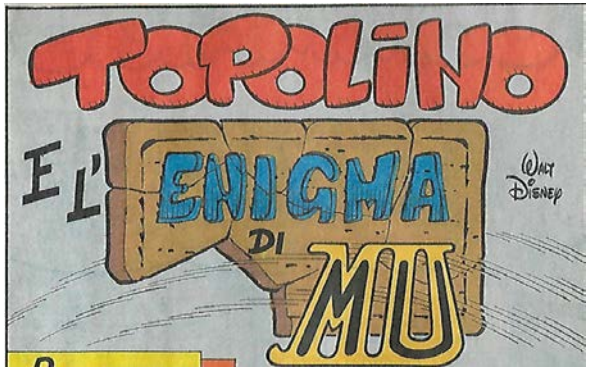
OPERATING SYSTEMS

VIRTUAL MACHINES, CLOUD COMPUTING, IoT



Topolino 1238 - 19.08.1979

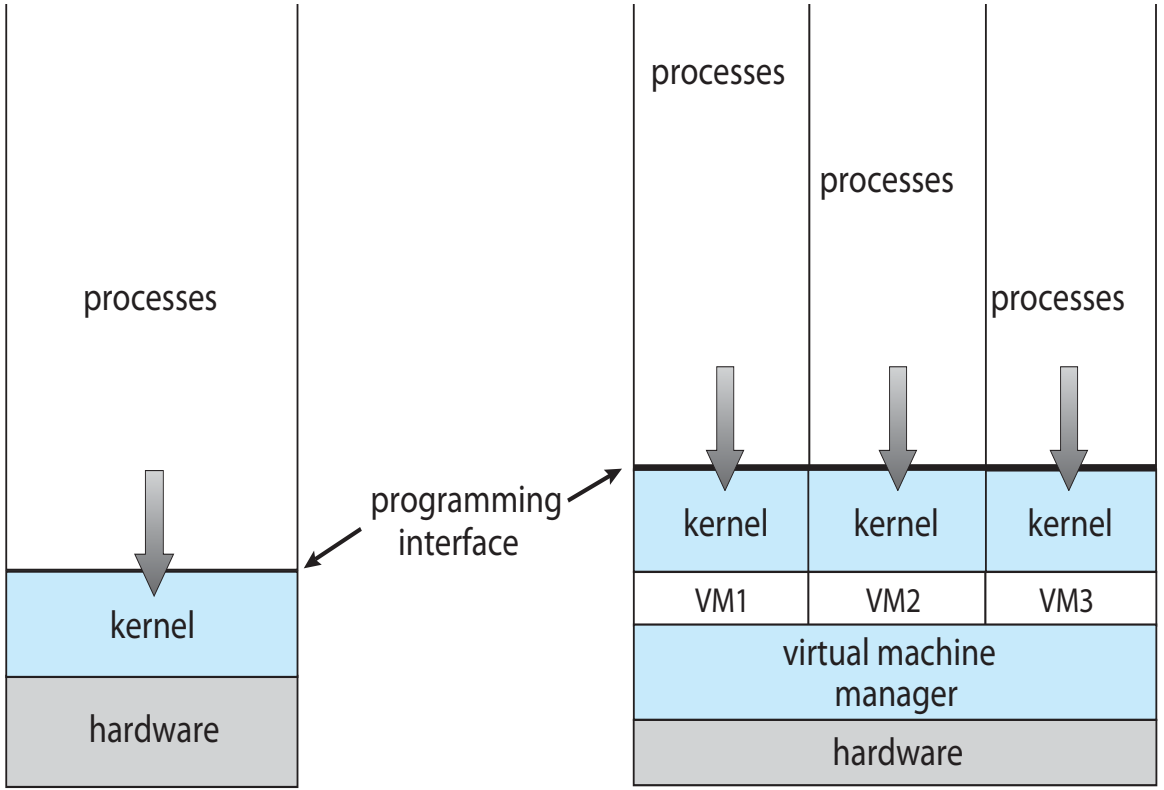
https://www.topolino.it/character/professor-zapotec/

Overview

- Virtualization *abstracts* hardware of a single computer into several different execution environments
  - Similar to the layered approach, each layer creating a virtual system on which OS or applications can run
- Components
  - **Host** – underlying hardware system
  - **Virtual machine manager (VMM) or hypervisor** – creates and runs virtual machines by providing an interface that is identical to the host
  - **Guest** – process provided with virtual copy of the host, usually an operating system

A single physical machine can run multiple operating systems concurrently, each in its own virtual machine

System Model



Virtual Machines (VM)

Virtual Machines

- A Virtual Machine is a **software construct** that mimics the characteristics of a physical server
  - It is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports
- A VM can be powered on like a physical server, loaded with an operating system and software solutions, and utilized in the manner of a physical server
  - Unlike a physical server, this virtual server only sees the resources it has been configured with, not all of the resources of the physical host itself

Virtual Machines Files

- The configuration file describes the attributes of the virtual machine
  - It contains the server definition, how many virtual processors (vCPUs) are allocated to this virtual machine, how much RAM is allocated, which I/O devices the VM has access to, how many network interface cards (NICs) are in the virtual server, and more
  - It also describes the storage that the VM can access
- When a virtual machine is powered on, or instantiated, additional files are created for logging, for memory paging, and other functions
  - Since VMs are already files, copying them produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself

History

- Virtualization first introduced by IBM in the CP-40 research systems in 1967.
- First appeared in **IBM mainframes in 1972**
  - Allowed multiple users to share a batch-oriented system
- Since then ,virtualization is a feature of mainframe IBM operating systems, currently named z/OS

History

- Formal definition of virtualization helped move it beyond IBM
  - A VMM provides an environment for programs that is essentially identical to the original machine
  - The VMM is in complete control of system resources
- In late 1990s Intel CPUs fast enough for researchers to try virtualizing on general purpose PCs
  - Xen and VMware created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs
- Intel x86 architectures (IA-32 e IA-64) provide support to virtualization
  - Intel: VT-x
    AMD: AMD-V

Key Reasons for Using Virtualization

- **Consolidation**
  - A large-capacity or high-speed resource can be used more efficiently by sharing the resource among multiple applications simultaneously
- **Aggregating**
  - Virtualization makes it easy to combine multiple resources in to one virtual resource, such as in the case of storage virtualization
- **Dynamics**
  - Hardware resources can be easily allocated in a dynamic fashion, enhancing load balancing and fault tolerance
- **Ease of management**
  - Virtual machines facilitate deployment and testing of software
- **Increased availability**
  - Virtual machine hosts are clustered together to form pools of computing resources

Key Reasons for Using Virtualization

- **Legacy hardware**
  - Applications built for legacy hardware can still be run by virtualizing the legacy hardware, enabling the retirement of the old hardware
- **Rapid deployment**
  - A new VM may be deployed in a matter of minutes
- **Versatility**
  - Hardware usage can be optimized by maximizing the number of kinds of applications that a single computer can handle

Benefits and Features

- **Host system protected from VMs, VMs protected from each other**
  - Sharing is provided though via shared file system volume, network communication
- **Freeze, suspend, running VM**
  - A VM can be moved or copied somewhere else and resume
    - Snapshot of a given state, able to restore back to that state
- **Run multiple, different OSes on a single machine**
  - Consolidation, app dev, research, etc.

App Development

- **Templating**
  create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- **Live migration**
  move a running VM from one host to another with no interruption of user access
- **Cloud computing** is a consequence of virtualization
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops

Virtualization Technologies

Definitions

- **Hardware virtualization**
  - A software-defined virtual hardware environment
  - It is supported by the underlying hardware platform

- **Application-level virtualization**
  - often referred to as *run-time systems*
  - allows the execution of a program on a virtualexecution environment that is not a replica of a hardware platform

  JAVA

  .NET

Hardware virtualizarion

- **Complete virtualization**
  - each virtual environment is identical to the underlying hardware
- **Emulation**
  - A legacy or special-purpose CPU (e.g., an embedded systems) is emulated on a different environment (e.g., a workstation)
    QEMU (https://www.qemu.org) is one of the most popular processor emulator
- **Container**
  Virtualization at the operating system level, that creates isolated execution environments

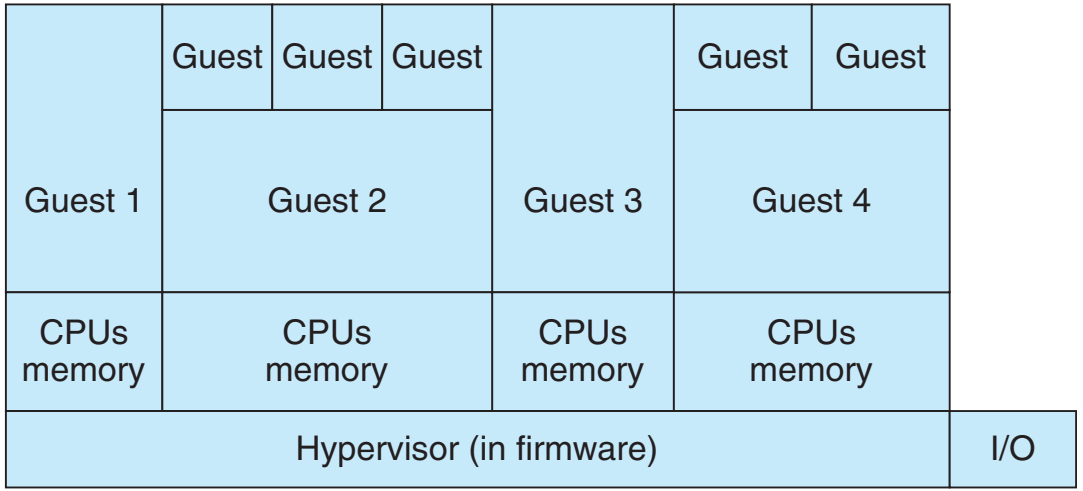
Implementation of Virtual Machine Managers

Implementation of VMMs

- **Type 0 hypervisors**
  Hardware-based solutions that provide support for virtual machine creation and management via firmware
  - IBM LPARs and Oracle LDOMs are examples
- **Type 1 hypervisors**
  Operating-system-like software built to provide virtualization
  - Including VMware ESX, Xen
  - Also include general-purpose OS that provide standard functions as well as VMM functions, such as Microsoft Windows with Hyper-V and RedHat Linux with KVM
- **Type 2 hypervisors**
  Applications that run on standard operating systems but provide VMM features to guest operating systems
  - Including VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

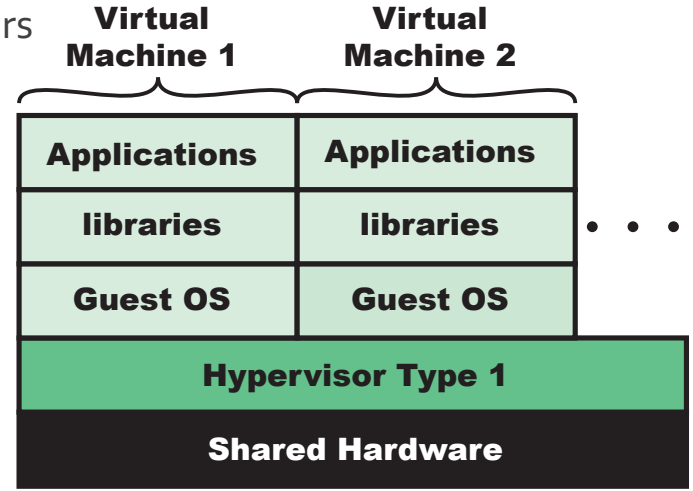
Type 0 Hypervisor

- Old idea, under many names by HW manufacturers
  - "partitions", "domains"
  - a HW feature implemented by firmware
  - Each guest has dedicated HW



Type 1 Hypervisor

- **Special purpose operating systems that run natively on HW**
  - **Rather than providing system call interface, create run and manage guest OSes**
  - Run in kernel mode
  - Guests generally don't know they are running in a VM
  - Implement device drivers for host HW because no other component can
  - Also provide other traditional OS services like CPU and memory management

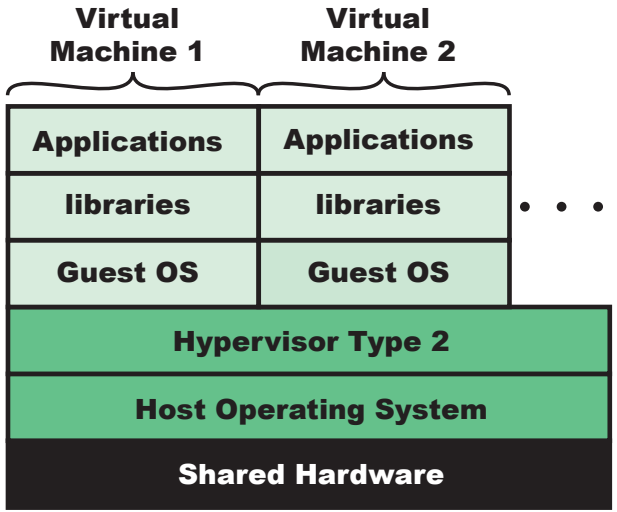


Type 1 Hypervisor

- **Another variation is a general-purpose OS that also provides VMM functionality**
  - RedHat Enterprise Linux with KVM, Windows with Hyper-V, Oracle Solaris
  - Perform normal duties as well as VMM duties
  - Typically less feature rich than dedicated Type 1 hypervisors
- In many ways, **treat guest OS as just another process**

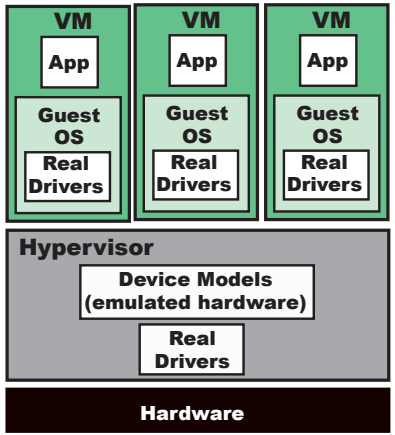
Type 2 Hypervisor

- The aim is to let the user run different *guest* operating systems within a *host* operating system
  - the emphasis is not on performances.
    Allows a user of a specific OS to occasionally run applications built for a different OSs
    - VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox
- VMM is simply another process, run and managed by host
  - Even the host doesn't know that a VMM is running guests
- Very little OS involvement in virtualization

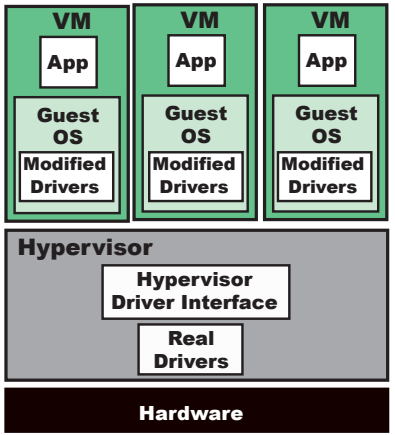


Paravirtualization

- A **software assisted virtualization technique** that uses specialized APIs to link virtual machines with the hypervisor
  - The OS in the virtual machine has specialized paravirtualization support as part of the kernel, as well as specific paravirtualization drivers



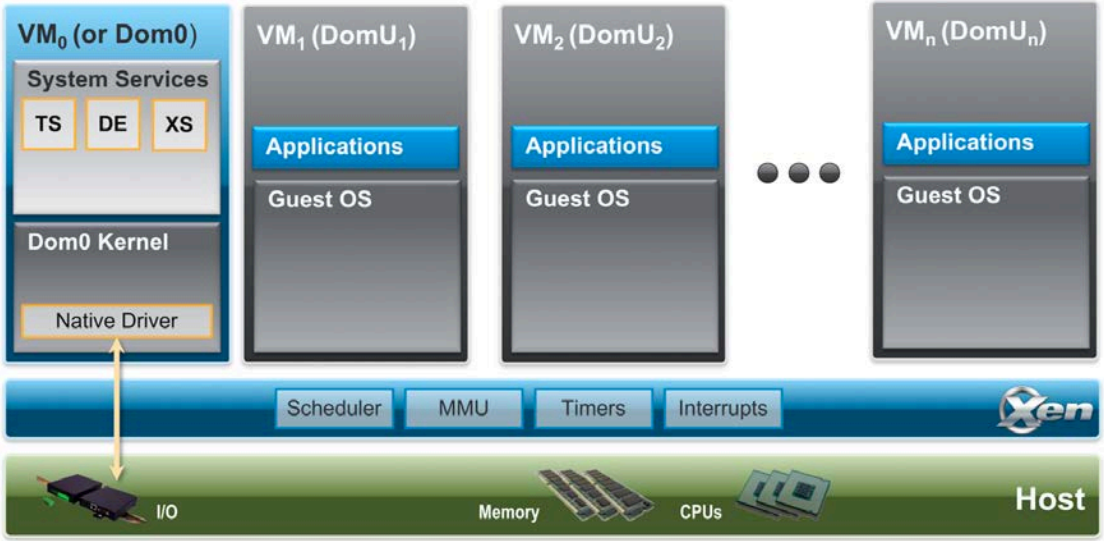
(a) Type 1 Hypervisor



(b) Paravirtualized Type 1 Hypervisor with Paravirtualized Guest OSs

Paravirtualization example

- The kernel of the guest OS is modified so that some system calls are replaced by hypervisor calls



Programming Environment Virtualization

- Also not-really-virtualization
- Programming language is designed to run within custom-built virtualized environment
  - for example, Oracle Java has many features that depend on running in Java Virtual Machine (JVM)
- Similar to interpreted languages

Application Containment

- Achieve the goals of virtualization without full-fledged virtualization
  - Segregation of apps, performance and resource management, easy start, stop, move, and management of applications
- Example: Oracle containers / zones create a virtual layer between OS and apps
  - Only one kernel running – host OS
  - OS and devices are virtualized, providing resources within zone
  - Each zone has its own applications; networking stack, addresses, and ports; user accounts, etc.
  - CPU and memory resources divided between zones
    - Zone can have its own scheduler to use those resources

Container Virtualization

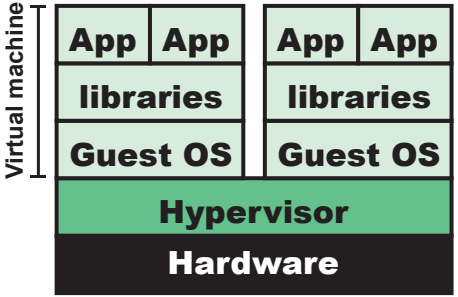
Container virtualization is a relatively recent approach to virtualization

In this approach, software, known as a **virtualization container,** runs on top of the host OS kernel and provides an isolated execution environment for applications

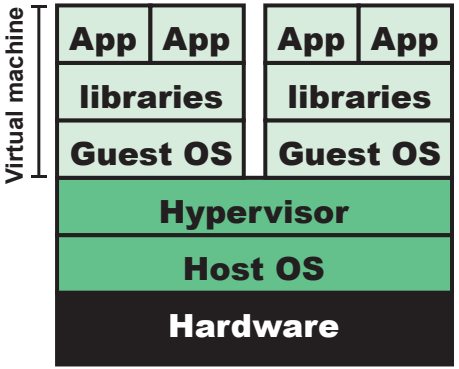
Unlike hypervisor-based VMs, containers do not aim to emulate physical servers; instead, all containerized applications on a host share a common OS kernel

This eliminates the resources needed to run a separate OS for each application and can greatly reduce overhead

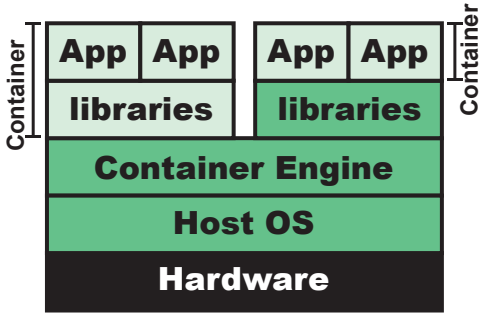
Comparison of Virtual Machine and Containers



(a) Type 1 Hypervisor

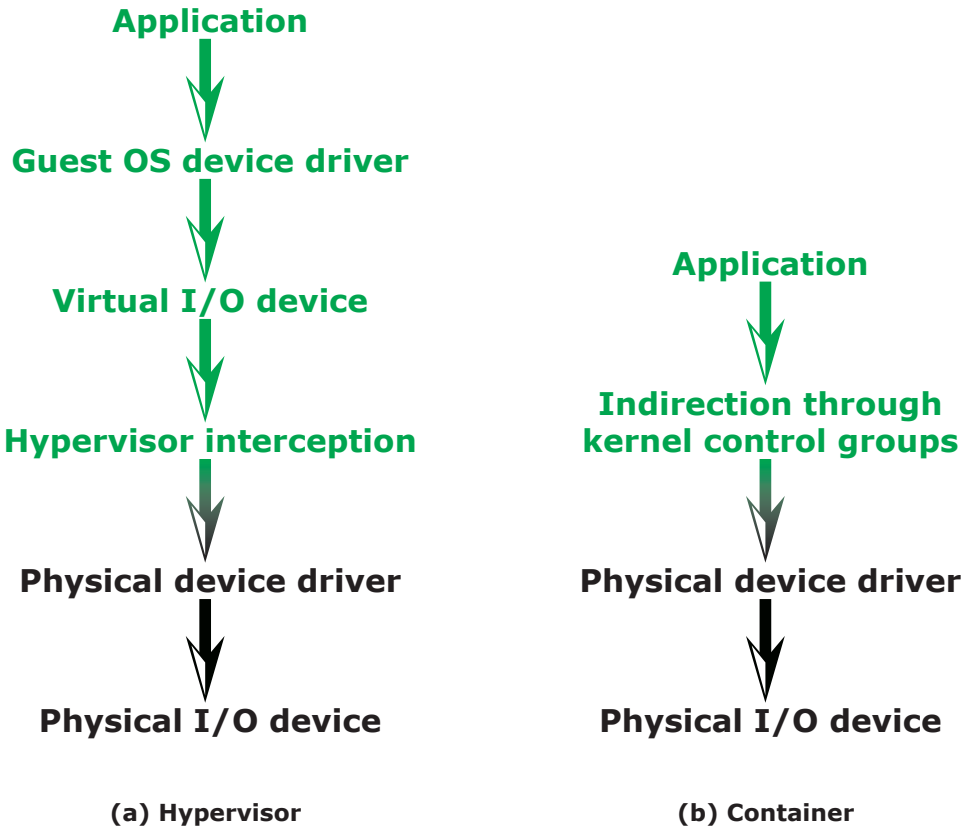


(b) Type 2 Hypervisor



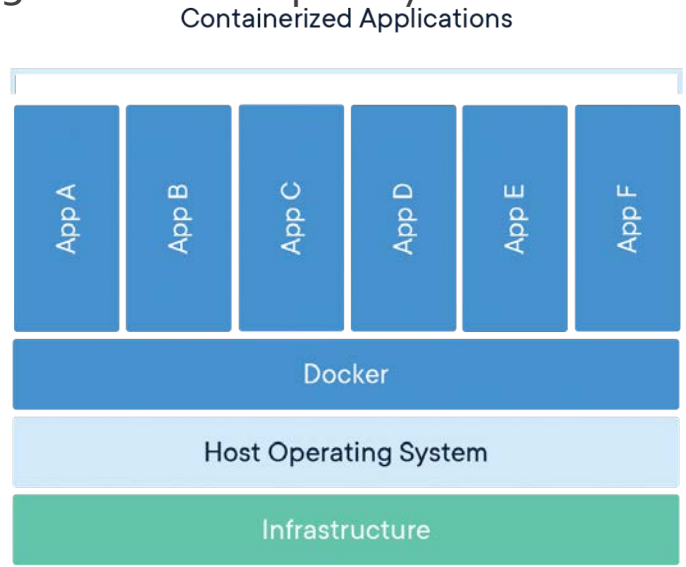
(c) Container

I/O Operation via Hypervisor and Container



Docker

- Provides a simpler and more standardized way to run containers
- It was first started in 2013 and is developed by Docker Inc.
- The popularity of the Docker container is due to its ability to load a container image on a host operating system in a simple and quick manner

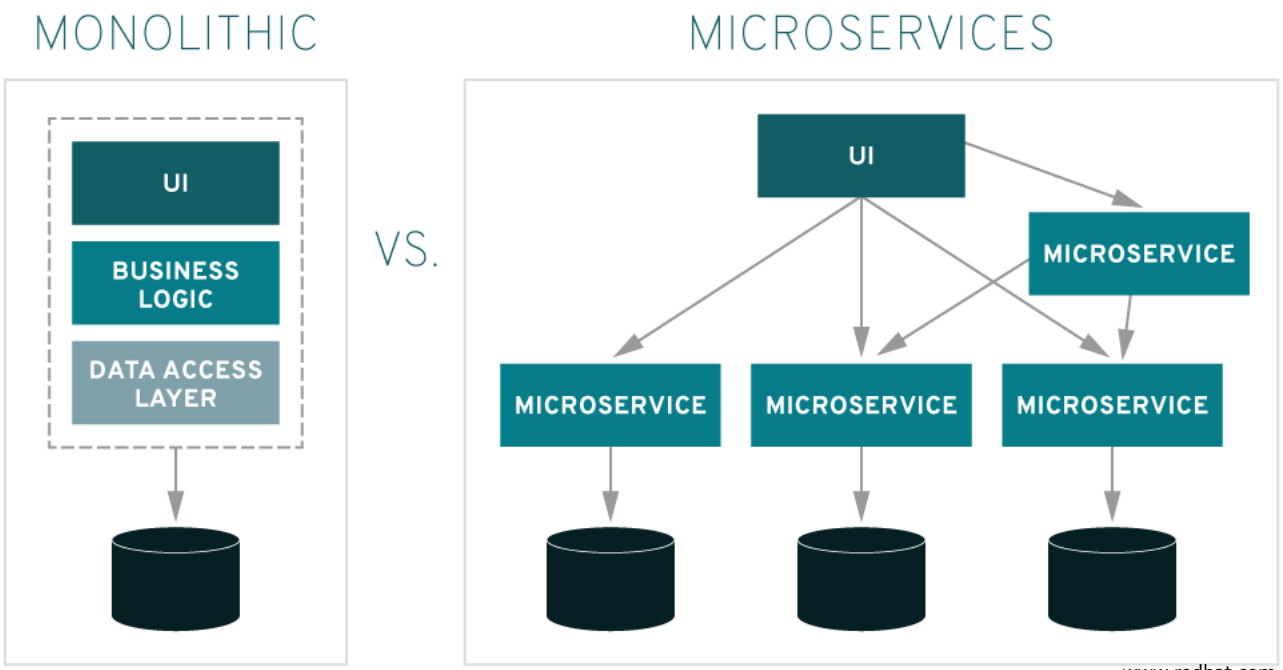


Microservices

- NIST SP 800-180 (NIST Definition of Microservices, Application Containers and System Virtual Machines) defines a microservice as

  a basic element that results from the architectural decomposition of an application's components into loosely coupled patterns consisting of self-contained services that communicate with each other using a standard communication protocol and a set of well-defined APIs, independent of any vendor, product, or technology

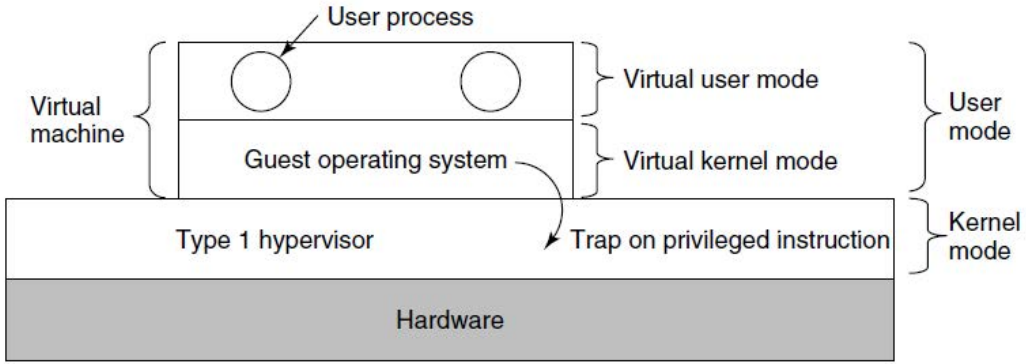
Microservices



Operating Systems issues

CPU modes and kernel execution

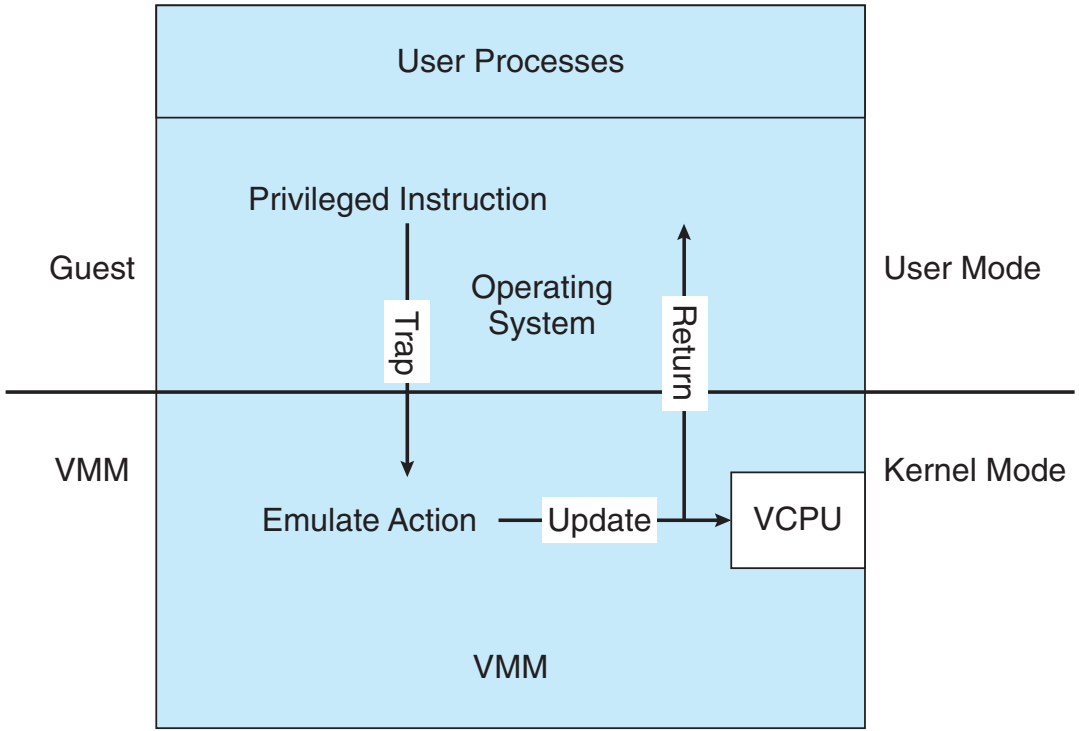
- Dual mode CPU means
  - guest executes in user mode
  - kernel runs in kernel mode
- Not safe to let guest kernel run in kernel mode too
  - VM needs two modes, virtual user mode and virtual kernel mode, both of which run in real user mode



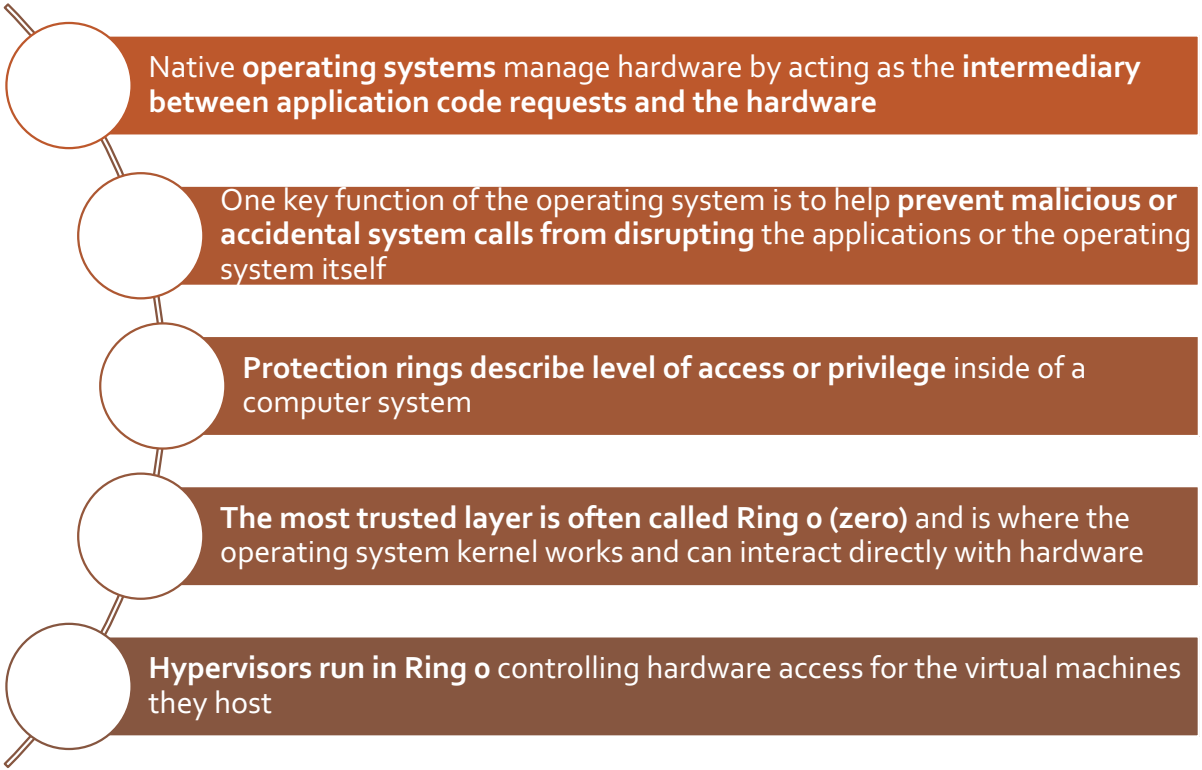
Trap-and-Emulate

- Switch from virtual user mode to virtual kernel mode
  - Attempting a privileged instruction in user mode causes an error -> trap
  - VMM gains control, analyzes error, executes operation as attempted by guest
  - Returns control to guest in user mode
  - Most virtualization products use this at least in part
- user mode code in guest runs at same speed as if not a guest
- kernel mode code runs slower due to trap-and-emulate
- CPUs adding hardware support in the form or more CPU modes to improve virtualization performance

Trap-and-Emulate Virtualization Implementation



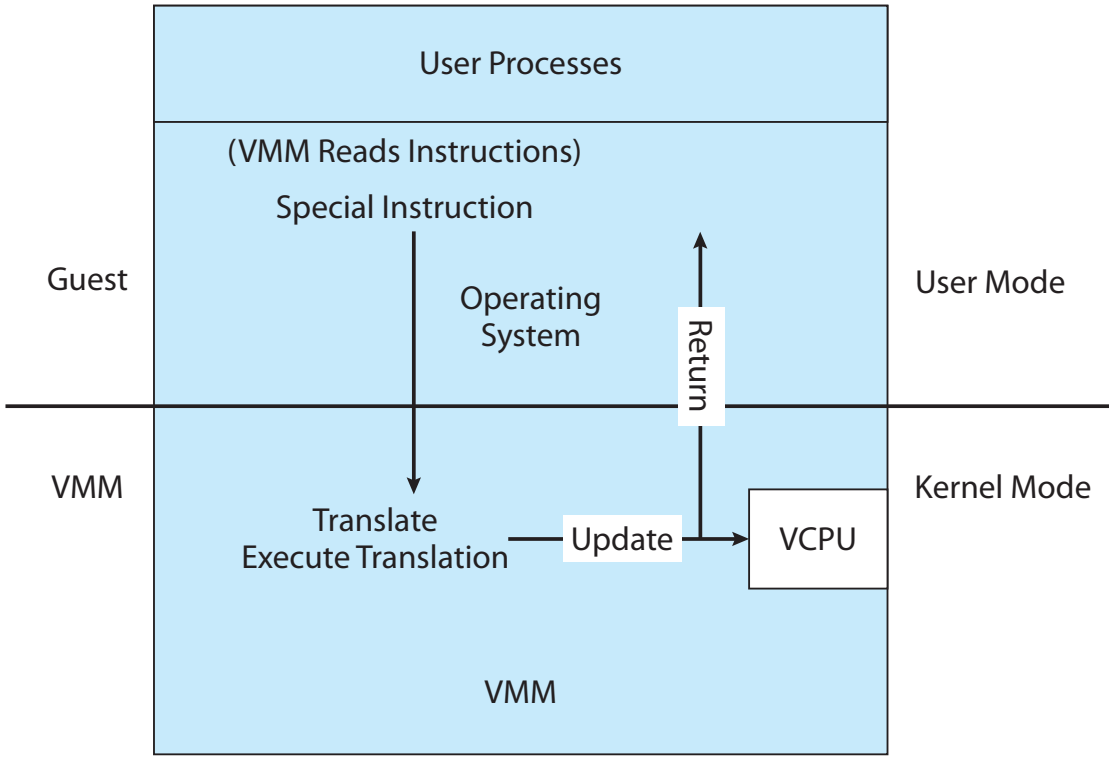
Ring O


Native **operating systems** manage hardware by acting as the **intermediary between application code requests and the hardware**
One key function of the operating system is to help **prevent malicious or accidental system calls from disrupting** the applications or the operating system itself
Protection rings describe level of access or privilege inside of a computer system
The most trusted layer is often called Ring o (zero) and is where the operating system kernel works and can interact directly with hardware
Hypervisors run in Ring o controlling hardware access for the virtual machines they host

Binary Translation

- Some CPUs don't have clean separation between privileged and nonprivileged instructions
  - Earlier Intel x86 CPUs are among them, wbile earliest Intel CPU designed for a calculator
- Trap-and-emulate method considered impossible until 1998
- Binary translation solves the problem
  - Basics are simple, but implementation very complex
  - If guest VCPU is in user mode, guest can run instructions natively
  - If guest VCPU in kernel mode (guest believes it is in kernel mode)
    - VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
    - Special instructions translated into new set of instructions that perform equivalent task

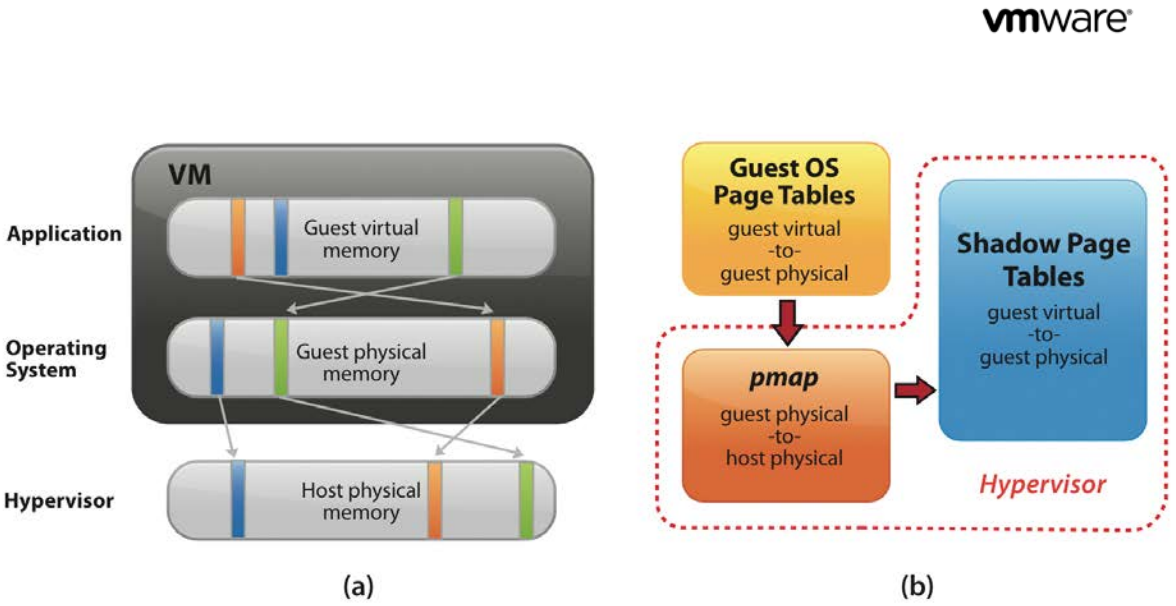
Binary Translation Virtualization Implementation



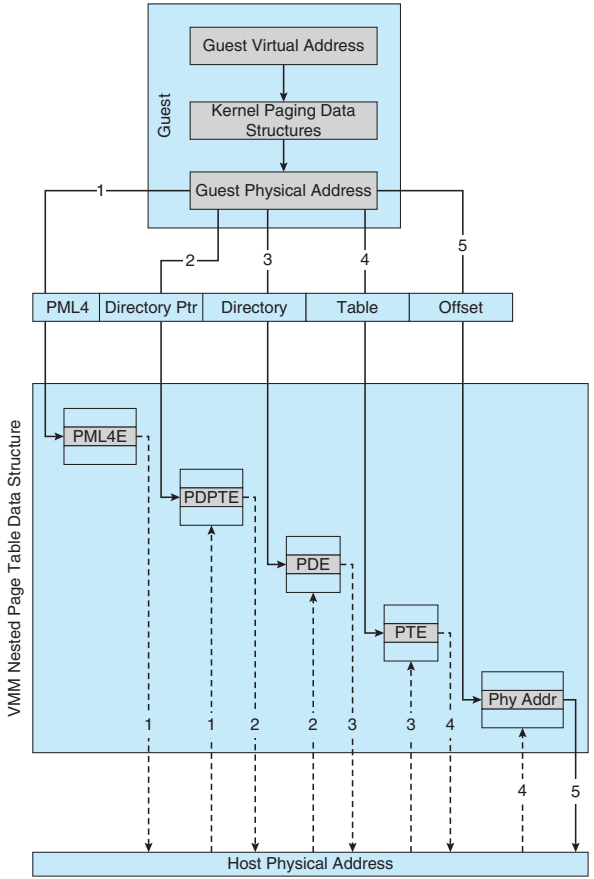
Memory management

- VMM keeps page-table state for guests through Nested Page Tables (NPTs)
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per guest NPTs to represent guest's page-table state
    - Just as VCPU stores guest CPU state
  - When guest on CPU -> VMM makes that guest's NPTs the active system page tables
  - Guest tries to change page table -> VMM makes equivalent change to NPTs and its own page tables
  - Can cause many more TLB misses -> much slower performance

Memory Management in vmware



Nested Page Tables



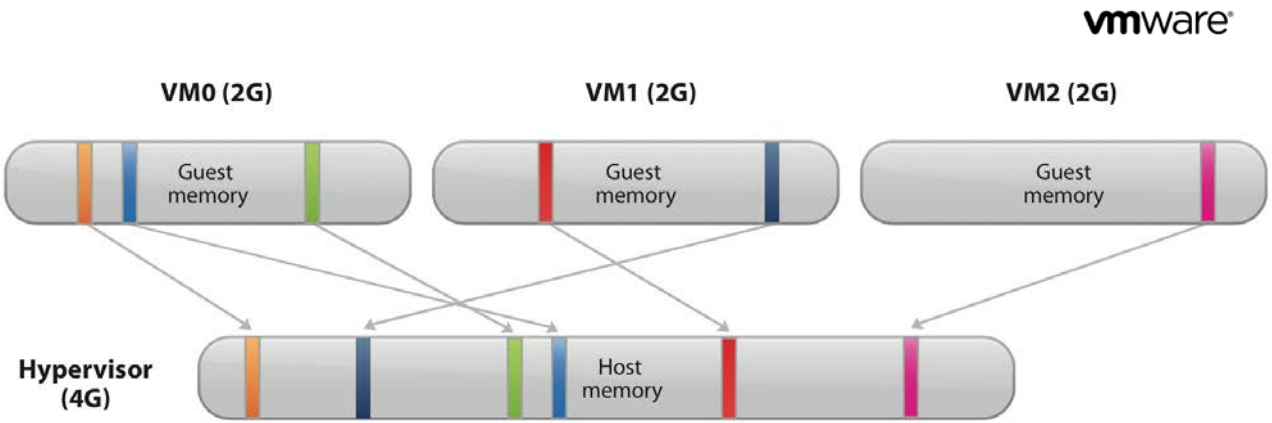
Memory Management
VMware example

- VMware ESX guests have a configured amount of physical memory.
  ESX uses 3 methods of memory management
  - **Double-paging**, in which the guest page table indicates a page is in a physical frame but the VMM moves some of those pages to backing store
  - Install a **pseudo-device driver** in each guest kernel that adds kernel-mode code to the guest
    - **Balloon memory manager** communicates with VMM and is told to **allocate or deallocate memory** to decrease or increase physical memory use of guest, causing guest OS to free or have more memory available

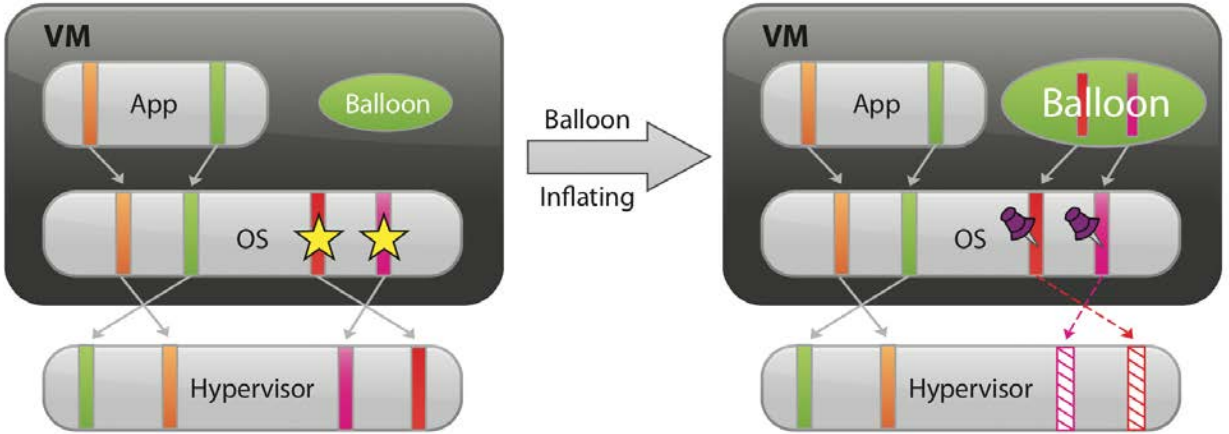
Memory Management

- The virtual machine operating systems are unaware of what is happening in the physical system
- Ballooning
  - The hypervisor activates a balloon driver that (virtually) inflates and presses the guest operating system to flush pages to disk
  - Once the pages are cleared, the balloon driver deflates and the hypervisor can use the physical memory for other VMs
- Memory overcommit
  - The capability to allocate more memory than physically exists on a host

Memory overcommitment



Ballooning



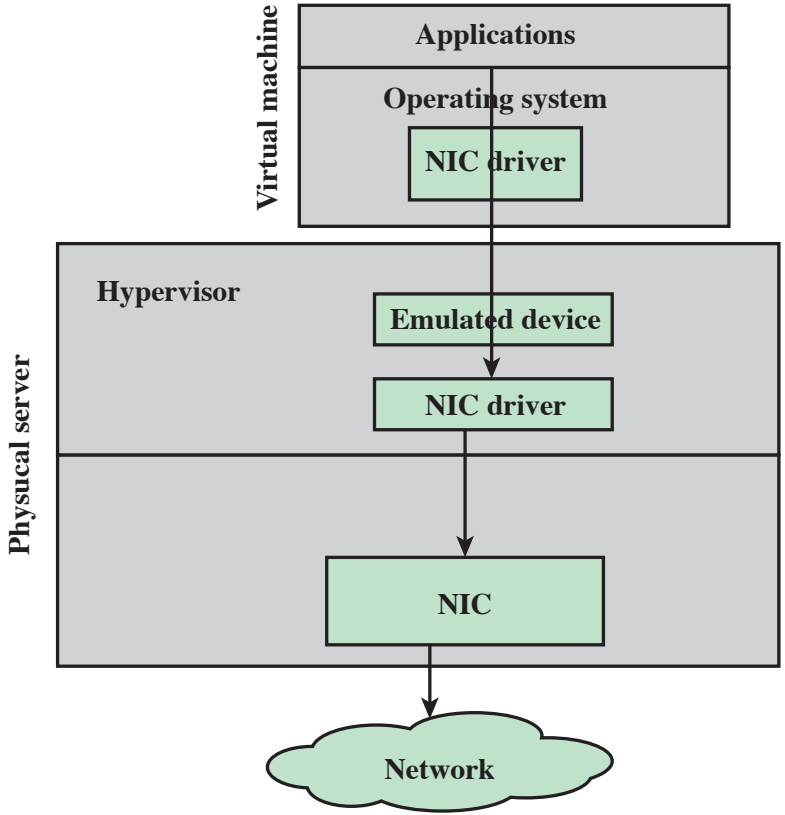
- The VM is assigned 4 frames
- At a certain point of the execution only 2 frames are used by the VM
- The hypervisor cannot see that some frames assigned to the VM are not currently in use

- The *balloon driver* can identify the unused frames, and communicate the addresses to the hypervisor
- The pages can be temporarily used by the hypervisor for requests from other VMs

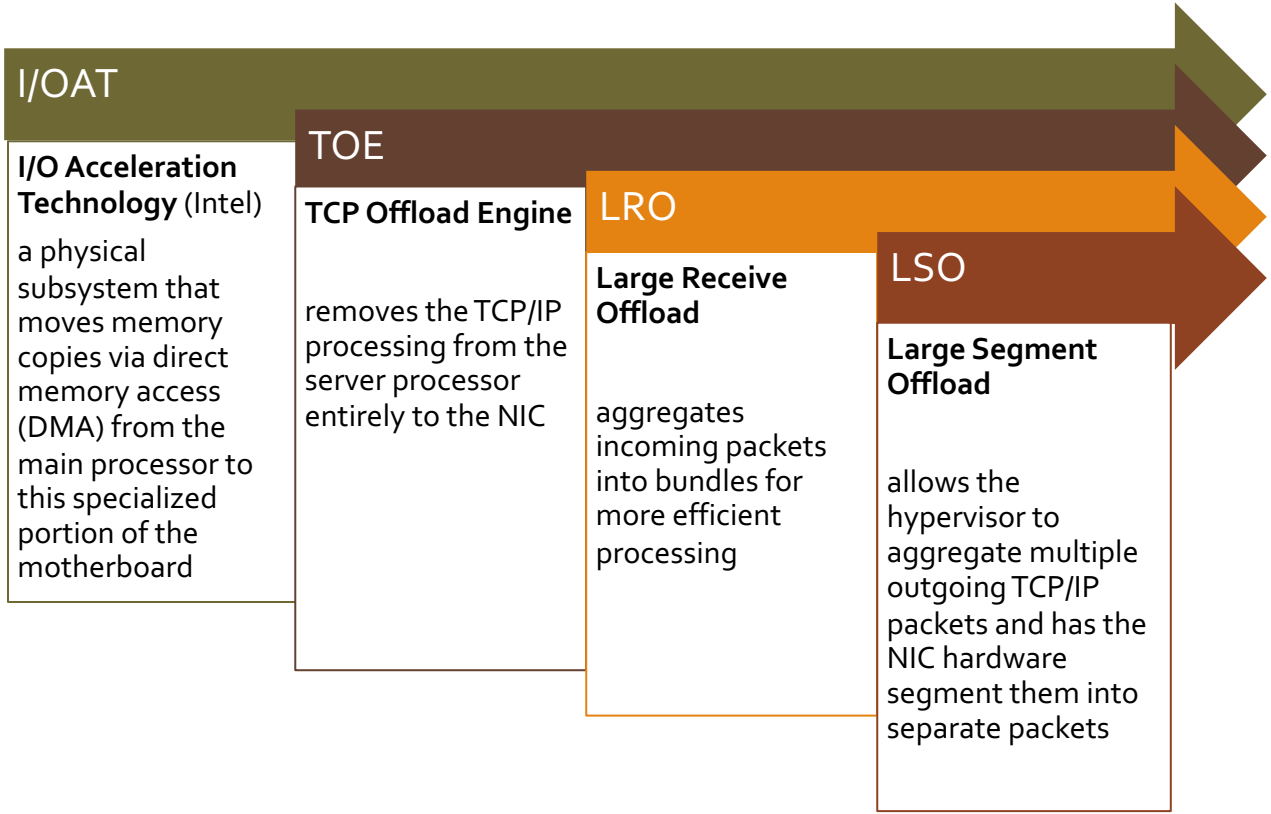
CPU Scheduling

- Generally VMM has one or more physical CPUs and number of threads to run on them
- When enough CPUs for all guests, the VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
- Usually not enough CPUs -> CPU overcommitment
  - VMM can use standard scheduling algorithms to put threads on CPUs
  - Some add fairness aspect

I/O in a virtual environment



I/O Performance improvement



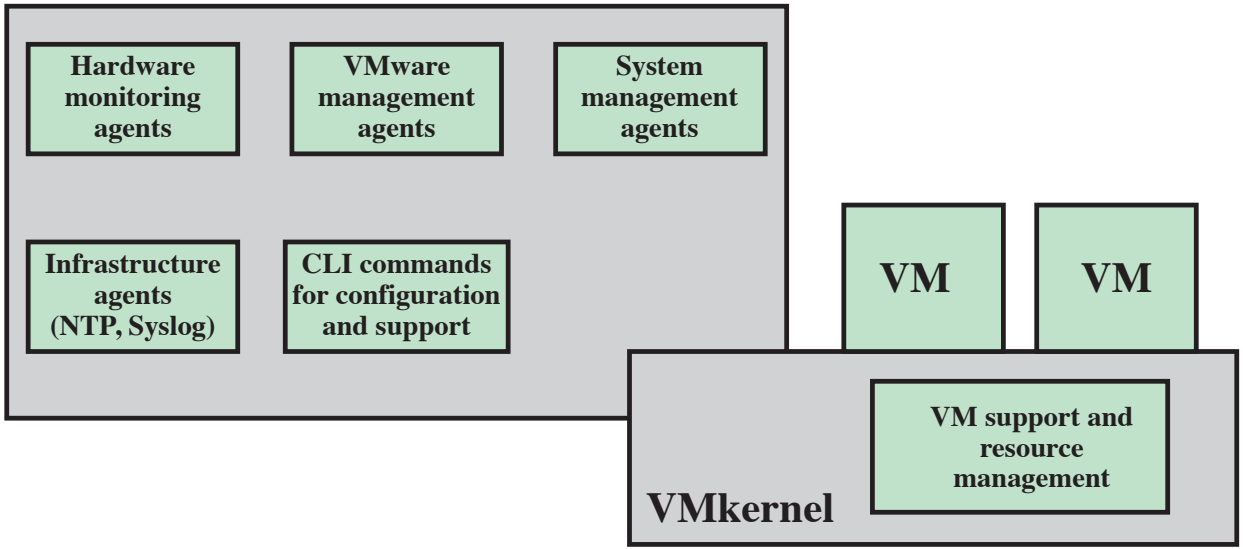
Examples

VMware ESXi

- VMware started the development of virtualization solutions for x86 architectures by the end of the 90s
- VMware is one of the major players in the enterprise virtualization business
- ESXi is a *bare-metal* Type-1 hypervisor, part of the vSphere solution
  - ESX is the former version

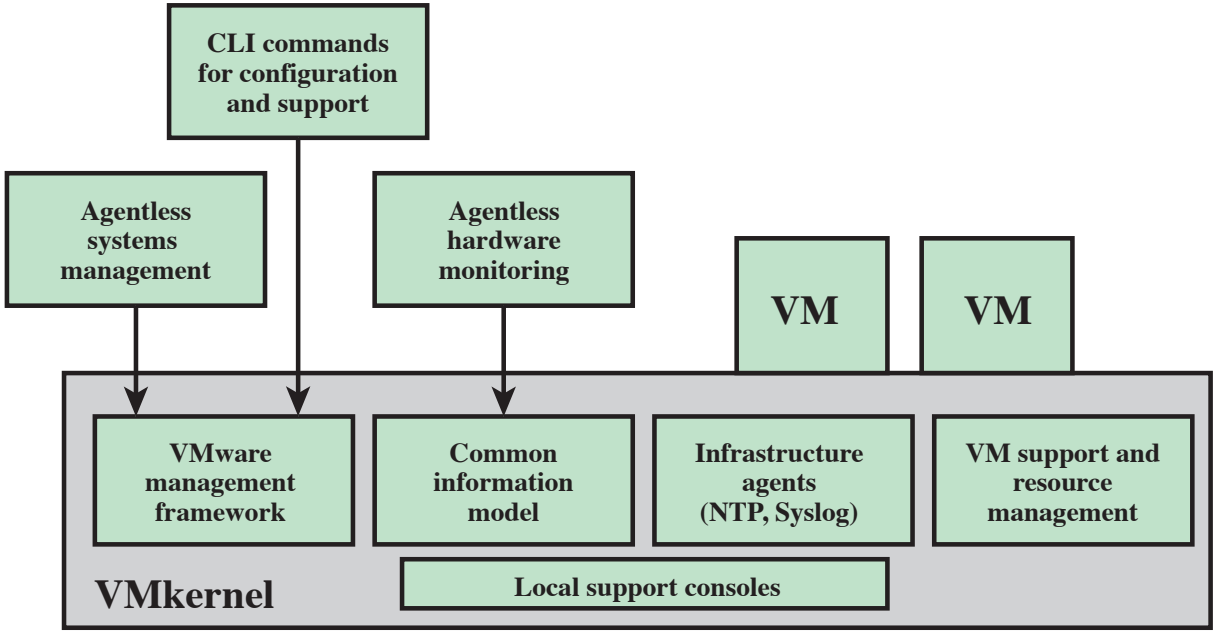
https://www.vmware.com/products/esxi-and-esx.html

VMware ESX architecture



The early versions of the VMware hypervisor leveraged on a Linux kernel for system management

VMware ESXi architecture

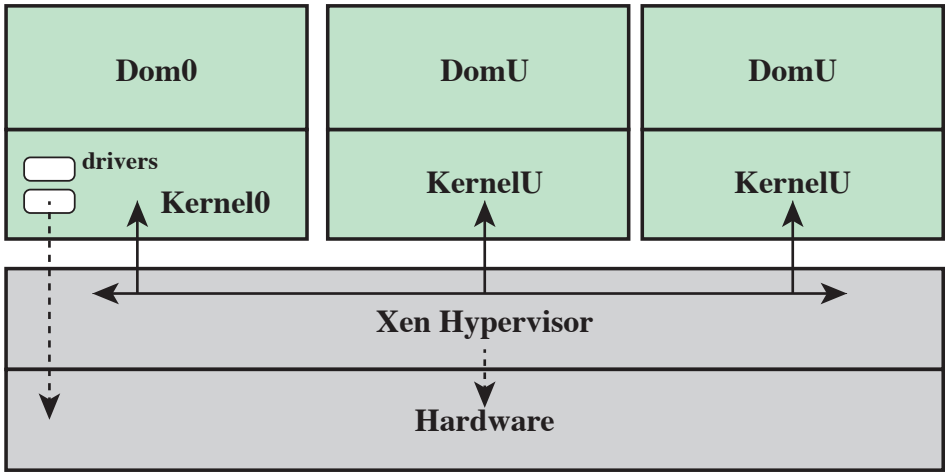


VMware ESXi main features

Storage VMotion	Permits the relocation of the data files that compose a virtual machine, while that virtual machine is in use
Fault Tolerance	Creates a lockstep copy of a virtual machine on a different host --- if the original host suffers a failure, the virtual machine's connections get shifted to the copy without interrupting users or the application they are using
Site Recovery Manager	Uses various replication technologies to copy selected virtual machines to a secondary site in the case of a data center disaster
Storage and Network I/O Control	Allows an administrator to allocate network bandwidth in a virtual network in a very granular manner
Distributed Resource Scheduler (DRS)	Intelligently places virtual machines on hosts for startup and can automatically balance the workloads via VMotion based on business policies and resource usage

XEN

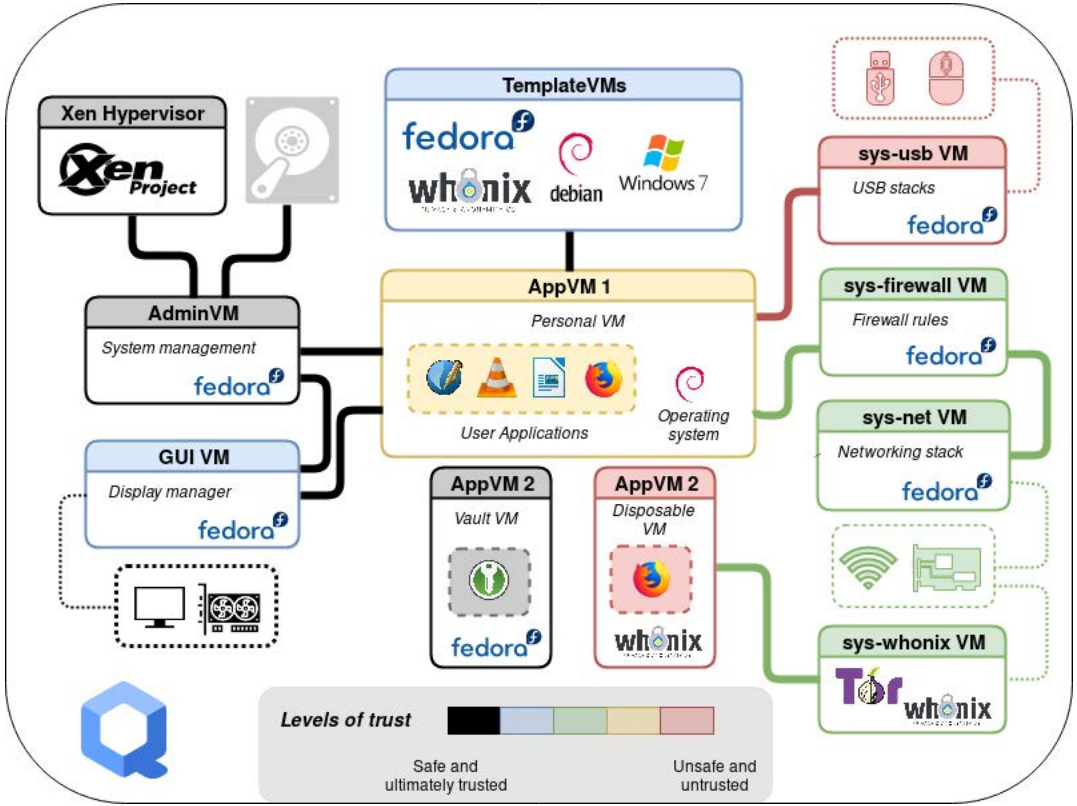
- Open-source project started by Cambridge University at the beginning of the new century, and supported by the Linux Foundation. Shipped with some EL distro
- XEN needs an operating systems to run the hypervisor, called DOMo (DOM-zero)



https://xenproject.org

Qubes OS

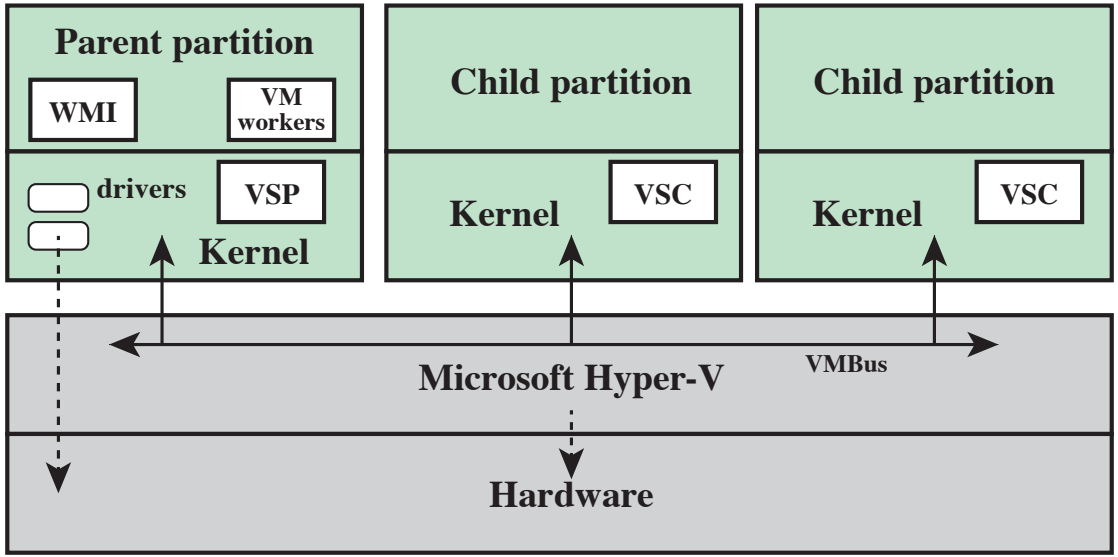
leverages xen virtualization to create and manage isolated VMs called qubes.



https://www.qubes-os.org

Microsoft Hyper-V

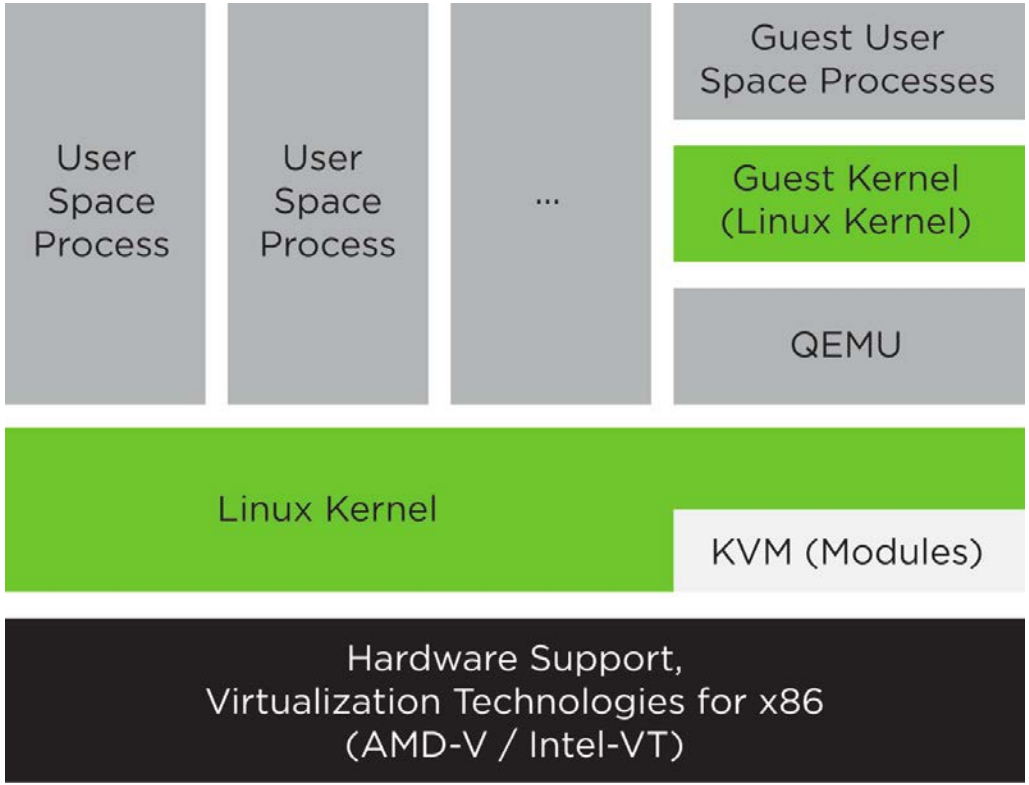
- Type-1 hypervisor released in 2008 as part of Windows Server 2008
- Now shipped with Windows 10



KVM

- Kernel-based Virtual Machine (KVM) is a virtualization module in the Linux kernel that allows the kernel to function as a Type-1 bare-metal hypervisor.
- It was merged into the Linux kernel mainline in kernel version 2.6.20, since 2007.
- KVM requires a processor with hardware virtualization extensions, such as Intel VT or AMD-V.

KVM Architecture



https://doc.opensuse.org/documentation/leap/virtualization/html/book.virt/cha-kvm-intro.html

Run-Time Environments

Java VM

- Programming language designed to run within custom-built virtualized environment
- In this case virtualization is defined as providing APIs that define a set of features made available to a language and programs written in that language to provide an improved execution environment
- JVM compiled to run on many systems
  - Similar to interpreted languages

JVM

- It is an abstract machine comprising
  - the instruction set
  - the program counter
  - the stack to store variables and results
  - the heap for runtime data and garbage collection
  - the *method area* to store the code and constants

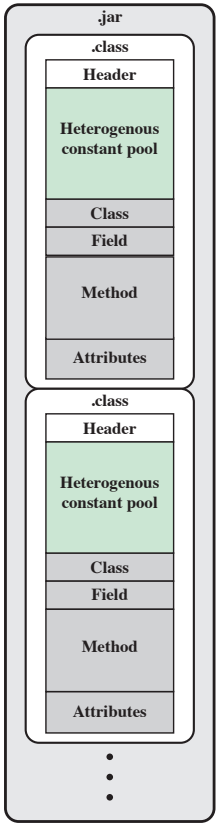
Android Runtime (ART)

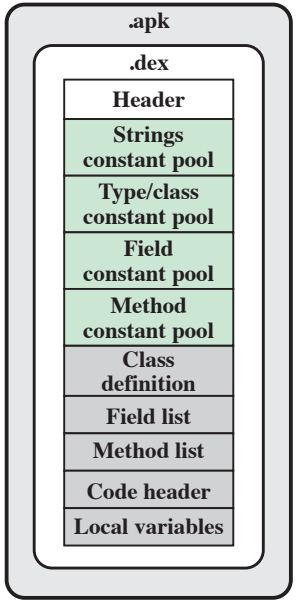
Previously named Dalvik

(DVM) executes files in .dex format
(Dalvik Executable)

Every Android Application is executed in an isolated ART

Java VM and Dalvik VM



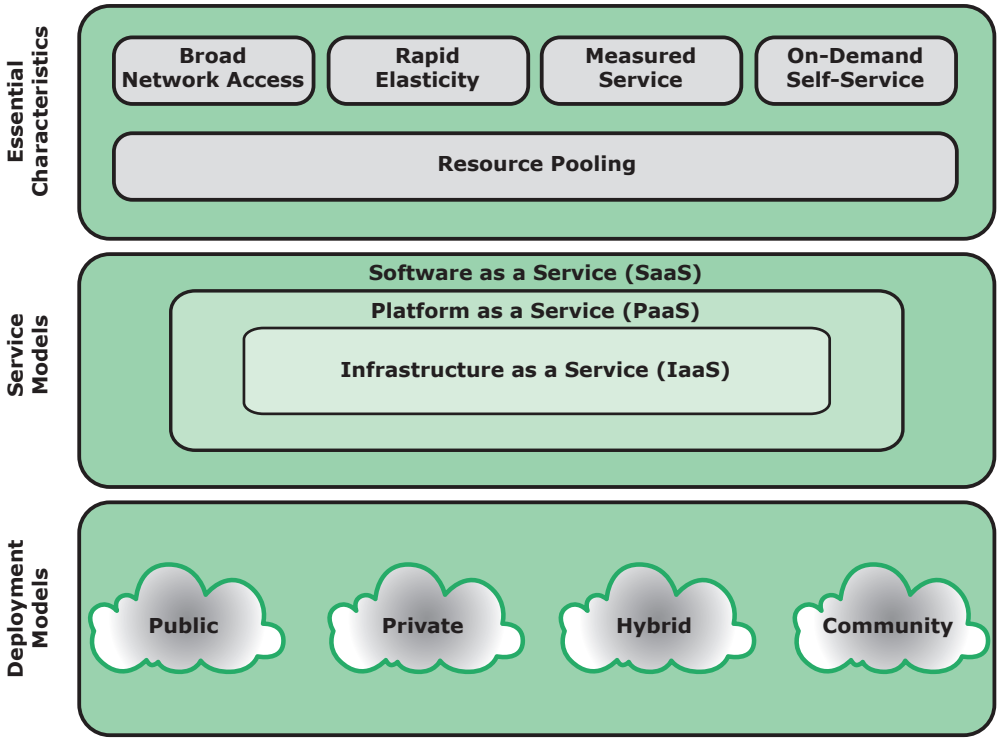


Zygote

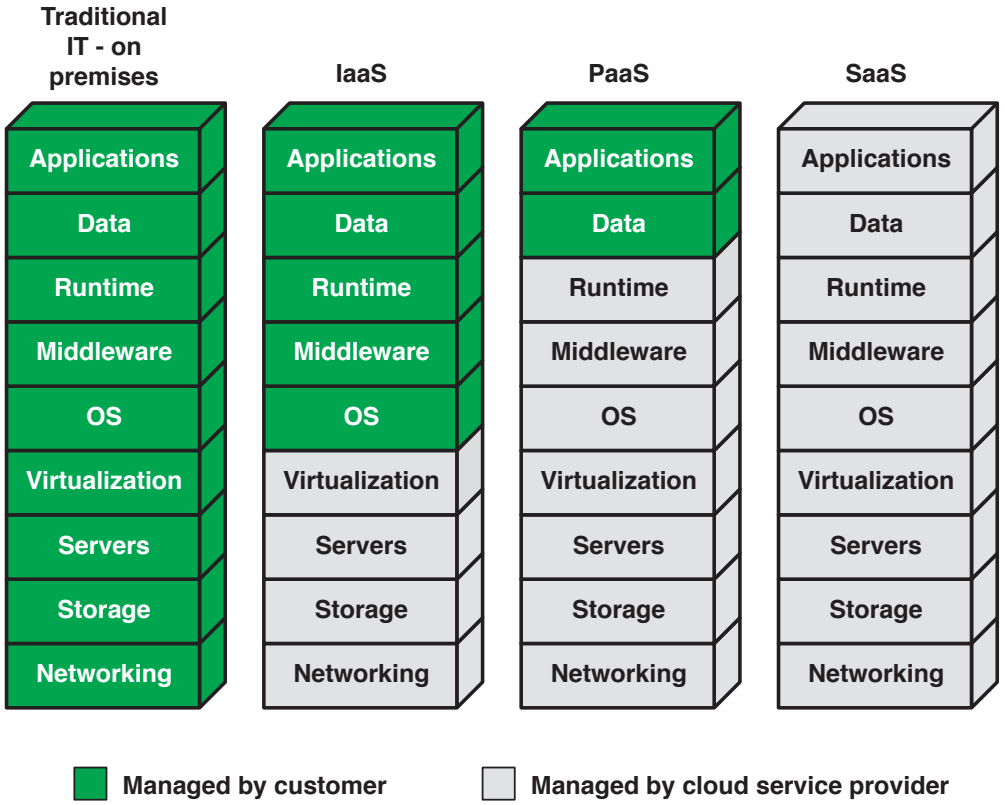
- This is the name of the process that is executed at system startup. It contains a copy of the ART
- Zygote forks a new ART as soon as an applications is requested to start
  - The time needed to create a new ART is reduced thanks to a clever management of shared memory locations
- At startup all Java core classes and resources are loaded and initialised

Cloud Computing and IoT

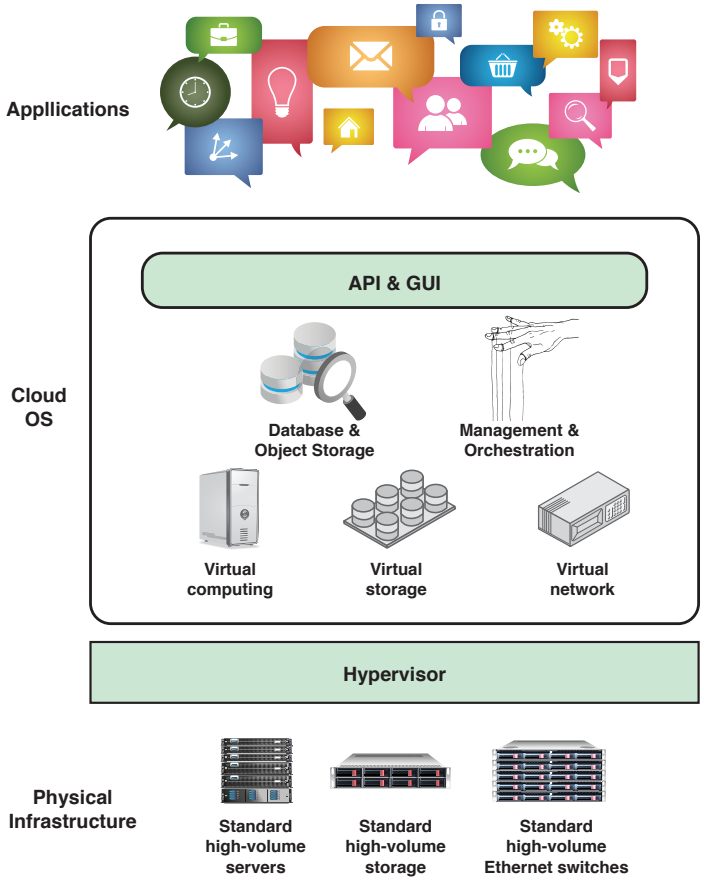
Cloud Computing Components

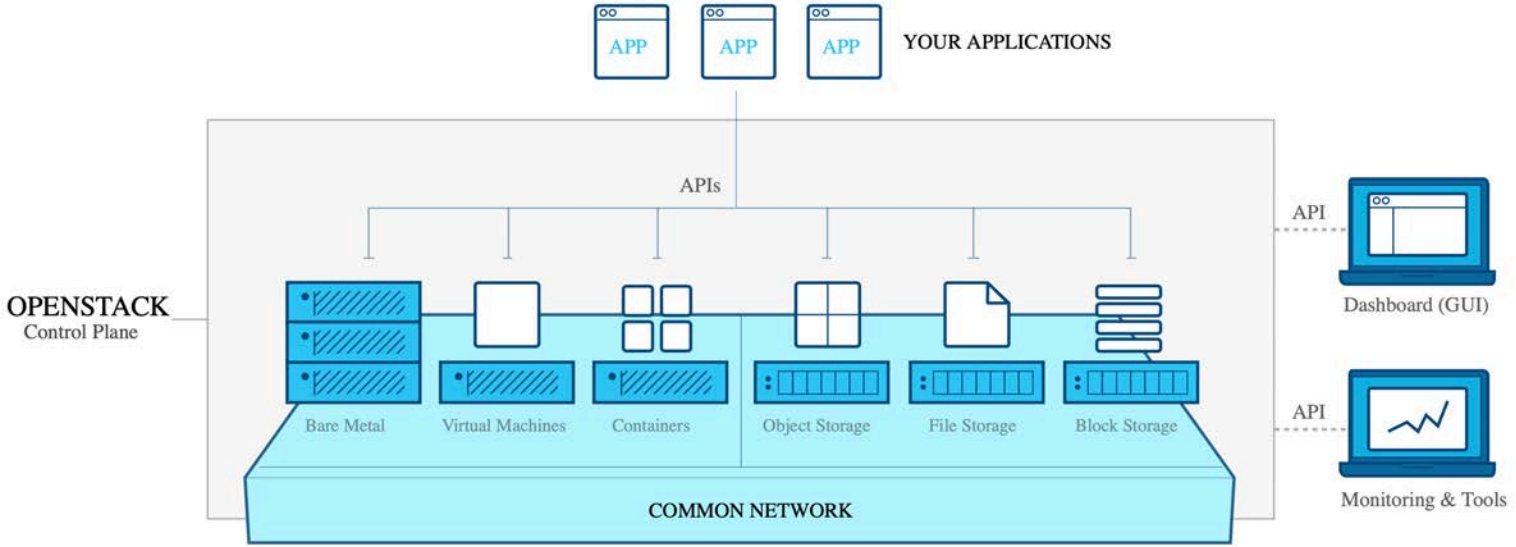


Cloud Service Models



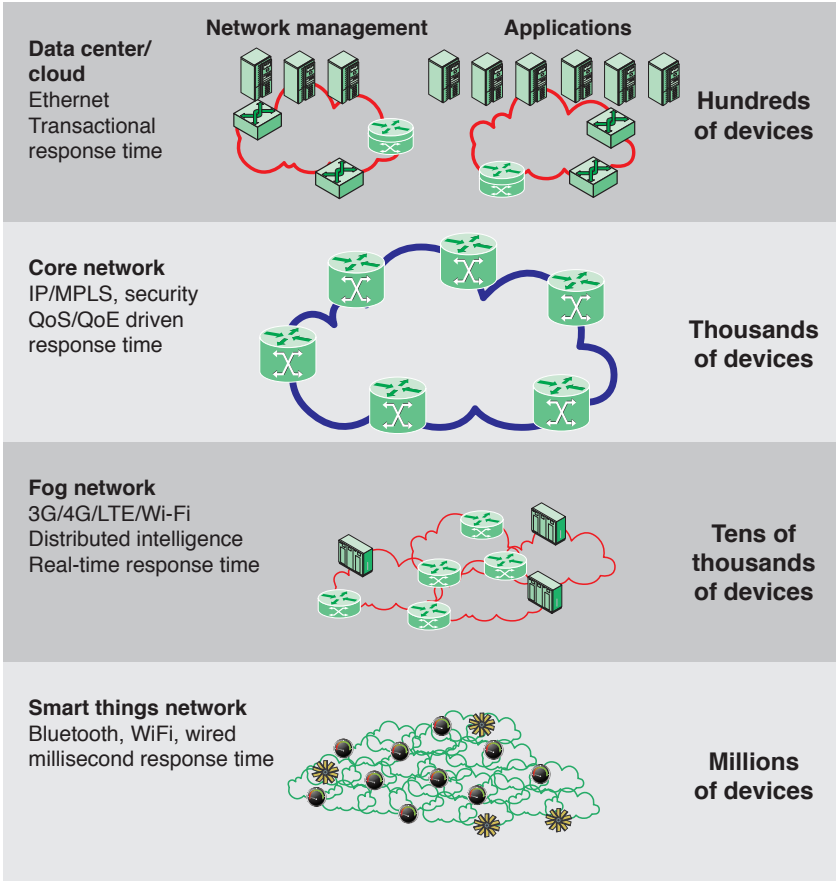
Cloud Operating Systems





OpenStack
http://www.openstack.org

Internet-of-Things and Cloud



OS Architecture for IoT