Platform Virtualization

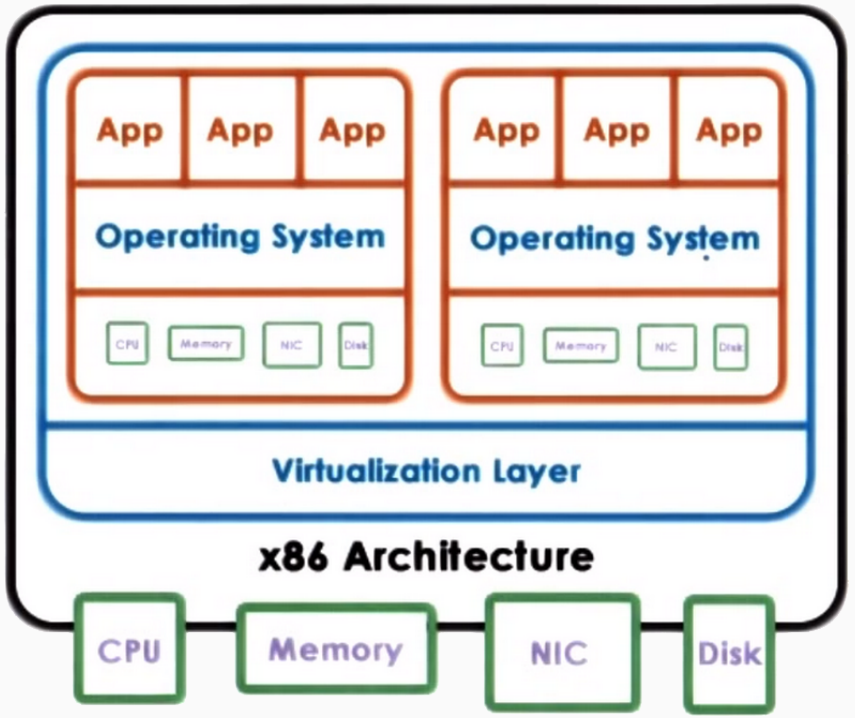
Florent Gluck - Florent.Gluck@hesge.ch

February 22, 2023

*Thanks to Prof. Ada Gavrilovska for letting me use some of the contents of her course "Introduction to Operating Systems" thought at Georgia Institute of Technology

What is Platform virtualization?

- Virtualization of a **whole hardware platform** $\rightarrow$ allows concurrent execution of multiple OSes on the same physical machine (host system)
- **Virtual machine (VM)** or guest domain = efficient, isolated duplicate of the real physical machine
- A VM is supported by a virtualization layer = **virtual machine monitor (VMM)** or **hypervisor**



Popek and Goldberg requirements for a VMM

In 1974, Popek and Goldberg provide three major requirements for a VMM:

1. **Equivalence**: provide an environment essentially identical to the original machine; software on the VMM executes nearly identically to how it would on the real hardware

Popek and Goldberg requirements for a VMM

In 1974, Popek and Goldberg provide three major requirements for a VMM:

1. **Equivalence**: provide an environment essentially identical to the original machine; software on the VMM executes nearly identically to how it would on the real hardware
2. **Efficiency**: a statistically dominant fraction of machine instructions must be executed by the guest OS without VMM intervention; these instructions **run directly on the underlying physical hardware**

Popek and Goldberg requirements for a VMM

In 1974, Popek and Goldberg provide three major requirements for a VMM:

1. **Equivalence**: provide an environment essentially identical to the original machine; software on the VMM executes nearly identically to how it would on the real hardware
2. **Efficiency**: a statistically dominant fraction of machine instructions must be executed by the guest OS without VMM intervention; these instructions **run directly on the underlying physical hardware**
3. **Safety & isolation**: VMM is in complete control of the virtualized resources (hardware)

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM
  - no: equivalence and efficiency not satisfied

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM
  - no: equivalence and efficiency not satisfied
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM
  - no: equivalence and efficiency not satisfied
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller
  - yes

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM
  - no: equivalence and efficiency not satisfied
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller
  - yes
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Realtek RTL8111E network card, SCSI Adaptec Disk controller

Quiz: VMM

Based on the classical definition of virtualization by Popek & Goldberg, which of the following is a VMM?

We consider the following host: CPU Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller

- Java Virtual Machine (JVM)
  - no: equivalence, efficiency and isolation not satisfied
- QEMU running a Raspberry PI VM
  - no: equivalence and efficiency not satisfied
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Intel Pro 1000 network card, IDE PIIX Disk controller
  - yes
- Oracle VirtualBox running a VM with: Intel Core i7, VGA card, Realtek RTL8111E network card, SCSI Adaptec Disk controller
  - no: equivalence not satisfied

Modern VMM definition

- Popek and Goldberg **efficiency** and **safety & isolation** requirements for a VMM remain true to this day
- However, it is not true of the **equivalence** requirement
- Nowadays, a VMM might expose different hardware devices than the physical hardware present on the host and still be called a VMM
  - most VMM fall in this category
  - in fact, most VMM expose virtual hardware devices instead (for performance reasons)

Platform virtualization

- Platform virtualization is a type of virtualization that **virtualizes a whole hardware system**:
  - CPU
  - memory (MMU - Memory Managing Unit)
  - devices: hard drive, disk controllers, display, mouse, keyboard, etc.
- Platform virtualization is also called “hardware virtualization”

Types of platform virtualization

Platform virtualization uses various techniques to virtualize the hardware:

- Full virtualization
- Paravirtualization
- Hardware-assisted virtualization

Types of virtualization: full virtualization

Full virtualization is a type of virtualization where the guest OS **does NOT know** it's running on top of a VMM

- Benefits
  - guest OS doesn't need to be modified
  - any existing OS and existing drivers for real hardware will work
    - provided the VMM emulates the necessary devices
  - guest OS can run on real hardware
- Downsides
  - inefficient $\rightarrow$ low performances
  - complex implementation

Types of virtualization: paravirtualization (1/2)

Paravirtualization is a type of virtualization where the guest OS **knows** it's running on top of a VMM

- Can be used for CPU virtualization and/or device virtualization
- **CPU paravirtualization**: **guest kernel must be modified**
  - slight modifications (few %)
- **Device paravirtualization**: VMM exposes a virtual device (whose hardware doesn't exist) that's easy to program for
- Guest OS makes explicit calls to the VMM through **hypercalls**
  - typically for privileged operations

Types of virtualization: paravirtualization (2/2)

- Benefits
  - much better performances than full virtualization
    - for both CPU and devices
  - much simpler driver implementation
- Downsides
  - guest kernel must be modified (for CPU paravirtualization)
  - guest OS cannot run on real hardware

Types of virtualization: hardware-assisted virtualization (1/2)

Hardware-assisted virtualization is a type of virtualization that makes full virtualization much more **efficient** through dedicated hardware instructions

- Exists for CPU virtualization and devices virtualization
- **Guest kernel does not need to be modified**

Types of virtualization: hardware-assisted virtualization (2/2)

- Benefits
  - guest OS doesn't need to be modified
  - much better performances than non hardware-assisted full virtualization
- Downsides
  - CPU must feature support for virtualization instructions

Platform virtualization nowadays

Nowadays, VMM that implement platform virtualization use a combination of virtualization types:

- **Hardware-assisted** virtualization for CPU and devices
- **Paravirtualization** for devices
  - typically for performance-critical devices, such as disk and network
- **Full virtualization** for devices
  - typically for devices for which performance is non-critical

How to virtualize the machine?

In platform virtualization, 3 main components must be virtualized:

- CPU
- devices (also called Input/Output or I/O)
- memory (MMU)

Platform virtualization: CPU virtualization

CPU virtualization

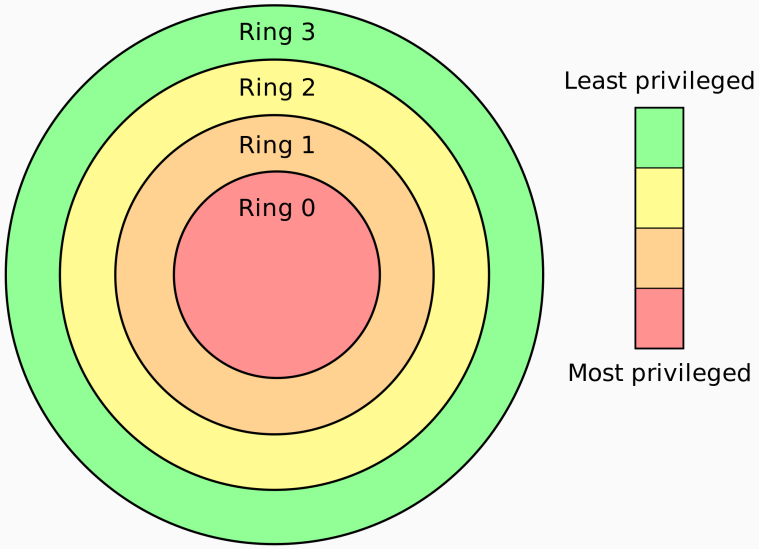
There are several ways of virtualizing the CPU when implementing platform virtualization

- CPU full virtualization via:
  - Trap-and-Emulate
  - Dynamic Binary Translation
  - Hardware-assisted
- CPU paravirtualization

Here, we present these techniques and the history behind them

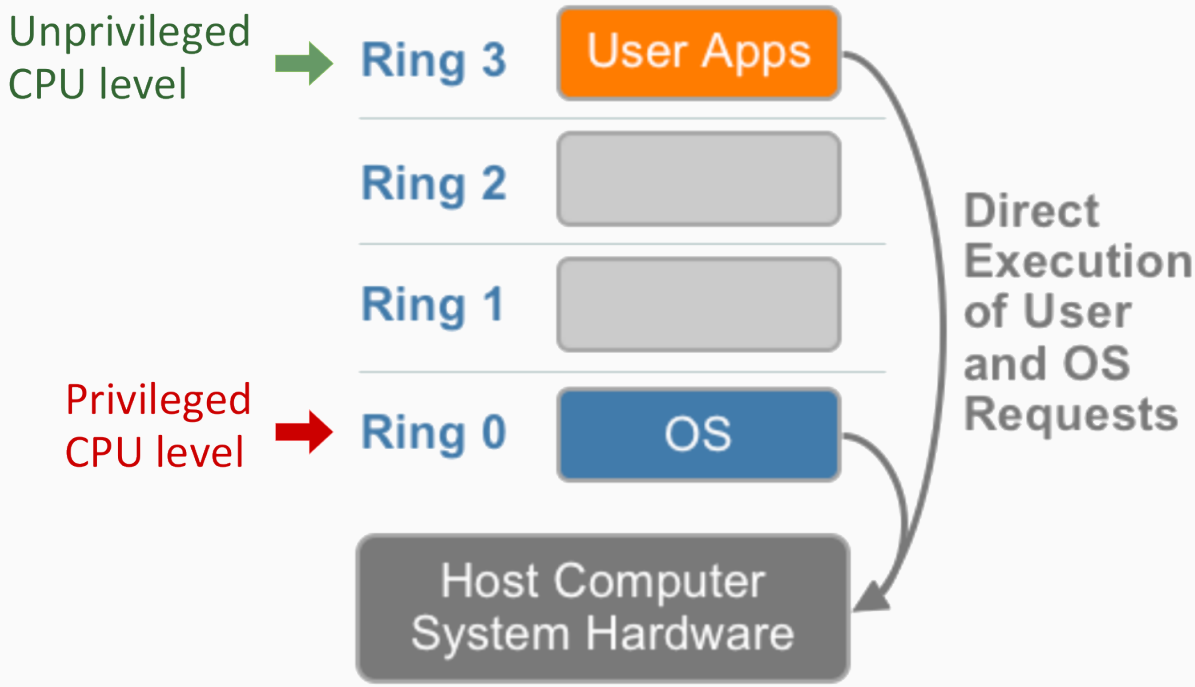
Hardware protection levels on x86 CPU

- Commodity hardware has more than two **protection levels**
- For example, the original Intel 32-bit architecture has **four protection levels**, called rings
- **OS (kernel)** runs in **ring 0**
- **Applications** runs in **ring 3**



Direct execution without virtualization

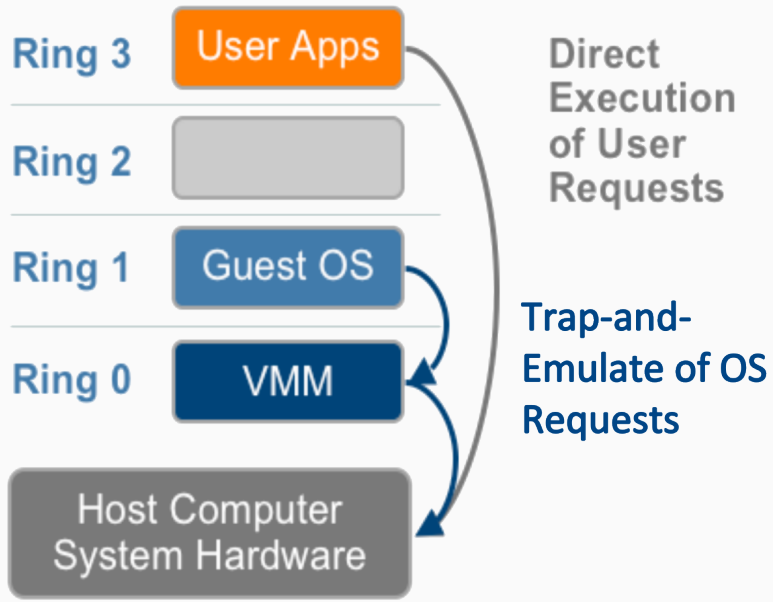
Application execution when executed on top of an OS (x86) **without** any virtualization:



CPU full virtualization: Trap-and-Emulate

Guest OS instructions are executed **directly** by the hardware:

- VMM does not interfere with every instruction/memory access issued by guest OS or applications
- For non-privileged operations → runs at hardware speed = **efficient**
- For privileged operations: **trap** to VMM
- If illegal operation: **terminate** VM
- If legal operation: **emulate** the behavior the guest OS was expecting from the hardware



CPU full virtualization: issue with Trap-and-Emulate

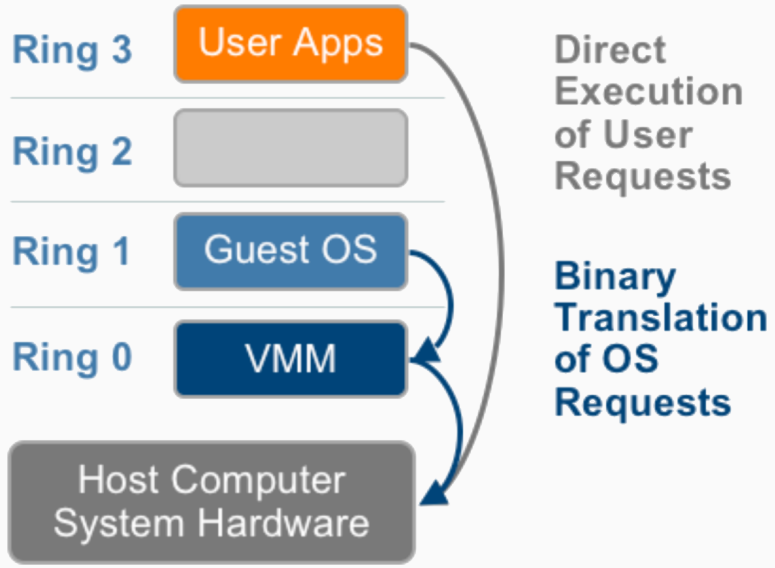
- Trap-and-Emulate worked well on mainframes because their CPUs were designed to support virtualization
- In the 90s, when the need to apply virtualization to x86 architecture arose $\rightarrow$ Trap-and-Emulate model did NOT work!
- x86 pre-2005:
  - 17 privileged instructions do not trap $\rightarrow$ **fail silently!** (doesn't pass control back to VMM)
  - e.g. interrupt enable/disable bit in privilege register; `pushf`/`popf` instructions that access it from ring1 fail silently
    - VMM doesn't know, so doesn't try to change settings
    - guest OS doesn't know, so assume change was successful!

CPU full virtualization: Binary Translation

- Main idea: VMM modifies the guest OS at runtime to never use those 17 instructions
- Pioneered by Mendel Rosenblum's group at Stanford, commercialized as VMware in 1998
  - one of VMware's five co-founders is Swiss computer scientist Edouard Bugnion
- Goal: to **not modify** the guest OS!

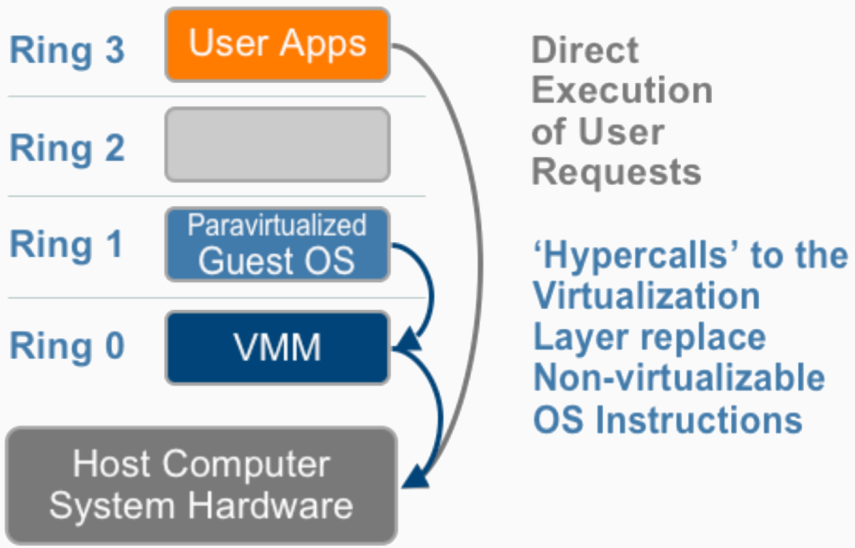
CPU full virtualization: Dynamic Binary Translation

- Dynamic because a sequence may depend on the runtime state
- Input dependent $\rightarrow$ can't be done statically
- Dynamically capture code blocks
- Inspect code blocks to be executed for the 17 instructions
- If needed, translate to alternate instructions sequence (to emulate desired behavior and avoid traps)
- Otherwise $\rightarrow$ run at hardware speed
- Cache translated blocks to amortize translation costs



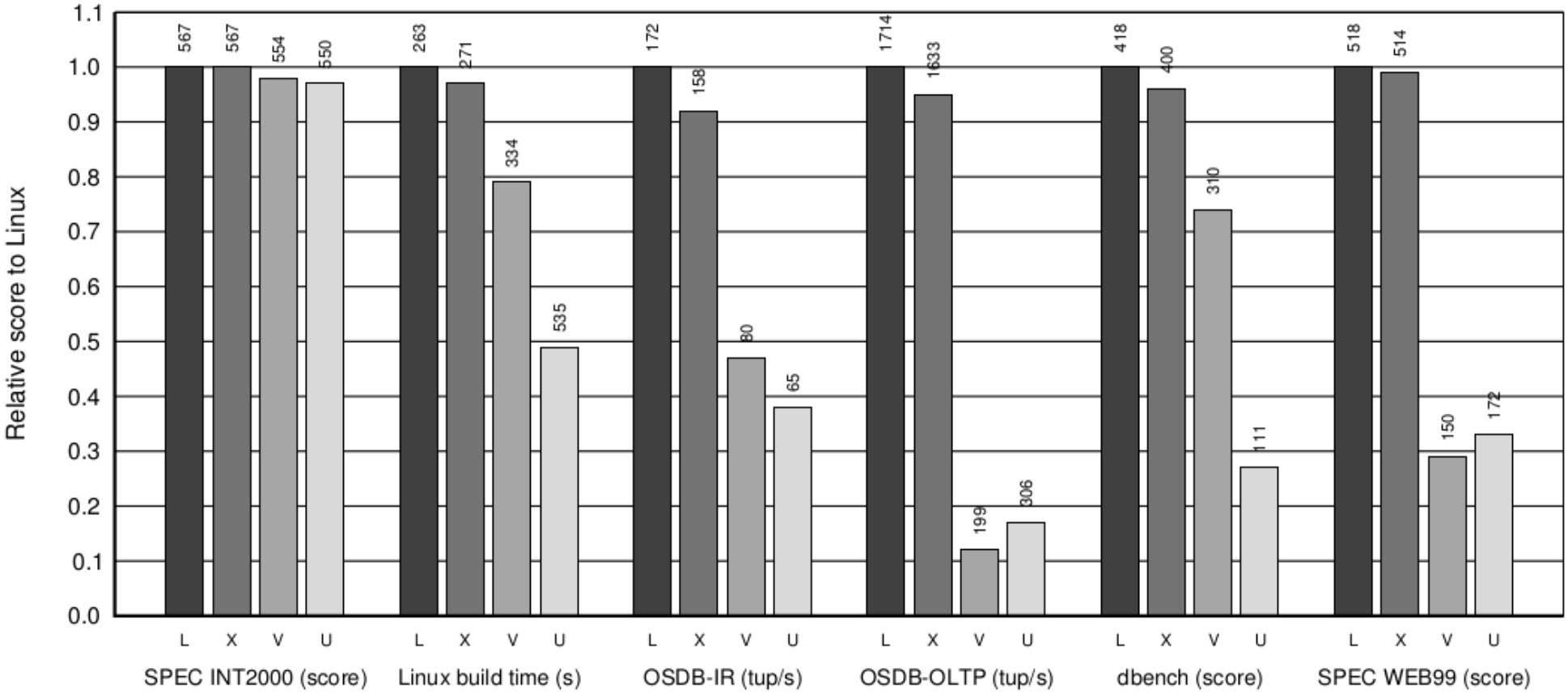
CPU paravirtualization [1]

- Goal: **better performance** by avoiding overhead/complexity required to support unmodified guest OSes:
  - guest OS is slightly modified
  - guest OS knows it's running on top of a VMM
  - for privilege operations, guest OS makes explicit calls to VMM through hypercalls
  - hypercalls from guest OS to VMM $\approx$ system calls from application to OS



[1] Pioneered by the XEN hypervisor at the University of Cambridge in 2003

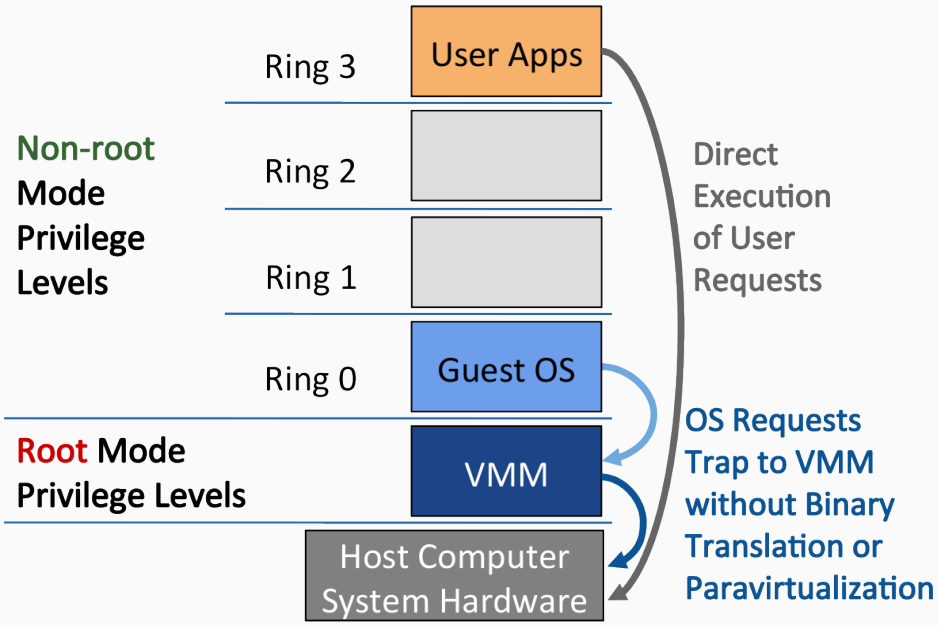
Paravirtualization vs Dynamic Binary Translation performance comparison



Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

CPU full virtualization: hardware-assisted

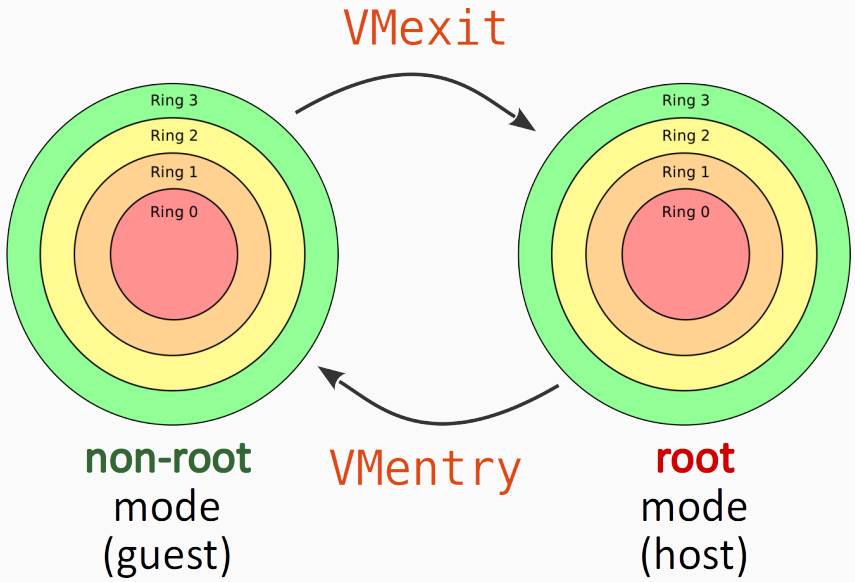
- Also called "Accelerated Virtualization" and "Hardware Virtual Machine" (HVM)
- Exists since the release of Intel VT-x & AMD-V Pacifica (2005):
  - close "holes" in x86 ISA (Instruction Set Arch.)
  - adds new modes: root*/non-root



*Completely unrelated to root user in Linux/UNIX!

CPU hardware-assisted virtualization, root/non-root modes

- In non-root mode, certain privileged operations cause traps (VMexits) $\rightarrow$ trigger switch to root mode (VMM)
- A VMM could use the following:
  - non-root (guest):
    - ring 3: applications
    - ring 0: OS
  - root (VMM):
    - ring 0: VMM



Platform virtualization: device virtualization

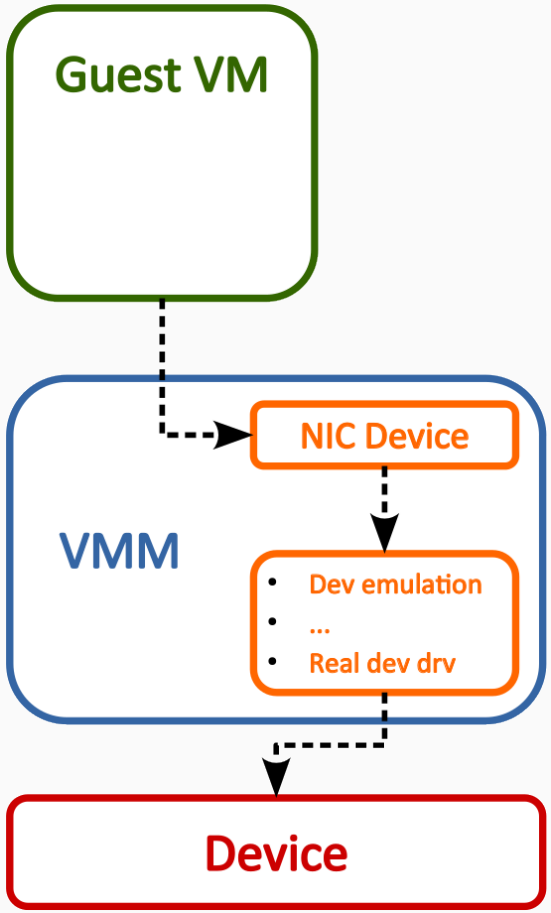
Device virtualization

Three models for device virtualization (pre VT-d hardware):

- Emulated
- Paravirtualized
- Passthrough

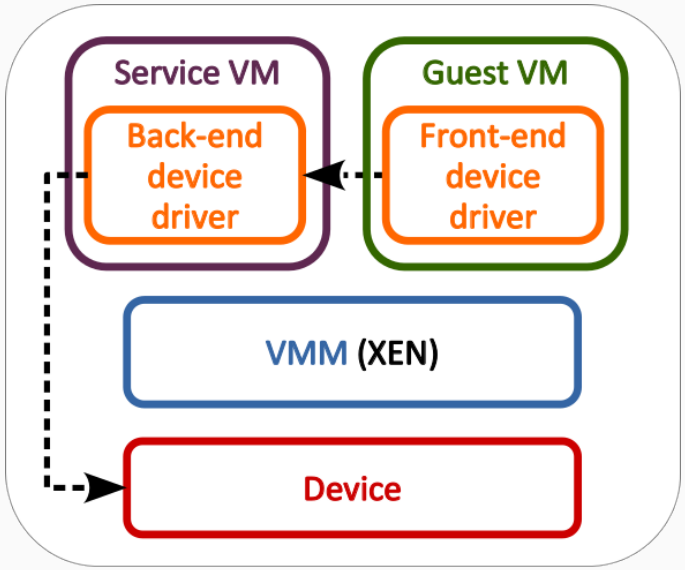
Device virtualization: emulated

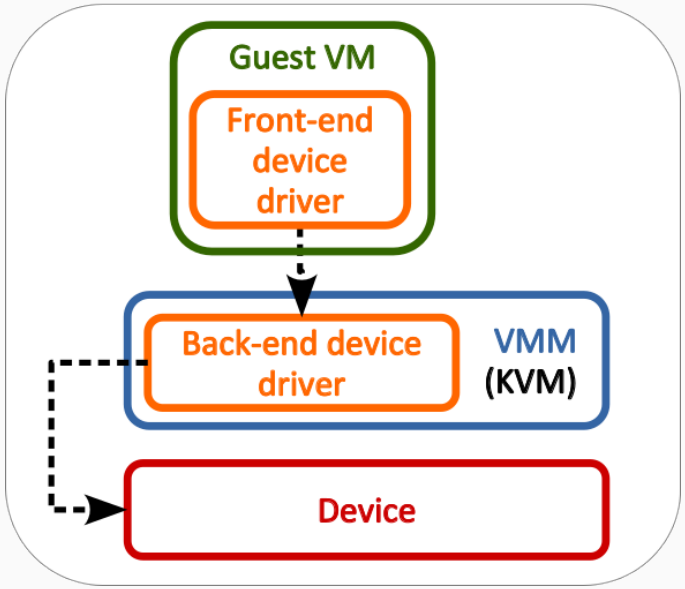
- VMM **intercepts** all device accesses
- VMM **emulates** an existing device that's likely not physically present on the host
- **Pros**
  - VM decoupled from physical device
  - device sharing
  - VM migration
- **Cons**
  - low performance
  - emulation can be complex to implement



Device virtualization: paravirtualized

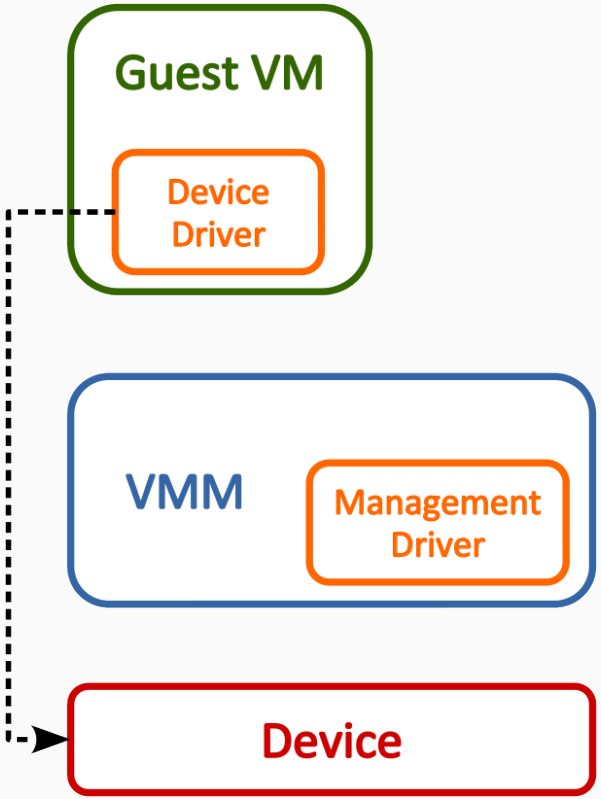
- Guest VM presents a virtual device
- Device access control **split** between:
  - **front-end** driver in guest VM
  - **back-end** driver in VMM
  - guest VM & VMM use hypercalls and shared memory to communicate
- **Pros**
  - VM decoupled from physical device
  - device sharing
  - VM migration
  - no need to emulate a real device
  - easy to implement
  - high performance
- **Cons**
  - requires dedicated drivers in Guest OS





Device virtualization: passthrough

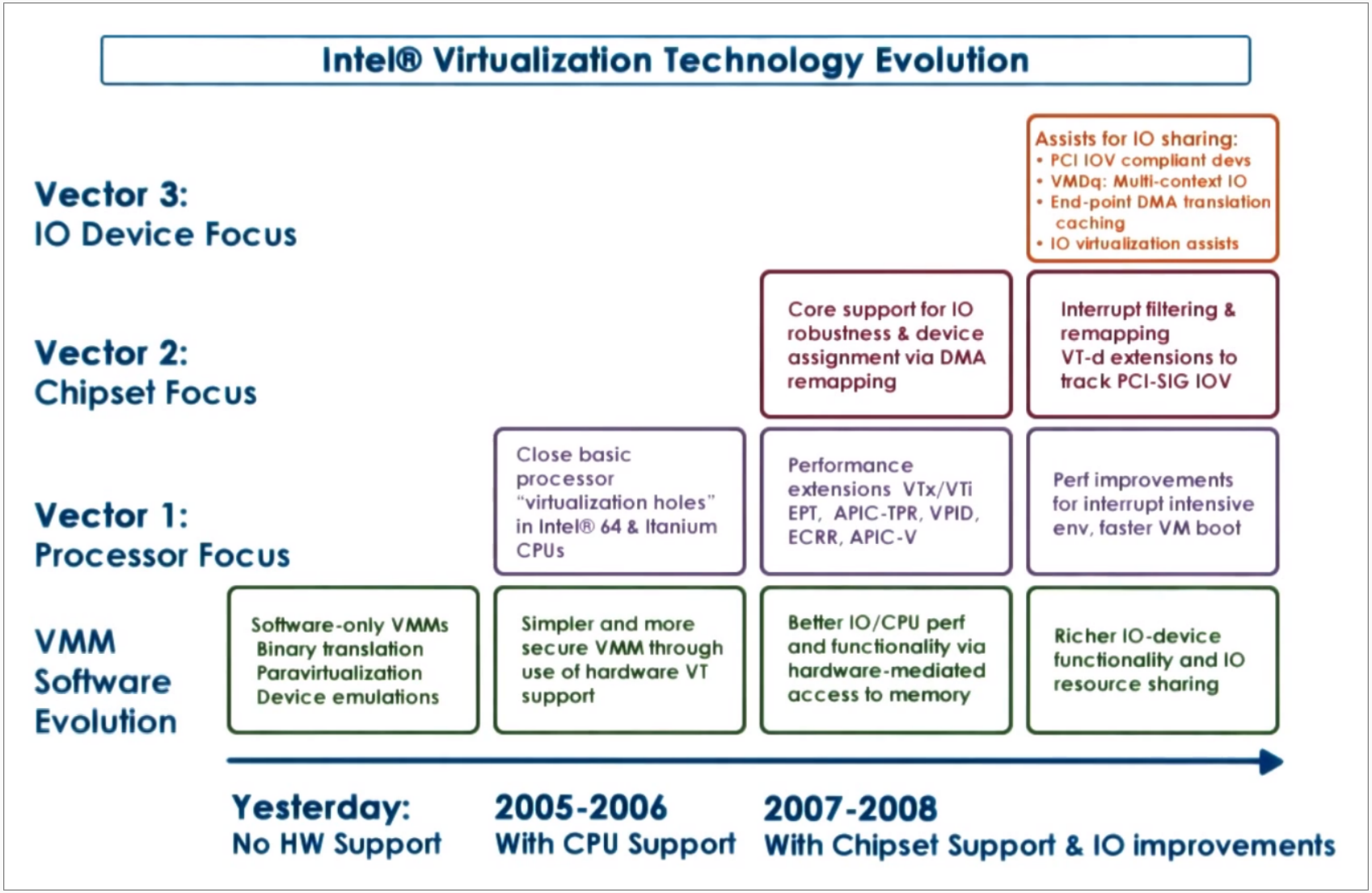
- VMM gives guest VM exclusive direct access to physical device
- **Pros**
  - native (highest) performance
- **Cons**
  - device cannot be shared (or very difficult)
  - VM migration difficult
  - host must have the exact device type expected by the guest VM



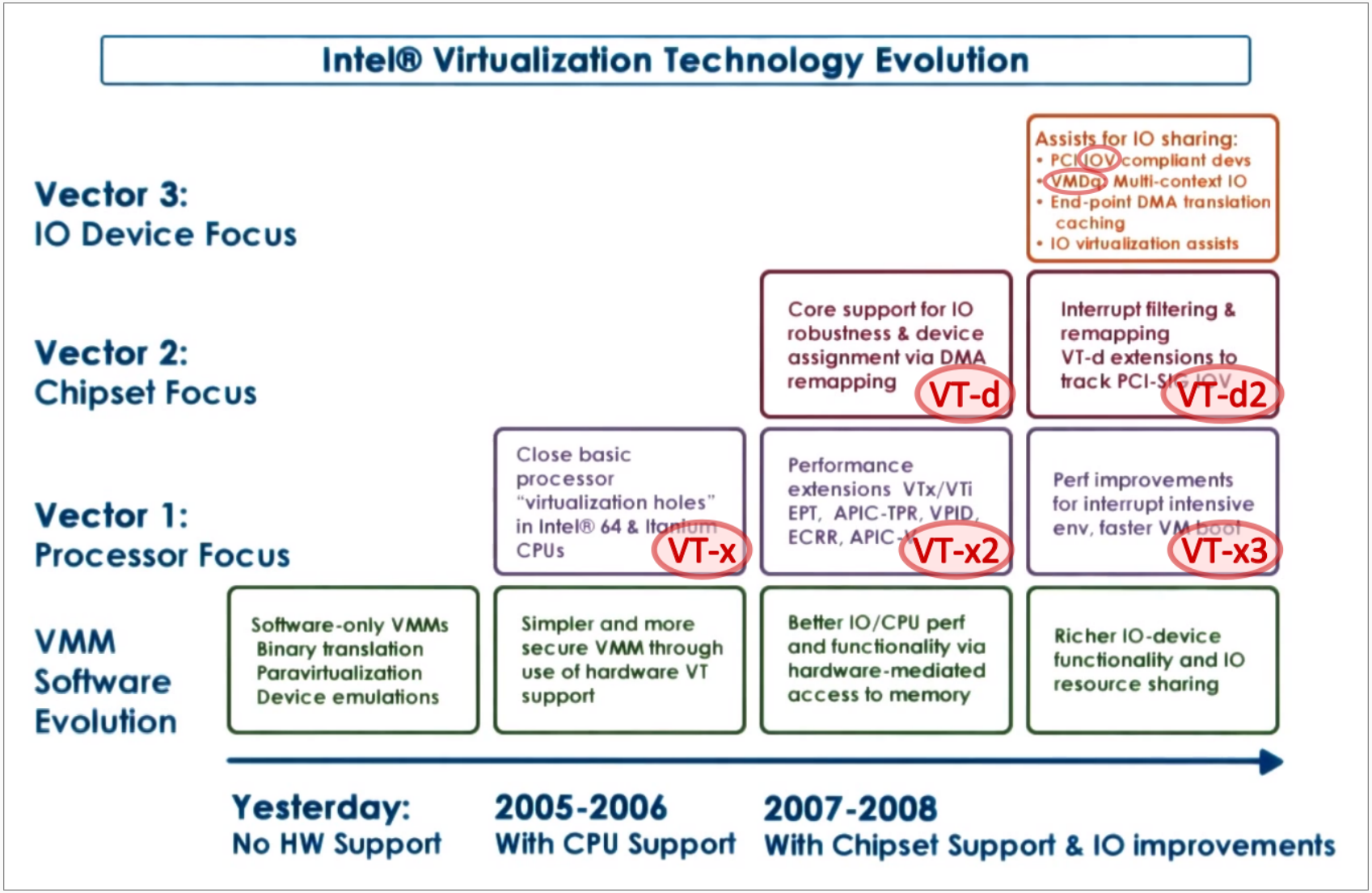
Hardware accelerated device virtualization

- Intel VT-d and AMD-V hardware expose a new feature for virtualizing I/O: IOMMU
- IOMMU creates a virtual address space for devices and sets limits on a device's range of addressable memory
  - allow VMM to manage and regulate DMA from devices
  - VMM can assign devices to guests and restrict devices' memory access to specific pages
- PCI-Passthrough uses IOMMU isolation for protection
- IOMMU similar to MMU in CPUs but for devices
  - generates exceptions which VMM must handle

x86 Intel-VT revolution



x86 Intel-VT revolution



Hypervisor models

Hypervisor models

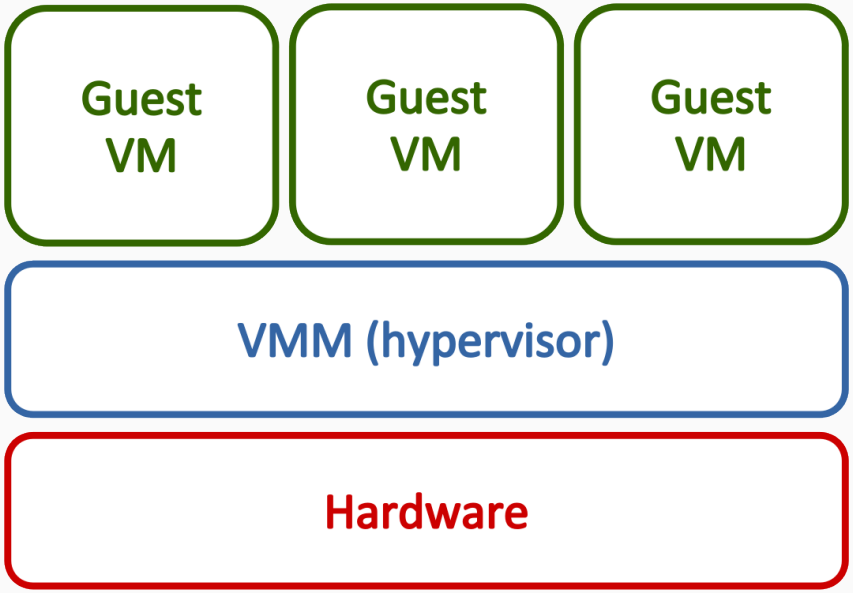
Two hypervisor models:

- **Bare-metal** (also called **type 1** hypervisors)
- **Hosted** (also called **type 2** hypervisors)

Bare-metal (type 1) virtualization model

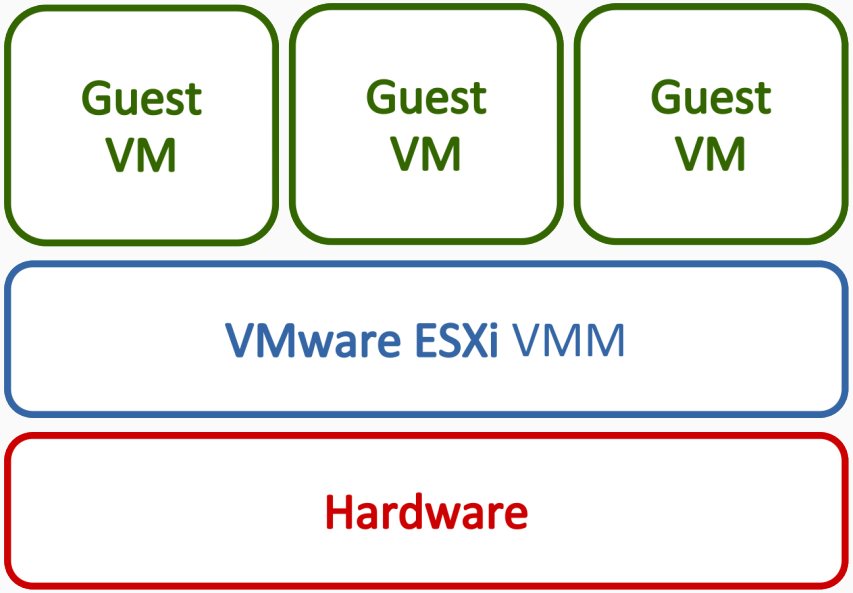
Bare-metal = hypervisor runs above hardware (metal)

- VMM manages all hardware resources and supports execution of VMs → hypervisor must manage all possible devices (drivers)



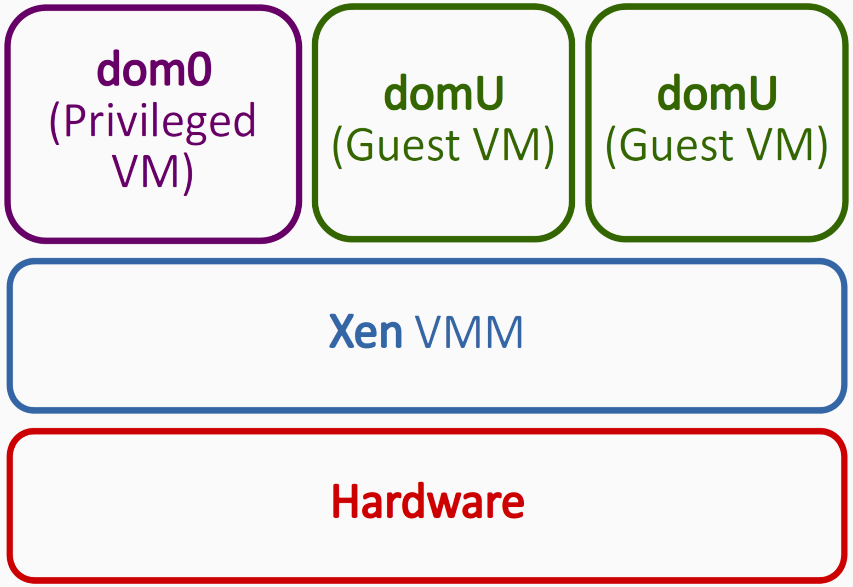
Bare-metal model, variant 1: VMware ESXi

- VMM manages all hardware resources and supports execution of VMs
- Device manufacturers provide device drivers for the hypervisor
- **Limited** hardware support: ESXi only available for dedicated servers
- First released in 2001



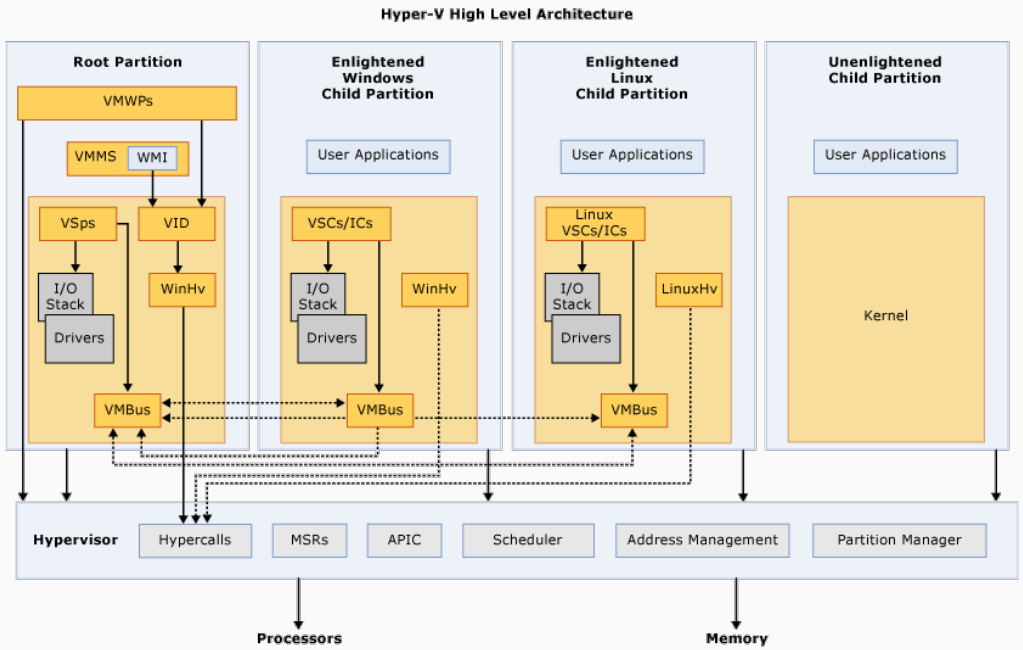
Bare-metal model, variant 2: Xen

- VMM manages all hardware resources and supports execution of VMs
- Create a privileged VM (Dom0) that runs Linux OS with full hardware privileges
  - run all device drivers
  - manage unprivileged VMs (DomU)
  - system config and management, e.g. how VMM share resources across VMs
  - uses QEMU for emulation of physical devices
- First released in 2003



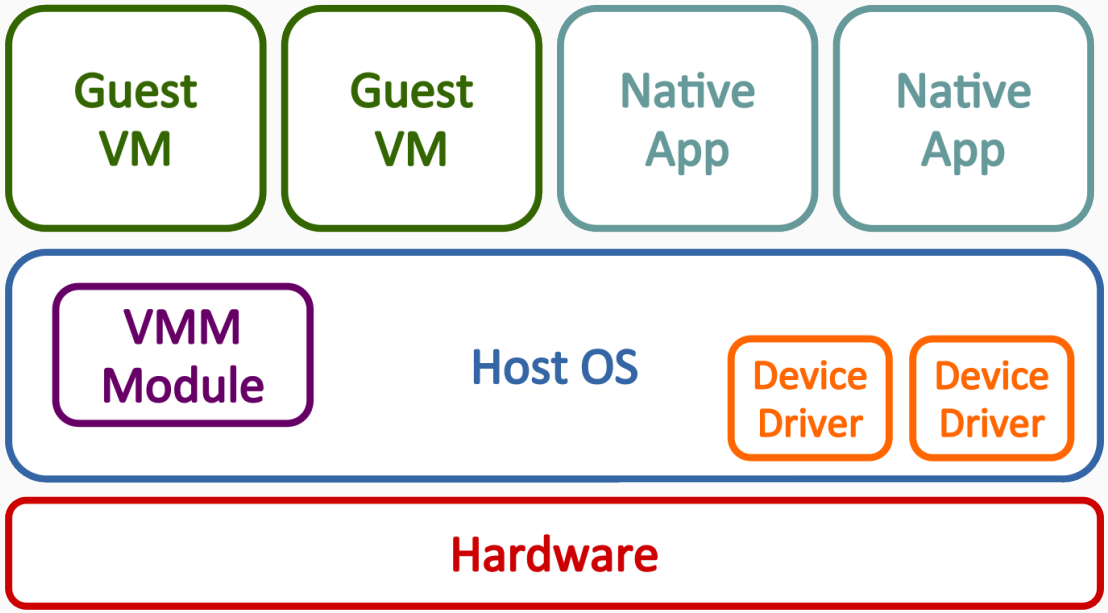
Bare-metal model, variant 2: Microsoft Hyper-V

- VMM manages all hardware resources and supports execution of VMs
- Parent partition runs Windows Server OS with full hardware privileges
  - direct access to hardware devices
  - create/manage child partitions which host VMs with guest OSes
  - system config and management
- First released in Windows Server 2008



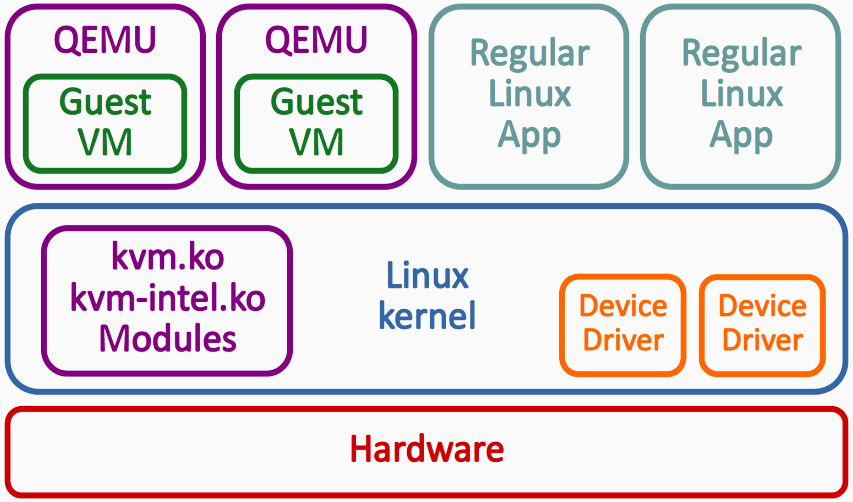
Hosted (type 2) virtualization model

- Host OS owns all hardware
- Special VMM module (driver) provides hardware interfaces to VMs and deals with VM context switching



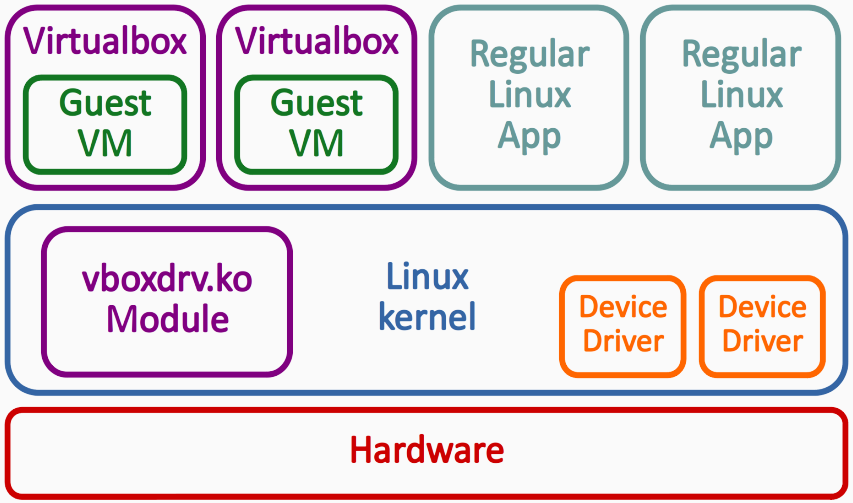
Hosted virtualization example: KVM

- KVM = Kernel Virtual Machine
- KVM is a Linux kernel module
- Linux host provides hardware management + runs regular Linux applications
- Guest VMs support through KVM + QEMU (platform emulator + virtualizer)
- Massive Linux kernel re-use
- Leverages advances in Linux open-source community
- First released in 2006



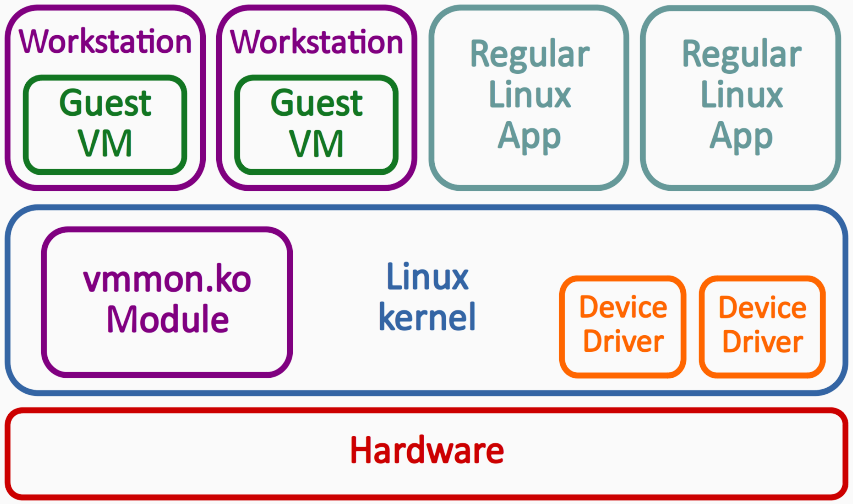
Hosted virtualization example: VirtualBox

- Similar to KVM + QEMU
- Uses its own kernel module which provides similar features to KVM
- Original version of Virtualbox used parts of QEMU code
- First released in 2007



Hosted virtualization example: VMware Workstation

- Similar to KVM + QEMU and Virtualbox
- Uses its own kernel module which provides similar features to KVM
- First released in 1999



Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX
  - hosted

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX
  - hosted
- Citrix Hypervisor

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX
  - hosted
- Citrix Hypervisor
  - bare-metal

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX
  - hosted
- Citrix Hypervisor
  - bare-metal
- VMware Player

Quiz: bare-metal or hosted

Among the following software, which ones are **bare-metal** (type 1) hypervisors and which ones are **hosted** (type 2)?

- FreeBSD bhyve
  - hosted
- VMware Fusion
  - hosted
- Amazon Web Services
  - originally bare-metal, then hosted
- Apple Hypervisor for OSX
  - hosted
- Citrix Hypervisor
  - bare-metal
- VMware Player
  - hosted

What's in use out there?

- Amazon AWS & EC2 $\rightarrow$ from Xen to Nitro which is a stripped-down version of Linux/KVM
- Microsoft Azure $\rightarrow$ Hyper-V
- Google Cloud Platform $\rightarrow$ Linux/KVM
- Most cloud providers (e.g. DigitalOcean) $\rightarrow$ Linux/KVM
- Everything OpenStack $\rightarrow$ generally powered by Linux/KVM
- VMware remains leader in solutions for business/private environments

Resources

- "Bringing Virtualization to the x86 Architecture with the Original VMware Workstation"; E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, E. Wang; ACM Transactions on Computer Systems, 2012
- "Virtual Machine Monitors" from “Operating Systems: Three Easy Pieces”; Remzi H. et Andrea C. Arpaci-Dusseau; Arpaci-Dusseau Books, 2020
- “Hardware and Software Support for Virtualization”; E. Bugnion, J. Nieh, D. Tsafrir; Morgan & Claypool Publishers, 2017
- “Virtual Machines: Versatile Platforms for Systems and Processes”; J. Smith, R. Nair; Morgan Kaufmann, 2005
- Xen and the Art of Virtualization, P. Barham et al., ACM SIGOPS Operating Systems Review, Volume 37, Issue 5, 2003
- Virtualization in Xen 3.0
- Hyper-V Architecture