEs share the same operating system kernel but can be customized for different users.

Alternate Name:

- Known as **single-OS image virtualization** because all VEs use a single shared OS kernel.

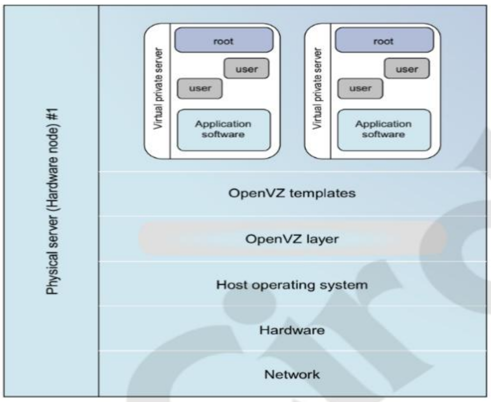


FIGURE 3.3

The OpenVZ virtualization layer inside the host OS, which provides some OS images to create VMs quickly.

Advantages of OS Extensions

Benefits of OS-Level Virtualization Compared to Hardware-Level Virtualization:

1. Minimal **startup/shutdown costs**, low resource requirements, and high scalability.
2. Ability to **synchronize state changes** between a VM and its host environment when needed.

Mechanisms to Achieve Benefits:

1. All OS-level VMs share a **single operating system kernel**.
2. Virtualization layer allows VM processes to access host machine resources without modifying them.

Disadvantages of OS Extensions

Main Disadvantage of OS Extensions:

- All OS-level VMs on a single container must use the same operating system family (e.g., Windows cannot run on a Linux-based container).

Challenge in Cloud Computing:

- Users prefer diverse operating systems (e.g., Windows, Linux), which poses a limitation for OS-level virtualization.

Virtual Root Directories:

- Created using commands like chroot in UNIX systems.
- Methods to implement virtual root directories:
  1. **Duplicating Resources**:
     - Copies common resources to each VM partition.
     - High resource costs and overhead on the physical machine.
  2. **On-Demand Copies**:
     - Shares resources with the host environment and creates private copies only as needed.
     - More resource-efficient.

OS-Level Virtualization Limitations:

- Incurs performance and scalability challenges compared to hardware-assisted virtualization.

Concept of OS-Level Virtualization:

- Inserts a virtualization layer inside the OS to partition hardware resources.
- Allows multiple VMs to run applications in isolated environments.

Implementation Requirements:

- Isolated execution environments (VMs) rely on a single OS kernel.
- Access requests from a VM are redirected to the VM's local resource partition on the physical machine.

Virtualization on Linux or Windows Platforms

Linux Dominance in OS-Level Virtualization:

- Most OS-level virtualization systems are based on Linux.
- Virtualization support for Windows platforms is still under research.

Role of the Linux Kernel:

- Provides an abstraction layer for processes to interact with resources without needing hardware details.
- New hardware may require a new Linux kernel for compatibility.
- Patched kernels enable extended functionality for some Linux platforms.

Simultaneous VMs:

- Linux platforms not tied to a special kernel can run multiple VMs on the same hardware.

Examples of OS-Level Virtualization Tools:

1. Linux vServer and OpenVZ: Enable Linux platforms to run other platform-based applications.
2. FVM: Specifically developed for virtualization on the Windows NT platform.

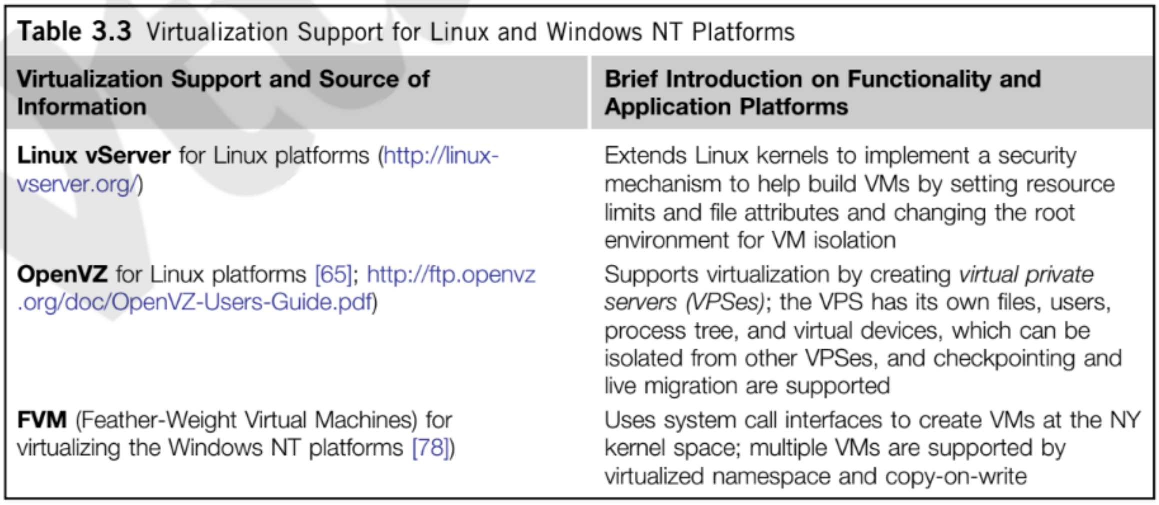


Table 3.3 Virtualization Support for Linux and Windows NT Platforms

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

Middleware Support for Virtualization

Library-Level Virtualization:

- Also called **user-level Application Binary Interface (ABI)** or **API emulation**.
- Creates execution environments for running alien programs on a platform.
- Does not require creating a virtual machine (VM) for an entire operating system.

Key Functions:

1. **API Call Interception**.
2. **API Call Remapping**.

Examples of Library-Level Virtualization Systems:

- **Windows Application Binary Interface (WABI)**.
- **lxrun**.

- **WINE**.
- **Visual MainWin**.
- **vCUDA**.

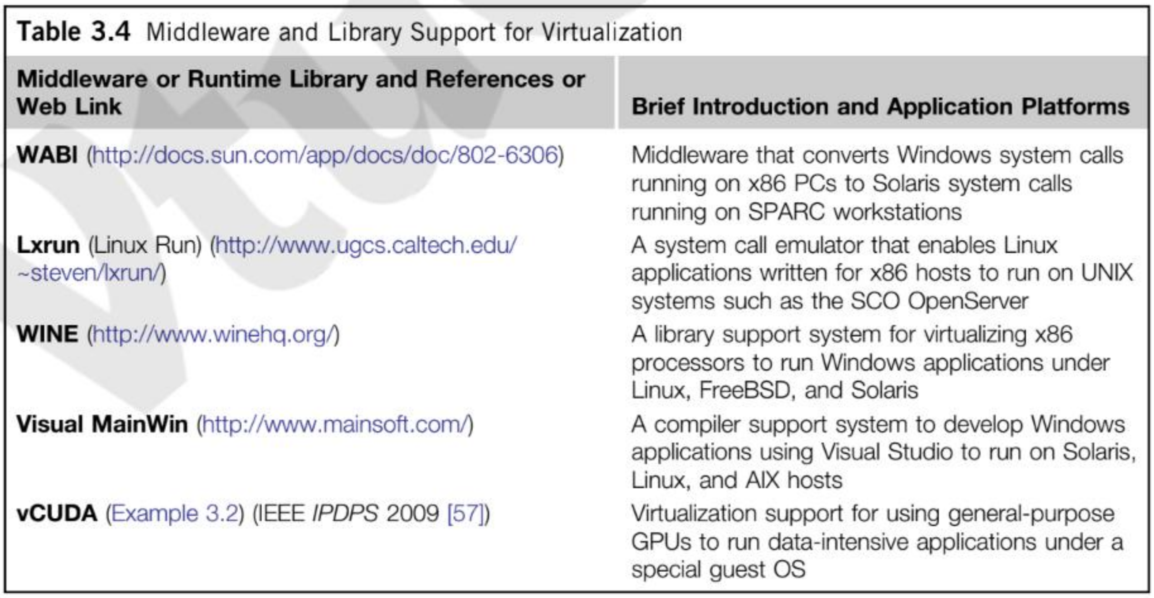


Table 3.4 Middleware and Library Support for Virtualization

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

3.2 VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

Before Virtualization:

- The operating system directly manages the hardware.

After Virtualization:

- A virtualization layer is added between the hardware and the operating system.
- This layer converts real hardware into virtual hardware.
- Enables multiple operating systems (e.g., Linux and Windows) to run on the same physical machine simultaneously.

Classes of VM Architectures:

1. Hypervisor Architecture:
   - Also known as the Virtual Machine Monitor (VMM).
   - Performs virtualization operations.
2. Paravirtualization.
3. Host-Based Virtualization.

Hypervisor and Xen Architecture

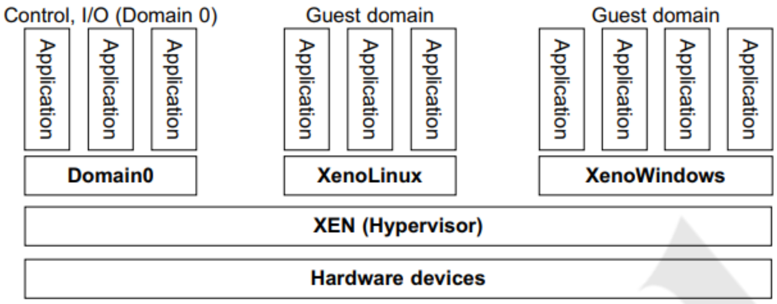


FIGURE 3.5

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

Role of **Hypervisor**:

- Supports hardware-level virtualization on bare metal devices (e.g., CPU, memory, disk, network interfaces).
- Acts as a virtualization layer between physical hardware and the operating system (OS).
- Provides **hypercalls** for guest OSes and applications.

Types of Hypervisor Architectures:

1. **Micro-Kernel Hypervisor**:
   - Example: **Microsoft Hyper-V**.
   - Includes only basic functions (e.g., memory management, processor scheduling).
   - Device drivers and changeable components remain outside the hypervisor.
   - Smaller hypervisor code size.
2. **Monolithic Hypervisor**:
   - Example: **VMware ESX**.
   - Implements all functions, including device drivers.
   - Larger hypervisor code size.

Functionality:

- Converts physical devices into virtual resources for VMs.

The Xen Architecture

Overview of Xen:

- Open-source hypervisor developed by Cambridge University.
- A **microkernel hypervisor** separating mechanism (handled by Xen) from policy (handled by Domain 0).
- Does not include native device drivers; provides mechanisms for guest OSes to access physical devices.

Key Features:

- Small hypervisor size due to minimal functionality.

- Acts as a virtual environment between hardware and OS.
- Commercial versions include **Citrix XenServer** and **Oracle VM**.

Core Components:

1. **Hypervisor**.
2. **Kernel**.
3. **Applications**.

Domains in Xen:

- **Domain 0** (privileged guest OS):
  - Manages hardware access and devices.
  - Allocates and maps resources for other domains (Domain U).
  - Boots first, without file system drivers.
  - Security risks exist if Domain 0 is compromised.
- **Domain U** (unprivileged guest OS): Runs on resources allocated by Domain 0.

Security:

- Xen is Linux-based with a **C2 security level**.
- Strong security policies are needed to protect Domain 0.

VM Capabilities:

- Domain 0 acts as a VMM, enabling users to create, save, modify, share, migrate, and roll back VMs like files.
- Rolling back or rerunning VMs allows fixing errors or redistributing content.

VM Execution Model:

- Traditional machine states are linear, while VM states form a **tree structure**:
  - Multiple instances of a VM can exist simultaneously.
  - VMs can roll back to previous states or rerun from saved points.

Binary Translation with Full Virtualization

Hardware Virtualization Categories:

1. **Full Virtualization:**
   - Does not require modification of the host OS.
   - Relies on binary translation to virtualize sensitive, nonvirtualizable instructions.
   - Guest OSes consist of both critical and noncritical instructions.
2. **Host-Based Virtualization:**
   - Utilizes both a host OS and a guest OS.
   - Includes a virtualization software layer between the host OS and guest OS.

Full Virtualization:

- **Noncritical Instructions**: Run directly on the hardware.
- **Critical Instructions**: Trapped and replaced with software-based emulation by the Virtual Machine Monitor (VMM).

Why Only Critical Instructions Are Trapped:

- **Performance**:
  - Binary translation of all instructions can lead to high performance overhead.
- **Security**:

  - Critical instructions control hardware and can pose security risks.
  - Trapping them ensures system security.
- **Efficiency**:
  - Running noncritical instructions on hardware improves overall efficiency.

Binary Translation of Guest OS Requests Using a VMM

Implementation:

- VMware and other companies implement full virtualization.
- The **Virtual Machine Monitor (VMM)** operates at **Ring 0**, while the guest OS operates at **Ring 1**.

Functionality:

- The VMM scans the instruction stream to identify **privileged, control-, and behavior-sensitive instructions**.
- These instructions are trapped and emulated by the VMM using **binary translation**.
- Full virtualization combines **binary translation** and **direct execution**.
- The guest OS is completely **decoupled** from the hardware and is unaware of the virtualization.

Challenges with Full Virtualization:

- **Binary Translation** is time-consuming, impacting performance.
- **I/O-Intensive Applications**: Virtualizing such workloads poses significant challenges.
- Performance is improved using a **code cache** for translated instructions, but this increases memory usage.

Performance Efficiency:

- On the x86 architecture, performance typically ranges from **80% to 97%** of the host machine's performance.

Host-Based Virtualization

Host-Based VM Architecture:

- A virtualization layer is installed on top of the host OS.
- The host OS remains responsible for managing hardware.
- Guest OSes run on top of the virtualization layer.
- Dedicated applications can run in VMs, while other applications may run directly on the host OS.

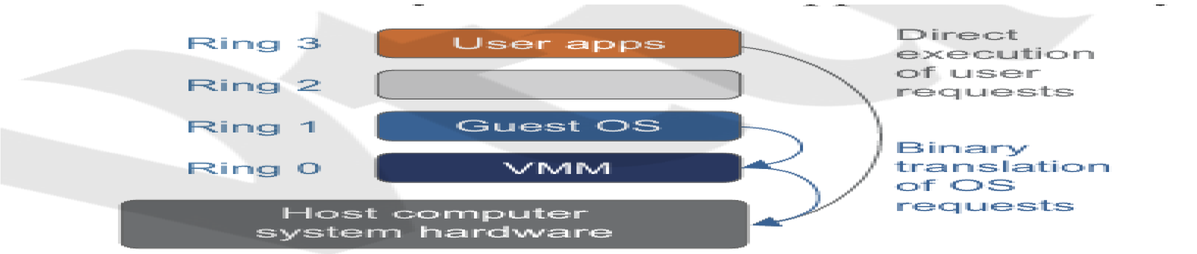

Ring 3
Ring 2
Ring 1
Ring 0
User apps
Guest OS
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

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.

Advantages:

1. Easy installation without modifying the host OS.
2. Simplifies VM design and deployment by relying on the host OS for device drivers and low-level services.
3. Compatible with a wide range of host machine configurations.

Disadvantages:

- Lower performance compared to hypervisor/VMM architecture.
- Involves **four layers of mapping** when applications request hardware access, significantly impacting performance.
- Requires **binary translation** when the ISA of the guest OS differs from the underlying hardware ISA, further decreasing efficiency.

Tradeoff:

- While flexible, host-based architectures may suffer from poor performance, making them less practical for some use cases.

Para-Virtualization with Compiler Support

Definition:

- Para-virtualization modifies the guest operating systems.
- It provides special APIs, requiring significant changes to the guest OS kernel and user applications.

Purpose:

- Aims to reduce virtualization overhead and improve performance.

Mechanism:

- The virtualization layer replaces nonvirtualizable OS instructions with hypercalls using an intelligent compiler.

x86 Processor Execution Rings:

- Four rings: Rings 0, 1, 2, and 3 (lower ring numbers have higher privileges).
- Ring 0: Privileged instructions executed by the OS.
- Ring 3: User-level applications run here.

Example of Para-Virtualization:

- KVM (Kernel-based Virtual Machine).

3.2.3.1 Para-Virtualization Architecture

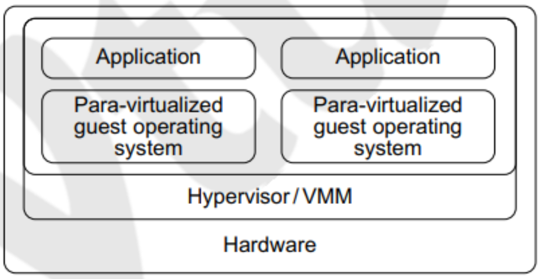

Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

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.)

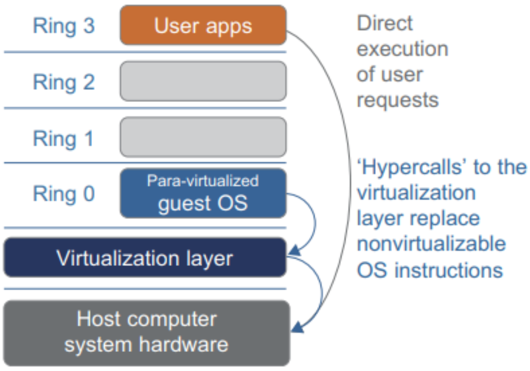

Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

FIGURE 3.8

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

(*Courtesy of VMWare [71]*)

Virtualization of x86 Processor:

- A virtualization layer is inserted between hardware and the operating system (OS).
- The virtualization layer operates at Ring 0 (highest privilege level).

Para-Virtualization Mechanism:

- Replaces nonvirtualizable instructions with hypercalls that interact with the hypervisor or Virtual Machine Monitor (VMM).
- Requires modifications to the guest OS kernel, making it unable to run directly on hardware.

Issues with Para-Virtualization:

1. Compatibility and Portability: Doubts arise as it must support unmodified OSes as well.
2. Maintenance Cost: Maintaining para-virtualized OSes is costly due to deep kernel modifications.
3. Performance Variability: Advantages of para-virtualization vary depending on workloads.

Comparison with Full Virtualization:

- Para-virtualization is easier and more practical.
- Full virtualization suffers from performance issues, especially in binary translation.

Examples:

- Commonly used virtualization products employing para-virtualization include Xen, KVM, and VMware ESX.

KVM (Kernel-Based VM)

KVM (Kernel-based Virtual Machine):

- A Linux para-virtualization system included in the Linux version 2.6.20 kernel.

Functionality:

- Linux Kernel handles memory management and scheduling.
- KVM performs the remaining virtualization tasks, making it simpler than a full hypervisor.

Key Features:

- Hardware-Assisted Para-Virtualization: Enhances performance.
- Supports unmodified guest OSes, including Windows, Linux, Solaris, and other UNIX variants.

Para-Virtualization with Compiler Support

Para-Virtualization Process:

- Handles privileged and sensitive instructions at compile time rather than runtime.
- Modifies the guest OS kernel to replace such instructions with hypercalls that interact with the hypervisor or Virtual Machine Monitor (VMM).

Architecture:

- Example: Xen employs a para-virtualization architecture.
- The guest OS typically runs at Ring 1, not Ring 0, meaning it cannot execute privileged instructions directly.

Privileged Instructions:

- Implemented via hypercalls to the hypervisor.
- The modified guest OS emulates the original OS behavior using these hypercalls.

On UNIX Systems:

- A system call generally involves an interrupt or service routine.
- In Xen, hypercalls use a dedicated service routine to fulfill this role.

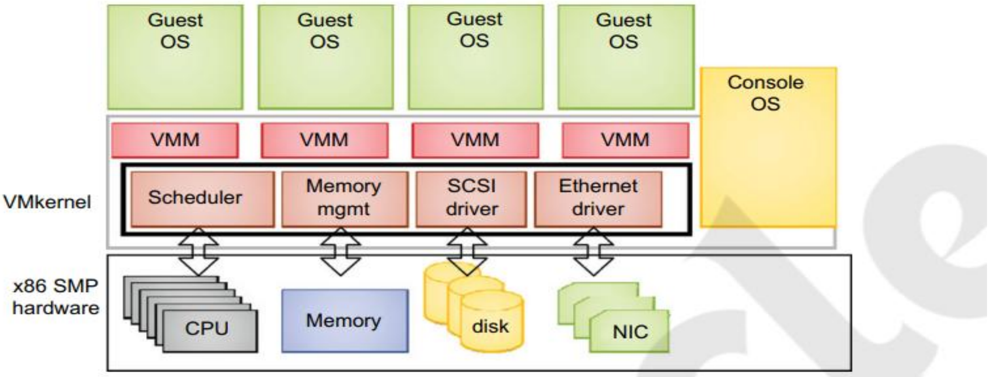


FIGURE 3.9

The VMware ESX server architecture using para-virtualization.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

Processors like x86 use hardware-assisted virtualization, which enables the Virtual Machine Monitor (VMM) and the guest operating system (OS) to operate in separate modes. Sensitive instructions from the guest OS and its applications are intercepted by the VMM. The hardware handles mode switching and saves processor states. Intel and AMD have developed their own technologies for this on the x86 architecture.

Hardware Support for Virtualization

1. **Processor Modes:**
   - Modern processors ensure controlled hardware access using two modes:
     - User Mode: For unprivileged instructions.
     - Supervisor Mode: For privileged instructions.
   - Without protection mechanisms, processes could cause hardware conflicts and crashes.
2. **Virtualized Environments:**
   - Virtualization adds complexity due to extra layers in the machine stack.
3. **Hardware Virtualization Solutions:**
   - VMware Workstation:
     - Software suite for x86 and x86-64 systems.
     - Allows running multiple virtual machines (VMs) alongside the host OS.
     - Uses host-based virtualization.
   - Xen Hypervisor:
     - Runs on IA-32, x86-64, Itanium, and PowerPC 970 hosts.
     - Modifies Linux to act as the hypervisor (privileged layer).
     - Supports multiple guest OSes on top.
   - KVM (Kernel-based Virtual Machine):
     - Linux-based virtualization infrastructure.
     - Supports hardware-assisted virtualization (Intel VT-x, AMD-v) and paravirtualization using the VirtIO framework.
     - VirtIO includes components like:
       - Paravirtual Ethernet card.
       - Disk I/O controller.
       - Memory adjustment (balloon device).
       - VGA graphics interface with VMware drivers.

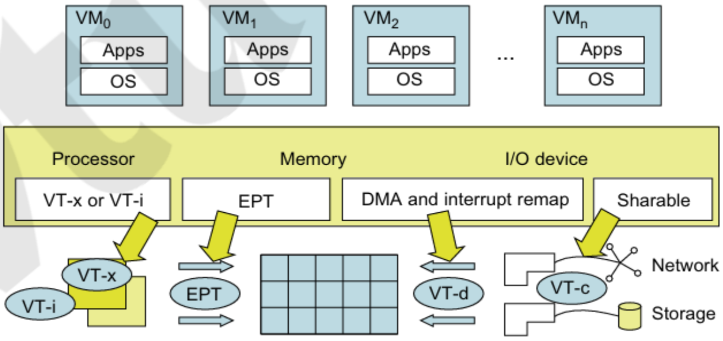


FIGURE 3.10

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

CPU Virtualization

1. **VM Overview:**
   - A Virtual Machine (VM) is a duplicate of a computer system.
   - Most VM instructions run natively on the host processor for efficiency.
   - Critical instructions are managed carefully for system correctness and stability.
2. **Critical Instruction Categories:**
   - Privileged Instructions: Can only execute in a privileged mode and are trapped otherwise.
   - Control-Sensitive Instructions: Change resource configurations.
   - Behavior-Sensitive Instructions: Behave differently depending on resource configurations.
3. **Virtualizable CPU Architecture:**
   - A CPU is virtualizable if VM instructions run in user mode while the Virtual Machine Monitor (VMM) runs in supervisor mode.
   - Privileged instructions are trapped in the VMM, ensuring stability and correctness.
4. **RISC vs. x86 CPU Architectures:**
   - RISC: Naturally virtualizable as all sensitive instructions are privileged.
   - x86: Not fully virtualizable since some sensitive instructions are not privileged (e.g., SGDT, SMSW).
5. **System Calls:**
   - On UNIX-like systems: System calls trigger the 80h interrupt, handled by the OS kernel.
   - On para-virtualized systems (e.g., Xen): The 80h interrupt triggers in the guest OS, while the 82h interrupt is triggered in the hypervisor. The hypervisor processes the call and returns control to the guest OS.
6. **Paravirtualization:**
   - Allows unmodified applications to run in the VM with slight performance penalties.

Hardware-Assisted CPU Virtualization

1. **Challenge in Virtualization:**
   - Full or paravirtualization is complex, prompting efforts to simplify.
2. **Privilege Mode Level (Ring-1):**
   - Intel and AMD introduced a privilege mode level (often called Ring-1) in x86 processors.
   - Operating systems run at Ring 0, and the hypervisor runs at Ring-1.
   - Privileged and sensitive instructions are automatically trapped in the hypervisor**.**
3. **Benefits:**
   - Eliminates the need for binary translation in full virtualization.
   - Allows operating systems to run in VMs without modification.
4. **CPU Enhancements:**
   - Additional instructions are added to manage VM CPU state.
5. **Implementations:**
   - Technologies like VT-x are used by Xen, VMware, and Microsoft Virtual PC hypervisors.
6. **Hardware-Assisted Virtualization:**
   - Offers high efficiency but incurs overhead due to mode-switching between the hypervisor and the guest OS.
   - Sometimes underperforms compared to binary translation.
7. **Hybrid Approach:**
   - Systems like VMware use a hybrid model: some tasks are offloaded to hardware, while others are handled by software.
8. **Combining Techniques:**
   - Paravirtualization and hardware-assisted virtualization are often combined for better performance.

Memory Virtualization

1. **Traditional Memory Mapping:**
   - Operating systems use page tables for one-stage mapping of virtual memory to machine memory.
   - MMU (Memory Management Unit) and TLB (Translation Lookaside Buffer) enhance performance.
2. **Virtual Environment:**
   - Involves two-stage mapping:
     - Virtual memory → Physical memory (Guest OS).
     - Physical memory → Machine memory (VMM).
   - MMU virtualization ensures transparency to the guest OS.
3. **Shadow Page Tables:**
   - VMM creates shadow page tables corresponding to guest OS page tables.
   - Leads to high memory overhead due to duplication.
4. **Nested Page Tables:**

- Additional layer maps physical memory to machine memory.
- Increases complexity and performance costs.

5. **Direct Mapping via TLB:**
   - VMware uses shadow page tables and TLB hardware to map virtual memory directly to machine memory, avoiding two-stage translations on every access.
6. **Hardware Assistance:**
   - AMD Barcelona processor (since 2007) introduced nested paging to support two-stage memory translation, reducing overhead.

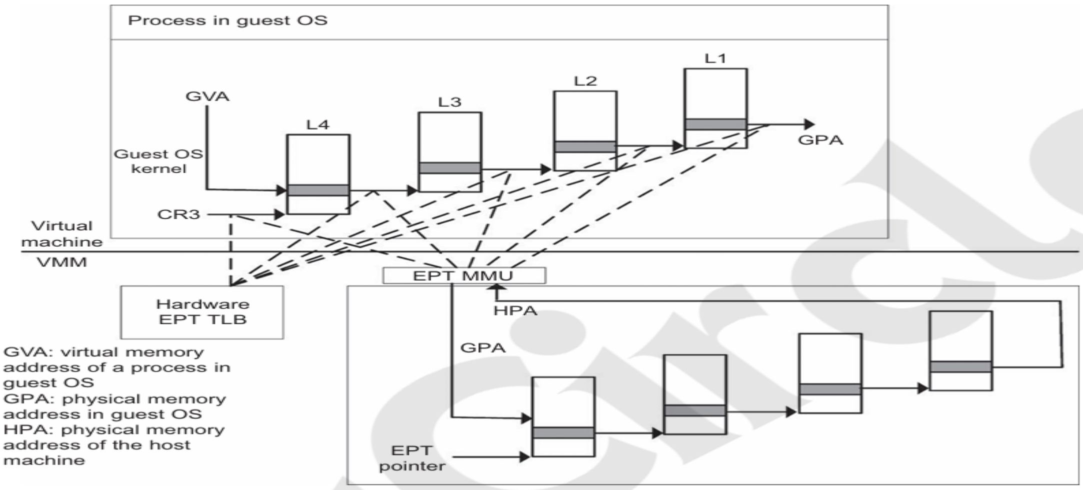

Process in guest OS
L1
L2
L3
L4
GVA
GPA
Guest OS kernel
CR3
Virtual machine
VMM
EPT MMU
Hardware EPT TLB
HPA
GPA
EPT pointer
GVA: virtual memory address of a process in guest OS
GPA: physical memory address in guest OS
HPA: physical memory address of the host machine

FIGURE 3.13

Memory virtualization using EPT by Intel (the EPT is also known as the shadow page table [68]).

I/O Virtualization

1. **Definition:**
   - Manages routing of I/O requests between virtual devices and shared physical hardware.
2. **Methods of Implementation:**
   - **Full Device Emulation:**
     - Emulates real-world devices in software within the VMM.
     - Slower than actual hardware due to software overhead.
   - **Para-Virtualization:**
     - Uses a split driver model (frontend in Domain U, backend in Domain 0).
     - Provides better performance than emulation but with higher CPU overhead.
   - **Direct I/O:**
     - VM accesses devices directly, achieving near-native performance.
     - Challenges exist for commodity hardware, particularly during workload migration.
3. **Hardware-Assisted I/O Virtualization:**
   - Critical to reduce high overhead from device emulation.

   - Intel VT-d supports remapping of DMA transfers and device-generated interrupts.

4. **Self-Virtualized I/O (SV-IO):**
   - Utilizes multicore processors for virtualizing I/O devices.
   - Provides virtual devices with APIs for VMs and the VMM.
   - Each Virtual Interface (VIF) has message queues (incoming and outgoing) and a unique ID.
5. **Challenges:**
   - Full emulation is slow, para-virtualization has higher CPU costs, and direct I/O faces hardware limitations.
   - Proper management is needed for reliability during device reassignment and workload migration.

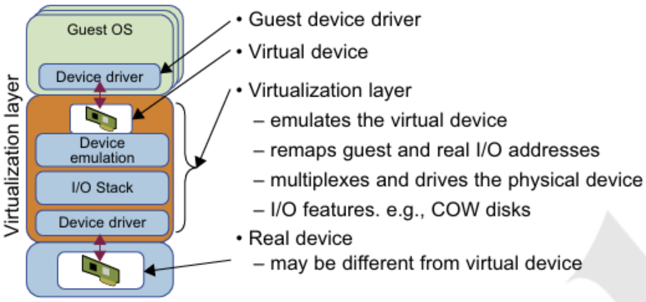

Guest OS
Device driver
Device emulation
I/O Stack
Device driver
Virtualization layer
• Guest device driver
• Virtual device
• Virtualization layer
– emulates the virtual device
– remaps guest and real I/O addresses
– multiplexes and drives the physical device
– I/O features. e.g., COW disks
• Real device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

FIGURE 3.14

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

Virtualization in Multi-Core Processors

Higher Complexity: Virtualizing multi-core processors is inherently more complicated than uni-core ones, despite their higher performance potential.

Challenges in Utilization:

- Parallelization: Programs must be parallelized to fully utilize multi-core processors, requiring better programming models, languages, and libraries.
- Task Assignment: Explicitly assigning tasks to cores is difficult, spurring research in scheduling algorithms and resource management, though finding the right balance between performance and complexity remains elusive.

Emerging Issues:

- Dynamic Heterogeneity: Mixing fat CPU cores with thin GPU cores adds complexity to resource management, stemming from factors like less reliable transistors and the challenges of using them efficiently.

Physical versus Virtual Processor Cores

method proposed by Wells et al. [74]:

- Multicore Virtualization: Provides a higher-level abstraction of processor cores for hardware designers.
- Efficient Resource Management: Reduces inefficiency and complexity in managing hardware resources through software.
- Position in System Architecture: Located beneath the Instruction Set Architecture (ISA) and remains unaffected by the OS or hypervisor (VMM).
- Dynamic VCPU Allocation:
  - A virtual CPU (VCPU) can switch between physical cores as needed.
  - If no suitable core is available, the VCPU's execution is temporarily suspended.

This technique optimizes hardware resource management while maintaining software transparency.

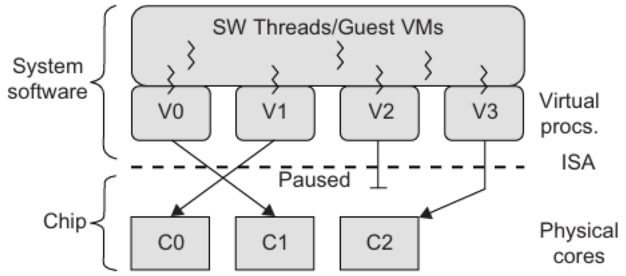


FIGURE 3.16

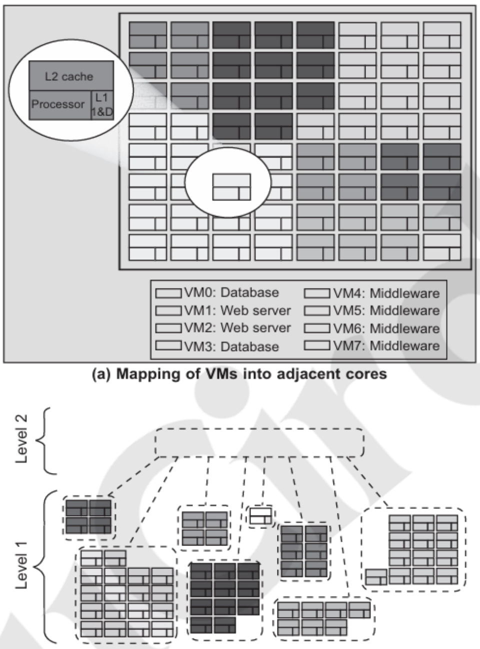
Multicore virtualization method that exposes four VCPUs to the software, when only three cores are actually present.

Virtual Hierarchy

Marty and Hill [39]:

- Many-Core CMPs and Space-Sharing:
  - Emerging many-core CMPs can support space-sharing, where jobs are assigned to separate core groups for long periods.
  - This approach replaces time-sharing on fewer cores.
- Virtual Hierarchies:
  - Introduced to overlay a coherence and caching hierarchy onto physical processors.
  - Can adapt to workloads, unlike fixed physical hierarchies, improving performance and isolation.
- Physical vs. Virtual Cache Hierarchies:
  - Physical hierarchies have static cache allocation and mapping.
  - Virtual hierarchies dynamically adapt to workload needs for:
    1. Faster data access near cores.
    2. Establishing shared-cache domains and points of coherence.
    3. Reducing miss access times.

- Space-Sharing Application:
  - Illustrates workload assignments: e.g., database, web servers, middleware in separate virtual core clusters (VMs).
  - Works for multiple VMs or a single OS environment.
- Two-Level Virtual Hierarchy:
  - Level 1: Isolates workloads/VMs, minimizing performance interference and miss times.
  - Level 2: Maintains globally shared memory for resource repartitioning without costly cache flushes.
- Benefits:
  - Efficient resource use for space-shared workloads (e.g., multiprogramming, server consolidation).
  - Easier system software integration and supports virtualization features like content-based page sharing.



(a) Mapping of VMs into adjacent cores

(b) Multiple virtual clusters assigned to various workloads

FIGURE 3.17

CMP server consolidation by space-sharing of VMs into many cores forming multiple virtual clusters to execute various workloads.

3.4 VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

Physical Cluster:

- Collection of interconnected servers via a physical network like LAN.
- Focuses on clustering techniques for physical machines.

Virtual Clusters:

- Built on virtualized platforms instead of physical servers.
- Enable efficient resource sharing and adaptability for various applications.

Key Design Issues:

1. Live Migration of VMs: Moving virtual machines between physical hosts without downtime.
2. Memory and File Migrations: Ensuring smooth transfer of memory and file systems.
3. Dynamic Deployment: Efficiently allocating and deploying virtual clusters in real-time.

Challenges with Traditional VM Initialization:

- Administrators must manually configure VMs, causing potential overloading or underutilization in networks.

Amazon EC2 Example:

- Provides elastic cloud computing power.
- Allows customers to create and manage virtual machines dynamically.

Bridging Mode:

- Supported by platforms like XenServer and VMware ESX Server.
- Enables VMs to appear as individual hosts on a network.
- Facilitates seamless communication through virtual network interface cards and automated configuration.

Physical versus Virtual Clusters

- **Definition of Virtual Clusters:**
  - Built using Virtual Machines (VMs) across distributed servers in one or more physical clusters.
  - VMs are interconnected through a virtual network spanning multiple physical networks.
  - Virtual clusters have dynamic and distinct boundaries.
- **Key Properties:**
  1. Nodes can be physical or virtual machines.
  2. Multiple VMs, running different operating systems, can coexist on the same physical server.
  3. VMs improve server utilization and flexibility by consolidating multiple functionalities on one server.
  4. VMs can be replicated across multiple servers for fault tolerance, distributed parallelism, and disaster recovery.
  5. Virtual clusters can dynamically scale in size (nodes can be added or removed).
  6. Physical node failure may impact VMs hosted there, but VM failure does not affect the host system.
- **Virtual Cluster Management:**
  - Deployment, monitoring, and managing VMs across large-scale physical nodes are critical for building high-performance environments.

  - Resource scheduling, load balancing, server consolidation, and fault tolerance are essential techniques.
- **Efficiency in VM Image Storage:**
  - Efficient storage of large numbers of VM images is crucial.
  - Preinstalled software packages (templates) allow users to build custom software stacks for specific needs.
- **Host and Guest Systems:**
  - Physical machines are host systems, and VMs are guest systems.
  - Hosts and guests may run different operating systems.
- **Dynamic Boundaries:**
  - Virtual cluster boundaries can change as VMs are added, removed, or migrated.

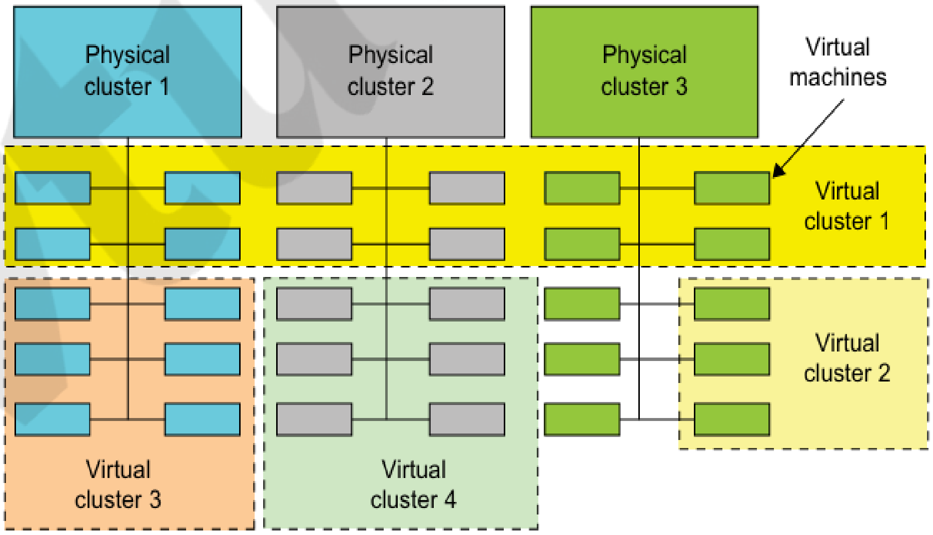


FIGURE 3.18

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

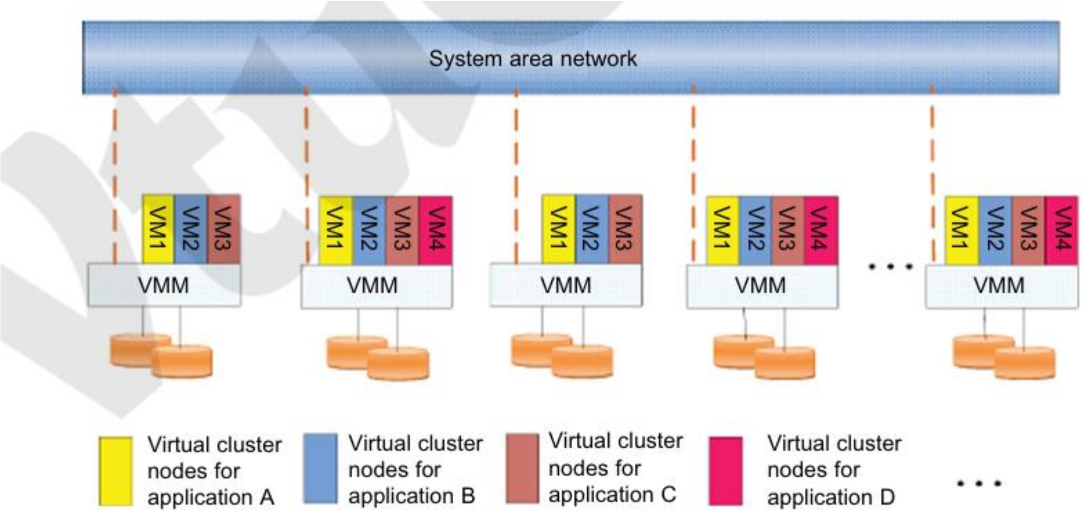

System area network
VM1
VM2
VM3
VM4
VMM
Virtual cluster nodes for application A
Virtual cluster nodes for application B
Virtual cluster nodes for application C
Virtual cluster nodes for application D
• • •

FIGURE 3.19

The concept of a virtual cluster based on application partitioning.

Fast Deployment and Effective Scheduling

- **Fast Deployment Capabilities:**
  1. Construct and distribute software stacks (OS, libraries, applications) rapidly to physical nodes in clusters.
  2. Quickly switch runtime environments between users' virtual clusters, shutting down or suspending unused clusters to save resources.
- **Green Computing Challenges:**
  - Traditional methods focus only on single workstations, not entire clusters.
  - Existing techniques are limited to homogeneous workstations and specific applications.
  - Live migration of VMs enables workload transfers but can introduce significant overhead, negatively impacting cluster utilization, throughput, and quality of service (QoS).
- **Design Challenges for Green Computing:**
  - Develop migration strategies to implement energy-efficient solutions without degrading cluster performance.
- **Load Balancing in Virtual Clusters:**
  - Achieved using load indexes and user login frequency.
  - Automatic scale-up and scale-down mechanisms increase resource utilization and reduce system response time.
  - Mapping VMs to appropriate physical nodes enhances performance.
- **Dynamic Load Adjustment:**
  - Live migration is used to redistribute workloads when cluster nodes are imbalanced.

High-Performance Virtual Storage

Customization of VMs:

- Template VMs can be distributed to physical hosts for customization.
- Existing software packages reduce customization time and ease switching of virtual environments.
- Efficient disk space management is crucial, achieved through reducing duplicate blocks using hash values.

User Profiles:

- Profiles store data block identification for corresponding VMs.
- New blocks created during data modifications are tracked in user profiles.

Steps for Deploying VMs:

1. Prepare the Disk Image: Use templates for easy creation.
2. Configure the VMs: Assign a name, disk image, network settings, CPU, and memory.
3. Choose Destination Nodes: Select appropriate physical hosts.
4. Execute Deployment Commands: Run the commands on every host.

Templates for Deployment:

- Templates include preinstalled operating systems and/or application software.
- Copy-on-Write (COW) format reduces disk space usage and speeds up deployment.

Efficiency in Configuration:

- Use preedited profiles for VMs with the same configurations to simplify the process.
- Automatically assign values for specific items like UUID, VM name, and IP address.

Host Selection Strategy:

- Match VMs to appropriate physical hosts based on requirements.
- Balance workloads across the entire network.

Live VM Migration Steps and Performance Effects

- **Cluster Built with Mixed Nodes:**
  - Physical nodes can failover to VMs on other hosts, ensuring flexibility in failover mechanisms.
  - VM failures won’t affect the host system, but host node failures impact all VMs on it.
  - VM Life Migration mitigates issues by transferring VM state files to new hosts.
- **Virtual Cluster Management Methods:**
  1. Guest-Based Manager: Resides on guest systems (e.g., openMosix, Sun’s cluster Oasis).
  2. Host-Based Manager: Supervises guest systems and restarts failed VMs (e.g., VMware HA system).
  3. Independent Cluster Manager: Handles both host and guest systems but increases complexity.
  4. Integrated Manager: Distinguishes between virtualized and physical resources**.**

  Applications of Virtual Clusters:
  - Found in computational grids, cloud platforms, and HPC systems.

- Useful for dynamic resource provisioning upon demand or after node failures.

Live VM Migration Steps:

1. Start Migration: Identify VM and destination host; migration may be triggered by load balancing.
2. Transfer Memory: Continuously transfer memory, including iterative pre-copying, while minimizing disruptions.
3. Suspend VM: Pause the VM to transfer final memory and non-memory data (e.g., CPU and network states); downtime is minimized.
4. Activate on New Host: Recover states and redirect network connections at the new host; remove VM from the source.

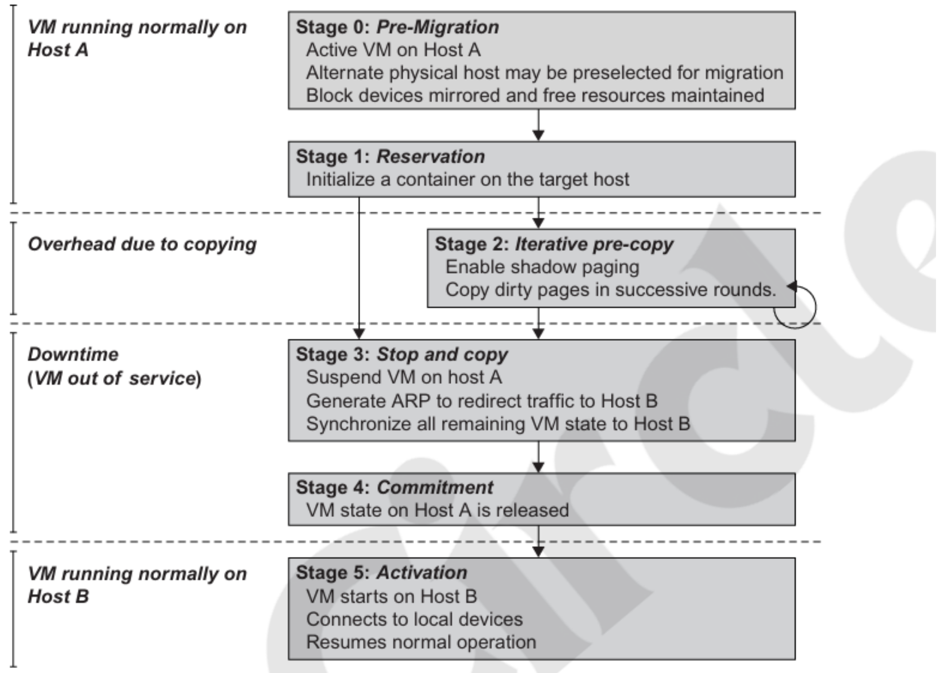

VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative pre-copy
Enable 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.

Trade-Offs in Live Migration:

- Ensure minimal downtime, low network bandwidth usage, and reasonable migration time.
- Avoid resource contention with active services (e.g., CPU, bandwidth).

VM States:

1. Inactive: Not enabled.
2. Active: Performing tasks.
3. Paused: Instantiated but waiting.
4. Suspended: Resources saved back to disk.

Performance Example:

- Live migration resulted in minimal downtime (165 ms) and small throughput impact, critical for dynamic reconfiguration and disaster recovery.

Advantages of VM Technology:

- Simplifies management of physical resources.

  - Enables clustering of inexpensive computers for scalable and reliable computing.

Migration of Memory, Files, and Network Resources

High Initial Costs:

- Clusters require significant investments in space, power conditioning, and cooling equipment.

Shared Clusters:

- Leasing or sharing clusters helps reduce costs and offers better resource utilization through economies of scale.
- Multiplexing enhances resource efficiency by accommodating varying demands.

Early Configuration Systems:

- Focused on scalable and expressive mechanisms to define clusters for specific services.
- Involved physical partitioning of cluster nodes for different service types.

Considerations for System Migration:

- Ensuring seamless transition from one physical node to another.
- Addressing potential performance impacts and resource allocation during migration.

Memory Migration

Memory Migration:

- Involves moving a VM's memory instance from one physical host to another.
- Memory size typically ranges from hundreds of megabytes to a few gigabytes.
- The chosen technique depends on the application/workload supported by the guest OS.

Internet Suspend-Resume (ISR) Technique:

- Exploits temporal locality, as memory states of suspended and resumed VMs are often similar, differing only in recent changes.
- Represents files as a tree of small subfiles to track modifications efficiently.
- Ensures only modified files are transmitted using caching, minimizing unnecessary data transfer.

Limitations of ISR:

- Designed for scenarios where live VM migration is unnecessary.
- Leads to higher downtime compared to other advanced live migration techniques.

File System Migration

File System Consistency:

1. Each VM needs a location-independent file system view across all hosts.
2. One solution is providing each VM with its own virtual disk mapped to the file system, but transferring entire disks over a network is inefficient due to high capacity.

Global File System Approach:

- A global file system across machines eliminates the need to copy files during migration since all files are network-accessible.

Distributed File System in ISR:

- Used as a transport mechanism for suspended VM states.
- Local file systems are accessed by the VMM, which explicitly moves VM files for suspend and resume operations.
- This avoids the complexity of implementing multiple file system calls but requires local storage for VM virtual disks.

Smart Copying:

- Leverages spatial locality (frequent movement between predictable locations like home and office).
- Transmits only changes between file systems at suspend and resume locations, reducing data transfer.
- In cases without locality, much of the state can be synthesized at the resuming site.

Proactive State Transfer:

- Effective when the resuming site can be confidently predicted.
- Focuses on reducing data transfer for operating systems and applications, which constitute the majority of storage space.

Network Migration

Network Connectivity During Migration:

- Migrating VMs must maintain open network connections without relying on forwarding or redirection mechanisms.
- Each VM is assigned a virtual IP and virtual MAC address distinct from the host machine.
- The VMM maps these virtual addresses to the corresponding VM and ensures the protocol states travel with the VM.

Handling IP Reconfiguration:

- On a single switched LAN, an unsolicited ARP reply from the migrating host advertises the VM's new location.
- Alternatively, the VM keeps its original Ethernet MAC address, and the network switch detects the move.

Live Migration Features:

- Moves a VM from one physical node to another while keeping the OS and applications running.
- Enables online system maintenance, load balancing, reconfiguration, and fault tolerance.
- Used in enterprise environments for efficient resource management (e.g., server consolidation, performance isolation).

Precopy Mechanism:

- Memory pages are transferred iteratively during the precopy phase.
- Only dirty pages (modified data) are copied in the final iteration, minimizing service downtime.

- Performance Limitation: Migration daemon uses network bandwidth, causing performance degradation.

Adaptive Rate Limiting:

- Reduces network impact but prolongs the total migration time significantly.
- Limits the number of precopy iterations due to convergence issues.

CR/TR-Motion (Checkpointing/Recovery and Trace/Replay):

- Transfers execution traces instead of dirty pages, reducing migration time and downtime.
- Effective if the log replay rate exceeds the log growth rate.

Postcopy Migration:

- Transfers memory pages only once, reducing total migration time.
- Causes higher downtime due to latency in fetching pages before resuming VM operations.

Memory Compression:

- Uses CPU resources to compress memory pages, significantly reducing transferred data.
- Decompression is simple, fast, and efficient with minimal overhead.

Live Migration of VM Using Xen

- **Xen Hypervisor (VMM):**
    - Allows multiple commodity operating systems to safely share x86 hardware.
    - Uses Domain 0 (Dom0) for managing tasks like creating, terminating, or migrating VMs.
    - Transfers states between VMs using a send/receive model.

  RDMA for Data Communication:
    - Reduces reliance on CPU, caches, or context switches.
    - Minimizes the impact on guest operating systems and hosted applications during migration.

Trade-Offs in Compression:

- A single compression algorithm for all memory data is ineffective.
- Different algorithms are needed to handle varying regularities in memory data.

Live Migration System:

- Migration daemons in Dom0 perform the migration tasks.
- Shadow Page Tables in the VMM layer trace changes to memory pages, flagging them in a dirty bitmap.
- The dirty bitmap is sent to the migration daemon at the start of each precopy round and then reset.

- **Precopy Phase:**
    - Memory pages flagged in the bitmap are extracted and compressed for transfer.
    - At the destination, the data is decompressed, ensuring continuity of the VM state.

This approach aims to balance migration efficiency and minimal disruption to the applications running on VMs.

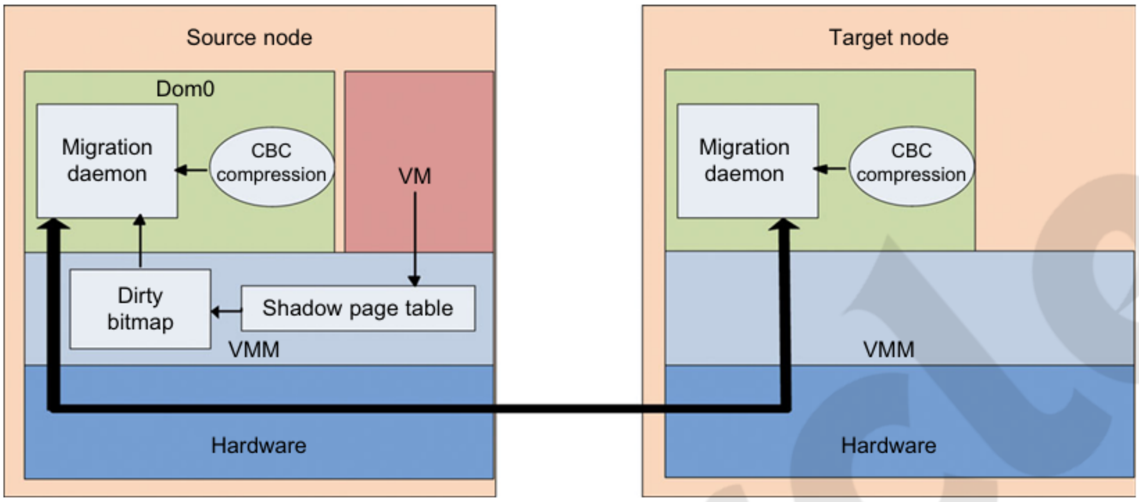


FIGURE 3.22

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

Dynamic Deployment of Virtual Clusters

1. **Cluster-on-Demand (COD) at Duke University:**
   - Objective: Dynamic resource allocation for virtual clusters.
   - Results: Enabled resource sharing and dynamic restructuring of virtual clusters using Sun GridEngine.
2. **Cellular Disco at Stanford University:**
   - Objective: Deploy virtual clusters on shared-memory multiprocessor systems.
   - Results: VMs were deployed on multiple processors under the Cellular Disco VMM.
3. **VIOLIN at Purdue University:**
   - Objective: Enhance resource utilization through dynamic adaptation.
   - Results: Achieved significant adaptation gains with minimal overhead.
4. **GRAAL Project at INRIA, France:**
   - Objective: Test parallel algorithm performance in Xen-enabled virtual clusters.
   - Results: Improved execution times by 75% with 30% resource slack.

Example 3.9: COD Project at Duke University:

- **Purpose:**
  - Manages dynamic server allocation for multiple virtual clusters.
  - Supports dynamic reconfiguration based on load changes.
- **Key Features:**
  - Resource reservation, adaptive provisioning, and idle resource scavenging.
  - Resource policies and templates managed by a configuration database.
- **Results:**
  - Demonstrated dynamic resizing of virtual clusters over time to match user workloads.
  - Enhanced resource utilization in real-life applications.

Example 3.10: VIOLIN Project at Purdue University:

- **Purpose:**
  - Utilizes live VM migration to reconfigure virtual clusters for better resource utilization.
  - Supports isolated virtual environments for parallel applications.
- **Key Features:**
  - Enables relocation and scaling of resources in a transparent manner.
  - Achieves resource adaptation with minimal overhead (20 seconds out of 1,200 seconds for large tasks).
- **Results:**
  - Demonstrated enhanced resource utilization with less than 1% increase in execution time.

3.5 VIRTUALIZATION FOR DATA-CENTER AUTOMATION

- Major IT companies (Google, Amazon, Microsoft, etc.) are heavily investing in data centers.
- Automation enables dynamic allocation of resources (hardware, software, databases) to millions of users while ensuring QoS and cost-effectiveness.

Virtualization Growth (2006–2011):

- Virtualization market share grew significantly (from $1.04 billion in 2006 to $3.2 billion in 2011).
- Key drivers: mobility, reduced downtime, high availability, backup services, workload balancing, and expanding client bases.

Server Consolidation in Data Centers:

1. **Workload Types:**
   - Chatty Workloads: Burst and silent periods (e.g., web video services).
   - Noninteractive Workloads: Continuous processing without user input (e.g., high-performance computing).
2. **Resource Allocation Challenges:**
   - Static resource allocation for peak demand leads to underutilized servers and wasted resources.
   - Server consolidation reduces physical server counts to optimize hardware use.

Advantages of Server Virtualization:

- Enhanced Utilization: Consolidates underutilized servers for higher efficiency.
- Agile Provisioning: OS and application images can be cloned and reused easily.
- Lower Costs: Reduces hardware purchases, maintenance, space, and energy costs.
- Improved Continuity: Guest OS crashes don't affect host OS or other VMs; easier VM transfers between servers.

Automation in Data Centers:

- Requires resource scheduling, power management, and autonomic management techniques.
- Scheduling should span multiple levels (VM, server, and data center), though current techniques handle only one or two levels.

Advanced Resource Management:

- Dynamic CPU Allocation: Based on VM utilization and QoS metrics.
- Two-Level Management:
  - Local controller at VM level.
  - Global controller at the server level.
- CMP (Chip Multiprocessor) Optimization:
  - Design virtual hierarchies for inter-VM sharing, reassignment, and memory access optimization.
  - VM-aware power budgeting to balance performance and energy saving.

This framework addresses the growing complexity and demands of modern data centers.

Virtual Storage Management

1. **Definition:**
   - Traditional storage virtualization: Aggregation and repartitioning of disks at coarse time scales for physical machines.
   - Modern virtual storage: Managed by Virtual Machine Monitors (VMMs) and guest OSes, involving VM images and application data.
2. **Encapsulation and Isolation:**
   - VMs encapsulate operating systems and isolate them from others, allowing multiple VMs to run on a single machine.
   - Storage systems lag in achieving seamless support, becoming a bottleneck in VM deployment.
3. **Storage Challenges:**
   - Guest OS operates as if using a real hard disk but cannot access it directly.
   - Multiple guest OSes contend for hard disk resources, increasing complexity for VMM storage management.
   - Primitive operations like volume remapping and checkpointing are cumbersome and sometimes unavailable.
4. **Key Solutions:**
   - Parallax: A distributed storage system designed for virtualization environments.
   - Content Addressable Storage (CAS): Reduces the size of VM images and supports large-scale VM systems.

Parallax Storage System:

1. **Architecture:**
   - Uses a federation of storage VMs on the same physical hosts as the client VMs.
   - Acts as a block virtualization layer between VMs and physical storage devices.
   - Provides virtual disks (VDIs) accessible transparently across physical hosts.
2. **Features:**
   - Supports paravirtualization and full virtualization techniques.
   - Enables storage functionality (e.g., snapshots) at the software level, reducing dependency on high-end hardware.

  - Allows live upgrades of block device drivers in active clusters.

3. **Implementation:**
   - VDIs are handled via Xen's block tap driver and implemented using a tapdisk library.
   - Storage appliance VMs manage block and network access, connecting to physical hardware.
4. **Scalability:**
   - Suitable for cluster-based environments with a shared administrative domain for storage management.
   - Optimized for dynamic, large-scale data centers.

Parallax demonstrates how innovative storage virtualization solutions can address key bottlenecks in system virtualization, improving scalability and efficiency.

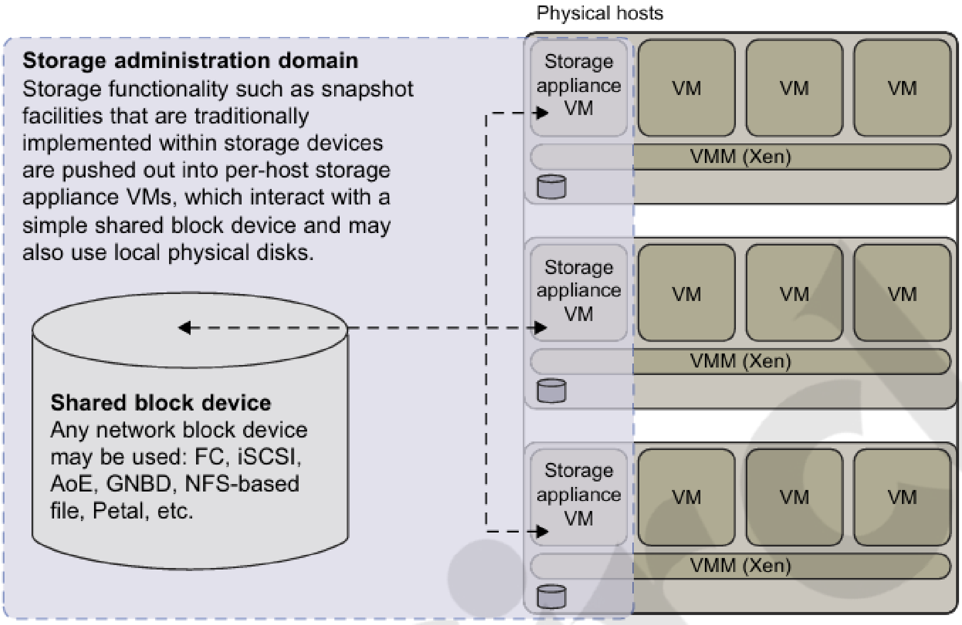

Physical hosts
Storage administration domain
Storage functionality such as snapshot facilities that are traditionally implemented within storage devices are pushed out into per-host storage appliance VMs, which interact with a simple shared block device and may also use local physical disks.
Shared block device
Any network block device may be used: FC, iSCSI, AoE, GNBD, NFS-based file, Petal, etc.
Storage appliance VM
VM
VM
VM
VMM (Xen)
Storage appliance VM
VM
VM
VM
VMM (Xen)
Storage appliance VM
VM
VM
VM
VMM (Xen)

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.

Cloud OS for Virtualized Data Centers

Four VI Managers/OSes:

1. **Nimbus:**
   - Platform: Linux.
   - Features: VM creation, virtual cluster management, and virtual networking.
   - Hypervisors: Xen and KVM.
   - Public Cloud Interface: EC2.
   - License: Apache v2.
2. **Eucalyptus:**
   - Platform: Linux.

- Features: Virtual networking and VM management for private clouds.
- Hypervisors: Xen and KVM.
- Public Cloud Interface: EC2.
- License: BSD.

3. **OpenNebula:**
    - Platform: Linux.
    - Features: Management of VMs, hosts, virtual networks, scheduling tools, and dynamic provisioning.
    - Hypervisors: Xen and KVM.
    - Public Cloud Interface: EC2.
    - License: Apache v2.
4. **vSphere 4:**
    - Platforms: Linux and Windows.
    - Features: Virtual storage, networking, data protection, dynamic provisioning, and scalability.
    - Hypervisors: VMware ESX and ESXi.
    - Public Cloud Interface: VMware vCloud.
    - License: Proprietary.

Example 3.12: Eucalyptus:

- Open-source system designed for IaaS (Infrastructure as a Service) clouds.
- Supports virtual networking, VM management, and interaction with private and public clouds (e.g., AWS EC2 and S3 compatibility).
- Key Components:
    1. Instance Manager: Handles VM execution and termination.
    2. Group Manager: Schedules and manages virtual instance networks.
    3. Cloud Manager: Acts as the main user interface for managing resources and scheduling.

Example 3.13: VMware vSphere 4:

- Commercial cloud operating system designed for data center virtualization and private clouds.
- **Functional Suites:**
    - **Infrastructure Services:**
        - vCompute: ESX, ESXi, and DRS for virtualization.
        - vStorage: VMFS and thin provisioning.
        - vNetwork: Distributed switching and networking functions.
    - **Application Services:**
        - Availability: VMotion, Storage VMotion, High Availability (HA), Fault Tolerance, Data Recovery.
        - Security: vShield Zones and VMsafe.
        - Scalability: DRS (Distributed Resource Scheduler) and Hot Add.
- Interfaces through VMware vCenter to integrate with existing or new applications.

These platforms play critical roles in virtualizing data centers and supporting cloud operations.

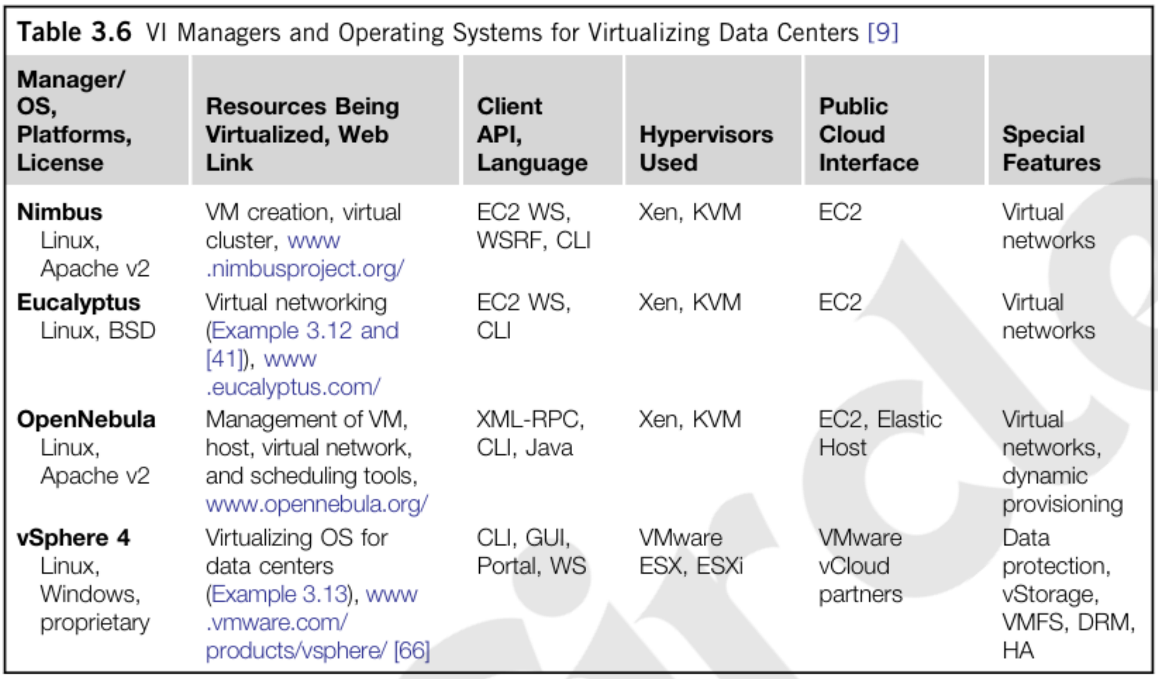


Table 3.6 VI Managers and Operating Systems for Virtualizing Data Centers [9]

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

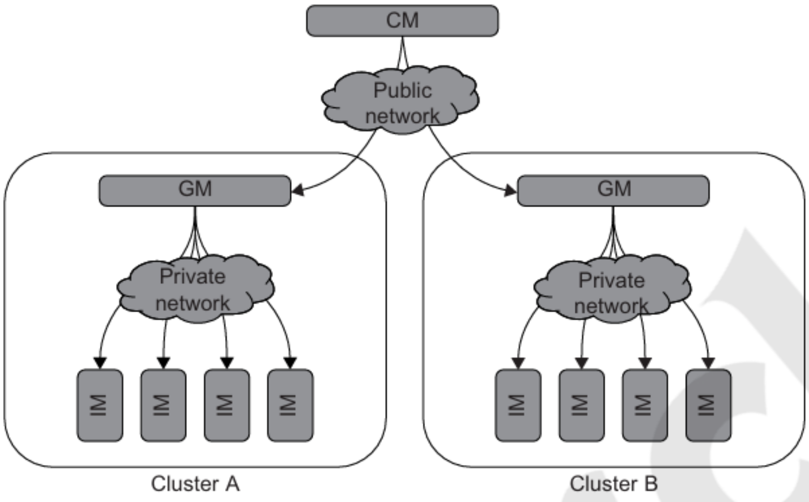


FIGURE 3.27

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

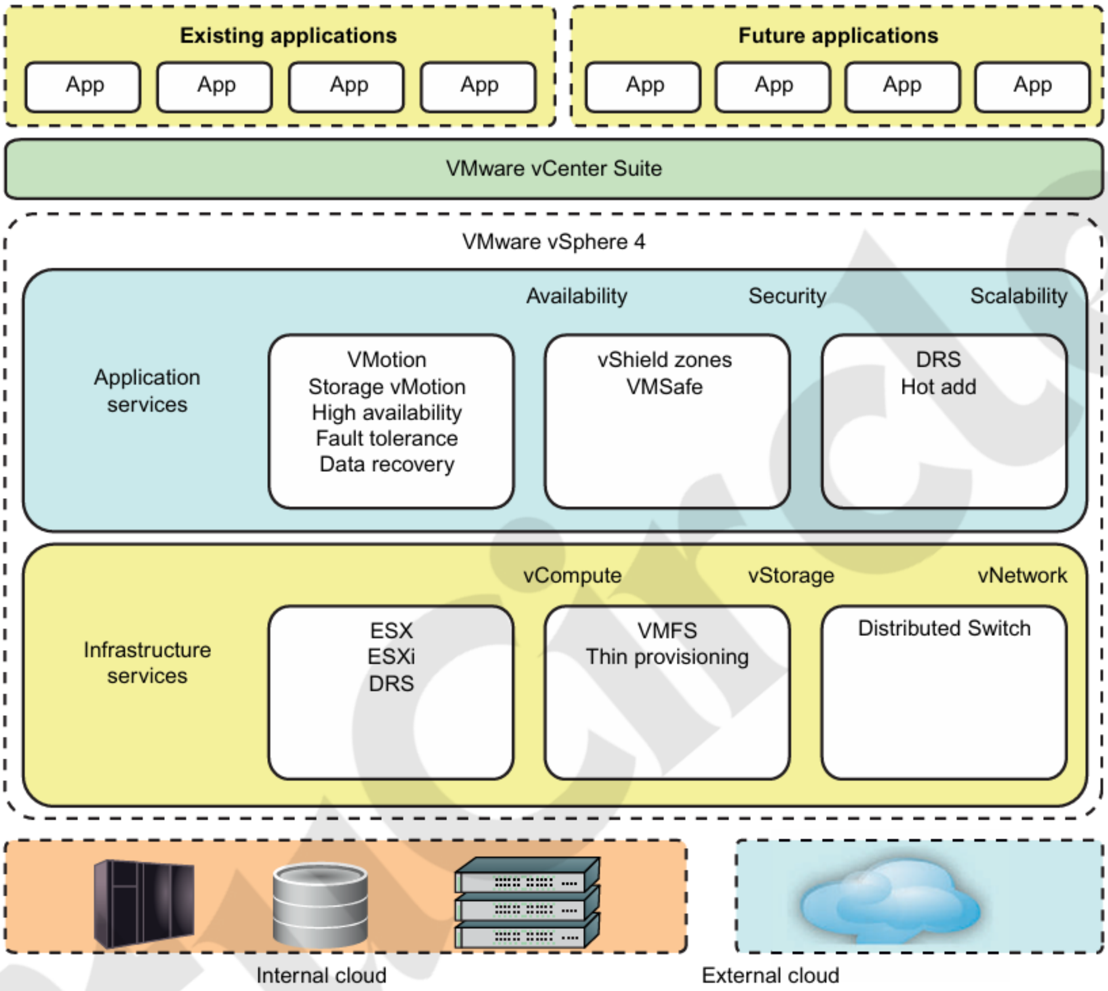


FIGURE 3.28

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

Trust Management in Virtualized Data Centers

1. **Role:**
   - A VMM creates a software layer between operating systems and hardware.
   - It enables one or more Virtual Machines (VMs) on a single physical platform.
   - Provides secure isolation, with hardware resources accessed only through VMM control.
2. **Encapsulation:**
   - Encapsulates VM states, allowing them to be copied, shared, or removed like files.

- Raises security concerns if encapsulated states are exposed or reused (e.g., random number reuse leading to cryptographic vulnerabilities).

3. **Security Risks:**
   - Compromising the VMM or management VM can endanger the entire system.
   - Protocols relying on "freshness" (e.g., session keys, TCP initial sequence numbers) face risks of hijacking or plaintext exposure.

VM-Based Intrusion Detection Systems (IDS):

1. **Types:**
   - Host-Based IDS (HIDS): Monitors specific systems but can be compromised if the host system is attacked.
   - Network-Based IDS (NIDS): Monitors network traffic but struggles with detecting fake actions.
2. **Virtualization-Based IDS:**
   - Isolates guest VMs on shared hardware, limiting the impact of attacks on other VMs.
   - Monitors and audits access to hardware/software, merging advantages of HIDS and NIDS.
   - Implemented as either:
     - A high-privileged VM running on the VMM.
     - Integrated directly into the VMM for higher privileges.
3. **Livewire Architecture:**
   - Contains a policy engine and policy module for monitoring events across guest VMs.
   - Uses tracing mechanisms (e.g., PTrace) for secure policies.

Additional Security Tools:

1. **Honeypots and Honeynets:**
   - Simulate systems to attract and monitor attackers.
   - Protect real systems while analyzing attack actions.
   - Can be implemented as physical or virtual systems.
2. **EMC Trusted Zones:**
   - Built to insulate virtual clusters for multiple tenants.
   - Countermeasures include anti-malware, encryption, intrusion detection, and access segregation.
   - Ensures isolation and compliance for tenant-specific virtual infrastructures.

This framework highlights the challenges and solutions for security and intrusion detection in virtual environments.

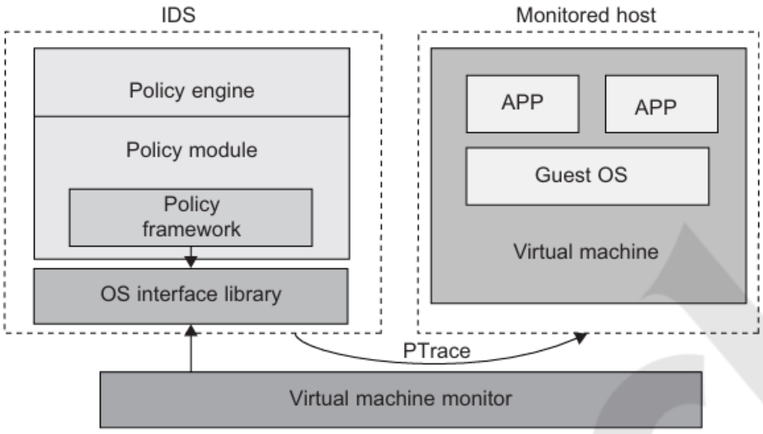


FIGURE 3.29

The architecture of livewire for intrusion detection using a dedicated VM.

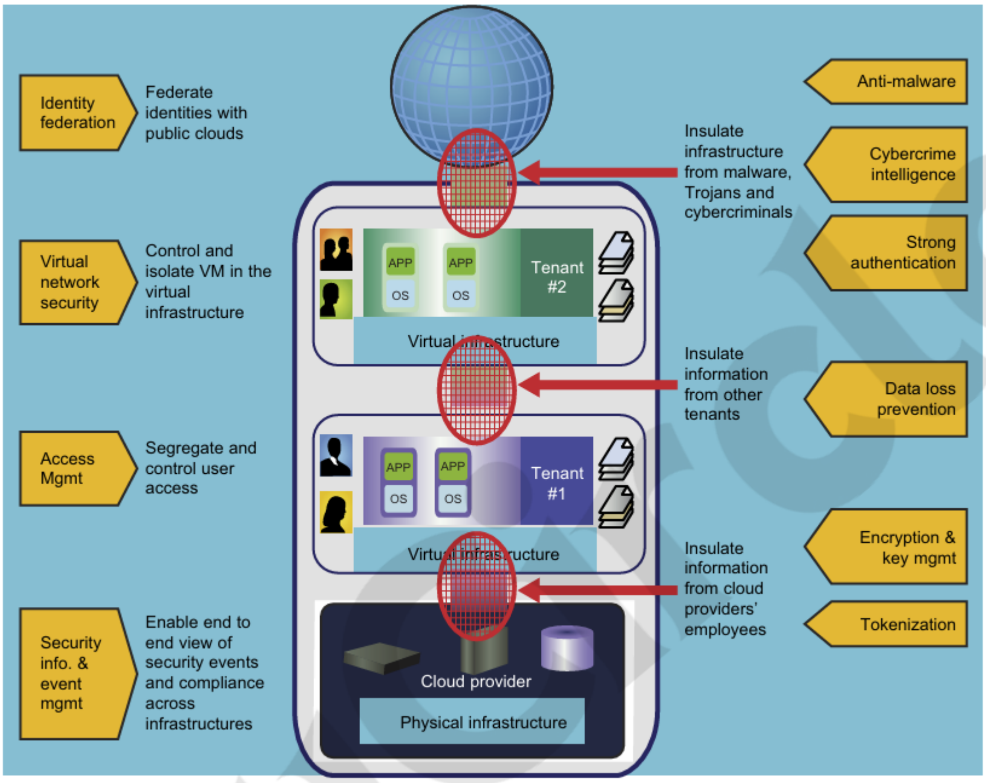

Identity federation
Federate identities with public clouds
Virtual network security
Control and isolate VM in the virtual infrastructure
Access Mgmt
Segregate and control user access
Security info. & event mgmt
Enable end to end view of security events and compliance across infrastructures
APP
OS
Tenant #2
Virtual infrastructure
Tenant #1
Cloud provider
Physical infrastructure
Insulate infrastructure from malware, Trojans and cybercriminals
Insulate information from other tenants
Insulate information from cloud providers' employees
Anti-malware
Cybercrime intelligence
Strong authentication
Data loss prevention
Encryption & key mgmt
Tokenization

FIGURE 3.30

Techniques for establishing trusted zones for virtual cluster insulation and VM isolation.