MODULE II

VIRTUALIZATION

- **VIRTUALIZATION**
  - **INTRODUCTION**
  - **VIRTUALIZATION AND CLOUD COMPUTING**
  - **Pros and Cons of Virtualization**
  - **Technology Examples**

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.

1.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 confluence 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 71 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 whereas they could be used without interruption on a 24/7/365 basis.

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.

- **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

- **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.

2.Characteristics of virtualized environments

Virtualization is a broad concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network.

In a virtualized environment there are three major components:

Guest

The guest represents the system component that interacts with the virtualization layer rather than with the host, as would normally happen.

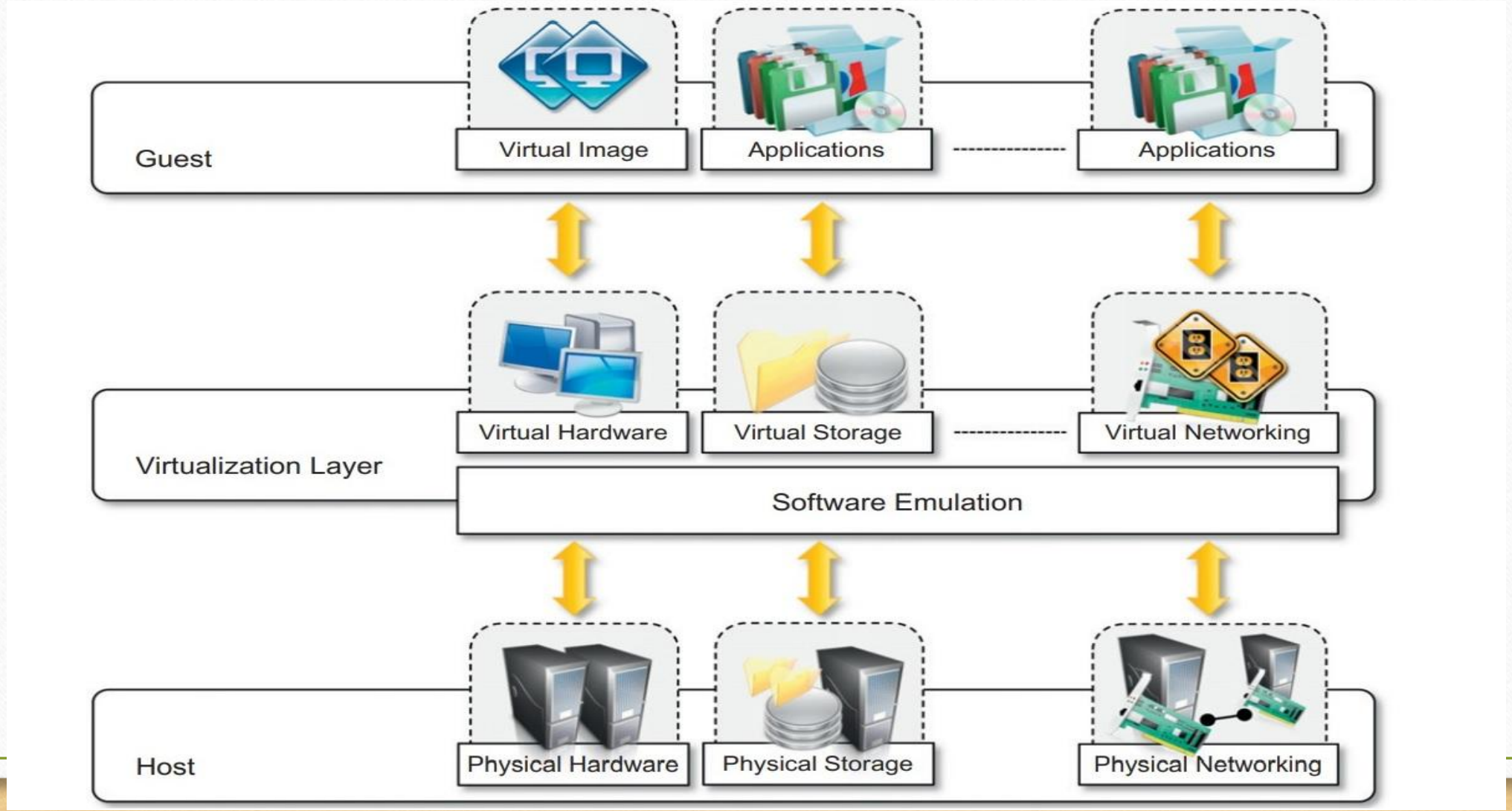
Host

The host represents the original environment where the guest is supposed to be managed.

Virtualization layer

The virtualization layer is responsible for recreating the same or a different environment where the guest will operate.

Virtualization Reference Model



- The most intuitive and popular is represented by hardware virtualization, which also constitutes the original realization of the virtualization concept.
- In the case of hardware virtualization, the guest is represented by a system image comprising an operating system and installed applications.
- These are installed on top of virtual hardware that is controlled and managed by the virtualization layer, also called the virtual machine manager.
- The host is instead represented by the physical hardware, and in some cases the operating system, that defines the environment where the virtual machine manager is running.
- In the case of virtual storage, the guest might be client applications or users that interact with the virtual storage management software deployed on top of the real storage system.
- The case of virtual networking is also similar: The guest— applications and users—interacts with a virtual network, such as a virtual private network (VPN), which is managed by specific software (VPN client) using the physical network available on the node.
- VPNs are useful for creating the illusion of being within a different physical network and thus accessing the resources in it, which would otherwise not be available.

- The main common characteristic of all these different implementations is the fact that the virtual environment is created by means of a **software program**.
- Characteristics of virtualized solutions.
  1. Increased Security
  2. Managed Execution
  3. Portability

Increased Security

- The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment.

- The virtual machine represents an emulated environment in which the guest is executed.
- All the operations of the guest are generally performed against the virtual machine, which then translates and applies them to the host.
- This level of indirection allows the virtual machine manager to control and filter the activity of the guest, thus preventing some harmful operations from being performed. 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 to install complex security policies.
- Increased security is a requirement when dealing with untrusted code.
- For example, applets downloaded from the Internet run in a sandboxed3 version of the Java Virtual Machine (JVM), which provides them with limited access to the hosting operating system resources.

Managed Execution

- Virtualization of the execution environment not only allows increased security, but a wider range of features also can be implemented.
- In particular, **sharing, aggregation, emulation, and isolation** are the most relevant features.

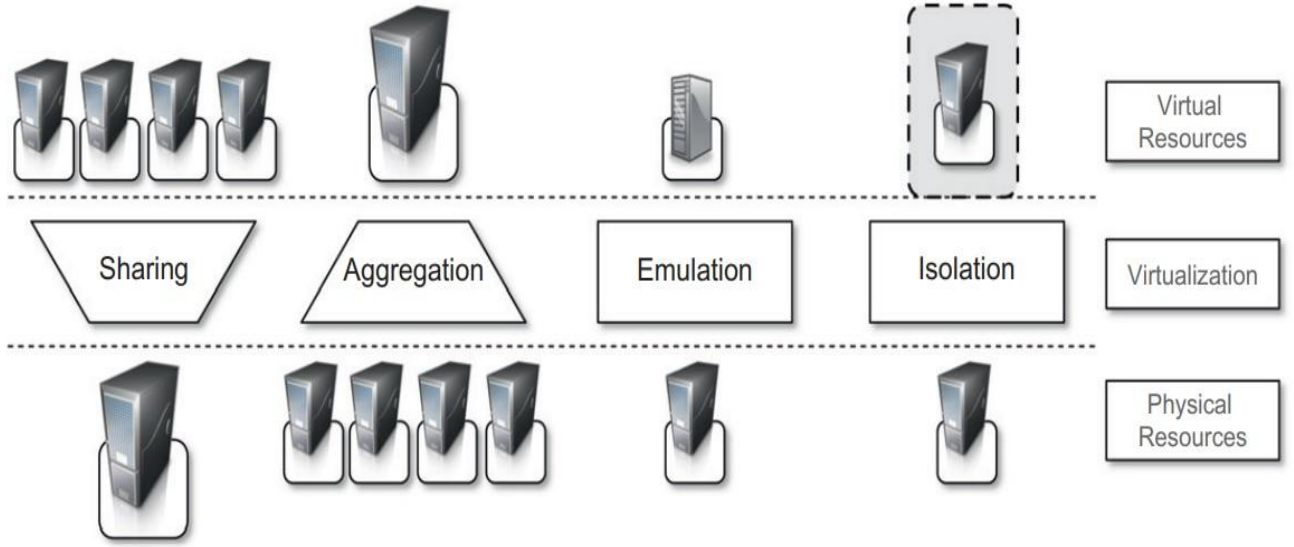


FIGURE 3.2

Functions enabled by managed execution.

- **Sharing.**

  Virtualization allows the creation of a separate computing environments within the same host. In this way it is possible to fully exploit the capabilities of a powerful guest, which would otherwise be underutilized.

  Sharing is a particularly important feature in virtualized data centers, where this basic feature is used to reduce the number of active servers and limit power consumption.

- **Aggregation.**

Not only is it possible to share physical resource among several guests, but virtualization also allows aggregation, which is the opposite process.

A group of separate hosts can be tied together and represented to guests as a single virtual host.

This function is naturally implemented in middleware for distributed computing, with a classical example represented by cluster management software, which harnesses the physical resources of a homogeneous group of machines and represents them as a single resource.

- **Emulation.**

- Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program.
- This allows for controlling and tuning the environment that is exposed to guests.
- For instance, a completely different environment with respect to the host can be emulated, thus allowing the execution of guest programs requiring specific characteristics that are not present in the physical host.
- This feature becomes very useful for testing purposes, where a specific guest has to be validated against different platforms or architectures and the wide range of options is not easily accessible during development.

- **Isolation.**

Virtualization allows providing guests—whether they are operating systems, applications, or other entities—with a completely separate environment, in which they are executed.

The guest program performs its activity by interacting with an abstraction layer, which provides access to the underlying resources.

Isolation brings several benefits; for example, it allows multiple guests to run on the same host without interfering with each other.

Second, it provides a separation between the host and the guest.

The virtual machine can filter the activity of the guest and prevent harmful operations against the host.

Portability

- The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged into a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines.
- Except for the file size, this happens with the same simplicity with which we can display a picture image in different computers.
- Virtual images are generally proprietary formats that require a specific virtual machine manager to be executed.
- In the case of programming-level virtualization, as implemented by the JVM or the .NET runtime, the binary code representing application components (jars or assemblies) can be run without any recompilation on any implementation of the corresponding virtual machine.
- This makes the application development cycle more flexible and application deployment very straightforward: One version of the application, in most cases, is able to run on different platforms with no changes.

3.Taxonomy of virtualization techniques

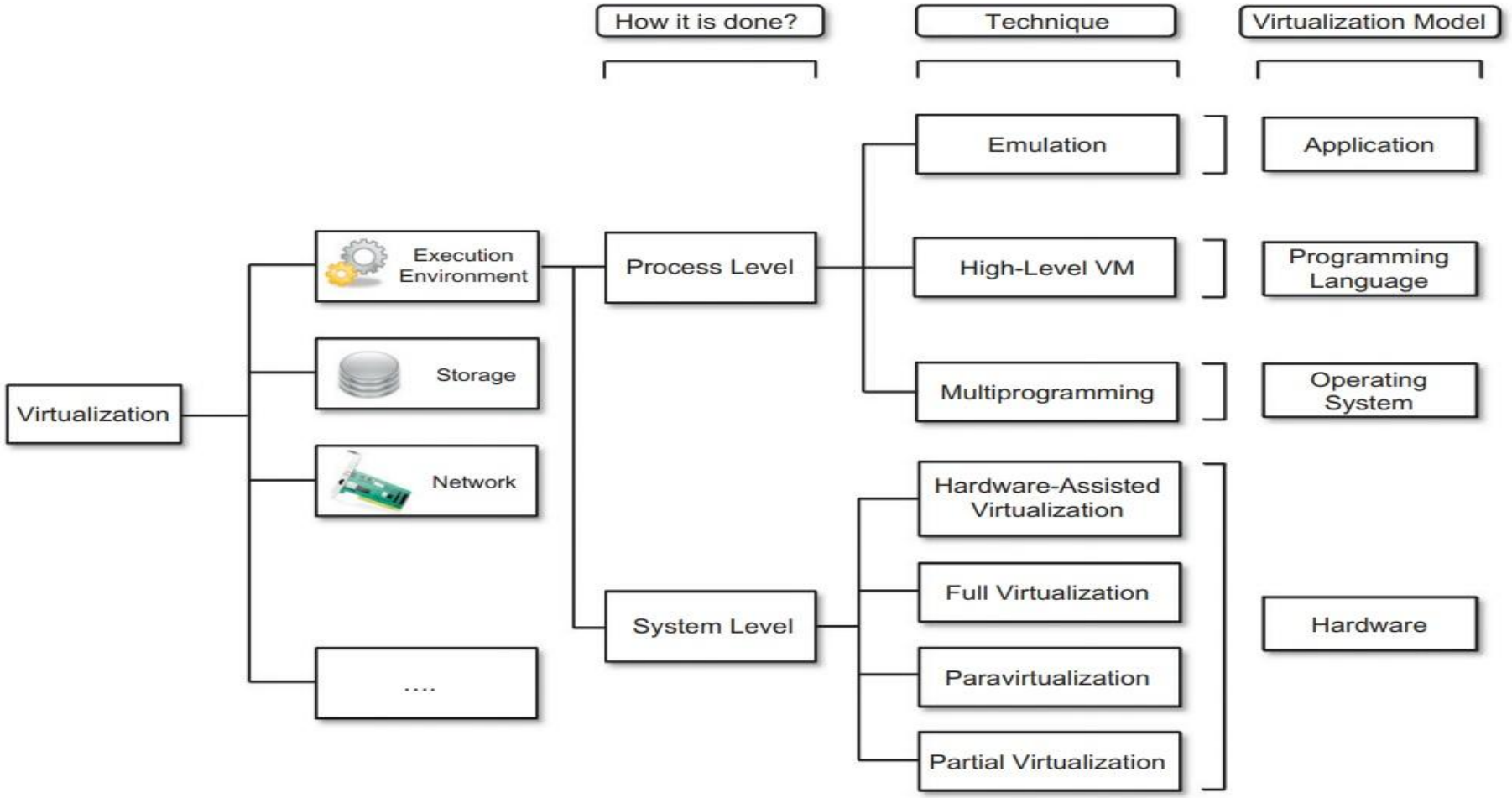


FIGURE 3.3

A taxonomy of virtualization techniques.

The first classification discriminates against the service or entity that is being emulated. Virtualization is mainly used to emulate

- execution environments
- Storage
- 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.

3.1 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.

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.

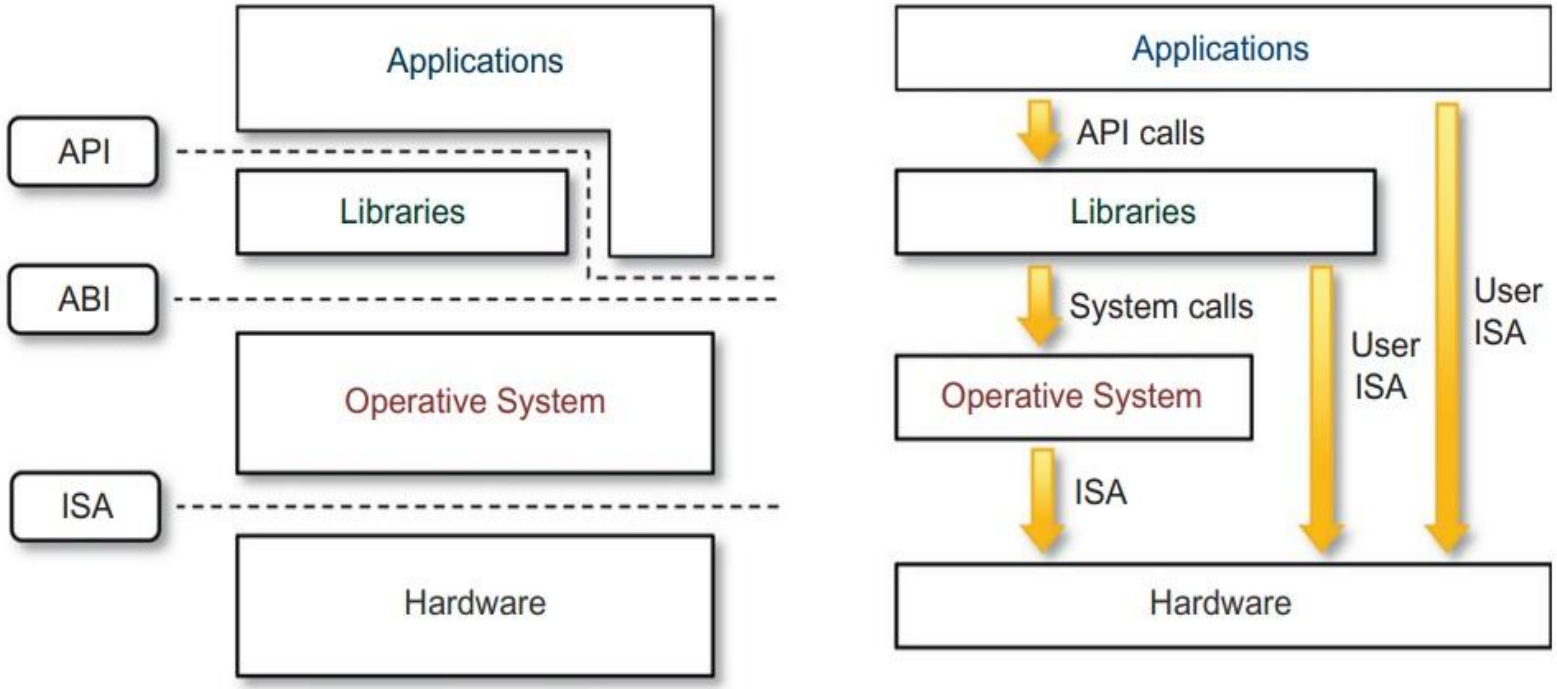


FIGURE 3.4

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 (OS) 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.

- For this purpose, the instruction set exposed by the hardware has been divided into different security classes that define who can operate with them.
- The first distinction can be made between
  - **Privileged instructions**
  - **Nonprivileged instructions.**
- **Nonprivileged instructions** are those instructions that can be used without interfering with other tasks because they do not access shared resources.
- This category contains, for example, all the floating, fixed-point, and arithmetic instructions.

- **Privileged instructions** are those that are executed under specific restrictions and are mostly used for sensitive operations, which expose (behavior-sensitive) or modify (control-sensitive) the privileged state.
- For instance, behavior-sensitive instructions are those that operate on the I/O, whereas control-sensitive instructions alter the state of the CPU registers.
- Some types of architecture feature more than one class of privileged instructions and implement a finer control of how these instructions can be accessed.

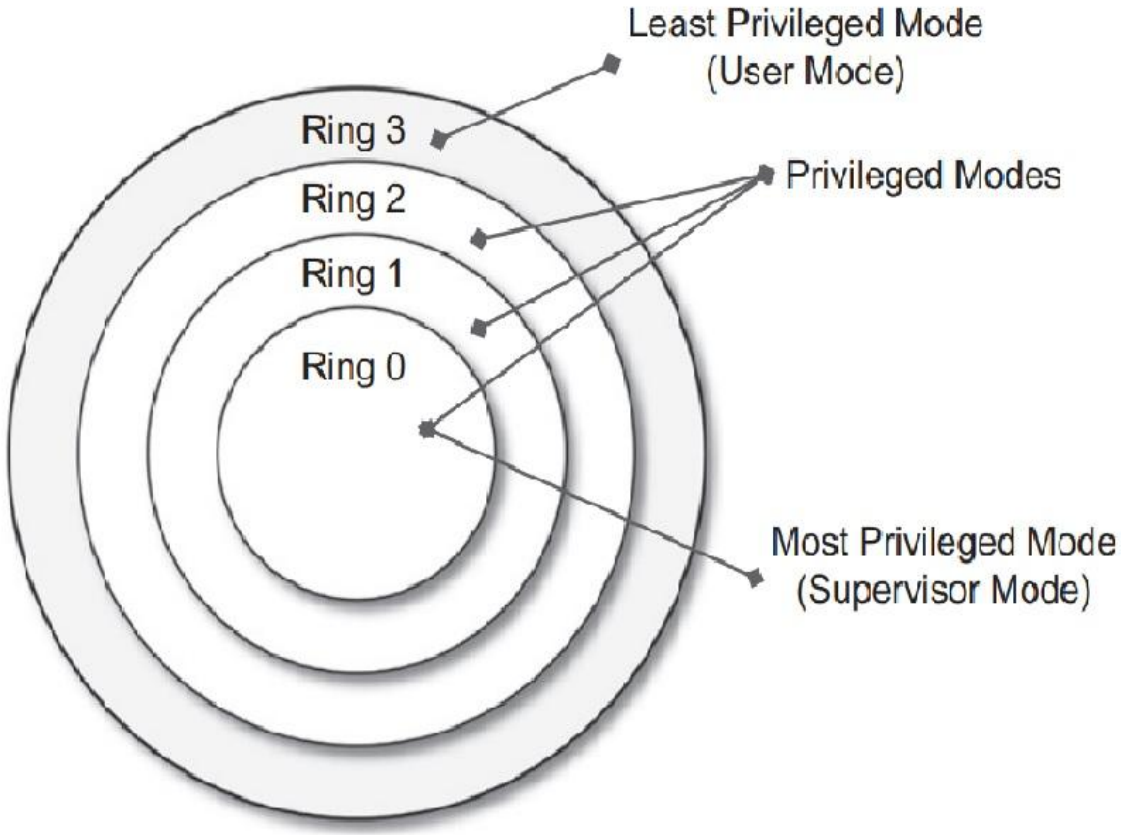


FIGURE 3.5

Security rings and privilege modes.

- For instance, a possible implementation features a hierarchy of privileges in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3.
- Ring 0 is in the most privileged level and Ring 3 in the least privileged level.
- Ring 0 is used by the kernel of the OS, rings 1 and 2 are used by the OS-level services, and Ring 3 is used by the user.
- Recent systems support only two levels, with Ring 0 for supervisor mode and Ring 3 for user mode.

- All the current systems support at least two different execution modes
  - Supervisor mode
  - User mode.

  In **Supervisor(kernel) mode,** it denotes an execution mode in which all the instructions (privileged and nonprivileged) can be executed without any restriction. This mode, also called master mode or kernel mode, is generally used by the operating system (or the hypervisor) to perform sensitive operations on hardware level resources.

  In **user mode**, there are restrictions to control the machine-level resources. If code running in user mode invokes the privileged instructions, hardware interrupts occur and trap the potentially harmful execution of the instruction. Despite this, there might be some instructions that can be invoked as privileged instructions under some conditions and as nonprivileged instructions under other conditions.