# Basics of Virtualization

Module 3

## Virtualization in Cloud Computing

**Virtualization** is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".

In other words, Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

A Virtual machine provides an environment that is logically separated from the underlying hardware.

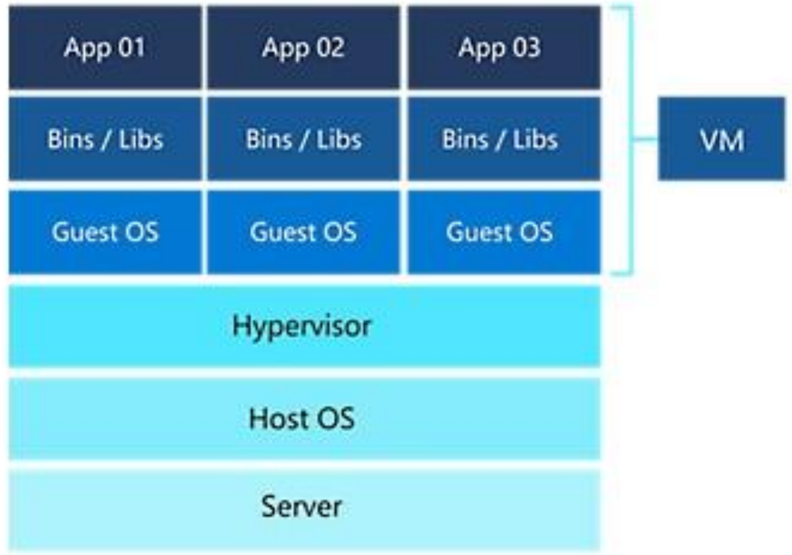
***The machine on which the virtual machine is going to create is known as Host Machine and that virtual machine is referred as a Guest Machine***

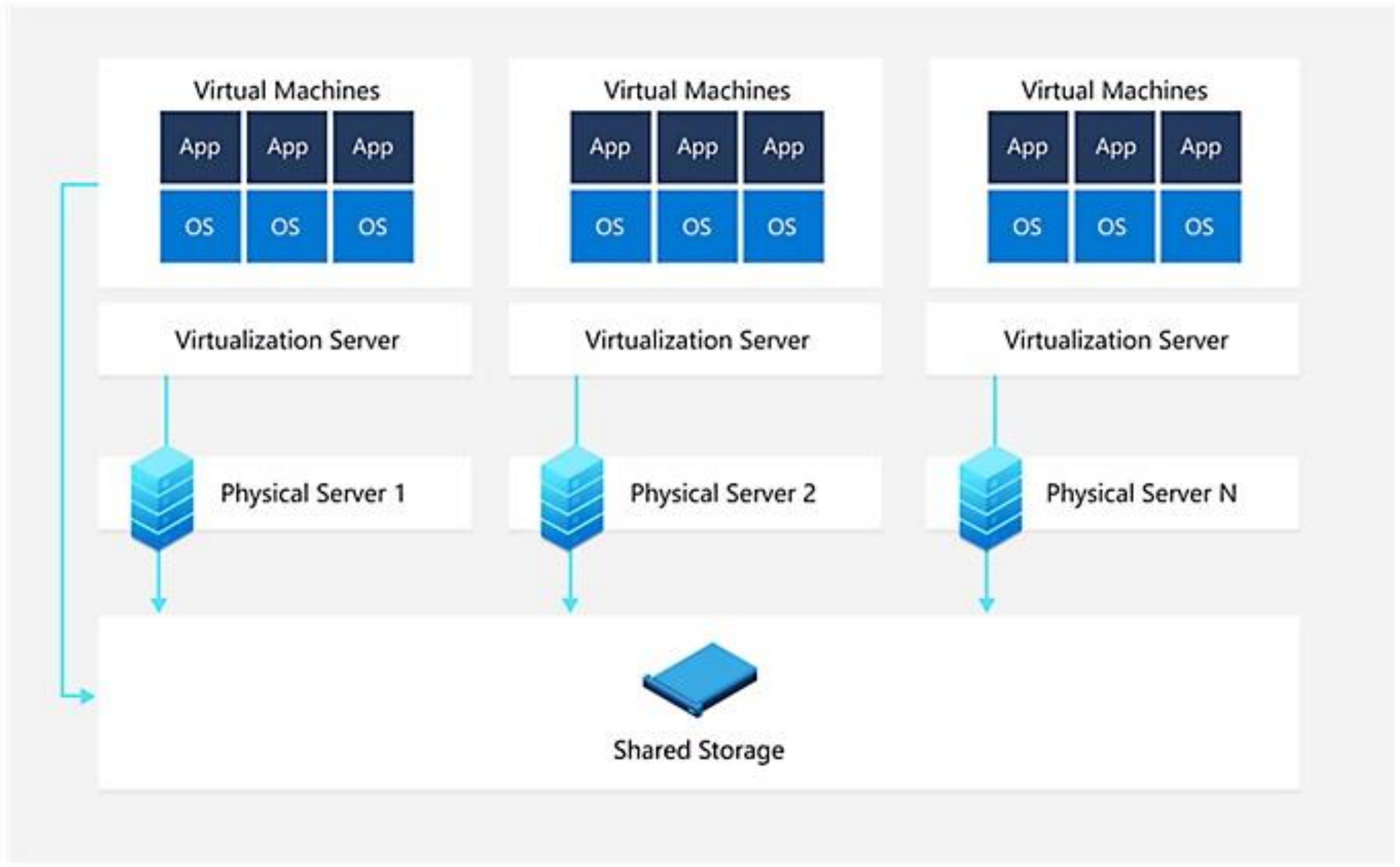
## Virtual machines: virtual computers within computers

Virtualization is the process of creating a software-based, or "virtual" version of a computer, with dedicated amounts of CPU, memory, and storage that are "borrowed" from a physical host computer—such as your personal computer— and/or a remote server—such as a server in a cloud provider's datacenter.

A virtual machine is a computer file, typically called an image, that behaves like an actual computer. It can run in a window as a separate computing environment, often to run a different operating system—or even to function as the user's entire computer experience—as is common on many people's work computers.

The virtual machine is partitioned from the rest of the system, meaning that the software inside a VM can't interfere with the host computer's primary operating system.


App 01
App 02
App 03
Bins / Libs
Bins / Libs
Bins / Libs
Guest OS
Guest OS
Guest OS
VM
Hypervisor
Host OS
Server


Virtual Machines
App
App
App
OS
OS
OS
Virtual Machines
App
App
App
OS
OS
OS
Virtual Machines
App
App
App
OS
OS
OS
Virtualization Server
Virtualization Server
Virtualization Server
Physical Server 1
Physical Server 2
Physical Server N
Shared Storage

## What are VMs used for?

•Building and deploying apps to the cloud.

•Trying out a new operating system (OS), including beta releases.

•Spinning up a new environment to make it simpler and quicker for developers to run dev-test scenarios.

•Backing up your existing OS.

•Accessing virus-infected data or running an old application by installing an older OS.

•Running software or apps on operating systems that they weren't originally intended for.

## What are the benefits of using VMs?

•**Cost savings**—running multiple virtual environments from one piece of infrastructure means that you can drastically reduce your physical infrastructure footprint. This boosts your bottom line—decreasing the need to maintain nearly as many servers and saving on maintenance costs and electricity.
•**Agility and speed**—Spinning up a VM is relatively easy and quick and is much simpler than provisioning an entire new environment for your developers.
•**Lowered downtime**—VMs are so portable and easy to move from one hypervisor to another on a different machine—this means that they are a great solution for backup, in the event the host goes down unexpectedly.
•**Scalability**—VMs allow you to more easily scale your apps by adding more virtual servers to distribute the workload across multiple VMs. As a result you can increase the availability and performance of your apps.
•**Security benefits**— Because virtual machines run in multiple operating systems, using a guest operating system on a VM allows you to run apps of questionable security and protects your host operating system.

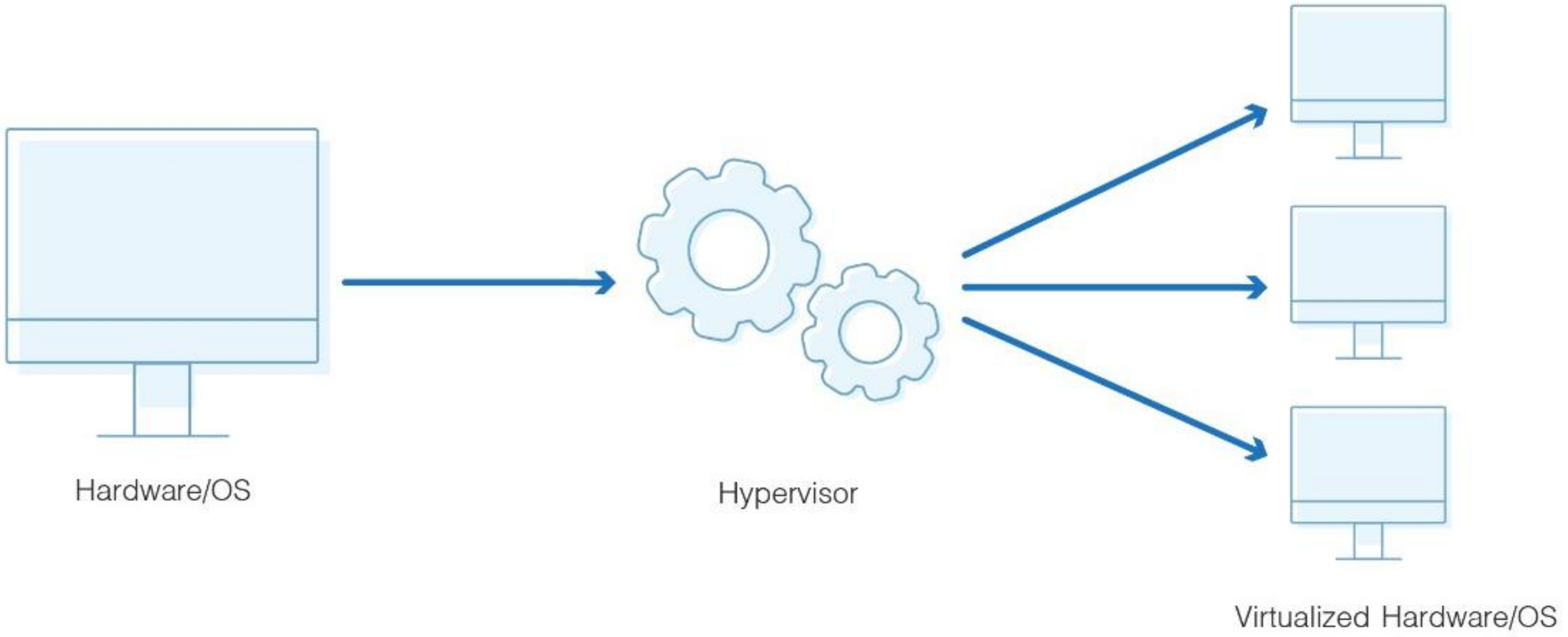
## What is a hypervisor?

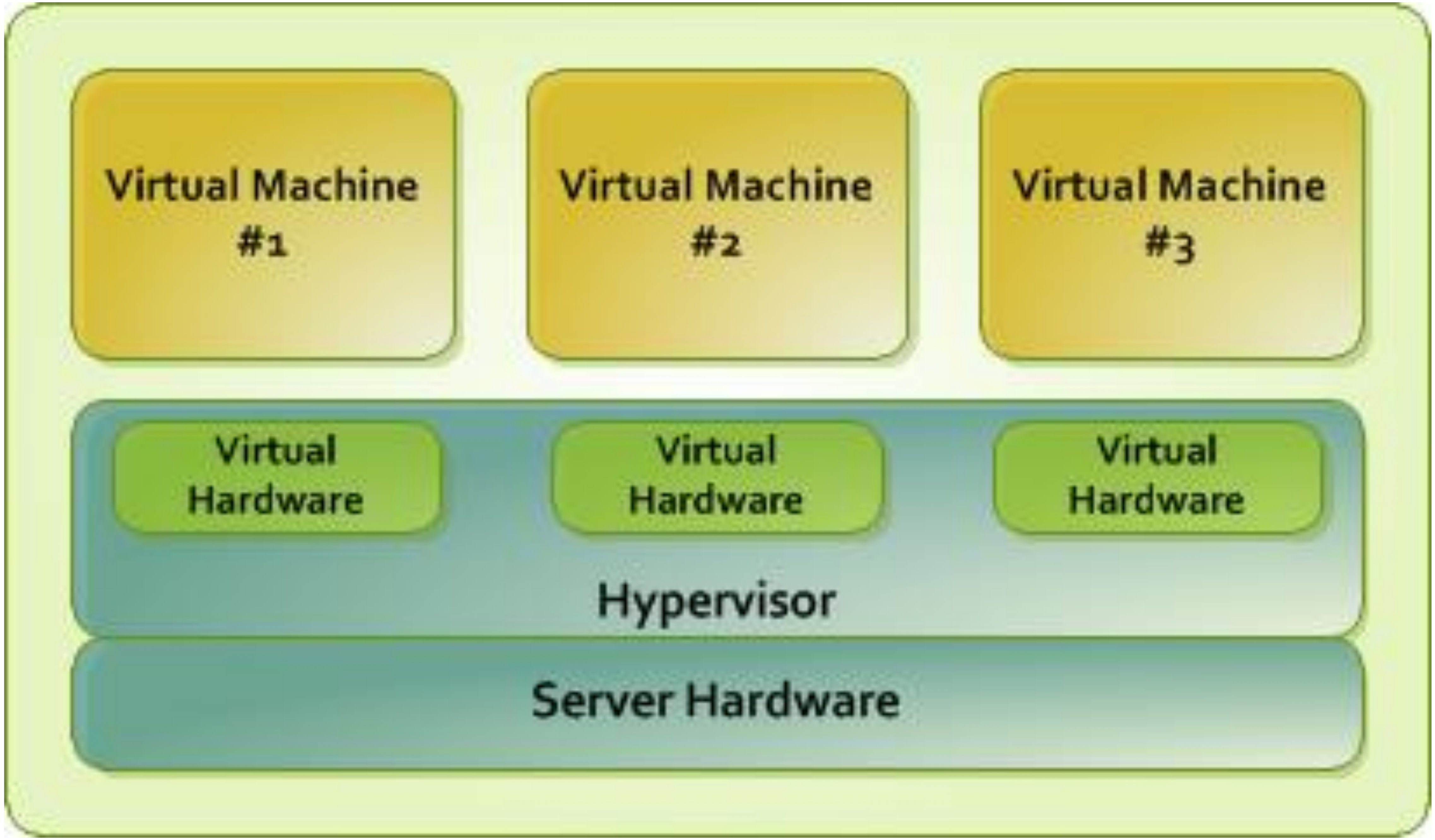
A **hypervisor**, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

The hypervisor treats resources—like CPU, memory, and storage—as a pool that can be easily reallocated between existing guests or to new virtual machines.

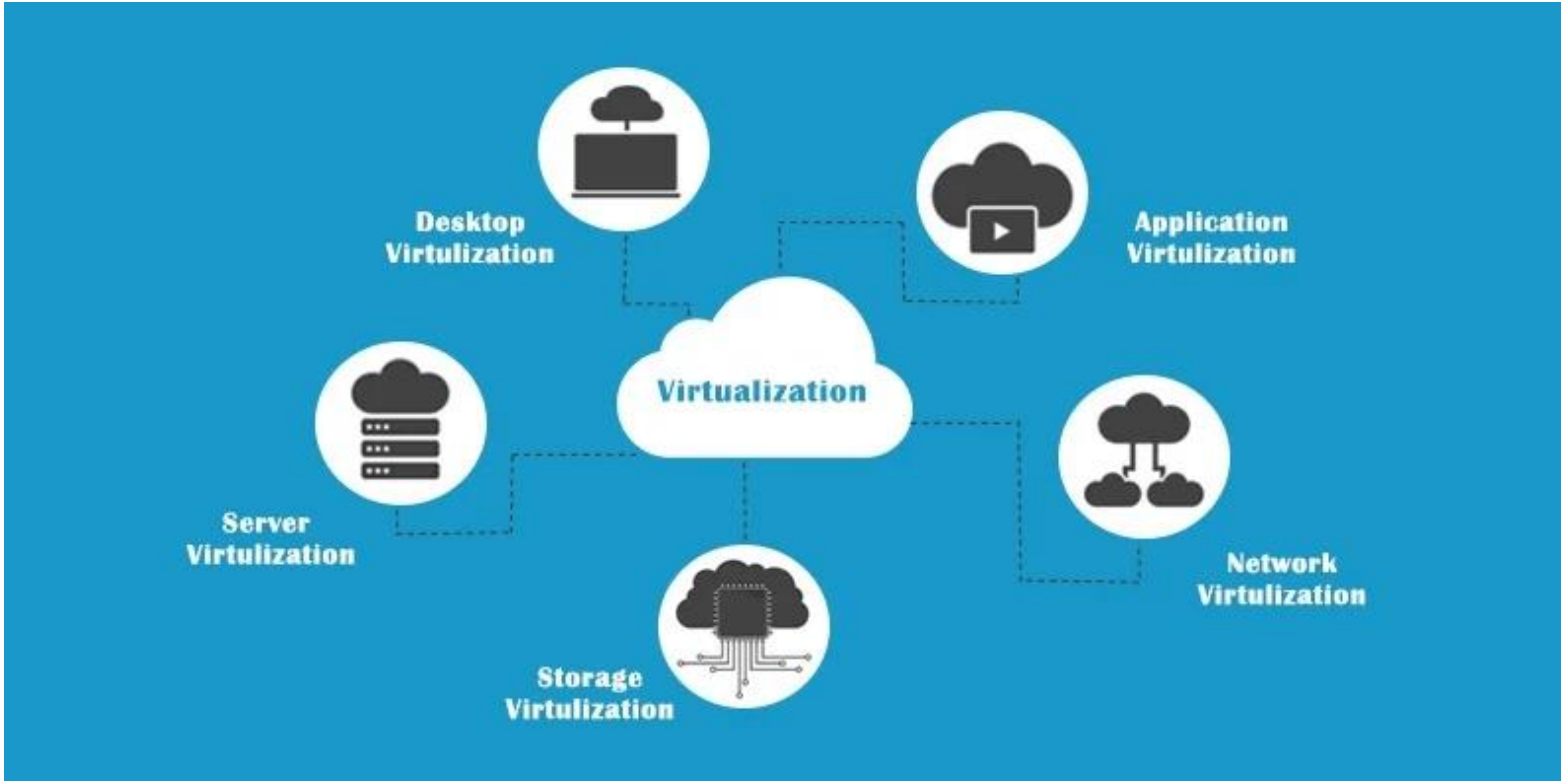
All hypervisors need some operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more—to run VMs.

## What Is a Hypervisor?




Virtual Machine #1
Virtual Machine #2
Virtual Machine #3
Virtual Hardware
Virtual Hardware
Virtual Hardware
Hypervisor
Server Hardware

## Types of Virtualization



## Server Virtualization

Server virtualization refers to partitioning the resources of a server, which consist of hardware, software and networking resources, and distributing them over a network.

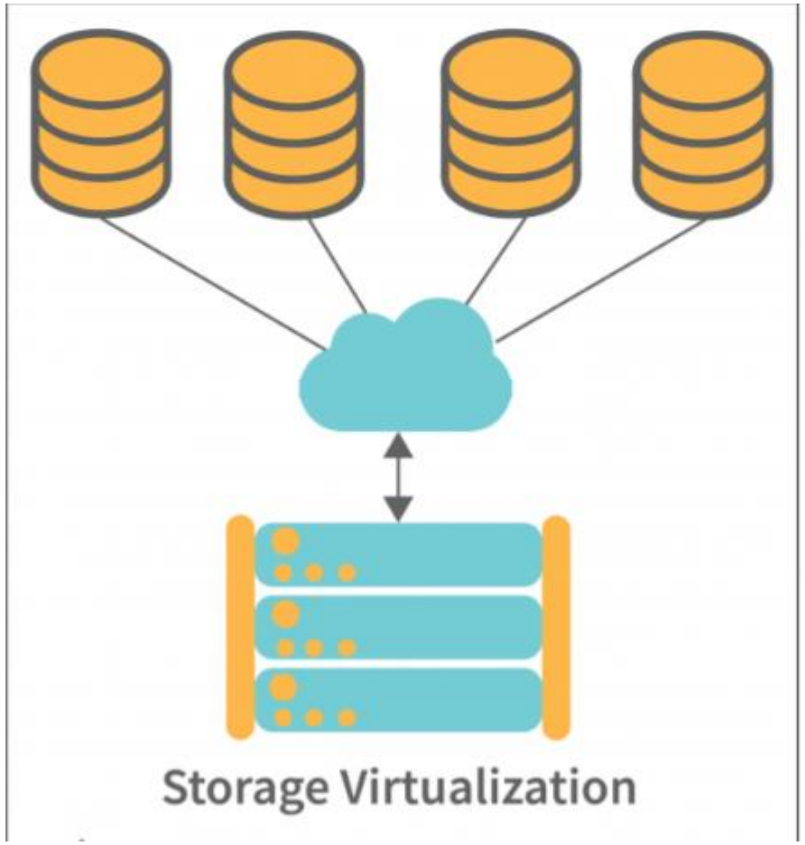
The partitions are instances of a powerful physical server lying in a remote location but acting like standalone servers. These partitions are also called virtual servers.

Server virtualization allows for flexible scalability as, depending upon their need, users can request variable configurations of storage, computing power, RAM, etc from the physical server.

Virtualizing a server comes in handy when users want to install different operating systems on a single assembly of computer components.

## Storage Virtualization

Storage virtualization works by gathering and merging multiple physical storage arrays and presenting them as a single storage location to the user over a network. It is employed typically by organizations and individuals looking to scale and maintain their systems' storage without investing in physical storage devices.



Virtualized storage is visible as a single storage entity (such as a 2 TB disk drive) on a user's system.

But, behind the scenes, storage virtualization is pooling several storage locations to offer 2 TB worth of storage to the user.

## RAID (redundant array of independent disks)

RAID (redundant array of independent disks) is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure

RAID works by placing data on multiple disks and allowing input/output (I/O) operations to overlap in a balanced way, improving performance. Because using multiple disks increases the mean time between failures, storing data redundantly also increases fault tolerance.

RAID arrays appear to the operating system (OS) as a single logical drive.

## Network Virtualization

Network virtualization refers to combining all the components of networks and administering them using only software. These network components include all the underlying hardware and software of a network with their respective functionalities.

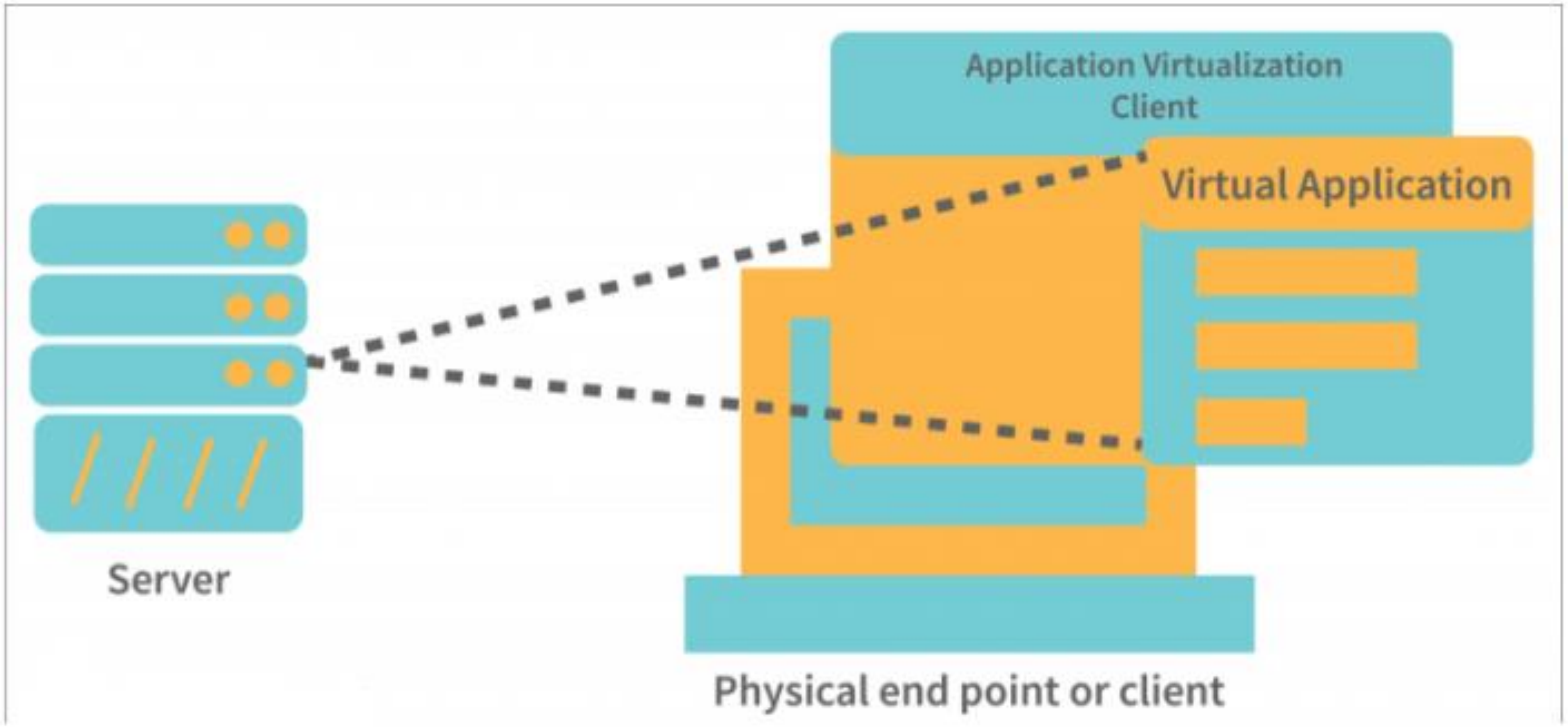
Virtualizing a network takes away its dependency on the software embedded in its underlying hardware, converting it into a virtual network.

Network virtualization is typically used to interconnect virtual machines, group several networks into one or subdivide the resources of a network.

Virtualizing networks enforces greater flexibility among the pooled networks and allows for better networking at reduced costs.

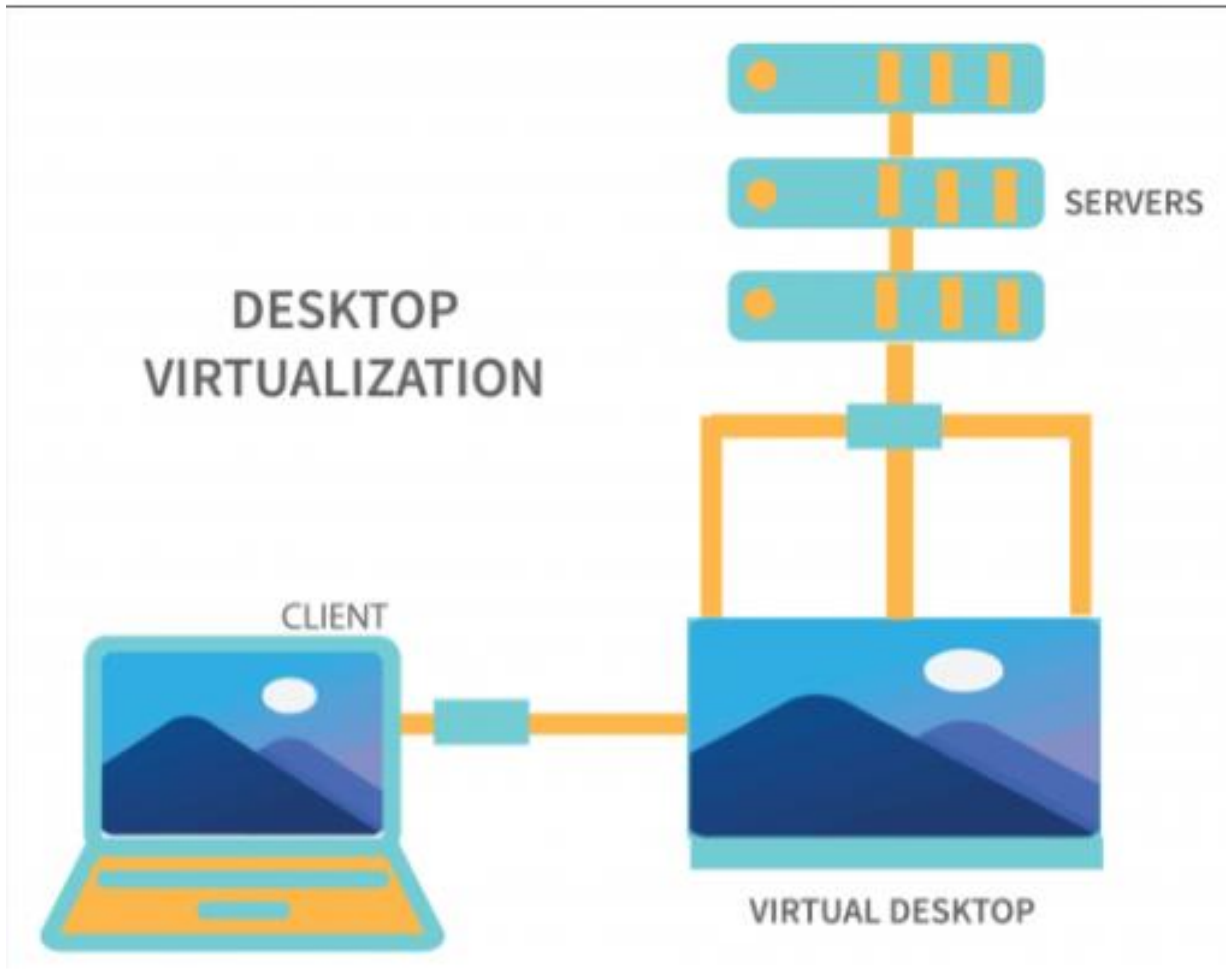
## Application Virtualization

Application virtualization refers to the process of deploying a computer application over a network (the cloud). The deployed application is installed locally on a server, and when a user requests it, an instance of the application is displayed to them. The user can then engage with that application as if it was installed on their system.

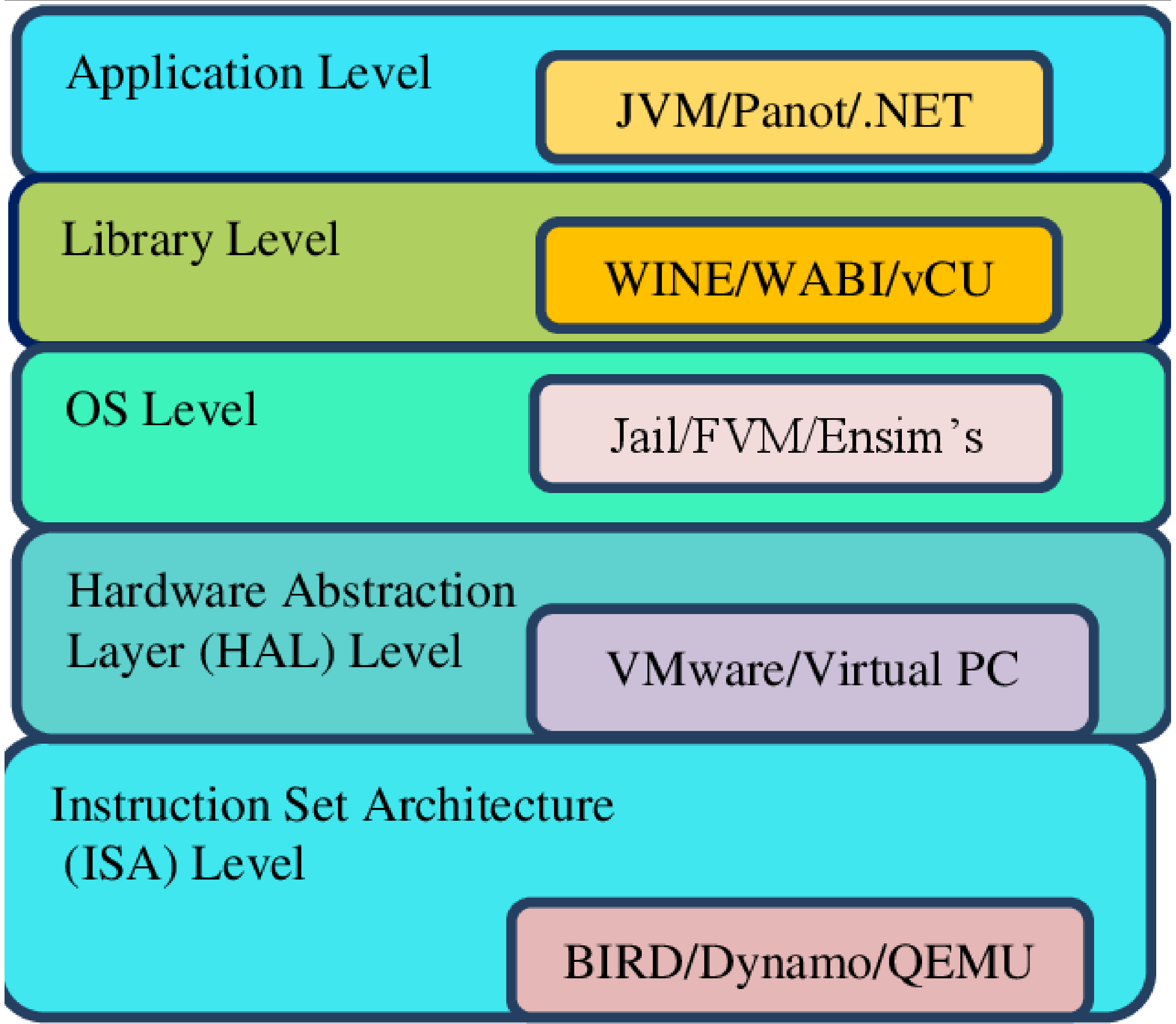

Application Virtualization Client
Virtual Application
Server
Physical end point or client

## Desktop Virtualization

Desktop virtualization is similar to application virtualization, but the apps are now replaced with whole desktop environments



# Implementation Levels of Virtualization



## Instruction Set Architecture Level (ISA)

ISA virtualization can work through ISA emulation. This is used to run many legacy codes written for a different hardware configuration.

These codes run on any virtual machine using the ISA. With this, a binary code that originally needed some additional layers to run is now capable of running on the x86 machines.

It can also be tweaked to run on the x64 machine. With ISA, it is possible to make the virtual machine hardware agnostic.

For the basic emulation, an interpreter is needed, which interprets the source code and then converts it into a hardware format that can be read. This then allows processing.

## Hardware Abstraction Level (HAL)

True to its name HAL lets the virtualization perform at the level of the hardware.

This makes use of a hypervisor which is used for functioning.

The virtual machine is formed at this level, which manages the hardware using the virtualization process.

It allows the virtualization of each of the hardware components, which could be the input-output device, the memory, the processor, etc.

## Operating System Level

At the level of the operating system, the virtualization model is capable of creating a layer that is abstract between the operating system and the application.

This is an isolated container on the operating system and the physical server, which uses the software and hardware. Each of these then functions in the form of a server.

When there are several users and no one wants to share the hardware, then this is where the virtualization level is used.

Every user will get his virtual environment using a dedicated virtual hardware resource.

In this way, there is no question of any conflict.

## Library Level

The operating system is cumbersome, and this is when the applications use the API from the libraries at a user level.

These APIs are documented well, and this is why the library virtualization level is preferred in these scenarios.

API hooks make it possible as it controls the link of communication from the application to the system.

## What is an API?

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications.

This contract defines how the two communicate with each other using requests and responses.

### API Hooks

API hooking is a vital part of how many operating systems work, and it enables developers to extend software functionality and to debug it.

Obviously, the ability to alter the code flow of APIs, and to intercept system messages and events has many legitimate uses

## Application Level

The application-level virtualization is used ***when there is a desire to virtualize only one application*** and is the last of the implementation levels of virtualization in Cloud Computing.

***One does not need to virtualize the entire environment of the platform***.

This is generally used when you run virtual machines that use high-level languages.

The application will sit above the virtualization layer, which in turn sits on the application program.

It lets the high-level language programs compiled to be used at the application level of the virtual machine run seamlessly.

**Table** Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

| Level of Implementation | Higher Performance | Application Flexibility | Implementation Complexity | Application Isolation |
|---|---|---|---|---|
| ISA | X | XXXXX | XXX | XXX |
| Hardware-level virtualization | XXXXX | XXX | XXXXX | XXXX |
| OS-level virtualization | XXXXX | XX | XXX | XX |
| Runtime library support | XXX | XX | XX | XX |
| User application level | XX | XX | XXXXX | XXXXX |

## VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

In general, there are three typical classes of VM architecture.

Figure showed the architectures of a machine before and after virtualization.

Before virtualization, the operating system manages the hardware.

After virtualization, a virtualization layer is inserted between the hardware and the operating system.

In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware. Therefore, different operating systems such as Linux and Windows can run on the same physical machine, simultaneously.

Depending on the position of the virtualization layer, ***there are several classes of VM architectures, namely the hypervisor architecture, para-virtualization, and host-based virtualization.***

## Hypervisor and Xen Architecture

The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces.

The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor.

The hypervisor provides hypercalls for the guest OSes and applications.

***Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.***

## The Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University.

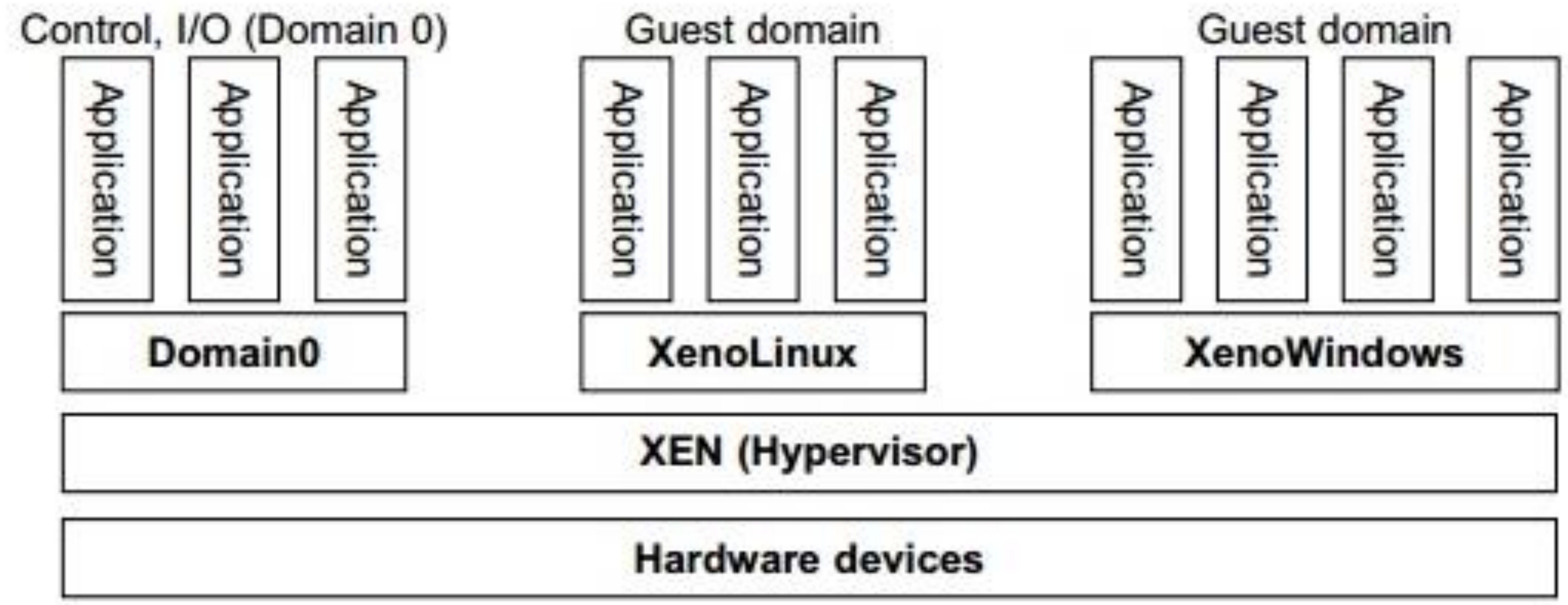
Xen is a micro-kernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0.

Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices.

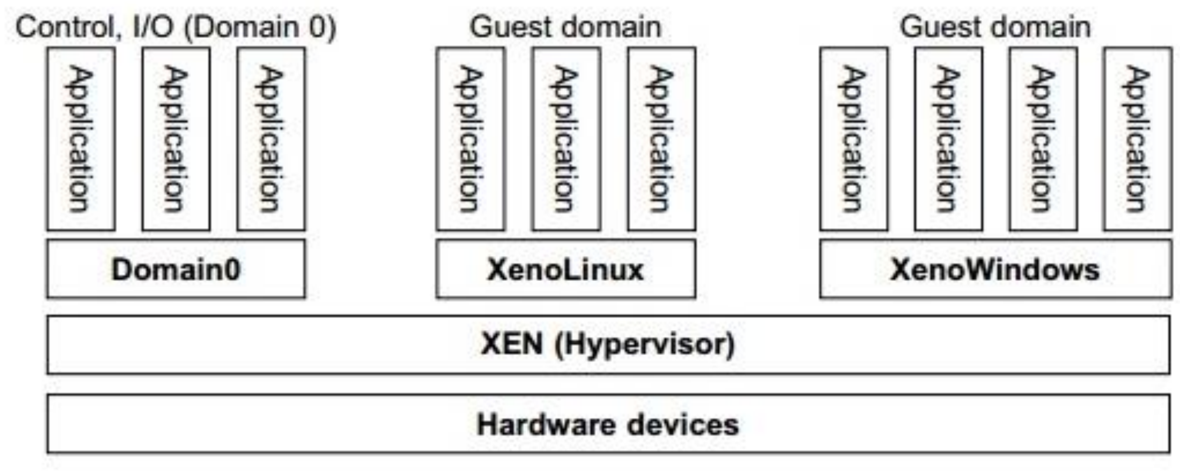
As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS.

***The core components of a Xen system are the hypervisor, kernel, and applications.***

## The Xen Architecture



# The Xen Architecture



The organization of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor. However, not all guest OSes are created equal, and one in particular controls the others.

The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen.

It is first loaded when Xen boots without any file system drivers being available. ***Domain 0 is designed to access hardware directly and manage devices.*** Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).

## Binary Translation with Full Virtualization

***Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.***

Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, nonvirtualizable instructions.

The guest OSes and their applications consist of noncritical and critical instructions.

In a host-based system, both a host OS and a guest OS are used. ***A virtualization software layer is built between the host OS and guest OS***

## Full Virtualization

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software

**Why are only critical instructions trapped into the VMM?**

This is because binary translation can incur a large performance overhead.

Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do.

Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

## Host-Based Virtualization

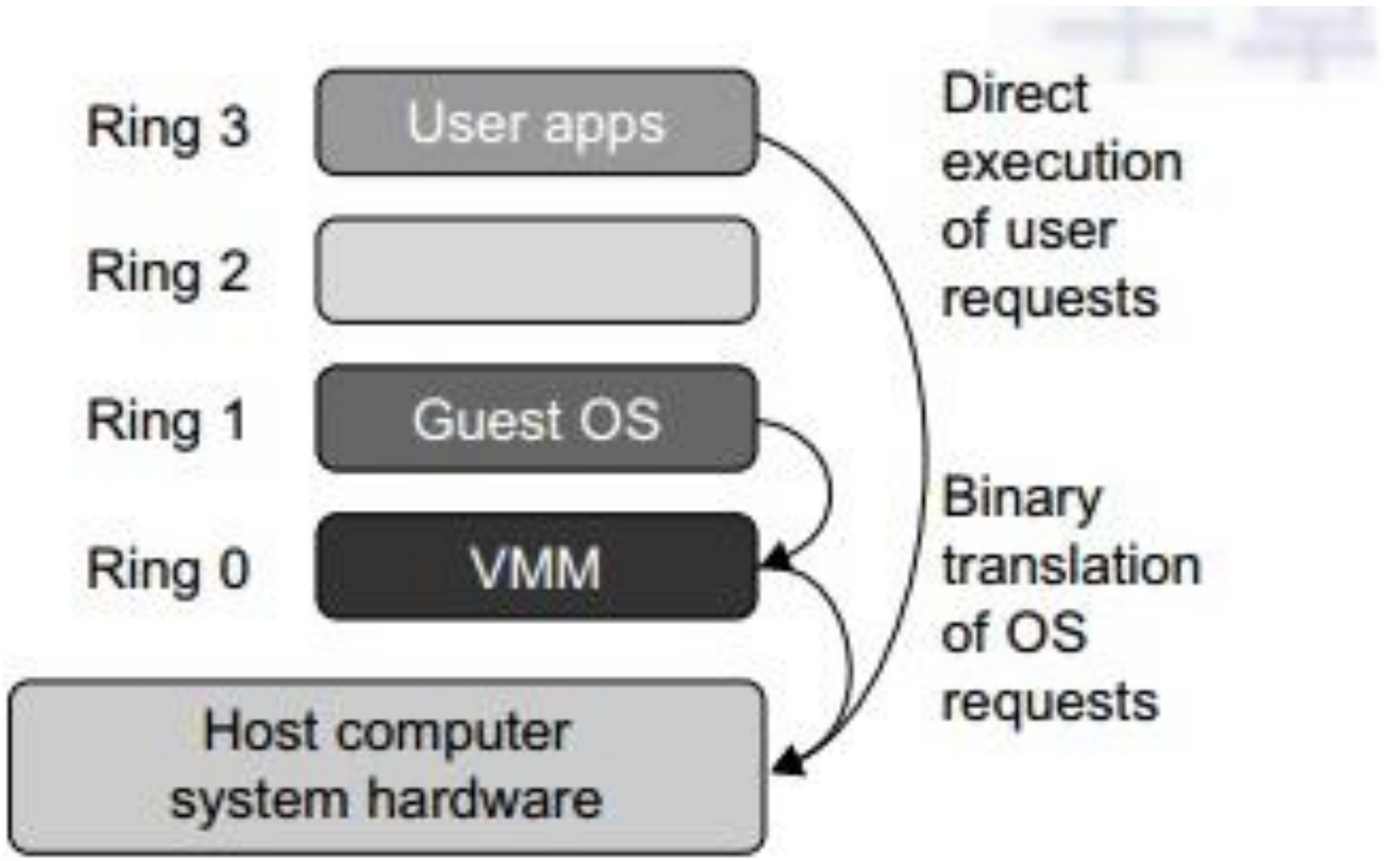
An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer.

First, the user can install this VM architecture without modifying the host OS.  The virtualizing software can rely on the host OS to provide device drivers and other low-level services.

This will simplify the VM design and ease its deployment.

Second, the host-based approach appeals to many host machine configurations.

Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low.


Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

## Para-Virtualization with Compiler Support

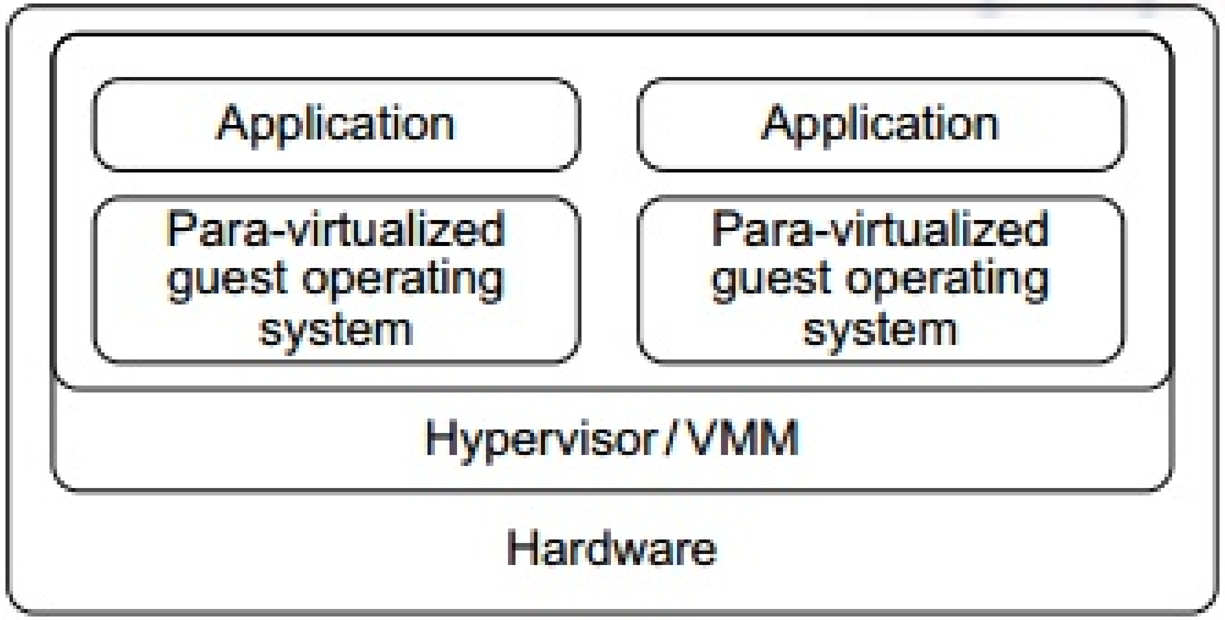
Para-virtualization needs to modify the guest operating systems.

A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Performance degradation is a critical issue of a virtualized system.

No one wants to use a VM if it is much slower than using a physical machine.

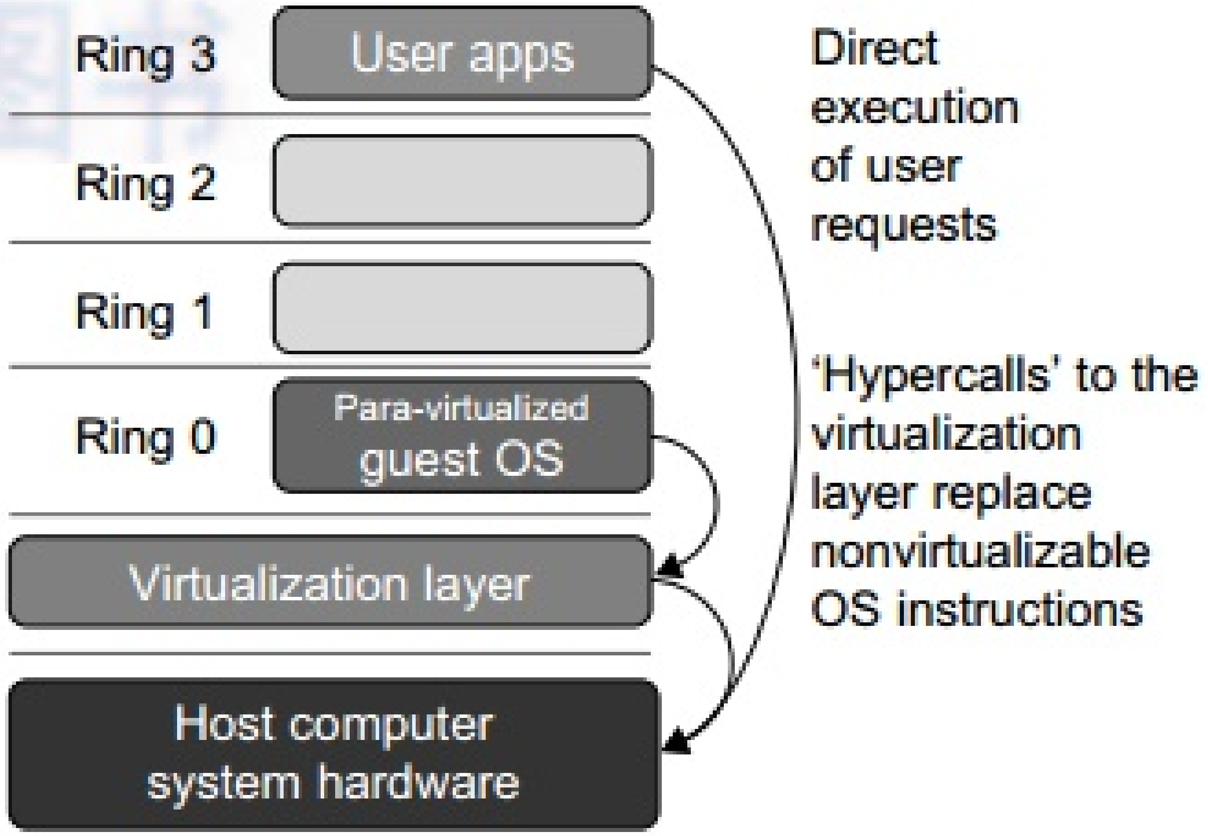
The virtualization layer can be inserted at different positions in a machine soft-ware stack.

However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.


Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process

Fig a.


Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

*(Courtesy of VMWare [71])*

Fig b.

## Para-Virtualization with Compiler Support

Figure a illustrates the concept of a paravirtualized VM architecture. The guest operating systems are para-virtualized.

They are assisted by an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls as illustrated in Figure b.

The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed.

The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3

## Para-Virtualization Architecture

When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS.

According to the x86 ring definition, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems.

In Figure b, the para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM.

However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

## Virtualization of CPU

CPU virtualization emphasizes performance and runs directly on the processor whenever possible. The underlying physical resources are used whenever possible and the virtualization layer runs instructions only as needed to make virtual machines operate as if they were running directly on a physical machine.

A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency.

The critical instructions are divided into three categories: privileged instructions, control-sensitive instructions, and behavior-sensitive instructions. Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode. Control-sensitive instructions attempt to change the configuration of resources used. Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

## Virtualization of CPU

A CPU architecture is virtualizable if it supports the ability to run the VM’s privileged and unprivileged instructions in the CPU’s user mode while the VMM runs in supervisor mode.

When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM. I

n this case, the VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system.

However, not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions.

On the contrary, x86 CPU architectures are not primarily designed to support virtualization.

## Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems.

In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.

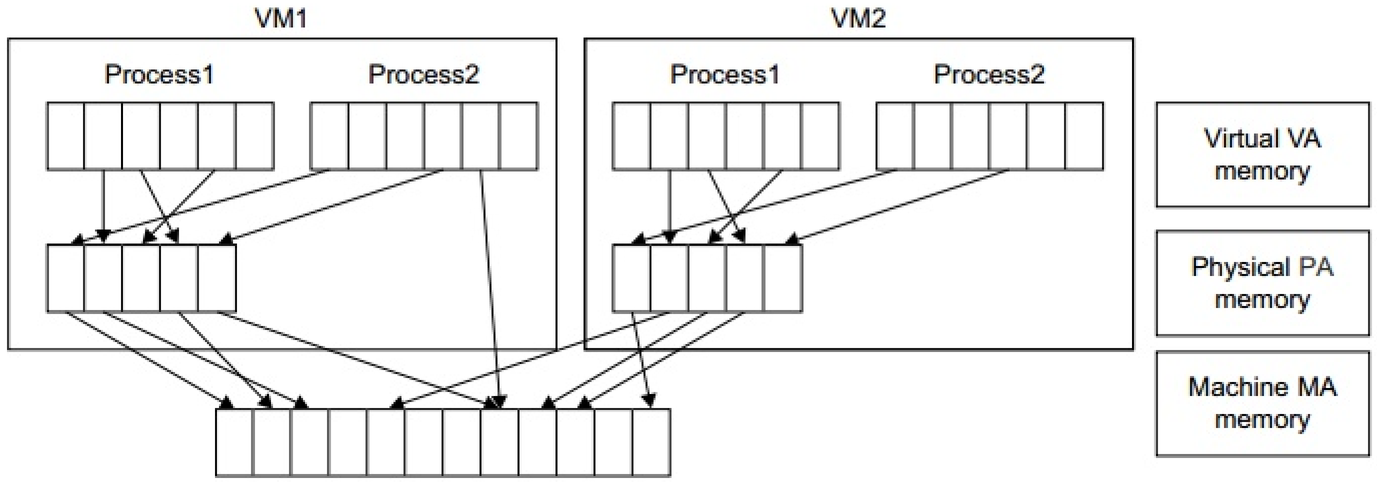
All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.

However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

## Memory Virtualization

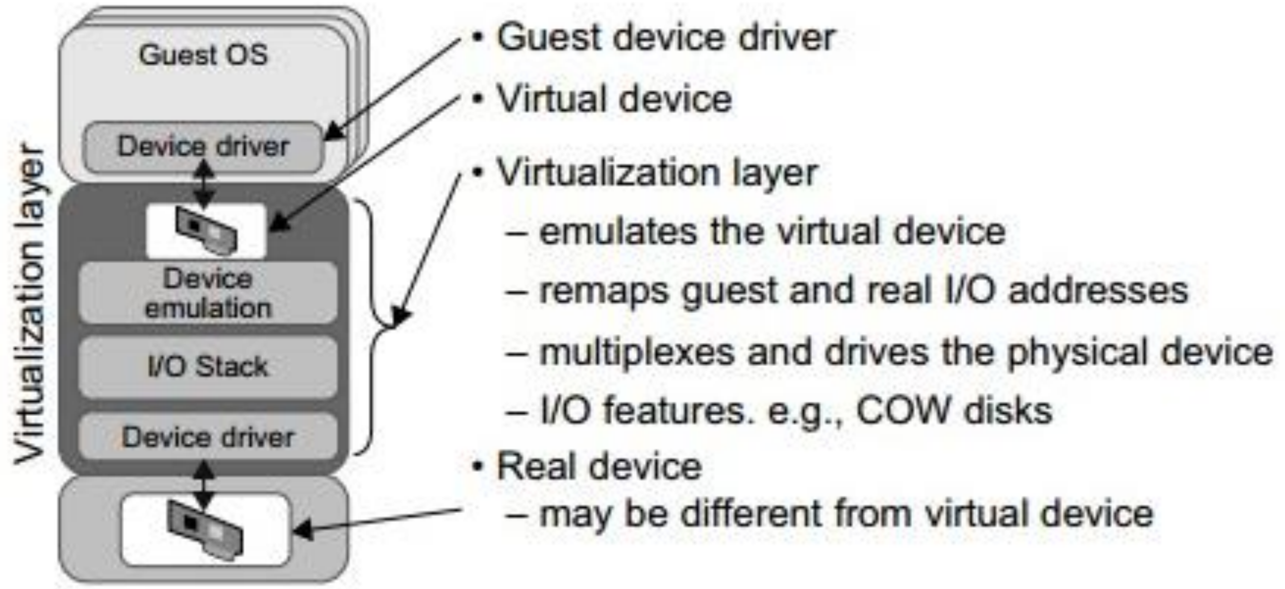
That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.

Furthermore, MMU virtualization should be supported, which is transparent to the guest OS. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory. The VMM is responsible for mapping the guest physical memory to the actual machine memory.



## I/O Virtualization

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. There are three ways to implement I/O virtualization: full device emulation, para-virtualization, and direct I/O. Full device emulation is the first approach for I/O virtualization.


Guest OS
Device driver
Device emulation
I/O Stack
Device driver
Virtualization layer
• Guest device driver
• Virtual device
• Virtualization layer
– emulates the virtual device
– remaps guest and real I/O addresses
– multiplexes and drives the physical device
– I/O features. e.g., COW disks
• Real device
– may be different from virtual device

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

## I/O Virtualization

A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates.

The para-virtualization method of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver.

The frontend driver is running in Domain U and the backend driver is running in Domain 0. They interact with each other via a block of shared memory.

The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs.

*Although para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.*

## I/O Virtualization

Direct I/O virtualization lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs.

However, current direct I/O virtualization implementations focus on networking for mainframes. There are a lot of challenges for commodity hardware devices.

For example, when a physical device is reclaimed (required by workload migration) for later reassignment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system.

Since software-based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical. Intel VT-d supports the remapping of I/O DMA transfers and device-generated interrupts. The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or “virtualization-aware” guest OSes.

## I/O Virtualization

Direct I/O virtualization lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs.

However, current direct I/O virtualization implementations focus on networking for mainframes. There are a lot of challenges for commodity hardware devices.

For example, when a physical device is reclaimed (required by workload migration) for later reassignment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system.

Since software-based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical. Intel VT-d supports the remapping of I/O DMA transfers and device-generated interrupts. The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or “virtualization-aware” guest OSes.

## VIRTUAL CLUSTERS

When a traditional VM is initialized, the administrator needs to manually write configuration information or specify the configuration sources.

When more VMs join a network, an inefficient configuration always causes problems with overloading or underutilization.

Amazon’s Elastic Compute Cloud (EC2) is a good example of a web service that provides elastic computing power in a cloud. EC2 permits customers to create VMs and to manage user accounts over the time of their use.

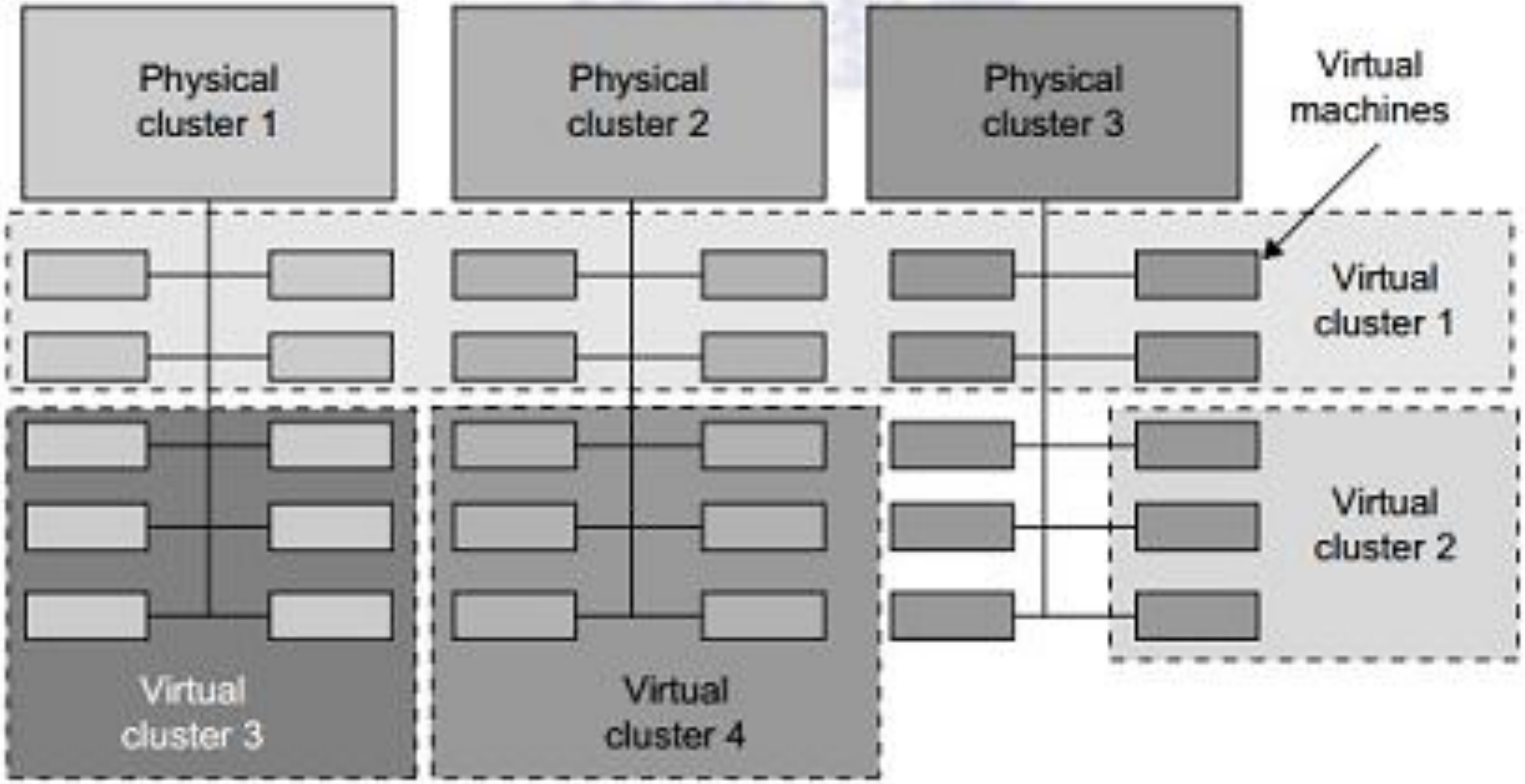
Most virtualization platforms, including XenServer and VMware ESX Server, support a bridging mode which allows all domains to appear on the network as individual hosts. By using this mode, VMs can communicate with one another freely through the virtual network interface card and configure the network automatically.

## Physical versus Virtual Clusters

Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters.

The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks.

Each virtual cluster is formed with physical machines or a VM hosted by multiple physical clusters. The virtual cluster boundaries are shown as distinct boundaries.



A cloud platform with four virtual clusters over three physical clusters shaded differently.

## Physical versus Virtual Clusters

The provisioning of VMs to a virtual cluster is done dynamically to have the following interesting properties:

- The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSes can be deployed on the same physical node.

- A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.

- The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance server utilization and application flexibility.

VMs can be colonized (replicated) in multiple servers for the purpose of promoting distributed parallelism, fault tolerance, and disaster recovery.

## RESOURCE MANAGEMENT

Scheduling in a computing system

□ deciding how to allocate resources of a system, such as CPU cycles, memory, secondary storage space, I/O and network bandwidth, between users and tasks.

□ Policies and mechanisms for resource allocation.

□ principles guiding decisions.

□ the means to implement policies.

## RESOURCE MANAGEMENT

**Cloud resource management**

 Requires complex policies and decisions for multi-objective optimization.

 It is challenging - the complexity of the system makes it impossible to have accurate global state information.

 Affected by unpredictable interactions with the environment, e.g., system failures, attacks.

 Cloud service providers are faced with large fluctuating loads which challenge the claim of cloud elasticity

## Cloud resource management (CRM) policies

1. Admission control  prevent the system from accepting workload in violation of high-level system policies.

2. Capacity allocation  allocate resources for individual activations of a service.

3. Load balancing  distribute the workload evenly among the servers.

4. Energy optimization  minimization of energy consumption.

5. Quality of service (QoS) guarantees  ability to satisfy timing or other conditions specified by a Service Level Agreement.

## Mechanisms for the implementation of resource management policies

- Control theory - uses the feedback to guarantee system stability and predict transient behavior.

- Machine learning - does not need a performance model of the system.

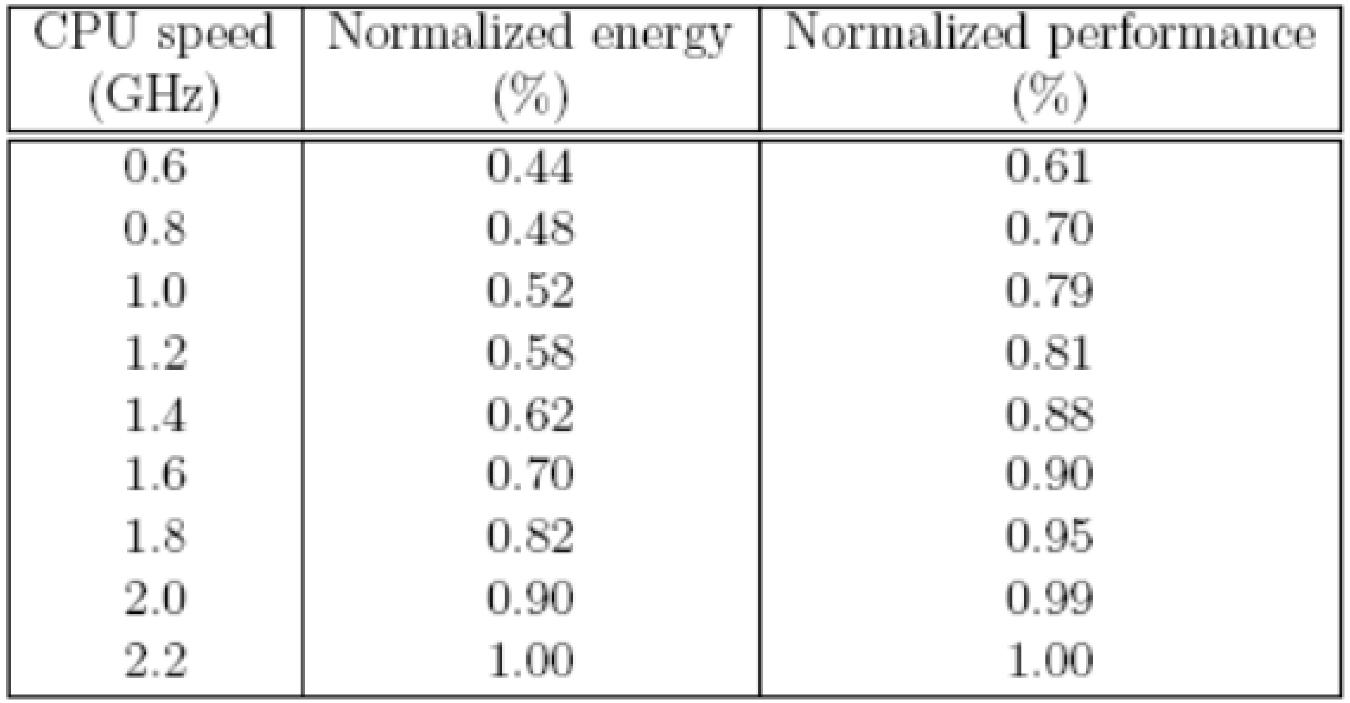
- Utility-based - require a performance model and a mechanism to correlate user-level performance with cost.

- Market-oriented/economic - do not require a model of the system, e.g., combinatorial auctions for bundles of resources.

## Tradeoffs

- To reduce cost and save energy we may need to concentrate the load on fewer servers rather than balance the load among them.

- We may also need to operate at a lower clock rate; the performance decreases at a lower rate than does the energy



| CPU speed (GHz) | Normalized energy (%) | Normalized performance (%) |
|---|---|---|
| 0.6 | 0.44 | 0.61 |
| 0.8 | 0.48 | 0.70 |
| 1.0 | 0.52 | 0.79 |
| 1.2 | 0.58 | 0.81 |
| 1.4 | 0.62 | 0.88 |
| 1.6 | 0.70 | 0.90 |
| 1.8 | 0.82 | 0.95 |
| 2.0 | 0.90 | 0.99 |
| 2.2 | 1.00 | 1.00 |

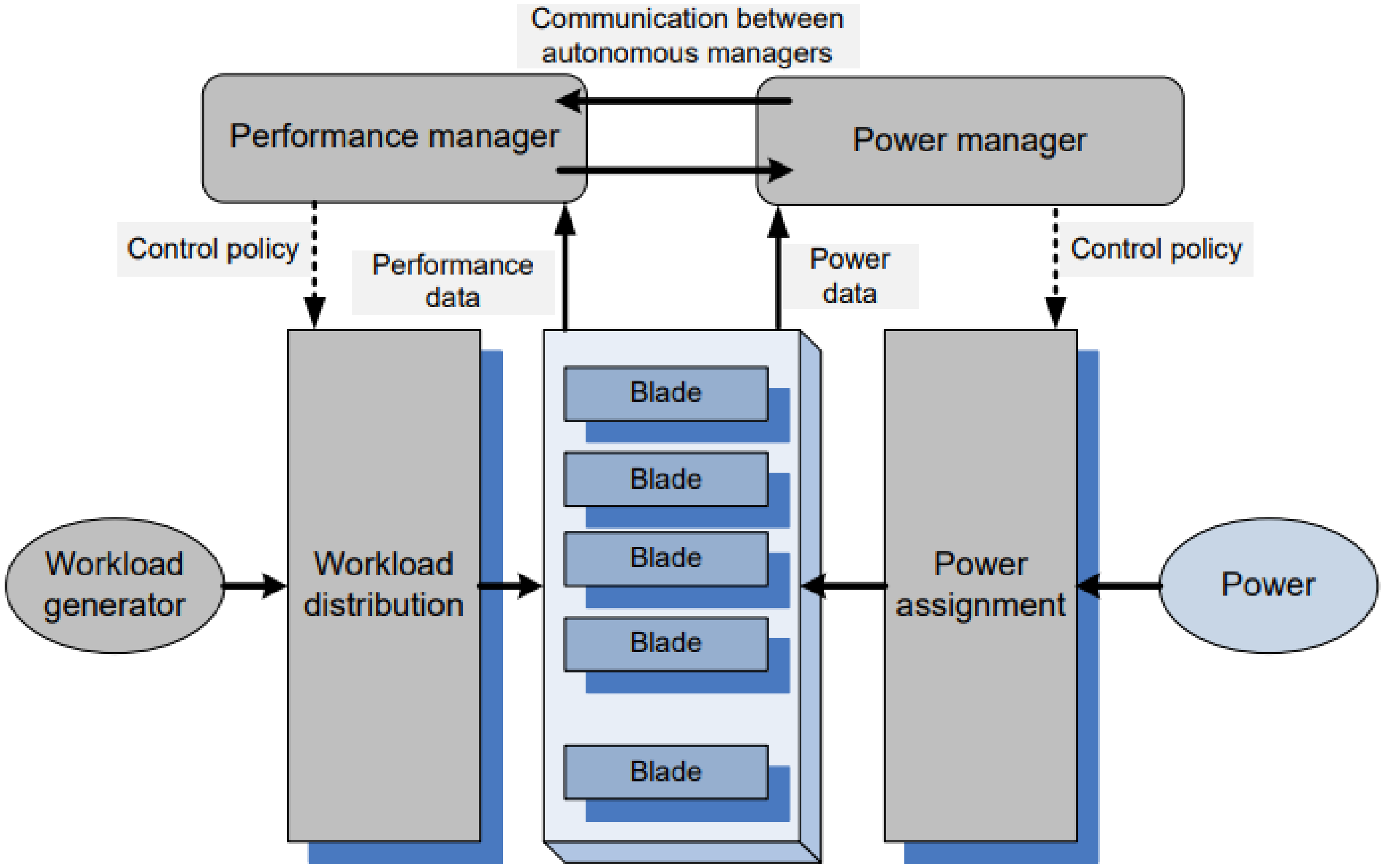
## Control theory application to cloud resource management (CRM)

The main components of a control system:

- The inputs - the offered workload and the policies for admission control, the capacity allocation, the load balancing, the energy optimization, and the QoS guarantees in the cloud.

- The control system components - sensors used to estimate relevant measures of performance and controllers which implement various policies.

- The outputs - the resource allocations to the individual applications



Autonomous performance and power managers cooperate to ensure prescribed performance and energy optimization; they are fed with performance and power data and implement the performance and power management policies

## Virtualization for Data-center automation

**Data Center**

A data center is a physical facility that organizations use to house their critical applications and data.

A data center's design is based on a network of computing and storage resources that enable the delivery of shared applications and data.

The key components of a data center design include routers, switches, firewalls, storage systems, servers, and application-delivery controllers.

***A data center is a physical room, building or facility that houses IT infrastructure for building, running, and delivering applications and services, and for storing and managing the data associated with those applications and services.***

## Virtualization for Data-center automation

Data center virtualization is the process of creating a modern data center that is highly **scalable, available and secure**.

With data center virtualization products you can increase **IT agility** and create a seamless foundation to manage private and public cloud services alongside traditional on-premises infrastructure.

Data-center automation means that huge volumes of hardware, software, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QoS and cost-effectiveness.

## Server Consolidation in Data Centers

In data centers, a large number of heterogeneous workloads can run on servers at various times.

These heterogeneous workloads can be roughly divided into two categories: ***chatty workloads and noninter-active workloads***.

Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day.

Noninteractive workloads do not require people's efforts to make progress after they are submitted. High-performance computing is a typical example of this. At various stages, the requirements for resources of these workloads are dramatically different. However, to guarantee that a workload will always be able to cope with all demand levels, the workload is statically allocated enough resources so that peak demand is satisfied.

## Server Consolidation in Data Centers

Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers.

Among several server consolidation techniques such as centralized and physical consolidation, virtualization-based server consolidation is the most powerful.

Data centers need to optimize their resource management. Yet these techniques are performed with the granularity of a full server machine, which makes resource management far from well optimized.

Server virtualization enables smaller resource allocation than a physical machine.

In detail, server consolidation has the following effects:

- Consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

- This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

- The total cost of ownership is reduced. In this sense, server virtualization causes deferred purchases of new servers, a smaller data-center footprint, lower maintenance costs, and lower power, cooling, and cabling requirements.

- This approach improves availability and business continuity. The crash of a guest OS has no effect on the host OS or any other guest OS. It becomes easier to transfer a VM from one server to another, because virtual servers are unaware of the underlying hardware.