COMP5349 – Cloud Computing

Week 2: Virtualization Technology

Dr. Ying Zhou
School of Computer Science

Outline

- **Virtualization Technology Overview**
  - **Definition**
  - **Early Examples**
  - **Technology areas and subareas**
- **Server Virtualization**
- **Resource managements**
- **Brief Intro to AWS EC2 and Related Concepts**

Last Week

- Cloud computing allows users to rent various resources from providers on hourly(secondly), daily, monthly, yearly base
- A few general models, e.g. XaaS, are used to describe the spectrum of cloud computing
- IaaS and some PaaS use virtualization technology to present a virtual machine to the user
  - Virtualization technology is relatively standard and mature
- A related technology, container, is gaining popularity in cloud computing

What is virtualization?

- Narrow and very common perception of virtualization in cloud era
  - “Virtualization lets you run *multiple virtual machines* on a *single physical machine*, with each virtual machine sharing the resources of that one physical computer across multiple environments.”
- Broadly, the term is used in many other context, some are closely related with each other
  - E.g. virtual memory, storage, network, virtual reality

Early virtualization examples I

- Virtual Memory
  - Provide more accessible memories to CPU than those physically presented on the board
  - An important component of modern operating system
    - Address translation
    - Data transfer
    - Data placement strategy for swap in/out

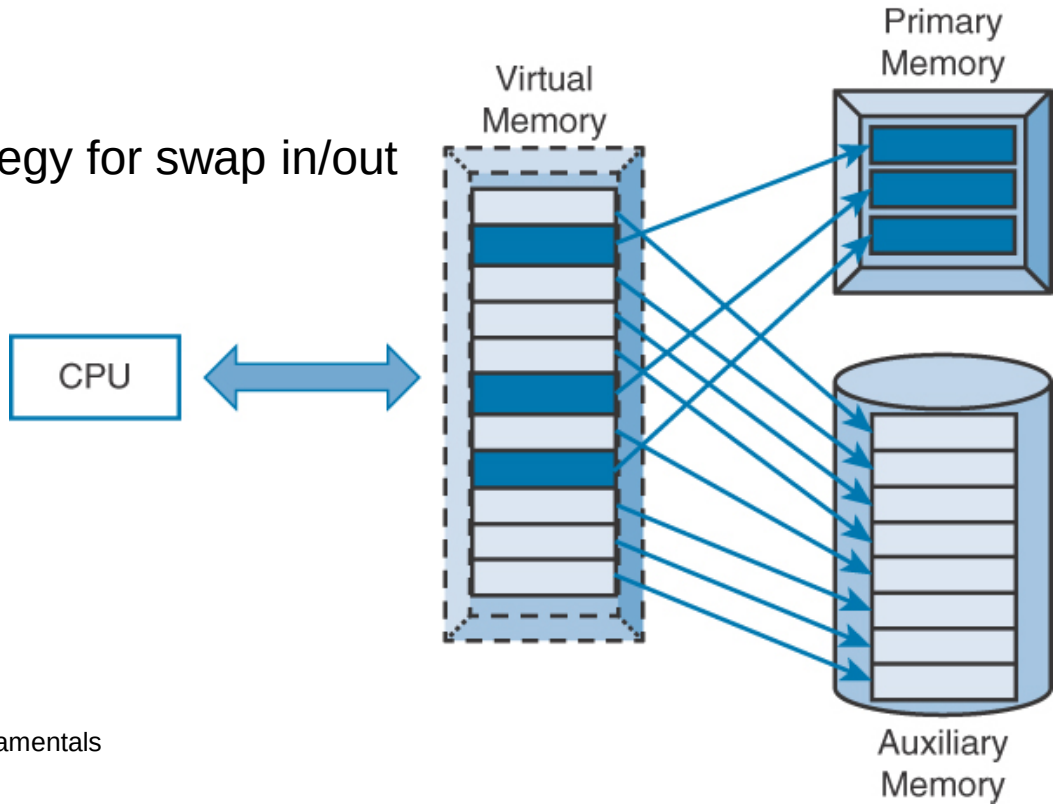


Figure 1.4 in Data Center Virtualization Fundamentals

Early virtualization examples II

- Mainframe Virtualization
  - IBM first released mainframe virtualization solution in 1972
  - The overall architecture looks similar to modern system VM
  - Solve the problem of OS migration triggered by hardware release
  - It was initialized by the time-sharing technologies in 1960s

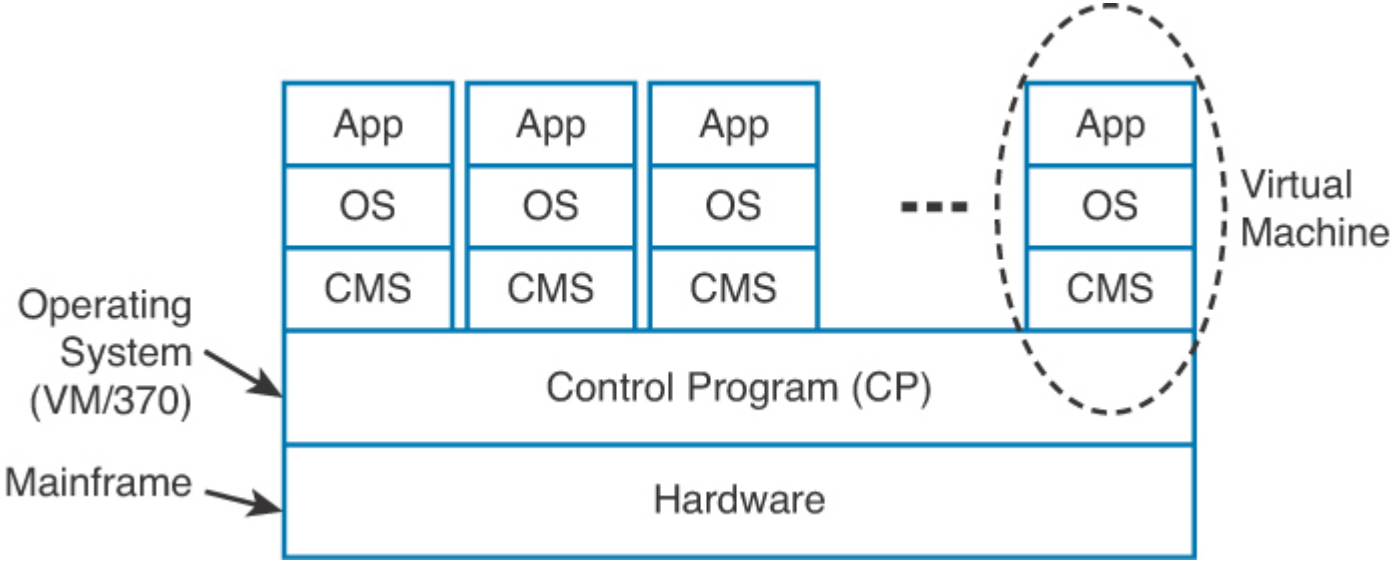


Figure 1.5 in Data Center Virtualization Fundamentals

Early virtualization examples III

- **Hot Standby Router Protocol**
  - **Solve the problem of single point of failure**
    - **One host can only define a default gateway**
  - **Use virtual IP and preconfigured priority to decide which physical router would act as the default gateway**
  - **Other router with the same virtual IP can take over if the default one fails**

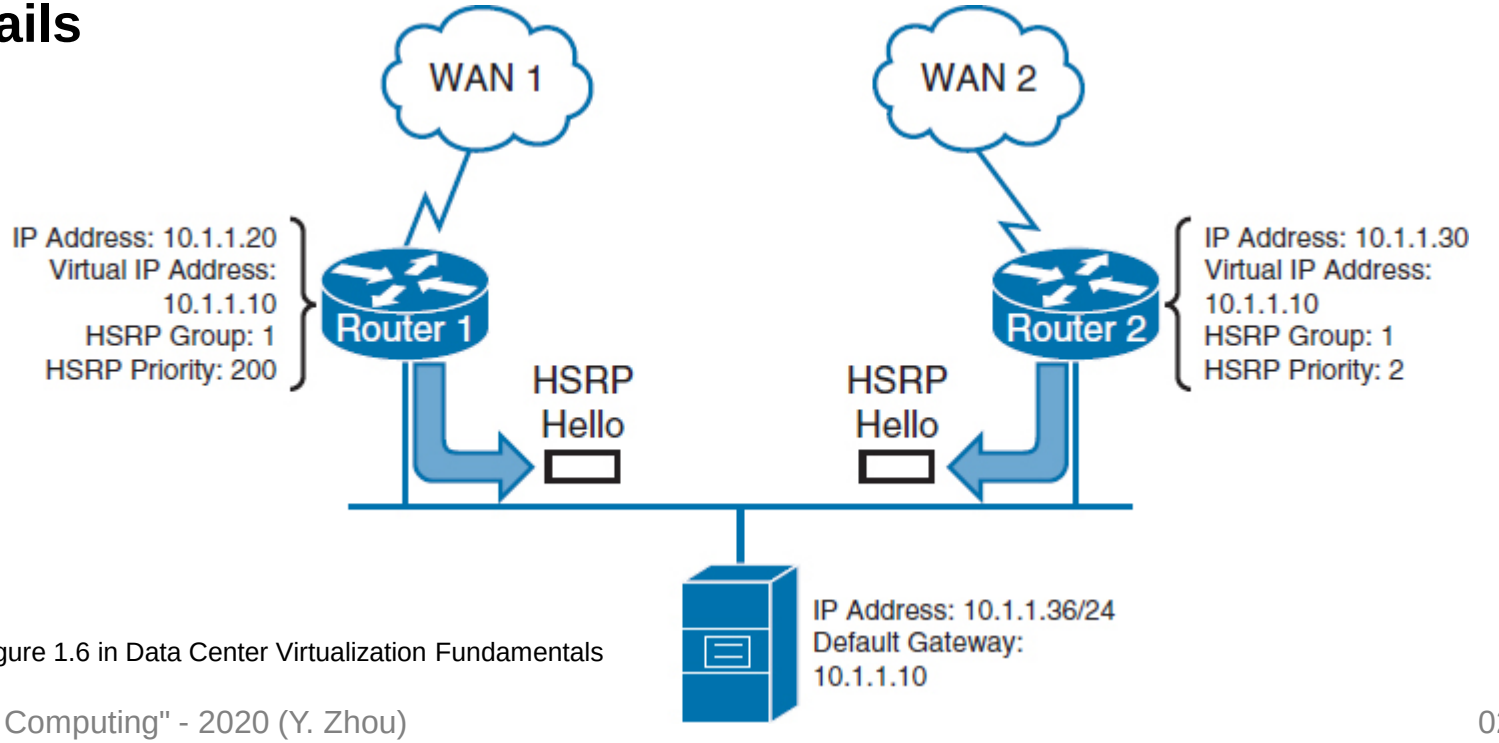
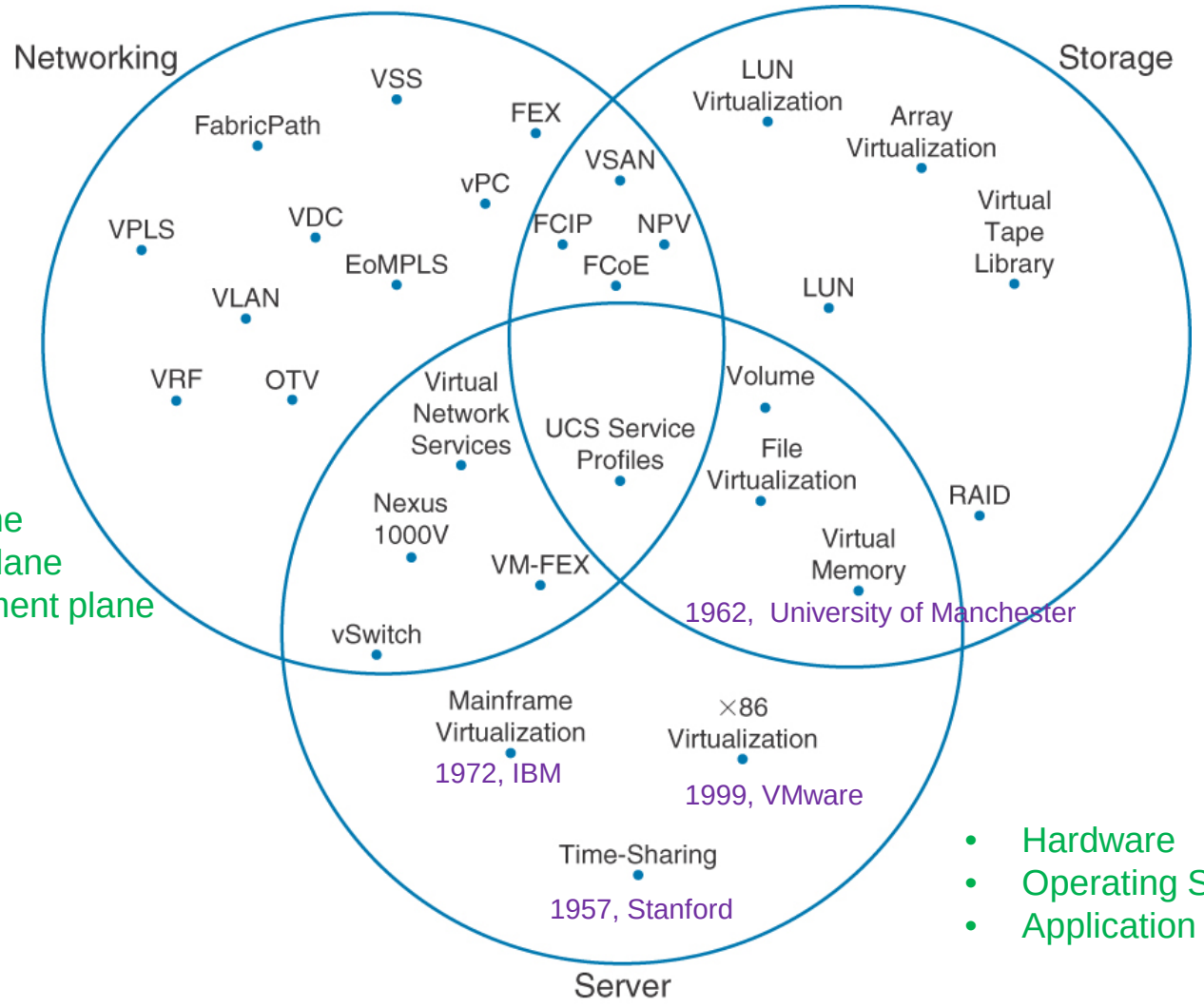


Figure 1.6 in Data Center Virtualization Fundamentals

Common features of virtualization

- Emulation
  - A preexisiting IT resources was emulated (memory, mainframe, IP address, etc)
- Transparency
  - The consumers of the resources (CPU, mainframe users, network hosts) cannot distinguish between real and emulated resources
- Benefits
  - Compared with directly using actual resources, virtualization brings various benefits (memory expansion, resource optimization, high availability, etc)
- A broader definition of virtualization
  - “Virtualization is the transparent emulation of an IT resource producing to its consumers benefits that were unavailable in its physical form.”

Virtualization Technology Areas



- Data plane
- Control plane
- Management plane

- Storage device
- Host
- Interconnect

- Hardware
- Operating System
- Application

Figure 1.9 in Data Center Virtualization Fundamentals

Subareas

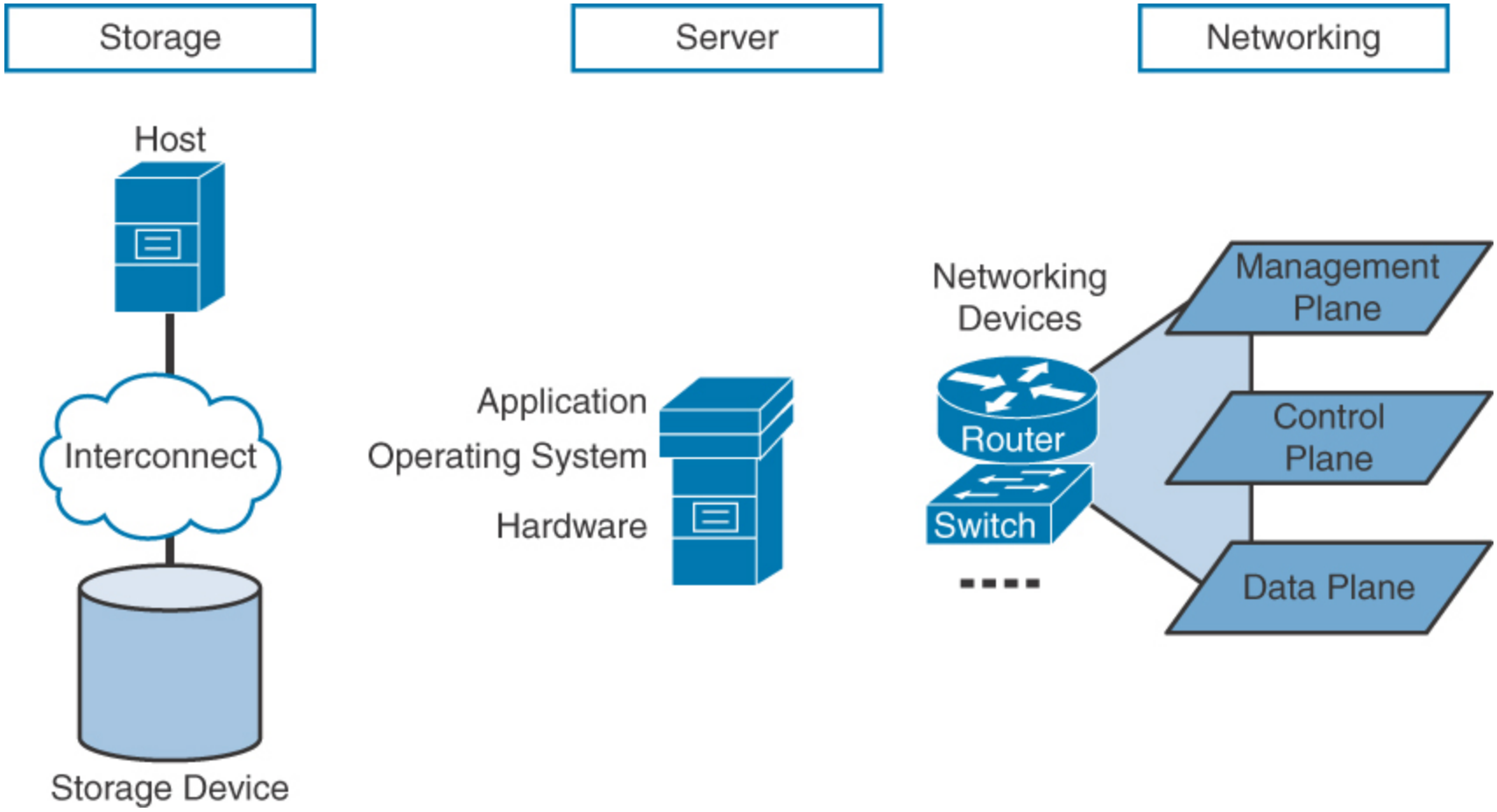


Figure 1.10 in Data Center Virtualization Fundamentals

Outline

- Virtualization Technology Overview
- **Server Virtualization**
  - **Motivation**
  - **Hypervisor**
    - **Types and Roles**
    - **Example Hypervisors**
  - **Virtual Machine**
    - **VM vs Native OS**
    - **Managing Critical Instruction**
- Resource managements
- Brief Intro to AWS EC2 and Related Concepts

Well-Known Utilization Problem: Server Sprawl

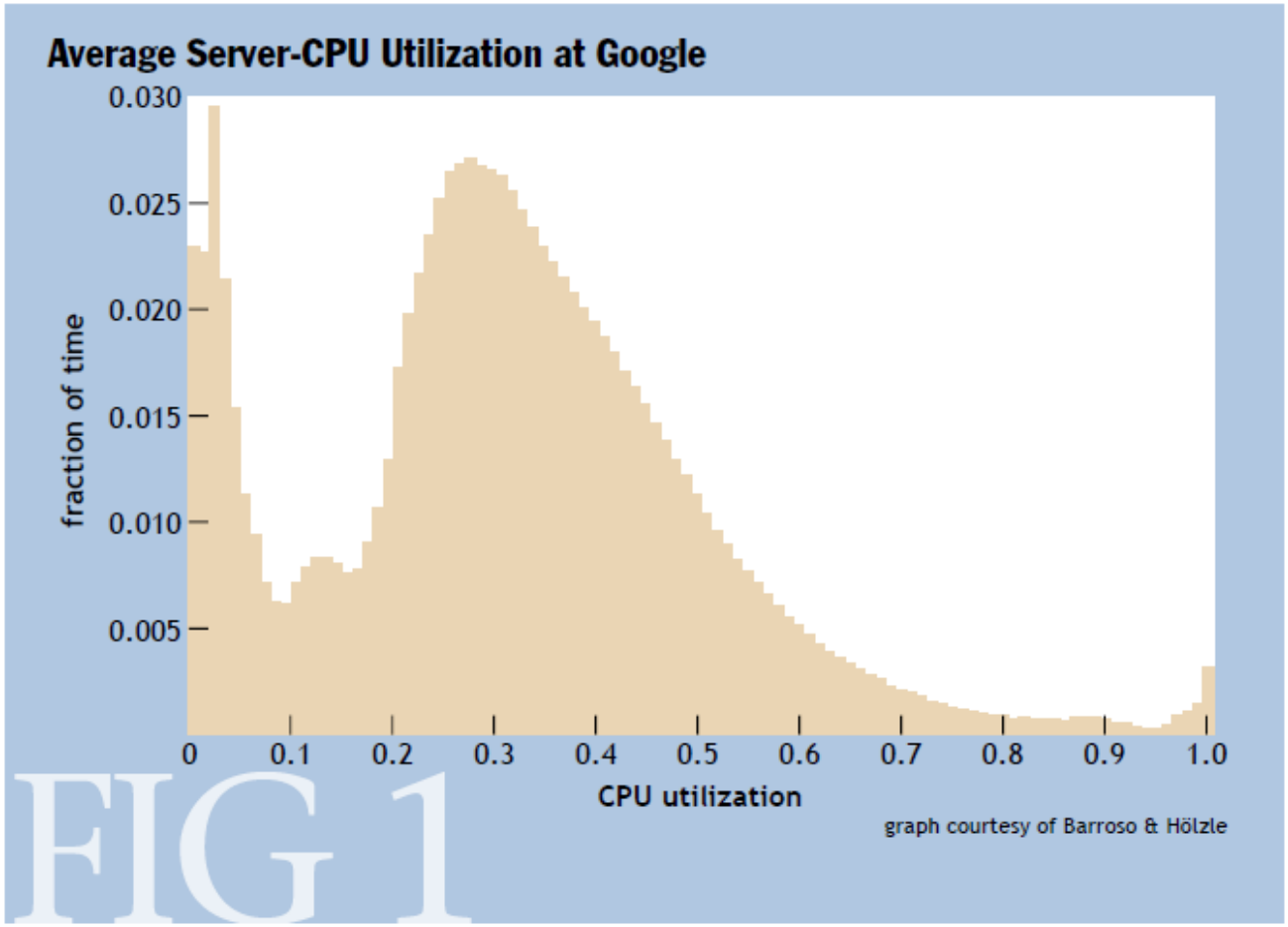
- *Server Sprawl*
  (large number of underutilized servers)
  a major problem in many IT departments
- Main causes:
  - Requirement by vendors to run their applications in isolation
  - Operating system heterogeneity
    - Mail server may require Windows server;
      a database may best run on Linux or Solaris.
  - Mergers and other integration projects

- "81 percent of CIOs were using virtualization technologies to drive consolidation, according to a recent survey by CIO" (2008 survey)

Increasing Utilization is Hard!

The problem only magnifies on a data center scale:

Average CPU utilization of 5000+ servers at Google during a six- month period



Beyond Server Consolidation. Werner Vogels. ACM Queue, Jan/Feb 2008.

100% Utilization is not the Goal

- Workload in the enterprise are heterogeneous
- Demand is uncertain and often occurs in spikes
- Operating System starts to behave unpredictable under high CPU and IO load
- for pure CPU-bound environments, 70 percent seems to be achievable for highly tuned applications; for environments with mixed workloads, 40 percent is a major success, and 50 percent has become the Holy Grail.
- Real world estimates of server utilization in datacenters range from 5% to 20%!
  - 30% is seen as a very good value already…

Server Consolidation

- Solution: Pool heterogeneous services together
  - To do so, run several **virtual machines** on the same shared hardware, coordinated by a **hypervisor**
    - (also known as: virtual machine manager/monitor (VMM))
    - Hypervisor abstracts / hides physical computing platform
  - Allows to share commodity hardware without sacrificing security and performance

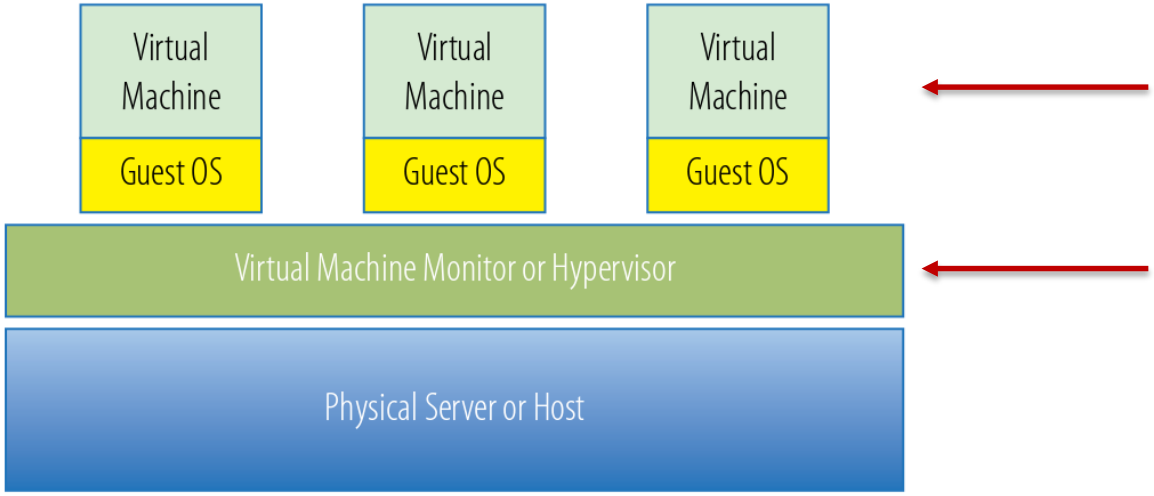


FIGURE 1.1 A basic virtual machine monitor (VMM)

Server Architectures Evolution

- Mainframes (1940s ~1970s)
- RISC Server (1970s ~ present)
  - **R**educed **I**nstruction **S**et **C**omputing
  - E.g. Sun Microsystem's Solaris workstation, IBM AIX workstation, etc
- X86 Servers(1981~ present)
  - Computers that inherited the basic architectures from Intel 8086 processors and its descendants (80286, 80386, 80486, etc)
  - This is the start of Personal Computer era

Main components of x86 server

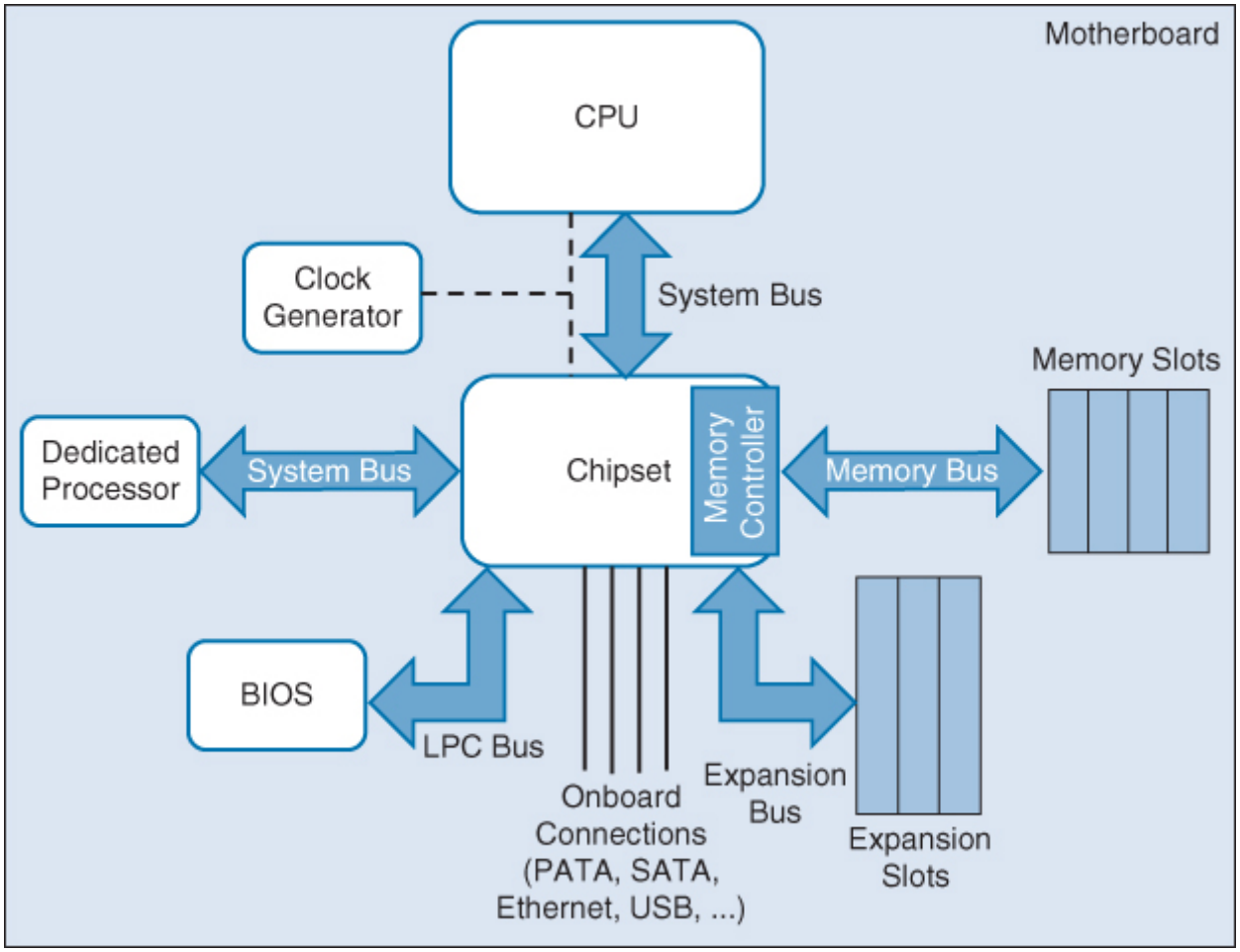


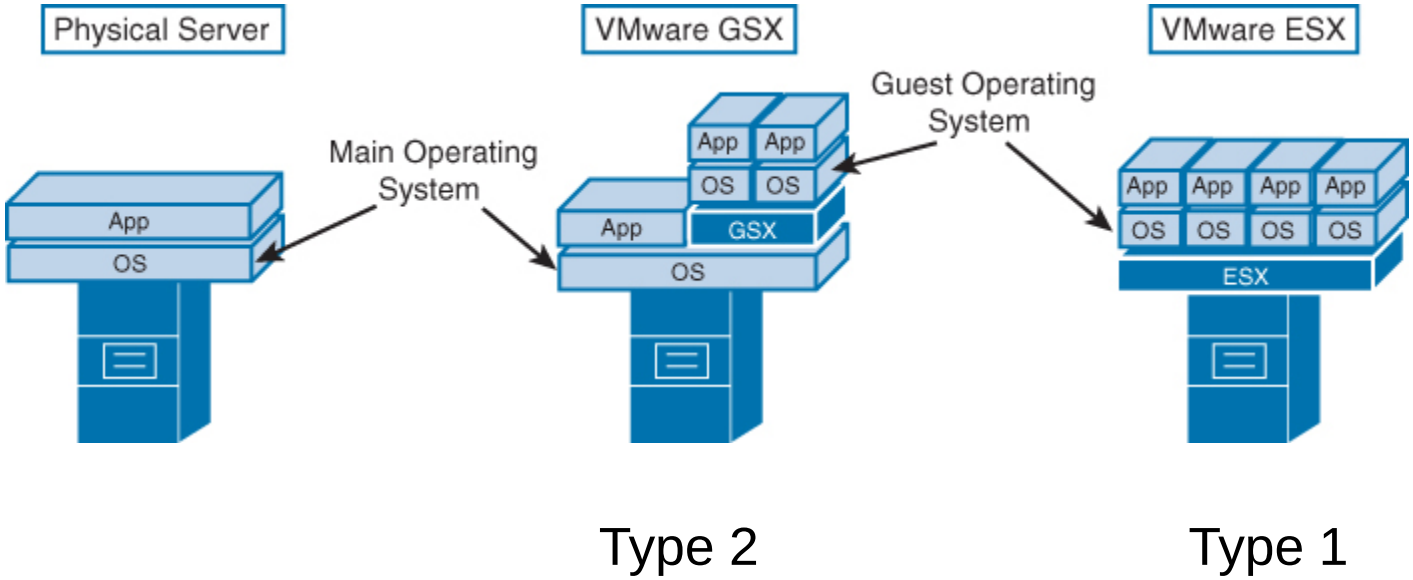
Figure 13.1 in Data Center Virtualization Fundamentals

Hypervisor

- It is a piece of software responsible for the *creation of the VMs*, *resource sharing*, *device management*, *virtual storage management*, and other traditional operating system tasks.

Two types of Hypervisor

Physical Server and VMware Solutions in 2001

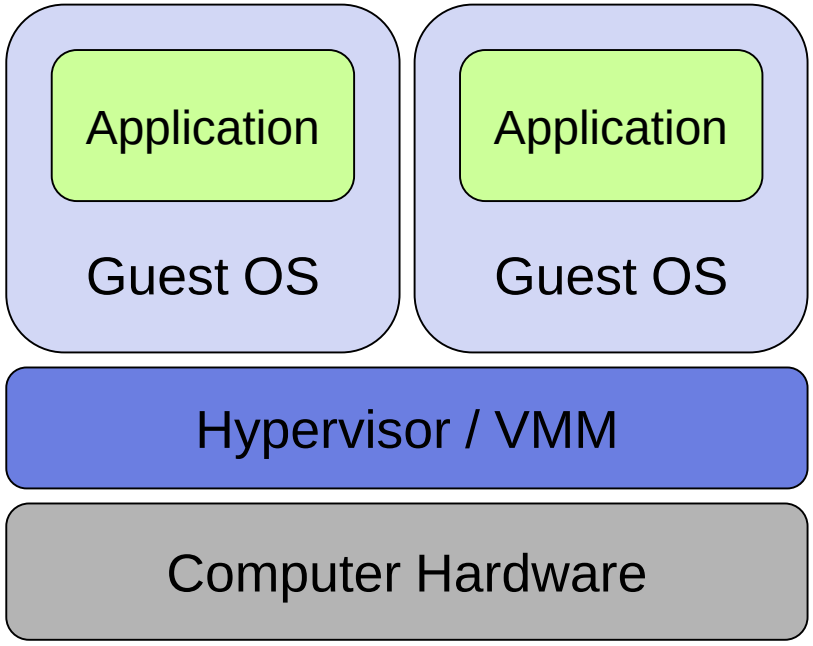


Type 2 Type 1

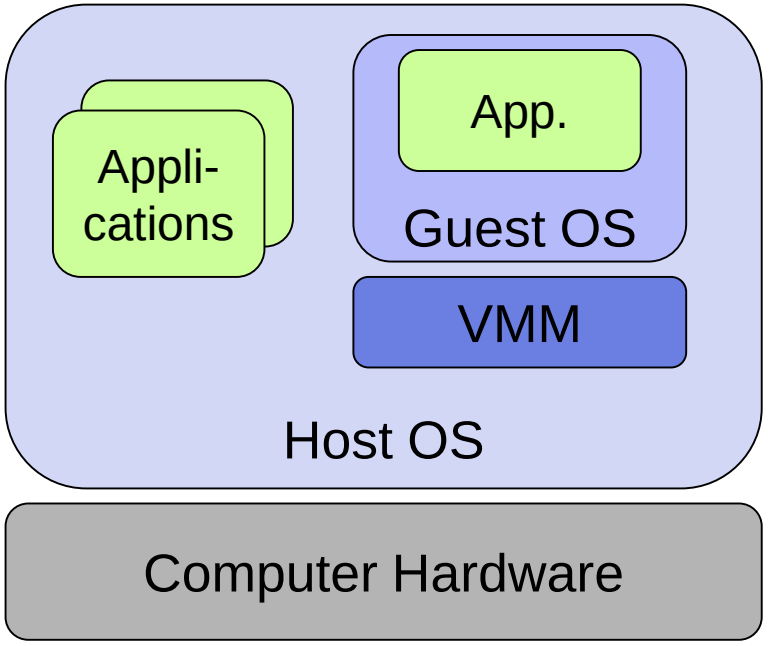
Figure 13.6 in Data Center Virtualization Fundamentals

Two Types of Hypervisor

Type I: Bare-metal (native) VM



Type II: Hosted VM



The role of a Hypervisor

- Provide an environment identical to the physical environment.
- Provide that environment with minimal performance cost.
- Retain complete control of the system resources

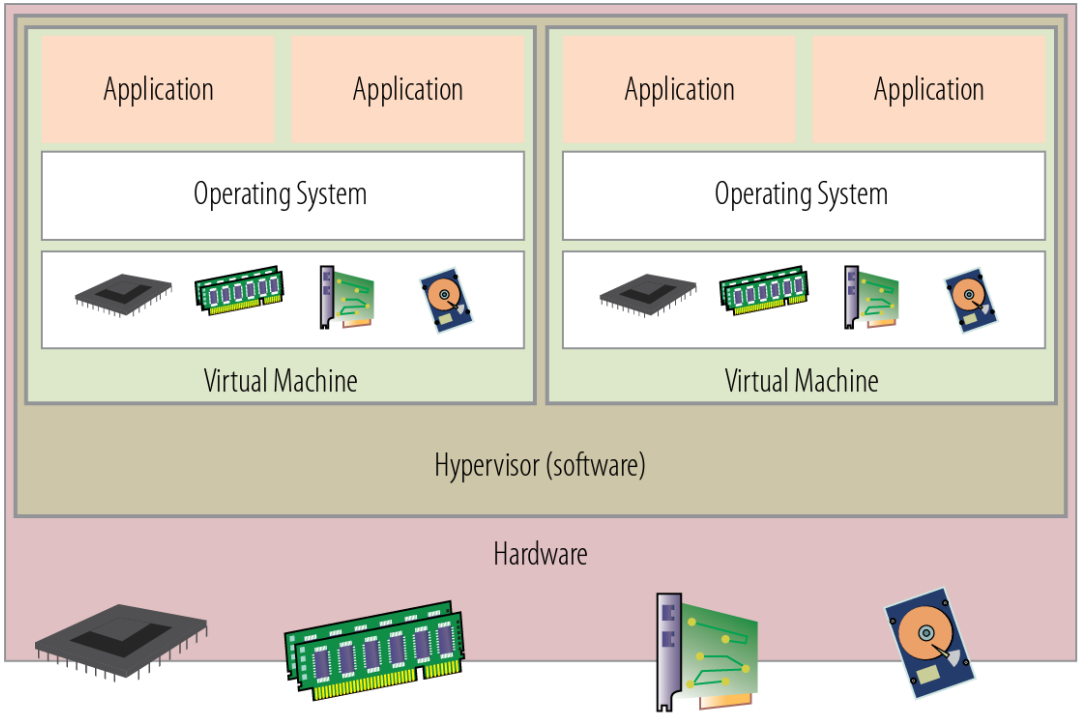


Figure 2.6 in Virtualization Essentials

Managing Resources

- Each VM is presented with a fraction of the resources of the physical host
  - E.g. A host may have 256 GB of physical memory installed in its frame, but a guest VM may believe that it has 4 GB
- Hypervisor abstracts the hardware resources and controls access to it to ensure each VM get its allocated resources
  - All access to hardware resources goes through hypervisor
    - Access to memory, storage, network

Example Hypervisor: VMWare ESXi

- VMkernel is a POSIX-like operating system which provides certain functionality similar to that found in other operating systems, such as process creation and control, signals, file system, and process threads.
- It is designed specifically to support running multiple virtual machines and provides such core functionality as: Resource scheduling, I/O stacks, Device drivers

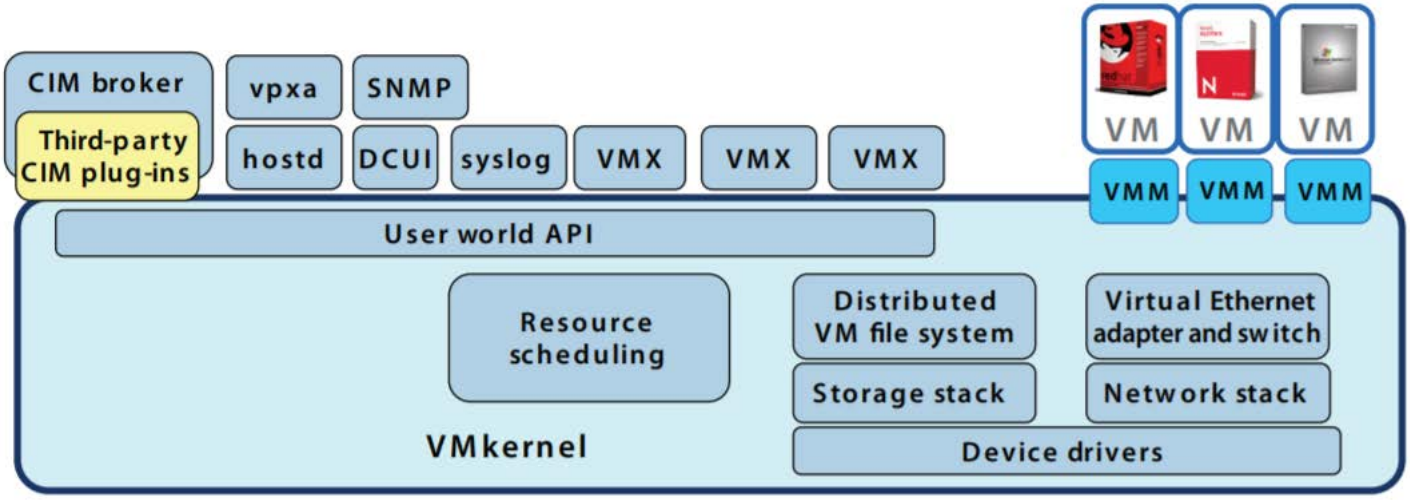
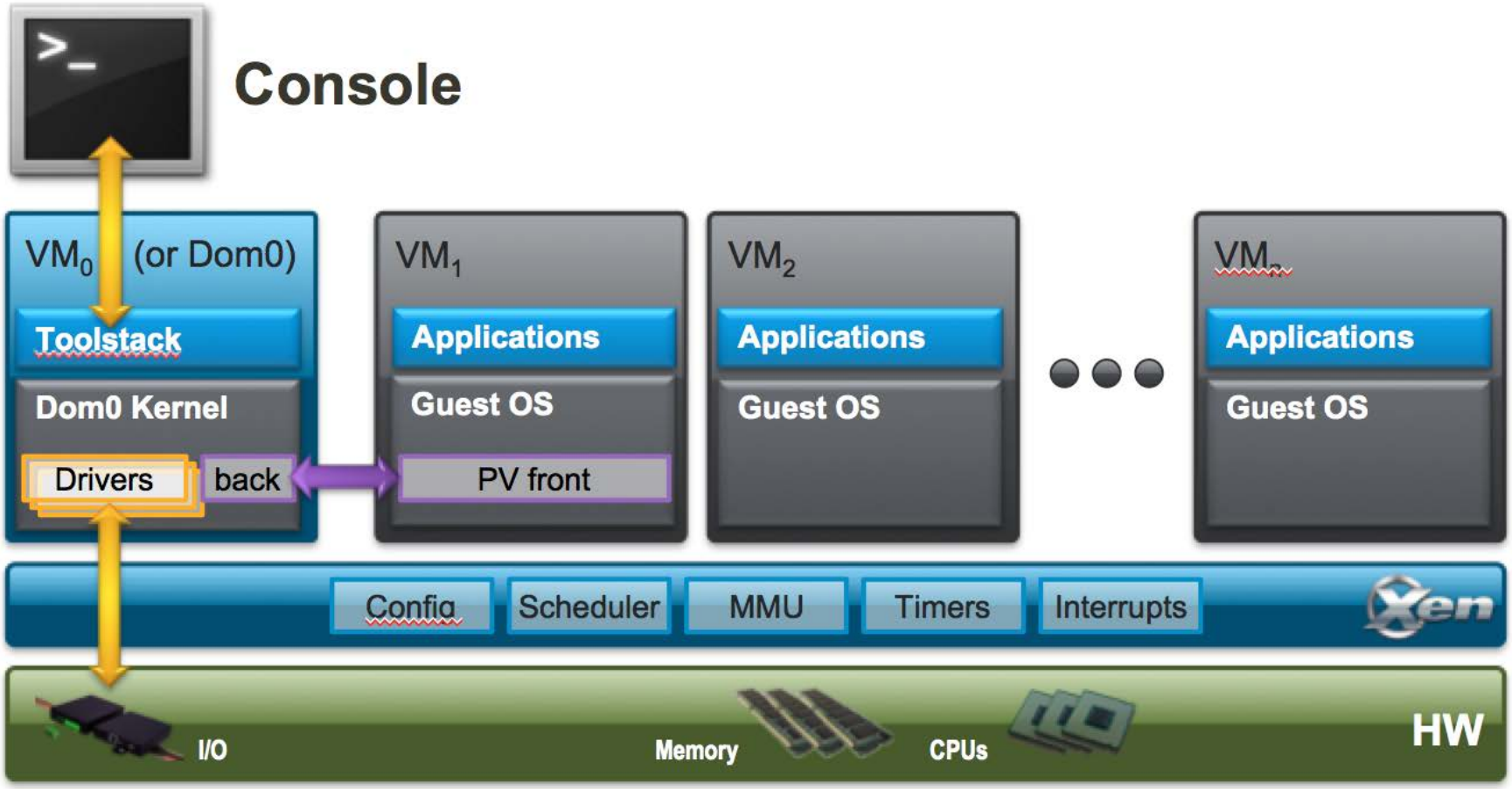


Figure 1: The streamlined architecture of VMware ESXi eliminates the need for a service console.

https://microage.com/wp-content/uploads/2016/02/ESXi_architecture.pdf

Example Hypervisors: Xen

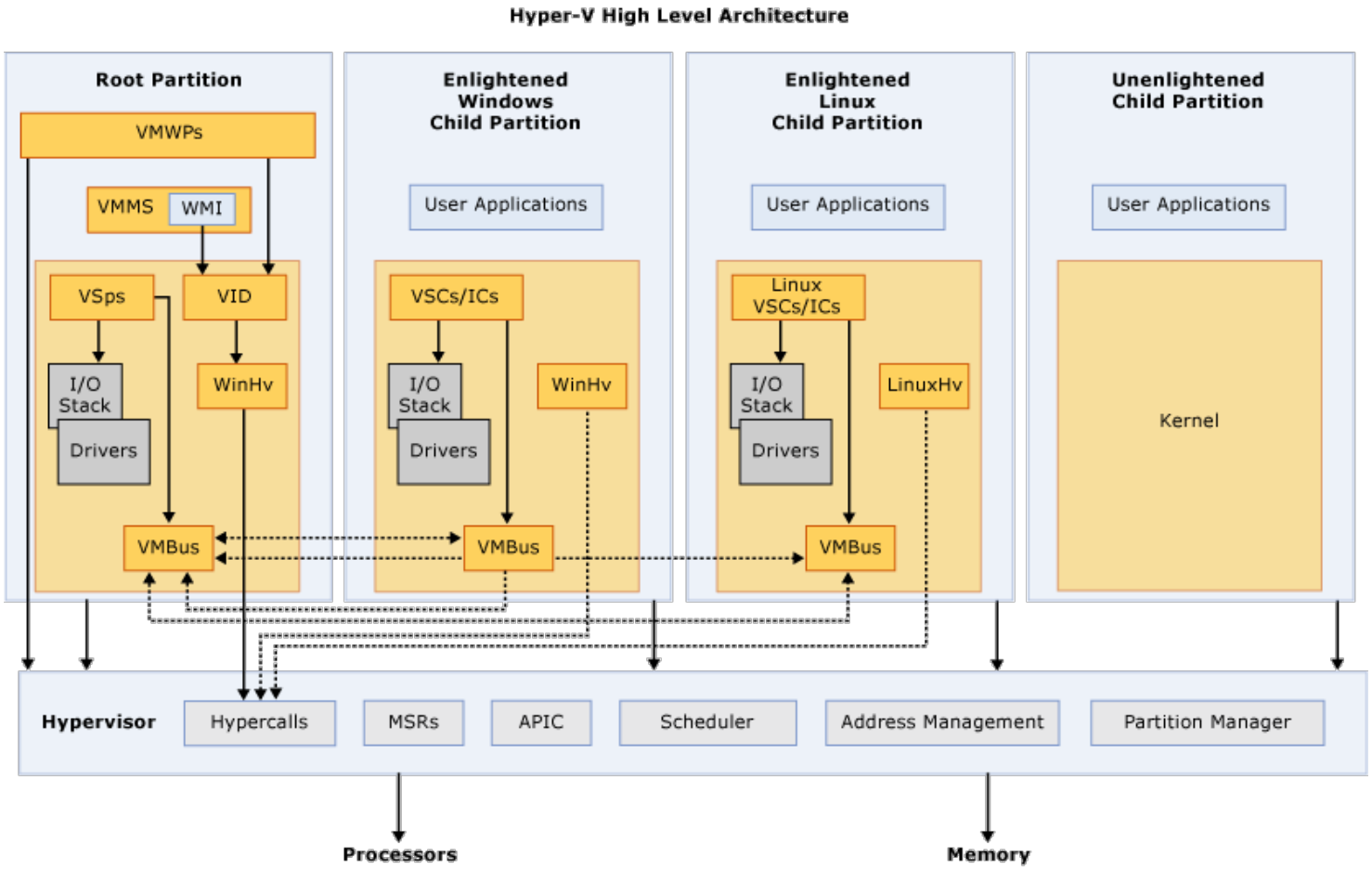


https://wiki.xen.org/wiki/Xen_Project_Software_Overview

Xen Hypervisor: Component

- **The Xen Hypervisor**
  - A thin software layer runs directly on the hardware and is
  - Responsible for managing CPU, memory, and interrupts
- **The Control Domain (or Domain 0)** is
  - A specialized Virtual Machine for handling I/O and for interacting with the other Virtual Machines.
  - It also exposes a control interface to the outside world, through which the system is controlled.
- **Guest Domains/Virtual Machines**
  - The VM allocated to users
- **Toolstack and Console**:
  - Admin interface for creating, destroying and configuring guest domains

Example Hypervisor: Hyper-V



https://docs.microsoft.com/en-us/biztalk/technical-guides/appendix-b-hyper-v-architecture-and-feature-overview

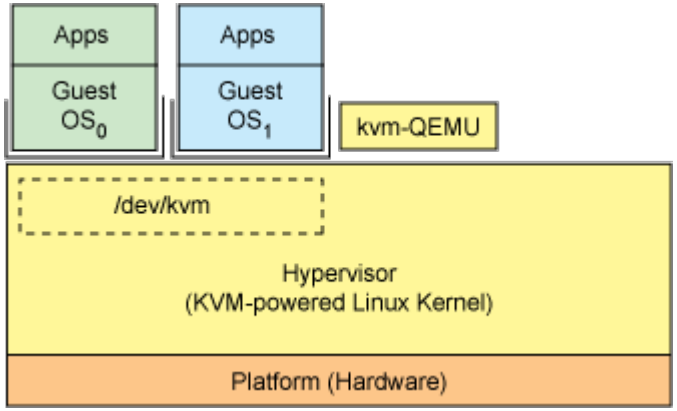
Hyper-V Components

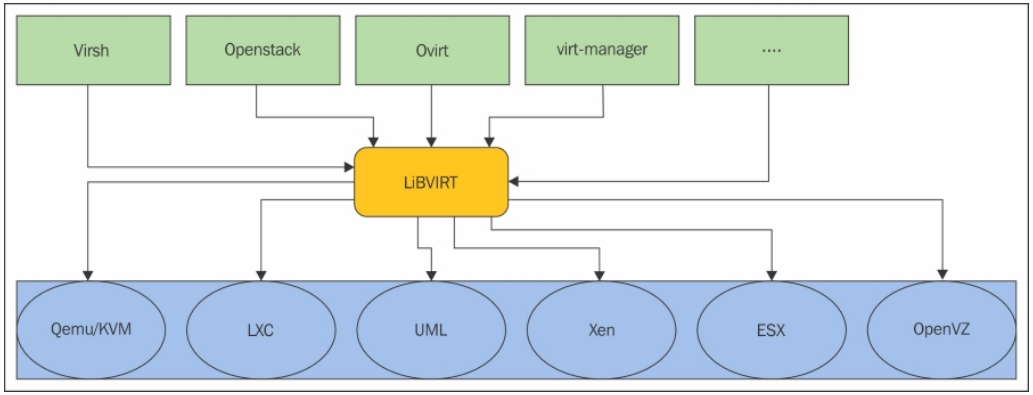
- **Hypervisor**
  - A layer of software that sits between the hardware
  - Its primary job is to provide isolated execution environments called partitions. The hypervisor controls and arbitrates access to the underlying hardware.
- **Partition**
  - Basically the virtual machine running on hypervisor
- **Root Partition**
  - Similar to Xen's domain 0, which manages I/O and communicate with other partitions
- **Child Partition**
  - All other guest VMs

Example Hypervisor: KVM

- Hypervisor performs lots of traditional operating system tasks
- Kernel-based Virtual Machine (KVM) development was started mid 2006 at Qumranet, which was acquired by Red Hat in 2008.
- It turns Linux kernel into a hypervisor with the help of a few modules: KVM, QEMU and libvert
- The development started soon after major chip manufacturers introduced hardware support for virtualization at processor level
  - Intel introduced VT-x in November 2005
  - AMD introduced AMD-V in May 2006
- Both Google Cloud Platform and AWS use KMV based virtualization

KVM architecture





https://developer.ibm.com/tutorials/l-hypervisor/

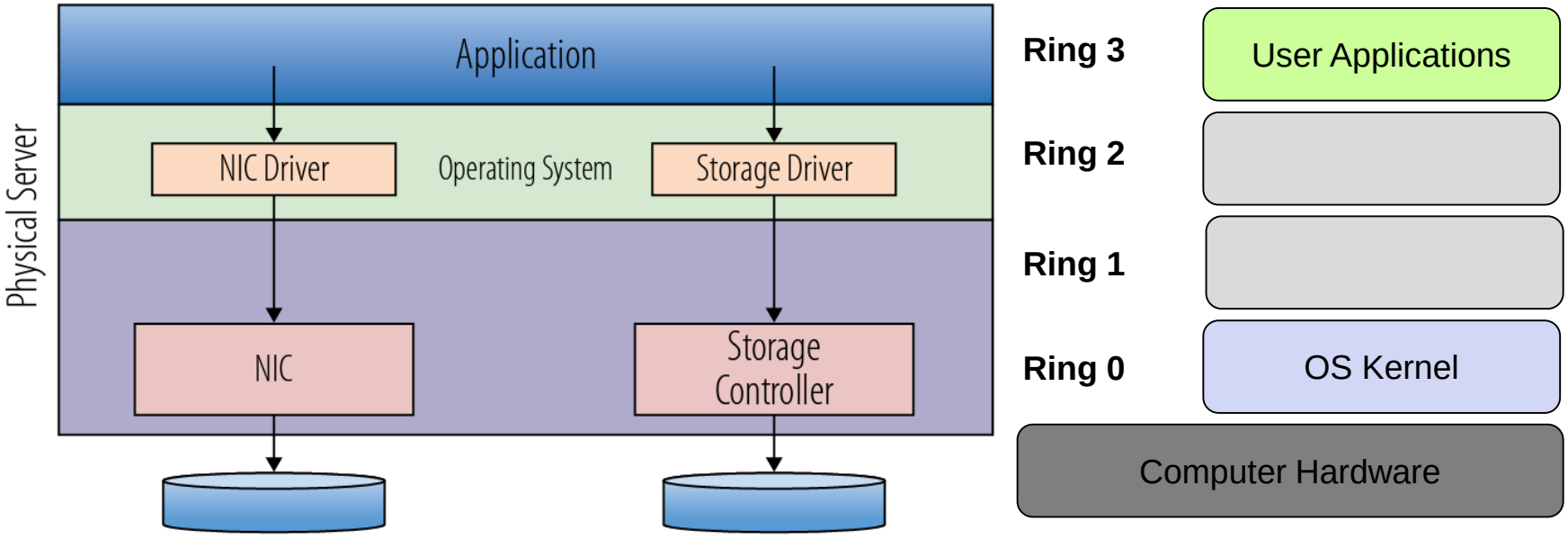
- **KVM** turns a Linux kernel into a hypervisor
  - Several hardware-aware versions
  - When a new operating system is booted on KVM, it becomes a *process* of the host operating system and therefore scheduleable like any other process. But it is in executed in different mode to regular OS
- QEMU (Quick Emulator)
  - A generic emulator that virtualizes I/O devices as well
- Libvert provides a common management layer to manage virtual machines running on hypervisors

Virtual Machine

- Virtual machine can be viewed as a *software computer* running an operating system and associated applications.
- The software computer runs on top of a hypervisor on a physical server
- A virtual machine is described by a set of files
  - Emulated hardware definition file
  - Virtual disk data
  - VM BIOS
- The design to describe a VM through a set of files makes it very easy to save, copy and clone VMs across physical servers.

Hardware Management in Native OS

- Application's request for hardware (disk, network, cpu, memory) need to go through OS for safety and security reason
- Those measures are built into x86 architecture by having protection ring mode at processor level

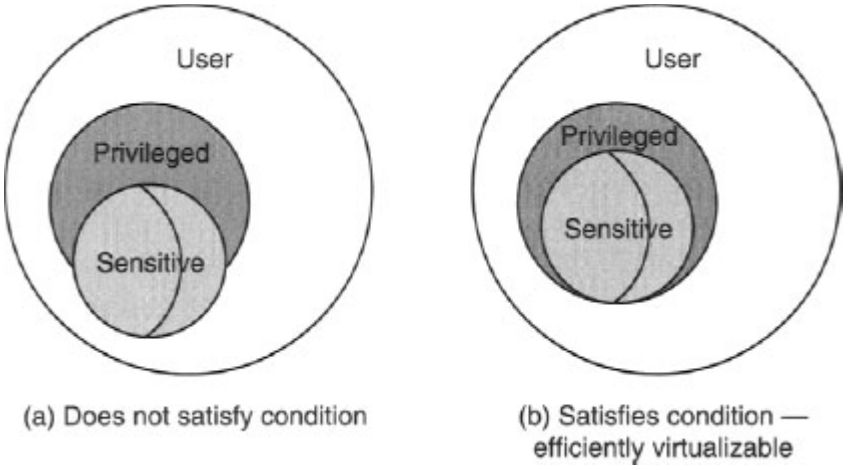


Challenges for Hypervisor Implementation

- Each VM runs its OS (guest OS) which assumes full control of the all hardware resources
  - But that is not true, guest OS has only a portion of the hardware to use in virtualization
  - OS is designed to be the most privileged software system on a machine, now it is not
  - Do we need to modify OS implementation?
- The extra layer may introduce performance overhead
  - Before: OS + App
  - Now: VMM + OS + APP
  - Relationship between guest OS and VMM should be carefully managed.

Type of Instructions

- Privileged Instruction
  - One that traps when it is in user mode and does not trap when it is in system mode
  - E.g. set CPU timer
- Sensitive Instruction
  - Instructions that interact with hardware
  - E.g. memory address translation
- Others

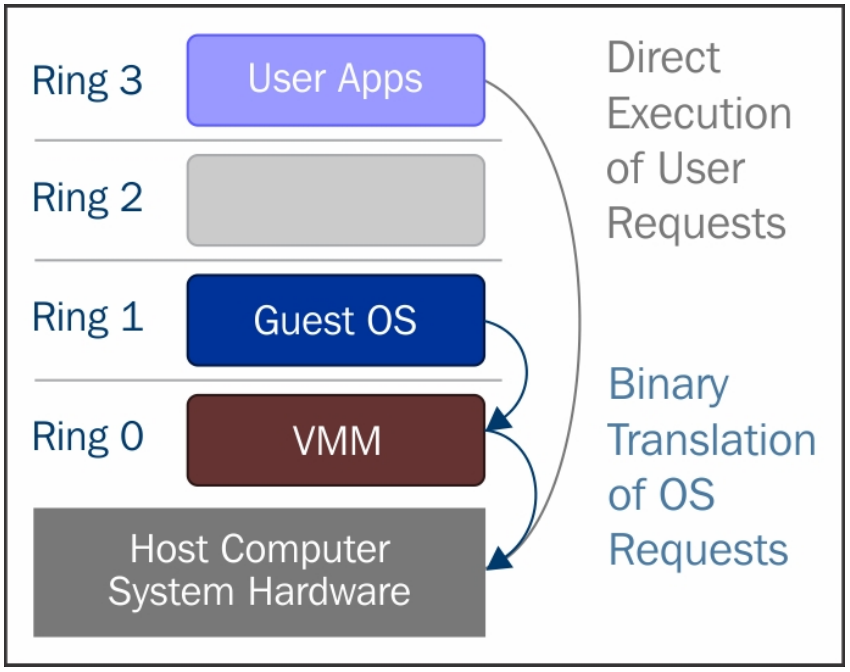


Efficient Full Virtualization

- When all sensitive instructions are also privileged
  - E.g. they trap when are not executing in system mode
  - Full virtualization can be achieved by running guest OS in user mode and VMM in system mode
  - When guest OS attempts to run a sensitive instruction in user mode, it will be trapped to VMM, who is responsible for managing the hardware globally
- When some sensitive instructions are not privileged
  - They don't trap automatically
  - They are referred to as "***critical instructions***"
  - The existence of such critical instruction makes virtualizing x86 based system very hard in early days

Full virtualization with traditional ring

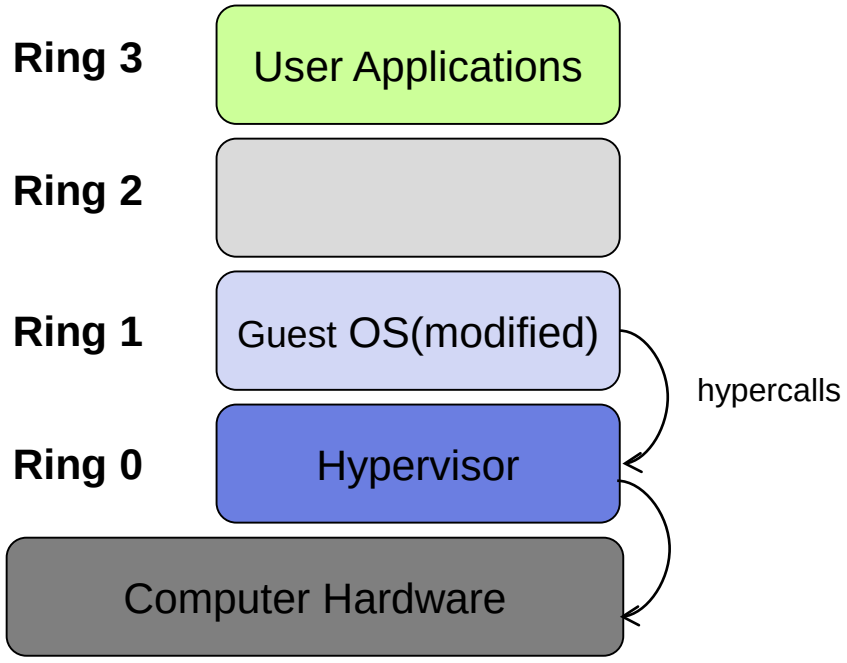
- VMWare products
- No Guest OS modification
- It relies on techniques, such as binary translation to trap and virtualize the execution of certain instructions.
- These instructions are discovered (statically or dynamically at runtime) and replaced with traps into the hypervisor(VMM) that are to be emulated in software.
- A binary translation can incur a large performance overhead



Mastering KVM Virtualization

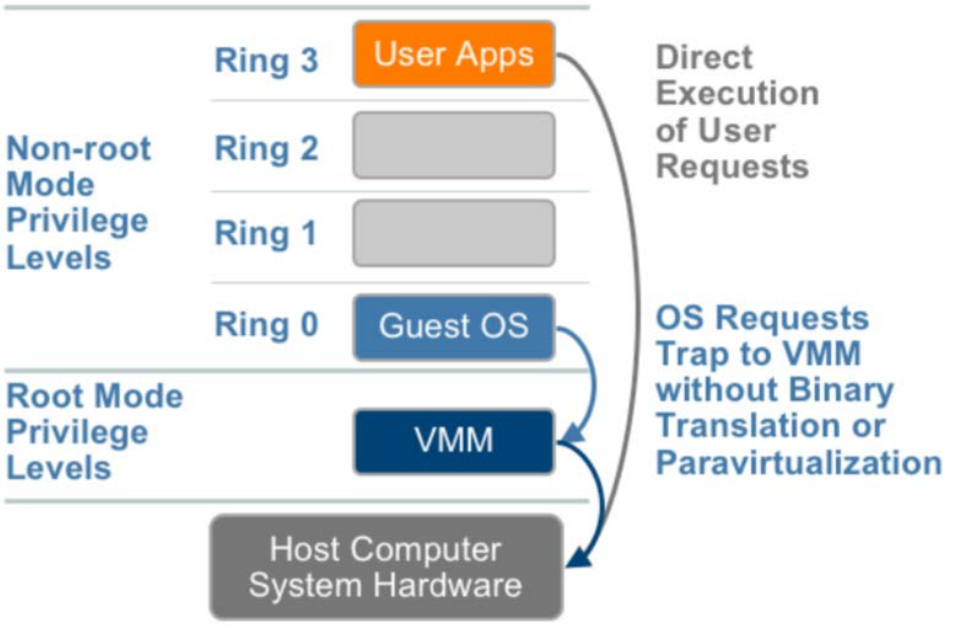
Paravirtualization

- Example: Xen hypervisor
- The hypervisor runs in ring 0, where the kernel normally runs.
- The guest kernel is moved to ring 1.
- User apps. runs in ring 3 - just like it does today – no user space changes necessary.
- Guest OS is **modified** to call through Xen (‘*hypercalls*’) for privileged instructions; Guest OS is aware that it is running on a VM to some extent
- Xen validates first whether allowed then executes on behalf of the guest OS



Hardware-Assisted Full Virtualization

- Intel and AMD independently created new processor extensions of the x86 architecture, called Intel VT-x and AMD-V respectively.
- Hypervisor runs in the new mode (e.g. root mode in VT-x)
- OS requests are trapped to hypervisor without binary translation or paravirtualization
- OS modification is not required since certain instructions will be trapped automatically to hypervisor
- KVM is a hardware assisted full virtualization



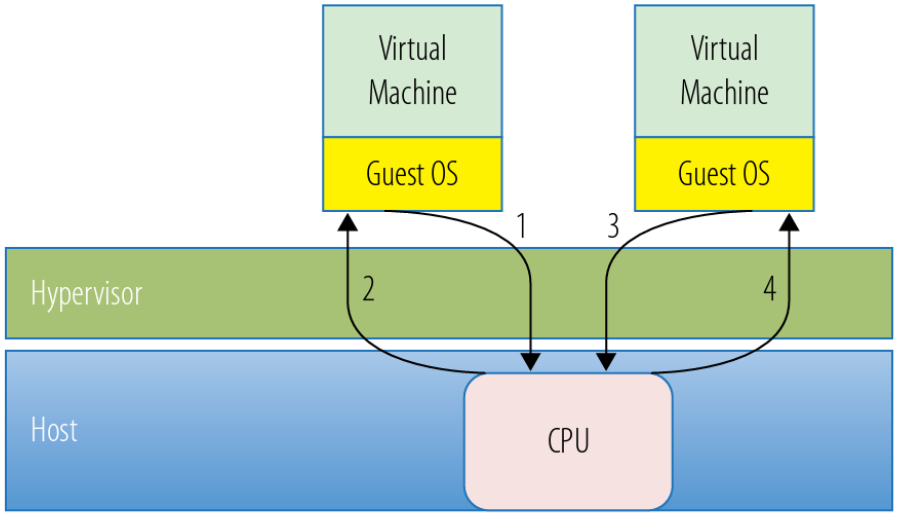
https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf

Outline

- Virtualization Technology Overview
- Server Virtualization
- **Resource managements**
  - **CPU**
  - **Memory**
  - **I/O**
  - **GPU**
- Brief Intro to AWS EC2 and Related Concepts

Managing CPUs for a VM

- Physical CPU cores are time shared by virtual CPUs(vCPU) in server virtualization
- The dynamic scheduling of vCPUs on physical CPU is managed by hypervisor following customized scheduler algorithm
  - E.g. credit scheduler of Xen



Multiprocessors, cores and hyperthreading

- A host server may be configured with multiple processor chips, each with multiple core
- The core may support hyperthreading to have two or more logic cores for one physical core

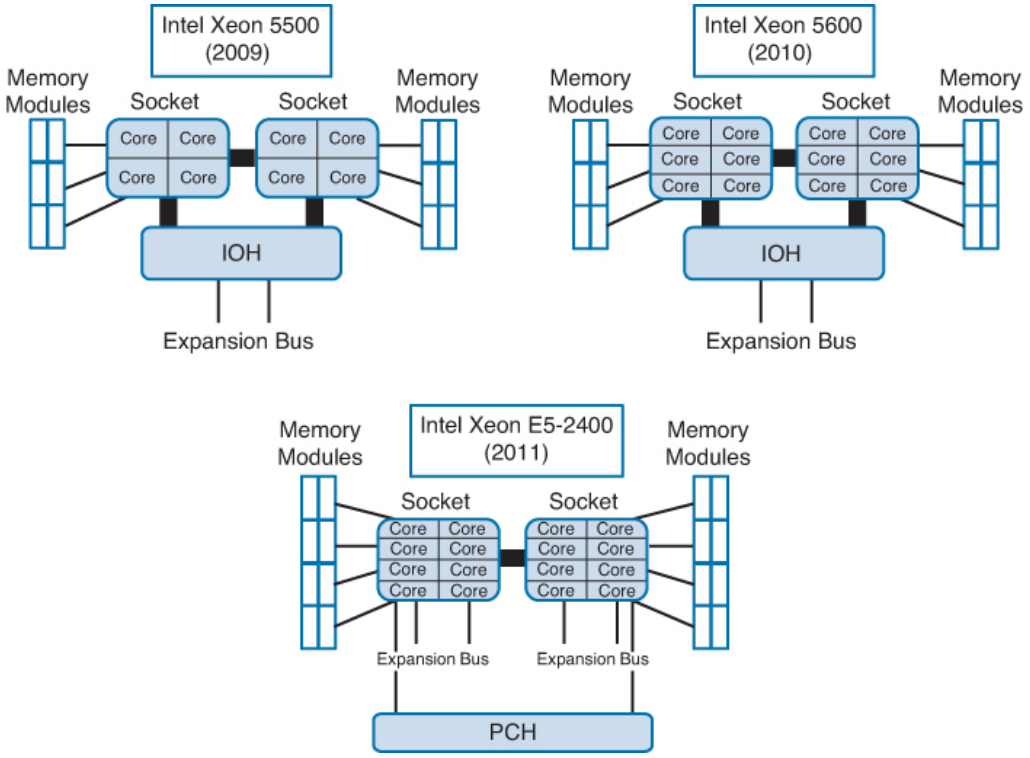


Figure 13.2 in Data Center Virtualization Fundamentals

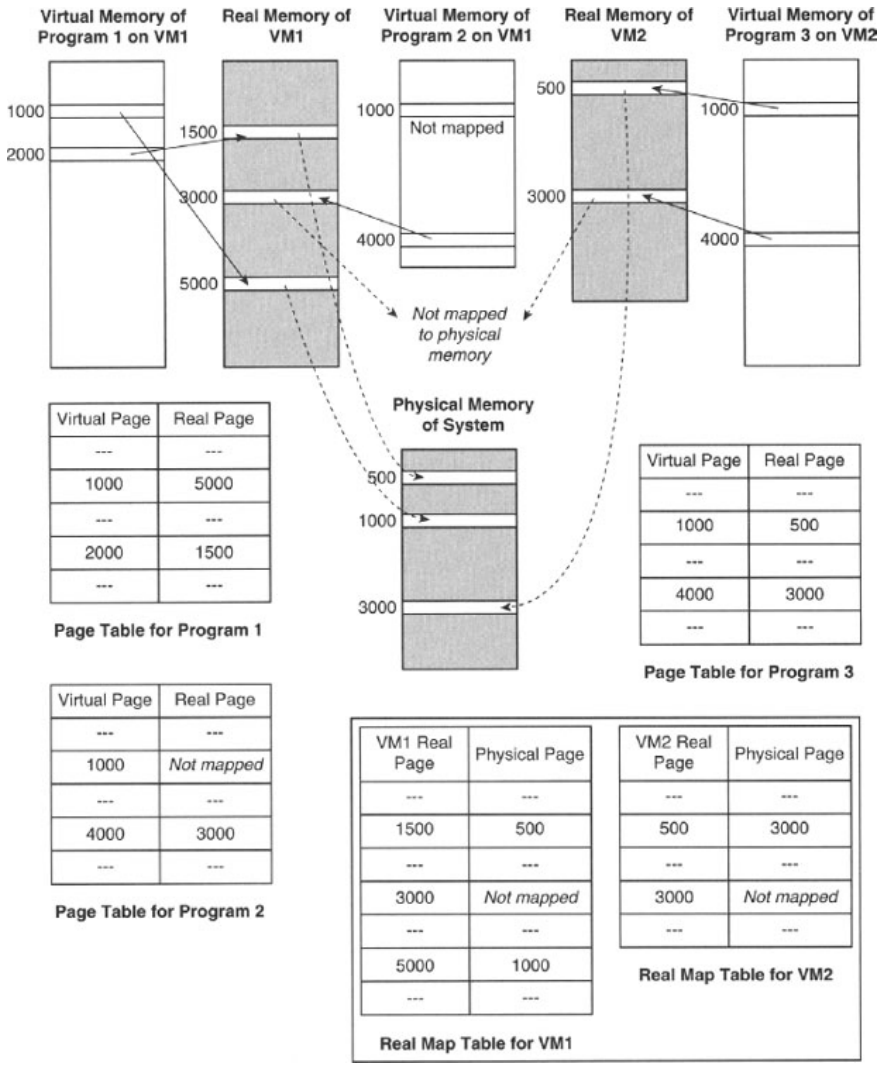
VM with multiple vCPUs

- vCPU count of VM
  - Means the maximum number of threads that the VM can run at any given moment.
- vCPU and physical CPU(pCPU)
  - A VM can run on any and all of the host CPUs over a period of time
    - Some hypervisor may support pinning
  - For a virtual machine with multiple vCPUs, the hypervisor will need to schedule each vCPU on a different pCPU (logic core also count)
    - The maximum number of vCPU you can assign to a single VM is bound by the hardware capacity e.g. maximum number of cores
    - A VM with more vCPUs may have longer waiting time to get scheduled
  - Most applications are not designed to run on multiple threads
    - Configure a VM with too many vCPUs may not have any benefit
    - VM with 2 vCPU is a very typical setting

Virtualizing Memory Management

- Memory Management
  - OS is designed to manage the mapping of virtual memory and physical memory through TLB and page tables
  - Guest OS usually gets a dedicated portion of the physical memory to ensure strong isolation
  - Guest OS does not know other part of the memory and should not interfere other VM's memory
  - It should not even know which part of the physical memory it is using
  - Only VMM has access to the physical memory and can manage the mapping
  - There is a level of indirection here
    - Two level mapping/translation
  - TLB (translation lookaside buffer) is used to speed up the translation
    - Could be managed jointly by hardware and OS

Memory Mapping in Guest OS



Virtualizing I/O

- I/O refers to all input/output devices
  - Disk, network, printer, monitor, mouse
- On the one hand there are many devices with different ways of controlling them
- On the other hand, operation systems have efficient ways to deal with I/O by providing standard interface and by loading various drivers.

Disk and Network

- Disk is partitioned device
  - Each VM can get a partition of it exposed as virtual disk
  - External disk drivers can be mounted easily
- Network adapter is shared device
  - The physical adapter can be time shared by each VM
  - A virtual network adapter is presented to the VM
- The request are passed to and managed by VMM

I/O pathway

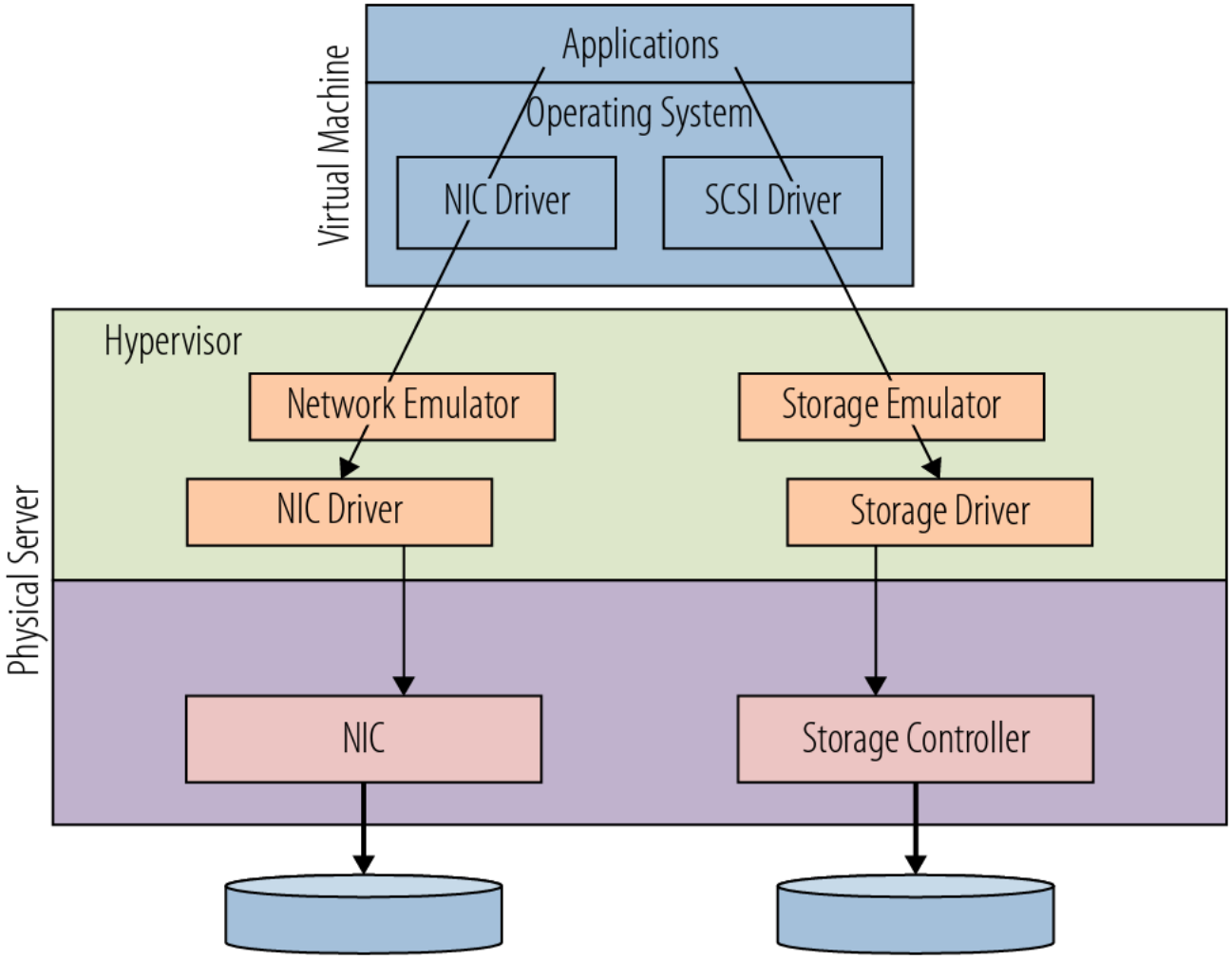


Figure 9.1 in Virtualization Essentials

I/O pathway in Xen

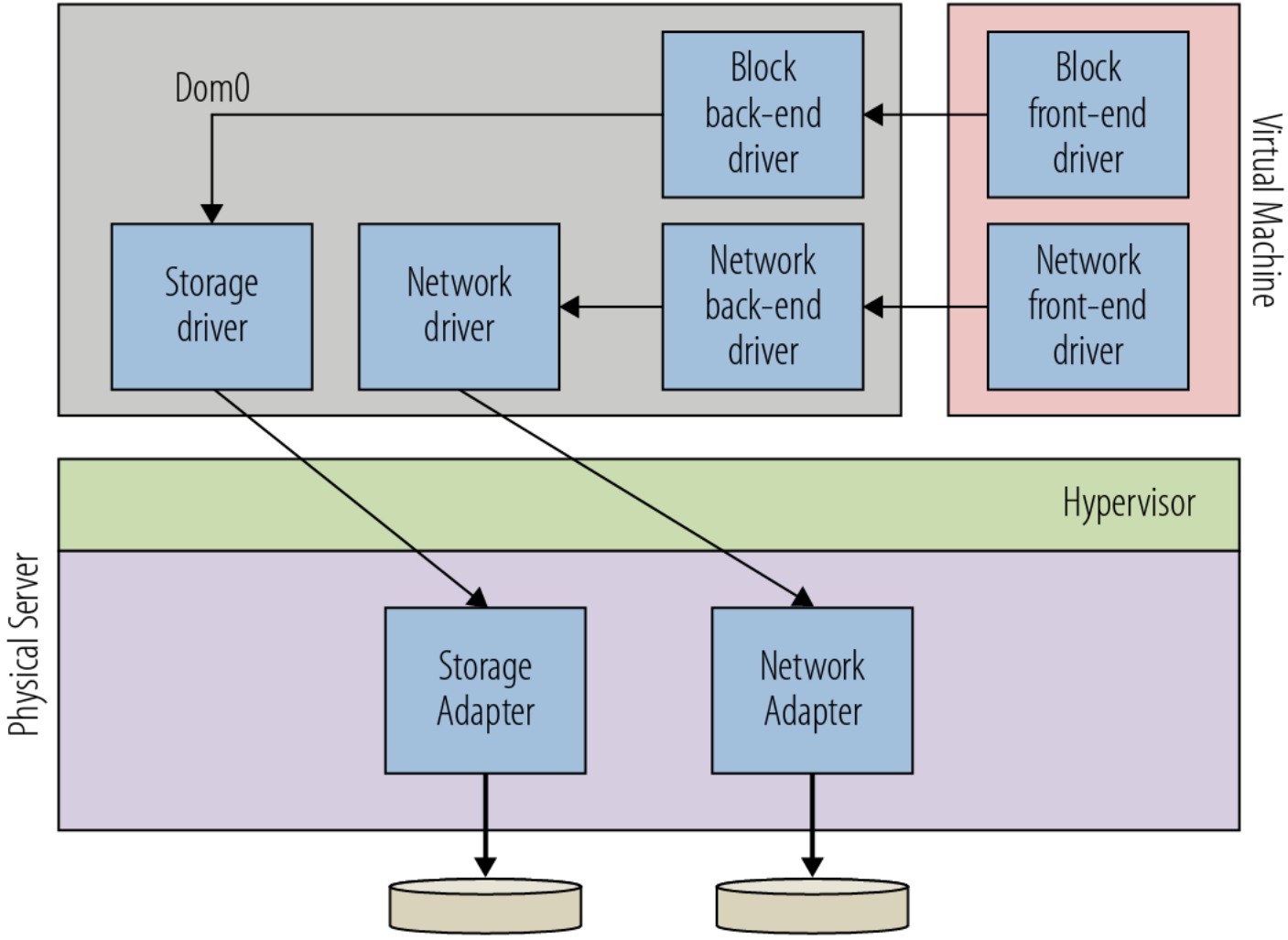


Figure 9.2 in Virtualization Essentials

Within host communication

- Hypervisor can provide two types of virtual switches
  - External virtual switch is connected to network card
  - Internal virtual switch only connects VMs on the same host

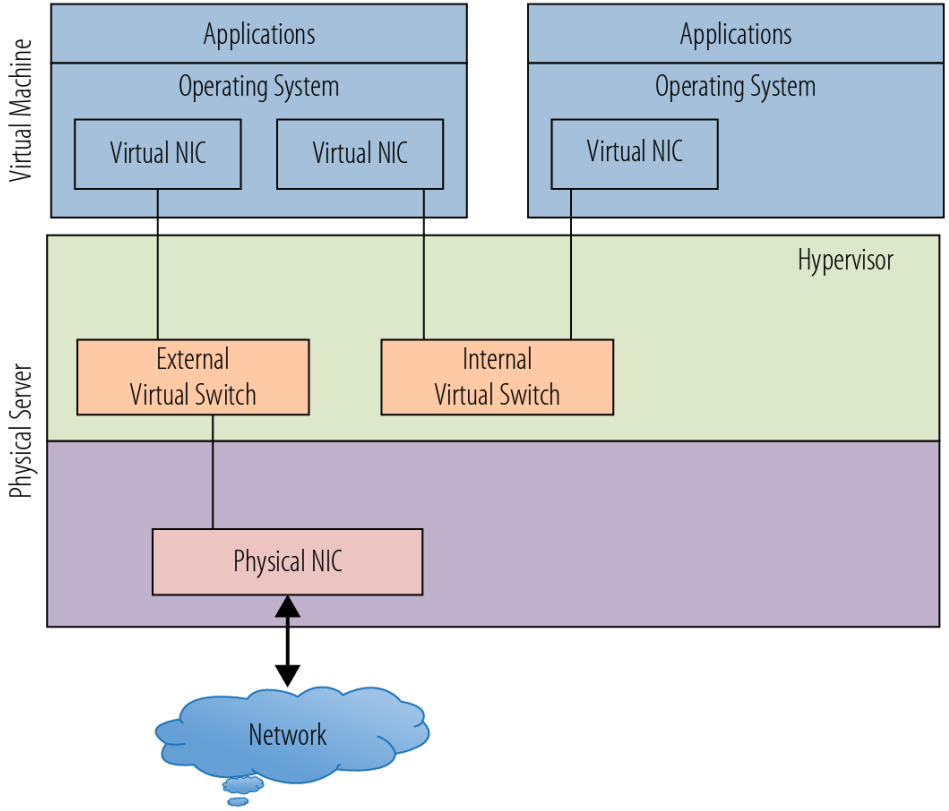


Figure 10.2 in Virtualization Essentials

Configure Network Segmentation

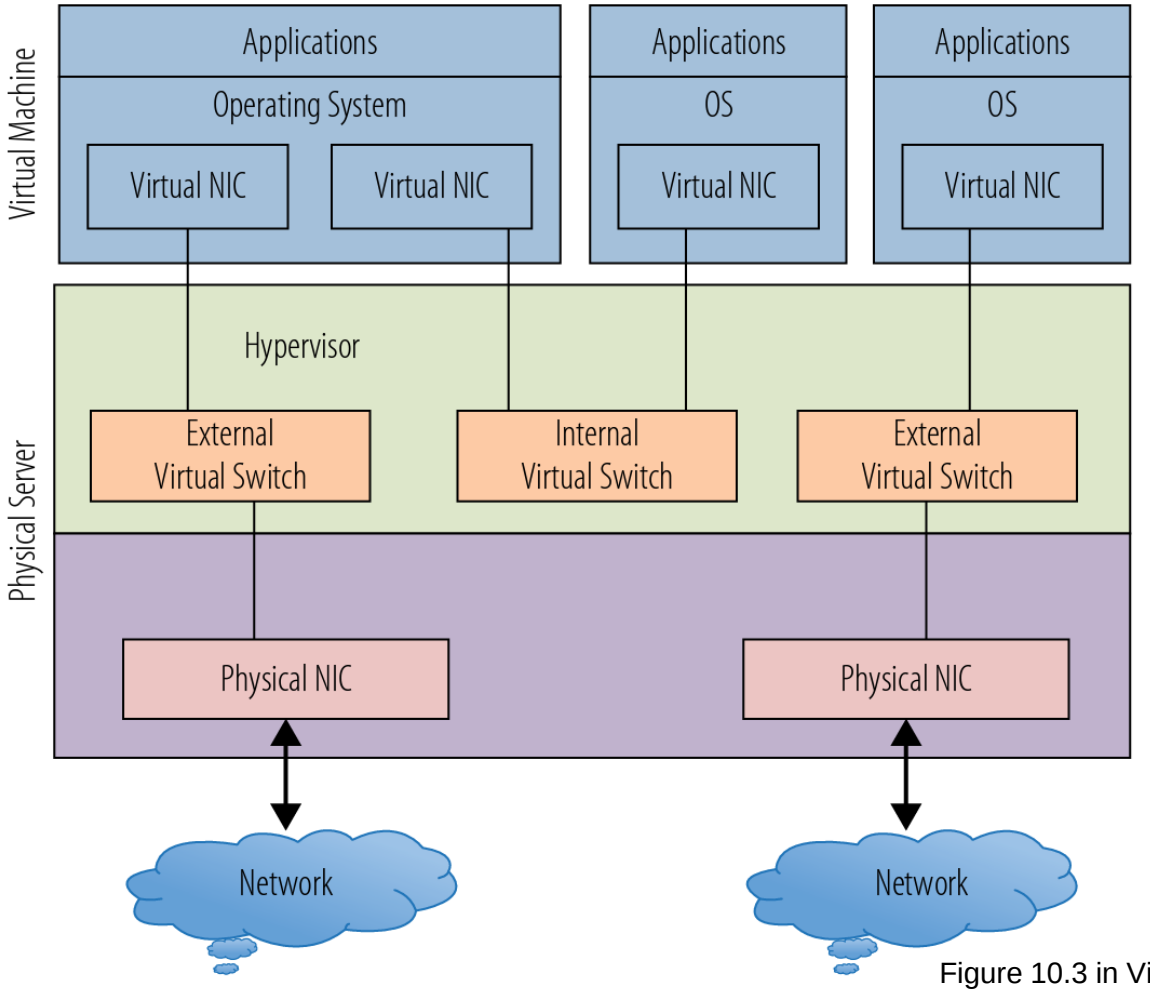
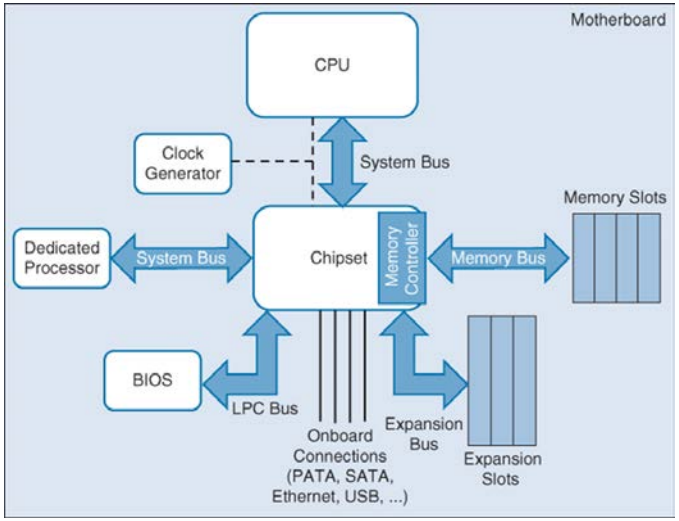
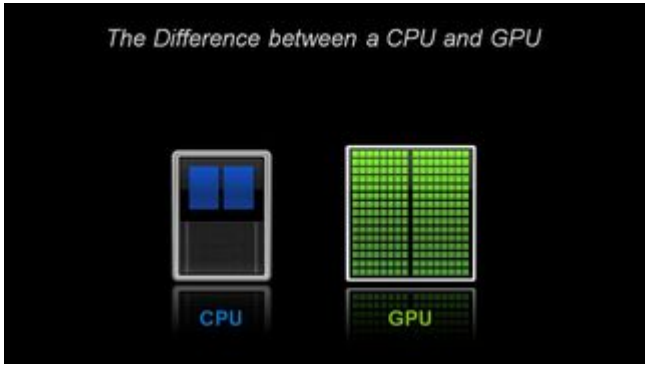


Figure 10.3 in Virtualization Essentials

GPU vs. CPU



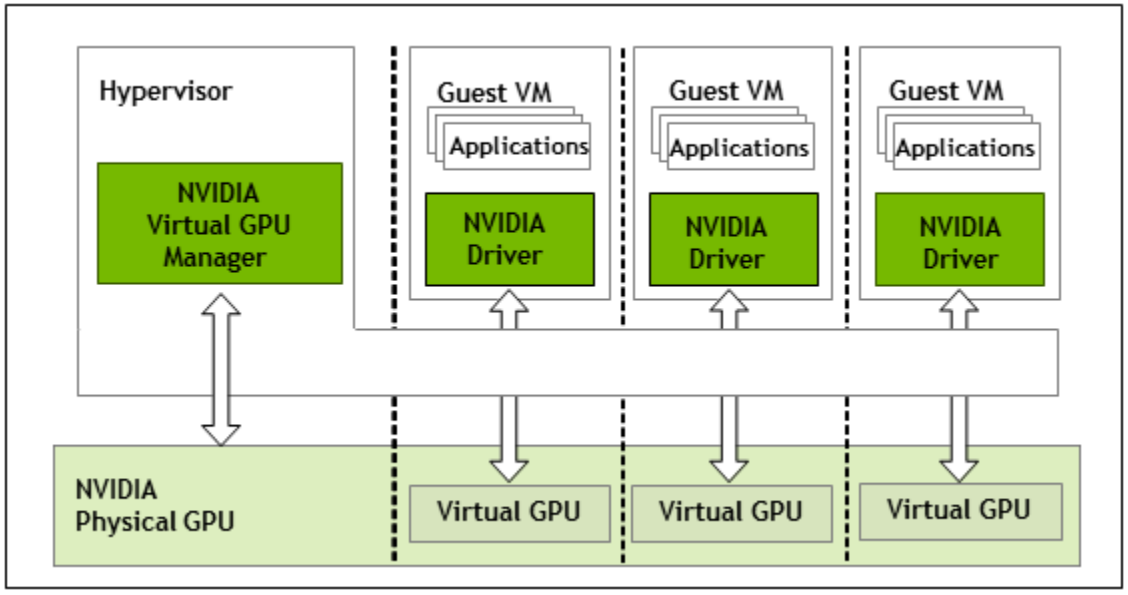


CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once

https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

GPU Virtualization

- Dedicated GPU per VM using Hypervisor pass through
  - An entire physical GPU is directly assigned to a virtual machine. GPU is accessed by corresponding drivers running in the VM. It is not shared among VMs
- Share physical GPU among multiple VMs
  - E.g. using special vGPU management software to slice the GPU resources among VMs.



https://docs.nvidia.com/grid/5.0/grid-vgpu-user-guide/index.html#architecture-grid-vgpu

Outline

- Virtualization Technology Overview
- Server Virtualization
- Resource managements
- **Brief Intro to AWS EC2 and Related Concepts**

AWS EC2

- Just virtual machines (instances) running on Amazon's data centers
- Related concepts
  - Instance Type
  - Amazon Machine Image (AMI)
  - EC2 Key Pair
  - Security Group
  - Elastic IP
  - Elastic Block Storage (EBS)
  - EC2 CloudWatch
  - EC2 Elastic Load Balancer
  - EC2 Auto Scale

Instance Types

Model	vCPU*	CPU Credits/hour	Mem (GiB)	Storage	Network Performance (Gbps)
t3.nano	2	6	0.5	EBS-Only	Up to 5
t3.micro	2	12	1	EBS-Only	Up to 5
t3.small	2	24	2	EBS-Only	Up to 5
t3.medium	2	24	4	EBS-Only	Up to 5

T3 instances accumulate CPU credits when a workload is operating below baseline threshold. Each earned CPU credit provides the T3 instance the opportunity to burst with the performance of a full CPU core for one minute when needed.

https://aws.amazon.com/ec2/instance-types/

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0052 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0104 per Hour
t3.small	2	Variable	2 GiB	EBS Only	$0.0208 per Hour
t3.medium	2	Variable	4 GiB	EBS Only	$0.0416 per Hour

https://aws.amazon.com/ec2/pricing/on-demand/

AWS dedicated GPU instance

P3 | P2 | Inf1 | G4 | G3 | F1

P3 instances are the latest generation of general purpose GPU instances.

Features:

- Up to 8 NVIDIA Tesla V100 GPUs, each pairing 5,120 CUDA Cores and 640 Tensor Cores
- High frequency Intel Xeon E5-2686 v4 (Broadwell) processors for p3.2xlarge, p3.8xlarge, and p3.16xlarge.
- High frequency 2.5 GHz (base) Intel Xeon P-8175M processors for p3dn.24xlarge.
- Supports NVLink for peer-to-peer GPU communication
- Provides up to 100 Gbps of aggregate network bandwidth.
- EFA support on p3dn.24xlarge instances

Instance	GPUs	vCPU	Mem (GiB)	GPU Mem (GiB)	GPU P2P	Storage (GB)	Dedicated EBS Bandwidth	Networking Performance
p3.2xlarge	1	8	61	16	-	EBS-Only	1.5 Gbps	Up to 10 Gigabit
p3.8xlarge	4	32	244	64	NVLink	EBS-Only	7 Gbps	10 Gigabit

AWS shared GPU instance

G3 instances are optimized for graphics-intensive applications.

Features:

- High frequency Intel Xeon E5-2686 v4 (Broadwell) processors
- NVIDIA Tesla M60 GPUs, each with 2048 parallel processing cores and 8 GiB of video memory
- Enables NVIDIA GRID Virtual Workstation features, including support for 4 monitors with resolutions up to 4096x2160. Each GPU included in your instance is licensed for one “Concurrent Connected User"
- Enables NVIDIA GRID Virtual Application capabilities for application virtualization software like Citrix XenApp Essentials and VMware Horizon, supporting up to 25 concurrent users per GPU
- Each GPU features an on-board hardware video encoder designed to support up to 10 H.265 (HEVC) 1080p30 streams and up to 18 H.264 1080p30 streams, enabling low-latency frame capture and encoding, and high-quality interactive streaming experiences
- Enhanced Networking using the Elastic Network Adapter (ENA) with 25 Gbps of aggregate network bandwidth within a Placement Group

Instance	GPUs	vCPU	Mem (GiB)	GPU Memory (GiB)	Network Performance
g3s.xlarge	1	4	30.5	8	Up to 10 Gigabit
g3.4xlarge	1	16	122	8	Up to 10 Gigabit

Amazon Machine Image

- Very similar to the root drive of a computer
- Contains OS and additional software customized for various purpose
  - Windows server
  - Windows server various MS products
  - Linux server
  - LAMP Web Starter
  - Deep Learning AMI
  - Container AMI
  - Etc.

EC2 Key Pair

- Key based authentication is a more secure option than standard password authentication
  - Involves a pair of public and private keys
  - The server stores the public keys of all legitimate users, who supply respective private keys to login to their accounts
- When an EC2 instance is created, it only allows key based authentication
  - User needs to specify an existing key pair or create a new key pair
  - AWS stores only the public key and associate it with the initial account of that instance
    - User will be prompted to download the private key after the pair is created
  - The initial account (ec2-user, ubuntu, …) is given **sudo** privilege
    - Root level access
  - User can enable password based authentication on their own instance

Security Group

- EC2 instances are protected by a firewall, which blocks all internal and external connectivity by default. The only port opened is 22
- Security group is an easy way of configuring the firewall rules
- Instances within a single security group can communicate freely with each other

Group Name	Rules
web_access	allow HTTP access (port 8080) from anywhere allow SSH access (port 22) from anywhere
db_access	allow access to MySQL(port 3306) from anywhere allow SSH access (port 22) from anywhere

Elastic IP and Elastic Block Store

- Each instance by default is assigned a public IP address. it is gone when the instance is terminated.
- Elastic IP is a stable IP that can be assigned to any instances associated with an AWS account
- EBS is an addressable disk volume that can be attached to any running instance and used it as local disk
  - Similar to NFS **mount** command
  - Many instance types only supports EBS storage

Persistent and Ephemeral Resources

- Persistent resources
  - Elastic IP Address
  - Elastic Block Storage Volumes
  - Elastic Load Balancers
  - Security Groups
  - AMI stored in Amazon S3
  - Key pairs (public key)
- Ephemeral resources
  - EC2 instance (IP, memory, and its local storage!)
- Instances may be stopped or terminated
  - A *stopped* instance has its image and EBS saved in S3 and can be restarted later; a new IP address will be assigned; storage cost is incurred
  - A *terminated* instance is gone forever

Next Week

- The topic for next week is “Container Technology”
- Readings for next week:
  - Dirk Merkel, Docker: Lightweight Linux Containers for Consistent Development and Deployment, Linux Journal, May 19, 2014 https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment
  - Container’s Anatomy, https://www.slideshare.net/jpetazzo/anatomy-of-a-container-namespaces-cgroups-some-filesystem-magic-linuxcon

References

- Gustavo A., A. Santana: **Data Center Virtualization Fundamentals: Understanding Techniques and Designs for Highly Efficient Data Centers with Cisco Nexus, UCS, MDS, and Beyond**
  - **Chapter 1, chapter 13**
- Virtualization Essentials, Second Edition
  - Matthew Portnoy
- Jim Smith (Author), Ravi Nair , **Virtual Machines: Versatile Platforms for Systems and Processes,** Morgan Kaufmann; 1 edition (June 17, 2005)
  - Chapter 1
  - Chapter 8 System Virtual Machines
- *Xen and the Art of Virtualization*. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. In Proceedings of the Nineteenth ACM Symposium on Operating systems principles (SOSP '03), 2003.