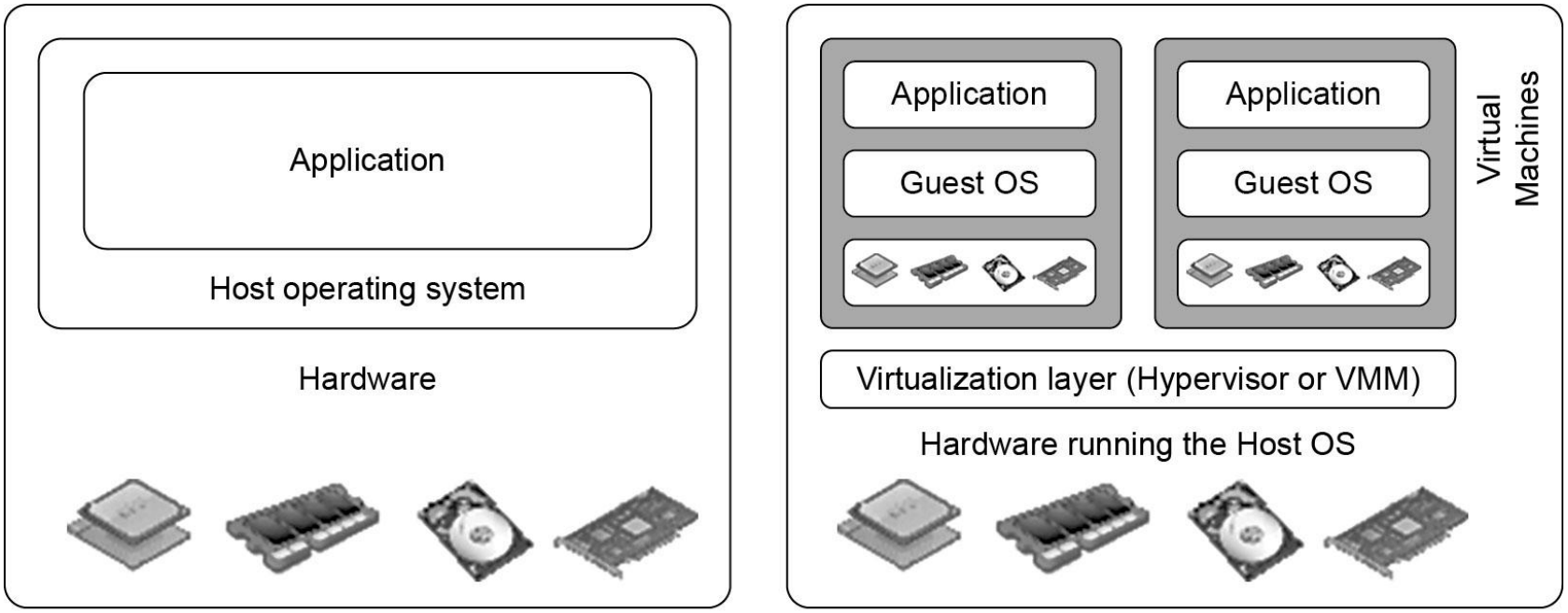
Virtualization

Virtual Machines and Virtualization of datacenters

Acknowledgement: 1. Prof. Rajkumar Buyya for providing few figures appear in this presentation
2. Internet

Difference between Traditional Computer and Virtual machines



(a) Traditional computer

(b) After virtualization

Ref.: Distributed and Cloud Computing: From Parallel Processing to the Internet of Things Kai Hwang , Jack Dongarra , Geoffrey C. Fox.

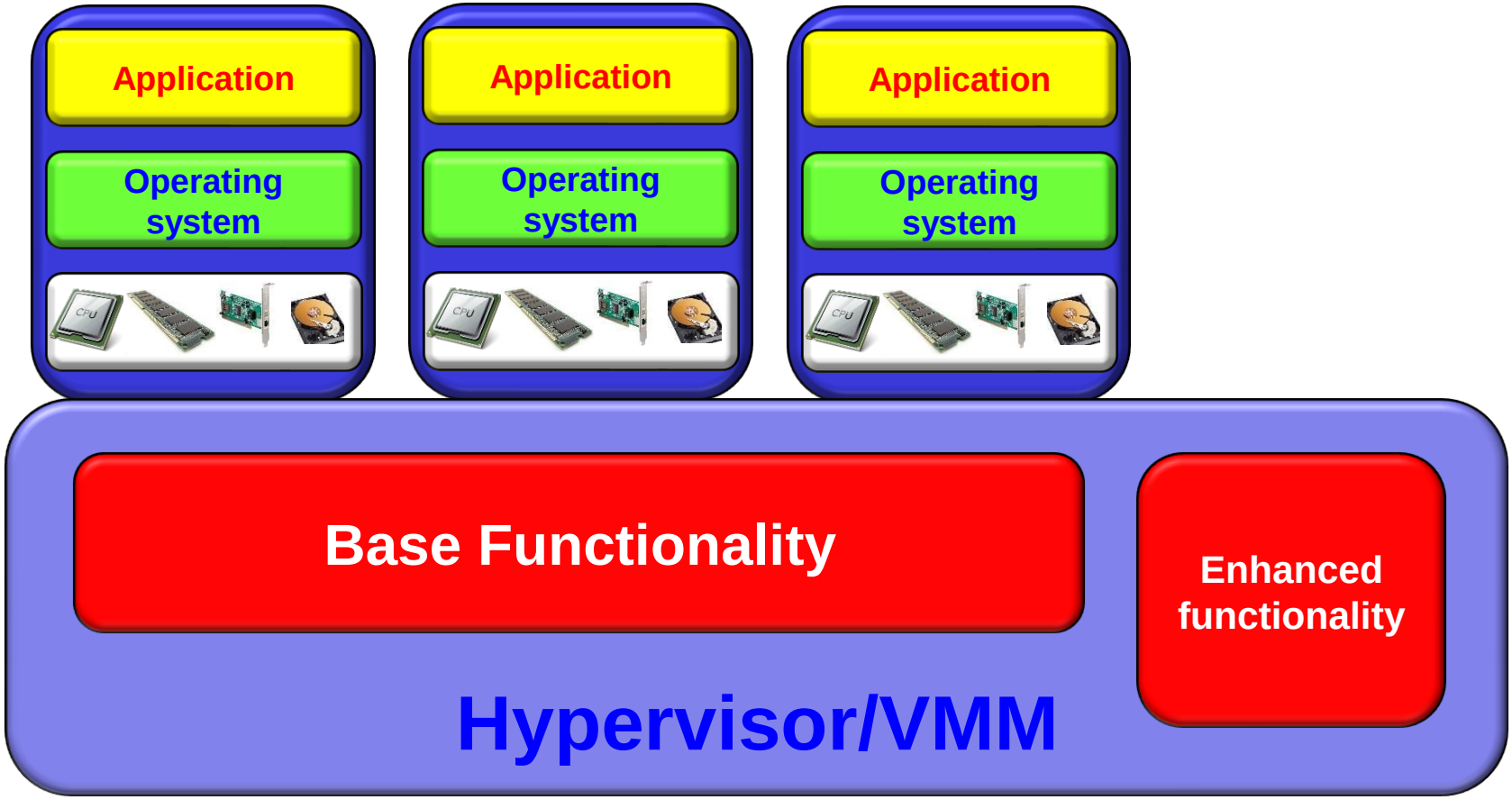
Introduction

- **Virtualization** is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether this is virtual hardware and/or operating system—to run applications.
- This term is often synonymous with *hardware virtualization*, which plays fundamental role in efficiently delivering *Infrastructure-as-a-Service* solutions for *Cloud computing*.

Virtualization

- The virtualization layer is the software responsible for hosting and managing all virtual machines on virtual machine monitors (VMMs).
- In certain cases virtualization layer is a hypervisor running directly on the hardware
- VMM running as a hypervisor implements the VM hardware abstraction and is responsible for running a guest OS.
- VMM has to partition and share the CPU, memory and I/O devices to successfully virtualize the system.

The hypervisor manages hosted virtual machines



Motivation

Virtualization technologies are not new (1960, CP-40 OS on System 360 mainframe) but have gained popularity recently due to the convergence of different phenomena:

- *Increased performance and computing capacity*
- *Underutilized hardware and software resources*
- *Lack of space*
- *Greening initiatives*
- *Rise of administrative costs*

Define Virtualization!

- Virtualization refers to the creation of a virtual resource such as a server, desktop, operating system, file, storage or network.
- *Virtualization is a level of indirection between hardware and software*
- Virtual Machine abstraction
  - Run all software written for physical machines

What is Emulation?

- **Emulation** refers to the ability of a computer program in an electronic device to **imitate** another program or device.
- VM can emulate complete hardware, means and unmodified guest OS for one PC can be run
  - Bochs, BIRD, QEMU

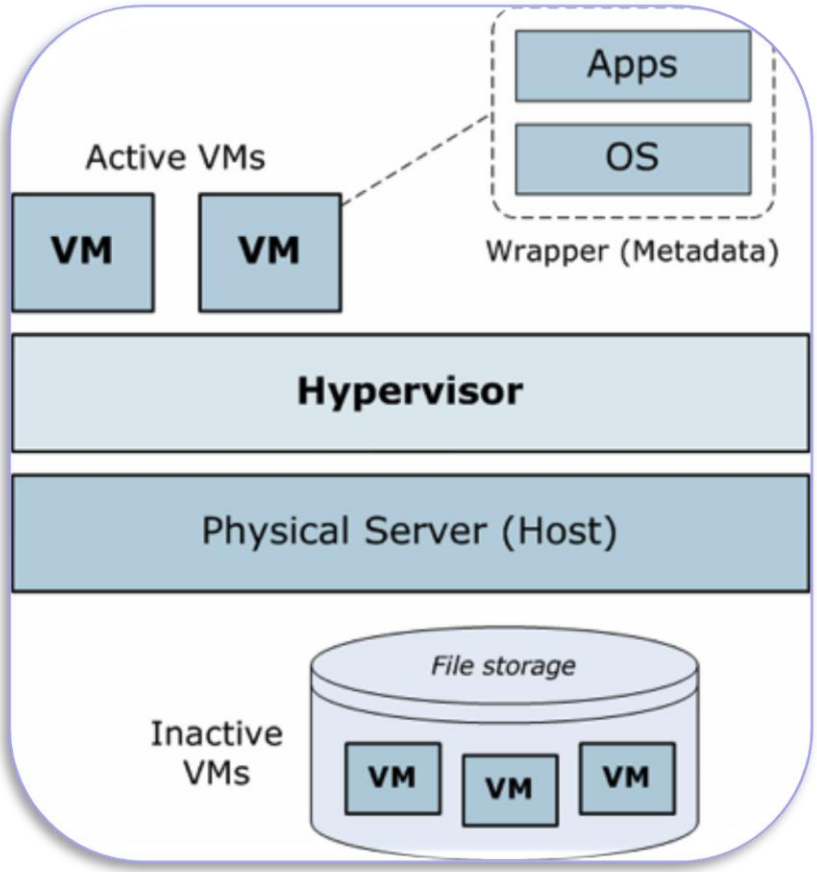
Difference between Virtualization and Emulation

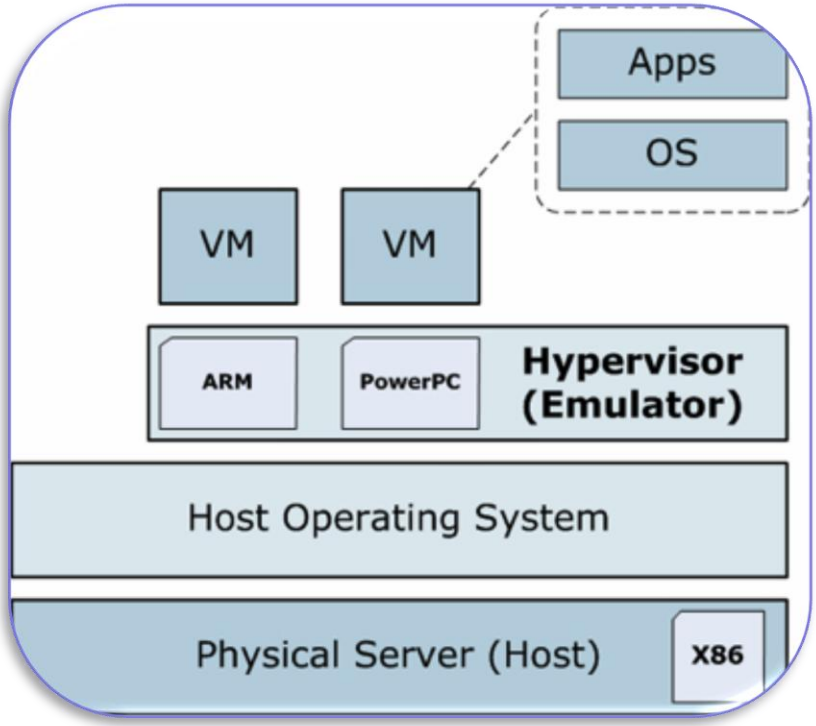
- Virtualization involves simulating parts of a computer's hardware – **just enough** for a guest operating system to run unmodified - but most operations still occur on the real hardware for efficiency reasons.
- It is faster.
- Ex. VMWare can provide a virtual environment for running a virtual WindowsXP machine. However VMWare cannot work on any real hardware other than a real x86 PC.

Difference between Virtualization and Emulation

- In emulation the virtual machine simulates the complete hardware in software.
- This allows an operating system for one computer architecture to be run on the architecture that the emulator is written for.

Difference between Virtualization and Emulation

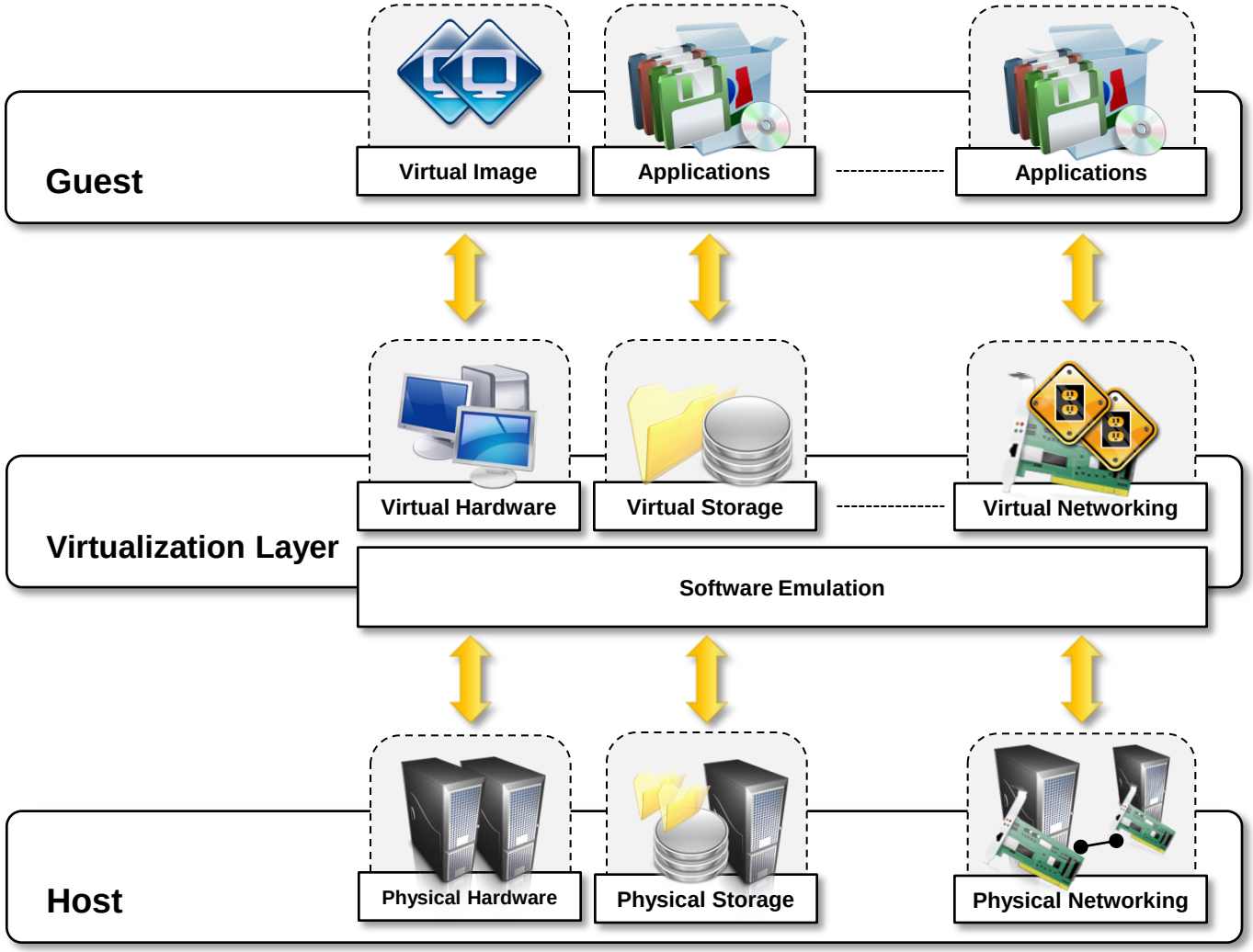




PowerPC: Performance Optimization With Enhanced RISC-Performance Computing, (1991), Apple-IBM-Motorola

Components of Virtualized Environments

- In a virtualized environment there are three major components: *guest*, *host*, and *virtualization layer*.
  - The *guest* represents the system component that interacts with the virtualization layer rather than with the host as it would normally happen.
  - The *host* represents the original environment where the guest is supposed to be managed.
  - The *virtualization layer* is responsible for recreating the same or a different environment where the guest will operate.



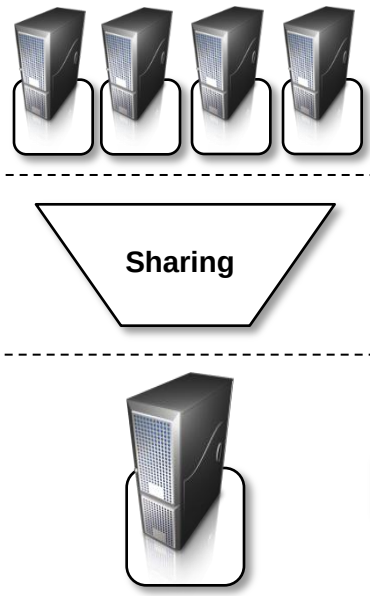
The main common characteristic: virtual environment is created by means of a *software program*.

Managed Execution

- Virtualization of the **execution environment** does not only allow the increased security but a wider range of features can be implemented.
  - *Sharing*,
  - *aggregation*,
  - *emulation*, and
  - *isolation*

  are the most relevant.

Managed Execution Conti..

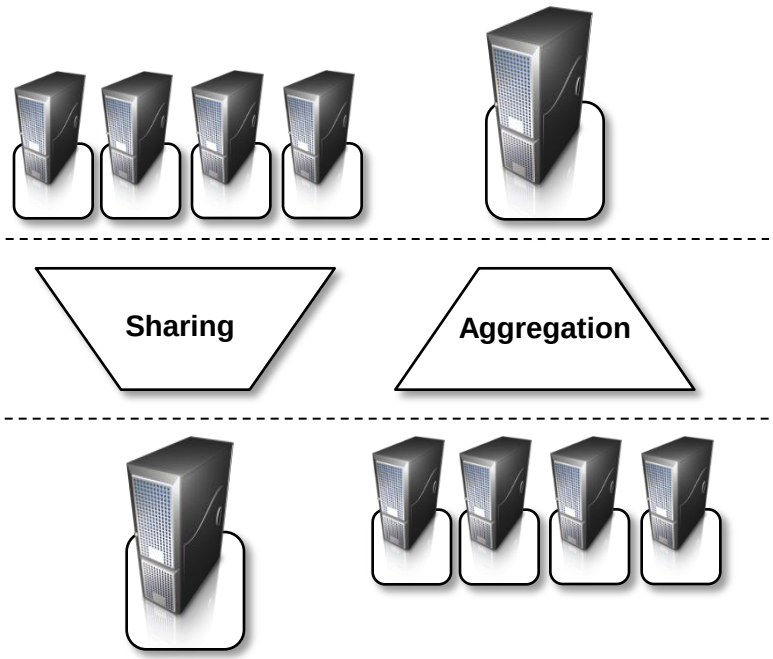


Virtual Resources

Virtualization

Physical Resources

Managed Execution Conti..

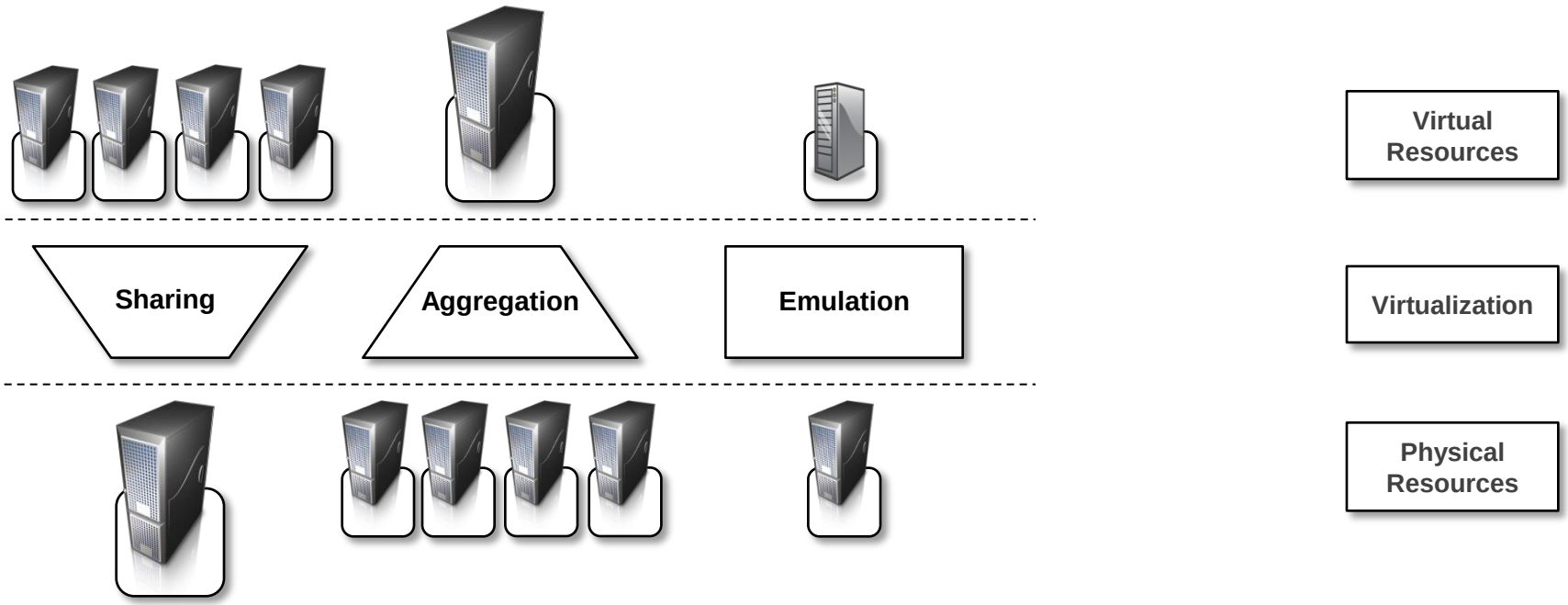


Virtual Resources

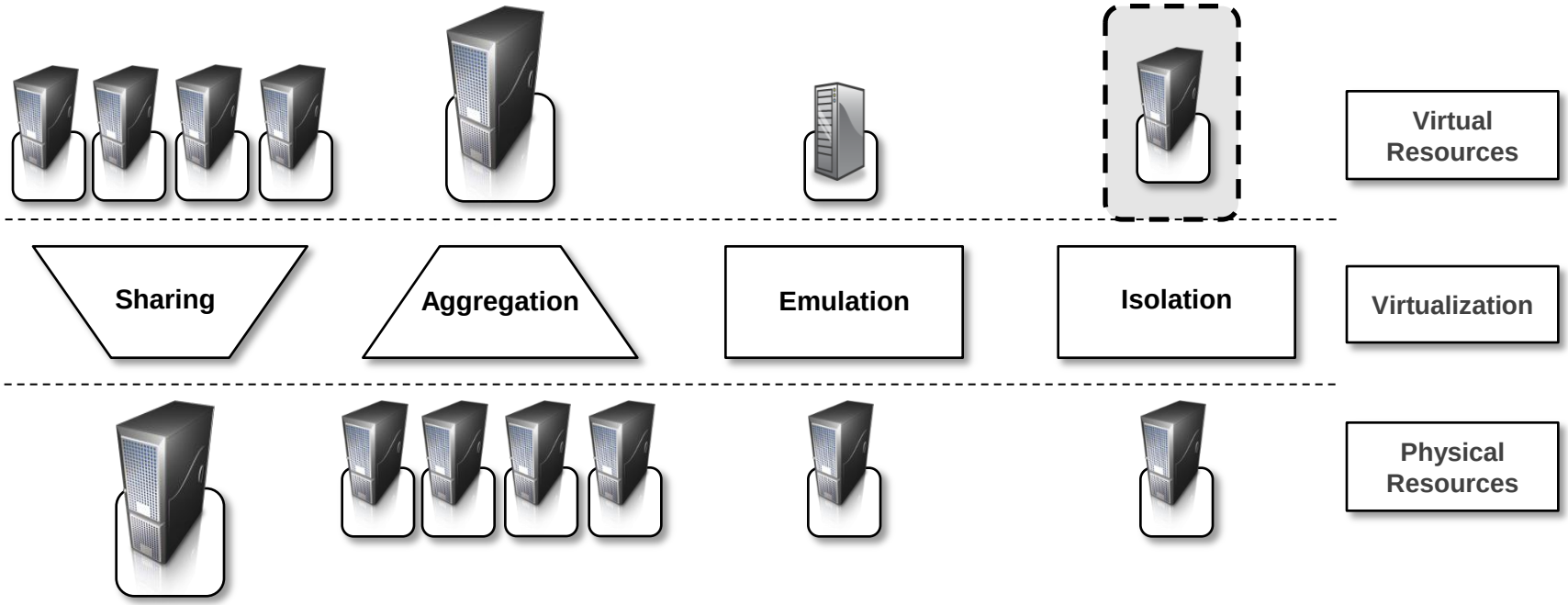
Virtualization

Physical Resources

Managed Execution Conti..



Managed Execution Conti..



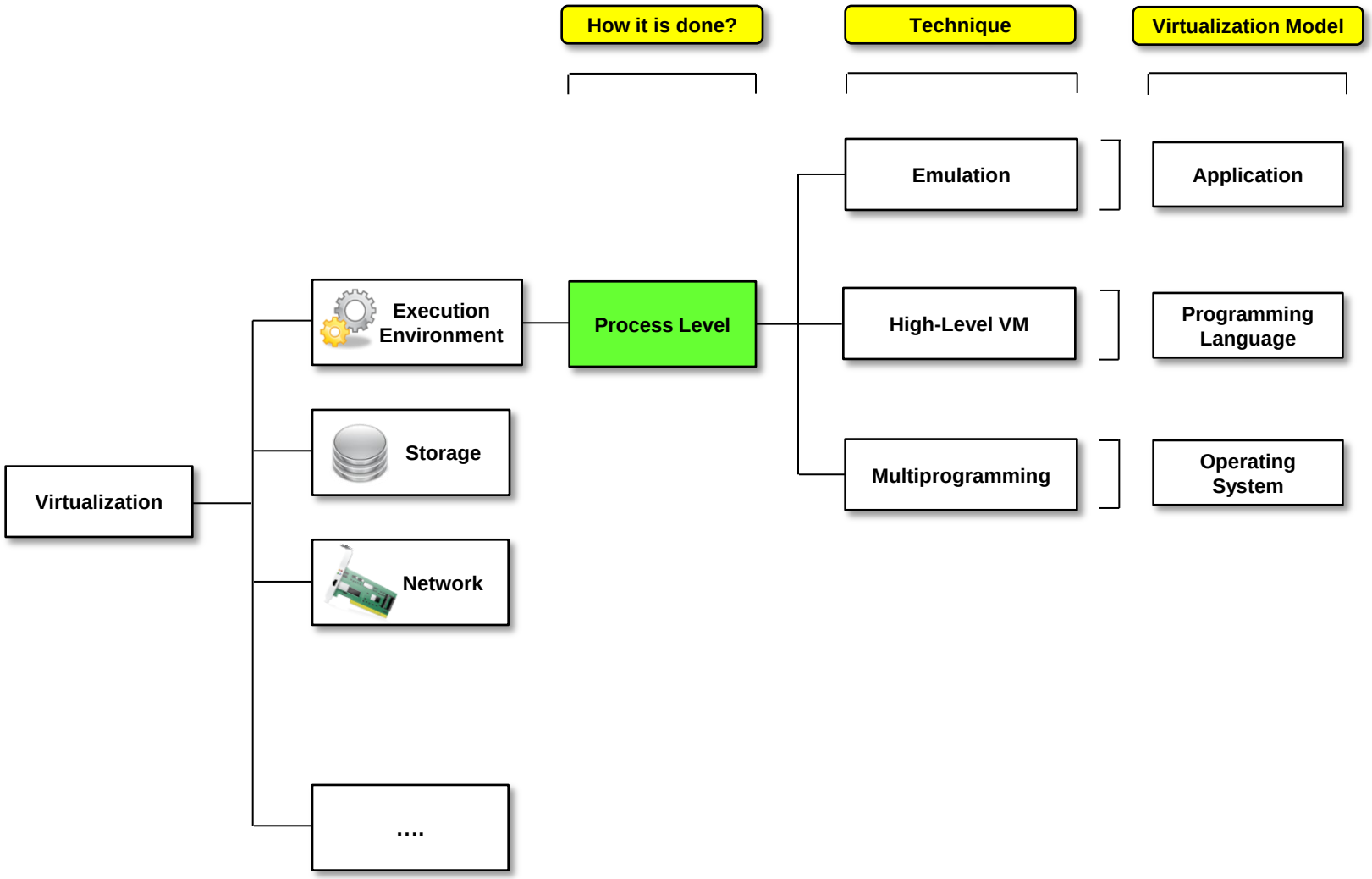
Managed Execution Conti..

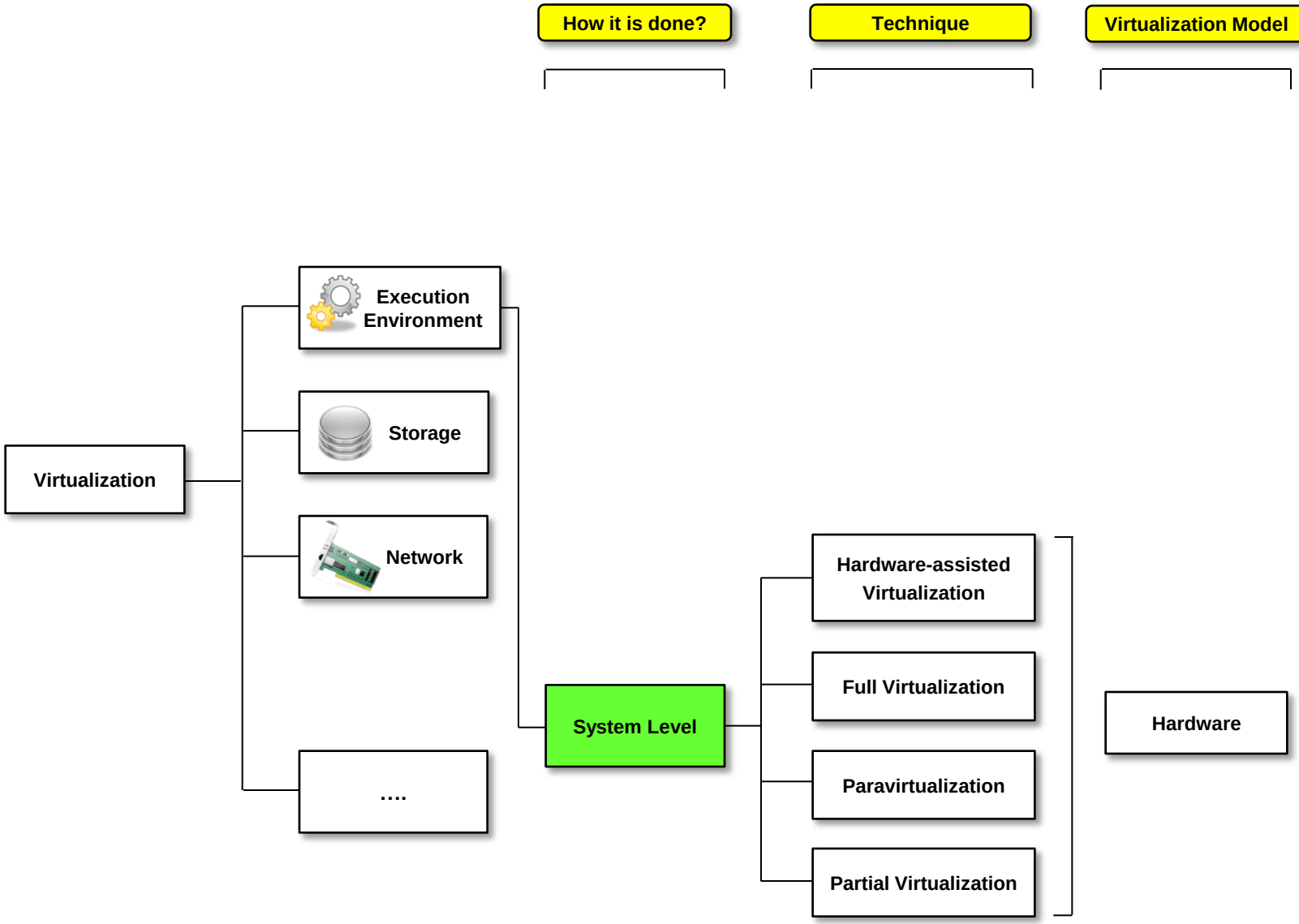
- *Sharing.* Virtualization allows the creation of a separate computing environment within the same host.
- *Aggregation.* Virtualization also allows the aggregation resources, which is the opposite process.
- *Emulation.* Guests are executed within an environment that is controlled by the virtualization layer, which ultimately is a program. *Emulation* allows for controlling and tuning the environment that is exposed to guests.

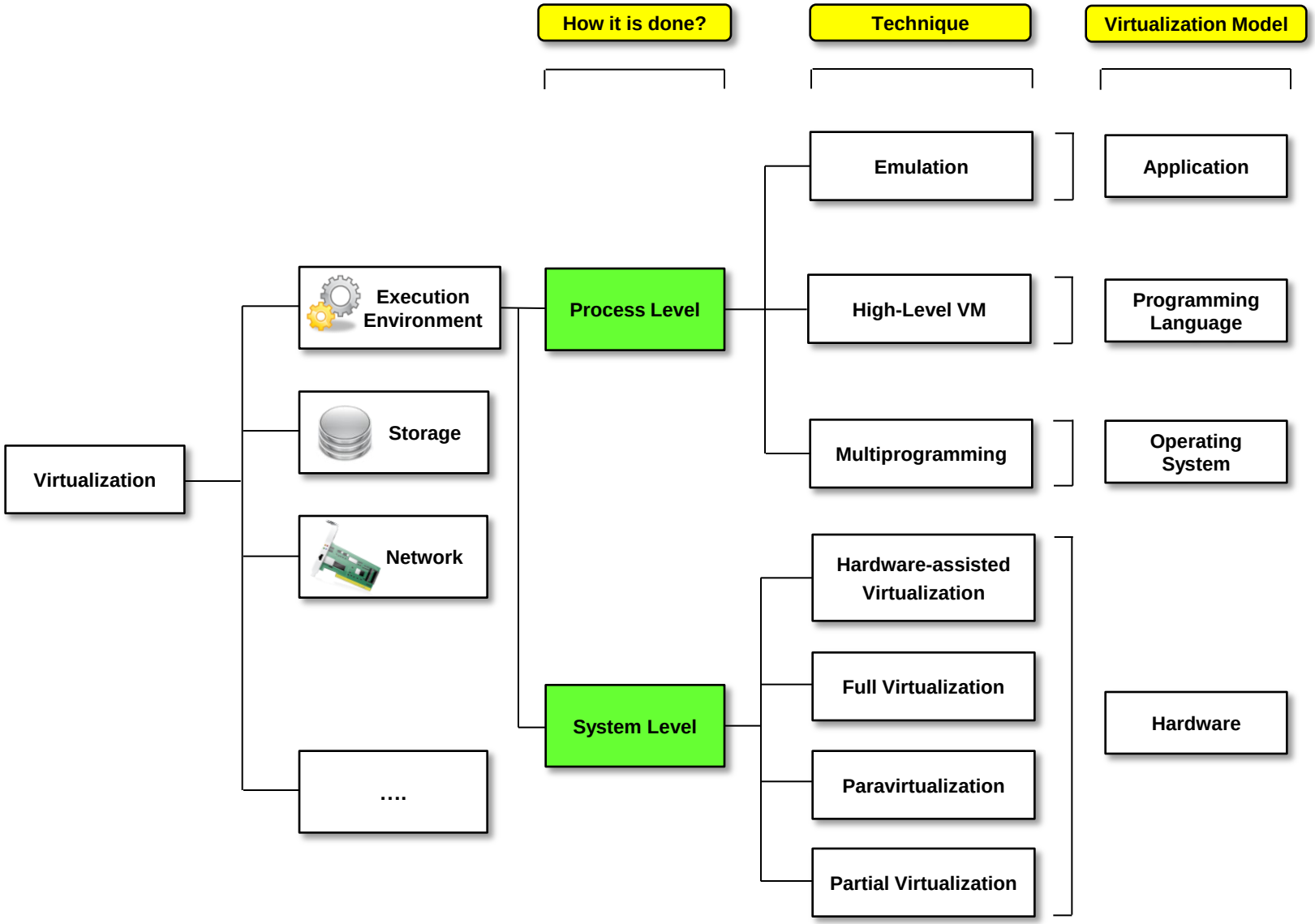
Managed Execution Conti..

- *Isolation.* Virtualization allows providing guests—whether they are operating systems, applications, or other entities—with a complete separate environment, in which they are executed.
- *performance tuning*. Control the performance of the *guest* by finely tuning the properties of the resources exposed to effectively implement a QoS infrastructure that more easily fulfill the SLA established for the guest.

Taxonomy of Virtualization Techniques


How it is done?
Technique
Virtualization Model
Virtualization
Execution Environment
Storage
Network
....
Process Level
Emulation
High-Level VM
Multiprogramming
Application
Programming Language
Operating System


How it is done?
Technique
Virtualization Model
Virtualization
Execution Environment
Storage
Network
....
System Level
Hardware-assisted Virtualization
Full Virtualization
Paravirtualization
Partial Virtualization
Hardware

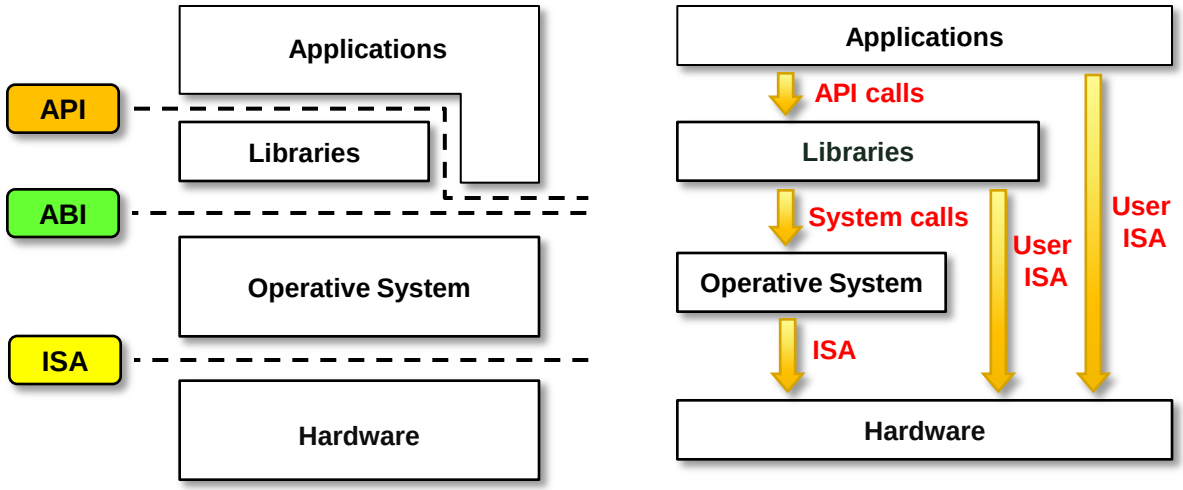

How it is done?
Technique
Virtualization Model
Virtualization
Execution Environment
Storage
Network
....
Process Level
System Level
Emulation
High-Level VM
Multiprogramming
Hardware-assisted Virtualization
Full Virtualization
Paravirtualization
Partial Virtualization
Application
Programming Language
Operating System
Hardware

Execution Virtualization

- It includes all those techniques whose aim is to *emulate* an execution environment that is separate from the one which is hosting the virtualization layer.
- It can be implemented directly
  - on top of the hardware,
  - by the operating system,
  - an application,
  - libraries dynamically
  - statically linked against an application image

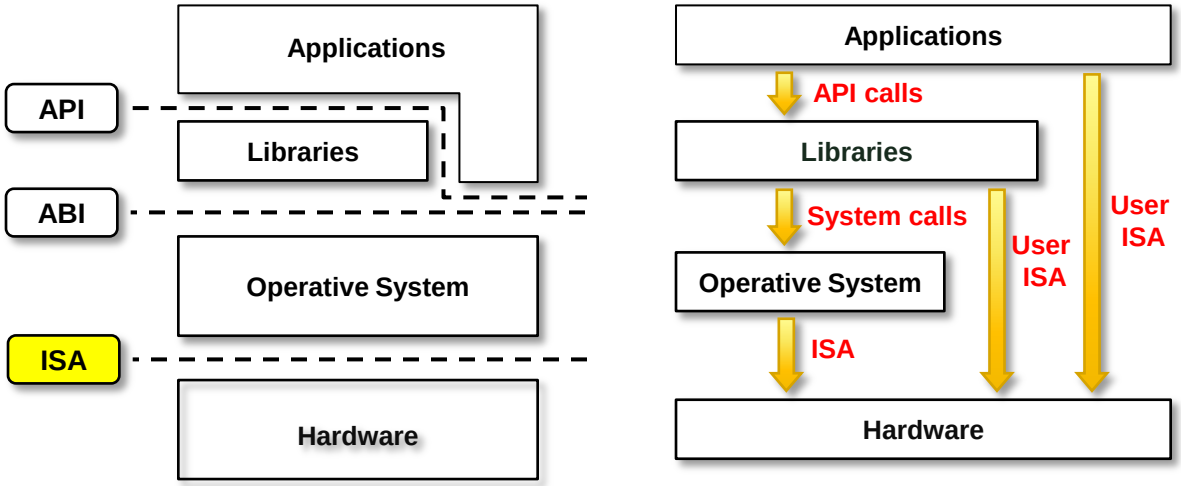
Machine Reference Model

- Reference model defines the interfaces between the levels of abstractions, which hide implementation details.



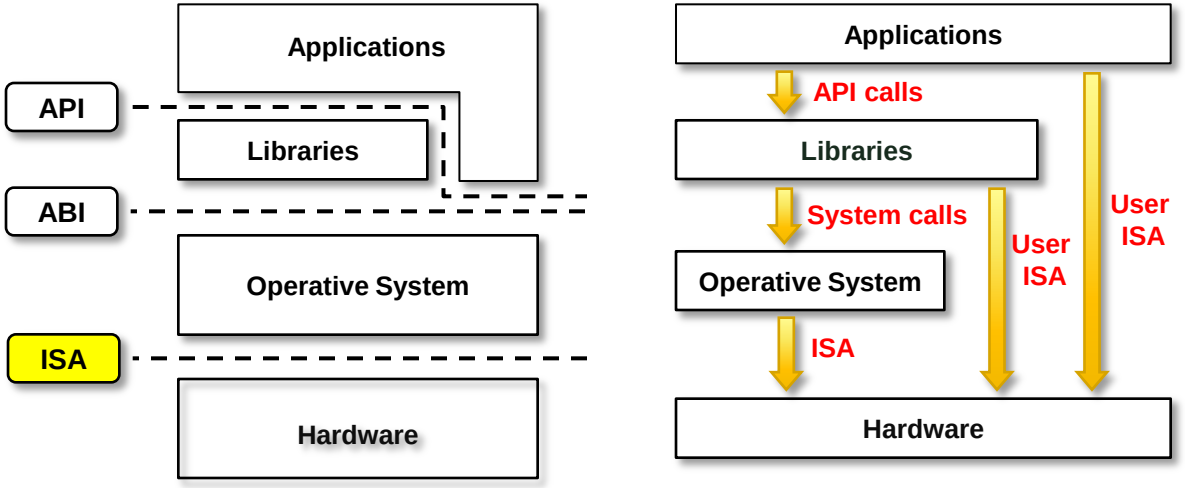
ISA

- 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 interrupts management.



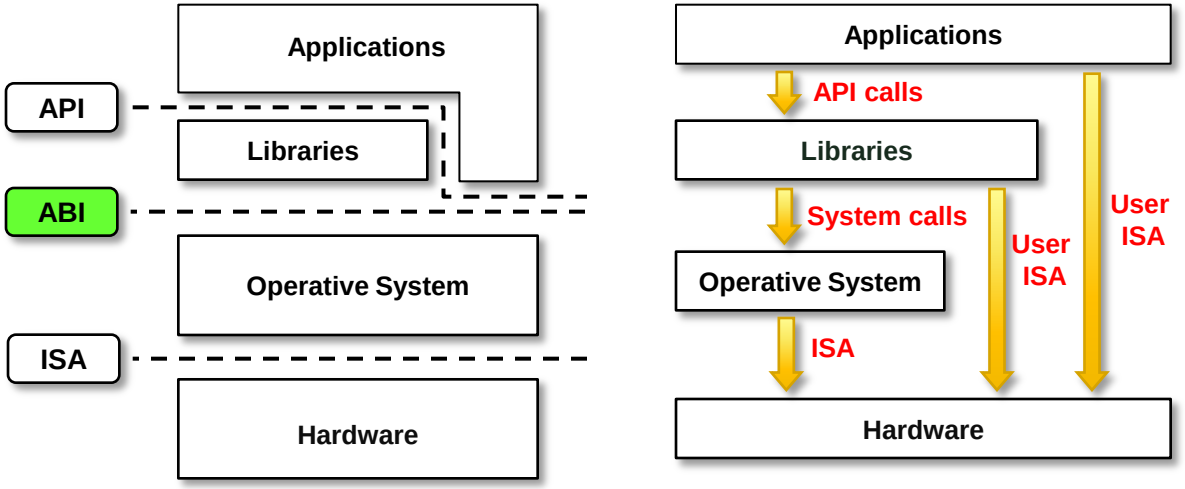
ISA Conti..

- ISA is the interface between HW/SW and it is important for the OS developer (using *System ISA*), and applications developers to directly manage the underlying hardware (using *User ISA*).



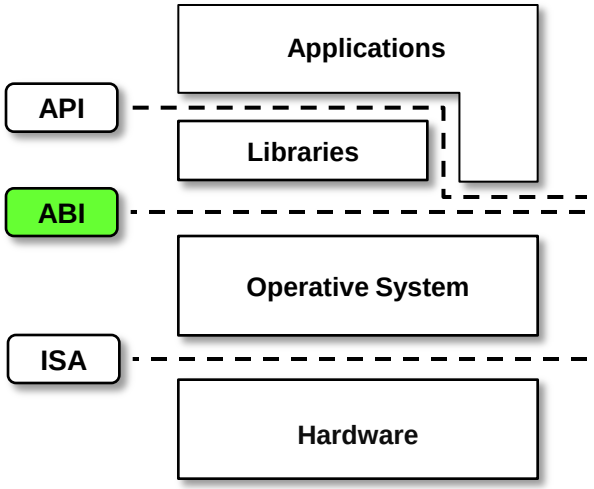
ABI

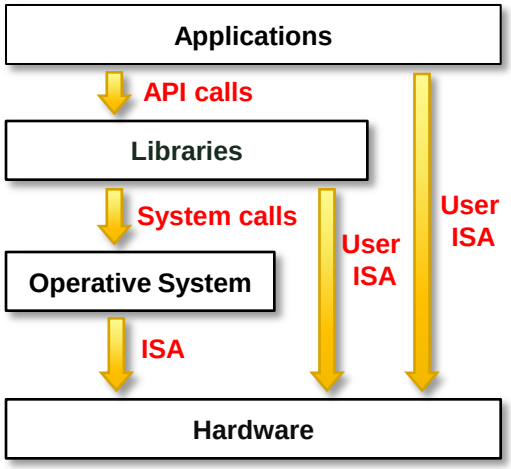
- The *Application Binary Interface (ABI)* separates the OS layer from the applications and libraries, which are managed by the OS.



ABI Conti..

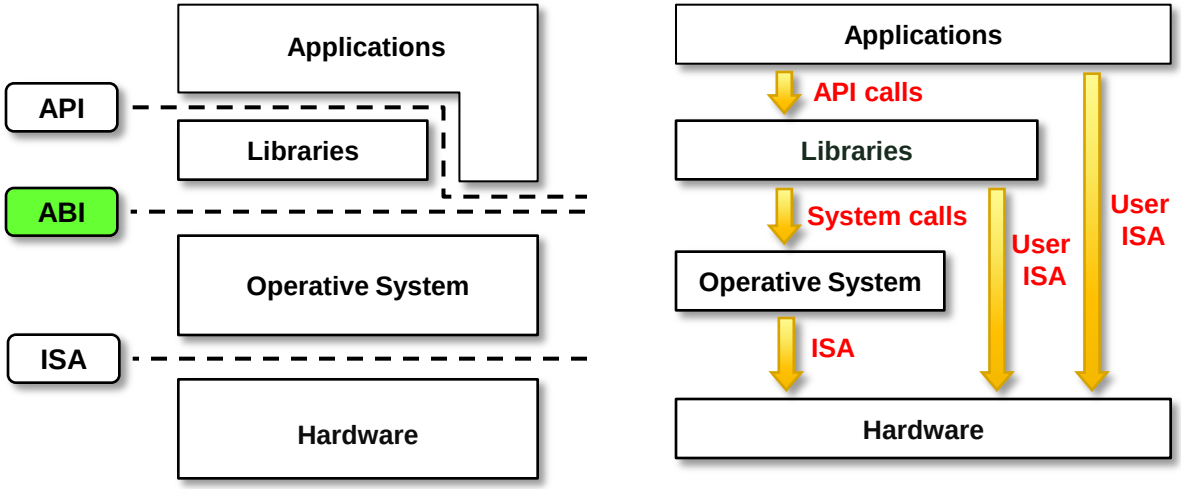
- ABI covers details such as low-level data types, alignment, and call conventions and defines a format for executable programs. System calls are defined at this level.





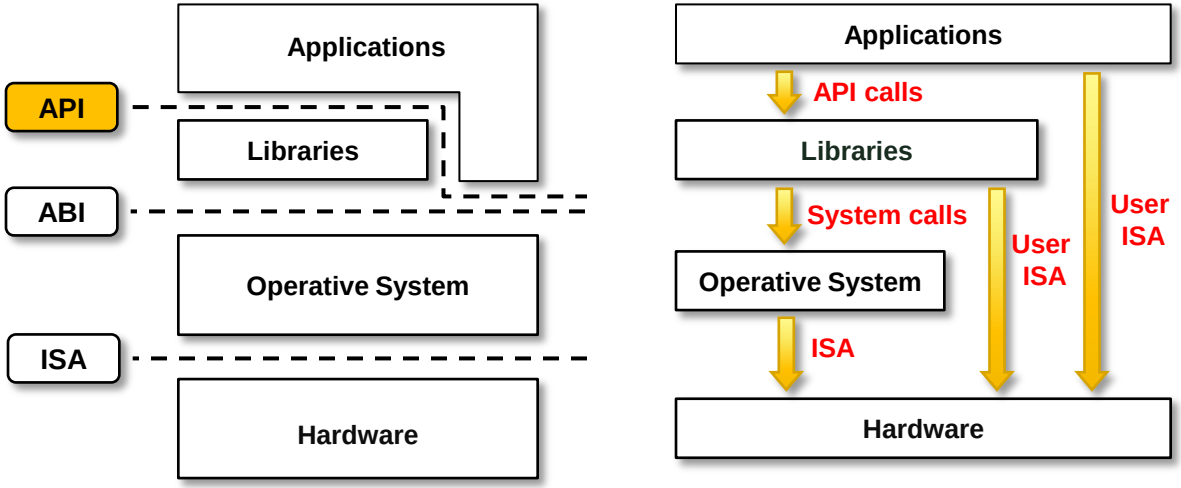
ABI Conti..

- This interface allows portability of applications and libraries across operating systems that implement the same ABI.

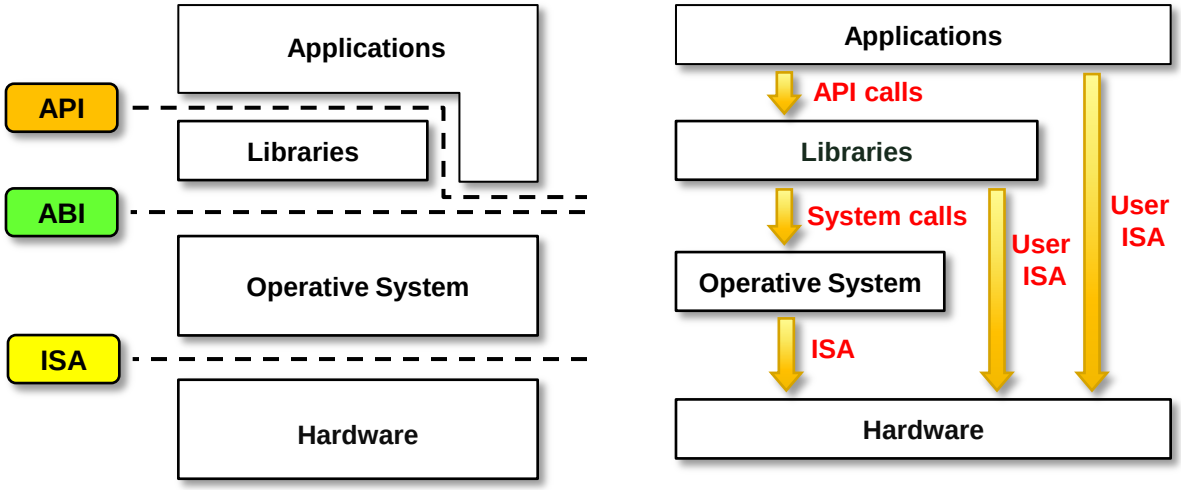


API

- The highest level of abstraction is represented by the *Application Programming Interface (API)*, which interfaces applications to libraries and/or the underlying operating system.



- Such reference model requires limited knowledge of the entire computing stack, and also provides ways for implementing a minimal security model for managing and accessing shared resources.



Privileged and *non-privileged* instructions

- Non-privileged 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 and *non-privileged* 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, while control sensitive instructions alter the state of the CPU registers.