Module 2

Virtualization

Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure-based services. Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are untrusted, without affecting other users' applications. The basis of this technology is the ability of a computer program—or a combination of software and hardware—to emulate an executing environment separate from the one that hosts such programs. For example, we can run Windows OS on top of a virtual machine, which itself is running on Linux OS. Virtualization provides a great opportunity to build elastically scalable systems that can provision additional capability with minimum costs.

Introduction

Virtualization is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether virtual hardware or an operating system—to run applications. The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering *Infrastructure-as-a-Service (IaaS)* solutions for cloud computing. In fact, virtualization technologies have a long trail in the history of computer science and have been available in many flavors by providing virtual environments at the operating system level, the programming language level, and the application level. Moreover, *virtualization technologies provide a virtual environment for not only executing applications but also for storage, memory, and networking.*

Virtualization technologies have gained renewed interested recently due to the merging of several phenomena:

•**Increased performance and computing capacity**. Nowadays, the average end-user desktop PC is powerful enough to meet almost all the needs of everyday computing, with extra capacity that is rarely used. Almost all these PCs have resources enough to host a *virtual machine manager* and execute a virtual machine with by far acceptable performance. The same consideration applies to the high-end side of the PC market, where supercomputers can provide immense compute power that can accommodate the execution of hundreds or thousands of virtual machines.

• **Underutilized hardware and software resources.** Hardware and software underutilization is occurring due to (1) increased performance and computing capacity, and (2) the effect of limited or sporadic use of resources. Computers today are so powerful that in most cases only a fraction of their capacity is used by an application or the system. Moreover, if we consider the IT infrastructure of an enterprise, many computers are only partially utilized .

For example, desktop PCs mostly devoted to office automation tasks and used by administrative staff are only used during work hours, remaining completely unused overnight. Using these resources for other purposes after hours could improve the efficiency of the IT infrastructure. To transparently provide such a service, it would be necessary to deploy a completely separate environment, which can be achieved through virtualization.

• **Lack of space**. The continuous need for additional capacity, whether storage or compute power, makes data centers grow quickly. Companies such as Google and Microsoft expand their infrastructures by building data centers as large as football fields that are able to host thousands of nodes. Although this is viable for IT giants, in most cases enterprises cannot afford to build another data center to accommodate additional resource capacity. This condition, along with hardware underutilization, has led to the diffusion of a technique called *server consolidation*, for which virtualization technologies are fundamental.

• **Greening initiatives.** Recently, companies are increasingly looking for ways to reduce the amount of energy they consume and to reduce their carbon footprint. Data centers are one of the major power consumers; they contribute consistently to the impact that a company has on the environment. Maintaining a data center operation not only involves keeping servers on, but a great deal of energy is also consumed in keeping them cool. Infrastructures for cooling have a significant impact on the carbon footprint of a data center. Hence, reducing the number of servers through server consolidation will definitely reduce the impact of cooling and power consumption of a data center. Virtualization technologies can provide an efficient way of consolidating servers.

• **Rise of administrative costs.** Power consumption and cooling costs have now become higher than the cost of IT equipment. Moreover, the increased demand for additional capacity, which translates into more servers in a data center, is also responsible for a significant increment in administrative costs. Computers—in particular, servers—do not operate all on their own, but they require care and feeding from system administrators. Common system administration tasks include hardware monitoring, defective hardware replacement, server setup and updates, server resources monitoring, and backups. These are

labor-intensive operations, and the higher the number of servers that have to be managed, the higher the administrative costs. Virtualization can help reduce the number of required servers for a given workload, thus reducing the cost of the administrative personnel.

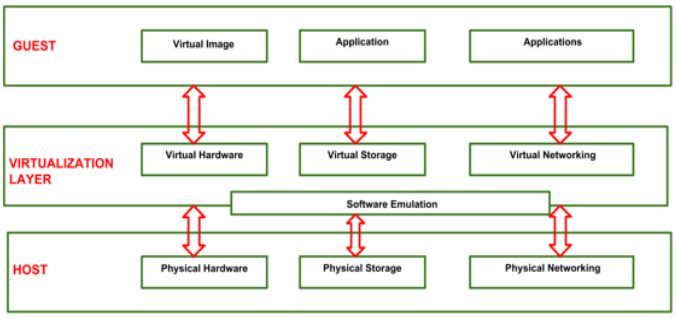
CHARACTERISTICS OF VIRTUALIZED ENVIRONMENTS

Virtualization is a broad concept and it refers to the creation of a virtual version of something, whether this is hardware, software environment , storage, or network. In a virtualized environment , there are three major components: *gust*, *host* and *virtualization layer*.

1. **Guest**: represents the system component that interact with the virtualization layer .

2. **Host**: represents the original environment where the guest is supposed to be managed.

3. **Virtualization layer** : is responsible for recreating the same or a different environment where the guest will operate.



The advantages that are characterized by virtualized solutions are :

1. **Increased security**

   - The ability to control the execution of a gust in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment.
   - Resources exposed by the host can then be hidden or simply protected from the guest. Moreover, sensitive information that is contained in the host can be naturally hidden without the need of installing complex security policies. Increased security is a requirement when dealing with untrusted code.

- Both the JVM and .NET runtime provide extensive security policies for customizing the execution environment of applications.
- *Hardware virtualization* solutions such as 'Vmware Desktop', 'VirtualBox' and 'Parallels' provide the ability to create a virtual computer with customized virtual hardware on top of which a new OS can be installed.
- By default the file system exposed by the virtual computer is completely separate from the one of the host machine. This becomes the perfect environment for running applications without affecting other users in the environment.

2. Managed execution.

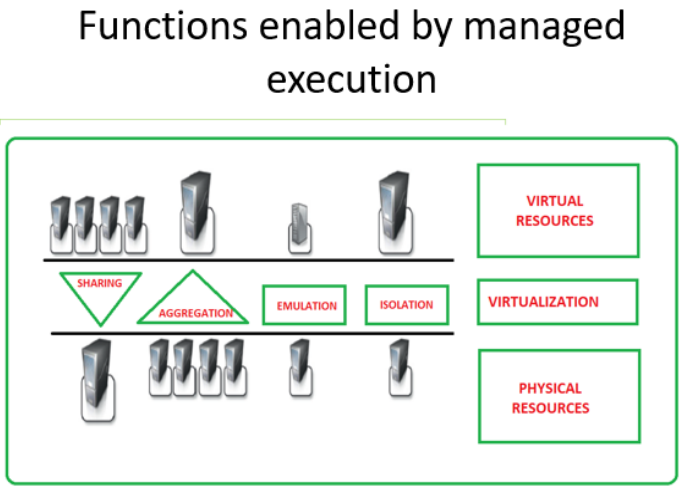
Virtualization of the execution environment allow increased security and provide a wide range of features:

i. *Sharing* : virtualization allows the creation of a separate computing environment within the same host. In this way , it is possible to fully exploit the capabilities of a powerful host , which would be otherwise underutilized.

ii. *Aggregation* : A group of separate hosts can be tied together and represented to gusts as a single virtual host. This function is naturally implemented in middleware for distributed computing .

iii. *Emulation* : A complete different environment with respect to the host can be emulated, allowing the execution of guests requiring specific characteristics that are not present in the physical host.

iv. *Isolation :* Virtualization allows providing guests- whether they are OS, applications or other entities- with a complete environment , in which they are executed.



3.Portability

Portability applies in different ways in different type of virtualization.

- In the case of *hardware virtualization* the guest is packaged into a virtual image that can be safely move and executed on top of different virtual machines.
- In case of *programming level virtualization* the binary code representing application components can be run without any recompilation on any implementation of the corresponding virtual machine.
- Portability allows having our own system always with us and ready to use.

Taxonomy/Classification of virtualization techniques

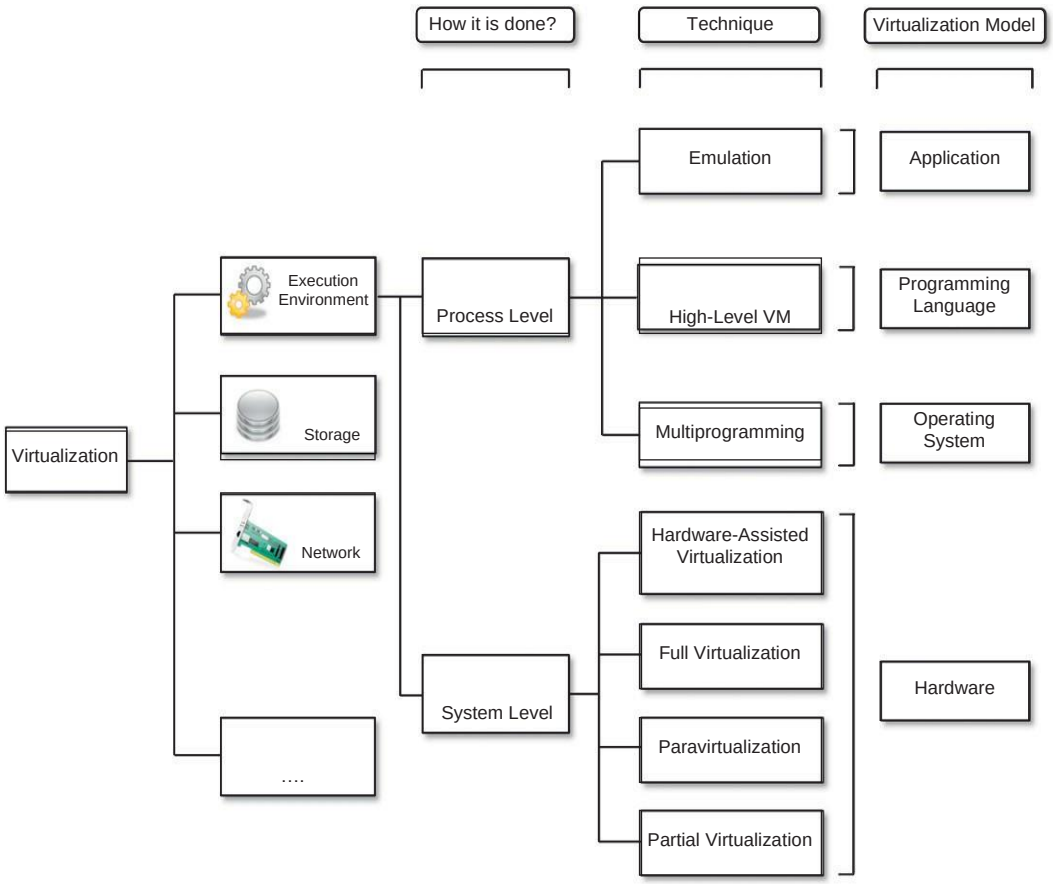
Virtualization covers a wide range of emulation techniques that are applied to different areas of computing. The first classification discriminates against the service or entity that is being emulated. Virtualization is mainly used to emulate execution environments, storage, and networks. Among these categories, execution virtualization constitutes the oldest, most popular, and most developed area. Therefore, it deserves major investigation and a further categorization.

In particular we can divide these execution virtualization techniques into two major categories by considering the type of host they require. Process-level techniques are implemented on top of an existing operating system, which has full control of the hardware. System-level techniques are implemented

directly on hardware and do not require—or require a minimum of support from—an existing operating system. Within these two categories we can list various techniques that offer the guest a different type of virtual computation environment: bare hardware, operating system resources, low-level programming language, and application libraries.

Execution Virtualization

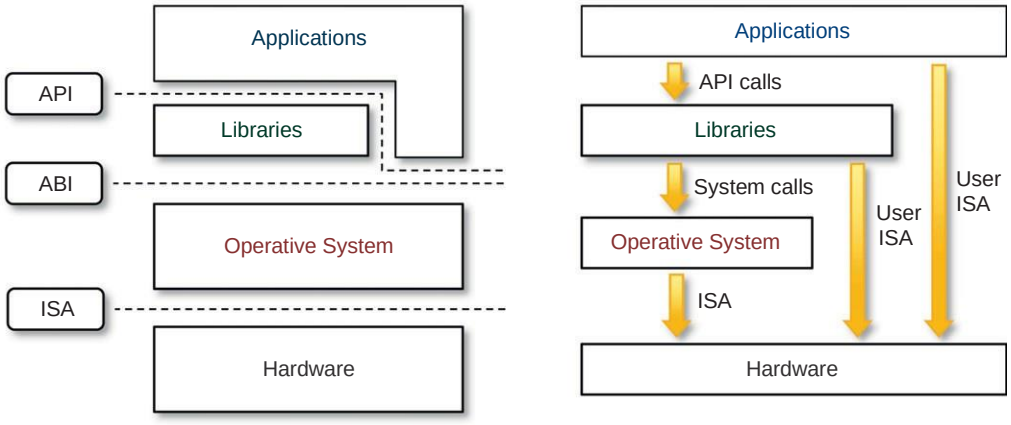
Execution virtualization includes all techniques that aim to emulate an execution environment that is separate from the one hosting the virtualization layer. All these techniques concentrate their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model, or an application. Therefore, execution virtualization can be implemented directly on top of the hardware by the operating system, an application, or libraries dynamically or statically linked to an application image.



A taxonomy of virtualization techniques.

1. MACHINE REFERENCE MODEL

Virtualizing an execution environment at different levels of the computing stack requires a reference model that defines the interfaces between the levels of abstractions, which hide implementation details. From this perspective, virtualization techniques actually replace one of the layers and intercept the calls that are directed toward it. Therefore, a clear separation between layers simplifies their implementation, which only requires the emulation of the interfaces and a proper interaction with the underlying layer.



A machine reference model.

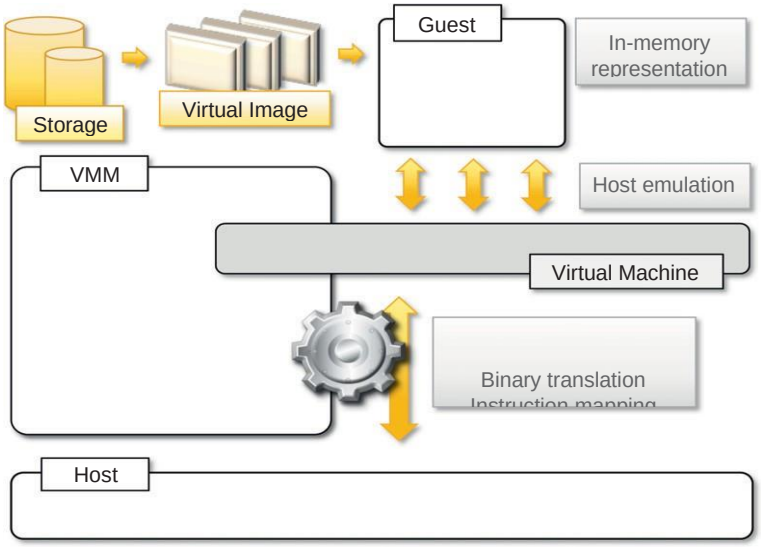
At the bottom layer, the model for the hardware is expressed in terms of the *Instruction Set Architecture (ISA), which defines the instruction set for the processor, registers, memory, and interrupt management*. ISA is the interface between hardware and software, and it is important to the operating system developer (*System ISA*) and developers of applications that directly manage the underlying hardware (*User ISA*). *The Application Binary Interface (ABI) separates the operating system layer from the applications and libraries, which are managed by the OS.* ABI covers details such as low-level data types, alignment, and call conventions and defines a format for executable programs. System calls are defined at this level. This interface allows portability of applications and libraries across operating systems that implement the same ABI. The highest level of abstraction is represented by the *Application Programming Interface (API),* which interfaces applications to libraries and/or the underlying operating system.

For any operation to be performed in the application level API, ABI and ISA are responsible for making it happen. The high-level abstraction is converted into machine-level instructions to perform the actual operations supported by the processor. The machine-level resources, such as processor, registers and main memory capacities, are used to perform the operation at the hardware level of the central processing unit (CPU). This layered approach simplifies the development and implementation of computing systems and simplifies the implementation of multitasking and the coexistence of multiple executing environments. In fact, such a model not only requires limited knowledge of the entire computing stack, but it also provides ways to implement a minimal security model for managing and accessing shared resources.

This reference model, it is possible to explore and better understand the various techniques utilized to virtualize execution environments and their relationships to the other components of the system.

2. HARDWARE-LEVEL VIRTUALIZATION

Hardware-level virtualization is a virtualization technique that provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can be run. In this model, the ***guest*** is represented by *the operating system*, the ***host*** by the *physical computer hardware*, the virtual machine by its emulation, and the virtual machine manager by the hypervisor. The ***hypervisor*** *is generally a program or a combination of software and hardware that allows the abstraction of the underlying physical hardware.*



A hardware virtualization reference model.

Hardware-level virtualization is also called *System Virtualization*, since it provides ISA to virtual machines, which is the representation of the hardware interface of a system.

❖ **Hypervisors**

A fundamental element of hardware virtualization is the hypervisor, or virtual machine manager (VMM). It recreates a hardware environment in which guest operating systems are installed. There are two major types of hypervisor: *Type I* and *Type II*.

Type I hypervisors:

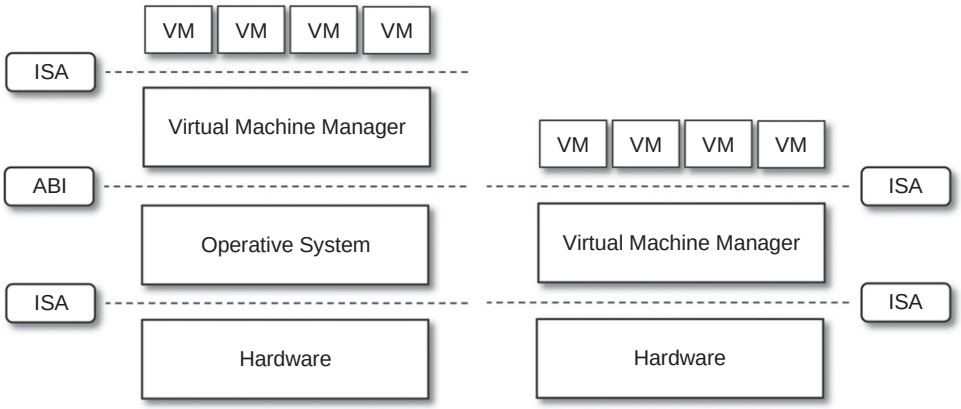
- It is run directly on top of the hardware.
- Therefore, they take the place of the operating systems and interact directly with the ISA interface

exposed by the underlying hardware, and they emulate this interface in order to allow the management of guest operating systems.

- It is also called a ***native virtual machine*** since it runs natively on hardware.

***Type II* hypervisors:**

- It require the support of an operating system to provide virtualization services.
- This means that they are programs managed by the operating system, which interact with it through the ABI and emulate the ISA of virtual hardware for guest operating systems.
- It is also called a ***hosted virtual machine*** since it is hosted within an operating system.

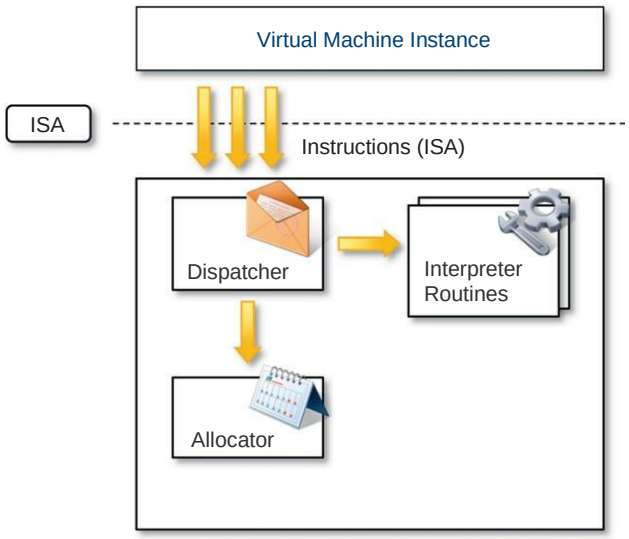


Hosted (left) and native (right) virtual machines.

Three main modules coordinate their activity in order to emulate the underlying hardware:

1. Dispatcher : The dispatcher constitutes the entry point of the *monitor and re-routes the instructions* issued by the virtual machine instance to one of the two other modules.
2. Allocator : The allocator is responsible for deciding the system resources to be provided to the VM: whenever a virtual machine tries to execute an instruction that results in *changing the machine resources* associated with that VM, the *allocator is invoked by the dispatcher.*
3. Interpreter : This module consists of interpreter routines. These are executed whenever a virtual machine *executes a privileged instruction*: a trap is triggered and the corresponding routine is executed.

The design and architecture of a ***Virtual Machine Manager***, together with the underlying hardware design of the host machine, determine the full realization of hardware virtualization, where a guest operating system can be transparently executed on top of a VMM as though it were run on the underlying hardware.



A hypervisor reference architecture.

This reference model represents what we generally consider classic virtualization—that is, the ability to execute a guest operating system in complete isolation. To a greater extent, hardware-level virtualization includes several strategies that differentiate from each other in terms of which kind of support is expected from the underlying hardware, what is actually abstracted from the host, and whether the guest should be modified or not.

3. HARDWARE VIRTUALIZATION TECHNIQUES

a. Hardware-assisted virtualization.

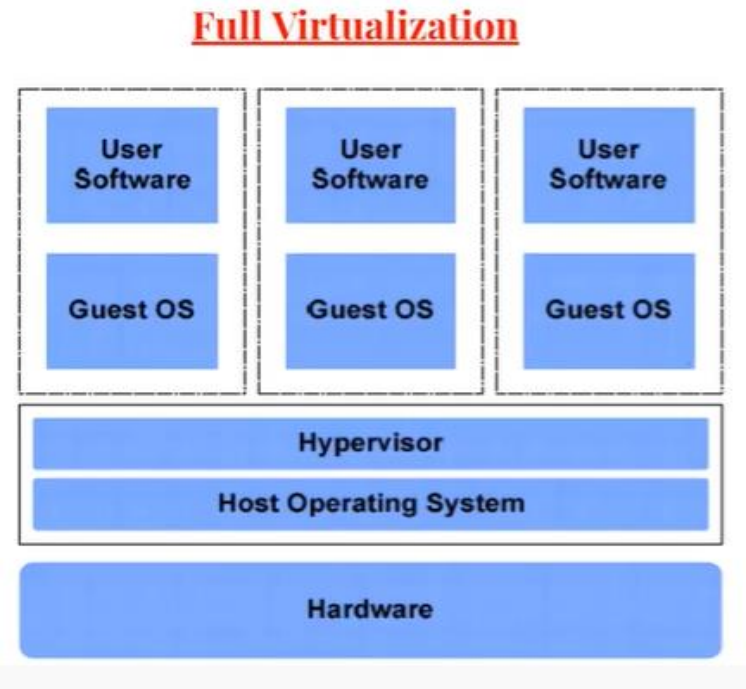
- This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation.

b. Full virtualization.

- *Full virtualization* refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw

hardware. To make this possible, virtual machine managers are required to provide a complete emulation of the entire underlying hardware.

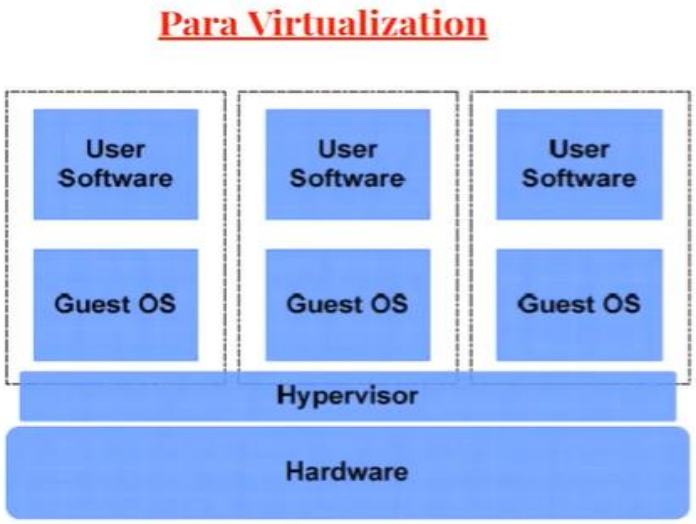
- The advantage of full virtualization is *complete isolation*, which leads to *enhanced security, ease of emulation of different architectures*, and *coexistence of different systems* on the same platform.
- A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host.



c. Paravirtualization.

- It expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.
- The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution.
- This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host.
- To take advantage of such an opportunity, guest operating systems need to be modified and explicitly ported by remapping the performance-critical operations through the virtual machine software interface.

- This is possible when the source code of the operating system is available, and this is the reason that paravirtualization was mostly explored in the open-source and academic environment.



FULL VIRTUALIZATION	PARA VIRTUALIZATION
A common and cost effective type of virtualization in which computer service requests are separated from the physical hardware that facilitate them.	An enhancement of virtualization technology in which a guest OS is recompiled prior to installation inside a virtual machine.
Allows guest OS to work independently.	Allows guest OS to communicate with the hypervisor.
Guest OS issues hardware calls to access hardware.	Guest OS directly communicate with the hypervisor using drivers.

d. Partial virtualization.

- It provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation.
- *It allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization.*
- An example of partial virtualization is *address space virtualization* used in *time-sharing systems;* this allows multiple applications and users to run concurrently in a separate memory space, but they still share the same hardware resources (disk, processor, and network).

- Historically, partial virtualization has been an important milestone for achieving full virtualization
- Address space virtualization is a common feature of contemporary operating systems.

4. OPERATING SYSTEM-LEVEL VIRTUALIZATION

Operating system-level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently. Differently from hardware virtualization, *there is no virtual machine manager or hypervisor, and the virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances.* The *kernel* is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other. A user space instance in general contains a proper view of the file system, which is completely isolated, and separate IP addresses, software configurations, and access to devices.

This virtualization technique can be considered an evolution of the *chroot mechanism* in *Unix systems*. The *chroot* operation changes the file system root directory for a process and its children to a specific directory. As a result, the process and its children cannot have access to other portions of the file system than those accessible under the new root directory. Because Unix systems also expose devices as parts of the file system, by using this method it is possible to completely isolate a set of processes. Following the same principle, *operating system-level virtualization aims to provide separated and multiple execution containers for running applications.* Compared to hardware virtualization, this strategy imposes little or no overhead because *applications directly use OS system calls and there is no need for emulation.* There is no need to modify applications to run them, nor to modify any specific hardware, as in the case of hardware-assisted virtualization. On the other hand, operating system-level virtualization does not expose the same flexibility of hardware virtualization, since all the user space instances must share the same operating system.

This technique is an efficient solution for *server consolidation* scenarios in which multiple application servers share the same technology: operating system, application server framework, and other components. When different servers are aggregated into one physical server, each server is run in a different user space, completely isolated from the others.

Examples of operating system-level virtualizations are FreeBSD Jails, IBM Logical Partition (LPAR), SolarisZones and Containers, Parallels Virtuozzo Containers, OpenVZ, iCore Virtual Accounts, Free Virtual Private Server (FreeVPS), and others.

5. PROGRAMMING LANGUAGE-LEVEL VIRTUALIZATION

It is mostly used to achieve ease of deployment of applications, managed execution, and portability

across different platforms and operating systems. It consists of a virtual machine executing the *byte code* of a program, which is the result of the compilation process. Compilers implemented and used this technology to produce a binary format representing the machine code for an abstract architecture. The characteristics of this architecture vary from implementation to implementation. Generally these virtual machines constitute a simplification of the underlying hardware instruction set and provide some high-level instructions that map some of the features of the languages compiled for them.

The main advantage of programming-level virtual machines, also called *process virtual machines*, is the ability to provide a uniform execution environment across different platforms. Programs compiled into byte code can be executed on any operating system and platform for which a virtual machine able to execute that code has been provided. From a development life-cycle point of view, this simplifies the development and deployment efforts since it is not necessary to provide different versions of the same code. The implementation of the virtual machine for different platforms is still a costly task, but it is done once and not for any application. Moreover, process virtual machines allow for more control over the execution of programs since they do not provide direct access to the memory. Security is another advantage of managed programming languages; by filtering the I/O operations, the process virtual machine can easily support sandboxing of applications.

Example : java platform – java virtual machine (jvm)

.Net provides common language infrastructure.

They are stack based virtual machines.

Implementations of this model are also called *high-level virtual machines*, since high-level programming languages are compiled to a conceptual ISA, which is further interpreted or dynamically translated against the specific instruction of the hosting platform.

6. APPLICATION-LEVEL VIRTUALIZATION

Application-level virtualization is a technique allowing applications to be run in runtime environments that do not natively support all the features required by such applications. In this scenario, applications are not installed in the expected runtime environment but are run as though they were. In general, these techniques are mostly concerned with partial file systems, libraries, and operating system component emulation. Such emulation is performed by a thin layer—a program or an operating system component—that is in charge of executing the application. Emulation can also be used to execute

program binaries compiled for different hardware architectures. In this case, one of the following strategies can be implemented:

Interpretation. In this technique every source instruction is interpreted by an emulator for executing native ISA instructions, leading to poor performance. Interpretation has a minimal startup cost but a huge overhead, since each instruction is emulated.

Binary translation. In this technique every source instruction is converted to native instructions with equivalent functions. After a block of instructions is translated, it is cached and reused. Binary translation has a large initial overhead cost, but over time it is subject to better performance, since previously translated instruction blocks are directly executed.

Emulation, as described, is different from hardware-level virtualization. The former simply allows the execution of a program compiled against a different hardware, whereas the latter emulates a complete hardware environment where an entire operating system can be installed.

Application virtualization is a good solution in the case of missing libraries in the host operating system; in this case a replacement library can be linked with the application, or library calls can be remapped to existing functions available in the host system. Another advantage is that in this case the virtual machine manager is much lighter since it provides a partial emulation of the runtime environment compared to hardware virtualization. Moreover, this technique allows incompatible applications to run together. Compared to programming-level virtualization, which works across all the applications developed for that virtual machine, application-level virtualization works for a specific environment: It supports all the applications that run on top of a specific environment.

Example:- *Wine*, which is a software application allowing Unix-like operating systems to execute programs written for the Microsoft Windows platform.

7. OTHER TYPES OF VIRTUALIZATION

Other than execution virtualization, other types of virtualization provide an abstract environment to interact with. These mainly cover storage, networking, and client/server interaction.

a.Storage Virtualization

It is a system administration practice that allows decoupling the physical orga- nization of the hardware from its logical representation. Using this technique, users do not have to be worried about the specific location of their data, which can be identified using a logical path.

Storage virtualization allows us to harness a wide range of storage facilities and represent them under a single logical file system. There are different techniques for storage virtualization, one of the most popular being network-based virtualization by means of *storage area networks (SANs)*. SANs use a network-accessible device through a large bandwidth connection to provide storage facilities.

b. Network virtualization

It combines hardware appliances and specific software for the creation and management of a virtual network. Network virtualization can aggregate different physical networks into a single logical network (*external* network virtualization) or provide network-like functionality to an operating system partition (*internal* network virtualization). The result of external network virtualization is generally a *virtual LAN (VLAN)*. A VLAN is an aggregation of hosts that communicate with each other as though they were located under the same broadcasting domain.

Internal network virtualization is generally applied together with hardware and operating system-level virtualization, in which the guests obtain a virtual network interface to communicate with. There are several options for implementing internal network virtualization: The guest can share the same net- work interface of the host and use Network Address Translation (NAT) to access the network; the virtual machine manager can emulate, and install on the host, an additional network device, together with the driver; or the guest can have a private network only with the guest.

c. Desktop virtualization

It abstracts the desktop environment available on a personal computer in order to provide access to it using a client/server approach. Desktop virtualization provides the same out-come of hardware virtualization but serves a different purpose. Similarly to hardware virtualization, desktop virtualization makes accessible a different system as though it were natively installed on the host, but this system is remotely stored on a different host and accessed through a network connection. Moreover, desktop virtualization addresses the problem of making the same desktop environment accessible from everywhere. Although the term *desktop virtualization* strictly refers to the ability to remotely access a desktop environment, generally the desktop environment is stored in a remote server or a data center that provides a high-availability infrastructure and ensures the accessibility and persistence of the data.

In this scenario, an infrastructure supporting hardware virtualization is fundamental to provide access to multiple desktop environments hosted on the same server; a specific desktop environment is stored in a virtual machine image that is loaded and started on demand when a client connects to the desktop environment. This is a typical cloud computing scenario in which the user leverages the virtual

infrastructure for performing the daily tasks on his computer. The advantages of desktop virtualization are high availability, persistence, accessibility, and ease of management.

The basic services for remotely accessing a desktop environment are implemented in software components such as Windows Remote Services, VNC, and X Server. Infrastructures for desktop virtualization based on cloud computing solutions include Sun Virtual Desktop Infrastructure (VDI), Parallels Virtual Desktop Infrastructure (VDI), Citrix XenDesktop, and others.

d. Application server virtualization

It abstracts a collection of application servers that provide the same services as a single virtual application server by using load-balancing strategies and providing a high-availability infrastructure for the services hosted in the application server. This is a particular form of virtualization and serves the same purpose of storage virtualization: providing a better quality of service rather than emulating a different environment.

Virtualization and cloud computing

- Virtualization plays an important role in cloud computing , it allows for the appropriate degree of customization , security, isolation, and manageability that are fundamental for delivering IT services on demand.
- Virtualization technologies are primarily used to offer configurable computing environments and storage.
- Hardware and programming language virtualization are the techniques adopted in cloud computing.
- Hardware virtualization is an enabling factor for solutions in **IaaS.**
- Programming language virtualization leveraged in **PaaS.**
- Virtualization simplify the leasing of services and their accountability on the vendor side.

Server consolidation and virtual machine migration are principally used in the case of hardware virtualization.

Server consolidation

- It allows reducing the number of active resources by aggregating virtual machines over smaller number of resources that become fully utilized.
- It allows reducing the power consumption of a data center and resolving hardware underutilization.

Live migration

- Live Migration is the process of moving a running virtual machine (VM) from one physical server to another while not disrupting the availability of the virtual machine to users.
- It helps better resource utilization by load balancing.

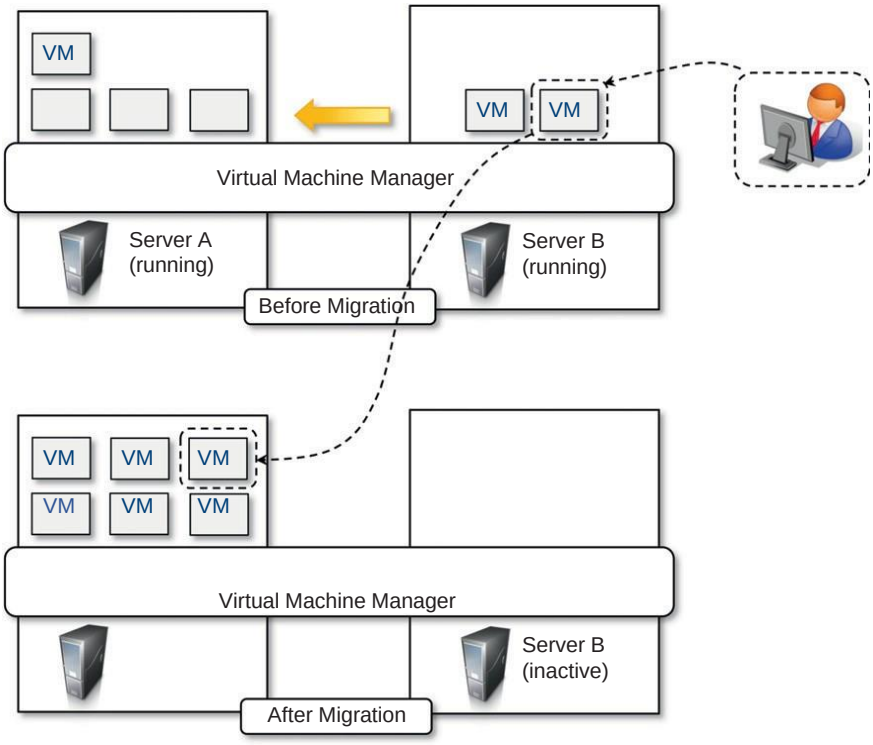


Fig:- Live migration and server consolidation.

- It is important to notice that cloud computing is strongly leveraged for the development of applications that need to scale on demand.
- In most cases, this is because applications have to process increased workloads or serve more requests, which makes them server applications.

- In this scenario, it is evident that live migration offers a better solution because it does not create any service interruption during consolidation.

Pros and Cons of Virtualization

Virtualization has now become extremely popular and widely used, especially in cloud computing. The primary reason for its wide success is the elimination of technology barriers that prevented virtualization from being an effective and viable solution in the past. The most relevant barrier has been performance. Today, the capillary diffusion of the Internet connection and the advancements in computing technology have made virtualization an interesting opportunity to deliver on-demand IT infrastructure and services. Despite its renewed popularity, this technology has benefits and also drawbacks.

Advantages of Virtualization:

1. ***Managed execution and isolation****:* These characteristics allow building secure and controllable computing environments. This enables fine tuning of resources , which is very important in server consolidation scenario.
2. ***Portability*** : Virtual machine instances are normally represented by one or more files that can be easily transported with respect to physical systems. Java programs are "compiled once and run everywhere" –they only require the java virtual machine to be installed on the host.
3. ***Self-containment*** : Contribute to reduce the costs for maintenance , since the number of hosts is expected to be lower than the number of virtual machine instances.
4. ***Efficient use of resources***: Multiple systems can securely coexist and share the resources of the underlying host, without interfering with each other. Thus save energy consumption and have less impact on the environment.

Disadvantages of Virtualization:

1. Performance degradation :

Performance is definitely one of the major concerns in using virtualization technology. Since virtualization interposes an abstraction layer between the guest and the host, the *guest can experience increased latencies.*

For instance, in the case of hardware virtualization, where the intermediate emulates a bare

machine on top of which an entire system can be installed, the causes of performance degradation can be traced back to the overhead introduced by the following activities:

- Maintaining the status of virtual processors
- Support of privileged instructions (trap and simulate privileged instructions)
- Support of paging within VM
- Console functions

Furthermore, when hardware virtualization is realized through a program that is installed or executed on top of the host operating systems, a major source of performance degradation is represented by the fact that the virtual machine manager is executed and scheduled together with other applications, thus sharing with them the resources of the host.

Similar consideration can be made in the case of virtualization technologies at higher levels, such as in the case of programming language virtual machines (Java, .NET, and others). *Binary translation and interpretation can slow down the execution of managed applications.*

2. Inefficiency and degraded user experience

Virtualization can sometime lead to an inefficient use of the host. In particular, some of the specific features of the host cannot be exposed by the abstraction layer and then become inaccessible. In the case of hardware virtualization, this could happen for *device drivers*. In the case of programming-level virtual machines, some of the features of the underlying operating systems may become inaccessible unless specific libraries are used.

3. Security holes and new threats

Virtualization opens the door to a new and unexpected form of *phishing.* The capability of emulating a host in a completely transparent manner led the way to malicious programs that are designed to extract sensitive information from the guest.

In the case of hardware virtualization, malicious programs can preload themselves before the operating system and act as a thin virtual machine manager toward it. The operating system is then controlled and can be manipulated to extract sensitive information of interest to third parties.

The same considerations can be made for programming-level virtual machines: Modified versions of the runtime environment can access sensitive information or monitor the memory

locations utilized by guest applications while these are executed. To make this possible, the original version of the runtime environment needs to be replaced by the modified one, which can generally happen if the malware is run within an administrative context or a security hole of the host operating system is exploited.

Technology examples

A wide range of virtualization technology is available especially for virtualizing computing environments.

1. Xen: Paravirtualization

- Xen is an open-source initiative implementing a virtualization platform based on paravirtualization.
- Initially developed by a group of researchers at the University of Cambridge in the UK.
- Xen-based technology is used for either *desktop virtualization* or *server virtualization*, and recently it has also been used to provide cloud computing solutions by means of Xen Cloud Platform (XCP).
- At the basis of all these solutions is the *Xen Hypervisor*, which constitutes the core technology of Xen.
- Recently Xen has been advanced to support full virtualization using hardware-assisted virtualization.
- Xen is the most popular implementation of *paravirtualization*, which, in contrast with full virtualization, allows high-performance execution of guest operating systems.
- A Xen-based system is managed by the *Xen hypervisor*, which runs in the highest privileged mode and controls the access of guest operating system to the underlying hardware.

2. VMware: Full Virtualization

- VMware's technology is based on the concept of *full virtualization*, where the underlying hardware is replicated and made available to the guest operating system, which runs unaware of such abstraction layers and does not need to be modified.

- VMware implements full virtualization either in the desktop environment, by means of *Type II* hypervisors, or in the server environment, by means of *Type I* hypervisors.
- In both cases, full virtualization is made possible by means of *direct execution* (for nonsensitive instructions) and *binary translation* (for sensitive instructions), thus allowing the virtualization of architecture such as x86.
- Besides these two core solutions, VMware provides additional tools and software that simplify the use of virtualization technology either in a desktop environment, with tools enhancing the integration of virtual guests with the host, or in a server environment, with solutions for building and managing virtual computing infrastructures.
- VMware also provides full virtualization of I/O devices such as network controllers and other peripherals such as keyboard, mouse, disks, and universal serial bus (USB) controllers.

3. Microsoft Hyper-V

- Hyper-V is an infrastructure virtualization solution developed by Microsoft for server virtualization.
- It uses a hypervisor-based approach to hardware virtualization, which leverages several techniques to support a variety of guest operating systems.
- Hyper-V supports multiple and concurrent execution of guest operating systems by means of *partitions*. A partition is a completely isolated environment in which an operating system is installed and run.