CS 444 Operating Systems

Chapter 7 - Virtualization and the Cloud

J. Holly DeBlois

April 8, 2025

Ch7 Summary - Let's start by looking at section 7.13 Tanenbaum's Summary, p524

- "**Virtualization is the technique of simulating a computer [or a part of it], but with high performance.** Typically one computer runs many virtual machines at the same time. This technique is widely used in data centers to provide cloud computing. In this chapter we looked at how virtualization works, especially for paging, I/O, and multicore systems. We also studied an example: VMWare." (Tanenbaum, Ch7, p524)

Where in the textbook does Tanenbaum cover Virtualization?

- Ch1, Introduction: Section 1.7.5 OS Structure - Virtual Machines
- Ch7, Virtualization and the Cloud: Introduction and Sections 7.1-7.10
- Ch7, Case Study on VMWare, Section 7.11
- Ch10, Case Study: Unix, Linux and Android, Figure 10-3, Structure of the Linux kernel, p722, and Figure 10-39, Android Process Hierarchy, p802
- Ch11, Case Study: Windows 11, Figure 11.11 Windows kernel-mode organization, p895

What is covered in Ch7?

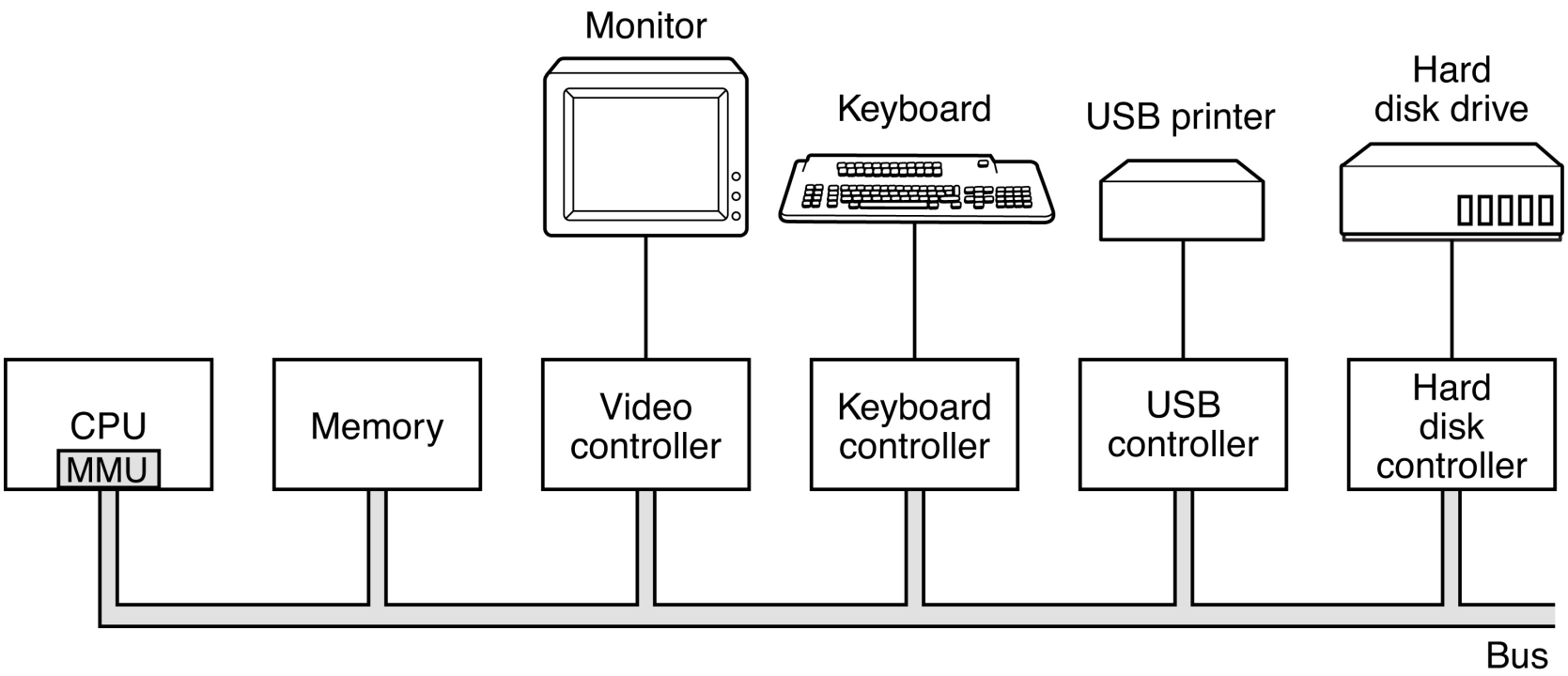
- Ch7, Introduction and 7.1 History
- Ch7, Virtualizing the CPU, Sections 7.2-7.5
- Ch7, Virtualizing Memory, Section 7.6
- Ch7, Virtualizing I/O, Section 7.7
- Ch7, Virtual Machines on Multicore Chips, Section 7.8
- Ch7, Clouds, Section 7.9
- Ch7, OS-Level Virtualization, Section 7.10
- Ch7, Case Study on VMWare, Section 7.11

Ch1, Introduction: Section 1.7.5 OS Structure - Virtual Machines

- Let's review a little of Ch1
- and then focus on Section 1.7.5 Virtual Machines, which is just the basics

Hardware Overview

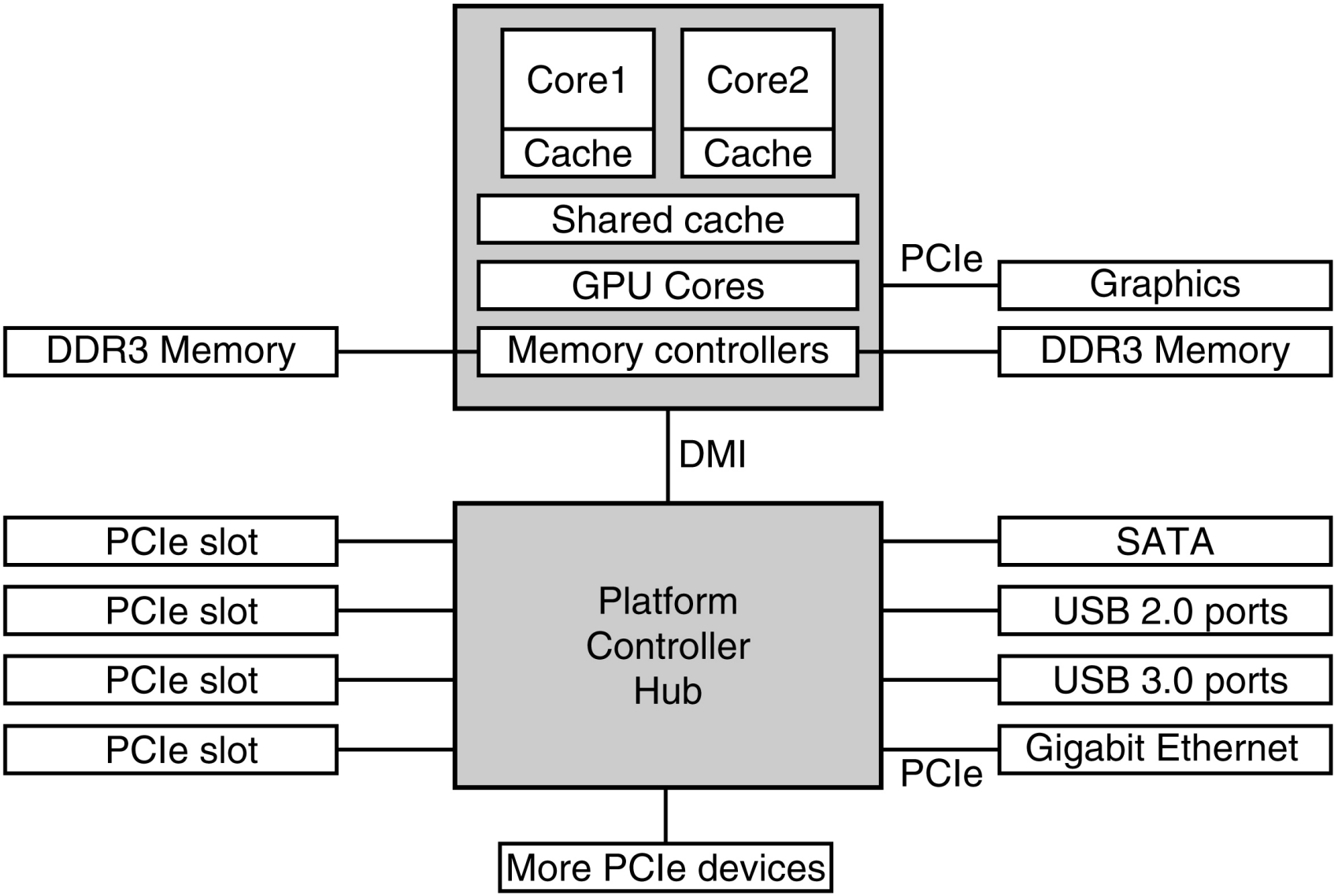
- Figure 1-6 Some of the components of a simple personal computer



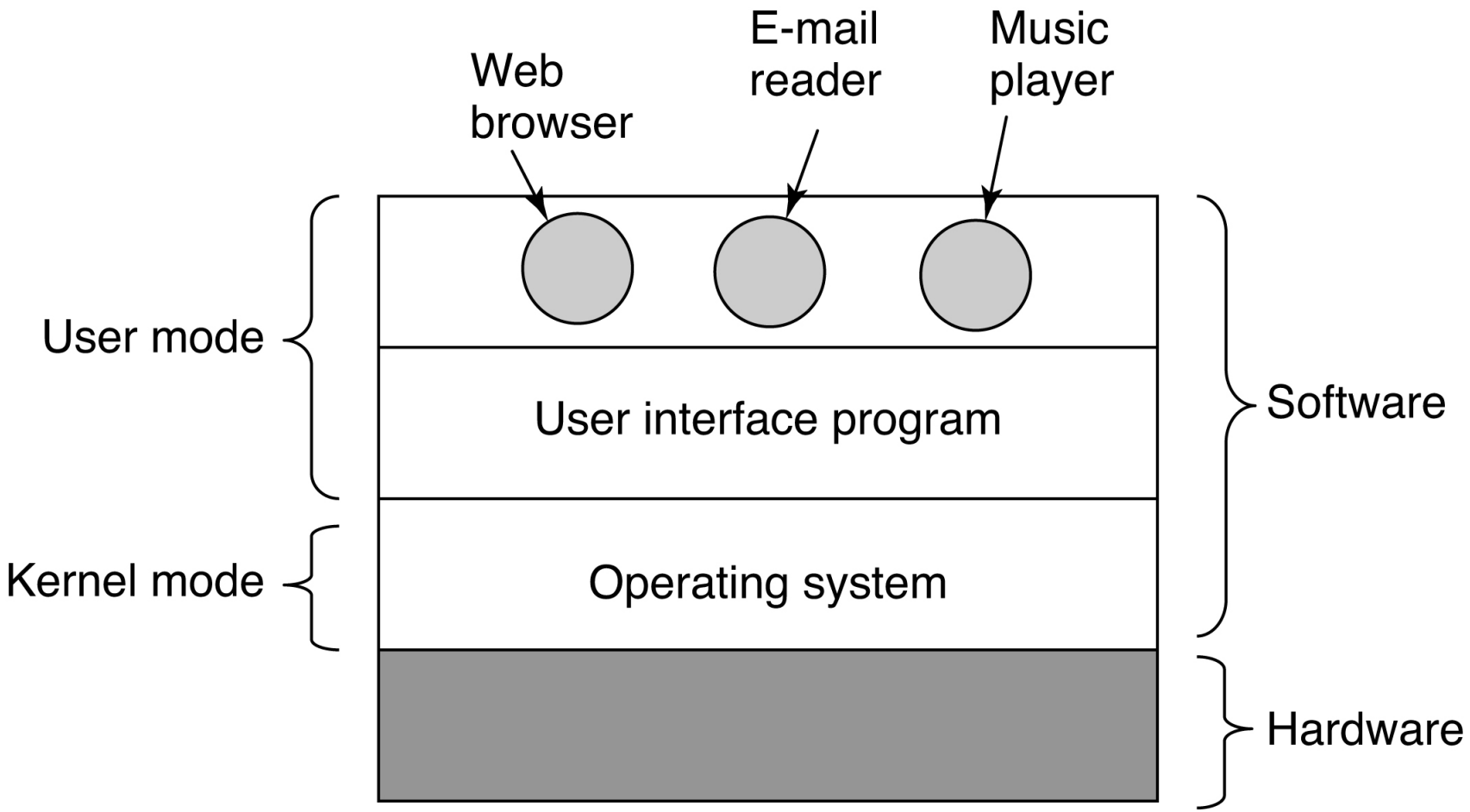
CPU, x86 line and instruction sets

- "The CPU, the "brain" of the computer, fetches instruction and executes them." (p20, Tanenbaum)
- "The x86 stands for all the intel processors from the IBM PC on." (p20, Tanenbaum)
- "Each type of CPU has a specific set of instructions." (p21, Tanenbaum)
- "Please note that we will use the term x86 to refer to all the Intel processors descended from the 8088, which was used on the original IBM PC. These include the 286, 386, and Pentium series, as well as the modern Intel core i3, i5, and i7 CPUs (and their clones)." (p21, Tanenbaum)

Figure 1-12 shows a modern x86 with many buses. (p33, Tanenbaum)



In the simplest setup, the OS sits above the hardware, manages it and interfaces with user processes above it.

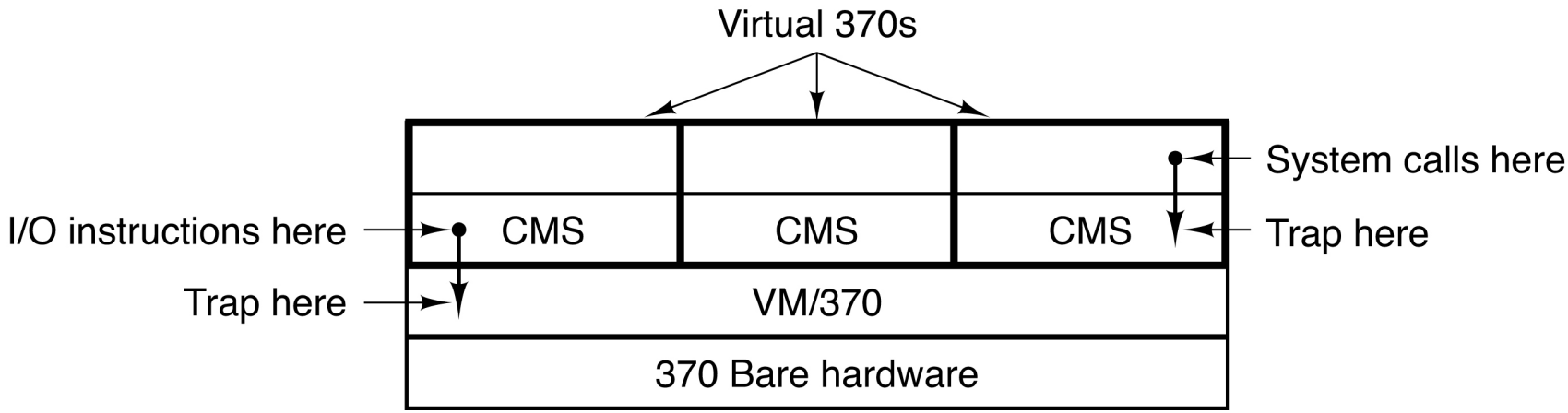


In 1979, IBM had the first Virtual Machine (VM)

- In 1979, IBM offered its VM/370, with virtual machine monitor running on bare hardware, able to do multiprogramming and provide several identical virtual machines to do timesharing. See Fig1-28, p70.
- But virtualization was virtually ignored in the PC world for 20 years!

Three Virtual Machines running on the IBM VM/370 layer

- Here is what IBM built:
- Figure 1-28, the structure of VM/370 with CMS



- System calls and I/O instructions trap to lower levels until they reach the 370 Bare hardware

The need for virtualization was there, but the x86 line of Intel PCs had a problem

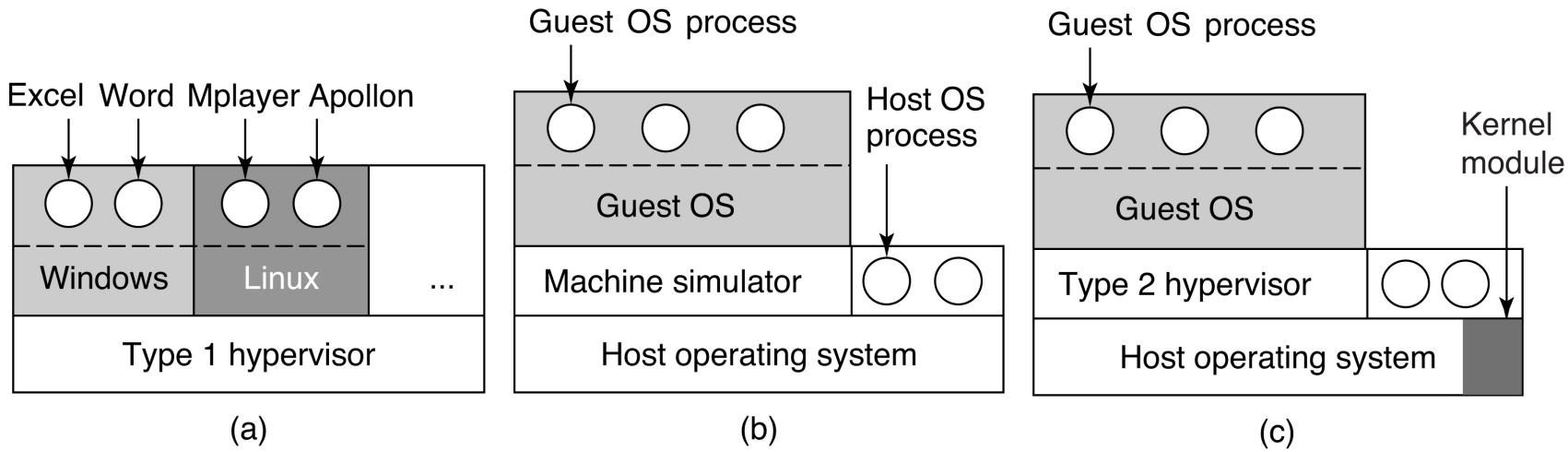
- In PCs, the need for virtualization arose: to run separate mail servers on the same machine, to rent and control a web host virtual machine, to appeal to users running several types of OS.
- Trying to execute a privileged instruction in user mode was simply ignored, so it did not trap to the OS and could not reach the VM.

To run VM software, a CPU's instructions must be "virtualizable." (Popek and Goldberg, 1974: I will post the paper on the website!)

- Some instructions behave differently in kernel mode than in user mode; these were called sensitive instructions;
- Some instructions cause a trap if executed in user mode; these were called privileged instructions;
- The sensitive instructions must be a subset of the privileged instructionsfor the machine to be virtualizable. (taken from p483, Tanenbaum)

Years Later, in 1999, Virtual Machines were rediscovered for PCs: VMware produced VMs

- Figure 1-29 shows the progression of the early VMware machines:
- (a) A virtual machine monitor, now called a type 1 hypervisor
- (b) A pure type 2 hypervisor that does binary translation
- (c) A practical type 2 (1.7) hypervisor with a kernel module - little black box in corner



Now let's turn to Ch7 and see what the more detailed view shows us

- Let's call that new layer the Virtual Machine Monitor (VMM)
- Let's ask more about how a group of OSs can appear to run on the same hardware
- Let's ask about how to virtualize not only the OS, but also memory and I/O devices
- Let's learn about other varieties of how to do similar things

Ch7 Virtualization and the Cloud: Introduction and Main Ideas

- "The main idea is that a VMM (Virtual Machine Monitor) creates the illusion of multiple (virtual) machines on the same physical hardware. A VMM is also known as a hypervisor." (p478, Tanenbaum)
- "As discussed in Sec. 1.7.5, we distinguish between type 1 hypervisors which run on the bare metal, and type 2 hypervisors that may make use of all the wonderful services and abstractions offered by an underlying operating system. Either way, virtualization allows a single computer to host multiple virtual machines, each potentially running a completely different operating system." (p478, Tanenbaum)

Figure 7-1: Location of type 1 and type 2 hypervisors. Type 1 is discussed in the Popek and Goldberg paper in 1974. Type 2 came along in 2012, p485

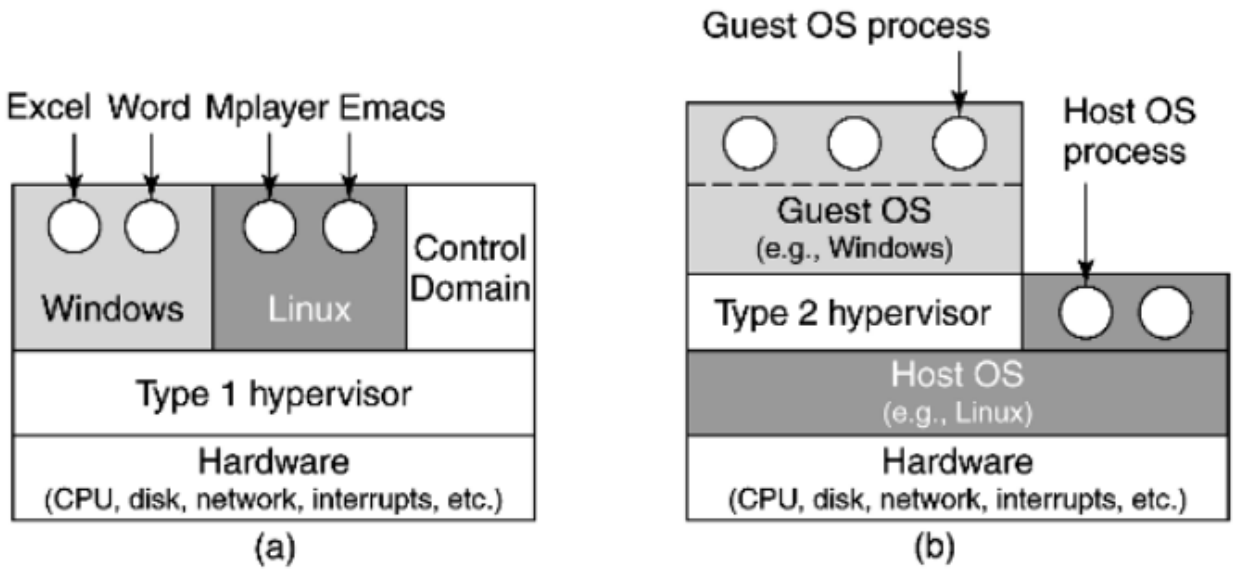


Figure 7-1. Location of type 1 and type 2 hypervisors.

Ch7 Virtualization: Three Requirements for Virtual Machines from p482 in Section 7.2

- "If we leave OS-level virtualization aside, it is important that virtual machines act just like the real McCoy. In particular, it must be possible to boot them like real machines and install arbitrary operating systems on them, just as can be done on the real hardware. It is the task of the hypervisor to provide this illusion and to do it efficiently." (p482, Tanenbaum)
- "Indeed, hypervisors should score well in three dimensions:
- 1. Safety: The hypervisor should have full control of the virtualized resources.
- 2. Fidelity: The behavior of a program on a virtual machine should be identical to that of the same program running on bare hardware.
- 3. Efficiency: Much of the code in the virtual machine should run with- out intervention by the hypervisor." (p482, Tanenbaum)

Thus, we can identify different Virtualization Methods and give examples as Models that were marketed

Virtualizaton method	Type 1 hypervisor	Type 2 hypervisor
Virtualization without HW support	ESX Server 1.0	VMware Workstation 1
Paravirtualization	Xen 1.0	VirtualBox 5.0+
Virtualization with HW support	vSphere, Xen, Hyper-V	VMware Fusion, KVM, Parallels
Process virtualization		Wine

Figure 7-2. Examples of hypervisors. Type 1 hypervisors run on the bare metal, whereas type 2 hypervisors use the services of an existing host operating system.

The Problem with the Instructions is restated in Section 7.2, p482

- "In a nutshell, every CPU with kernel mode and user mode has a set of instructions that behave differently when executed in kernel mode than when executed in user mode. These include instructions that do I/O, change the MMU settings, and so on. Popek and Goldberg called these sensitive instructions." (p482, Tanenbaum)
- "There is also a set of instructions that cause a trap if executed in user mode. Popek and Goldberg called these privileged instructions." (p482, Tanenbaum)
- "Their paper stated for the first time that a machine is virtualizable only if the sensitive instructions are a subset of the privileged instructions. In simpler language, if you try to do something in user mode that you should not be doing in user mode, the hardware should trap." (p482, Tanenbaum).
- The problem was solved in 2005 when Intel and AMD introduced Virtualization Technology.

Contradiction?

- The problem was solved in 2005, but the VMware machine was produced in 1999!
- The VMware machine used binary translation to rewrite the guest operating system running in ring 1 and was able to run the Type 1 hypervisor in ring 0.

How the binary translator can be combined with the hypervisor

- This is a very clever design!

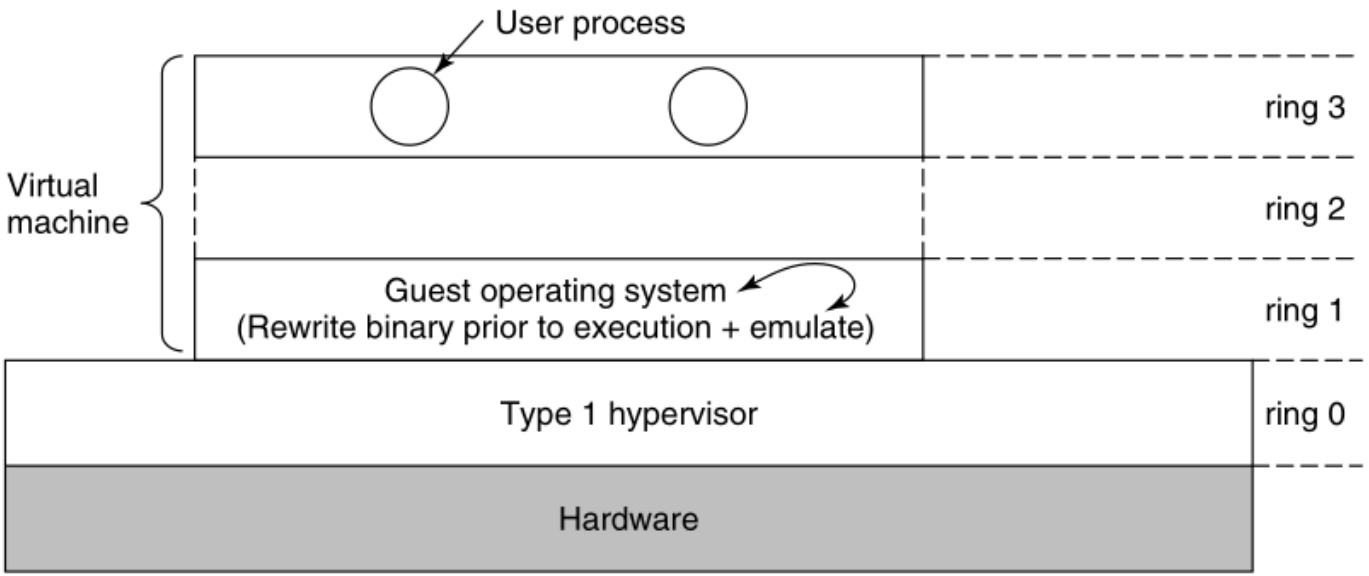


Figure 7-4. The binary translator rewrites the guest operating system running in ring 1, while the hypervisor runs in ring 0.

How Virtualization Works

- This is the basic technique, as it ended up

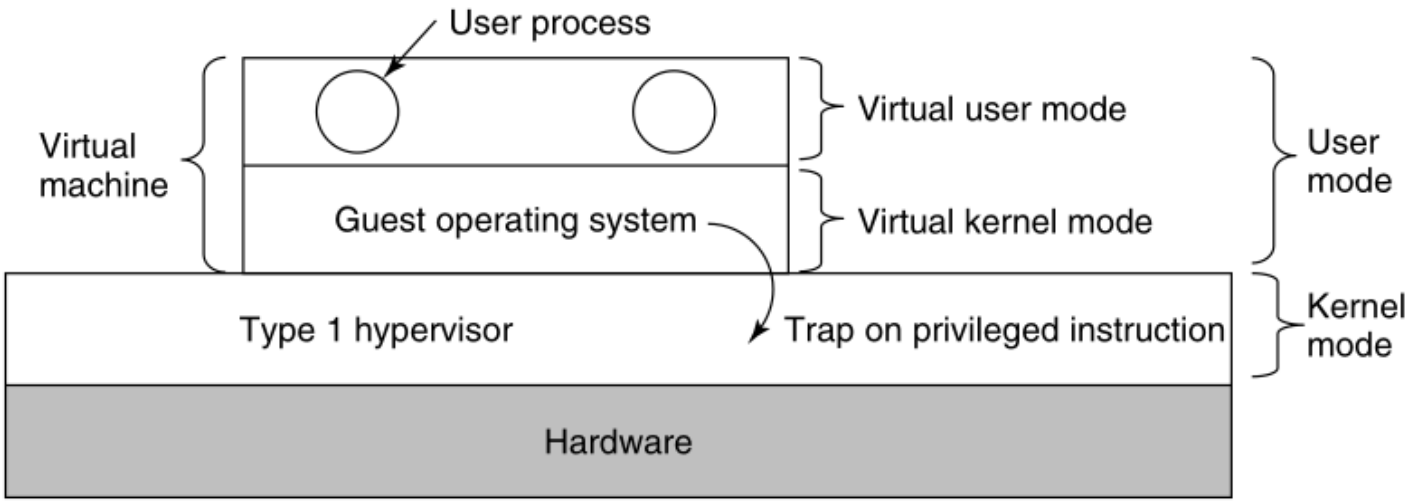


Figure 7-3. When the operating system in a virtual machine executes a kernel-only instruction, it traps to the hypervisor if virtualization technology is present.

True virtualization vs. paravirtualization

- True virtualization is on the left.
- Paravirtualization is on the right.

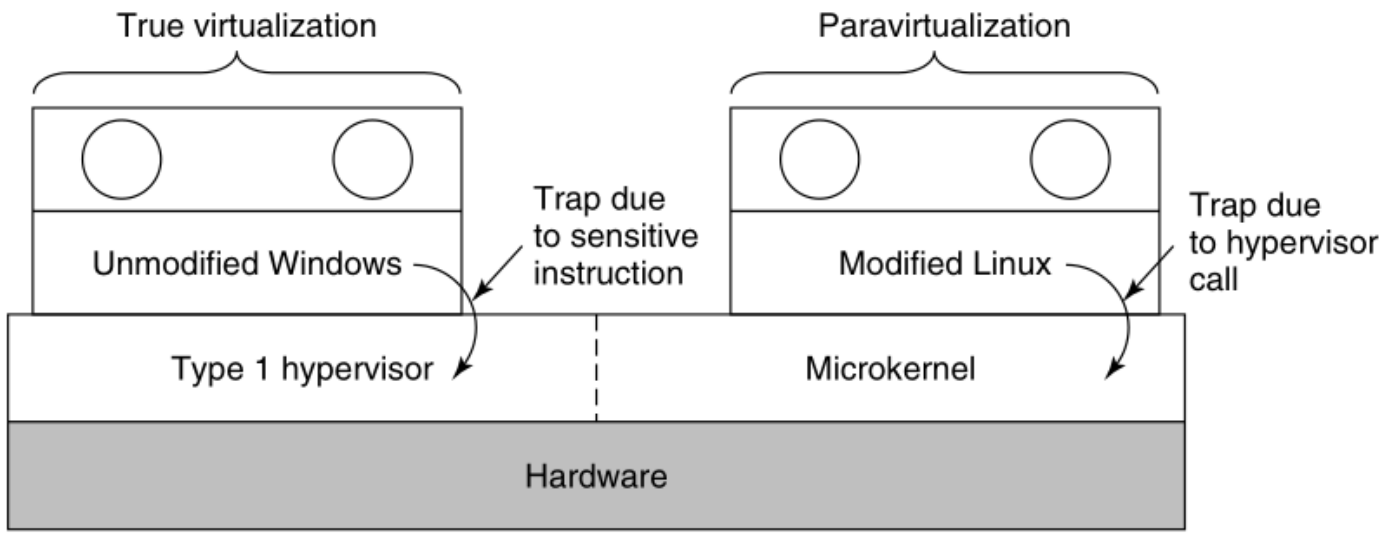
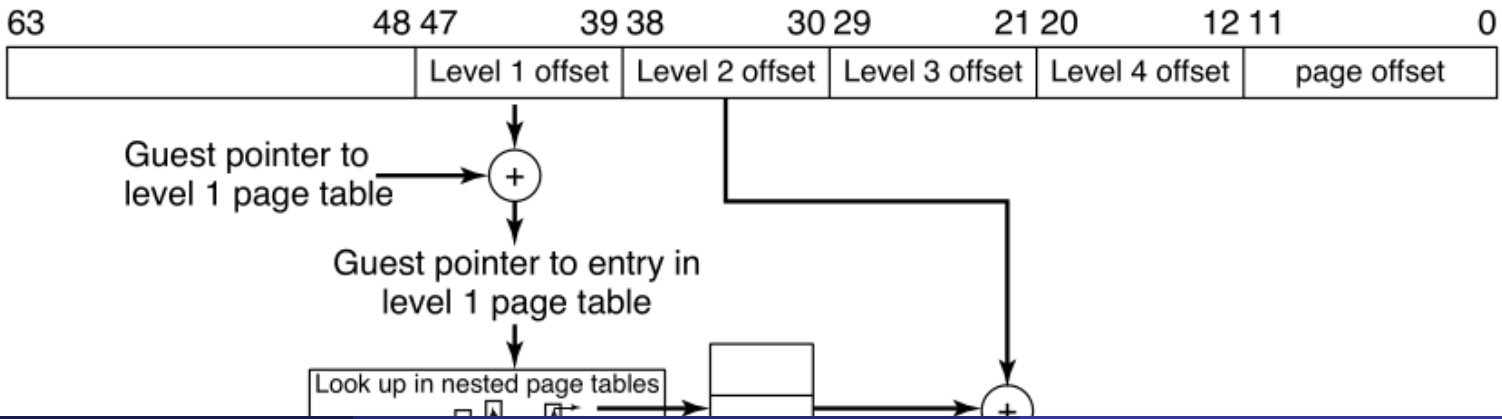


Figure 7-5. True virtualization and paravirtualization.

- This concludes the sections on how to virtualize the CPU.

Section 7.6, p493, Virtualized Memory

- The textbook explains how virtual memory works, not just to map virtual addresses to physical addresses (as we saw in Ch3), but also to help out when multiple virtual machines wants to acces the same physical page.
- The hypervisor creates a shadow page table for each additional virtual machine's mapppings.
- After several tries, these techniques worked but they were expensive due to page faults.
- So chip makers added hardware support for nested page tables.
- AMD calls them nested page tables. Intel calls them Extended Page tables.



Here is how the National Institute of Standards and Technology defines "Cloud":

- These are the essential characteristics.

1. **On-demand self-service.** Users should be able to provision resources automatically, without requiring human interaction.
2. **Broad network access.** All these resources should be available over the network via standard mechanisms so that heterogeneous devices can make use of them.
3. **Resource pooling.** The computing resource owned by the provider should be pooled to serve multiple users and with the ability to assign and reassign resources dynamically. The users generally do not even know the exact location of "their" resources or even which country they are located in.
4. **Rapid elasticity.** It should be possible to acquire and release resources elastically, perhaps even automatically, to scale immediately with the users' demands.
5. **Measured service.** The cloud provider meters the resources used in a way that matches the type of service agreed upon.

Now in section 7.11 there is the Case Study of VMware

- The challenges were many!

1. **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. A hypervisor needed to provide sufficient compatibility at the hardware level such that users could run whichever operating system (down to the update and patch version) they wished to install within a particular virtual machine, without restrictions.

2. **Performance**. The overhead of the hypervisor had to be sufficiently low that users could use a virtual machine as their primary work environment. As a goal, the designers of VMware aimed to run relevant workloads at near native speeds, and in the worst case to run them on then-current processors with the same performance as if they were running natively on the immediately prior generation of processors. This was based on the observation that most x86 software was not designed to run only on the latest generation of CPUs.

3. **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. Software running inside virtual machines had to be prevented from any access that would allow it to subvert the hypervisor. Similarly, a hypervisor had to ensure the privacy of all data not

Case Study: VMware

- High-level components are shown.

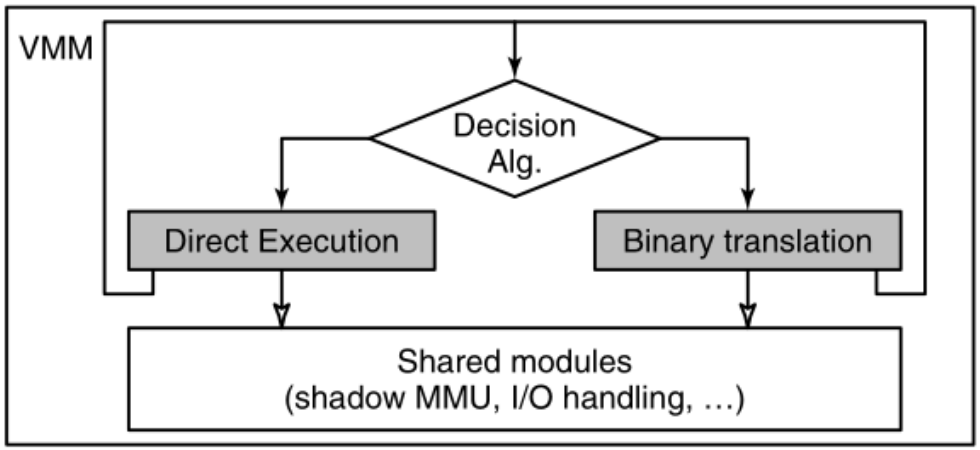


Figure 7-8. High-level components of the VMware virtual machine monitor (in the absence of hardware support).

1. The virtual machine is currently running in kernel mode (ring 0 in the x86 architecture).
2. 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).

Early compromises worked out well for them

- p 516

	Virtual Hardware (front end)	Back end
Multiplexed	1 virtual x86 CPU, with the same instruction set extensions as the underlying hardware CPU	Scheduled by the host operating system on either a uniprocessor or multiprocessor host
	Up to 512 MB of contiguous DRAM	Allocated and managed by the host OS (page-by-page)

Emulated	PCI Bus	Fully emulated compliant PCI bus
	4x IDE disks 7x Buslogic SCSI Disks	Virtual disks (stored as files) or direct access to a given raw device
	1x IDE CD-ROM	ISO image or emulated access to the real CD-ROM
	2x 1.44 MB floppy drives	Physical floppy or floppy image
	1x VMware graphics card with VGA and SVGA support	Ran in a window and in full-screen mode. SVGA required VMware SVGA guest driver
	2x serial ports COM1 and COM2	Connect to host serial port or a file
	1x printer (LPT)	Can connect to host LPT port
	1x keyboard (104-key)	Fully emulated; keycode events are generated when they are received by the VMware application
	1x PS-2 mouse	Same as keyboard

Here is the well-known world switch design

- We see the architecture, p518

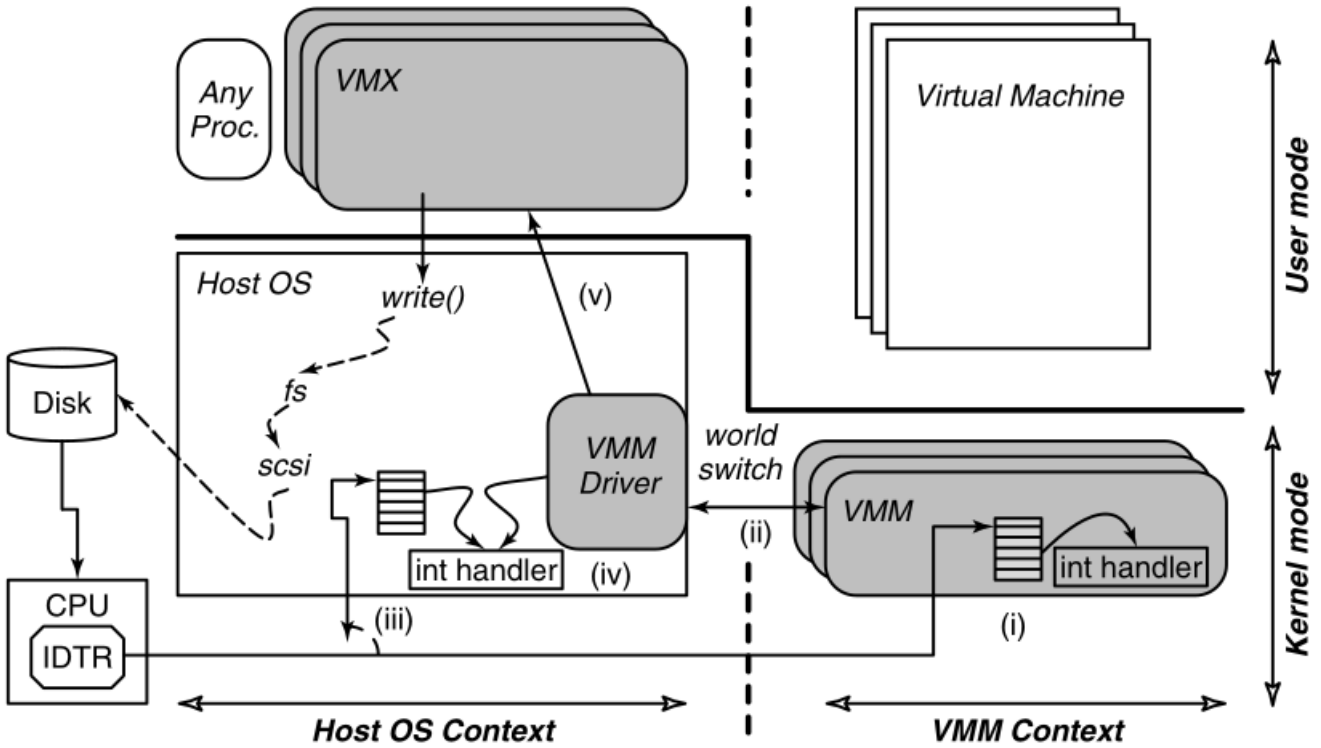


Figure 7-10. The VMware Hosted Architecture and its three components: VMX, VMM driver, and VMM.

Here is the definition of the terms in the diagram

- p518

These components each have different functions and operate independently from one another:

1. A user-space program (the **VMX**) which the user perceives to be the VMware program. 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. There is typically one multithreaded VMX process per virtual machine.
2. 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. There is one VMX driver installed in the host operating system, typically at boot time.
3. 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. In other words, it cannot rely directly on services offered by the host operating system, but it is also not constrained by any rules or conventions imposed by the host operating system. There is one VMM instance for each virtual machine, created when the virtual machine starts.

Compare the two types of context switching

- p520

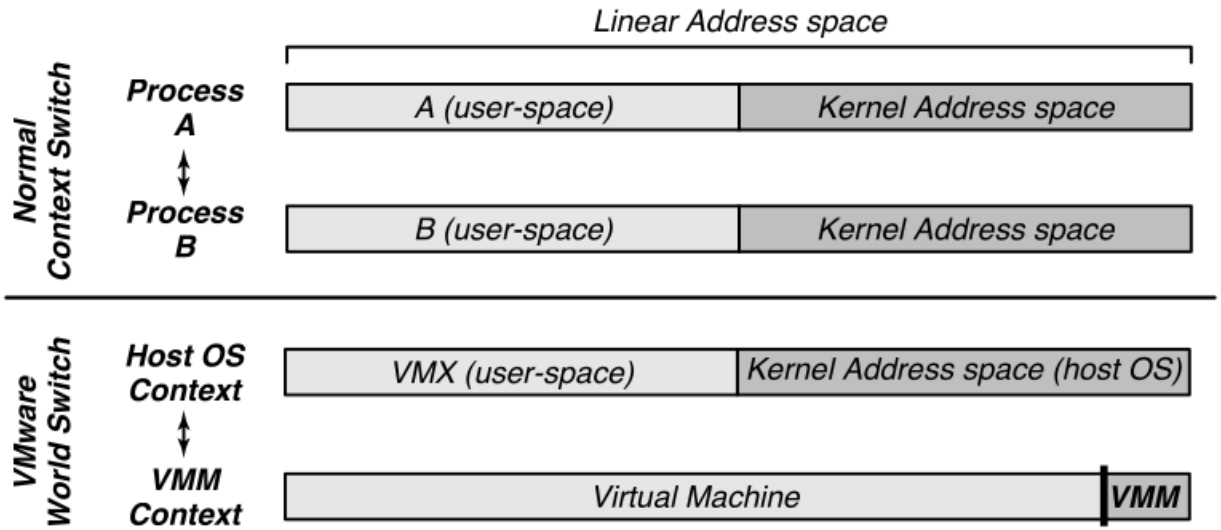


Figure 7-11. Difference between a normal context switch and a world switch.

Note the simplicity of the VMware ESX

- p522

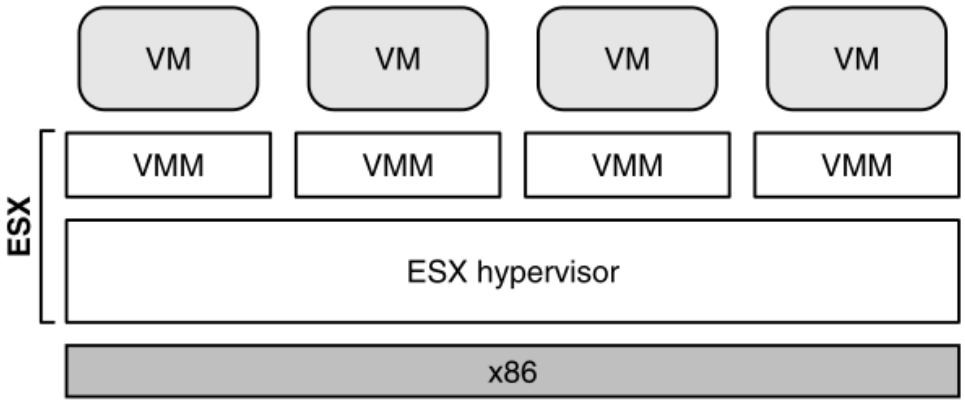


Figure 7-12. ESX Server: VMware's type 1 hypervisor.