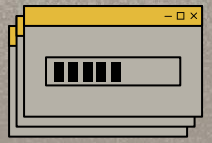
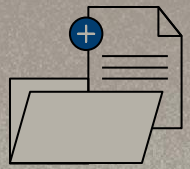


OSG202 - GROUP 6

CHAPTER 7 :

VIRTUALIZATION & THE CLOUDS



GROUP MEMBERS

01 NGUYỄN VŨ HIẾU
SE184611

02 LÝ VĂN MỸ
SE172543

03 TRƯƠNG CHÍ KIỆT
SE172544

04 ĐINH GIA BẢO
SE183741

05 HUỲNH HOÀNG TIẾN
SE172559

7.1 History

7.2 Requirements for Virtualization

7.4 Techniques for Efficient Virtualization

What is Virtualization ?

- Virtualization is the process of creating a virtual version of hardware, software, or an entire computer system.
- It allows multiple virtual machines to run on a single physical computer, enabling better resource utilization and greater flexibility.
- Virtualization is widely used in data centers, cloud computing, and software development environments.

7.1 History

- The concept of virtualization dates back to the 1960s when mainframe computer systems were first developed. IBM introduced virtualization technology with the CP-40 and subsequently the CP-67 operating systems.
- However, it was not until the 1990s and early 2000s that virtualization became more popular with the emergence of x86 servers and software solutions like VMware.
- Today, virtualization is a fundamental technology in modern computing environments.

7.2 Requirements for Virtualization

1. Hardware
2. Hypervisor
3. Operating System
4. Networking
5. Storage
6. Management Tools
7. Security
8. Backup and Disaster Recovery

7.2 Requirements for Virtualization

1. Hardware

- Adequate server-class hardware with sufficient processing power, memory (RAM), and storage capacity to host multiple virtual machines (VMs) simultaneously.
- Support for hardware virtualization extensions, such as Intel VT-x or AMD-V, which enhance virtualization performance.

2. Hypervisor

- A hypervisor is the software layer that enables virtualization. You need to choose a suitable hypervisor based on your needs, such as VMware vSphere, Microsoft Hyper-V, or open-source options like KVM or Xen.

3. Operating System

- The host system running the hypervisor requires an operating system (OS). The choice of OS depends on the hypervisor you select. For example, VMware vSphere uses ESXi as the underlying OS.

4. Storage

- Adequate storage resources are necessary to store VM images, snapshots, and virtual disks. This can be achieved using local storage, network-attached storage (NAS), or storage area networks (SAN).

5. Management Tools

- Virtualization management tools, such as VMware vCenter, Microsoft System Center Virtual Machine Manager (SCVMM), or open-source options like oVirt or Proxmox VE, are crucial for centralized management and monitoring of virtualized resources.

6. Security

- Implementing appropriate security measures is essential to protect the virtual infrastructure. This includes securing the host OS, hypervisor, VMs, and network communication between them.

7. Networking

- A robust network infrastructure is essential for virtualization. It should provide sufficient bandwidth, low latency, and support for features like VLANs, virtual switches, and network load balancing.

- Network adapters in the physical host should have appropriate drivers and support for virtualization technologies like SR-IOV (Single Root I/O Virtualization) if needed.

8. Backup & Disaster Recovery

- Establishing a proper backup strategy and disaster recovery plan is crucial. Regularly backing up VMs, configuring replication, and ensuring offsite backups are important to mitigate the risk of data loss.

7.4 Techniques for Efficient Virtualization

1. Hardware-assisted Virtualization
2. Para-virtualization
3. Full Virtualization with Binary Translation
4. Containerization
5. Dynamic Resource Allocation
6. Memory Overcommitment
7. Storage Virtualization
8. Live Migration
9. Network Virtualization
10. Monitoring and Performance Optimization

1. Hardware-assisted Virtualization:

- Hardware virtualization extensions, such as Intel VT-x or AMD-V, allow the hypervisor to offload virtualization tasks to the underlying hardware. This improves performance and reduces overhead.

2. Para-virtualization:

- In para-virtualization, the guest operating system is modified to be aware of the virtualization environment. This enables direct communication between the guest OS and the hypervisor, eliminating the need for certain virtualization overheads. It can significantly improve performance but requires modifications to the guest OS

3. Full Virtualization with Binary Translation:

- In cases where hardware-assisted virtualization is not available, binary translation techniques can be used. The hypervisor translates non-virtualizable instructions in real-time, allowing the execution of non-modified guest operating systems.

4. Containerization:

- Containerization, also known as operating system-level virtualization, provides lightweight virtualization by sharing the host OS kernel across multiple containers. Containers have less overhead compared to traditional VMs and enable efficient resource utilization.

5. Dynamic Resource Allocation:

- Virtualization platforms often support dynamic resource allocation. This allows resources such as CPU, memory, and storage to be allocated dynamically based on workload demands. It improves resource utilization and ensures optimal performance.

6. Memory Overcommitment:

- With memory overcommitment, the total memory allocated to running VMs can exceed the physical memory capacity. The hypervisor transparently handles memory paging and swapping to disk, ensuring that the active memory pages are in RAM while less-used pages are swapped out.

7. Storage Virtualization:

- Storage virtualization techniques, such as thin provisioning and deduplication, optimize storage utilization. Thin provisioning allows virtual disks to consume only the actual space they use, while deduplication eliminates redundant data.

8. Live Migration:

- Live migration enables the movement of running VMs from one physical host to another without service interruption. It allows workload balancing, hardware maintenance, and efficient resource utilization.

9. Network Virtualization:

- Network virtualization techniques, like virtual switches and virtual LANs (VLANs), allow the creation of isolated virtual networks within a physical network infrastructure. This provides flexibility, security, and efficient network resource utilization.

10. Monitoring and Performance Optimization:

- Regular monitoring of virtualized resources, performance tuning, and optimization are essential for efficient virtualization. Identifying and resolving performance bottlenecks, adjusting resource allocations, and utilizing performance analysis tools help improve efficiency.

7.3 Type 1&2 Hypervisors

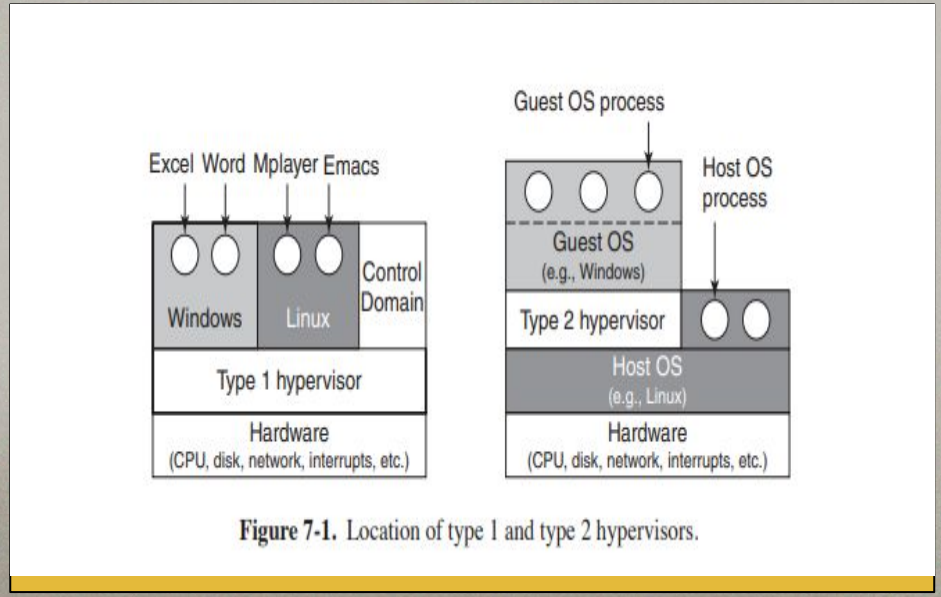


Figure 7-1. Location of type 1 and type 2 hypervisors.

- Type 1 hypervisor (illustrated in Figure 7-1(a)) works similarly to an operating system. It operates in the highest privileged mode and supports multiple virtual machines, which are copies of the actual hardware. It safely executes the computer's set of instructions.

- A type 2 hypervisor (illustrated in Figure 7-1(b)) is a program that relies on an existing operating system (for example, Windows or Linux) to allocate and schedule resources. It pretends to be a full computer with CPU and other devices. The Type 2 Hypervisor still safely executes the computer directive.

The operating system that runs on both types of hypervisor is called the guest operating system. In the case of a type 2 hypervisor, the operating system that runs on the hardware is known as the host operating system.

Type 2 hypervisor, also known as containerized hypervisor, depends on the host operating system (such as Windows, Linux, or OS X) for most of its functionality.

- Type 2 Hypervisor usually requires the host OS to install the guest OS onto the virtual disk. The hypervisor creates a virtual device (for example, a virtual DVD drive) to read the installation program from a virtual storage location (for example, an ISO file). Once installed, the guest OS can be up and running.

7.5 Are Hypervisors Microkernels done right ?

-- Type 1 and type 2 hypervisors both work with the guest operating system unchanged, but must implement complex methods to achieve good performance. Meanwhile, paravirtualization has another method by modifying the source code of the guest OS. Instead of executing sensitive directives, the paravirtualized guest OS executes hypercalls. The actual guest OS acts as a system call user program to invoke the operating systems (hypervisors). When this is done, the hypervisor must define an interface consisting of a set of procedure calls that the guest operating system can use. This set of calls forms an API (Application Programming Interface) that is efficient even though it is an interface for use by guest operating systems, not application programs.

- One step further, by removing all sensitive directives from the operating system and only requiring it to make hypercalls to get system services like I/O, we have turned hypervisor into a microkernel. The idea, explored in paravirtualization, is that emulating specific hardware instructions is an unpleasant and time-consuming task. It requires calling into the hypervisor and simulating the exact semantics of a complex directive. Instead, it's better for the guest OS to call the hypervisor (or microkernel) to perform I/O and similar tasks.
The main reason the first hypervisors emulated the complete machine was the lack of availability of source code for the guest operating system (e.g., for Windows) or the vast number of variants (e.g., for Linux).

The difference between true virtualization and paravirtualization is illustrated in Fig. 7-5

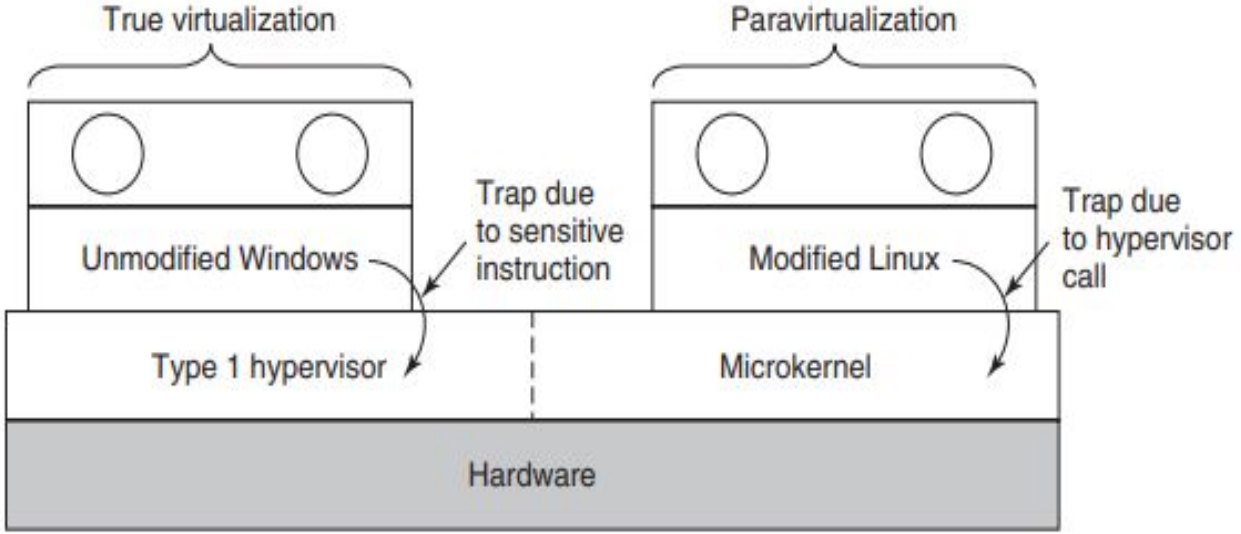


Figure 7-5. True virtualization and paravirtualization

- Paravirtualization poses some problems. First, if sensitive directives are replaced with calls to the hypervisor, how can the operating system run on native hardware? Because the hardware doesn't understand these hypercalls. And second, if there are different hypervisors available in the market, such as VMware, the original open source Xen from Cambridge University and Microsoft's Hyper-V, with slightly different hypervisor APIs, how to fix change the kernel to run on all of them?

- One proposed solution is to use special procedures in the kernel when sensitive operations need to be performed. These procedures, called the Virtual Machine Interface (VMI), form a low-level layer that communicates with the hardware or hypervisor. These procedures are designed to work in common and are not tied to any particular hardware platform or any particular hypervisor.

- There are other suggestions for virtual machine interfaces that have been made. Another popular interface is called paravirt ops. The idea is similar in principle to what has been described above, but differs in detail. A group of Linux vendors including companies such as IBM, VMware, Xen and Red Hat advocated a hypervisor-agnostic interface for Linux.

An example of this technique is given in Fig. 7-6 for a paravirtualized version of Linux they call VMI Linux (VMIL).

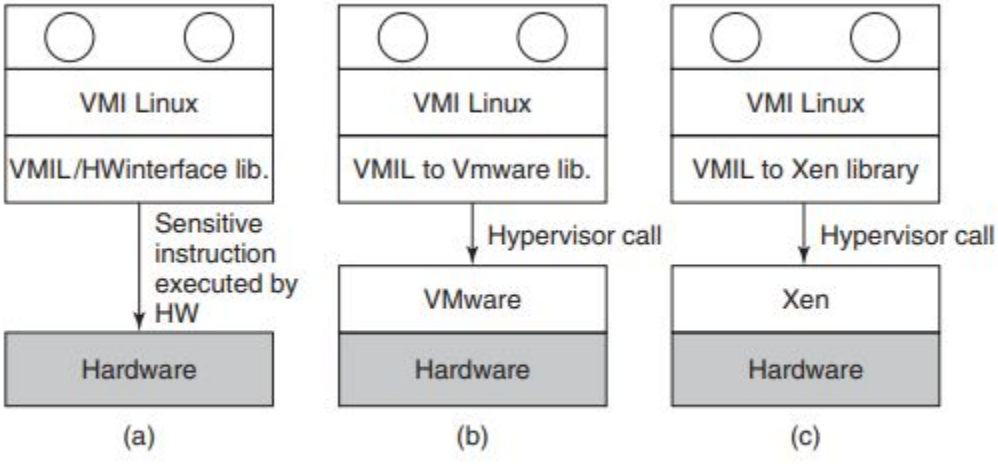


Figure 7-6. VMI Linux running on (a) the bare hardware, (b) VMware, (c) Xen

7.6 Memory Virtualization

- Modern operating systems support virtual memory, which maps virtual address space to physical memory pages using page tables.
- Virtualization complicates memory management, and hardware manufacturers made multiple attempts to address it.
- The hypervisor needs to allocate and manage physical memory pages for each virtual machine and maintain shadow page tables to map virtual pages to physical pages.

Nested Page Tables

- Handling shadow page tables incurs overhead and leads to expensive page faults and VM exits.
- Hardware support for nested page tables (AMD's Nested Page Tables or Intel's Extended Page Tables) reduces this overhead.
- With nested page tables, the CPU can handle intermediate level page-table manipulation in hardware, resulting in fewer VM exits and improved performance.

Reclaiming Memory

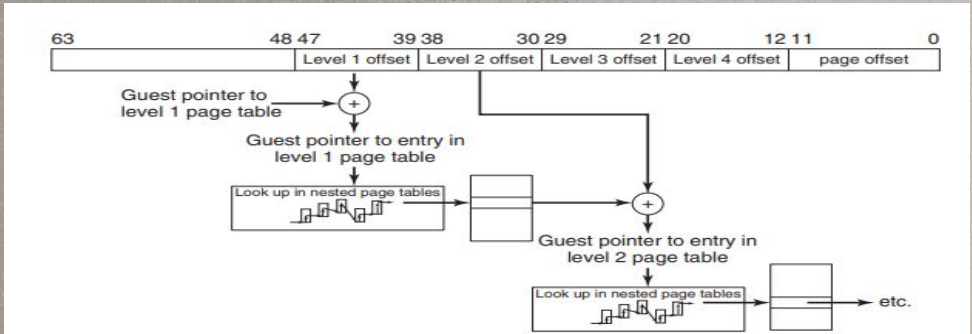


Figure 7-7. Extended/nested page tables are walked every time a guest physical address is accessed—including the accesses for each level of the guest's page tables.

- In scenarios of memory overcommitment, where virtual machines collectively require more memory than available, memory reclamation is necessary.
- The hypervisor cannot determine the most valuable pages to reclaim, and traditional paging methods are inefficient.
- A common solution is "ballooning," where a balloon module in each virtual machine dynamically allocates or deallocates memory to induce the guest operating system to page out less valuable pages.

7.7 I/O Virtualization

I/O MMUs

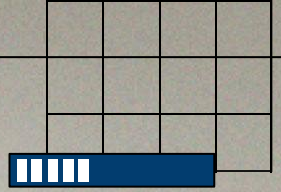
- I/O MMUs virtualize I/O similar to MMUs for memory
- Use page tables to map device addresses to physical addresses
- Ensures device isolation and prevents unauthorized access

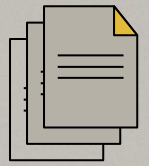
Device Domains

- Dedicate one virtual machine to handle I/O for other virtual machines
- Enhances efficiency and simplifies I/O management
- Paravirtualization improves command execution and communication

Single Root I/O Virtualization (SR-IOV)

- Enhances scalability and performance in virtualized environments
- Provides independent memory, interrupts, and DMA streams to each VM
- Allows virtual machines to believe they have exclusive ownership of devices





08

Virtual Appliances

09

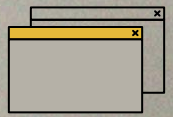
Virtual Machines on Multicore CPUs

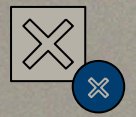
10

Licensing Issues

11

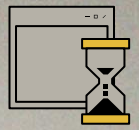
Clouds





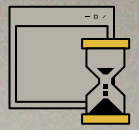
Virtual Appliances

- Virtual appliances offer a solution for installing new application programs, especially in the context of open source software.
- Applications often have dependencies on multiple other applications, libraries, compilers, scripting languages, and operating systems.
- Virtual machines allow developers to create carefully constructed units containing the required components, including the operating system, compilers, libraries, and application code.
- The entire virtual machine, known as a virtual appliance, can be packaged and distributed as a self-contained unit.
- Virtual appliances simplify the installation process for customers as they don't need to worry about individual dependencies.
- Customers can install or download virtual appliances regardless of their operating system or other software installations.
- Virtual appliances are commonly used in cloud environments, such as Amazon's EC2, where pre-packaged virtual appliances are offered as convenient software services.
- Virtual appliances provide a complete and functional package, ensuring that customers receive a working solution.
- They are often referred to as "shrink-wrapped" virtual machines, as they are packaged and ready to run.



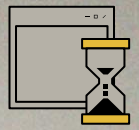
Virtual Machines on Multicore CPUs

- Virtual machines and multicore CPUs allow software to determine the number of available CPUs, enabling flexible configurations.
- Memory sharing in virtual machines is useful for hosting multiple versions of the same operating system on a single server.
- Deduplication techniques can avoid storing duplicate data, improving storage efficiency.
- The combination of virtual machines and multicore CPUs turns a single computer into a virtual multiprocessor.
- The integration of multicore CPUs, virtual machines, hypervisors, and microkernels will bring significant changes to computer systems.
- Current software needs to adapt to the programmer's ability to determine CPU requirements and choose between multicomputer or multiprocessor setups.
- Future software development will need to address these considerations and embrace the evolving landscape.
- Computer science and engineering professionals have an opportunity to contribute to this changing field.



Licensing Issues

- Some software is licensed on a per-CPU basis, granting the right to run the program on a single CPU.
- The definition of a CPU is not explicitly mentioned, leading to uncertainty regarding whether running the software on multiple virtual machines on the same physical machine is allowed.
- Software vendors face challenges in determining the appropriate licensing terms for virtualized environments.
- Companies with licenses allowing a certain number of machines to run the software simultaneously encounter difficulties when virtual machines are created and removed on-demand.
- Some software licenses explicitly prohibit running the software on virtual machines or unauthorized virtual machines, which can be problematic for companies relying solely on virtualized environments.
- The legal enforceability of such license restrictions and how users will respond to them are uncertain and remain to be seen.



Clouds

- Virtualization technology played a crucial role in the rise of cloud computing, which includes public and private clouds offering different resources and services.
- Clouds provide on-demand self-service, allowing users to provision resources automatically without human interaction.
- Resources in the cloud are accessible over the network, enabling heterogeneous devices to utilize them through standard mechanisms.
- Cloud providers pool computing resources to serve multiple users, dynamically assigning and re-assigning resources without users knowing their exact location.
- Cloud resources should have rapid elasticity, allowing users to acquire and release resources to scale immediately according to their demands.
- Measured service is an important aspect where the cloud provider meters the resources used, aligning with the agreed-upon service type.

7.12: Case Study

01 The Early History of VMware

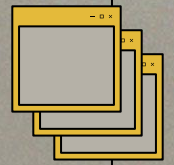
02 VMware Workstation

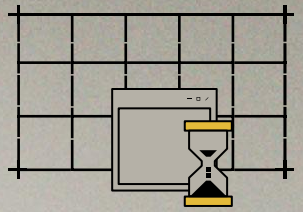
03 Challenges in Bringing Virtualization to the x86

04 VMware Workstation: Solution Overview

05 The Evolution of VMware Workstation

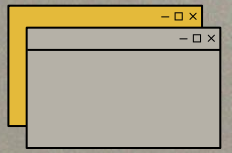
06 ESX Server: VMware's type 1 Hypervisor





Case Study

Since 1999, VMware, Inc. has been the leading commercial supplier of virtualization solutions, offering products for desktops, servers, the cloud, and now even mobile devices. Along with hypervisors, it also provides large-scale virtual machine management software.

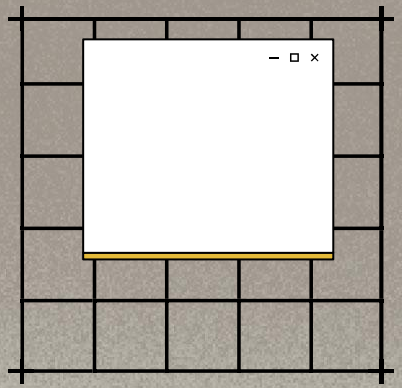


01

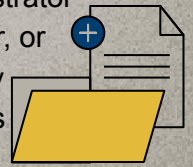
The Early History of VMware

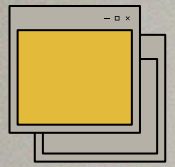
7.12.1: The Early History of VMware

- In order to run open-source operating systems, particularly UNIX, on the FLASH machine, a very large-scale multiprocessor that Stanford was developing at the time, three of the future founders of VMware built a prototype hypervisor called Disco in 1997.
    - Disco's main finding was that, while modern operating systems' high level of complexity made innovation challenging, a virtual machine monitor's relative simplicity and place in the software stack gave it a strong foothold to address operating systems' drawbacks.
    - And so, VMware, Inc. was founded in 1998 with the goal of bringing virtualization to the x86 architecture and the personal computer industry.



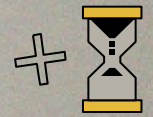
- When the product was first introduced in 1999, it was available in two versions: VMware Workstation for Windows and VMware Workstation for Linux, both of which functioned as type 2 hypervisors that ran on top of host Linux operating systems.
- The first management tool for ESX Server, then known as Virtual Center and now known as vSphere, was released by VMware in 2002. An IT administrator could now easily log into the Virtual Center application to control, monitor, or provision thousands of virtual machines running throughout the company because it provided a single point of management for a cluster of servers running virtual machines.





7.12.2 VMware Workstation

- VMware Workstation was the first virtualization product for 32-bit x86 computers. The subsequent adoption of virtualization had a profound impact on the industry and on the computer science community
- The benefits of virtualization could help address some of the known limitations of the WinTel platform, such as application interoperability, operating system migration, reliability, and security.
- The x86 industry, in contrast, was (and still is) divided into at least four distinct categories: (a) Intel and AMD produce the processors; (b) Microsoft sells Windows and the open source community sells Linux; (c) a third group of businesses creates the I/O devices and peripherals and their corresponding device drivers; and (d) a fourth group of system integrators, including HP and Dell, assembles computers for retail sale.
- VMware Workstation addressed these new challenges by combining well-known virtualization techniques, techniques from other domains, and new techniques into a single solution.



7.12.3 Challenges in Bringing Virtualization to the x86

- ***VMware adopted these core attributes of a virtual machine to an x86-based target platform as follows:***
  - **Compatibility**. The notion of an ‘‘essentially identical environment’’ meant that any x86 operating system, and all of its applications, would be able to run without modifications as a virtual machine.
  - **Performance**. The overhead of the hypervisor had to be sufficiently low that users could use a virtual machine as their primary work environment.
  - **Isolation**. A hypervisor had to guarantee the isolation of the virtual machine without making any assumptions about the software running inside. That is, a hypervisor needed to be in complete control of resources
- ***Four major challenges emerged:***
  - **The x86 architecture was not virtualizable.** It contained virtualization-sensitive, nonprivileged instructions, which violated the Popek and Goldberg criteria for strict virtualization
  - **The x86 architecture was of daunting complexity.** The x86 architecture was a notoriously complicated CISC architecture, including legacy support for multiple decades of backward compatibility
  - **x86 machines had diverse peripherals**. Although there were only two major x86 processor vendors, the personal computers of the time could contain an enormous variety of add-in cards and devices, each with their own vendor-specific device drivers
  - **Need for a simple user experienc**e. Classic hypervisors were installed in the factory, similar to the firmware found in today’s computers

7.12.4 VMware Workstation: Solution Overview

Virtualizing the x86 Architecture:

- The VMM runs the actual virtual machine; it enables it to make forward progress. A VMM built for a virtualizable architecture uses a technique known as trap-and-emulate to execute the virtual machine's instruction sequence directly, but safely, on the hardware
- Figure 7-8 shows the modular building blocks of the original VMware VMM. We see that it consists of a direct-execution subsystem, a binary translation subsystem, and a decision algorithm to determine which subsystem should be used
  - + The virtual machine is currently running in kernel mode (ring 0 in the x86 architecture).
  - + The virtual machine can disable interrupts and issue I/O instructions (in the x86 architecture, when the I/O privilege level is set to the ring level).
  - + The virtual machine is currently running in real mode, a legacy 16-bit execution mode used by the BIOS among other things.
- - The VMware binary translator performs none of these steps in the software. Instead, it configures the hardware so that this simple instruction can be reissued with the identical instruction
- There are, of course, complications and subtleties. One important aspect of the design is to ensure the integrity of the virtualization sandbox, that is, to ensure that no software running inside the virtual machine (including malicious software) can tamper with the VMM

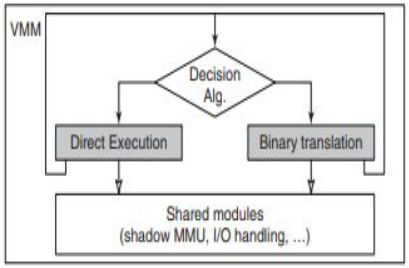


Figure 7-8. High-level components of the VMware virtual machine monitor (in the absence of hardware support).

A Guest Operating System Centric Strategy

- Ideally, a VMM should be designed without worrying about the guest operating system running in the virtual machine, or how that guest operating system configures the hardware
- This simplification did not change the overall design—the VMM still provided a faithful copy of the underlying hardware, but it helped guide the development process
- For example, the x86 architecture contains four privilege rings in protected mode (ring 0 to ring 3) but no operating system uses ring 1 or ring 2 in practice (save for OS/2, a long-dead operating system from IBM).

The Virtual Hardware Platform

- The diversity of I/O peripherals in x86 personal computers made it impossible to match the virtual hardware to the real, underlying hardware
- The virtualization platform consisted of a combination of multiplexed and emulated components. Multiplexing meant configuring the hardware so it can be directly used by the virtual machine, and shared (in space or time) across multiple virtual machines
- For the multiplexed hardware, each virtual machine had the illusion of having one dedicated CPU and a configurable, but a fixed amount of contiguous RAM starting at physical address 0.
- Architecturally, the emulation of each virtual device was split between a front-end component, which was visible to the virtual machine, and a back-end component, which interacted with the host operating system

The Role of the Host Operating System

- First, it would address the second part of peripheral diversity challenge. VMware implemented the front-end emulation of the various devices, but relied on the device drivers of the host operating system for the back end.
- Second, the product could install and feel like a normal application to a user, making adoption easier
- However, a normal application does not have the necessary hooks and APIs necessary for a hypervisor to multiplex the CPU and memory resources, which is essential to provide near-native performance
- In it, As shown in Fig. 7-10, the software is broken into three separate and distinct components.
- These components each have different functions and operate independently from one another:
    + The VMX performs all UI functions, starts the virtual machine, and then performs most of the device emulation (front end), and makes regular system calls to the host operating system for the back-end interactions
    + A small kernel-mode device driver (the VMX driver), which gets installed within the host operating system. It is used primarily to allow the VMM to run by temporarily suspending the entire host operating system.
    + The VMM, which includes all the software necessary to multiplex the

CPU and the memory, including the exception handlers, the trap-and-emulate handlers, the binary translator, and the shadow paging module. The VMM runs in kernel mode, but it does not run in the context of the host operating system..

- Figure 7-11 shows the difference between the two. The regular context switch between processes “A” and “B” swaps the user portion of the address space and the registers of the two processes, but leaves a number of critical system resources unmodified

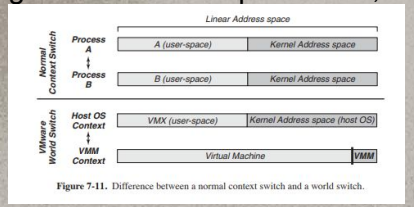


Figure 7-11. Difference between a normal context switch and a world switch.

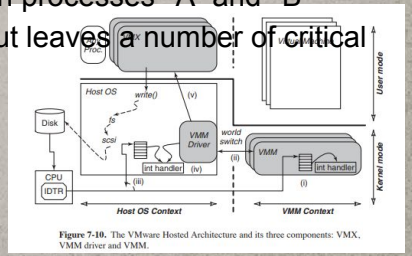


Figure 7-10. The VMware Hosted Architecture and its three components: VMX, VMM driver and VMM.

7.12.5 The Evolution of VMware Workstation

- The hosted architecture is still used today for state-of-the-art interactive hypervisors such as VMware Workstation, VMware Player, and VMware Fusion and even in VMware’s product aimed at cell phones.
- In contrast, the approach to the virtualization of the x86 architecture changed rather dramatically with the introduction of hardware-assisted virtualization.Hardware-assisted virtualizations, such as Intel VT-x and AMD-v were introduced in two phases
- The emergence of hardware support for virtualization had a significant impact on VMware’s guest operating system centric-strategy

7.12.6 ESX Server: VMware's type 1 Hypervisor

- In 2001, VMware released a different product, called ESX Server, aimed at the server marketplace
- Figure 7-12 shows the high-level architecture of ESX Server. It combines an existing component, the VMM, with a true hypervisor running directly on the bare metal.
- Despite the drawbacks, the trade-off made sense for dedicated deployments of virtualization in data centers, consisting of hundreds or thousands of physical servers, and often (many) thousands of virtual machines. For example:

+ The CPU scheduler ensures that each virtual machine gets a fair share of the CPU
+ The memory manager is optimized for scalability, in particular to run virtual machines efficiently even when they need more memory than is actually available on the computer
+ The I/O subsystem is optimized for performance. Although VMware Workstations and ESX servers often share the same front-end emulation components, but the back ends are totally different
+ The back ends also typically relied on abstractions provided by the host operating system
+ ESX Server made it easy to introduce new capabilities, which required the tight coordination and specific configuration of multiple components of a computer

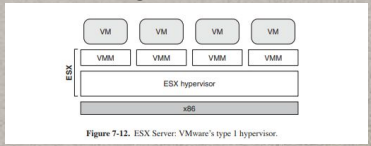
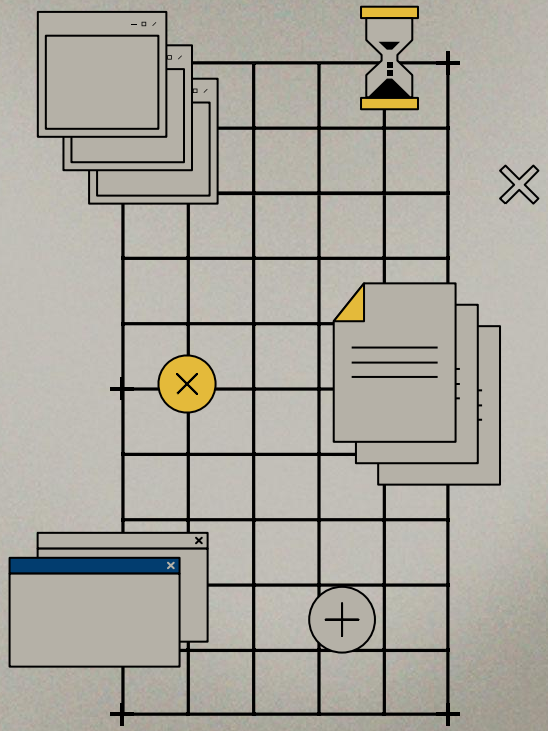


Figure 7-12. ESX Server: VMware's type 1 hypervisor.

7.13 Research on Virtualization and Cloud

- Virtualization technology and cloud computing are both extremely active research areas. The research produced in these fields is way too much to enumerate. Each has multiple research conferences.
- One interesting area which has seen a lot of interesting research is nested virtualization (Ben-Yehuda et al., 2010; and Zhang et al., 2011). The idea is that a virtual machine itself can be further virtualized into multiple higher-level virtual machines, which in turn may be virtualized and so on
- One of the nice things about virtualization hardware is that untrusted code can get direct but safe access to hardware features like page tables, and tagged TLBs.



THANKS!

Do you have any questions?

Please keep this slide for attribution