18.1 Overview

Concept of Virtual Machine:

1. **Abstraction of Hardware:**
   - Virtual machines abstract the hardware of a single computer into multiple execution environments.
   - Each environment appears to run on its own private computer.

Components of Virtual Machine:

1. **Host:**
   - Underlying hardware system running the virtual machines.
2. **Virtual Machine Manager (VMM):**
   - Creates and runs virtual machines.
   - Provides an interface identical to the host.
   - Also known as a hypervisor.
   - Enables multiple operating systems to run concurrently on a single physical machine.
3. **Guest Process:**
   - Provided with a virtual copy of the host.
   - Typically represents an operating system running within a virtual machine.

Implementation Variations:

1. **Hardware-Based Solutions:**
   - Type 0 hypervisors.
   - Provide virtual machine support via firmware.
   - Common in mainframes and large to midsized servers (e.g., IBM LPARs, Oracle LDOMs).
2. **Operating-System-Like Software:**
   - Type 1 hypervisors.
   - Built to provide virtualization (e.g., VMware ESX, Joyent SmartOS, Citrix XenServer).
3. **General-Purpose Operating Systems:**
   - Also known as Type 1 hypervisors (e.g., Microsoft Windows Server with Hyper-V, Red Hat Linux with KVM).
4. **Applications Providing VMM Features:**
   - Type 2 hypervisors (e.g., VMware Workstation, Parallels Desktop, Oracle VirtualBox).
5. **Paravirtualization:**
   - Guest OS modified to work with VMM for performance optimization.

6. **Programming-Environment Virtualization:**
    - VMMs create optimized virtual systems (e.g., Oracle Java, Microsoft .Net).
7. **Emulators:**
    - Allow applications to run on different hardware environments (e.g., different CPU types).
8. **Application Containment:**
    - Segregates applications from the operating system for security and manageability (e.g., Oracle Solaris Zones, BSD Jails, IBM AIX WPARs).

18.3 Benefits and Features

Advantages of Virtualization:

1. **Isolation and Protection:**
    - Host system is protected from virtual machines.
    - Virtual machines are isolated from each other, minimizing protection problems.
    - Viruses in guest operating systems are unlikely to affect the host or other guests.
2. **Resource Sharing:**
    - File-system volume sharing enables file sharing among virtual machines.
    - Virtual communication networks allow information exchange among virtual machines.
3. **Freezing and Snapshotting:**
    - Ability to freeze or suspend a running virtual machine.
    - Copies and snapshots can be made for cloning or restoration purposes.
    - Snapshots record points in time, facilitating rollback if necessary.
4. **System Development and Testing:**
    - Ideal for operating system research and development.
    - Changes to operating systems can be tested in virtual environments without disrupting normal system operation.
    - Multiple versions of programs can be tested concurrently in isolated operating systems.
5. **System Consolidation:**
    - Physical-to-virtual conversions consolidate multiple separate systems into virtual machines on one system.
    - Optimizes resource utilization by combining lightly used systems into one more heavily used system.
6. **Enhanced Management Tools:**
    - Management tools provided by VMM allow efficient management of multiple virtual machines.

- Templating facilitates the creation of multiple VMs from a standard image.
- Patch management, backup, restore, and resource monitoring are streamlined.

7. **Resource Management:**

- Live migration feature enables moving a running guest from one physical server to another without interruption.
- Facilitates resource optimization and maintenance without downtime.

8. **Application Deployment and Management:**

- Simplifies application deployment by preinstalling applications on virtual machines.
- Offers easier application management, tuning, and technical support.
- Standardization of virtual machine formats (e.g., "Open Virtual Machine Format") could further enhance adoption.

18.4 Building Blocks

Difficulty of Implementation:

- Implementing virtual machines requires significant effort, especially on dual-mode systems lacking hardware support for virtualization.

CPU Features and Techniques:

- Virtualization feasibility depends on CPU features.
- Techniques like trap-and-emulate and binary translation are used for virtualization.
- Hardware support is crucial for efficient virtualization.

Virtual CPU (VCPU):

- Represents the state of the CPU as perceived by the guest machine.
- Maintained by the VMM for each guest, facilitating context switching.

Trap-and-Emulate:

- Involves trapping privileged instructions and emulating them in the VMM.
- Provides guest isolation but may introduce performance overhead.

Binary Translation:

- Used for CPUs lacking clean separation of privileged and nonprivileged instructions (e.g., x86 architecture).
- Translates special instructions into equivalent tasks for guest execution.
- Performance optimizations like caching improve execution speed.

Hardware Assistance:

- Crucial for efficient virtualization.
- Modern CPUs provide extended hardware support (e.g., Intel VT-x, AMD-V).
- Enhancements like nested page tables (NPTs) and interrupt remapping improve performance and security.

- ARM architectures implement virtualization support via exception levels and special instructions like HVC.

Thin Hypervisors:

- Enabled by hardware-assisted virtualization.
- Examples include macOS's HyperVisor.framework, allowing lightweight virtual machine management via system calls.

18.5.1 The Virtual Machine Life Cycle:

- Describes the creation and deletion process of virtual machines, including setting parameters like CPUs, memory, networking details, and storage.
- Explains that resources may be dedicated or virtualized, depending on the hypervisor type.

18.5.2 Type 0 Hypervisor:

- Discusses type 0 hypervisors, which are hardware features encoded in firmware.
- Highlights their ability to split a system into multiple virtual systems with dedicated resources.
- Mentions challenges with I/O management when there are more guests than available I/O devices.

18.5.3 Type 1 Hypervisor:

- Covers type 1 hypervisors, which run directly on hardware and manage guest operating systems.
- Emphasizes their role in data centers and their ability to improve performance and control.
- Notes that they may require learning new management tools and methods.

18.5.4 Type 2 Hypervisor:

- Discusses type 2 hypervisors, which run as processes on a host operating system.
- Mentions their flexibility in running multiple operating systems simultaneously without replacing the host OS.
- Notes that they tend to have poorer performance compared to type 0 or type 1 hypervisors.

18.5.5 Paravirtualization:

- Explains paravirtualization as a method where guest operating systems are modified to run on paravirtualized virtual hardware.
- Highlights benefits such as more efficient resource usage and a smaller virtualization layer.

18.5.6 Programming-Environment Virtualization:

- Discusses virtualization of programming environments, such as Java running on the Java Virtual Machine (JVM), providing platform independence.

18.5.7 Emulation:

- Covers emulation as a method for running applications designed for one operating system on a different one, especially when they need to run on a different CPU architecture.

18.5.8 Application Containment:

- Describes application containment as a method to segregate and manage applications within the same operating system, using examples like Solaris containers and Linux LXC containers.

Each subsection provides insights into different virtualization techniques and their implementations.

18.6 Virtualization and Operating-System Components

18.6.1 CPU Scheduling:

- Discusses how virtualization affects CPU scheduling.
- Explains that virtualization software presents virtual CPUs to each virtual machine and schedules physical CPUs among them.
- Details scenarios where there may be enough or not enough CPUs for guest operating systems.
- Highlights challenges with scheduling algorithms and time-sharing operating systems in virtualized environments.
- Mentions the need for clock correction applications to address time discrepancies.

18.6.2 Memory Management:

- Discusses the challenges of memory management in virtualized environments.
- Describes memory optimization methods used by VMMs, such as double paging, balloon memory management, and page sharing.
- Explains how these methods enable guests to perform as if they had the full amount of memory requested.

18.6.3 I/O:

- Explores I/O mechanisms in virtualized environments.
- Discusses the flexibility of hypervisors in providing I/O to guests, including dedicated devices, virtualized devices, and shared access.
- Highlights the importance of VMM design and implementation for efficient I/O.

18.6.4 Storage Management:

- Examines storage management approaches in virtualized environments.
- Discusses how hypervisor types handle storage, such as root disk partitioning for type 0 hypervisors and disk image files for type 1 and type 2 hypervisors.
- Mentions methods for capturing physical systems as guests and vice versa.

18.6.5 Live Migration:

- Introduces the concept of live migration in hypervisors.

- Explains the process of live migration, where a running guest is transferred between systems with minimal interruption.
- Discusses the limitations of live migration, such as the inability to transfer disk state, and the reliance on network-based storage for disk access continuity.
- Highlights the benefits of live migration for resource management and hardware administration in data centers.

Each subsection provides detailed insights into how virtualization affects different aspects of operating system components, such as CPU scheduling, memory management, I/O, storage, and live migration