Virtualization

Course: Distributed sw Systems

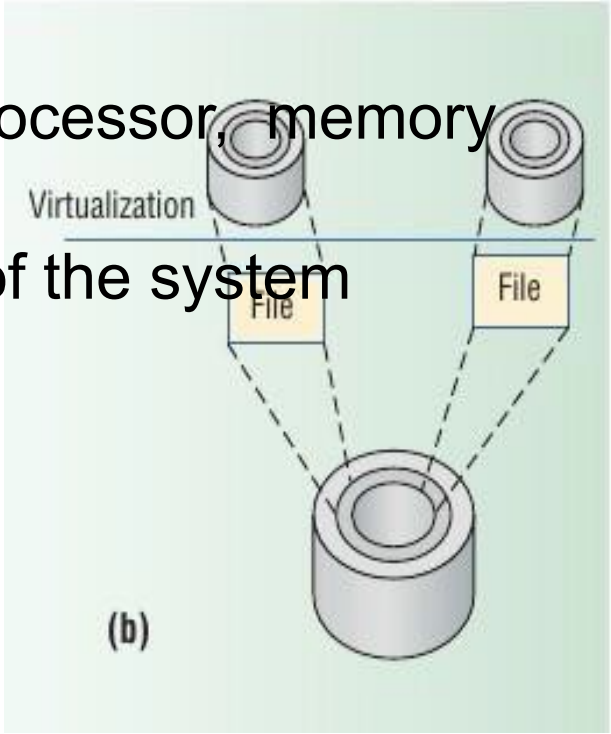
Lecture 12.1

Questions

- The goals of virtualization in distributed systems
- How virtualization works
- The distinction between a virtual machine *instance* and a virtual machine *image*
- The distinction between virtualizing hardware and virtualizing operating systems.

Virtualization

- Virtualization of system or components like: processor, memory or an I/O device
- It transforms a entire system or components of the system
- Example: disk storage



Network virtualization - examples

- VLAN

  Multiple logical networks on same physical wires
- VPN virtual private network
- Channel bonding (aka. link aggregation)

  multiple links combined offered a single, higher-bandwidth link
- Computer clusters

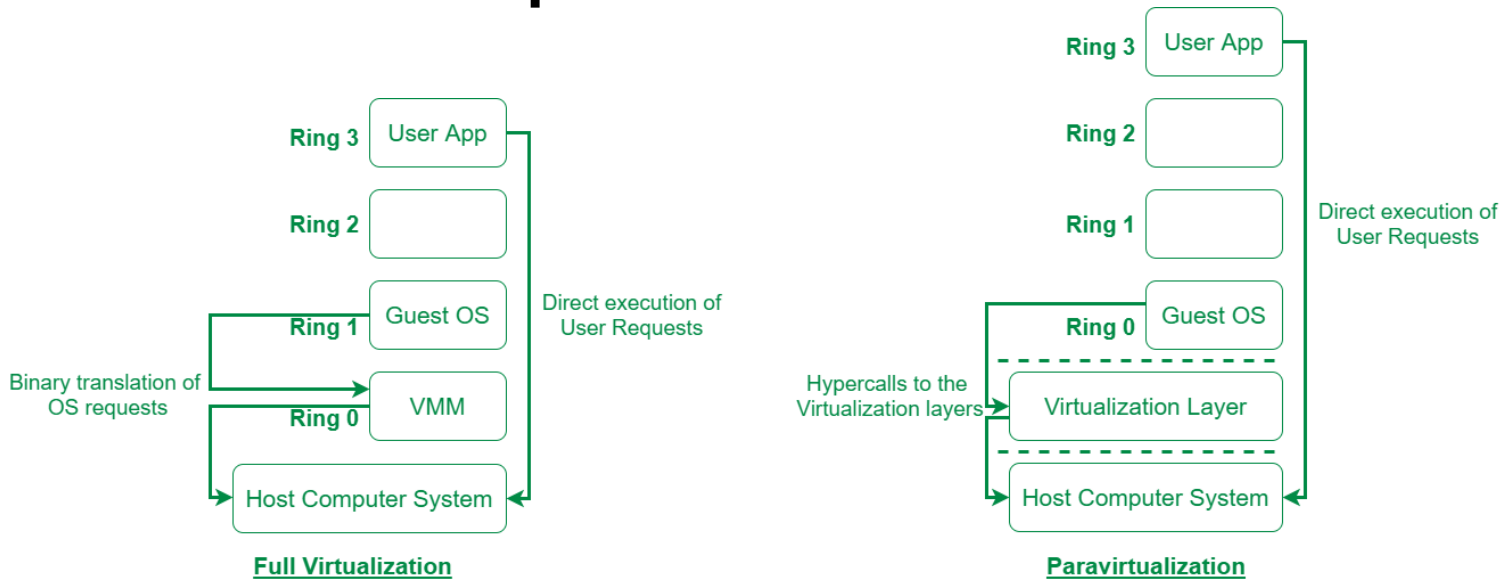
  multiple discrete computers into larger metacomputers e.g. Hadoop
- Virtual network interface cards (NICs) and bridges for VM communication

Please see RFC 8568 on network virtualization research challenges

Full virtualization vs paravirtualization

- Full virtualization – complete simulation of a real hardware to allow software environments, including a guest operating system and its apps, to run unmodified.
- Paravirtualization – the guest apps are executed in their own isolated domains, as if they are running on a separate system, but a hardware environment is not simulated. Guest programs need to be specifically modified to run in this environment.

Virtualization vs paravirtualization



Paravirtualization uses hypercalls for operations to handle instructions at compile time.

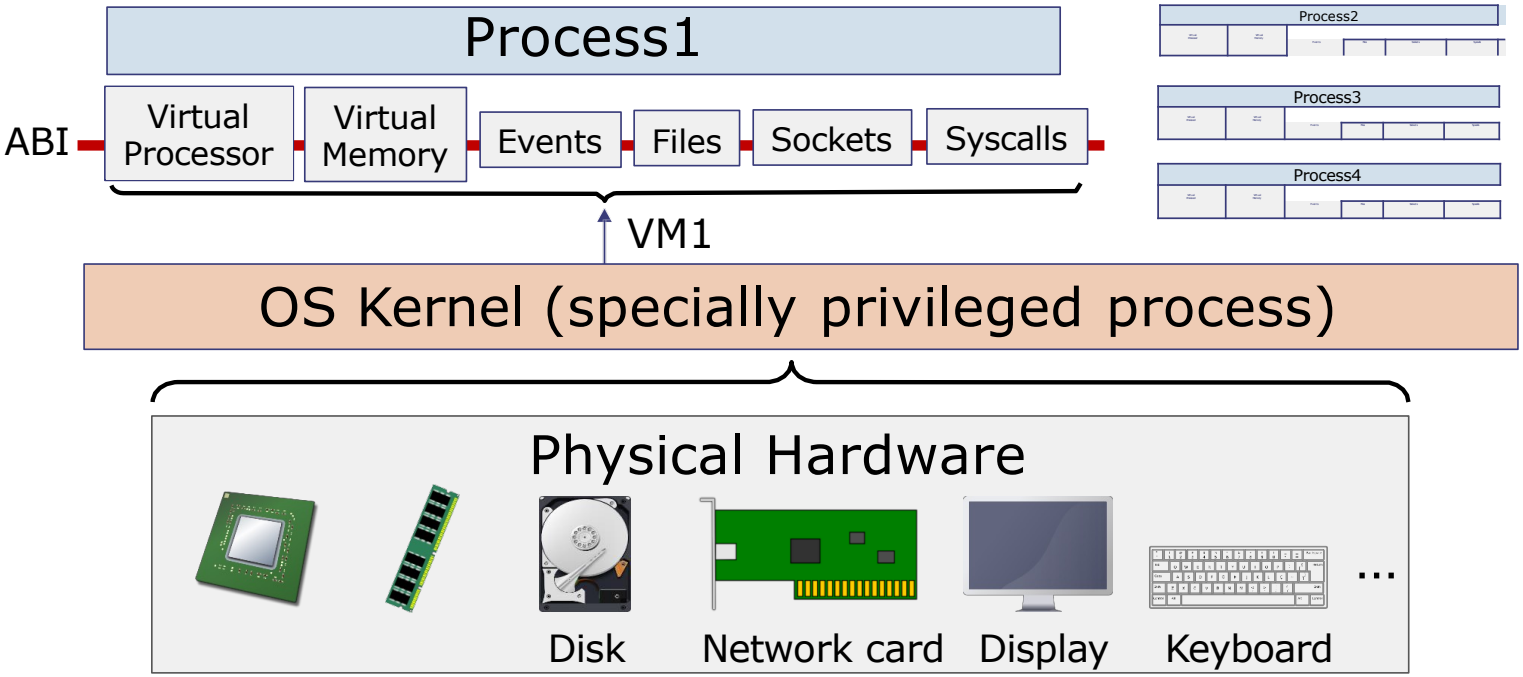
In paravirtualization, guest OS is not completely isolated but it is partially isolated by the virtual machine from the virtualization layer and hardware.

Xen, KVM, and VMware are some examples of paravirtualization

https://news.ycombinator.com/item?id=9728949

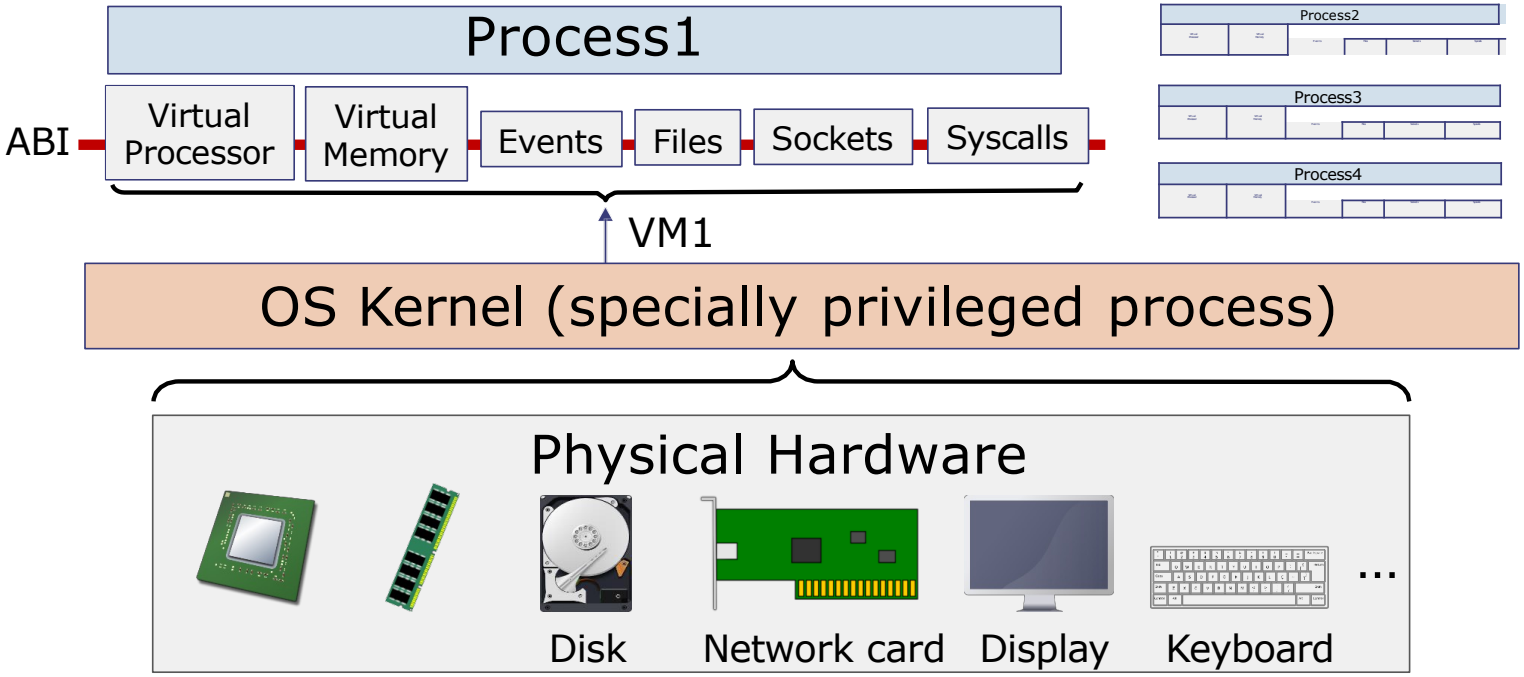
Virtual Machines: A New Layer of Abstraction

- The OS gives a Virtual Machine (VM) to each process
  - Each process believes it runs on its own machine…
  - …but this machine does not exist in physical hardware



Virtual Machines: A New Layer of Abstraction

- A Virtual Machine (VM) is an emulation of a computer system
  - Very general concept, used beyond operating systems



Implementing Virtual Machines

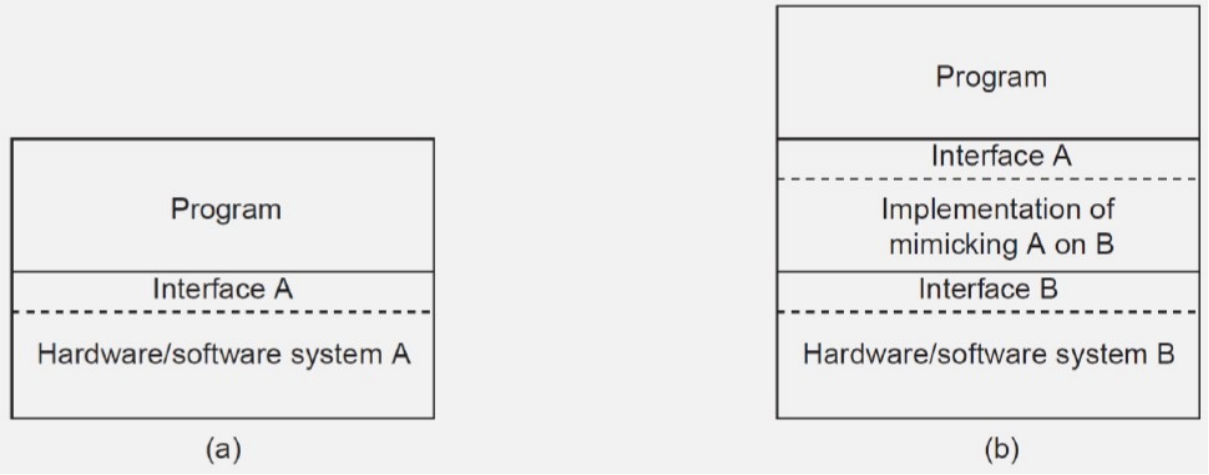
- Virtual machines can be implemented entirely in software, but at a performance cost
  - e.g., Python programs are 10-100x slower than native Linux programs due to Python interpreter overheads
  - The Java Virtual Machine (JVM) executes Java programs compiled «just in time», and is more efficient than Python. However it is slower than C or C++. We do not consider this kind of language-oriented virtual machine in this lecture (see https://en.wikipedia.org/wiki/Java_performance)
- We want to support operating systems with minimal overheads
- ➔ need hardware support for virtual machines!

Virtual Machine

- Virtualization can be applied to entire hw machine.
- VM can be implemented by *adding a* ***software layer*** **to a real machine** to *support desired architecture*.
- VM implementation lie at *architected interfaces*

Virtualization (emulation)

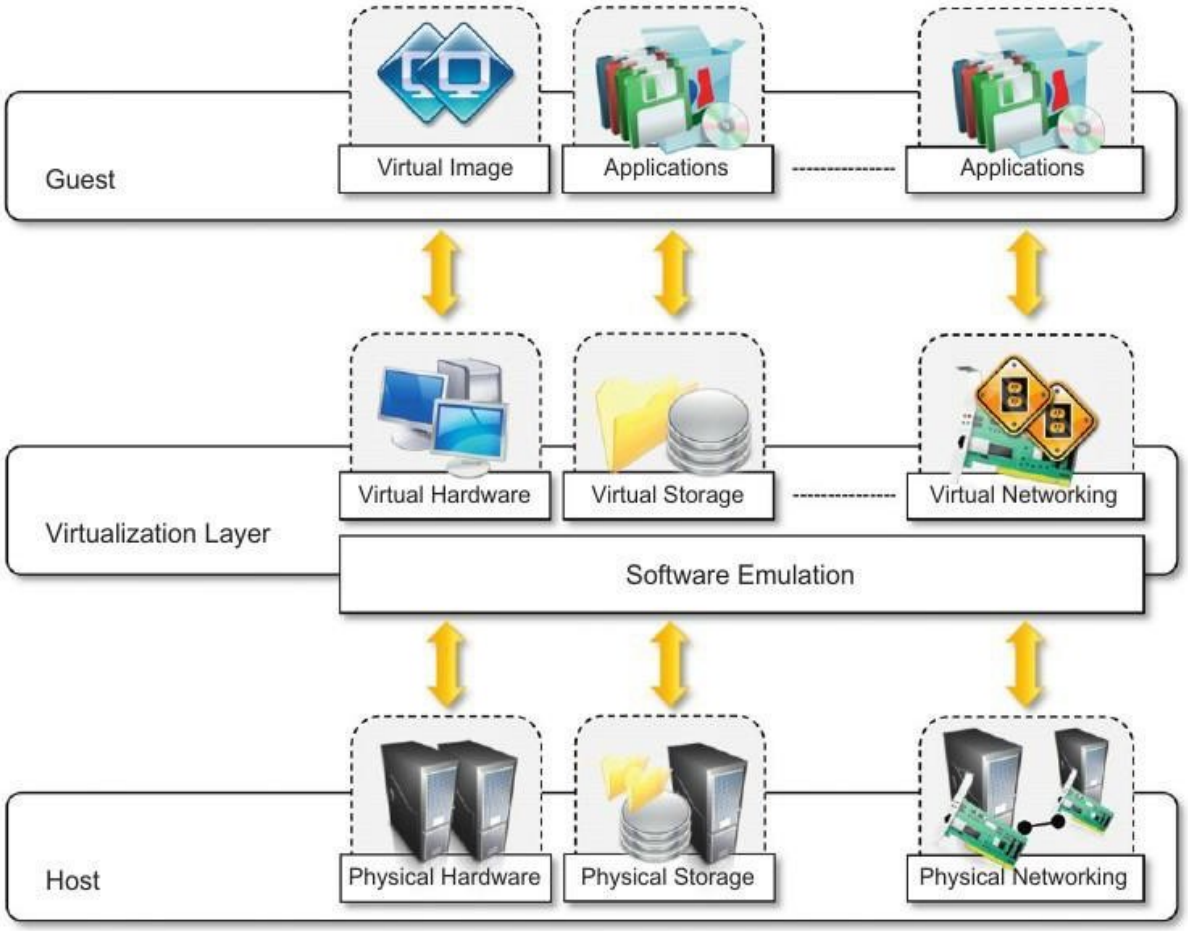
- Virtualization: extend or replace an existing interface to mimic the behavior of another system.
  - Introduced in 1970s: run legacy software on newer mainframe hardware
- Handle platform diversity by running apps in VMs
  - Portability and flexibility



Virtualized Environments

- Three major components of Virtualized Environments
  - **Guest** – system component that interacts with Virtualization Layer.
  - **Host** – original (physical) environment where guest runs.
  - **Virtualization Layer** – recreate the same or different environment where the guest will run.

Virtualization Reference Model

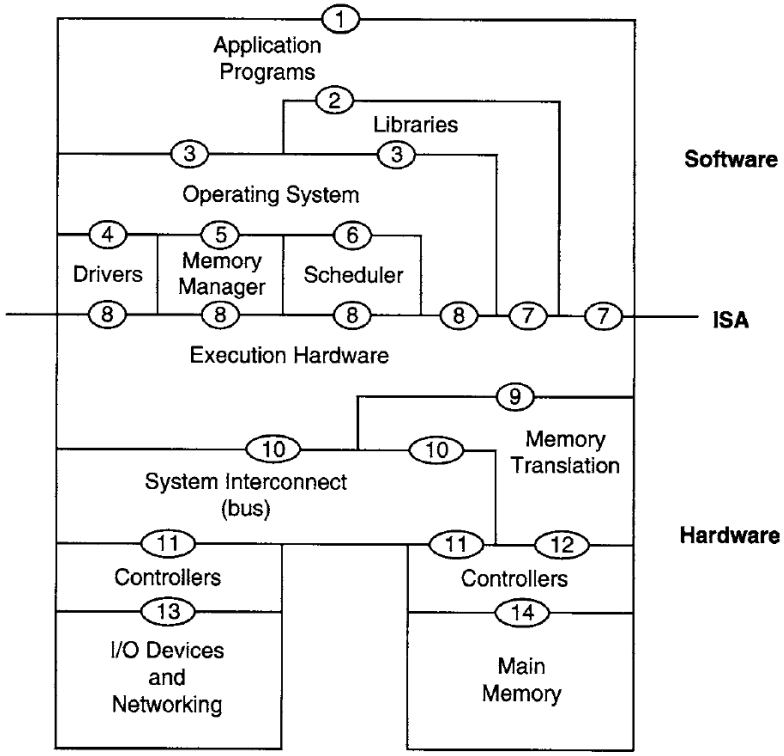


Machine reference model

- It defines the ***interfaces*** *between the levels* of abstractions, which ***hide implementation details***.
- Virtualization techniques actually *replace one of the layers* and **intercept the calls that are directed towards it**.
- A computing machine is defined by an interface.
  The following interfaces can be virtualized:
  - Instruction Set Architecture (ISA)
  - Application Binary Interface (ABI)
  - Application Programming Interface (API)

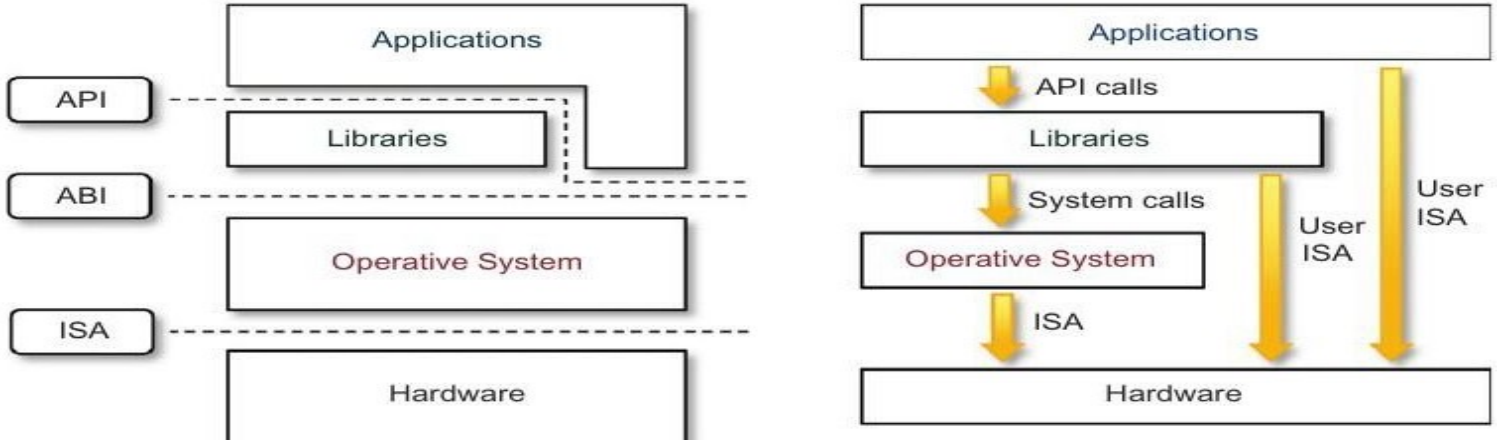
Computer system architecture

- There is an interface between an application program and standard libraries (interface 2).
- Another software interface is at the boundary of the operating system (interface 3).
- The interfaces in hardware include an I/O architecture that describes the signals that drive I/O device controllers (interface 11),
- a hardware memory architecture that describes the way addresses are translated (interface 9),
- an interface for the memory access signals that leave the processor (interface 12),
- another for the signals that reach the DRAM chips in memory (interface 14).
- The OS communicates with I/O devices through a sequence of interfaces: 4, 8, 10, 11, and 13.
- Of these interfaces and architectures, we are most interested in those at or near the hardware/software boundary.
- The instruction set architecture (ISA), which marks the division between hardware and software, is composed of interfaces 7 and 8



Smith & Nair, Virtual Machines, 2005 (fig.4)

Machine Reference Model

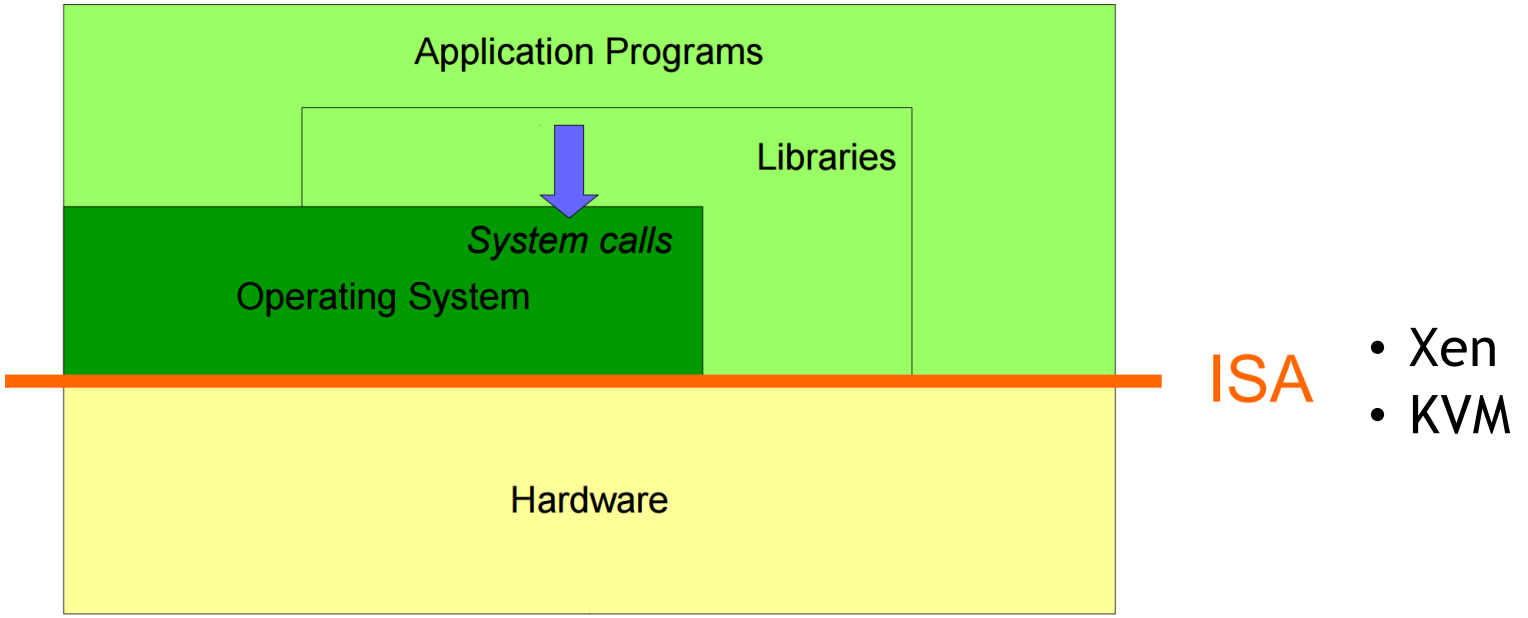


Hardware is expressed in terms of the ***Instruction Set Architecture(ISA).***

- *ISA for processor, registers, memory and the interrupt management.*

The Application Binary Interface (ABI) separates the OS layer from the application and libraries which are managed by the OS.

Interface 1: instruction set architecture ISA



- Virtualize a complete machine, running an OS supporting multiple processes
- = System VM

Machine reference model

The instructions in the ISA are divided into two security classes

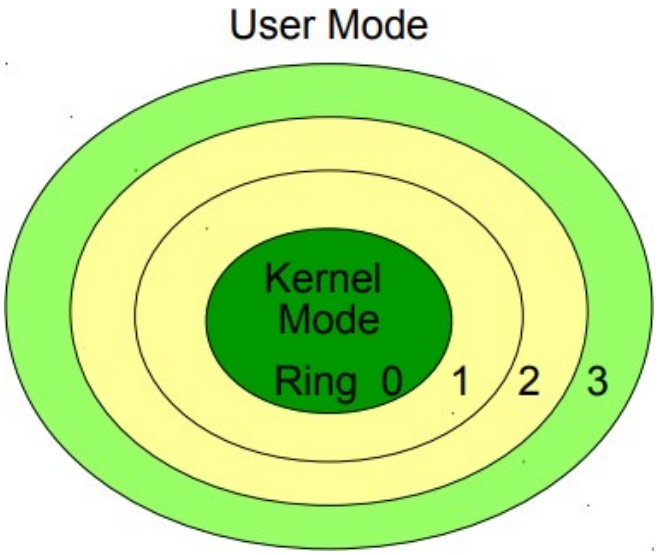
- **Privileged Instructions** (kernel mode)
- **Non privileged Instructions** (user mode)

ISA: Security Classes

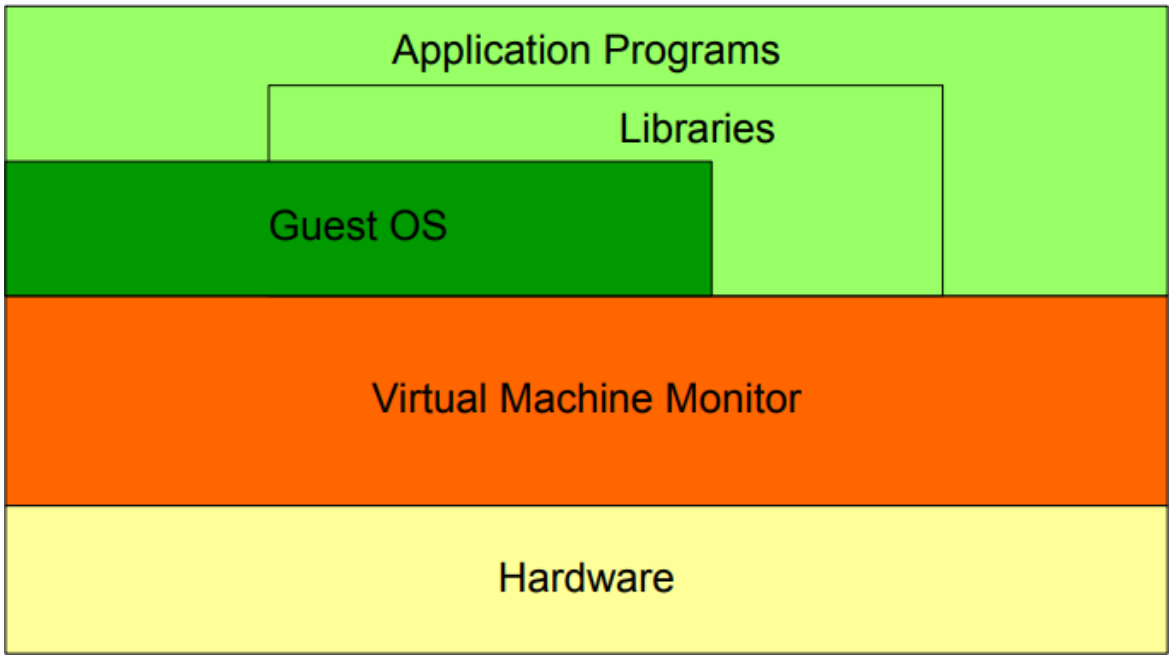
- **Non privileged instructions**
  - That can be used without interfering with other tasks because they ***do not access shared resources***.
- **Privileged instructions**
  - That are executed under ***specific restrictions*** and are mostly used for ***sensitive operations***, which expose (*behavior-sensitive*) or modify (*control- sensitive*) the privileged state.
    - **Behavior-sensitive** – operate on the I/O
    - **Control-sensitive** – alter the state of the CPU register.

System ISA

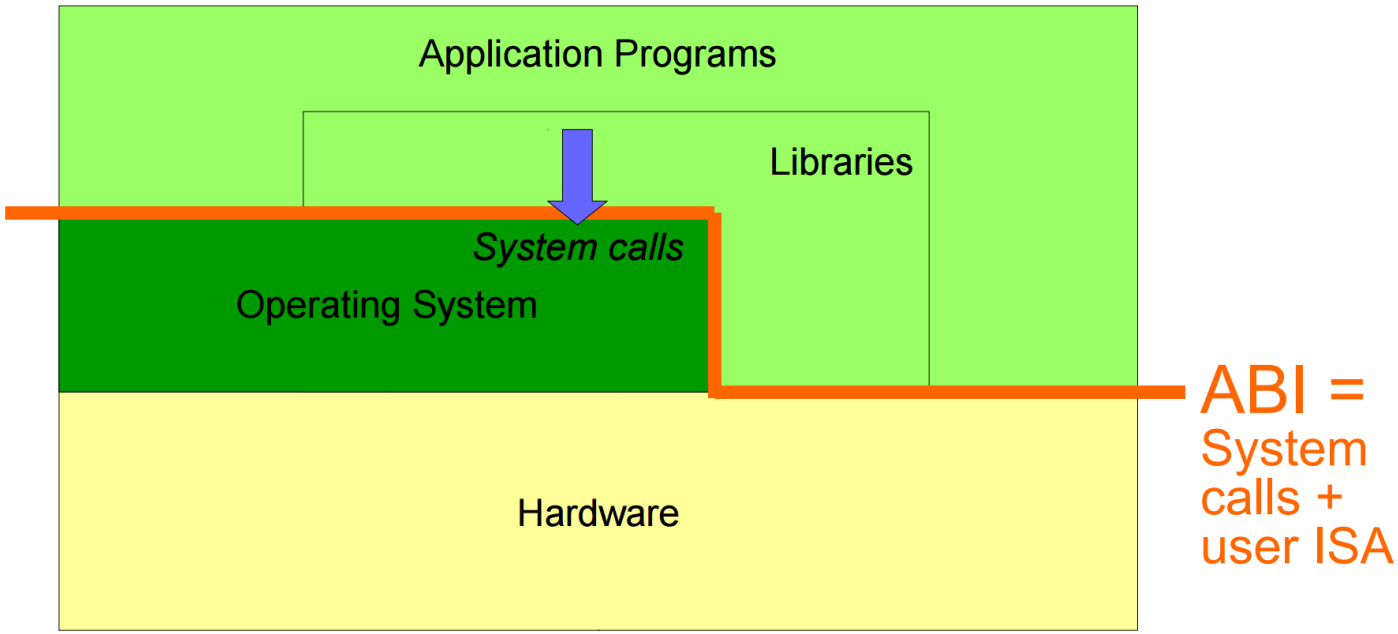
- OS requires special privileges over user programs
  - OS runs in CPU Kernel mode
  - Apps run in CPU User Mode
- x86: Implemented via 2 privilege levels / rings



System VM Implementation: Type 1 hypervisor



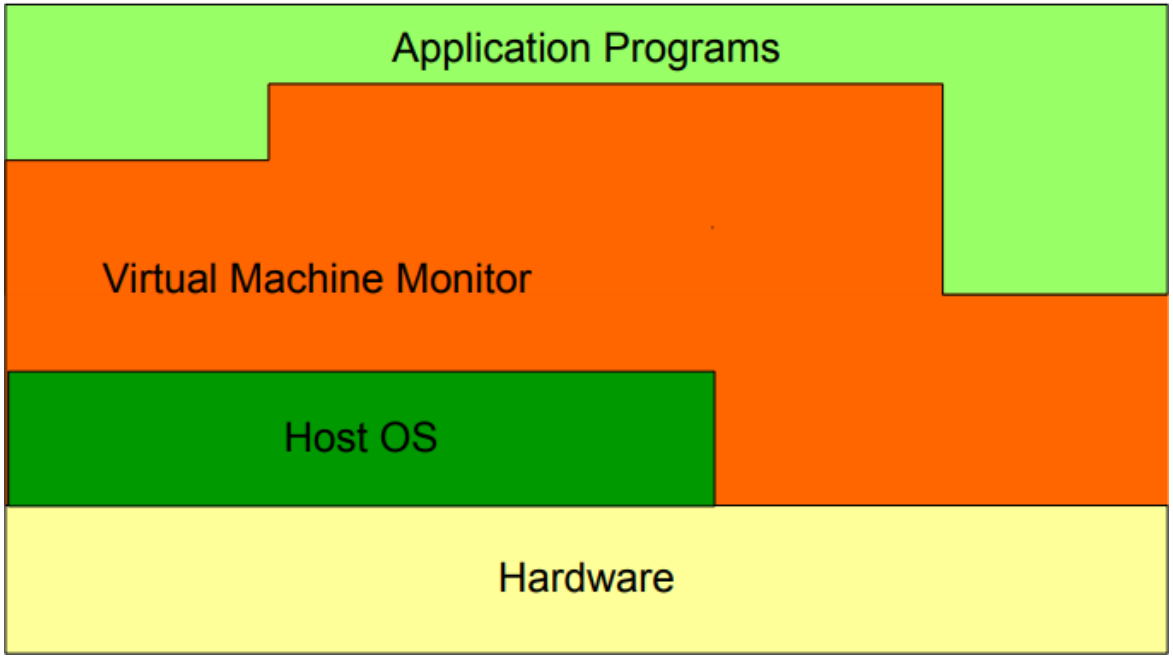
Interface 2: application binary interface ABI



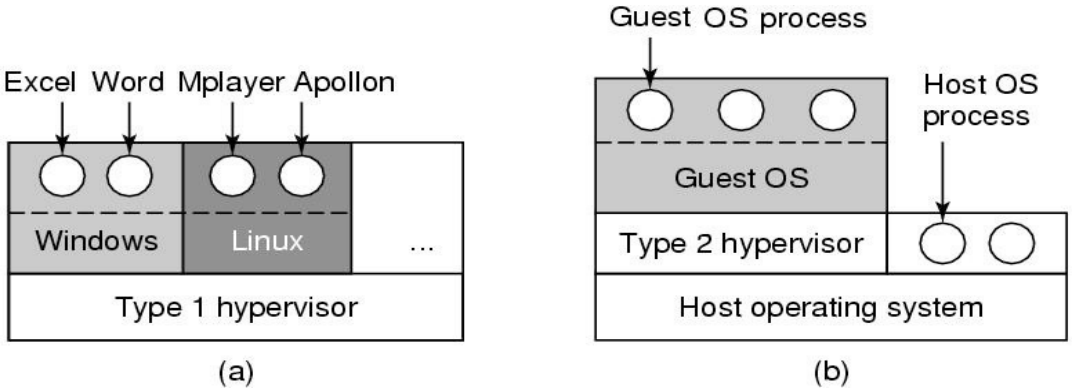
- Virtualize the environment of a single process = Process VM

An ABI defines how data structures or computational routines are accessed in machine code, which is a low-level, hw-dependent format. In contrast, an API defines this access in source code, which is a relatively high-level, hw-independent, often human-readable format.

Process VM Implementation: Type II hypervisor



Types of Hypervisors



- Hypervisor/VMM: virtualization layer
  - resource management, isolation, scheduling, …
- Type 1: hypervisor runs on “bare metal”
- Type 2: hypervisor runs on a host OS
  - Guest OS runs inside hypervisor
- Both VM types act like real hardware

How Virtualization works?

- CPU supports kernel and user mode (ring0, ring3)
  - Set of instructions that can only be executed in kernel mode
    - I/O, change MMU settings etc -- *sensitive instructions*
  - Privileged instructions: cause a trap when executed in user mode
- Result: type 1 virtualization feasible if sensitive instruction subset of privileged instructions
- Intel x86: ignores sensitive instructions in user mode
  - Can not support type 1 virtualization
- Recent Intel/AMD CPUs have hardware support
  - Intel VT, AMD SVM

Hypervisor (= Virtual Machine Monitor)

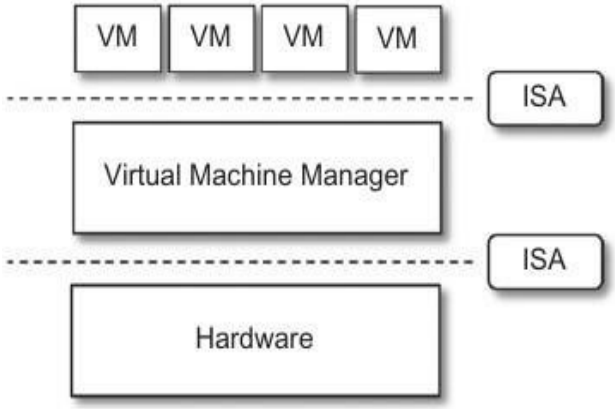
- The hypervisor runs at the supervisor mode.
- It is a piece of s/w that enables to run one

or more VMs on a physical server(host).

- Two major types of hypervisor: Type 1 and Type 2

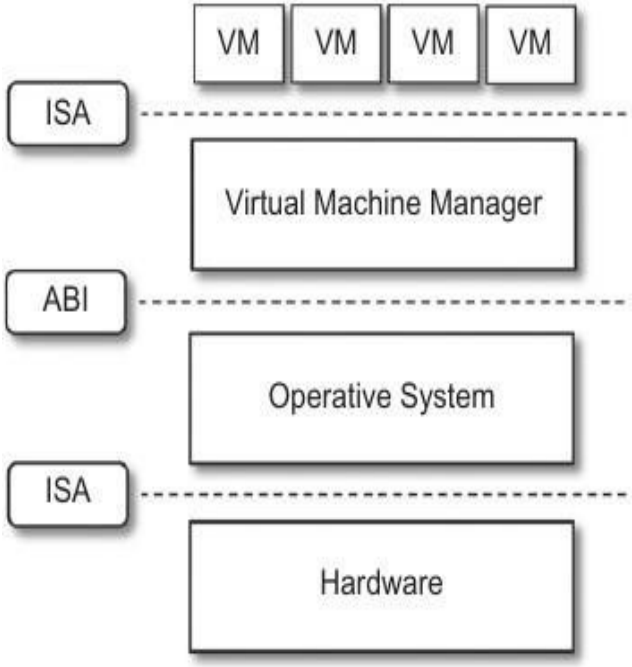
Type-I Hypervisor

- It runs directly on top of the hardware: it takes the place of the OS.
- Directly interact with the ISA exposed by the underlying hardware.
- Also known as *native virtual machine*.
- Examples: Citrix/Xen Server, VMware ESXi and Microsoft Hyper-V



Type-II Hypervisor

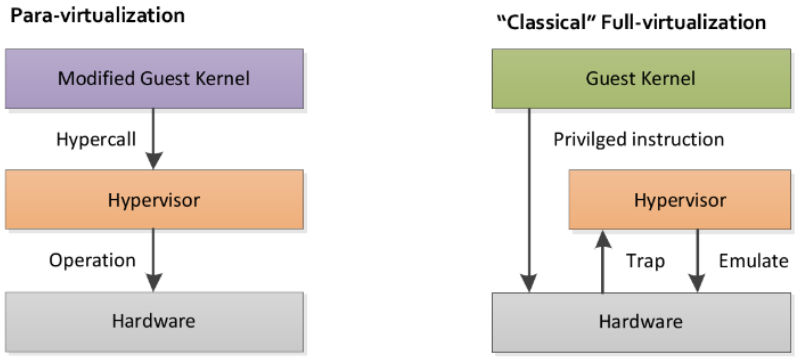
- Emulates the ISA of some virtual hw
- It requires the support of an OS to provide virtualization services.
- Programs are managed by the OS
- Type II hypervisor is also called *hosted virtual machine*.
- The hypervisor performs binary translation on the fly.



Type II hypervisors: examples

- Examples of type 2 Hypervisors include Microsoft Virtual PC, Oracle Virtual Box, VMware Workstation, Oracle Solaris Zones, VMware Fusion, Oracle VM Server for x86 and more.
- The origins of Type 2 Hypervisors go back to the days of x86 virtualization: this was a time when existing systems used pre-existing Operating Systems, while the hypervisor was deployed as an advanced software layer

Para virtualization



- Both type 1 and 2 hypervisors work on unmodified OS
- Type 2 was developed as a workaround for Type 1 hypervisor, that is needed to run on old hardware, which does not cause traps on sensitive instructions.
- Paravirtualization: modify OS kernel to replace all sensitive instructions with hypercalls
  - OS behaves like a user program making system calls

Paravirtualization

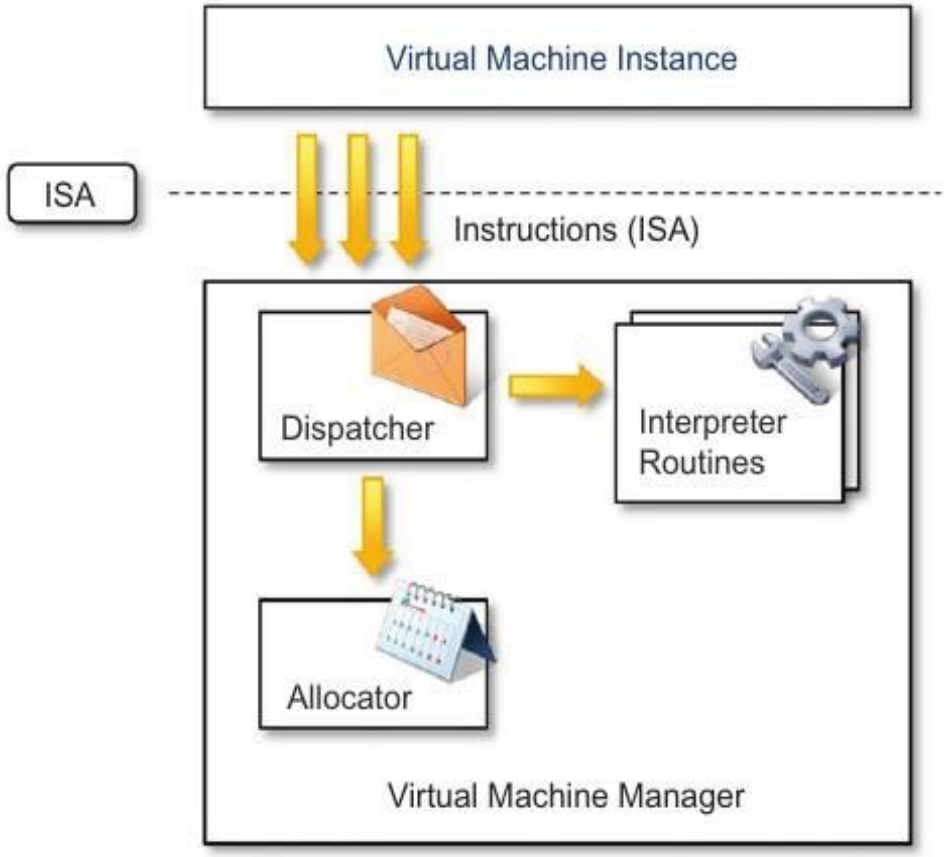
- The hypervisor executes the privileged operation invoked by hypercall
- Paravirtualization allows the OS to communicate directly with the hypervisor to conduct activities that would be time-consuming for the VM manager using hypercalls
- To support paravirtualization, the OS need to be modified to implement an API that lets to exchange hypercalls with the hypervisor.
- Paravirtualized hypervisors, like Xen, need OS support and special drivers, which are currently built into the Linux kernel

Virtual Machine Monitor (VMM)

Main modules:

- Dispatcher
  - Entry Point of VMM
  - Reroutes the instructions issued by VM instance
- Allocator
  - Deciding the system resources to be provided to the VM. Invoked by dispatcher
- Interpreter
  - Consists of interpreting routines
  - Executed whenever a VM executes a privileged instruction.
  - Trap is triggered and the corresponding routine is executed.

Virtual Machine Monitor (VMM)



Criteria* of VMM

- **Equivalence** – same behaviour as when it is e***xecuted directly*** on the physical host.
- **Resource control** – it should be in ***complete control of virtualized resources.***
- **Efficiency** – a statistically dominant fraction of the machine instructions should be ***executed without intervention*** from the VMM

** criteria are established by Goldberg and Popek in 1974*

Advantages of Virtualization

- **Increased Security**

– Ability to control the execution of a guest

– Guest is executed in emulated environment.

– Virtual Machine Manager control and filter the activity of the guest.

- Hidding of resources.
- Having no effect on other users/guest environment.

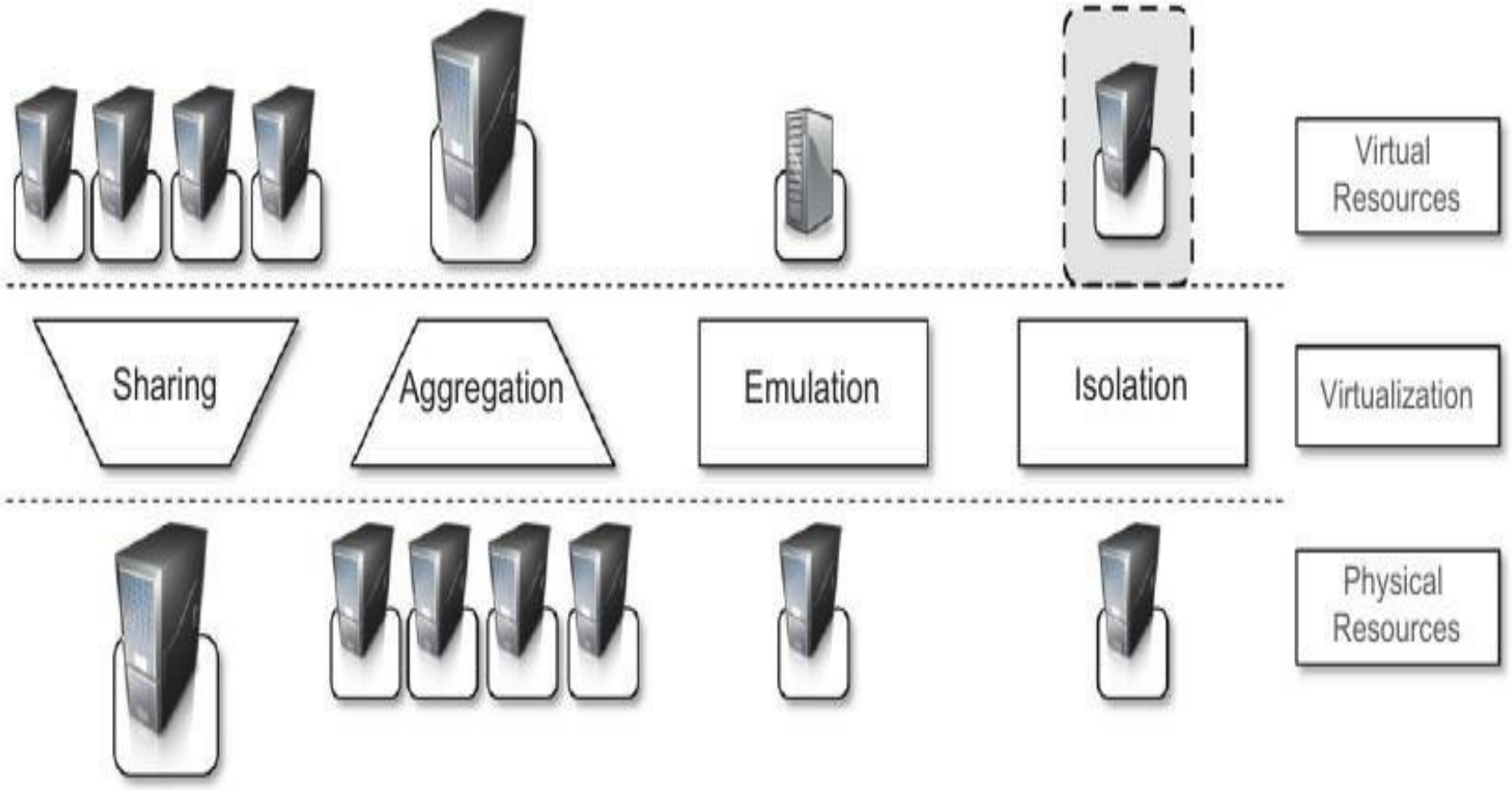
Advantages of Virtualization

- **PerformanceTuning**
  - control the performance of guest.
- **Virtual Machine Migration**
  - move virtual image into another machine.
- **Portability**
  - safely moved and executed on top of different virtual machine.
  - Availability of system is with you.

Advantages of Virtualization

- **Managed Execution types**
  - **Sharing**
    - Creating separate computing environment within the same host.
    - Underline host is fully utilized.
  - **Aggregation**
    - A group of separate hosts can be tied together and represented as single virtual host.
  - **Emulation**
    - Controlling & Tuning the environment exposed to guest.
  - **Isolation**
    - Complete separate environment for guests.

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

- virtualization also allows aggregation, which is the opposite process of sharing
- 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 guest sw. 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 is 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.

Emulation (continues)

- hardware virtualization solutions are able to provide virtual hardware and emulate a particular kind of device such as Small Computer System Interface (SCSI) devices for file I/O, without the hosting machine having such hardware installed.
- Old and legacy software that does not meet the requirements of current systems can be run on emulated hardware without any need to change the code: this is possible either by emulating the required hardware architecture or within a specific operating system sandbox, such as the MS-DOS mode in Windows 95/98.
- Another example of emulation is an arcade-game emulator that allows us to play arcade games on a normal personal computer.

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

Performance tuning

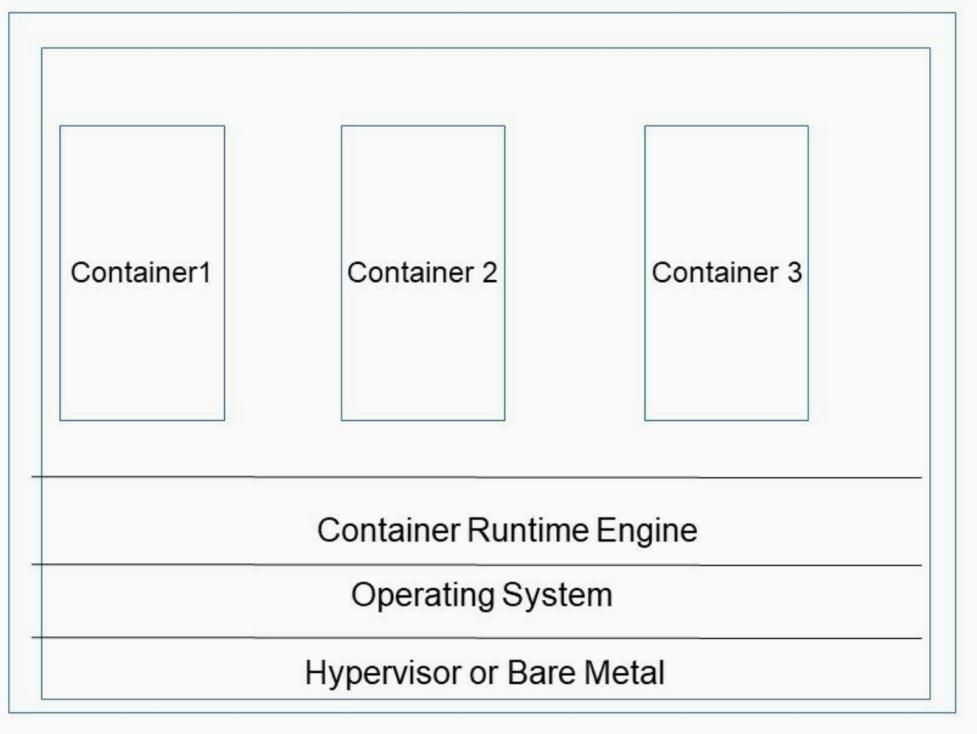
- It is easy to control the performance of the guest sw by tuning the properties of the resources exposed through the virtual environment.
- This capability provides a means to effectively implement a quality-of-service (QoS) infrastructure that fulfills the service-level agreement (SLA) established for the guest. For instance, software-implementing hardware virtualization solutions can expose to a guest operating system only a fraction of the memory of the host machine or set the maximum frequency of the processor of the virtual machine.
- Another advantage of managed execution is that it can allow capturing of the state of the guest program, persisting it, and resuming its execution.
- for example, virtual machine managers such as Xen Hypervisor can stop the execution of a guest operating system, move its virtual image into another machine, and resume its execution - in a completely transparent manner.
- This technique is called **virtual machine migration** and constitutes an important feature in virtualized data centers for optimizing their efficiency in serving application demands

Moving VMs is expensive

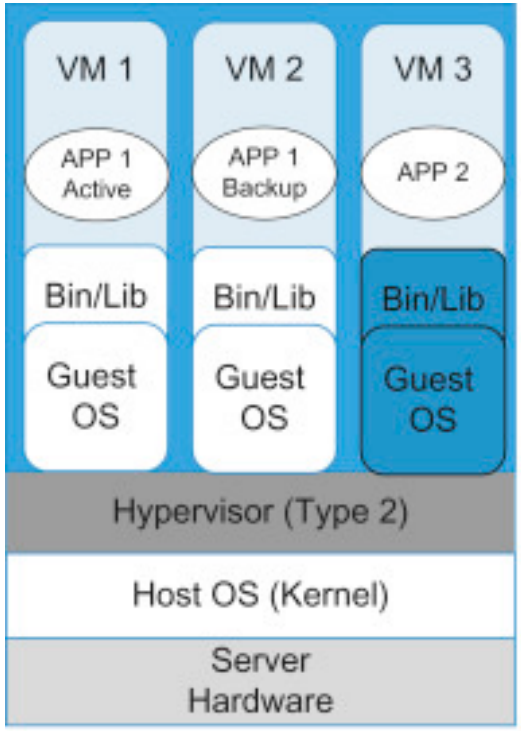
- VMs solve the problem of sharing resources and maintaining isolation.
- However, VM images can be large, and transferring VM images around the network is time consuming.
- Suppose you have an 8 GB(yte) VM image. You wish to move this from one location on the network to another. In theory, on a 1 Gb(it) per second network, this will take 64 seconds. However, in practice a 1 Gbps network operates at around 35% efficiency. Thus, transferring an 8 GB VM image will take around 3 minutes.
- After the image is transferred, the VM must boot the operating system and start its services, which takes still more time

Containers

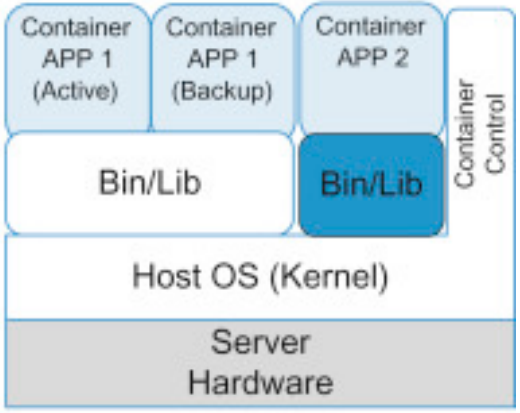
- Containers are a mechanism to maintain the advantages of virtualization while reducing image transfer time and startup time.
- In the Figure we see some containers operating under the control of a container engine which, in turn, is running on top of an operating system.
- Just as all VMs on a physical host share the same underlying physical hardware, all containers share the same operating system kernel (and through the operating system, they share the same underlying physical hardware).
- The operating system can be loaded either onto a bare-metal physical machine or a virtual machine



Comparing VMs and containers



(A) VMs



(B) Containers

- A container is an OS kernel abstraction in which the applications share the same kernel (and potentially the bins/services and libs—a sandbox)
- the VM is a hardware abstraction in which the applications run separate kernels

Conclusions

- Instruction set emulation is an important enabling technology for many virtual machine implementations.
- Emulation encompasses both interpretation, where guest instructions are emulated one by one in a simple fashion, and binary translation, where blocks of guest instructions are translated to the ISA of the host platform and saved for multiple executions.
- In many VM implementations the need to use emulation is obvious (i.e., the guest and host ISAs are different), but the same techniques are important in other VMs as well

Conclusions

- Hardware virtualization allows the creation of multiple VM sharing the same physical machine.
- It does this while enforcing isolation of CPU, memory, disk storage, and network of each VM.
- This allows the resources of the physical machine to be shared among several VMs and reduces the number of physical machines that an organization must purchase or rent.

Conclusions

- A VM instance is a virtual machine hosted by some hw infrastructure, eg. Cloud (https://cloud.google.com/compute/docs/instances)
- A VM image is the set of bits that are loaded into a VM instance to enable its execution
- VM images are created by various techniques for provisioning including using operating system functions or loading a pre-created image.
- Containers are virtualized operating systems and provide performance advantages over VMs: containers constructed in terms of layers are faster to deploy when a component changes.

Discussions (send me your answers)

- Where do you find the Ubuntu distribution? What are the steps that take a Linux kernel distribution and creates an Ubuntu distribution? Who provides the funding for the creation of the Ubuntu distribution?
- Enumerate different package managers used by the different distributions of Linux. What are the differences between them?
- Two VMs hosted on the same physical computer are isolated, but it is still possible for one VM to affect the other VM. How can this happen?
- We've focused on isolation among VMs that are running at the same time on a hypervisor. VMs shut down and stop executing, and new VMs start up. What does a hypervisor do to maintain isolation, or prevent leakage, between virtual machines?

References

- Bass & Klein, *Deployment and Operations for Software Engineers*, chapter 1
- Smith & Nair, *Virtual Machines*, Morgan Kaufmann, 2005