Virtual Machines and Virtualization of Clusters and Data Centers

What is Virtualization?

- The Software methods (including hypervisors)used to allow <u>multiple virtual computing resources to run on a single hardware platform.</u>

What is a VM?

- A <u>Software Virtualization of a Computer Hardware that executes program like a physical computer</u> and can be interacted with like a physical computer.

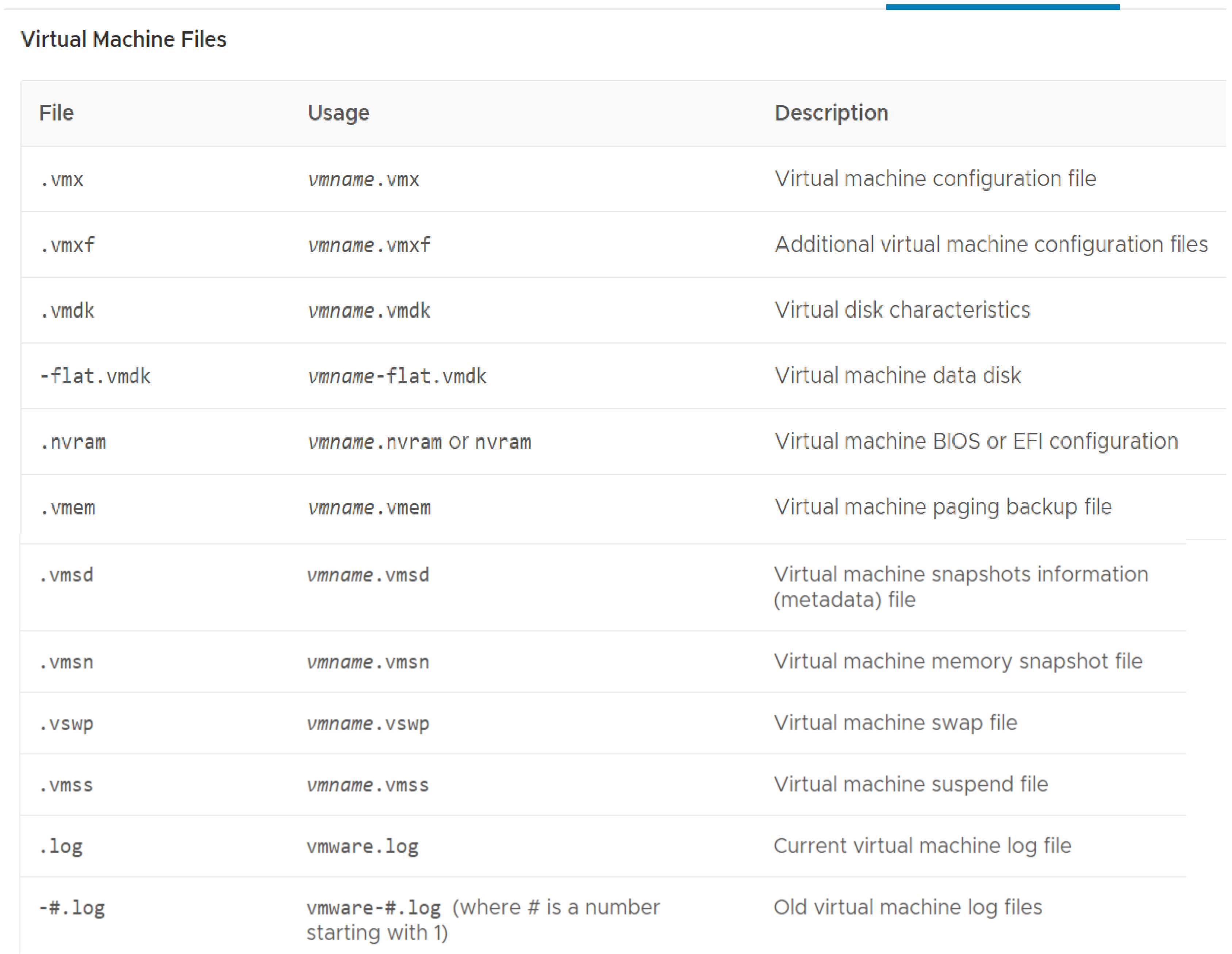
What is VM?

- Allows to run Multiple Operating Systems as VMs on a single computer

- A VM is a discrete set of files.

Main Files:

- Configuration file
- Virtual disk file
- NVRAM settings file
- Log file



Virtual Machine Files

File	Usage	Description
`.vmx`	*`vmname`*`.vmx`	Virtual machine configuration file
`.vmxf`	*`vmname`*`.vmxf`	Additional virtual machine configuration files
`.vmdk`	*`vmname`*`.vmdk`	Virtual disk characteristics
`-flat.vmdk`	*`vmname`*`-flat.vmdk`	Virtual machine data disk
`.nvram`	*`vmname`*`.nvram` or `nvram`	Virtual machine BIOS or EFI configuration
`.vmem`	*`vmname`*`.vmem`	Virtual machine paging backup file
`.vmsd`	*`vmname`*`.vmsd`	Virtual machine snapshots information (metadata) file
`.vmsn`	*`vmname`*`.vmsn`	Virtual machine memory snapshot file
`.vswp`	*`vmname`*`.vswp`	Virtual machine swap file
`.vmss`	*`vmname`*`.vmss`	Virtual machine suspend file
`.log`	`vmware.log`	Current virtual machine log file
`-#.log`	`vmware-#.log` (where # is a number starting with 1)	Old virtual machine log files

vmx:

- The .vmx file describes the current configuration information and hardware settings for the VM.
- This can contain a large variety of information regarding the virtual machine, to include its specific virtual hardware configuration (**amount of RAM, NIC settings, CD-ROM information, parallel/serial port information, and so on), as well as its advanced resource and power settings,**

vmx:

- The .vmx file is a plain-text file that functions as the structural definition of the VM.
- The .vmx file can be copied from the datastore and opened using a program that supports creation and saving of files using UTF-8 encoding, such as WordPad.

vmx:

```
.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "10"
nvram = "ExampleVM.nvram"
pciBridge0.present = "TRUE"
svga.present = "TRUE"
pciBridge4.present = "TRUE"
pciBridge4.virtualDev = "pcieRootPort"
pciBridge4.functions = "8"
pciBridge5.present = "TRUE"
pciBridge5.virtualDev = "pcieRootPort"
pciBridge5.functions = "8"
pciBridge6.present = "TRUE"
pciBridge6.virtualDev = "pcieRootPort"
pciBridge6.functions = "8"
pciBridge7.present = "TRUE"
pciBridge7.virtualDev = "pcieRootPort"
pciBridge7.functions = "8"
vmci0.present = "TRUE"
hpet0.present = "TRUE"
```

.vmtx:

- When a virtual machine is converted to a template, the virtual machine configuration file (.vmx) is replaced by the template configuration file (.vmtx).

.nvram:

- This is generally a fairly small file that contains the BIOS settings that the VM uses upon boot.
- This is similar to how a physical server that has a BIOS chip allows hardware configuration options.
- The virtual BIOS settings, contained in the .nvram file, can be accessed by pressing *F2* when the virtual machine is powered on.

- Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.

- The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility.
- Hardware resources (CPU, memory, I/O devices, etc.) or
- software resources (operating system and software libraries) can be virtualized in various functional layers.

Benefit of Virtualization

- Let us assume a system having Multicore Processor with 20 cores

Without Virtualization-

- 20 cores dedicated to one application

With virtualization =

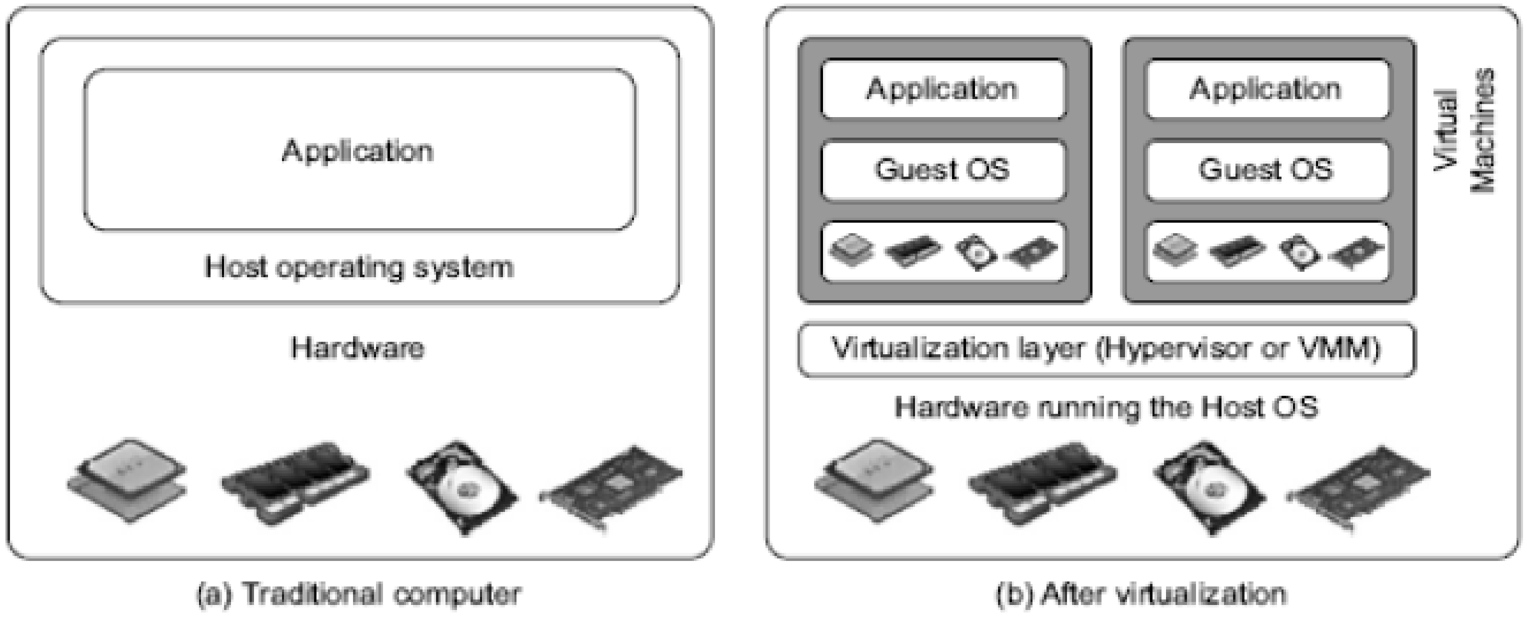
- 3 cores to Application A1
- 3 cores to Application A2
- 4 cores Application A3
- Remaining 10 cores for Host Machine

- Resources are shared
- Effective Utilization of Resources

- Virtualization=Software defined technology
- Software defined compute
- Software defined networking
- Software defined storage

4.1) Virtual Machines

Difference between Traditional Computer and Virtual machines



Virtual Machine, Guest Operating System, and VMM (Virtual Machine Monitor) :

VM-

A representation of a real machine using software that provides an operating environment that can run a guest OS

Guest OS-

An operating system running on a VM that would otherwise run on directly on a separate physical system.

VMM-
The Virtualization Layer is the middleware between the VM and the underlying Host platform , also called as Virtual Machine Monitor or VMM.

Virtual Machine Architecture

Three VM Architecture (b),(c),(d) compared with the traditional physical machine in (a)

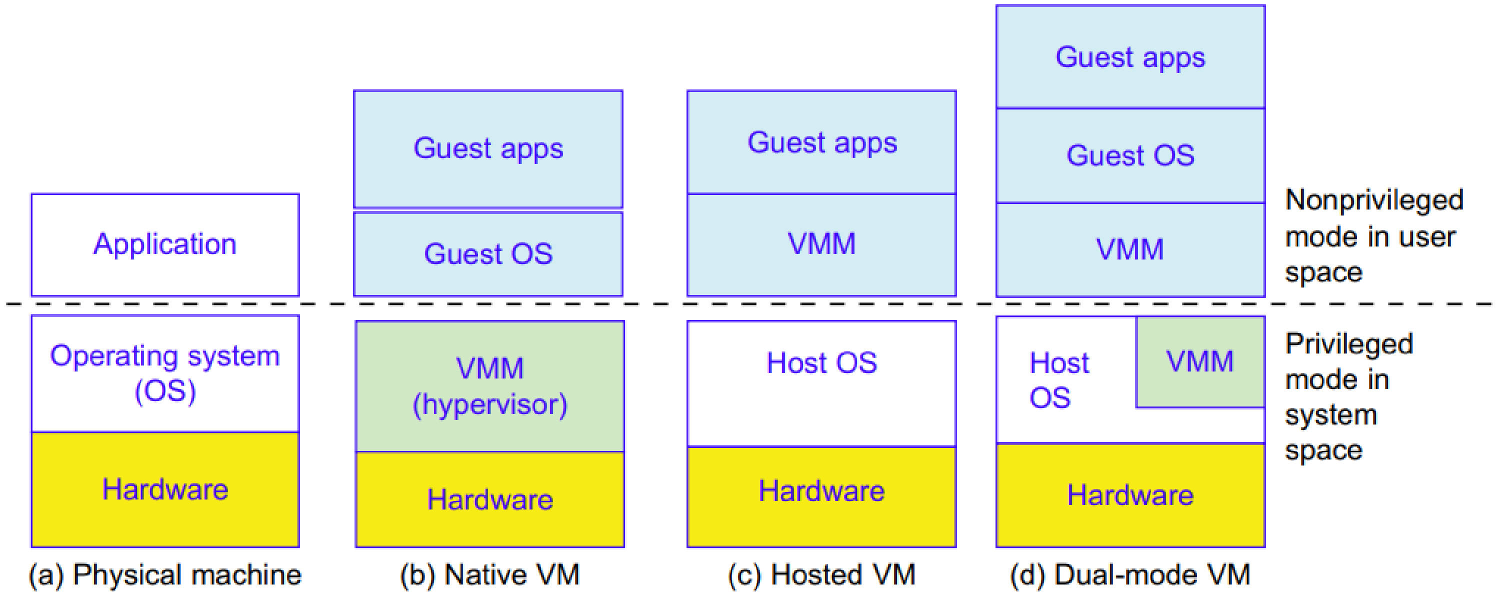


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

(*Courtesy of M. Abde-Majeed and S. Kulkarni, 2009 USC*) a)Physical Machine

- A host machine is equipped with the physical hardware
- Eg- x-86 architecture desktop running its installed Windows OS

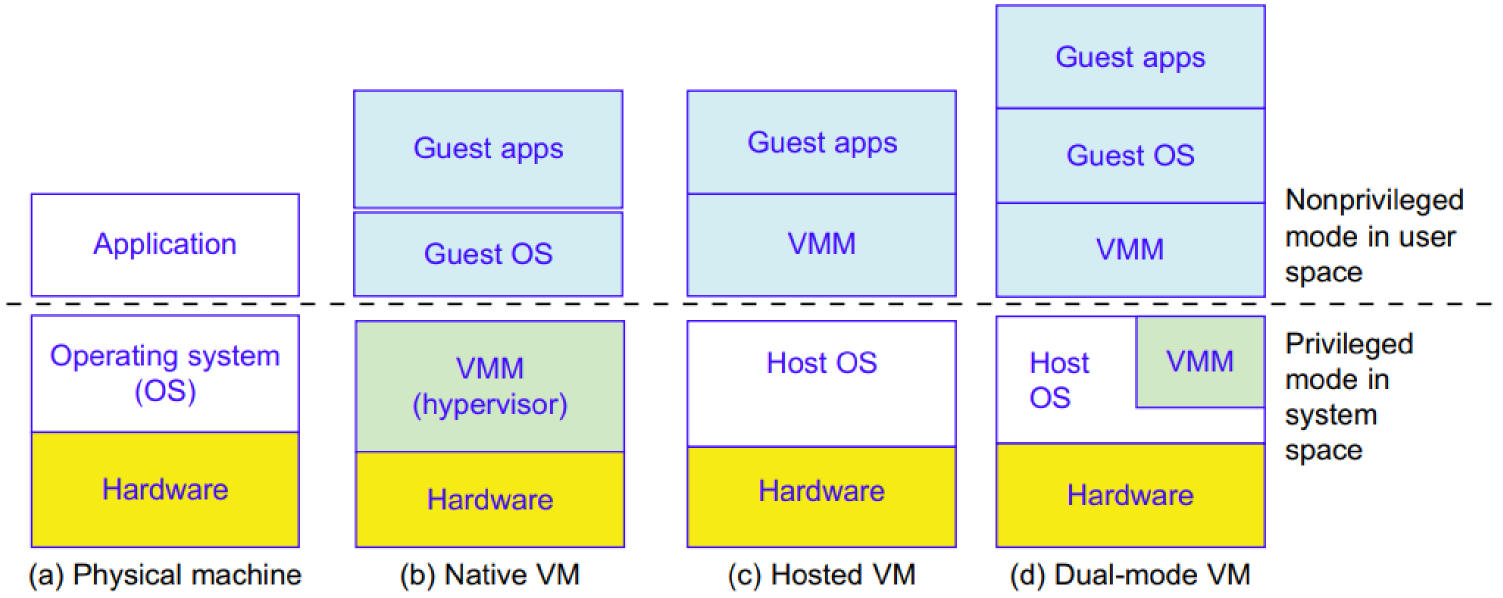


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

(Courtesy of M. Abde-Majeed and S. Kulkarni, 2009 USC)

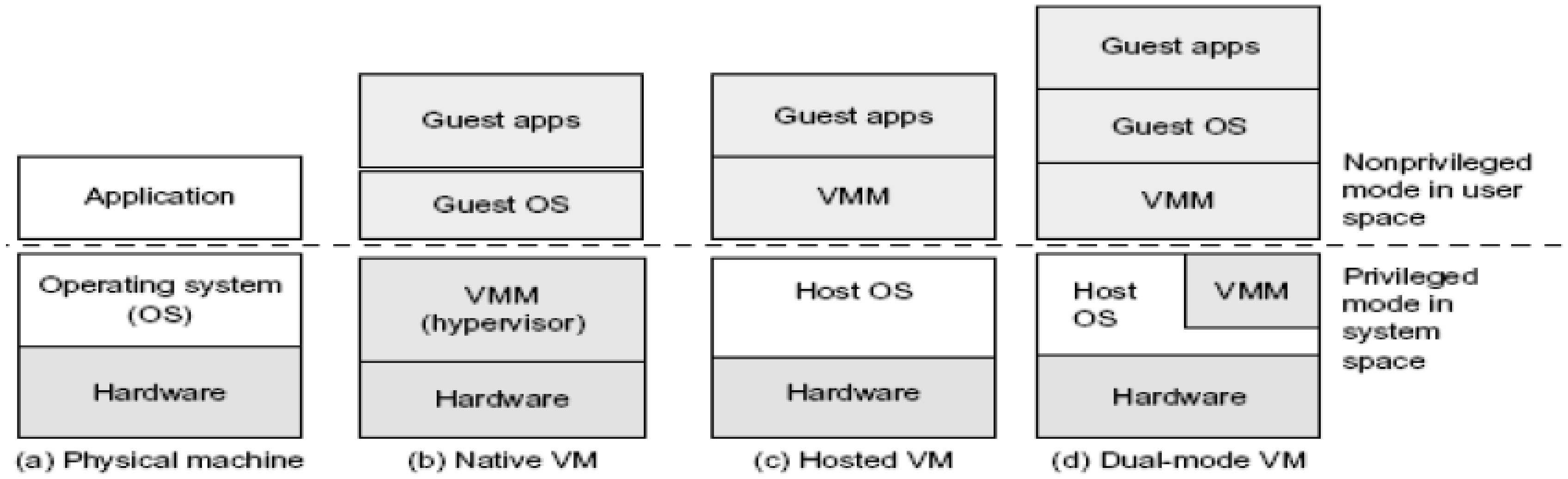
(b) Native VM approach

A native VM installed with the use of a VMM called a hypervisor in privileged mode

This Hypervisor handles the bare hardware(CPU, memory, I/O) directly, hence called as Bare Metal Hypervisor or Native VM

Eg-

The guest OS could be a Linux system and the hypervisor the XEN system developed at Cambridge University.



(c)Hosted VM Approach

- The Host OS needs not to be modified.
- Here the VMM runs in nonprivileged mode.

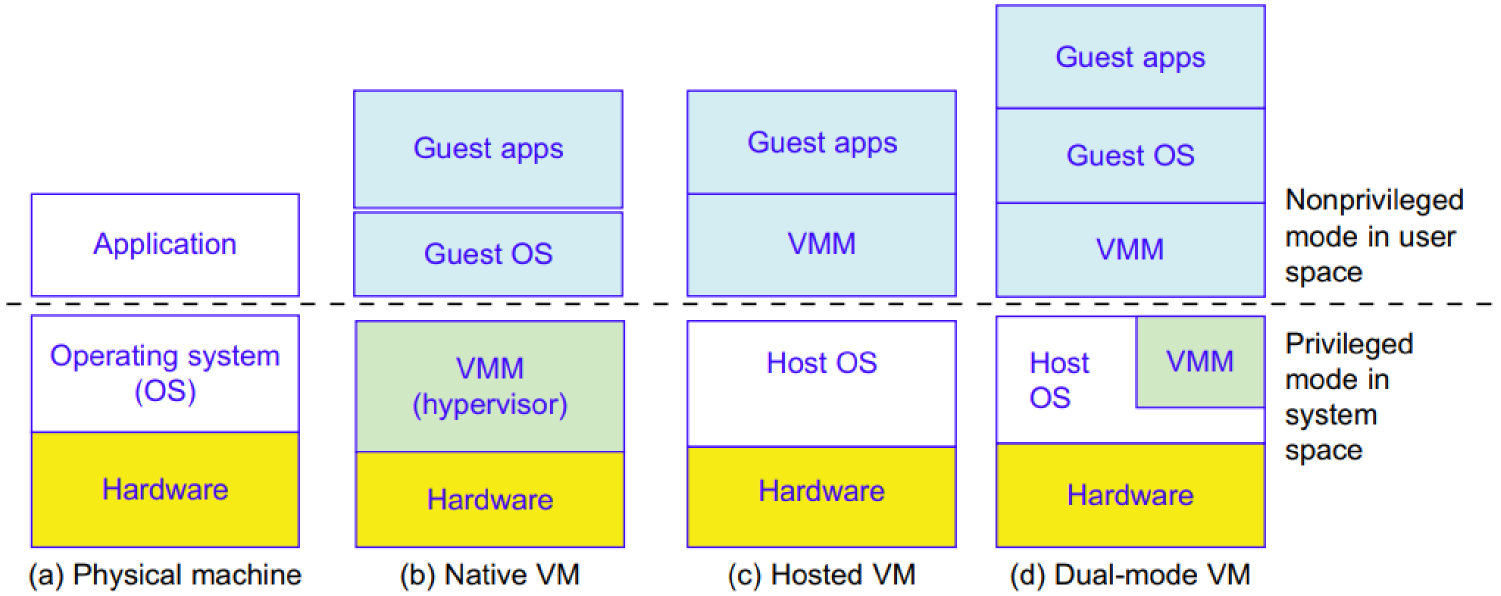


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

(Courtesy of M. Abde-Majeed and S. Kulkarni, 2009 USC)

(d)Dual Mode VM Approach

- Part of the VMM runs at the user level and another part runs at the supervisor level.
- In this case, the host OS may have to be modified to some extent.

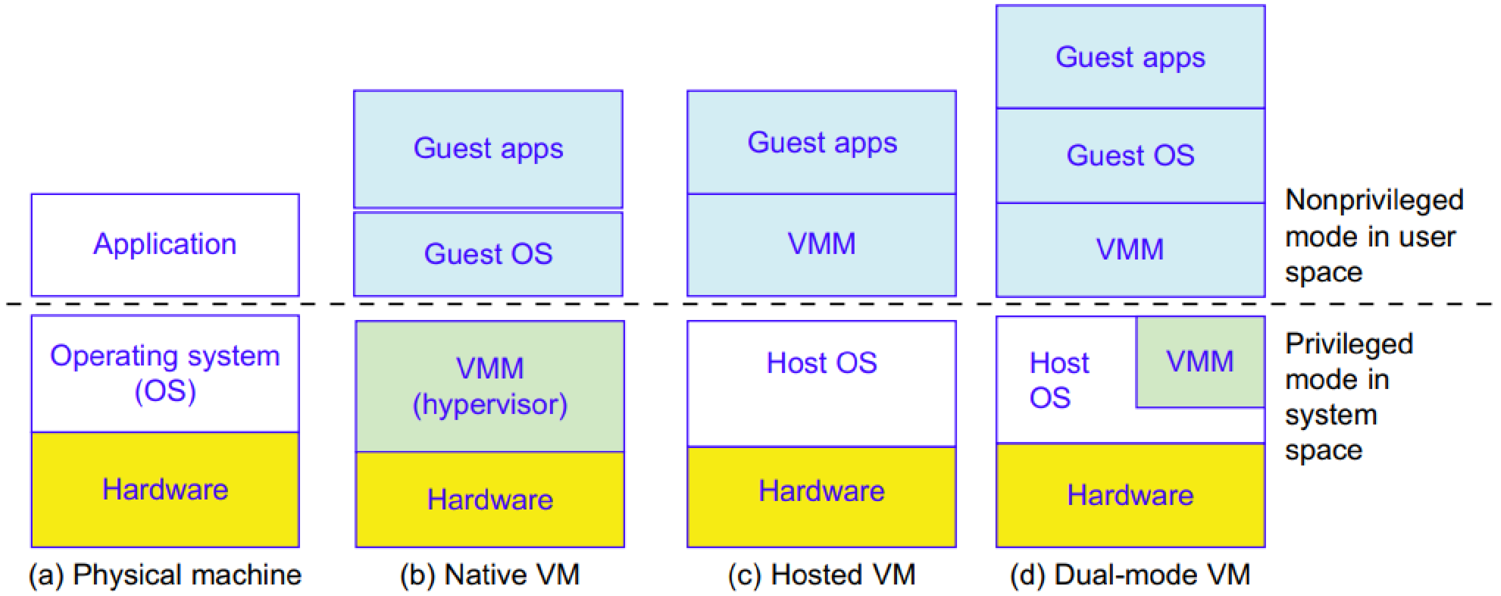
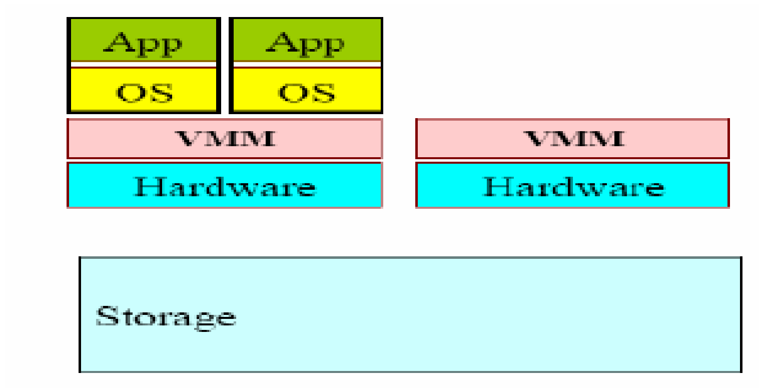


FIGURE 1.12

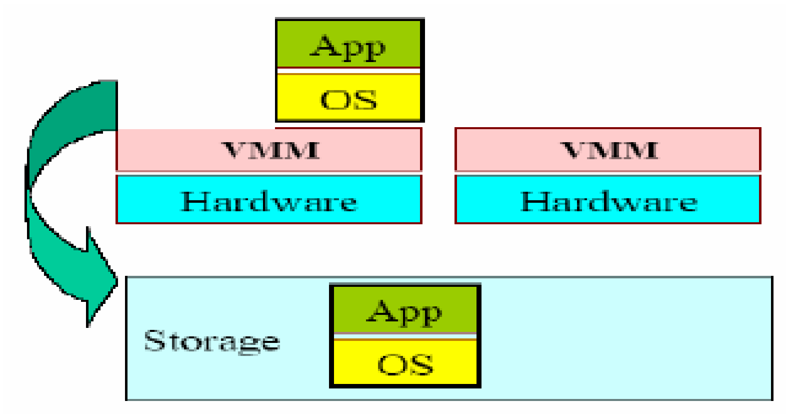
Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

(*Courtesy of M. Abde-Majeed and S. Kulkarni, 2009 USC*)

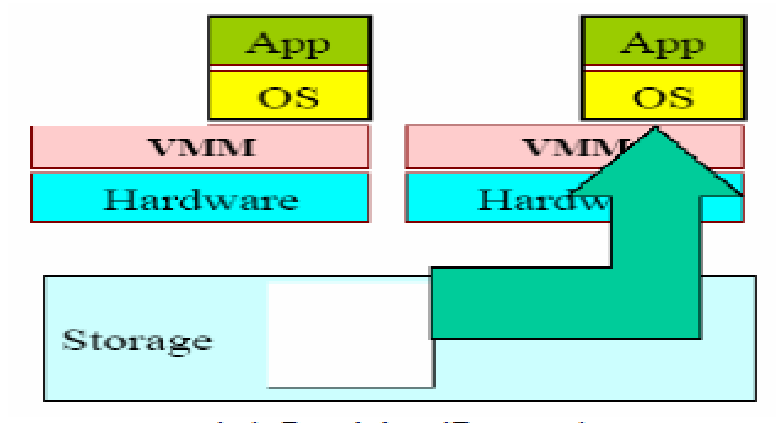
Primitive Operations in Virtual Machines:



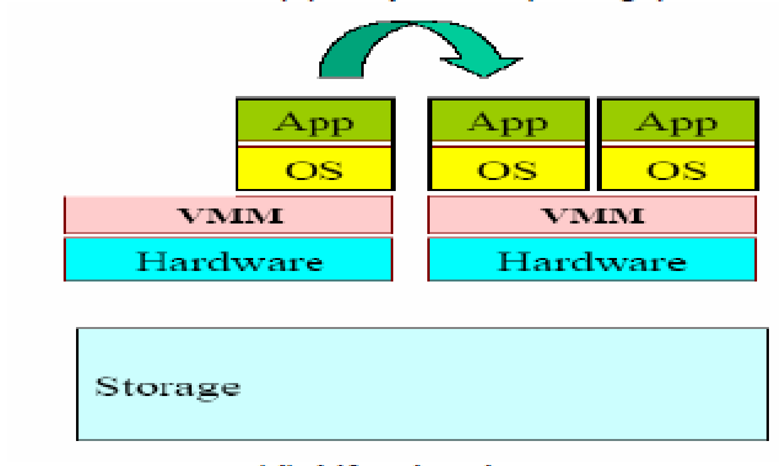
(a) Multiplexing



(b) Suspension (Storage)



(c) Provision (Resume)



(d) Life migration

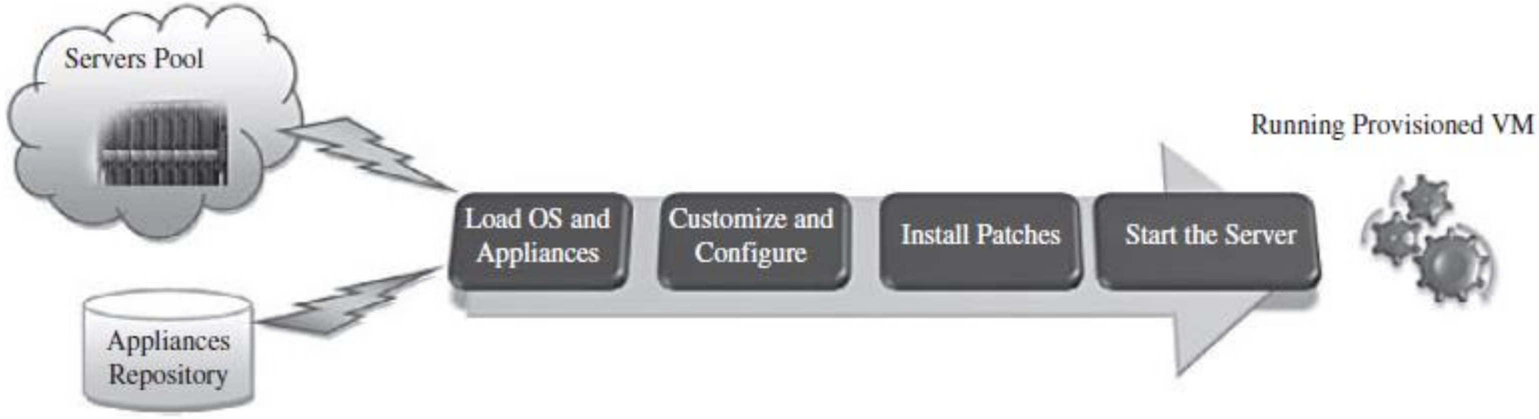
Primitive Operations in Virtual Machines:

- VMs can be multiplexed b/w hardware m/c
- A VM can be suspended and stored in stable storage
- A suspended VM can be resumed or provisioned to a new hardware platform.
  - You suspend a virtual machine when you want to save its current state.
  - When you resume the virtual machine, applications that were running before the virtual machine was suspended resume in their running state and their content is unchanged.
- A VM can be migrated from one hardware platform to another.

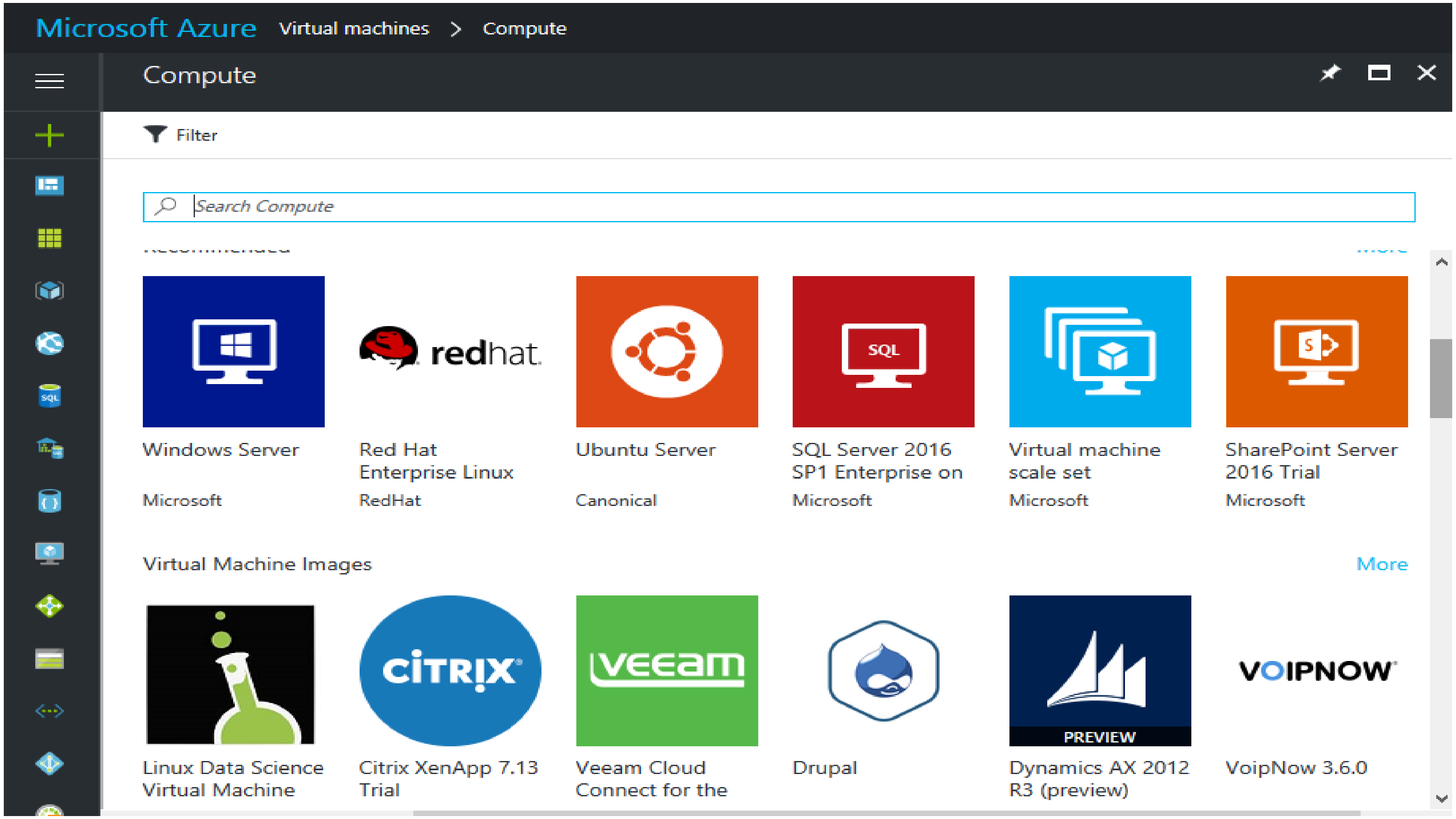
Provisioning VM

The steps of provisioning a Virtual Machine are:

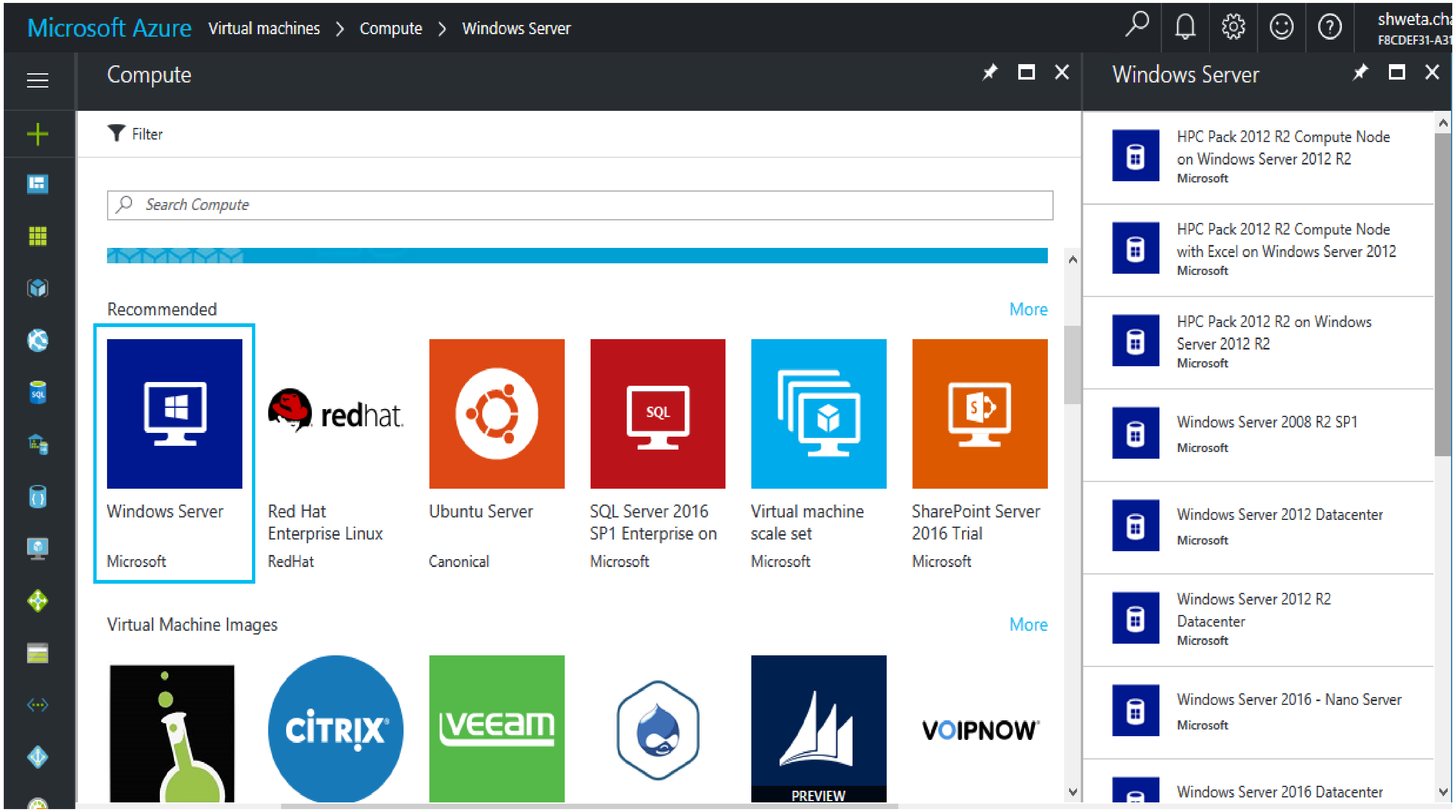
- Firstly, you need to select a server from a pool of available servers along with appropriate OS template you need to provision the VM
- Secondly, you need to load the appropriate software(OS you selected in the previous step, device drivers, middleware and the needed applications for the service required).



Eg-Selecting a server in Microsoft Azure



Eg-Then Selecting an OS in Microsoft Azure



Recommended

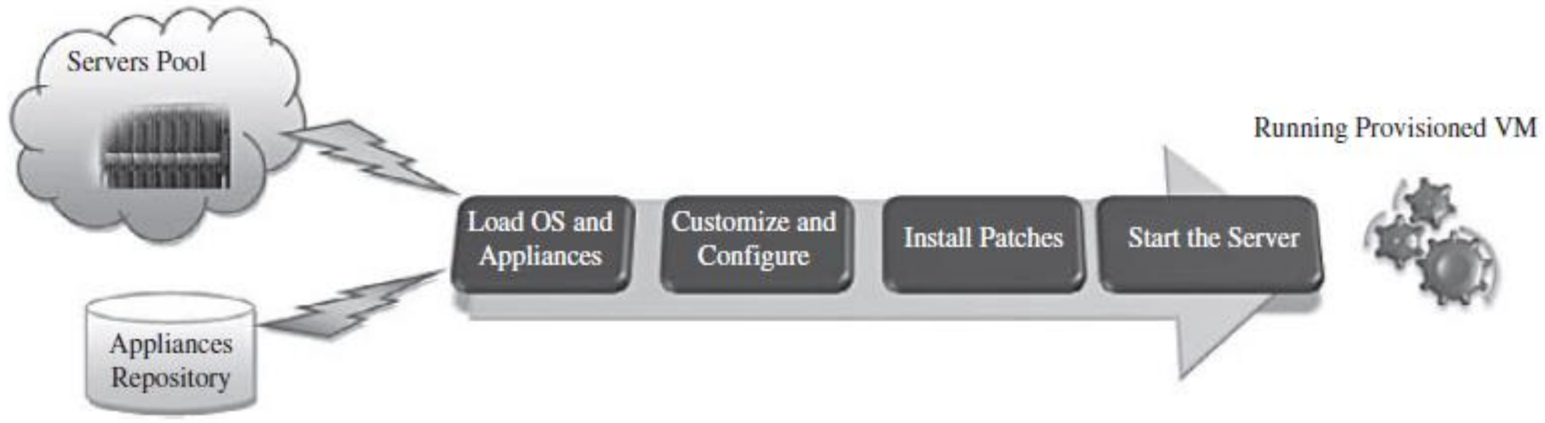
Windows Server	Microsoft
Red Hat Enterprise Linux	RedHat
Ubuntu Server	Canonical
SQL Server 2016 SP1 Enterprise on	Microsoft
Virtual machine scale set	Microsoft
SharePoint Server 2016 Trial	Microsoft

Windows Server

- HPC Pack 2012 R2 Compute Node on Windows Server 2012 R2 — Microsoft
- HPC Pack 2012 R2 Compute Node with Excel on Windows Server 2012 — Microsoft
- HPC Pack 2012 R2 on Windows Server 2012 R2 — Microsoft
- Windows Server 2008 R2 SP1 — Microsoft
- Windows Server 2012 Datacenter — Microsoft
- Windows Server 2012 R2 Datacenter — Microsoft
- Windows Server 2016 - Nano Server — Microsoft
- Windows Server 2016 Datacenter

Provisioning VM

- Thirdly, you need to <u>customize and configure the M/c(eg:IP address, Gateway) to configure an associated n/w and storage resources.</u>
- Finally, the VM is ready to start with its newly loaded software.



Live Migration

- Hot Migration
- Real-Time Migration
- Movement of a virtual m/c from one physical host to another while being powered on.

- Transparently move running Virtual Machines from one host to another without perceived downtime.

- During a planned maintenance event on a virtual machine (VM) instance's underlying hardware, Compute Engine might move the VM to another host.
- To keep a VM running during a host event, Compute Engine performs a *live migration* of the VM to another host in the same zone.

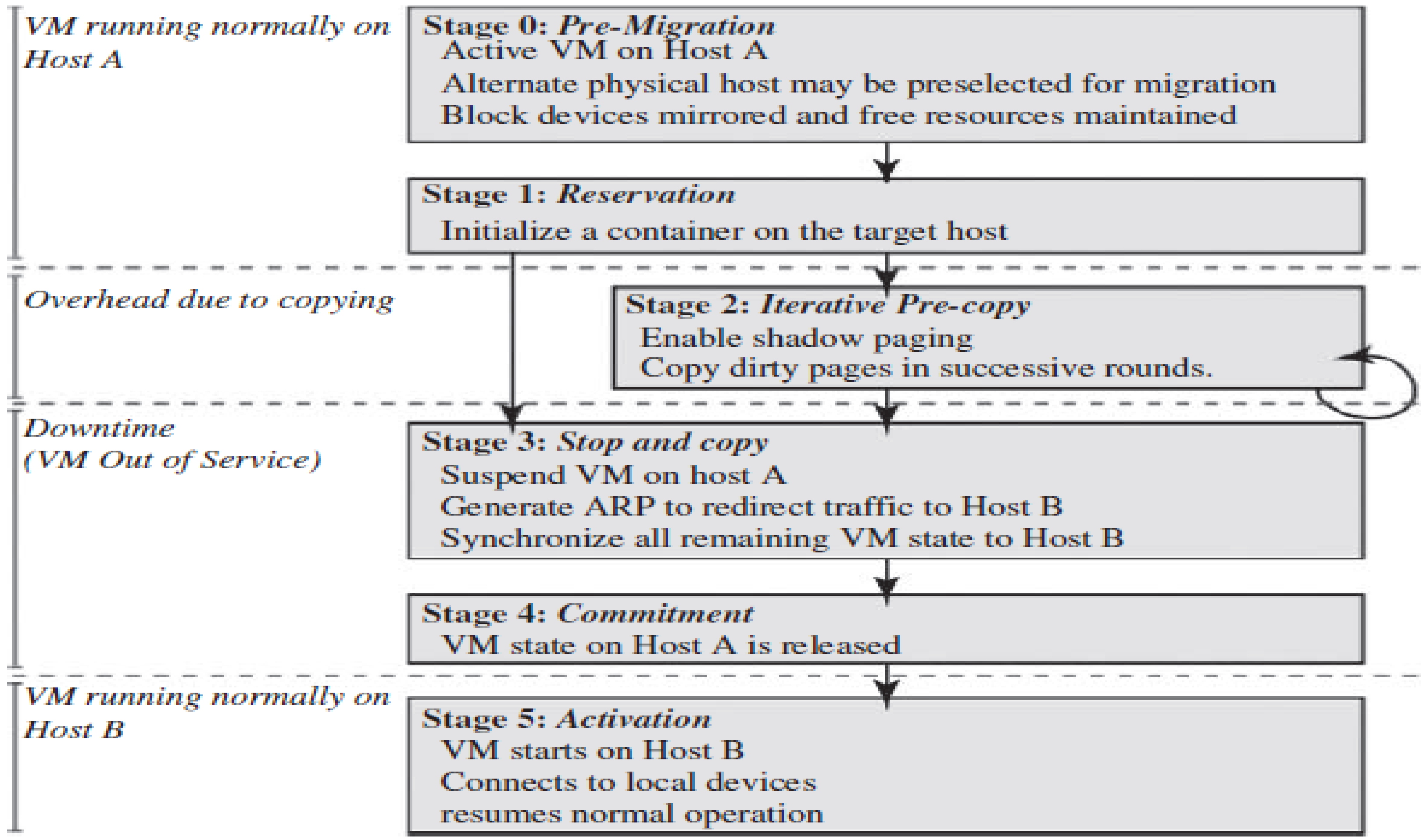
Live Migration

- When it is properly carried out,
  - this process takes place without any noticeable effect from the end user's point of view(a matter of milliseconds).

- Live migration is automatically started by strategies such as load balancing and server consolidation.

- Can also be used for load balancing
  - work is shared among computers
  - to optimize the utilization of available CPU Resources.

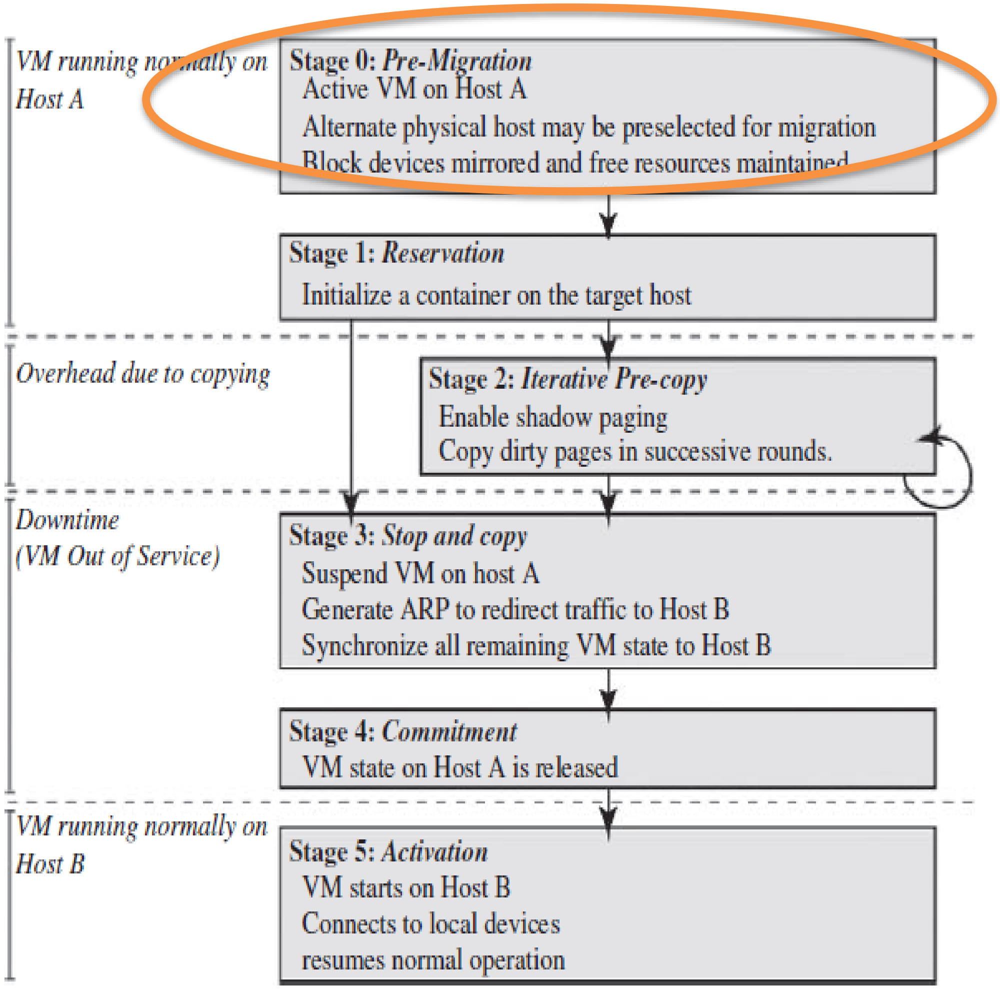
Live Migration


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime
(VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 0:Pre Migration:

- **Preparatory step**
- **Determining the migrating VM and the destination host**
- An Active VM exists on the physical host A.
- Alternate physical host may be selected for migration

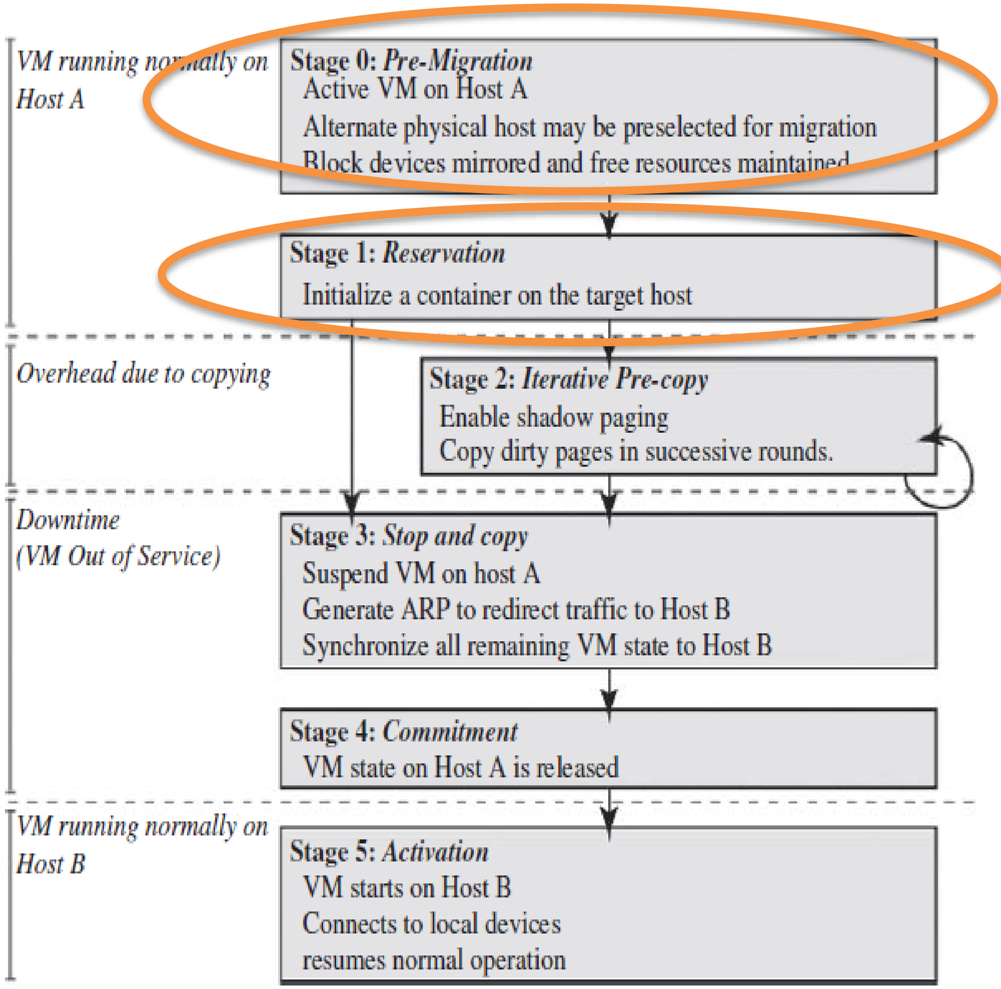


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 1:Reservation:

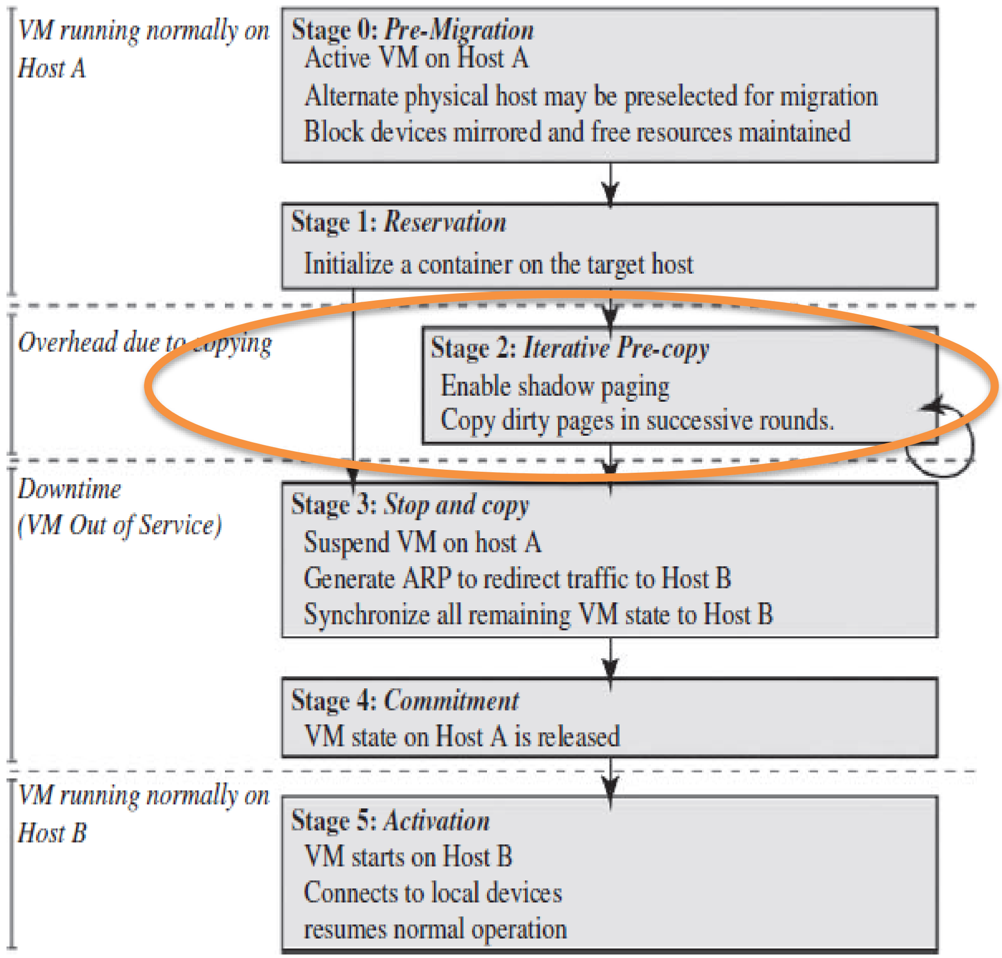
- A <u>request is issued to migrate an OS from host A to host B</u>
- A precondition is that <u>the necessary resources exist on B and has a VM container of that size</u>


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 2:Iterative precopy:

- Since the whole execution state of the VM is stored in memory,
  - **sending the VM's memory to the destination node** ensures continuity of the service provided by the VM.

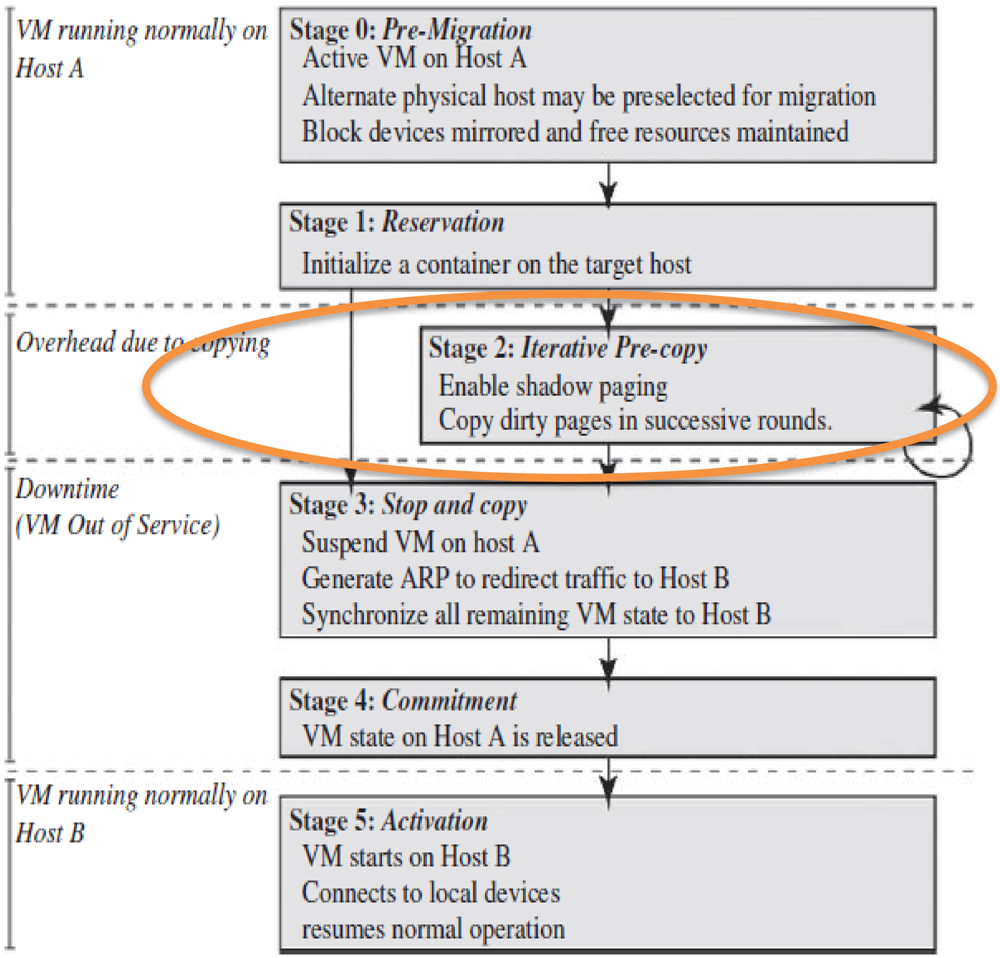


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 2:Iterative precopy:

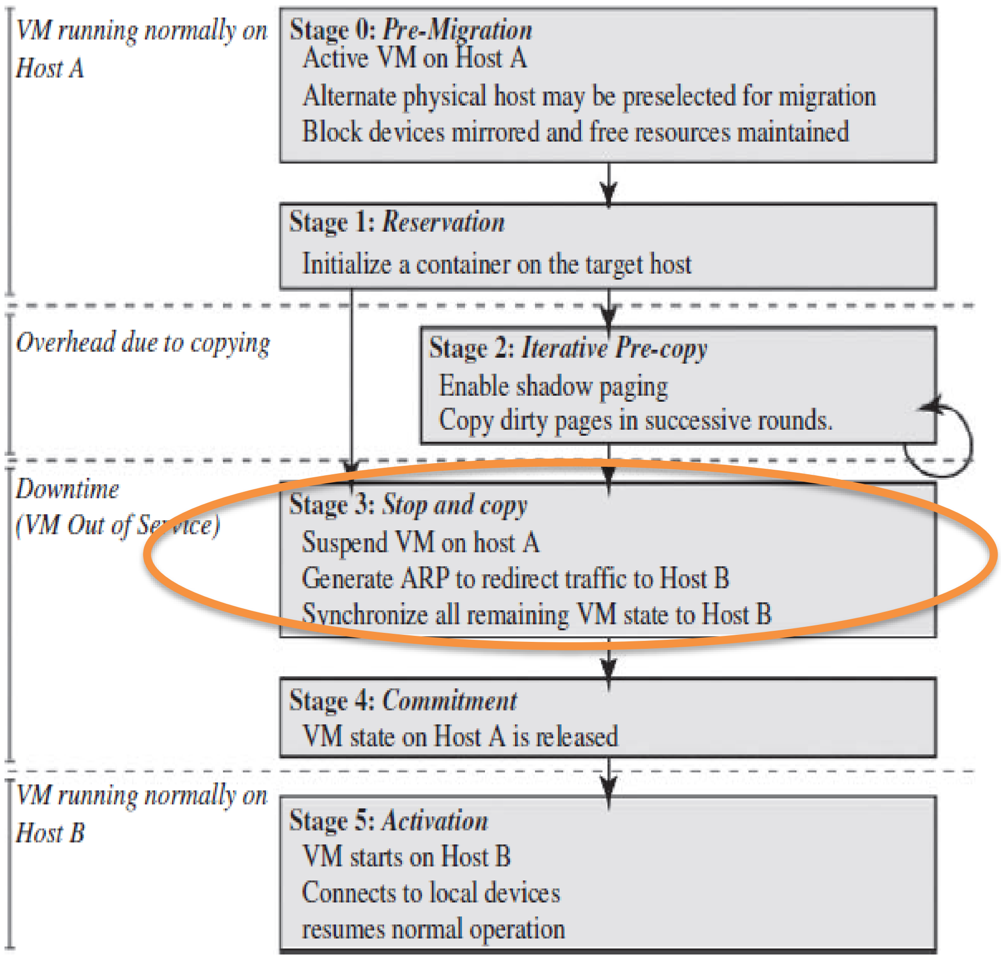
- During the first iteration,
  - all memory pages are transferred from A to B.
- Subsequent iterations
  - copy only those memory pages dirtied during the previous transfer phase.


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 3: Stop and Copy:

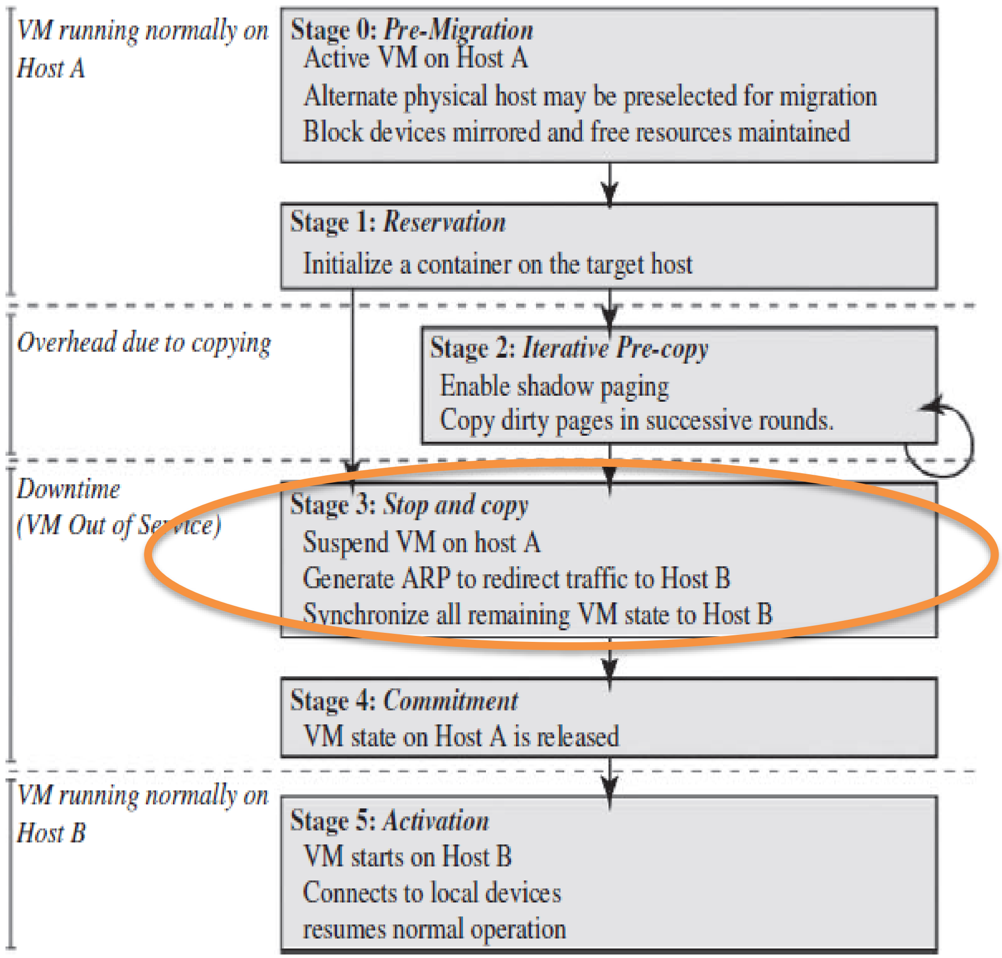
- Running OS instance at A is suspended
- **Other non-memory data such as CPU and network states should be sent as well.**
- Network traffic is redirected to B.
- CPU state and any remaining inconsistent memory pages are then transferred.


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 3: Stop and Copy:

- During this step, **the VM is stopped and its applications will no longer run.**
- This "service unavailable" time is called the **"downtime" of migration,** which should be as short as possible so that it can be negligible to users

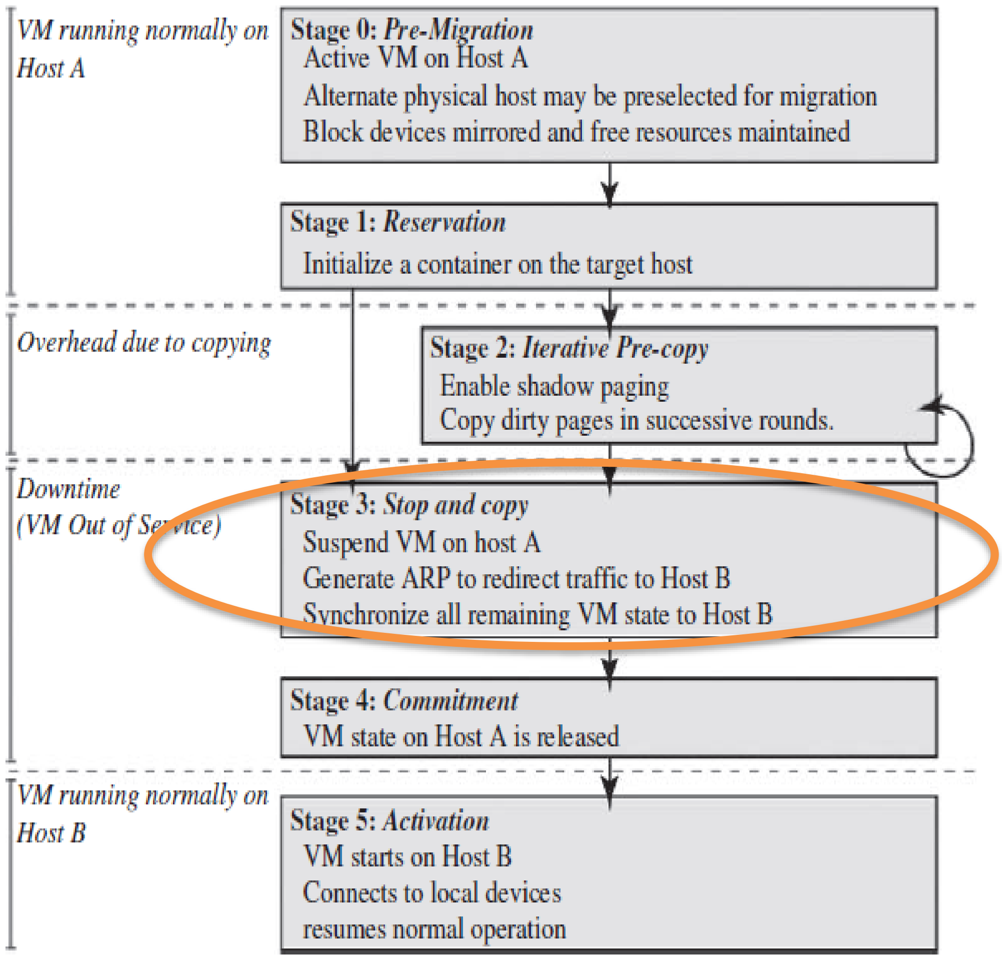


VM running normally on Host A
Overhead due to copying
Downtime (VM Out of Service)
VM running normally on Host B
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 3: Stop and Copy:

- At the end of this stage, there is
  - A <u>consistent suspended copy of the VM at both A and B.</u>
  - The copy at <u>A is considered primary</u>
  - <u>Resumed in case of failure.</u>

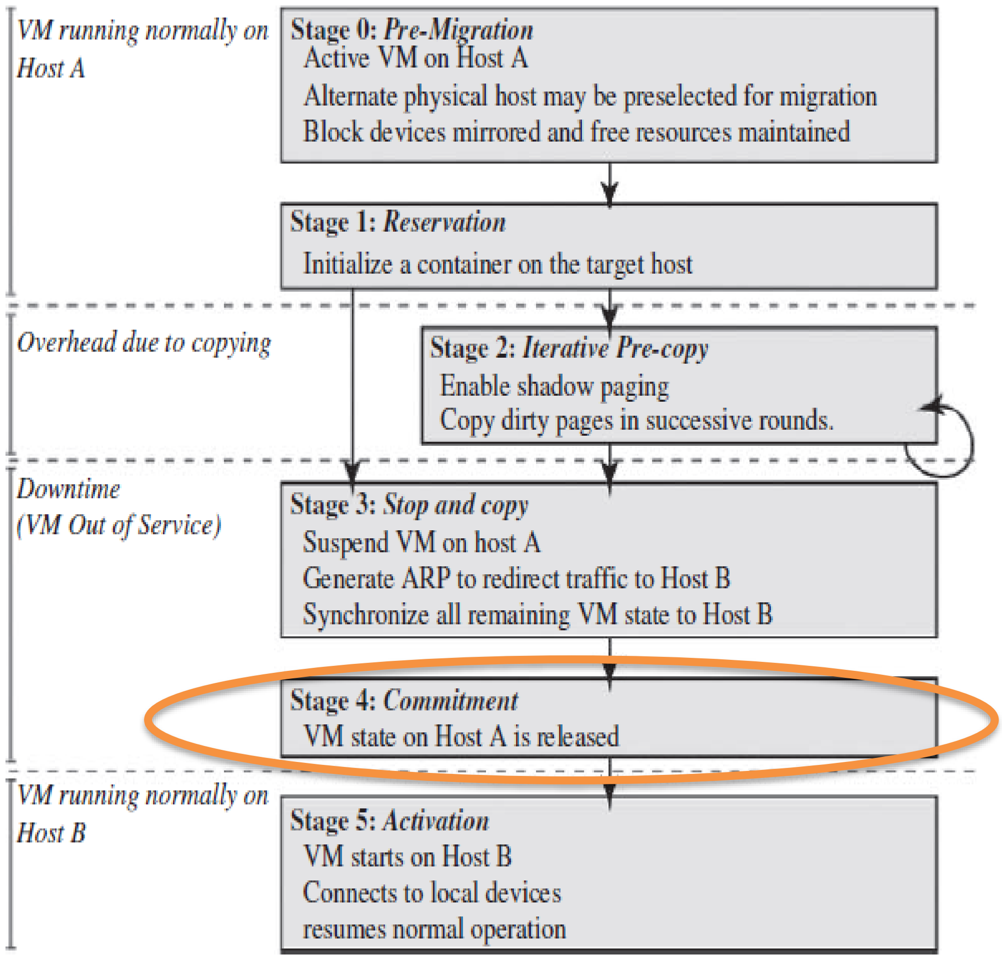


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 4:Commitment:

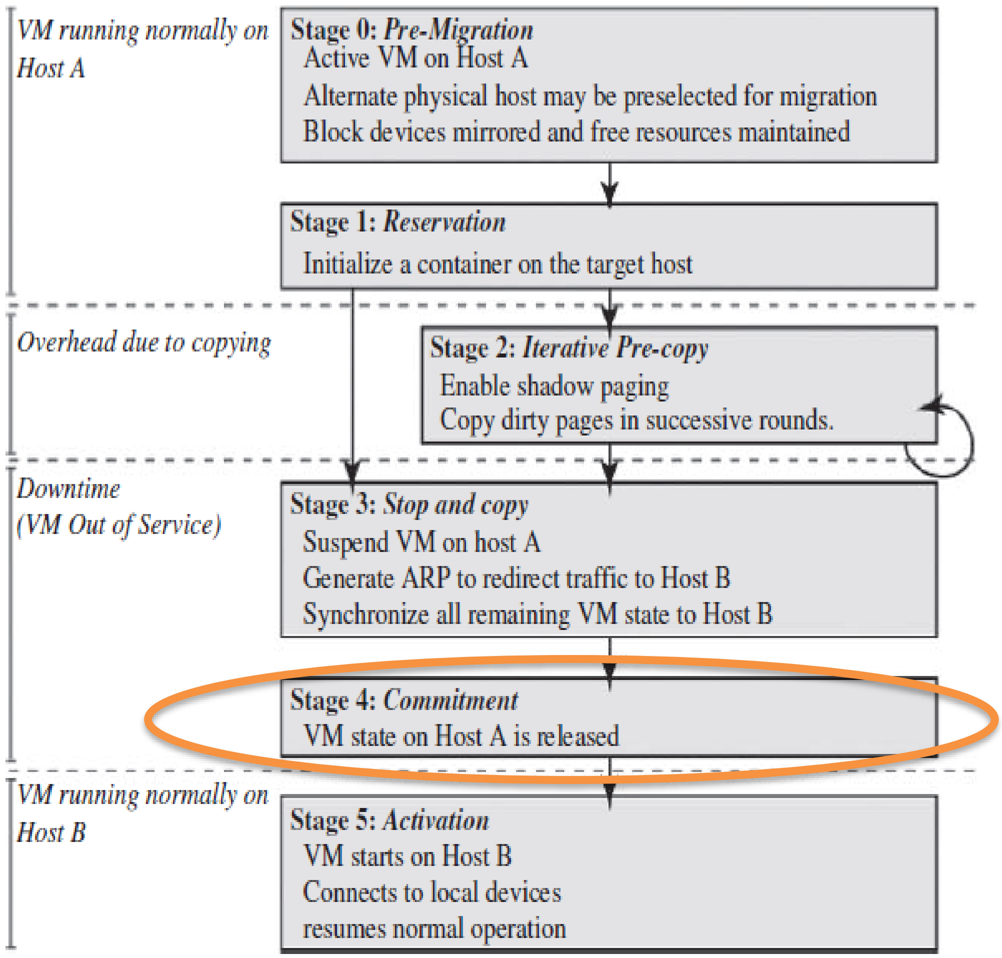
- Host B indicates to A that it has successfully received a consistent OS image.
- Host A acknowledges this message as a commitment of the migration transaction.


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime
(VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 4:Commitment:

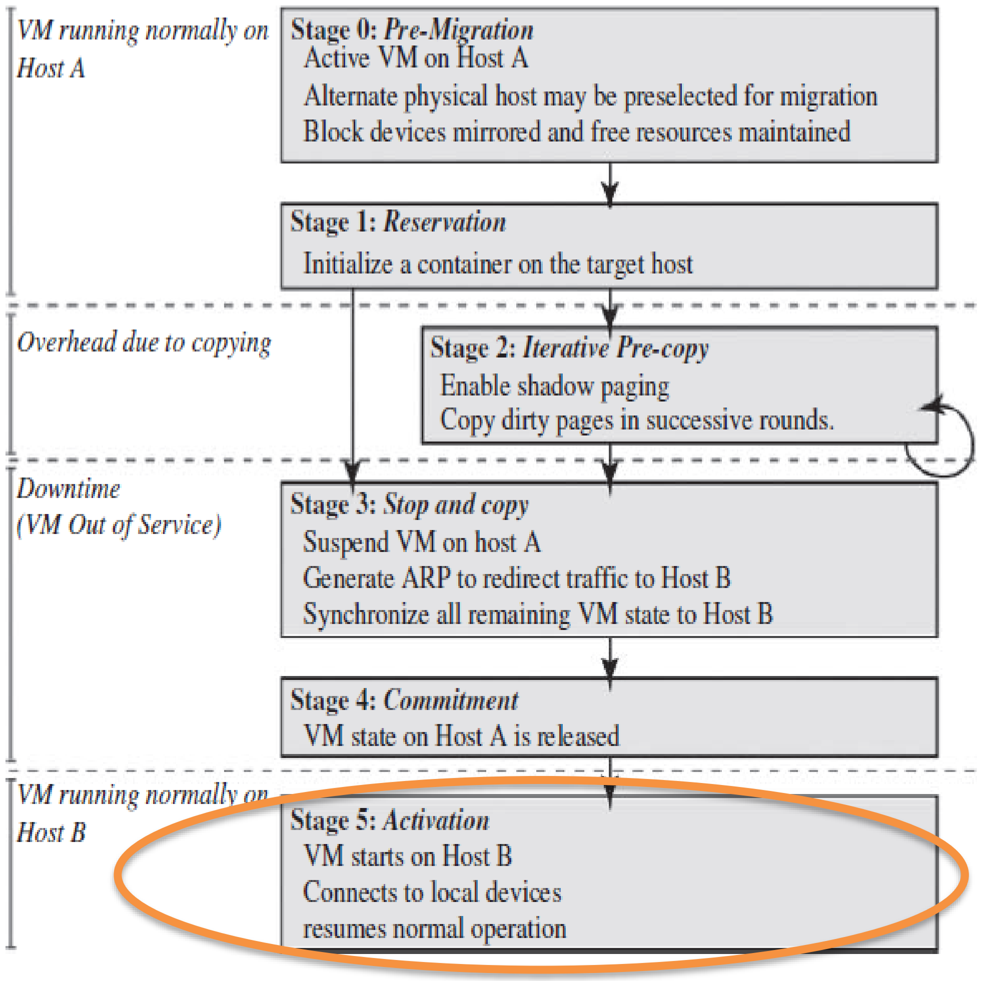
- Host A may now discard the original VM
- Host B becomes the primary host.


VM running normally on Host A
Stage 0: *Pre-Migration*
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: *Reservation*
Initialize a container on the target host
Overhead due to copying
Stage 2: *Iterative Pre-copy*
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: *Stop and copy*
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: *Commitment*
VM state on Host A is released
VM running normally on Host B
Stage 5: *Activation*
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

Stage 5:Activation:

- The migrated VM on B is now activated.
- Post Migration code runs to
  - reattach the device drivers to the new m/c and
  - Then the network connection is redirected to the new VM

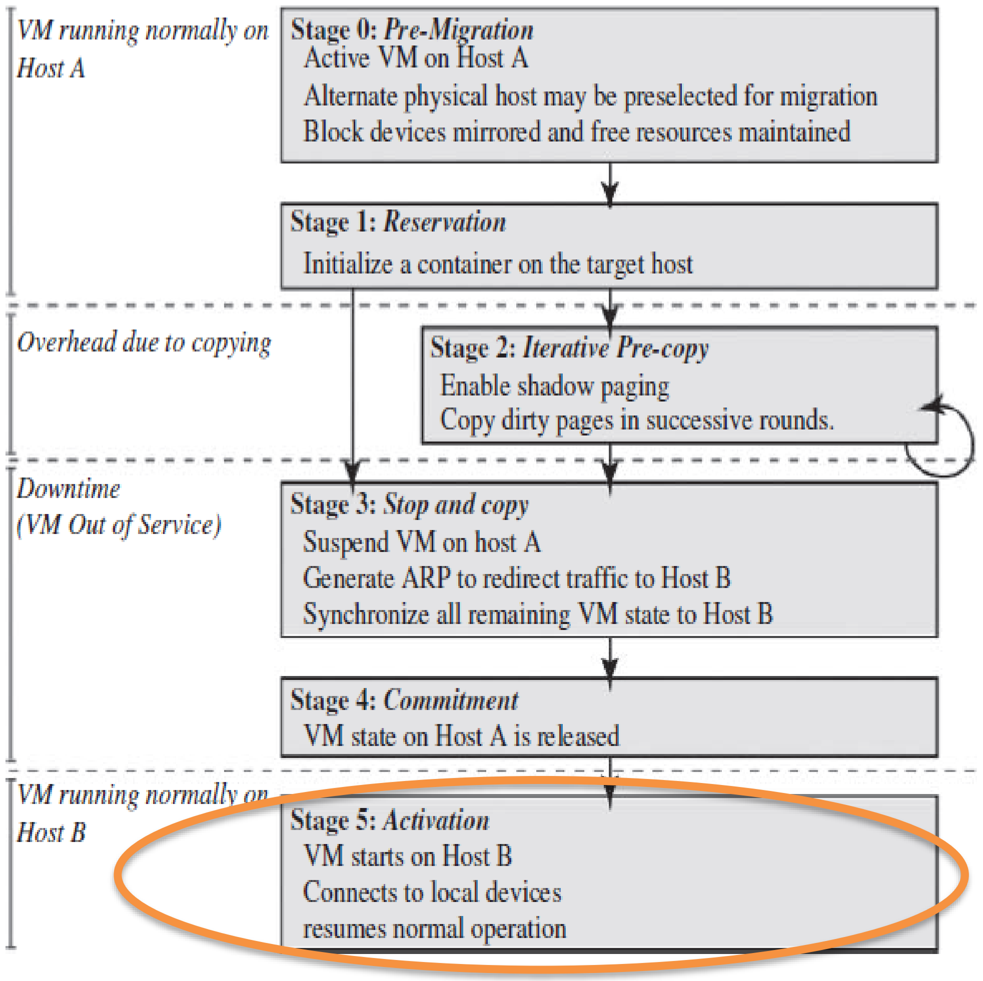


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

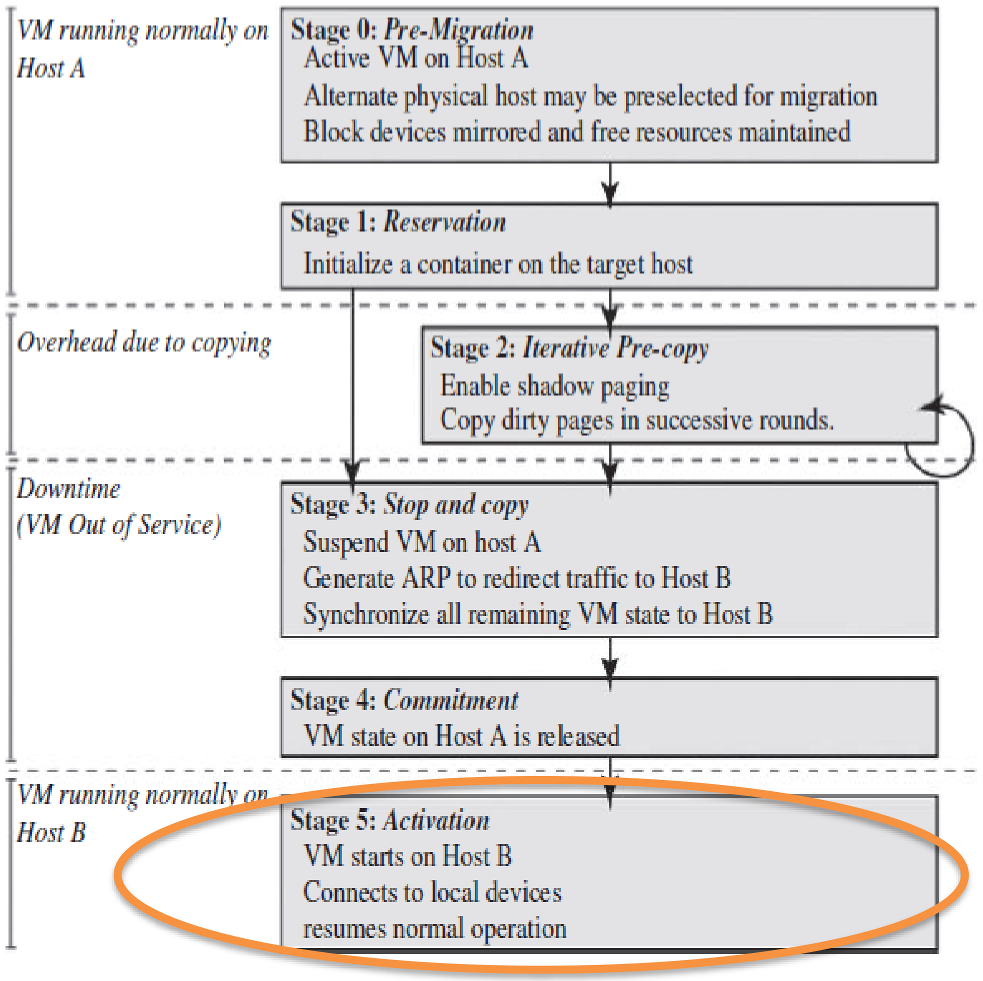
Stage 5:Activation:

- After all the needed data is copied, on the destination host,
- **The VM reloads the states and recovers the execution of programs in it, and the service provided by this VM continues.**


VM running normally on Host A
Stage 0: *Pre-Migration*
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: *Reservation*
Initialize a container on the target host
Overhead due to copying
Stage 2: *Iterative Pre-copy*
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: *Stop and copy*
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: *Commitment*
VM state on Host A is released
VM running normally on Host B
Stage 5: *Activation*
VM starts on Host B
Connects to local devices
resumes normal operation

Live Migration

- Atleast One host has a consistent VM image at all times during migration.


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
resumes normal operation

The Figure shows the effect on the data transmission rate (Mbit/second) of live migration of a VM from one host to another.

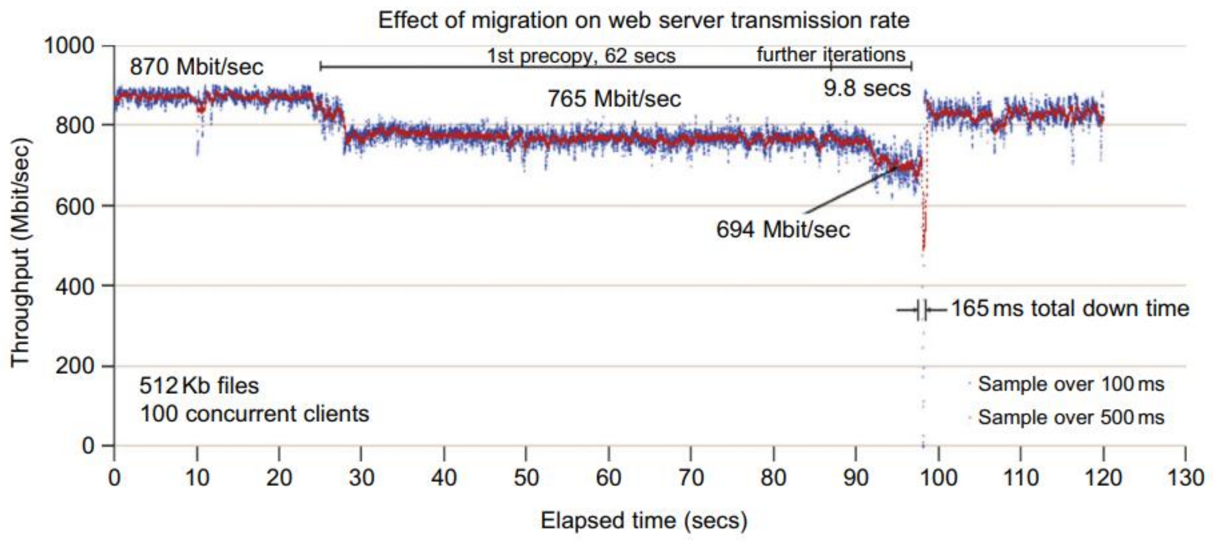


FIGURE 3.21

Effect on data transmission rate of a VM migrated from one failing web server to another.

(Courtesy of C. Clark, et al. [14])

1) Before copying the VM with 512 KB files for 100 clients, the data throughput was 870 MB/second.
2) The first precopy takes 63 seconds, during which the rate is reduced to 765 MB/second.
3) Then the data rate reduces to 694 MB/second in 9.8 seconds for more iterations of the copying process.
4) The system experiences only 165 ms of downtime, before the VM is restored at the destination host.

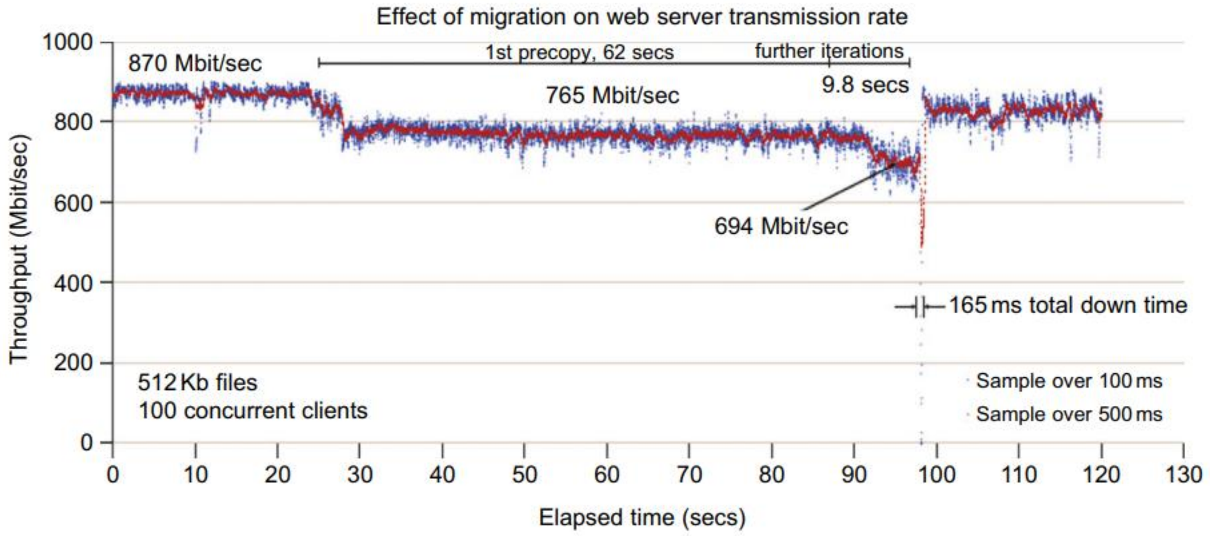


FIGURE 3.21

Effect on data transmission rate of a VM migrated from one failing web server to another.

(*Courtesy of C. Clark, et al.* [14])

- This experimental result shows a very small migration overhead in live transfer of a VM between host nodes.
- This is critical to achieve dynamic cluster reconfiguration and disaster recovery as needed in cloud computing.

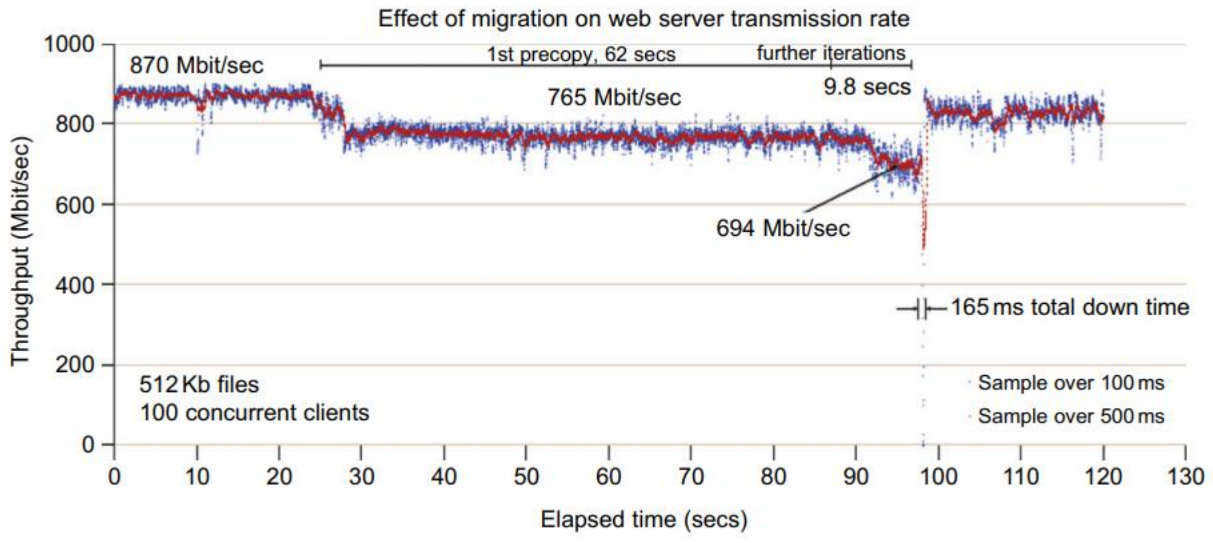


FIGURE 3.21

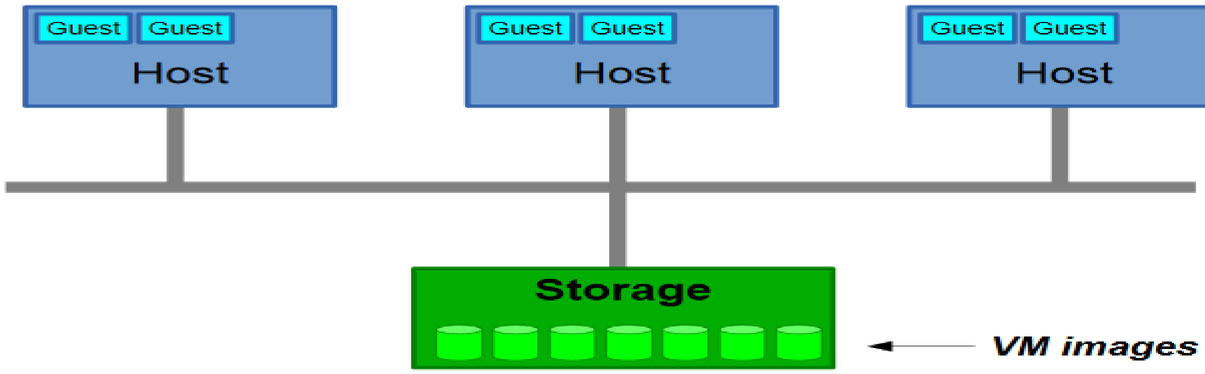
Effect on data transmission rate of a VM migrated from one failing web server to another.

(*Courtesy of C. Clark, et al.* [14])

Regular/Cold Migration

- Cold migration is the <u>migration of a powered-off virtual machine.</u>
- With cold migration, you have the option of moving the associated disks from one data store to another.
- In Cold Migration, The virtual machines are not required to be on a shared storage.

Traditional solution: shared storage



Regular/Cold Migration

- The two main differences b/w live migration and cold migration are:
  - Live migration needs a shared storage for virtual machines in the server's pool, but cold migration does not; also,
  - In live migration for a virtual machine between two hosts, there would be certain CPU compatibility checks to be applied; while in cold migration these checks do not apply.

Regular/Cold Migration

The cold migration process is simple to implement ,can be summarized as follows:

- The configuration files, including the NVRAM file (BIOS settings), log files, as well as the disks of the virtual machine, are moved from the source host to the destination host's associated storage area.
- The virtual machine is registered with the new host.
- After the migration is completed, the old version of the virtual machine is deleted from the source host.

Applications of Migration

??

Applications of Migration

- <u>Load balancing:</u>

When?

Applications of Migration

- <u>Load balancing:</u>
  - <u>Move VMs to a less busy host</u>
  - <u>Make use of newly-added capacity</u>

Applications of Migration

- Failure

  How?

Applications of Migration

- Load balancing:
  - Move VMs to a less busy host
  - Make use of newly-added capacity
- Recovery from host failure
  - Restart VM on a different host

Applications of Migration

- Maintenance

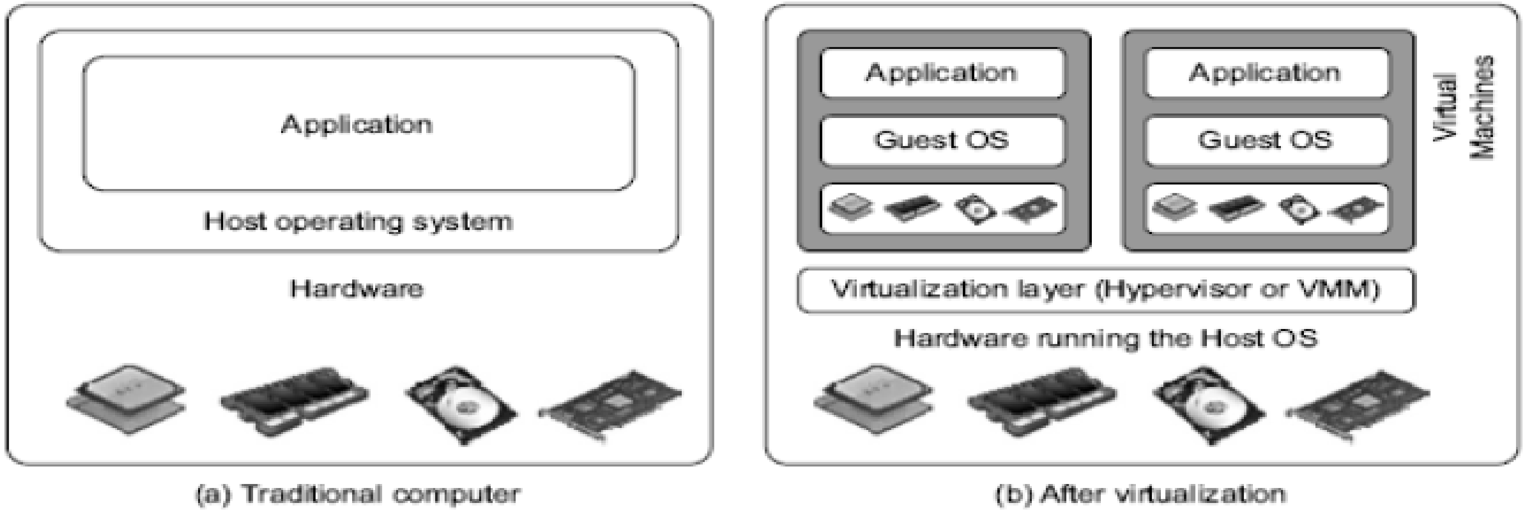
  How?

Applications of Migration

- Load balancing:
  - Move VMs to a less busy host
  - Make use of newly-added capacity
- Recovery from host failure
  - Restart VM on a different host
- Maintenance
  - Planned Shutdown:
    - Move VMs off a host before it is shut down

Levels of Virtualization Implementation

- The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively.
- This can be implemented **at various operational levels.**



Levels of Virtualization Implementation

- The virtualization software creates the abstraction of VMs by interposing a virtualization layer <u>at various levels of a computer system.</u>

Levels of Virtualization Implementation

- Common virtualization layers include the
  1) Instruction set architecture (ISA) level
  2) Hardware level
  3) Operating system level
  4) Library support level
  5) Application level

Levels of Virtualization Implementation

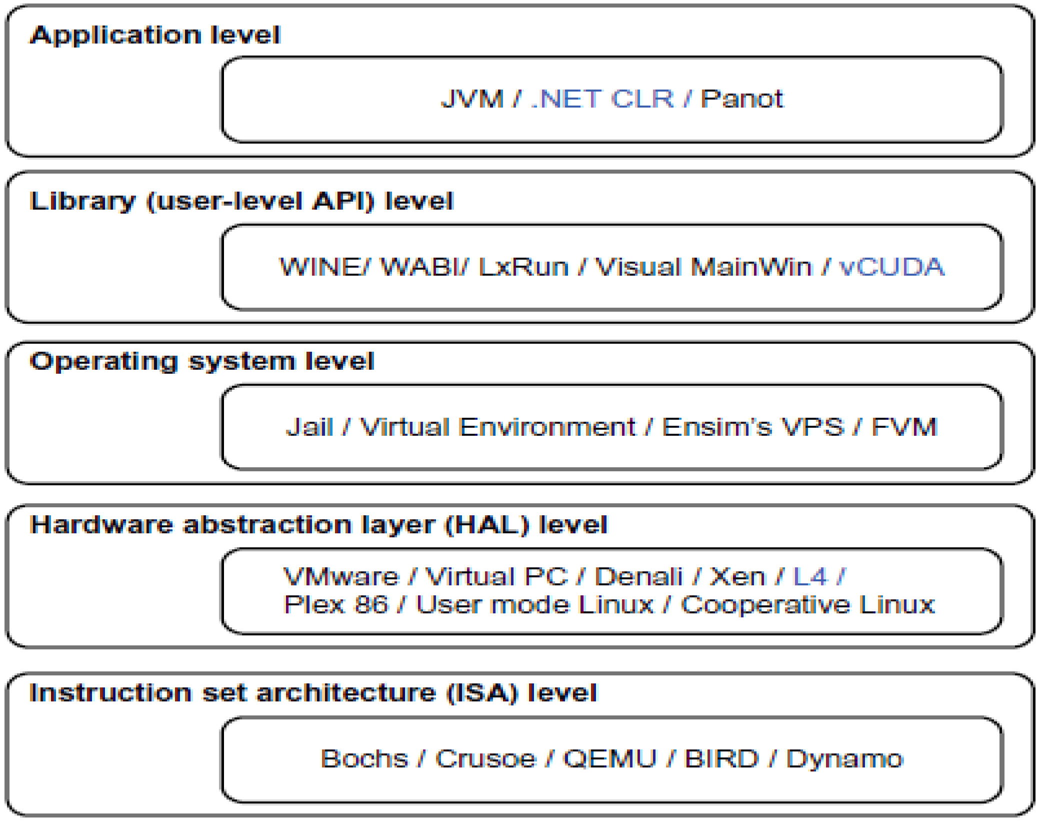

Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo

FIGURE 3.2

Virtualization ranging from hardware to applications in five abstraction levels.

Instruction Set Architecture Level

What is an Instruction Set?

What is an Instruction Set?

- The instruction set provides commands to the processor, to tell it what it needs to do.

What is an Instruction Set?

- The instruction set provides commands to the processor, to tell it what it needs to do.

- The instruction set consists of
  - addressing modes,
  - operation,
  - native data types,
  - registers,
  - memory architecture,
  - interrupt, and
  - exception handling, and
  - external I/O.

Code Sequence C = A + B for Four Instruction Sets

Accumulator	Register (register-memory)	Register (load-store)
Load A Add B Store C	Load R1, A Add R1, B Store C, R1	Load R1,A Load R2, B Add R3, R1, R2 Store C, R3

What is an Instruction Set?

- Example-
  - **x86 instruction set, which is common to find on computers today.**
  - is a family of complex instruction set computer (CISC) instruction set architectures initially developed by Intel based **on the Intel 8086 microprocessor and its 8088 variant.**
- Both the Intel Pentium and AMD Athlon processors use nearly the same x86 instruction set.
- Different computer processors can use almost the same instruction set while still having very different internal design.

Instruction Set Architecture Level

- Virtualization –
- Transforming the physical architecture of the system's instruction set completely into software.

Instruction Set Architecture Level

- **The guest systems issue instructions** for the emulator to process and execute.
- **The instructions are received by the emulator, which transforms them into a native instruction set.**
- **The native instructions are run on the host m/c's hardware.**
- The instructions include both the processor-oriented instructions and the I/O specific ones.

- At the ISA level, **virtualization is performed by emulating a given ISA by the ISA of the host machine.**

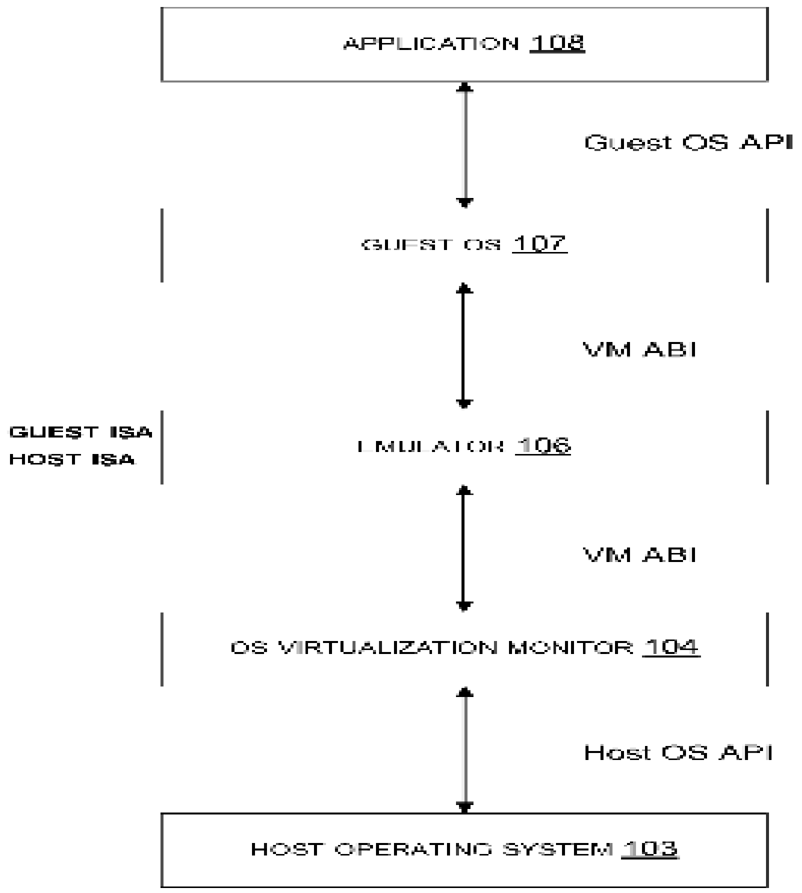
- For example,
- **MIPS binary code can run on an x86-based host machine with the help of ISA emulation.**
- With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine

- MIPS (Microprocessor without Interlocked Pipelined Stages) is a family of reduced instruction set computer (RISC) instruction set architectures (ISA) developed by MIPS Technologies, based in the United States.

- There are multiple versions of MIPS: including MIPS I, II, III, IV, and V; as well as five releases of MIPS32/64 (for 32- and 64-bit implementations, respectively).

Instruction Set Architecture Level

- Receiving a request to execute an application.
- The application can include first application instructions from a guest instruction set architecture.
- Loading an emulator
- The emulator can translate the first application instructions into second application instructions from a host instruction set architecture.
- Running the application by executing the second application instructions.



Instruction Set Architecture Level

- The basic emulation method is through **code interpretation.**
- **An interpreter program interprets the source instructions to target instructions one by one.**
- **One source instruction may require tens or hundreds of native target instructions to perform its function.**
- Obviously, this process is relatively **slow.**

Instruction Set Architecture Level

- For better performance, **dynamic binary translation is desired.**
- This approach translates **basic blocks of dynamic source instructions to target instructions** to increase translation efficiency.
- Instruction set emulation requires binary translation and optimization.
- **A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler**

Instruction Set Architecture Level

- **Advantage:**

Enables the host system to adjust to a change in the architecture of the guest system, if accomplishing a task can be possible through the instructions that are available with the host system.

Instruction Set Architecture Level

- **Advantage:**
- **The infrastructure provided by the virtualization of this kind can be used for creating VMs on a platform.**
- **For eg- x86 on any platform such a x86,Sparc or Alpha etc.**
- **With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine.**

Instruction Set Architecture Level

Disadvantage:

- The instructions need to be interpreted before being executed.
- An interpreter program interprets the source instructions to target instructions one by one.
- Therefore the system with the virtualization at ISA level shows a poor performance.
- **One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow.**

Qemu

- **QEMU is a generic and open source machine emulator and virtualizer.**
- Run operating systems for any machine, on any supported architecture

QEMU

A generic and open source machine emulator and virtualizer

Hardware Abstraction Level WHAT?

- Virtualization is performed right

“<u>**on top of the bare hardware.”**</u>

- On the one hand, this approach generates **a virtual hardware environment for a VM.**
- On the other hand, the process **manages the underlying hardware through virtualization**

Hardware Abstraction Level WHAT?

- The idea is to **virtualize a computer's resources**, such as its:
  - **Processors**
  - **memory, and**
  - **I/O devices.**

- **The intention is to upgrade the**
  - **hardware utilization rate by multiple users concurrently.**

Virtualization at Hardware Abstraction level: WHAT?

- Typical systems: VMware, Virtual PC, Denali, Xen

Virtualization at Hardware Abstraction level: WHAT?

- Example-
- **The Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.**

Hardware Abstraction Level(Privileged Instructions) WHAT?

What are Privileged Instructions?

- The Instructions that can run only in Kernel Mode are called Privileged Instructions

Privileged instructions by VMs

- Execution of privileged instructions need full attention of CPU.
- If not managed properly by the VMM, will raise an exception resulting into system crash.

Advantages:-

- Trapping and forwarding the privileged instructions to VMM helps in managing a system properly, thereby
  - avoiding various risks and
  - keeping individual virtual m/cs isolated.

Hardware Abstraction Level(Privileged Instructions) HOW?

- Hardware Level virtualization needs
  - trapping the execution of privileged instructions by the virtual m/c,
  - which must pass these instructions to the VMM for being handled properly

Hardware Abstraction Level(Privileged Instructions) HOW?

- This is required because of the possible existence of multiple virtual machines,
  - each having its own OS that might issue <u>separate privileged instructions.</u>

Virtualization at Hardware Abstraction level :

- It supports multiple OSs and applications to be run simultaneously which require
  - **no system reboot or dual-boot setup.**
- It gives the appearance of
  - **multiple separate m/cs each of which can be used as a normal system.**
- **The degree of isolation is high, whereas**
  - **implementation is less risky and maintenance is easy.**

Virtualization at Hardware Abstraction level:

- However, this technique permits you to have access to a raw computer
  - which requires a lot of time to be spent in the installation and administration of the virtual system
  - before you can think of testing or running applications.

- **<u>In case the physical and virtual Oss are the same,</u>**
  - **It results in <u>duplication of your efforts</u>, which should be avoided for an efficient use of the system.**

Hardware Abstraction Level

- Advantage: **has higher performance and good application isolation**

- Shortcoming & limitation: **very expensive to implement (complexity)**

Virtualization at Operating System level:

- To overcome the issues of redundancy and time consumption
  - we implement virtualization at a higher level, i.e. virtualization at the level of the OS.

- **In case the physical and virtual Oss are the same,**
  - It results in duplication of your efforts, which should be avoided for an efficient use of the system.

Virtualization at Operating System level:

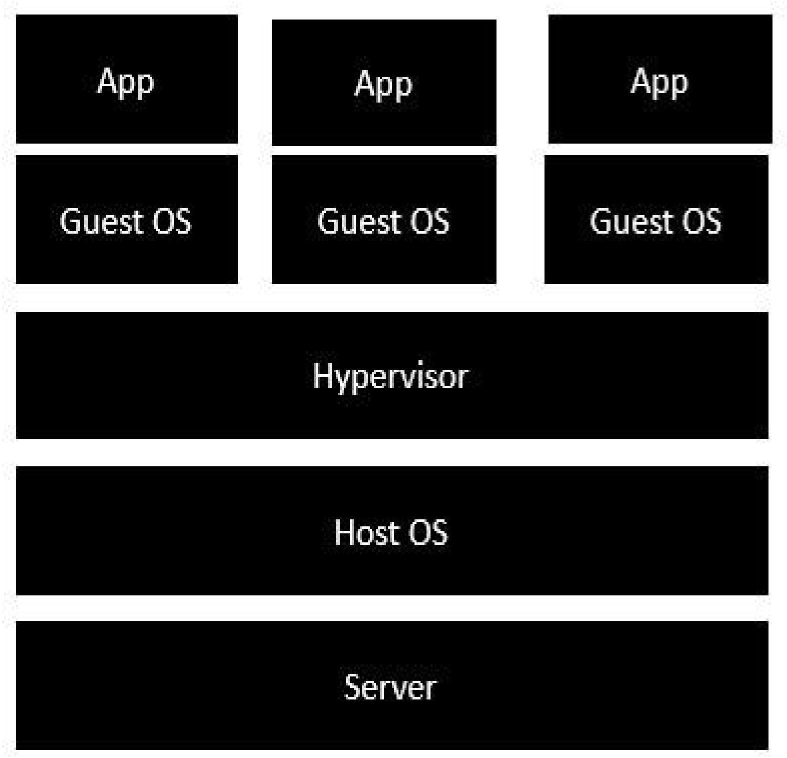
- It is an "**abstraction layer between traditional OS and user applications.**"

Virtualization at Operating System level:

- Typical systems: Jail / Virtual Environment / Ensim's VPS / FVM

Traditional Virtualization

- Server is the physical server that is used to host multiple virtual machines.
- Host OS is the base machine such as Linux or Windows.
- Hypervisor is either VMWare or Windows Hyper V that is used to host virtual machines.
- One would then install multiple operating systems as virtual machines on top of the existing hypervisor as Guest OS.
- One would then host your applications on top of each Guest OS.



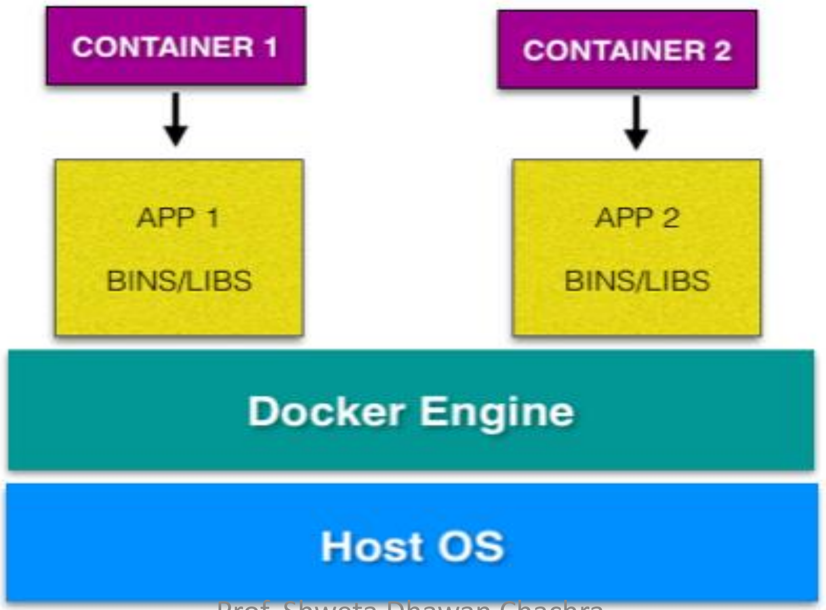
Dockers

- Dockers containers are **analogous to physical containers that you can use to store, package, and transport goods.**
- But instead of tangible goods, they're containers for software applications.

Dockers

- A container is an environment that **runs an application that is not dependent on the operating system.**

- A docker container is a portable unit of software—
- that has the application
- along with the associated dependency and configuration.



DOCKER ARCHITECTURE

What is a Container?

- **The kernel of the host operating system**
- **serves the needs of running different functions of an app, separated into containers.**

What is a Container?

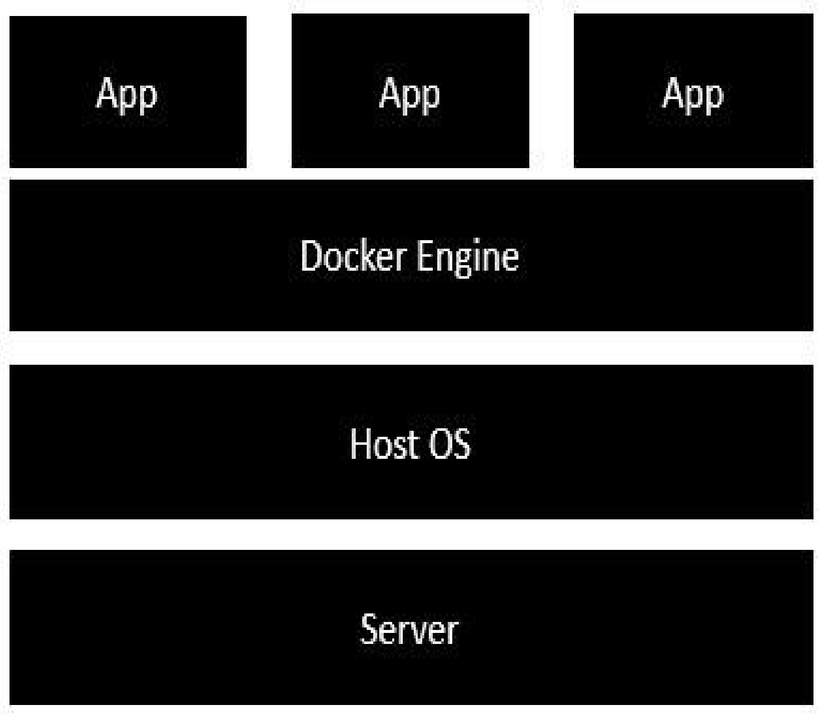
- **Each container runs isolated tasks.**
- **It cannot harm the host machine nor**
- **come in conflict with other apps running in separate containers.**

Docker – Architecture

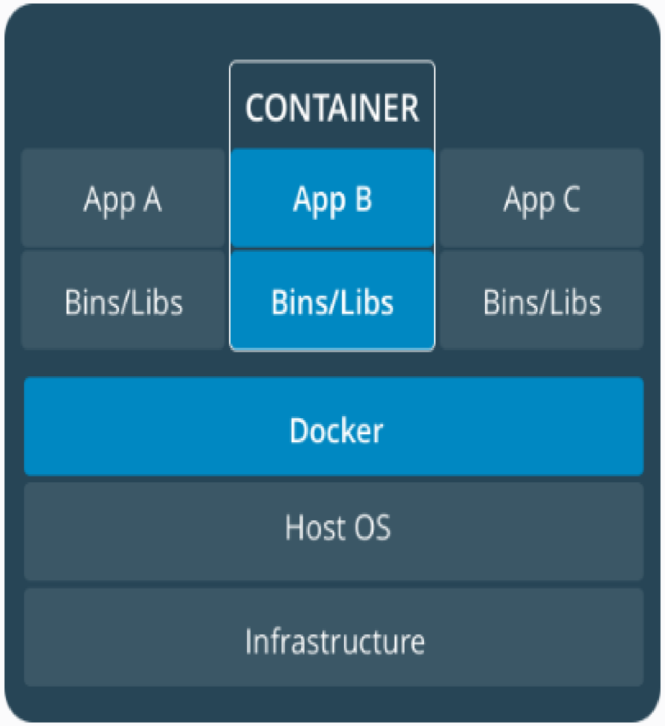
- **Docker engine-**

is used to virtualize the guest operating system which earlier used to run in virtual machines

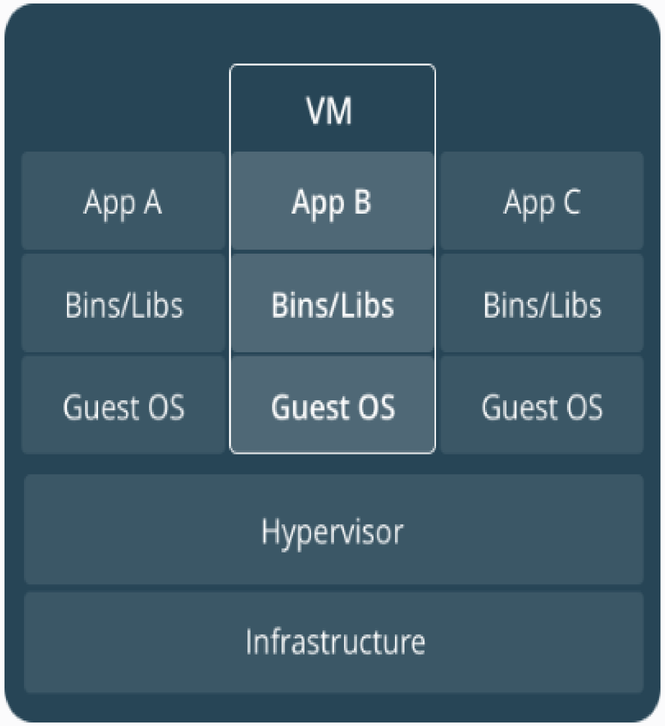
- All of the Apps now run as Docker containers.



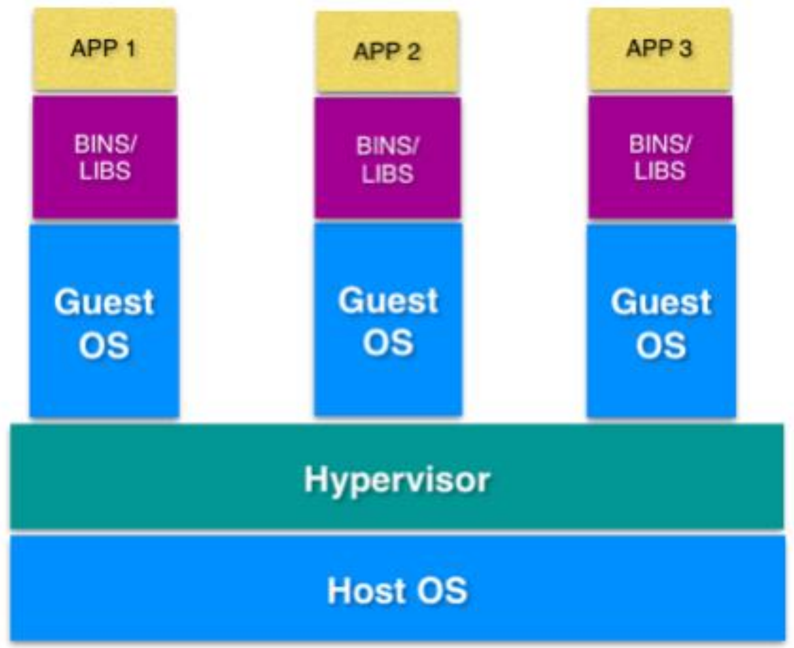
Containers vs Virtual Machines



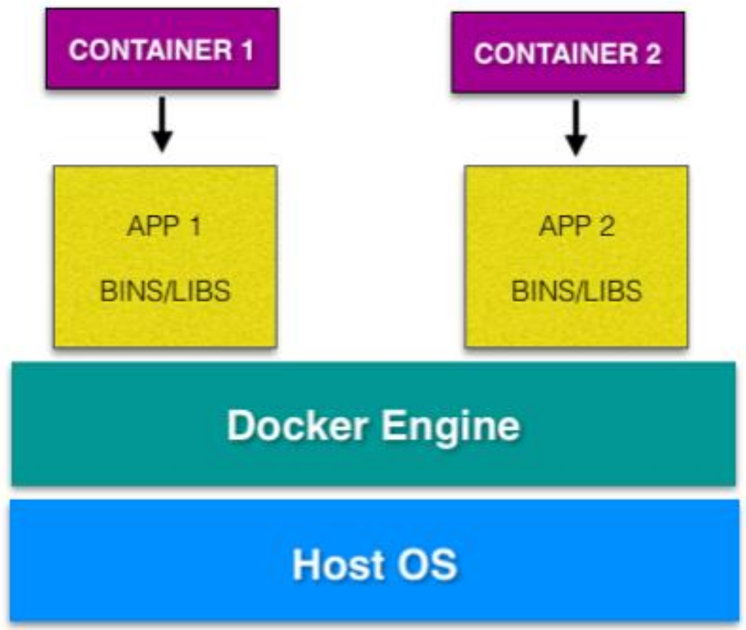
Container



VM



VIRTUAL MACHINE ARCHITECTURE



DOCKER ARCHITECTURE

Virtual Machines and Containers

- **Virtual machines** run guest operating systems - the OS layer in each box.
- Resulting disk image and **application state is**
- **an entanglement of OS settings,**
- **system-installed dependencies,**
- OS security patches

Virtual Machines and Containers

- **Containers** can share a single kernel **<u>and the only information that needs to be in a container image is the executable and its package dependencies, which never need to be installed on the host system.</u>**
- These processes run like native processes, and can be managed individually
- Because they contain all their dependencies, there is no configuration entanglement;
- A containerized app "runs anywhere"

Operating System Level

- The containers behave like real servers.

- OS-level virtualization is commonly used in creating virtual hosting environments to
  - allocate hardware resources <u>**among a large number of mutually distrusting users.**</u>

Library Support Level

- Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS.

APIs?

- <u>What?</u>
  - API can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses.

APIs?

How do APIs work?

- API architecture is usually explained in terms of client and server.
- The application sending the request is called the client, and the application sending the response is called the server.

- In the weather example, the bureau’s weather database is the server, and the mobile app is the client.

APIs?

How do APIs work?

- There are four different ways that APIs can work depending on when and why they were created.
- **SOAP APIs**
  - These APIs use Simple Object Access Protocol. Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.
- **RPC APIs**
  - These APIs are called Remote Procedure Calls. The client completes a function (or procedure) on the server, and the server sends the output back to the client.
- **Websocket APIs**
  - Websocket API is another modern web API development that uses JSON objects to pass data.
- **REST APIs**
  - These are the most popular and flexible APIs found on the web today. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

APIs?

- <u>APIS are used to</u>
  - Enable programmers to write programs easily.
  - Most systems provide well-documented APIs,

- **This is taken as <u>another candidate for virtualization.</u>**

Library level virtualization

- Also called
  - User-level Application Binary Interface (ABI) or
  - **API emulation.**

Library level virtualization

- It is done by API call interception and remapping.
- **Create execution environments for running alien programs on a platform**
- **rather than creating a VM to run the entire operating system.**
- Typical systems: Wine, WAB, LxRun , VisualMainWin

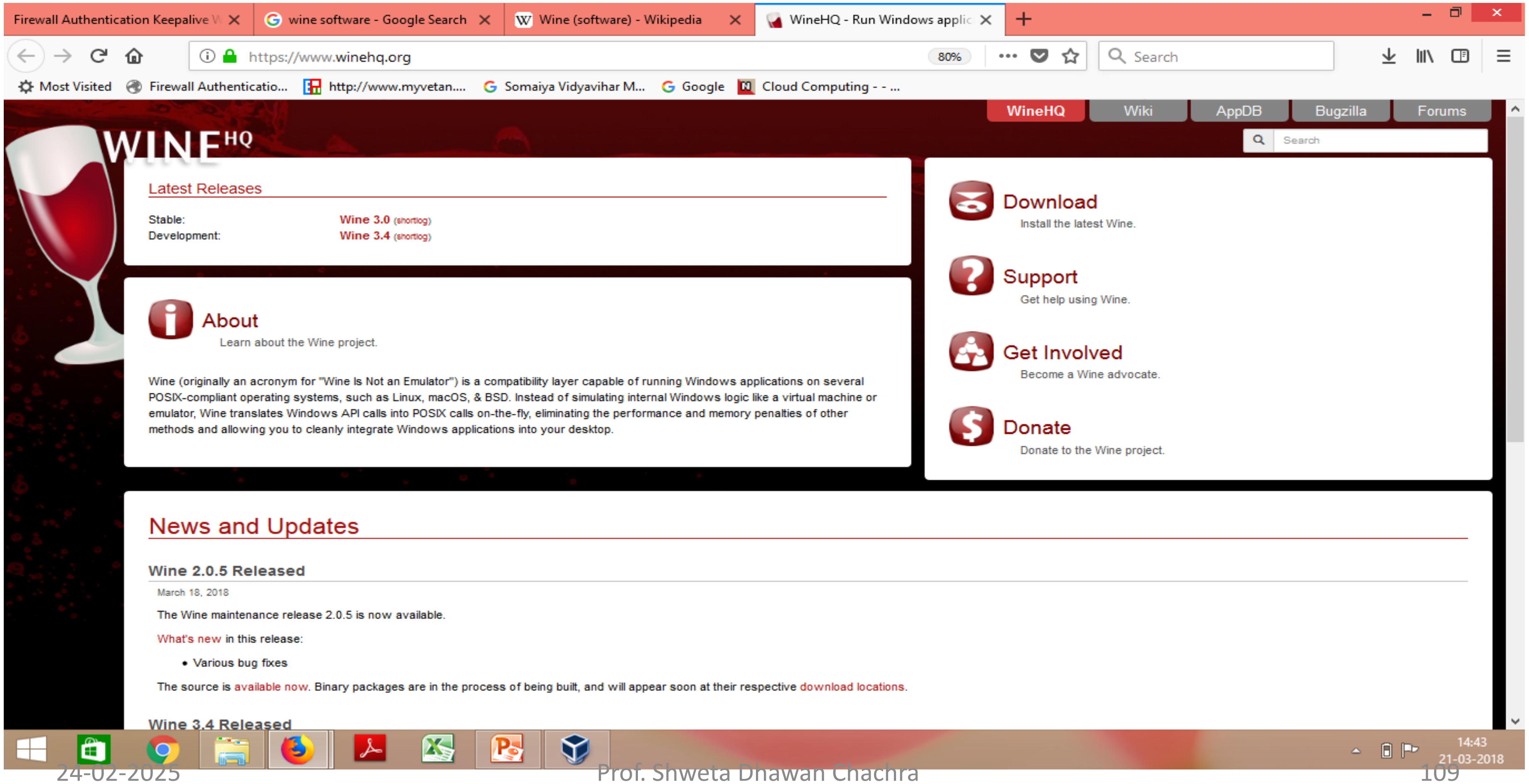
Library Support Level

- **Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks.**

- The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts.
  - *https://www.winehq.org/*

WINE

- https://www.winehq.org/
- Screenshot:



Latest Releases

Stable:	Wine 3.0 (shortlog)
Development:	Wine 3.4 (shortlog)

About

Learn about the Wine project.

Wine (originally an acronym for "Wine Is Not an Emulator") is a compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, macOS, & BSD. Instead of simulating internal Windows logic like a virtual machine or emulator, Wine translates Windows API calls into POSIX calls on-the-fly, eliminating the performance and memory penalties of other methods and allowing you to cleanly integrate Windows applications into your desktop.

Download
Install the latest Wine.

Support
Get help using Wine.

Get Involved
Become a Wine advocate.

Donate
Donate to the Wine project.

News and Updates

Wine 2.0.5 Released

March 18, 2018

The Wine maintenance release 2.0.5 is now available.

What's new in this release:

- Various bug fixes

The source is available now. Binary packages are in the process of being built, and will appear soon at their respective download locations.

Wine 3.4 Released

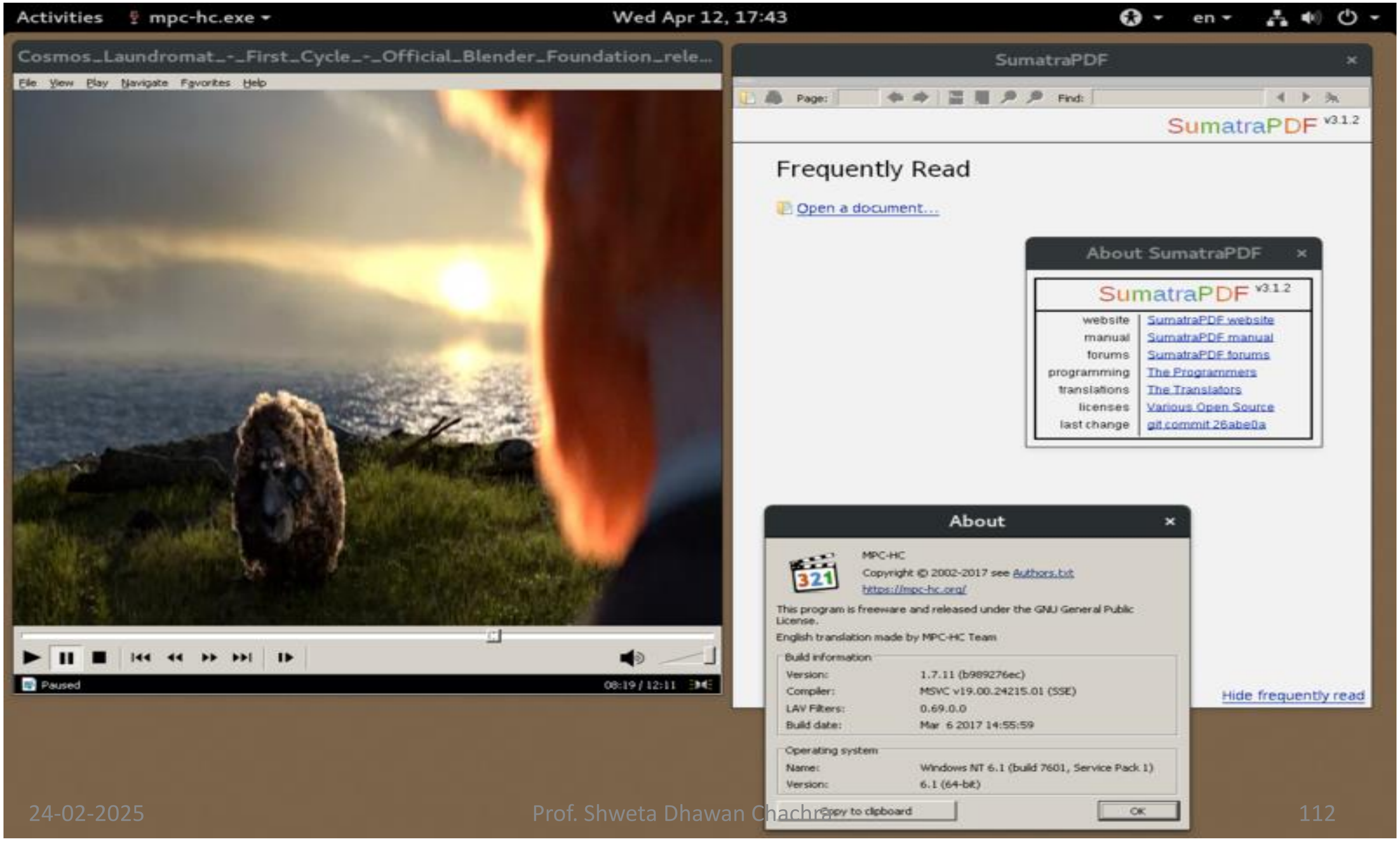
WINE

- Wine (recursive backronym for Wine Is Not an Emulator)
- A free and open-source compatibility layer that aims to allow
  - computer programs developed for Microsoft Windows
  - to run on Unix-like operating systems.

WINE

- <u>Instead of simulating internal Windows logic like a virtual machine,</u>
  - <u>Wine translates Windows API calls into POSIX((Portable Operating System Interface) calls on-the-fly,</u>
  - eliminating the performance and memory penalties of other methods and
  - allowing you to cleanly integrate Windows applications into your desktop.

WINE v2.0.5 running Sumatra PDF and Media Player Classic on Fedora



website	SumatraPDF website
manual	SumatraPDF manual
forums	SumatraPDF forums
programming	The Programmers
translations	The Translators
licenses	Various Open Source
last change	git commit 26abe0a

MPC-HC
Copyright © 2002-2017 see Authors.txt
https://mpc-hc.org/

This program is freeware and released under the GNU General Public License.

English translation made by MPC-HC Team

Build information

Version:	1.7.11 (b989276ec)
Compiler:	MSVC v19.00.24215.01 (SSE)
LAV Filters:	0.69.0.0
Build date:	Mar 6 2017 14:55:59

Operating system

Name:	Windows NT 6.1 (build 7601, Service Pack 1)
Version:	6.1 (64-bit)

Library Support level:

- Another example is the **vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.**

Library Support level:

- Advantage: It has very low implementation effort
- Shortcoming & limitation: poor application flexibility and isolation

User-Application Level

- Application level virtualization
- **Virtualizes an application as a VM.**
- On a traditional OS, an application often runs as a process. Thus known as **Process-level virtualization.**

User-Application level:

- **High level language (HLL) VMs**

- The virtualization layer sits as
  - **an application program on top of an operating system and**
  - **exports an abstraction of a VM that can run programs which are written and compiled to that particular abstract machine definition**.

- **Any program written in the HLL and compiled for this VM will run on it.**

User-Application Level

- This type of VM has become popular with **the Java programming language, which is implemented using the Java virtual machine.**

- Other examples include the **.NET Framework, which runs on a VM called the Common Language Runtime.**

User-Application level:

- **The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.**

- The arrival of Java Virtual Machine(JVM) brought a new dimension to virtualization.

JVM

- A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages but compiled to Java bytecode.

Revisiting facts about Java

- The Compiler of java called as **javac** converts source code into a Intermediate file known as **Bytecode** file.

- The Bytecode file is platform independent

- The Interpreter of java (**java**) converts Bytecode into the specific OS compatible machine code . This code will vary according to OS.

- **Java interpreter:**
  - The Java interpreter decodes and executes bytecode for the Java virtual machine.

- The Java interpreter is actually a part of JVM.

- Virtual machine is not just executing the bytecodes, it has lot of tasks to do. That full-fledged environment is referred to as a JVM.

JVM

- The Java Virtual Machine is responsible for **interpreting Java byte code and translating this into actions or Operating System calls.**

- For example, **a request to establish a socket connection to a remote machine will involve an Operating System call.**

- Different Operating Systems handle sockets in different ways - but the programmer doesn't need to worry about such details.

Stack based VMs

- Both JVM and . NET CLI are **stack based VMs.**
- Based on **Execution stack that is used to perform operations.**

Stack based VMs

- The byte code generated by compilers for these architectures
- contains a set of instructions that load operand on the stack, perform some operations with them and put the result on the stack.

Register based VMs

- An alternative based on registers.
- Example- **Parrot, A programming Language VM originally designed to support execution of PERL** and then generalized to host the execution of dynamic languages.

Application virtual machine

- A process VM,
  - also called an *application virtual machine*, or
  - *Managed Runtime Environment* (MRE),
  - runs as a normal application inside a host OS and supports a single process.

- It is created when that process is started and destroyed when it exits.
- Its purpose is to provide a platform-independent programming environment and allows a program to execute in the same way on any platform.

User-Application Level

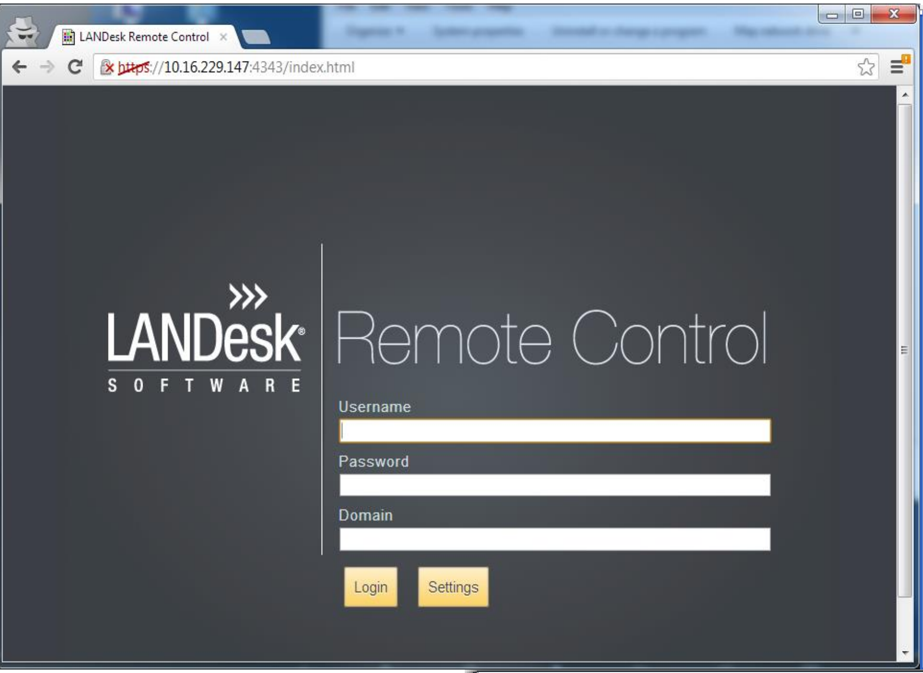
- Other forms of application-level virtualization are:
  - application isolation
  - application streaming.

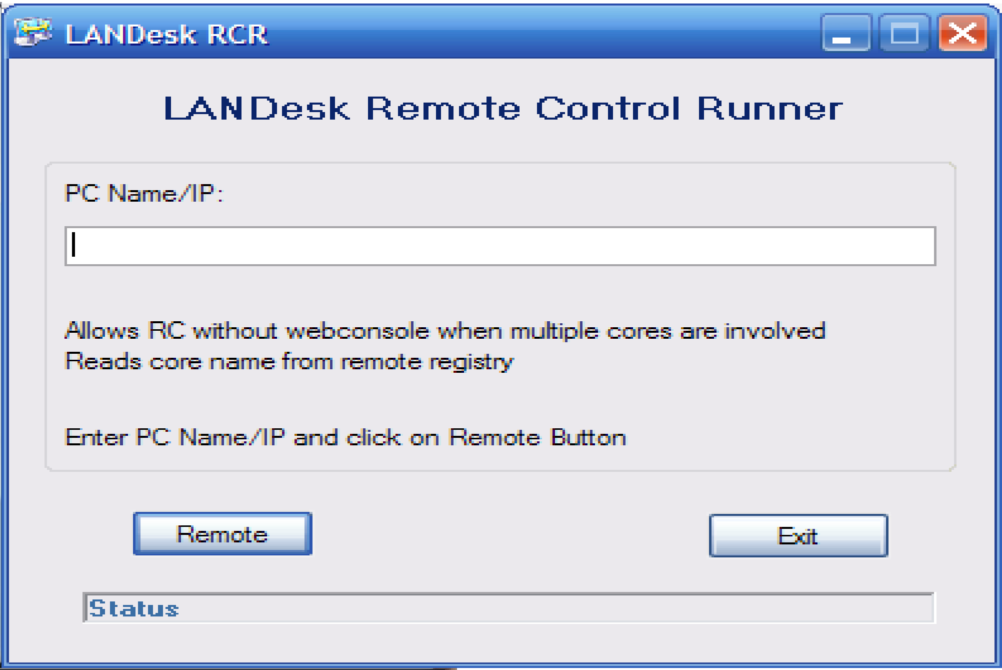
- The process involves wrapping the application in a layer that is isolated from the host OS and other applications.
- **The result is an application that is much easier to distribute and remove from user workstations.**

What is LANDesk?

- An example of application virtualization.

What is LANDesk?





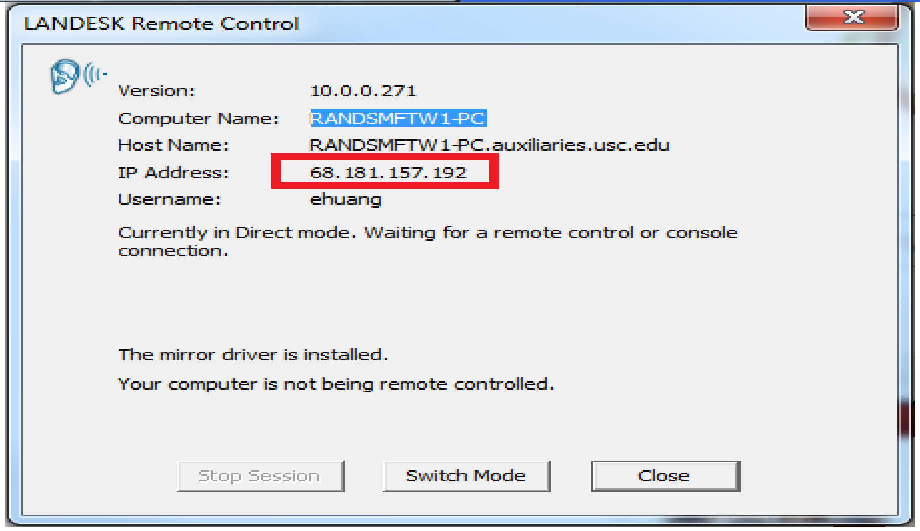
LANDesk RCR

LANDesk Remote Control Runner

PC Name/IP:

Allows RC without webconsole when multiple cores are involved
Reads core name from remote registry

Enter PC Name/IP and click on Remote Button



LANDESK Remote Control

Version:	10.0.0.271
Computer Name:	RANDSMFTW1-PC
Host Name:	RANDSMFTW1-PC.auxiliaries.usc.edu
IP Address:	68.181.157.192
Username:	ehuang

Currently in Direct mode. Waiting for a remote control or console connection.

The mirror driver is installed.
Your computer is not being remote controlled.

What is LANDesk?

- LANDESK's Remote Control capabilities allow information technology departments to minimize downtime by
  - <u>Fixing computer errors and updating systems from a remote location.</u>

- LANDesk is an asset management software system used to
  - <u>remotely inventory and manage desktop computers</u>.
  - <u>Reports on installed software and hardware, allow remote assistance, and install operating system security patches.</u>

User-Application Level

- Advantage: has the best application isolation
- Shortcoming & limitation: low performance, low application flexibility and high implementation complexity.

Relative Merits of Different Approaches

- The number of X's in the table cells reflects the advantage points of each implementation level.
- Five X's implies the best case and one X implies the worst case.

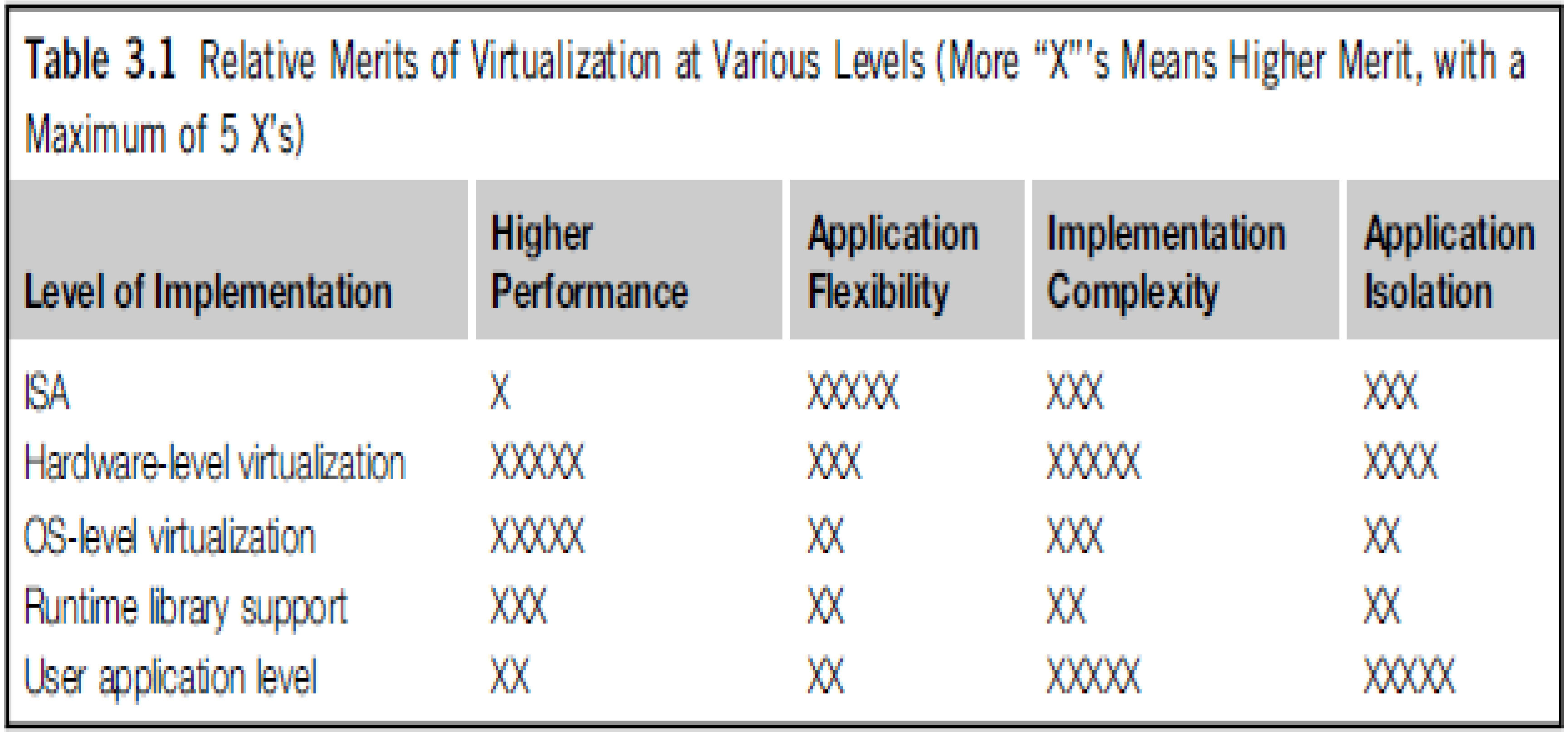


Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

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

Relative Merits of Different Approaches

- Overall, hardware and OS support will yield the highest performance.
- However, the hardware and application levels are also the most expensive to implement.
- User isolation is the most difficult to achieve.
- ISA implementation offers the best application flexibility.

Virtualization Design Requirements

- **Equivalence Requirement**
- **Efficiency Requirement**
- **Resource Control requirement**

Virtualization Design Requirements

Equivalence Requirement

- A m/c that is developed through virtualization must have a logical equivalence with the real m/cs :
  - **Needs to match the capabilities of the physical system in its computational performance.**

How??

Virtualization Design Requirements

Equivalence Requirement

- A m/c that is developed through virtualization must have a logical equivalence with the real m/cs :
  - Needs to match the capabilities of the physical system in its computational performance.
  - **To execute all the applications and programs that are designed to execute on the real m/cs.**

Virtualization Design Requirements

Efficiency Requirement

- While taking the route of Virtualization,
  - **the virtual m/c must be as efficient in its performance as a real system.**

- Virtualization is primarily done with a purpose of getting **efficient software**
  - **without the physical hardware.**

Virtualization Design Requirements

Resource Control requirement

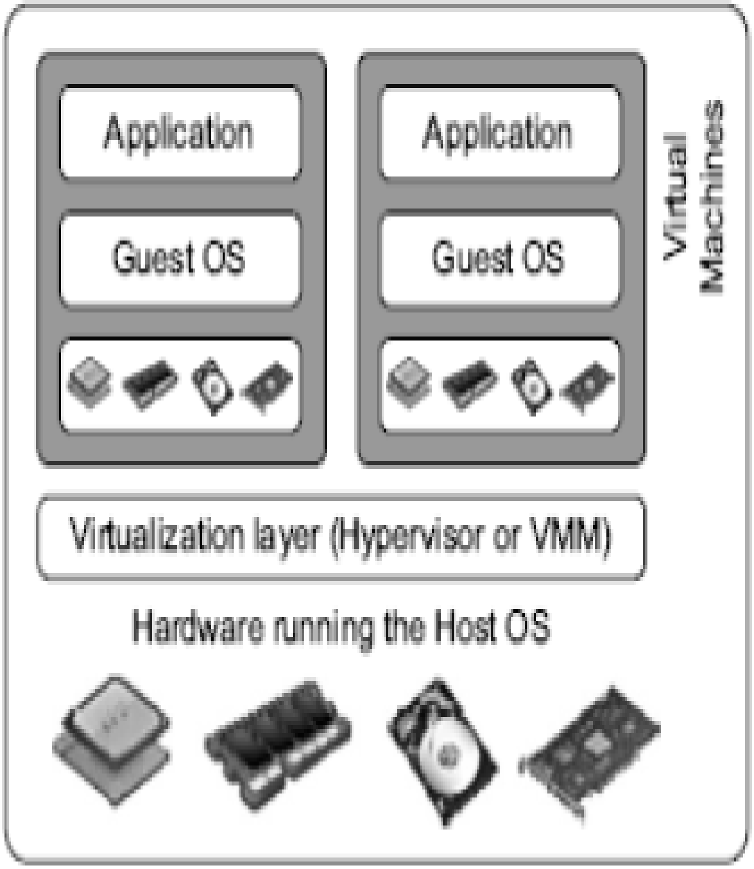
- A typical computer system is a combination of various resources, including processors, memory, and I/O devices.
  - **All these resources must be managed and controlled effectively by the VMM.**
  - The VMM must be in a state of enforcing isolation between the virtualized systems.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS
Hypervisor

- Hypervisor and Xen Architecture
- Binary Translation with Full Virtualization
- Para-Virtualization with Compiler Support

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS
Hypervisor

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


Application
Application
Guest OS
Guest OS
Virtual Machines
Virtualization layer (Hypervisor or VMM)
Hardware running the Host OS

Hypervisors

- A hypervisor is a hardware virtualization technique allowing multiple operating systems, called guests to run on a host machine.
- This is also called the Virtual Machine Monitor (VMM).

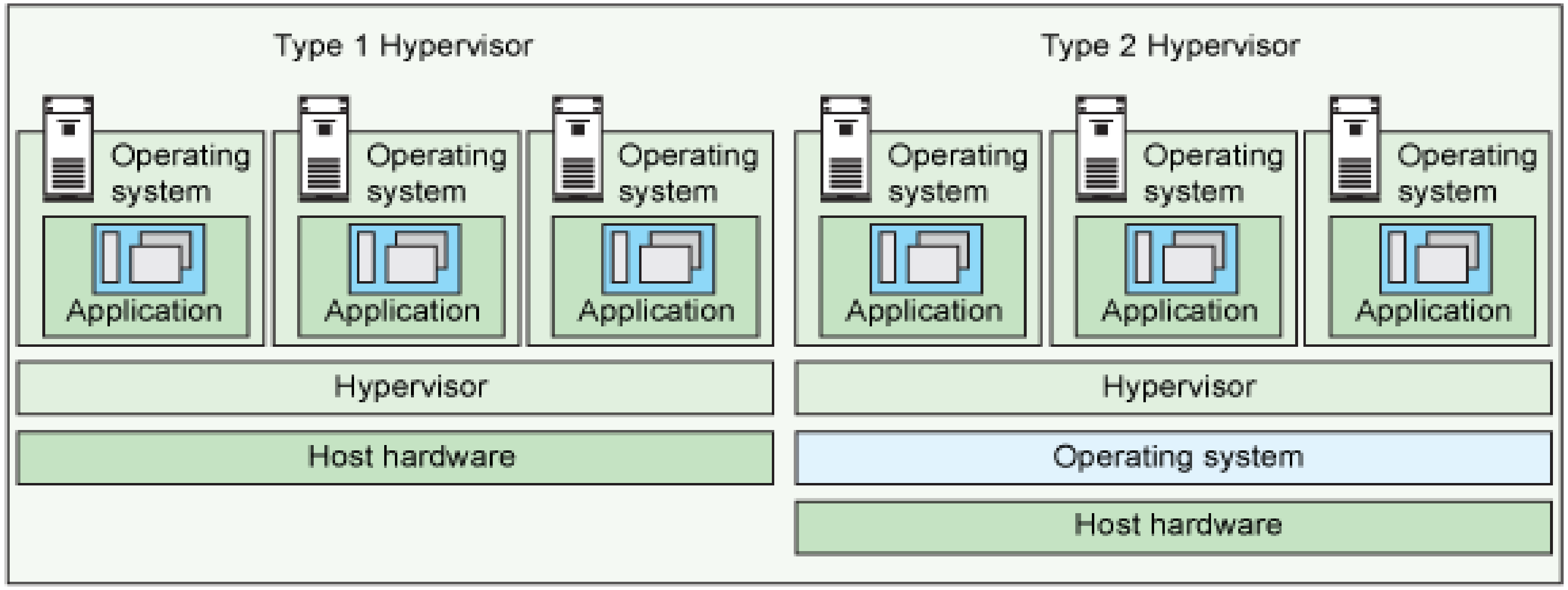
Hypervisors

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

Hypervisors

There are two types of hypervisors:

- Type 1 hypervisor:
  - hypervisors run directly on the system hardware – A “bare metal” embedded hypervisor,
- Type 2 hypervisor:
  - hypervisors run on a host operating system that provides virtualization services, such as I/O device support and memory management.



Type 1 Hypervisors

- Type 1 hypervisor or the bare metal hypervisor
- Sits on the bare metal computer hardware like the CPU, memory, etc.
- All the guest operating systems are a layer above the hypervisor.
- **So the hypervisor is the first layer over the hardware.**
- The original CP/CMS hypervisor developed by IBM was of this kind.
- Examples are Microsoft Hyper-V

Type 2 Hypervisors

- Type 2 or the hosted hypervisor
- Does not run over the bare metal hardware but they run over a host operating system.
- **The hypervisor is the second layer over the hardware.**
- The guest operating systems run a layer over the hypervisor and so they form the third layer.
- Examples are FreeBSD.

Architectures of Hypervisor

- Hypervisor can have 2 architectures:
  - a micro-kernel architecture
  - a monolithic hypervisor architecture

Architectures of Hypervisor

- A micro-kernel hypervisor
  - includes only the basic and unchanging functions (such as physical memory management and processor scheduling).
  - The device drivers and other changeable components are outside the hypervisor.
- A monolithic hypervisor
  - implements all the aforementioned functions, including those of the device drivers.
  - Size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor.

Architectures of Hypervisor

- Micro-kernel architecture like the Microsoft Hyper-V.
- Monolithic hypervisor architecture like the VMware ESX for server virtualization

The Xen Architecture Where & What?

- **Developed by Cambridge University.**
- Xen started as an **Open source Hypervisor project.**
- **Xen is a Type 1 Hypervisor**

The Xen Architecture

- **Xen is a microkernel hypervisor,**
- **Does not include any device drivers natively.**
  - It just provides a mechanism by which a guest OS can have direct access to the physical devices.
- As a result, the size of the Xen hypervisor is kept rather small.

The Xen Architecture

- It separates the policy from the mechanism.

- Xen hypervisor implements all the mechanisms,
  - leaving the policy to be handled by Domain 0

The Xen Architecture
Where & What?

- A base to other virtualization products both
  - commercial and open source.

- Xen currently forms the base of commercial hypervisors of a number of vendors,
  - most notably XenSource by Citrix.

Usage?

- Xen based technology is used for either
  - Desktop virtualization
  - Server virtualization
  - Recently for Cloud computing Solutions with Xen Cloud Platform

How?

- Core technology=Xen Hypervisor

- Primarily Based on Paravirtualization.

- Paravirtualization:
  - Modifying portion of the guest operating system by Xen

- Recently Xen has been advanced to support full virtualization using hardware assisted virtualization.

The Xen Architecture

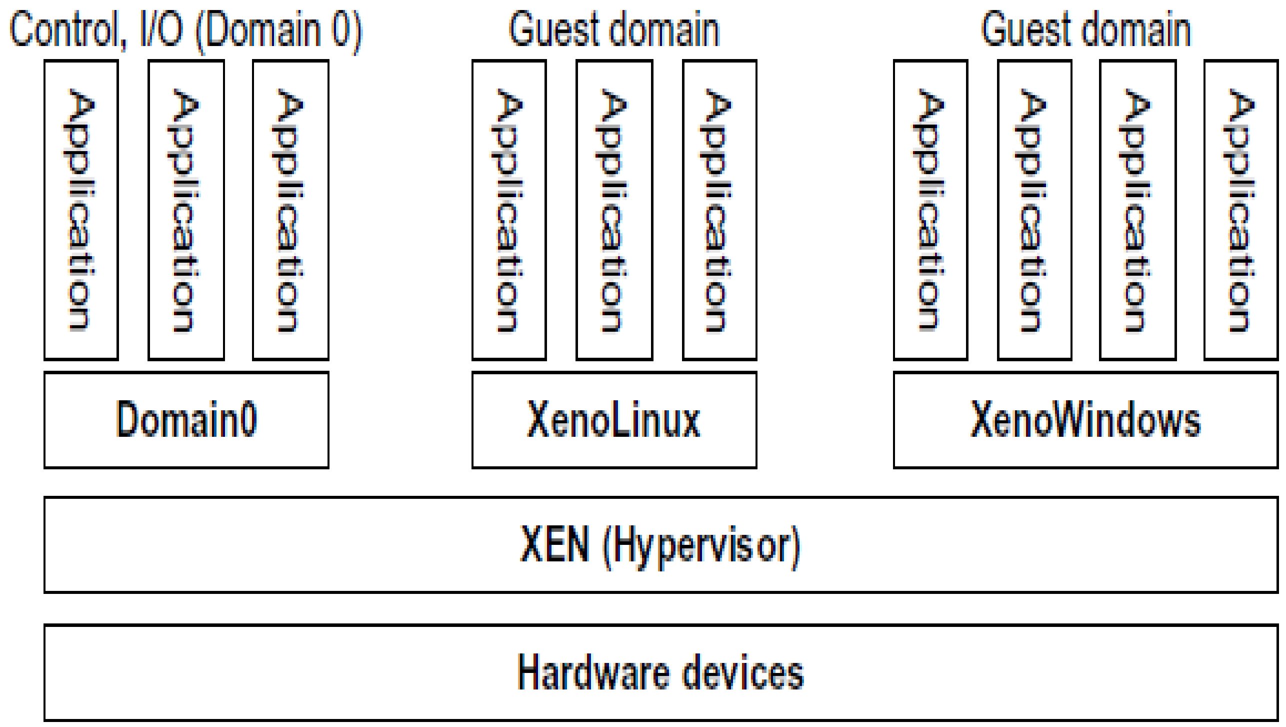


FIGURE 3.5

The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

(Courtesy of P. Barham, et al. [7])

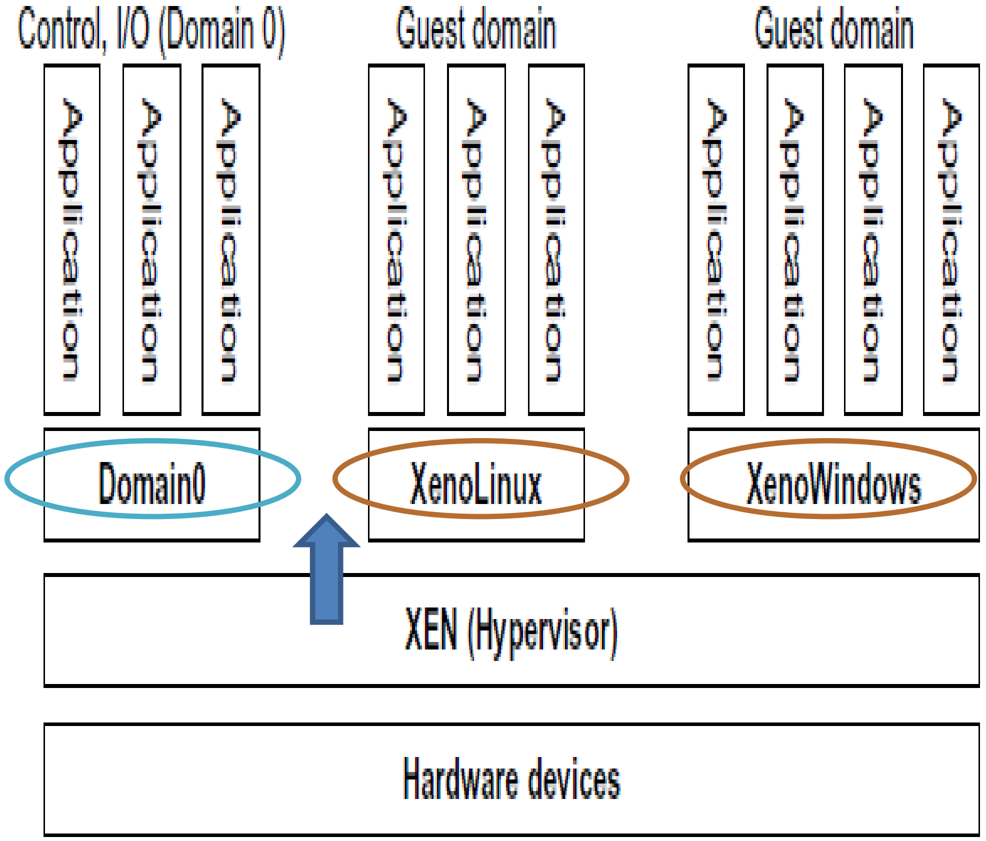
The Xen Architecture

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

- The organization of the three components is important.

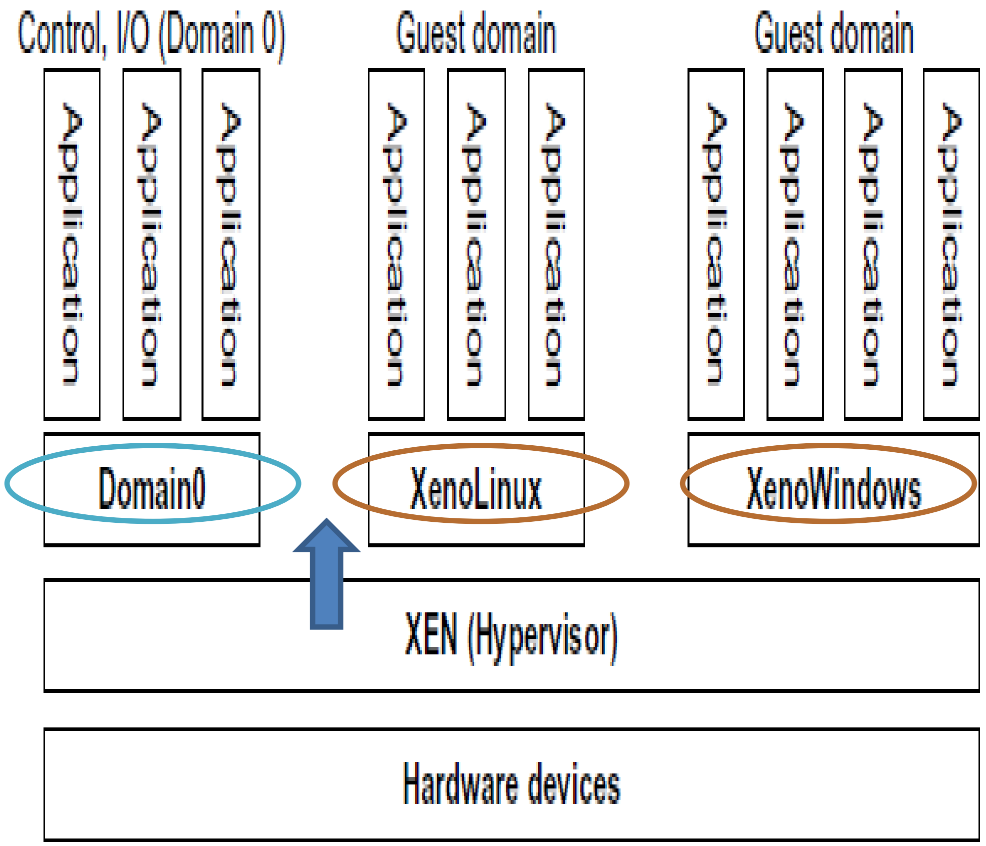
The Xen Architecture

- Many guest OSes can run on top of the hypervisor.
  - Not all guest OSes are created equal, and
  - One in particular controls the others.



The Xen Architecture

- **The guest OS, which has control ability, is called Domain 0**
- **The others are called Domain U.**



What is a Domain?

- Guest operating systems are executed within Domains which represent VMs instances.
- They are of 2 types:
  - Domain O
  - Domain U

Domain 0

- **Specific Control Software**
  - Controls all the other guest operating systems ,
  - is executed in a special domain called Domain 0.

- **Privileged guest OS of Xen**
  - Privileged access to the host

- Also called **Management VM**
  - As it has the privilege to manage other VMs implemented on the same host.

Domain 0

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

How does Domain 0 Work?

- The first one that is loaded once the VMM has been completely booted.
- It Hosts an HTTP server that serves requests for VM creation, configuration, migrate, suspension, resume and termination.
- Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate, and roll back VMs as easily as manipulating a file

Security of Domain 0

- If Domain 0 is compromised, the hacker can control the entire system.

- Security policies are needed to improve the security of Domain 0.

- The Series of security problems during the software life cycle and data lifetime have to be handled.

Domain U

- All the other domains running guest operating systems are called Domain U

HyperCalls

- “The hypervisor provides hypercalls for the guest OSes and applications. “

- Sensitive system calls need to be re-implemented with hypercalls.

- The Xen hypervisor is able to catch the execution of all the sensitive instructions manage them and return the control to the guest operating system by means of a handler.

Full Virtualization with Binary Translation

Full Virtualization

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

Full Virtualization

Why are only critical instructions trapped into the VMM?

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

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

Full Virtualization: Why only CI?

- FV relies on binary translation to trap and to virtualize the execution of certain sensitive, non-virtualizable instructions.

Popek and Goldberg Instructions

- The Popek and Goldberg virtualization requirements are **a set of conditions sufficient for a computer architecture to support system virtualization efficiently.**
- They were introduced by Gerald J. Popek and Robert P. Goldberg in their 1974 article "Formal Requirements for Virtualizable Third Generation Architectures".
- A convenient way of determining whether a computer architecture supports efficient virtualization and provide guidelines for the design of virtualized computer architectures.

Popek and Goldberg Instructions

- Privileged instructions
  - Those that trap if the processor is in user mode and do not trap if it is in system mode (supervisor mode).
- Control sensitive instructions
  - Those that attempt to change the configuration of resources in the system.
- Behavior sensitive instructions
  - Those whose behavior or result depends on the configuration of resources (the content of the relocation register or the processor's mode).

Binary Translation of Guest OS Requests Using a VMM

- VMware puts the VMM at Ring 0 and the guest OS at Ring 1.
- The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions.

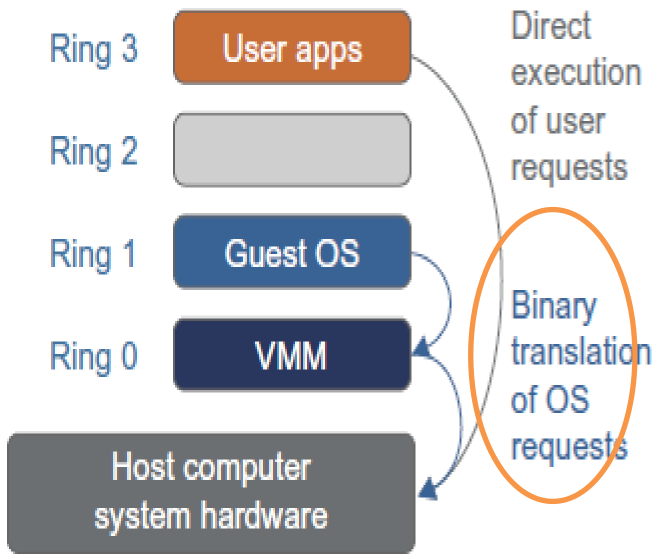

Ring 3
User apps
Direct execution of user requests
Ring 2
Ring 1
Guest OS
Binary translation of OS requests
Ring 0
VMM
Host computer system hardware

FIGURE 3.6

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

(Courtesy of VM Ware [71])

Binary Translation of Guest OS Requests Using a VMM

- They are trapped into the VMM, which emulates the behavior of these instructions.
- The method used in this emulation is called binary translation.

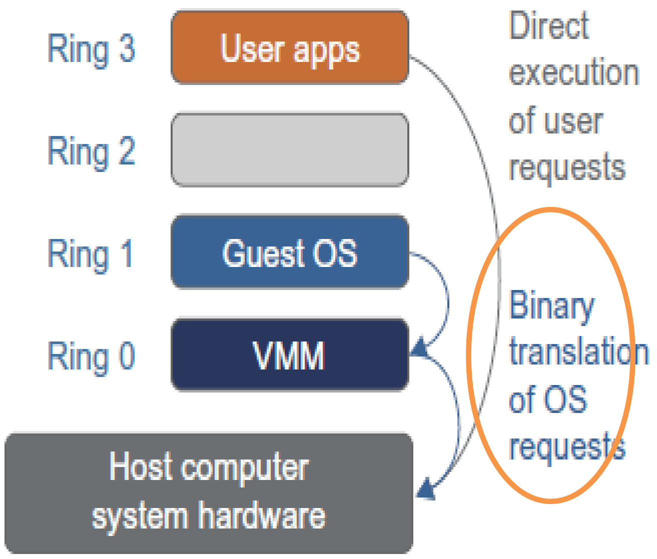

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

FIGURE 3.6

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

(Courtesy of VM Ware [71])

Translation

- The trap triggers the
  - translation of the offending instructions into
  - **an <u>equivalent set of instructions that achieve the same goal without generating exceptions.</u>**

Binary Translation of Guest OS Requests Using a VMM

- **Thus, full virtualization combines binary translation and direct execution.**

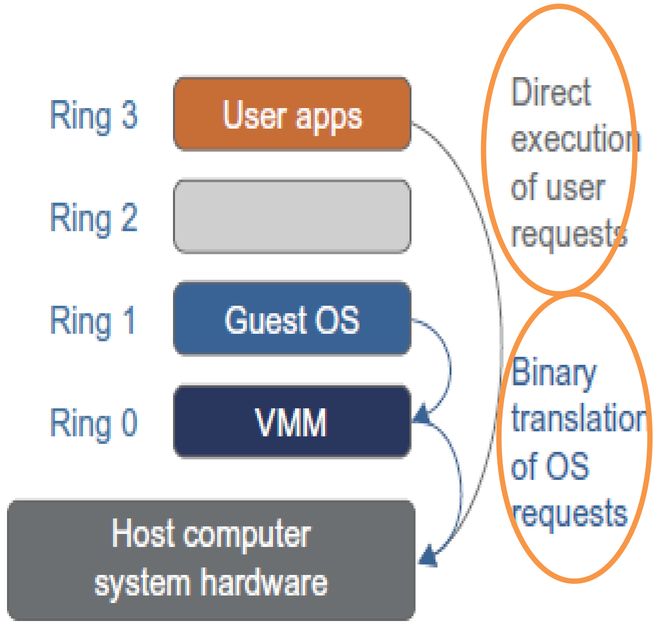

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

FIGURE 3.6

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

(Courtesy of VM Ware [71])

Binary Translation of Guest OS Requests Using a VMM

- **The guest OS is completely decoupled from the underlying hardware.**

- **The guest OS is unaware that it is being virtualized.**

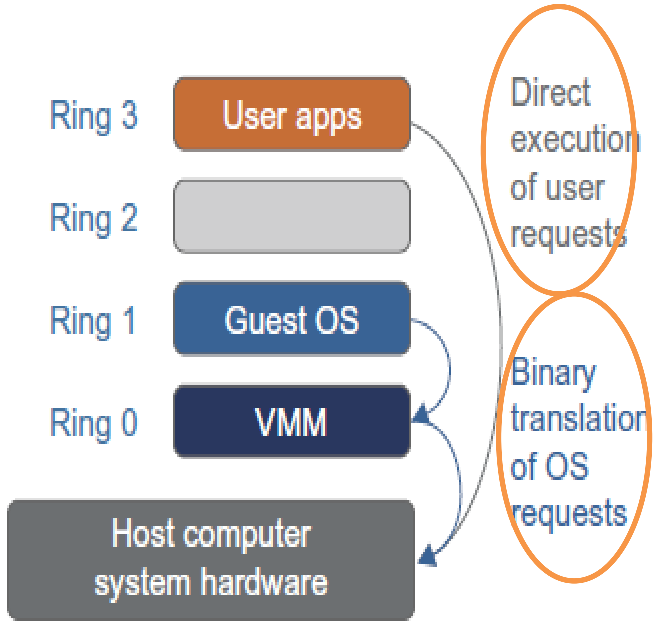

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

FIGURE 3.6

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

(Courtesy of VM Ware [71])

Full Virtualization

- Full virtualization does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation.

Advantage –

- No need to modify OS.
- However, this approach of binary translation slows down the performance a lot.
- Also binary translation can incur a large performance overhead.

Caching

- Binary translation employs a code cache
- To store translated hot instructions to improve performance,
- To improve efficiency,
  - the equivalent set of instruction is cached,
  - so that for further occurrences of the same instructions, translation is not necessary anymore
- But it increases the cost of memory usage.

Advantage ?

Disadvantage ?

Advantage

- Guests can run unmodified in a virtualized environment.
- Binary translation is only applied to a subset of the instruction set, while the others are managed through direct execution on the underlying hardware.
- This reduces the impact on performance of binary translation

Disadvantage

- Translating instructions at runtime introduces additional overhead that is not present in other approaches like para-virtualization and hardware assisted virtualization.

- Rather time-consuming, In particular for the full virtualization of I/O-intensive applications.

- The code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage.

Example of FV

VMWARE

- Full Virtualization.

- Underlying Hardware is made available to the guest operating System which
  - runs unaware of such abstraction layer and
  - does not need to be modified.

- Guest OS is <u>VMM unaware</u>

Para-Virtualization with Compiler Support

Para-Virtualization with Compiler Support

- Para-virtualization needs to **modify the guest operating systems.**

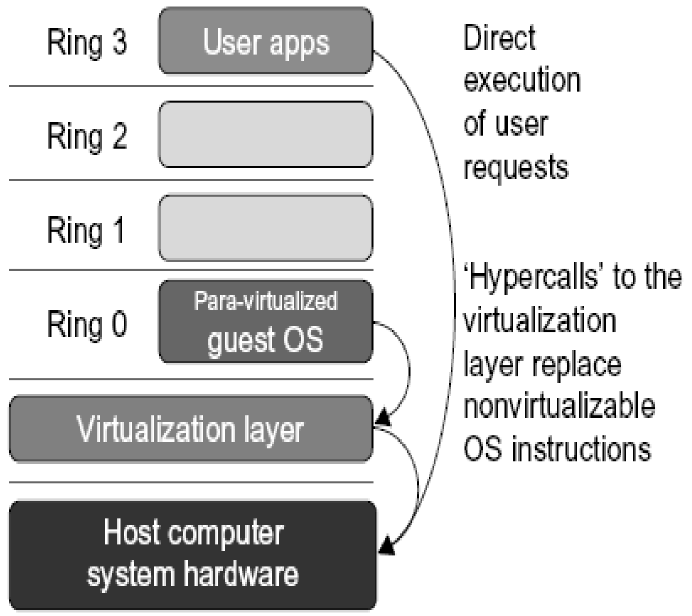
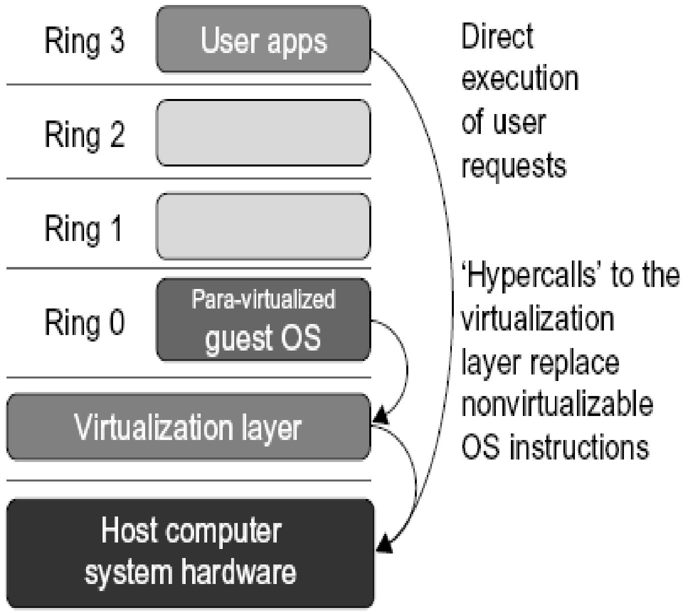

Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

FIGURE 3.8

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

Para-Virtualization with Compiler Support

- A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications.
- However, when the guest OS kernel is modified for virtualization**, it can no longer run on the hardware directly.**



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

FIGURE 3.8

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

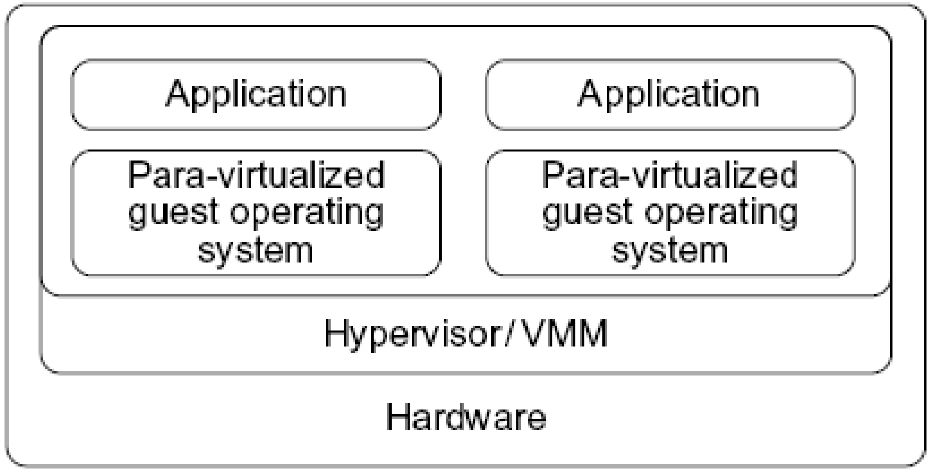
Performance Issue

- Performance degradation is a critical issue of a virtualized system.
  - No one wants to use a VM if it is much slower than using a physical machine.

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

Para-Virtualization: Intelligent Compiler

- The guest operating systems are para-virtualized.
- They are assisted by an **intelligent compiler to replace the non-virtualizable OS instructions by hypercalls** that communicate directly with the hypervisor or VMM.



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

FIGURE 3.7

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process (See Figure 3.8 for more details).

Hypercall

- Like syscall is to a kernel.
- Hypercall is to a hypervisor
- A hypercall is a software trap from a domain to the hypervisor, just as a syscall is a software trap from an application to the kernel.
- All are the same thing: a system call to the hypervisor below.
- Such calls require support in the "guest" operating system, which has to have hypervisor-specific code to make such calls.

Full Virtualization vs. Para-Virtualization

- Full virtualization does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation.

- On the other hand, para virtualization needs to modify guest OS, and non-virtualizable instructions are replaced by hypercalls that communicate directly with the hypervisor or VMM.
- Para-Virtualization with Compiler Support

W
O
R
K
I
N
G

Full Virtualization vs. Para-Virtualization

- As of full virtualization, the advantage is no need to modify OS. However, this approach of binary translation slows down the performance a lot.

- Para virtualization, reduce the virtualization overhead, and improve performance but the cost of maintaining paravirtualized OS is high. The improvement depends on the workload.

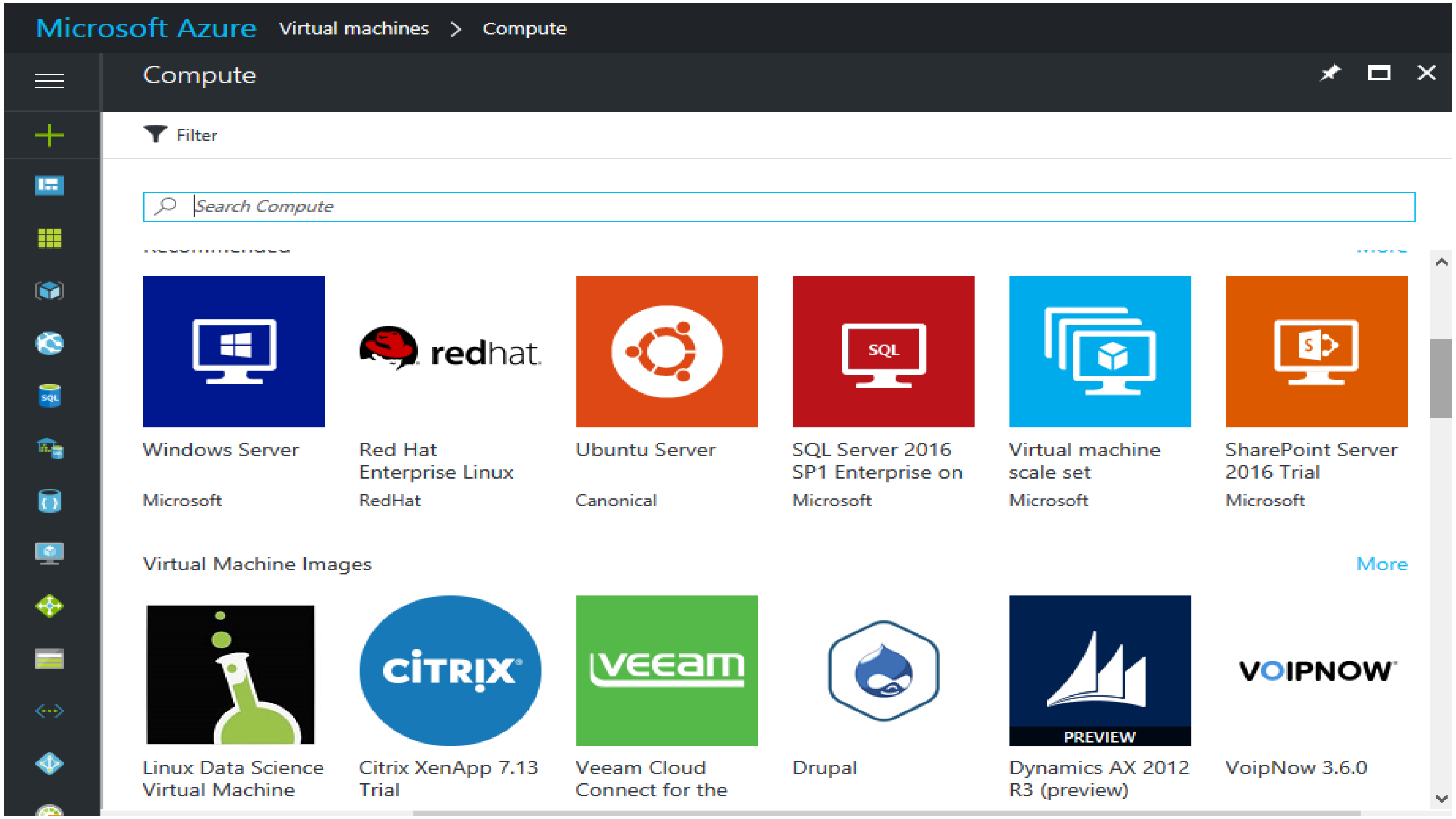
Microsoft Hyper-V

- Hyper-V is an infrastructure virtualization solution developed by Microsoft for server virtualization.
- As the name recalls, it uses a hypervisor-based approach to hardware virtualization, which leverages several techniques to support a variety of guest operating systems.
- Type 1 Hypervisor

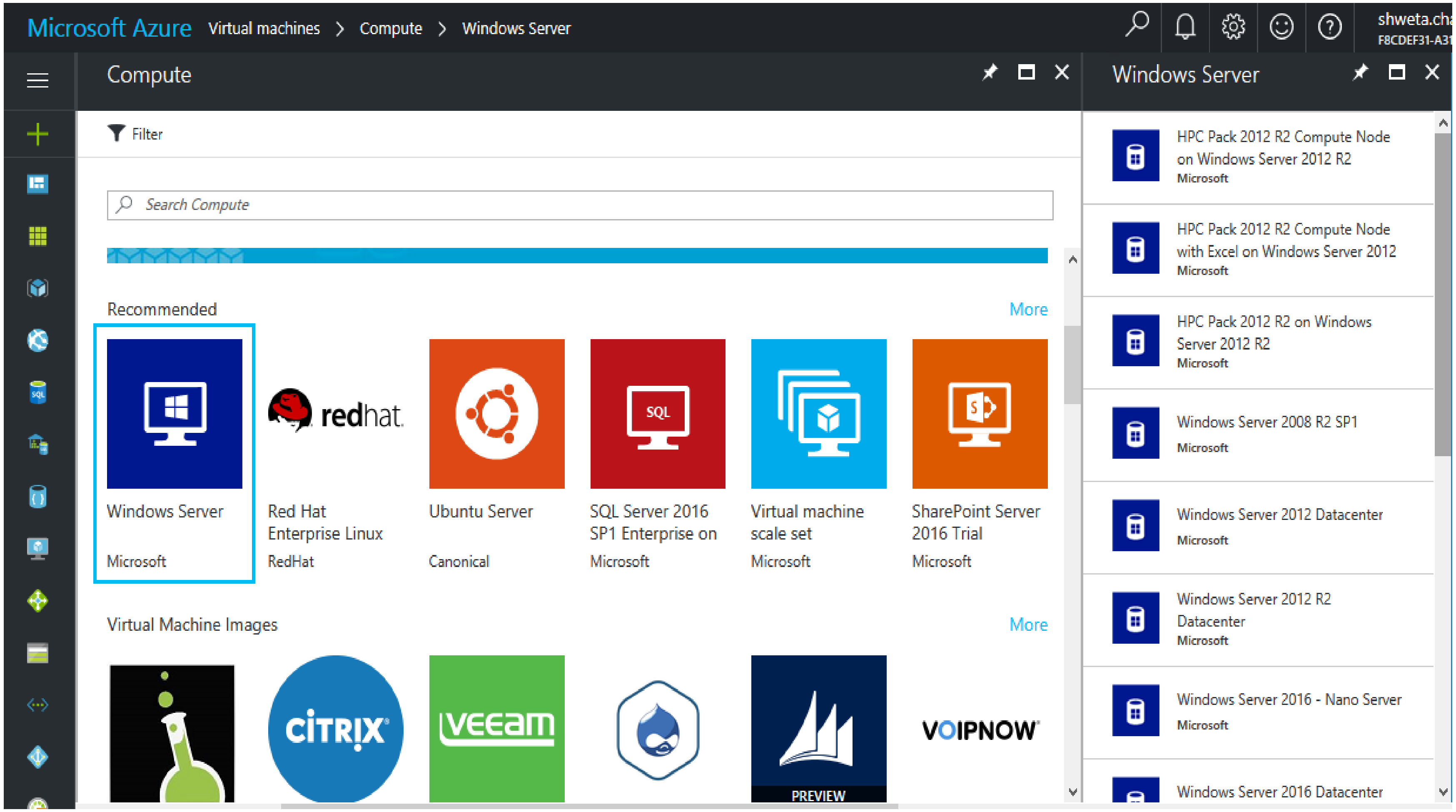
Microsoft Hyper-V

- Hyper-V is currently shipped as a component of Windows Server 2008 R2 that installs the hypervisor as a role within the server.

Eg-Selecting a server followed by OS in Microsoft Azure



Eg-Selecting a server followed by OS in Microsoft Azure



Microsoft Azure Virtual machines > Compute > Windows Server

shweta.ch
F8CDEF31-A3

Compute

Filter

Search Compute

Recommended

More

Windows Server		Microsoft
Red Hat Enterprise Linux		RedHat
Ubuntu Server		Canonical
SQL Server 2016 SP1 Enterprise on		Microsoft
Virtual machine scale set		Microsoft
SharePoint Server 2016 Trial		Microsoft

Virtual Machine Images

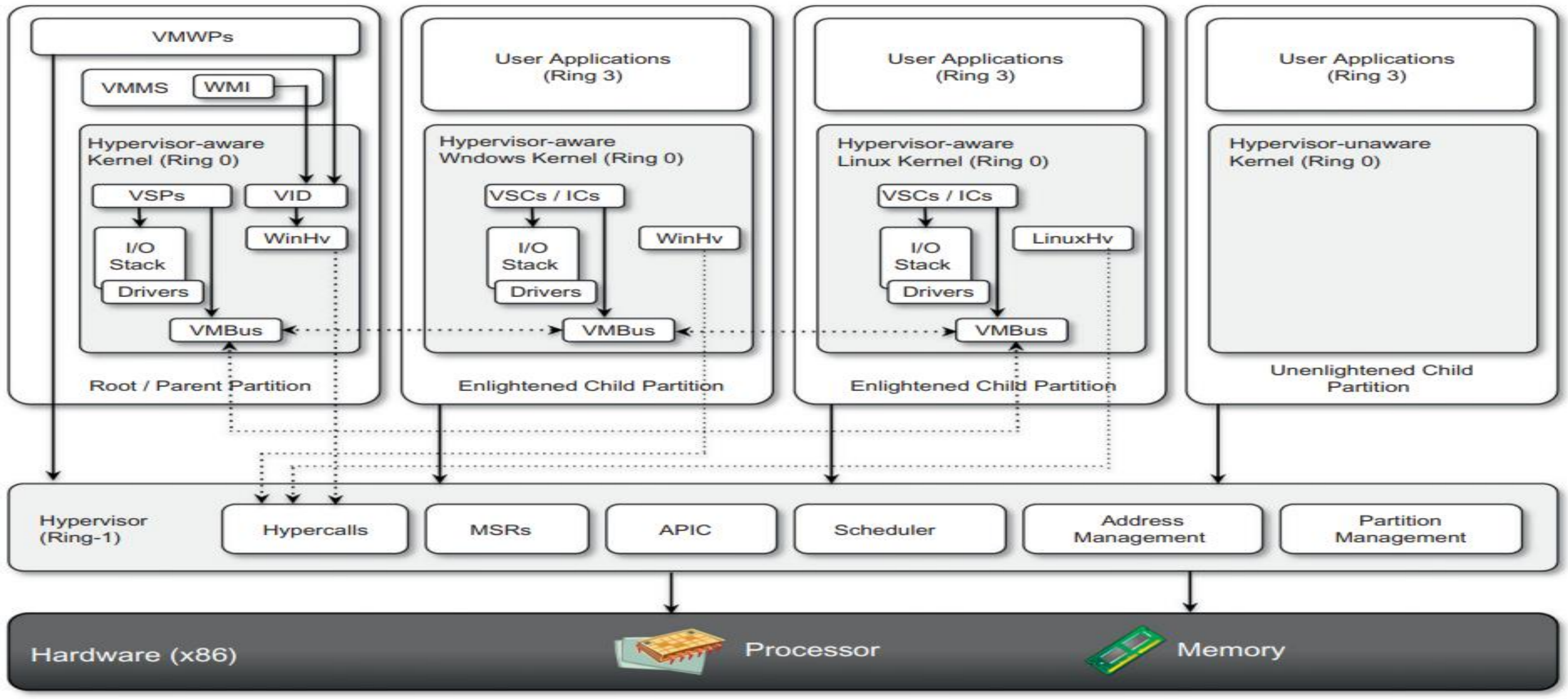
More

Windows Server

HPC Pack 2012 R2 Compute Node on Windows Server 2012 R2	Microsoft
HPC Pack 2012 R2 Compute Node with Excel on Windows Server 2012	Microsoft
HPC Pack 2012 R2 on Windows Server 2012 R2	Microsoft
Windows Server 2008 R2 SP1	Microsoft
Windows Server 2012 Datacenter	Microsoft
Windows Server 2012 R2 Datacenter	Microsoft
Windows Server 2016 - Nano Server	Microsoft
Windows Server 2016 Datacenter	

Architecture

- 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
- Figure provides an overview of the architecture of Hyper-V.

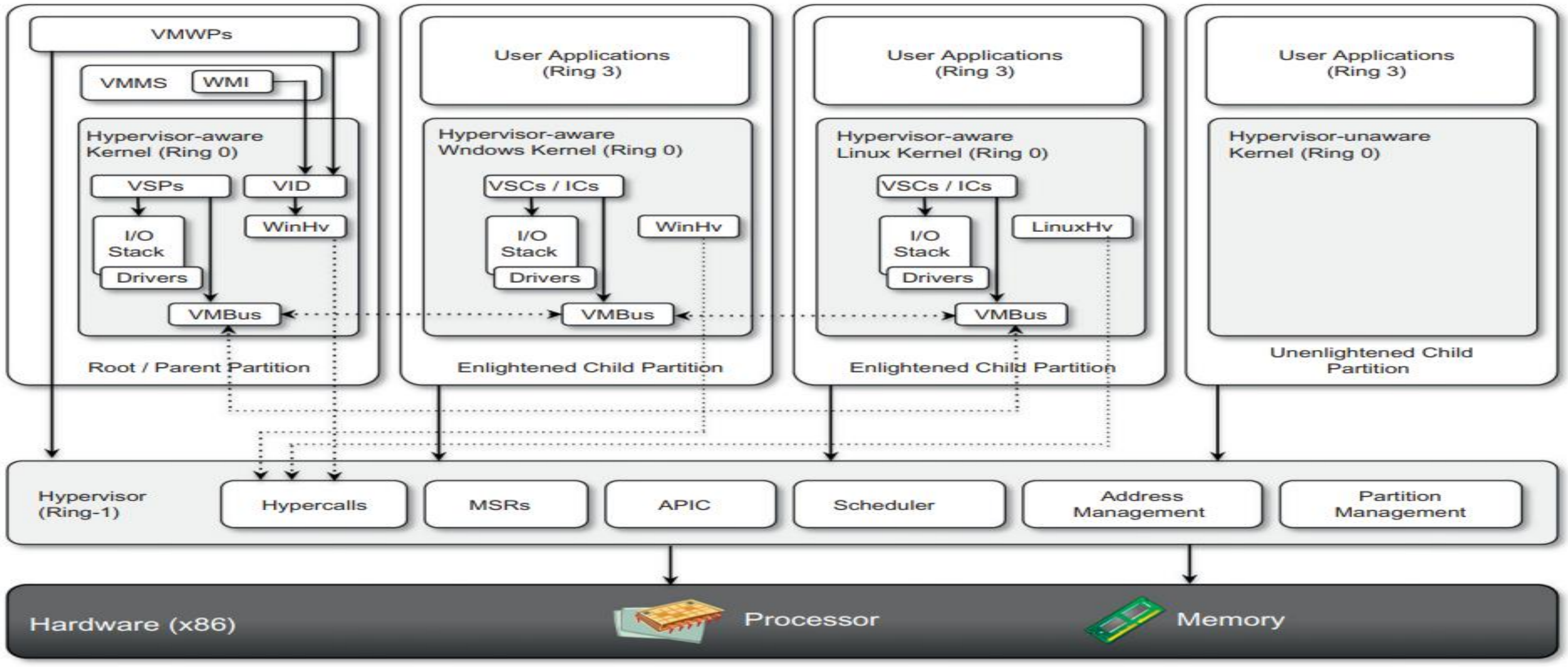


VMWPs
VMMS
WMI
Hypervisor-aware Kernel (Ring 0)
VSPs
VID
I/O Stack
Drivers
WinHv
VMBus
Root / Parent Partition
User Applications (Ring 3)
Hypervisor-aware Wndows Kernel (Ring 0)
VSCs / ICs
I/O Stack
Drivers
WinHv
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-aware Linux Kernel (Ring 0)
VSCs / ICs
I/O Stack
Drivers
LinuxHv
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-unaware Kernel (Ring 0)
Unenlightened Child Partition
Hypervisor (Ring-1)
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Management
Hardware (x86)
Processor
Memory

FIGURE 3.17
Microsoft Hyper-V architecture.

Architecture

- Despite its straightforward installation as a component of the host operating system,
- Hyper-V takes control of the hardware, and the host operating system becomes a virtual machine instance with special privileges, called the parent partition.



VMWPs
VMMS
WMI
Hypervisor-aware Kernel (Ring 0)
VSPs
VID
WinHv
I/O Stack
Drivers
VMBus
Root / Parent Partition
User Applications (Ring 3)
Hypervisor-aware Wndows Kernel (Ring 0)
VSCs / ICs
WinHv
I/O Stack
Drivers
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-aware Linux Kernel (Ring 0)
VSCs / ICs
LinuxHv
I/O Stack
Drivers
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-unaware Kernel (Ring 0)
Unenlightened Child Partition
Hypervisor (Ring-1)
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Management
Hardware (x86)
Processor
Memory

FIGURE 3.17
Microsoft Hyper-V architecture.

Architecture

- The parent partition (also called the root partition)
- Is the only one that has direct access to the hardware.
- It runs the virtualization stack, hosts all the drivers required to configure guest operating systems, and creates child partitions through the hypervisor.

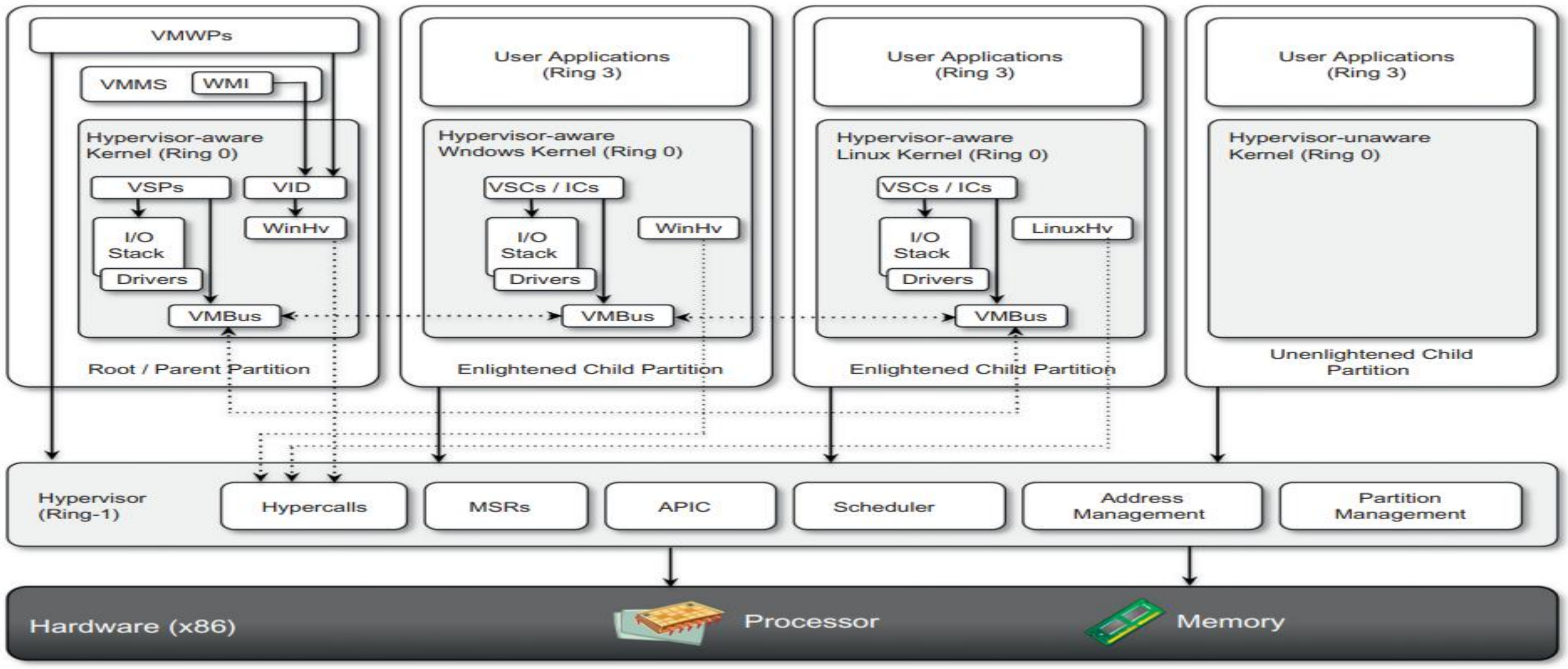

VMWPs
VMMS
WMI
Hypervisor-aware Kernel (Ring 0)
VSPs
VID
I/O Stack
WinHv
Drivers
VMBus
Root / Parent Partition
User Applications (Ring 3)
Hypervisor-aware Wndows Kernel (Ring 0)
VSCs / ICs
I/O Stack
WinHv
Drivers
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-aware Linux Kernel (Ring 0)
VSCs / ICs
I/O Stack
LinuxHv
Drivers
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-unaware Kernel (Ring 0)
Unenlightened Child Partition
Hypervisor (Ring-1)
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Management
Hardware (x86)
Processor
Memory

FIGURE 3.17
Microsoft Hyper-V architecture.

Architecture

- Child partitions are used to host guest operating systems and do not have access to the underlying hardware,
- but their interaction with it is controlled by either the parent partition or the hypervisor itself.

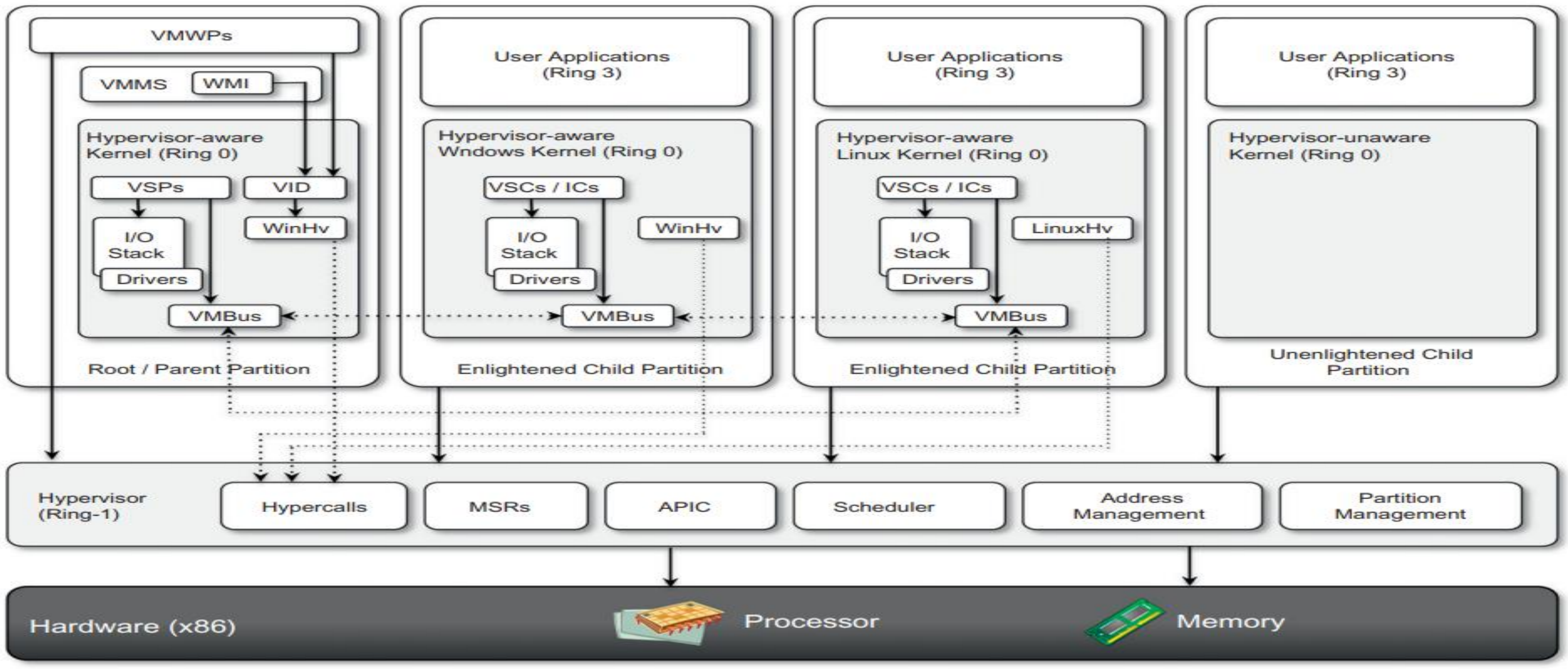
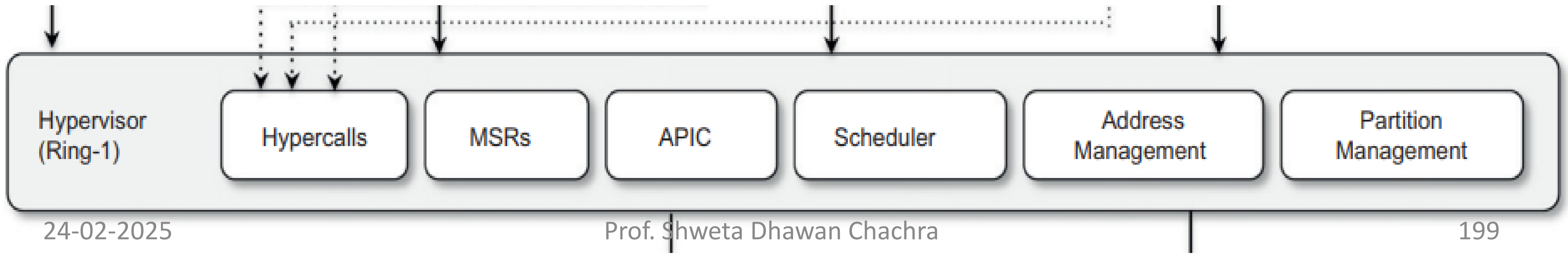

VMWPs
VMMS
WMI
Hypervisor-aware Kernel (Ring 0)
VSPs
VID
I/O Stack
Drivers
WinHv
VMBus
Root / Parent Partition
User Applications (Ring 3)
Hypervisor-aware Wndows Kernel (Ring 0)
VSCs / ICs
I/O Stack
Drivers
WinHv
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-aware Linux Kernel (Ring 0)
VSCs / ICs
I/O Stack
Drivers
LinuxHv
VMBus
Enlightened Child Partition
User Applications (Ring 3)
Hypervisor-unaware Kernel (Ring 0)
Unenlightened Child Partition
Hypervisor (Ring-1)
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Management
Hardware (x86)
Processor
Memory

FIGURE 3.17
Microsoft Hyper-V architecture.

Microsoft Hyper-V Hypervisor

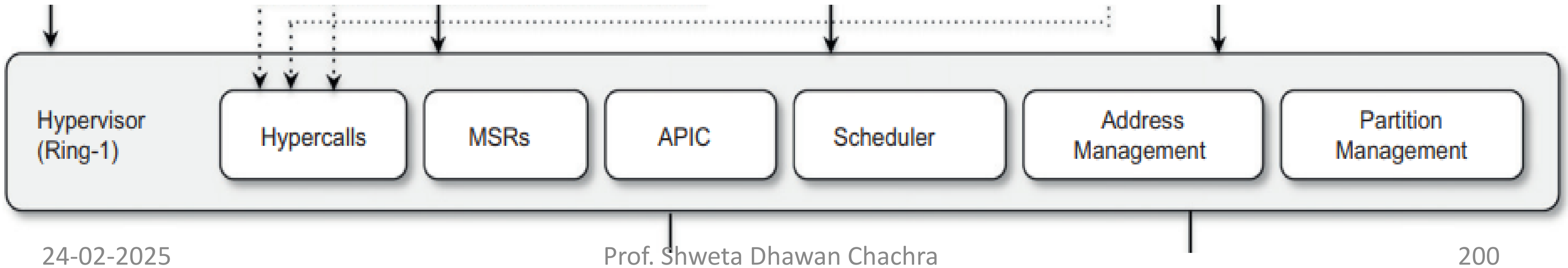
- The hypervisor is the component that directly manages the underlying hardware (processors and memory).
- It is logically defined by the following components:
  - **Hypercalls interface-**
  - **Memory service routines (MSRs)-**
  - **Advanced programmable interrupt controller (APIC).**
  - **Scheduler-**
  - **Address manager-**
  - **Partition manager-**



Microsoft Hyper-V Hypervisor

Hypercalls interface-

- This is the entry point for all the partitions for the execution of sensitive instructions.
- This is an implementation of the paravirtualization approach



Microsoft Hyper-V Hypervisor

Hypercalls interface-

- This interface is used by drivers in the partitioned operating system to contact the hypervisor using the standard Windows calling convention.
- The parent partition also uses this interface to create child partitions.

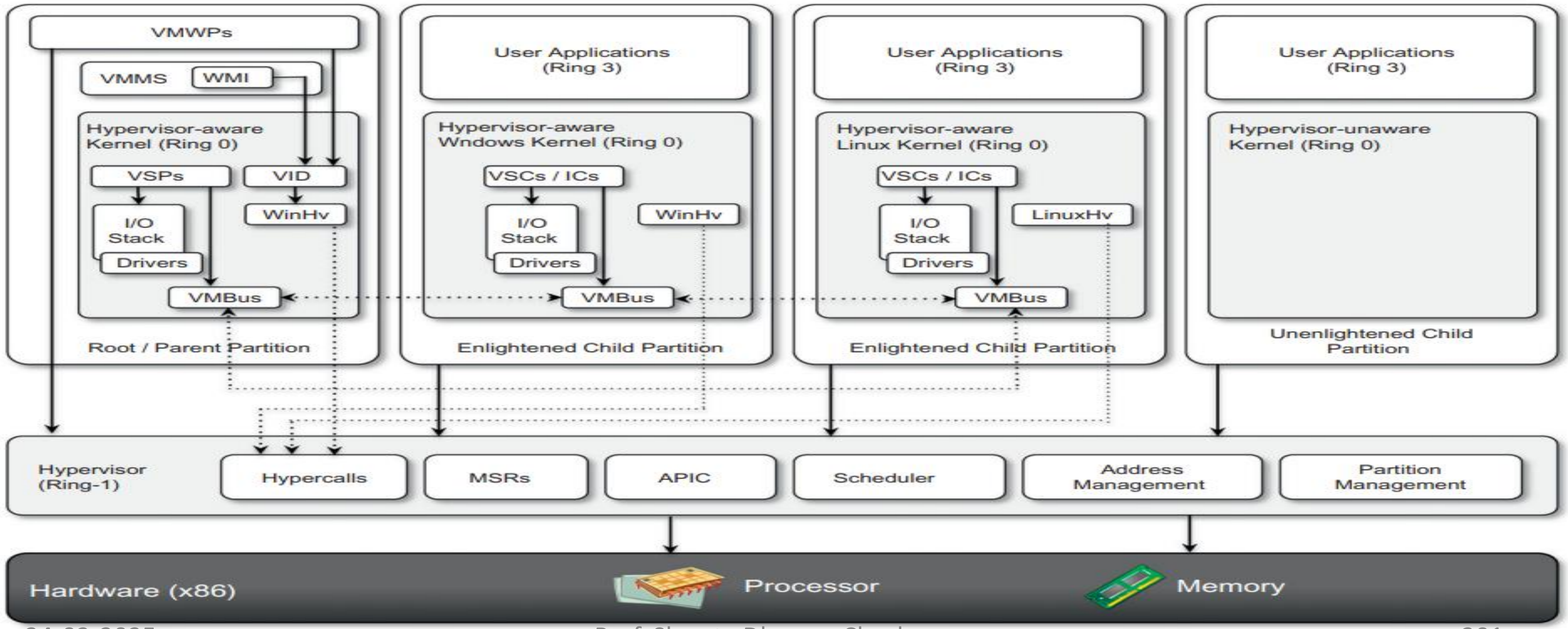

VMWPs
VMMS
WMI
Hypervisor-aware Kernel (Ring 0)
VSPs
VID
I/O Stack
Drivers
WinHv
VMBus
Root / Parent Partition
User Applications (Ring 3)
Hypervisor-aware Wndows Kernel (Ring 0)
VSCs / ICs
WinHv
VMBus
Enlightened Child Partition
Hypervisor-aware Linux Kernel (Ring 0)
LinuxHv
Hypervisor-unaware Kernel (Ring 0)
Unenlightened Child Partition
Hypervisor (Ring-1)
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Management
Hardware (x86)
Processor
Memory

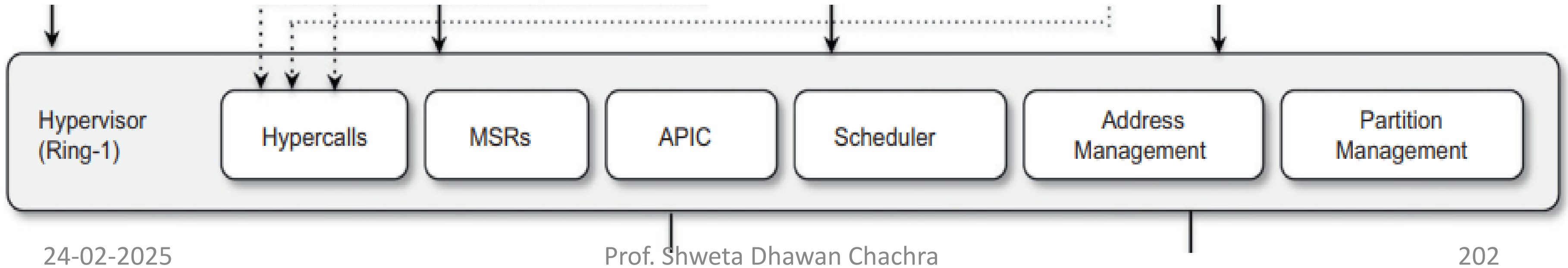
FIGURE 3.17

Microsoft Hyper-V architecture.

Microsoft Hyper-V Hypervisor

Memory service routines (MSRs)-

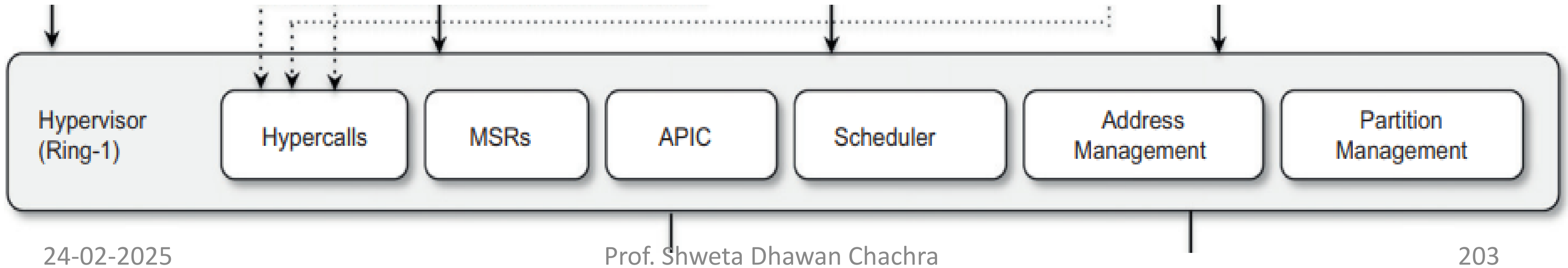
- These are the set of functionalities that control the memory and its access from partitions.
- By leveraging hardware-assisted virtualization, the hypervisor uses the Input/Output Memory Management Unit (I/O MMU or IOMMU) to fast-track access to devices from partitions by translating virtual memory addresses.



Microsoft Hyper-V Hypervisor

Scheduler-

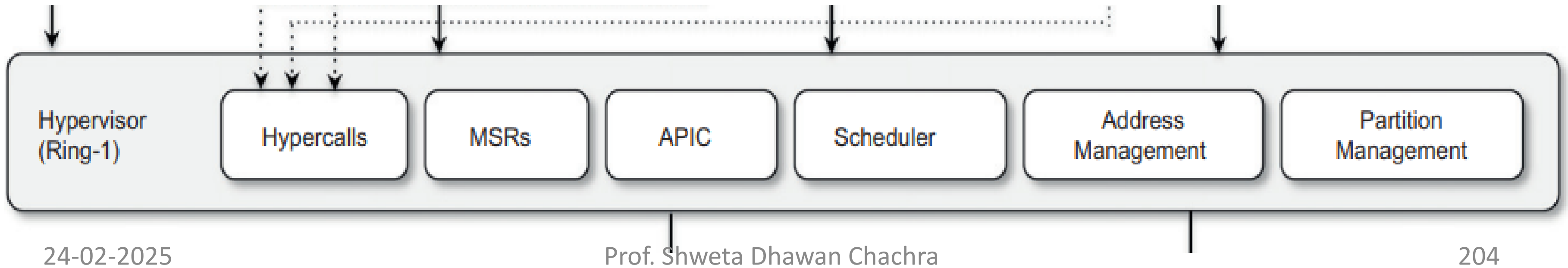
- This component **schedules the virtual processors to run on available physical processors.**
- The scheduling is **controlled by policies that are set by the parent partition.**



Microsoft Hyper-V Hypervisor

Address manager-

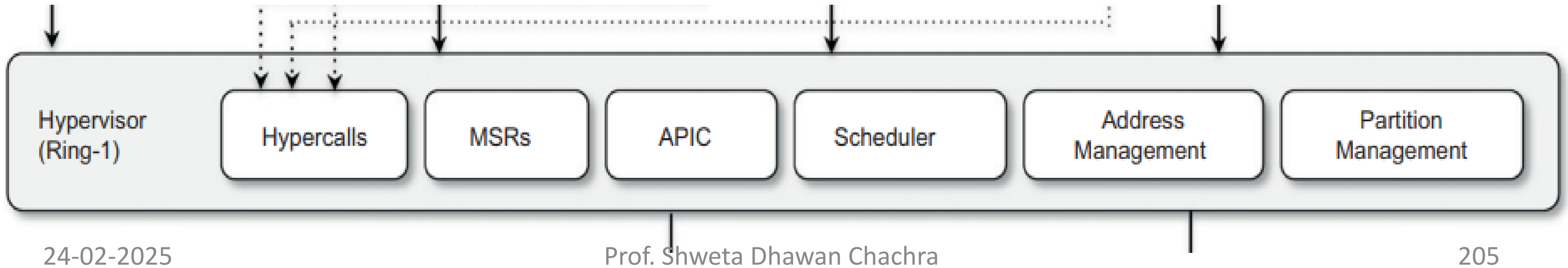
- This component is used to **manage the virtual network addresses that are allocated to each guest operating system.**



Microsoft Hyper-V Hypervisor

Partition manager-

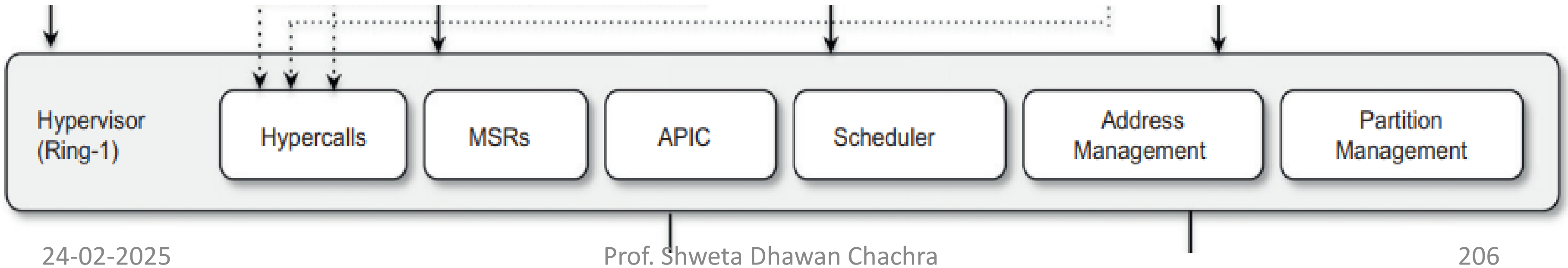
- This component is in charge of performing partition
- **creation,**
- **finalization,**
- **destruction,**
- **enumeration, and**
- **configurations.**



Microsoft Hyper-V Hypervisor

Advanced programmable interrupt controller (APIC)-

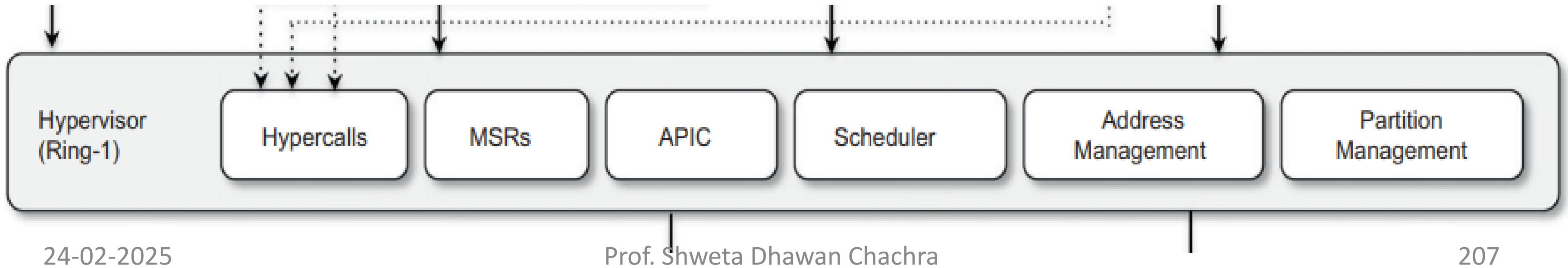
- This component represents the **interrupt controller**, which manages the **signals coming from the underlying hardware** when some event occurs
  - **timer expired,**
  - **I/O ready,**
  - **exceptions and**
  - **traps**



Microsoft Hyper-V Hypervisor

Advanced programmable interrupt controller (APIC)-

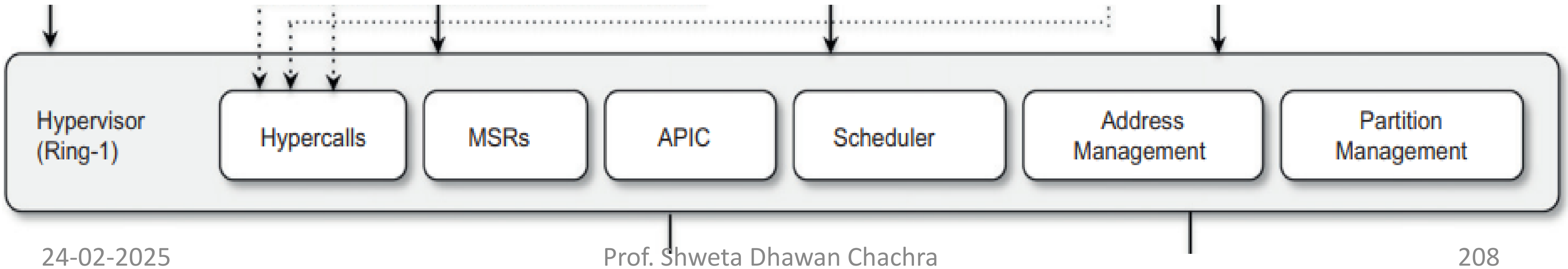
- **Each virtual processor is equipped with a synthetic interrupt controller (SynIC), which constitutes an extension of the local APIC.**



Microsoft Hyper-V Hypervisor

Advanced programmable interrupt controller (APIC)-

- **The hypervisor is responsible of dispatching the physical interrupts to the synthetic interrupt controllers.**
- Its services are available through the hypercalls interface API previously discussed.



Microsoft Hyper-V Hypervisor

- The hypervisor runs in Ring -1
- Therefore requires corresponding hardware technology that enables such a condition.

Microsoft Hyper-V Hypervisor

- By executing in this highly privileged mode, the hypervisor can support legacy operating systems that have been designed for x86 hardware.
- **Operating systems of newer generations can take advantage of the new specific architecture of Hyper-V** especially for the I/O operations performed by child partitions

VMware

- VMware is a pioneer in virtualization technology
- VMware offers a collection of virtualization solutions covering the entire range of the market, **from desktop computing to enterprise computing and infrastructure virtualization.**

VMware

- VMware’s technology is based on the concept of full virtualization
- VMware implements full virtualization
  1) Either in the desktop environment, by means of Type II hypervisors, or
  2) In the server environment, by means of Type I hypervisors.
- In both cases, full virtualization is made possible by means of
  1) direct execution (for non-sensitive instructions) and
  2) binary translation (for sensitive instructions),

End-user (desktop) virtualization

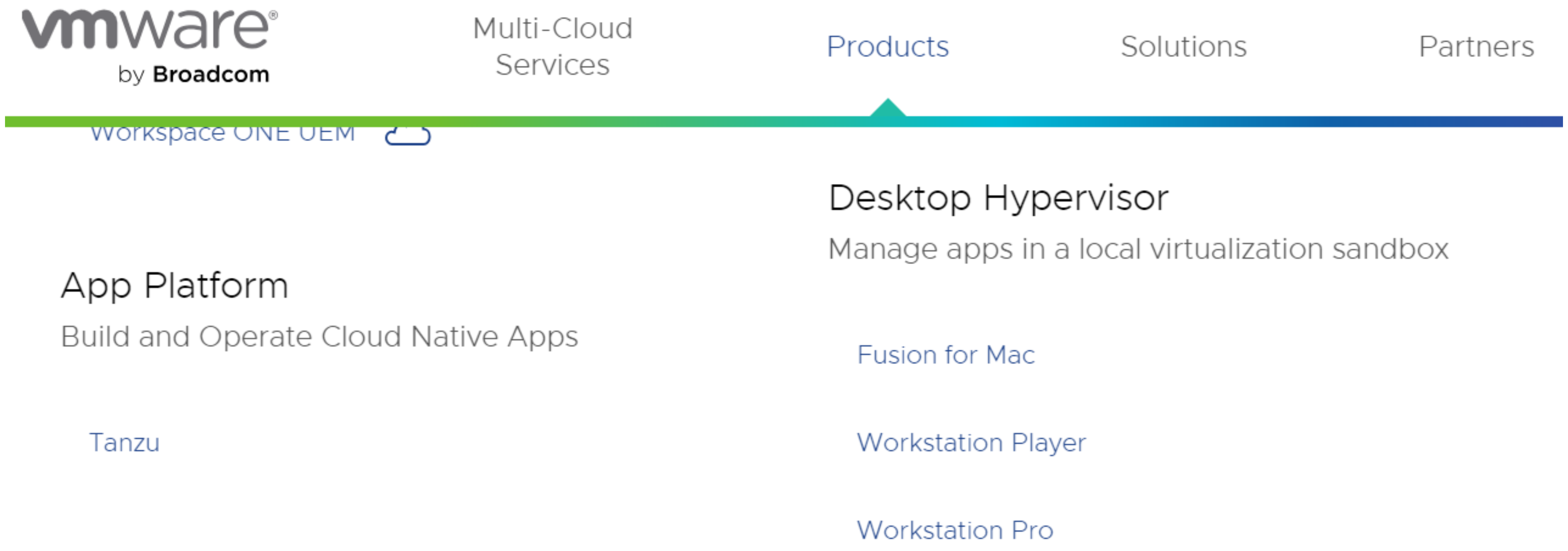
- VMware supports virtualization of operating system environments on enduser computers.

End-user (desktop) virtualization

Specific VMware software—

1. **VMware Workstation**, for Windows operating systems, and
2. **VMware Fusion**, for Mac OS X environments

- Are installed in the host operating system
- To create virtual machines and manage their execution.



End-user (desktop) virtualization

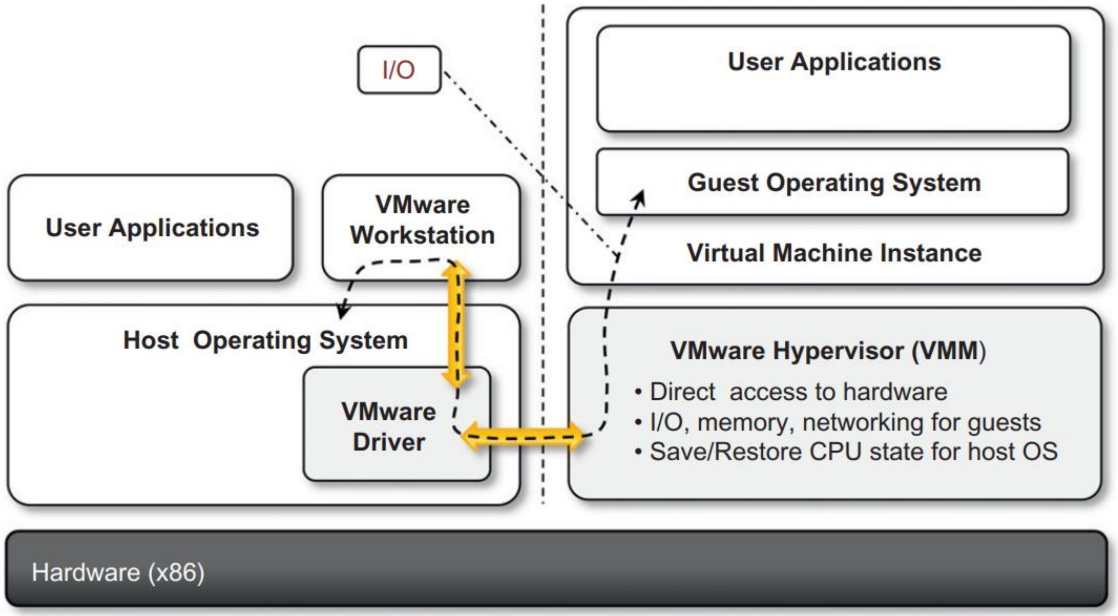


FIGURE 3.13

VMware workstation architecture.

- Virtualization is done by installing a specific driver in the host operating system that provides two main services:

End-user (desktop) virtualization

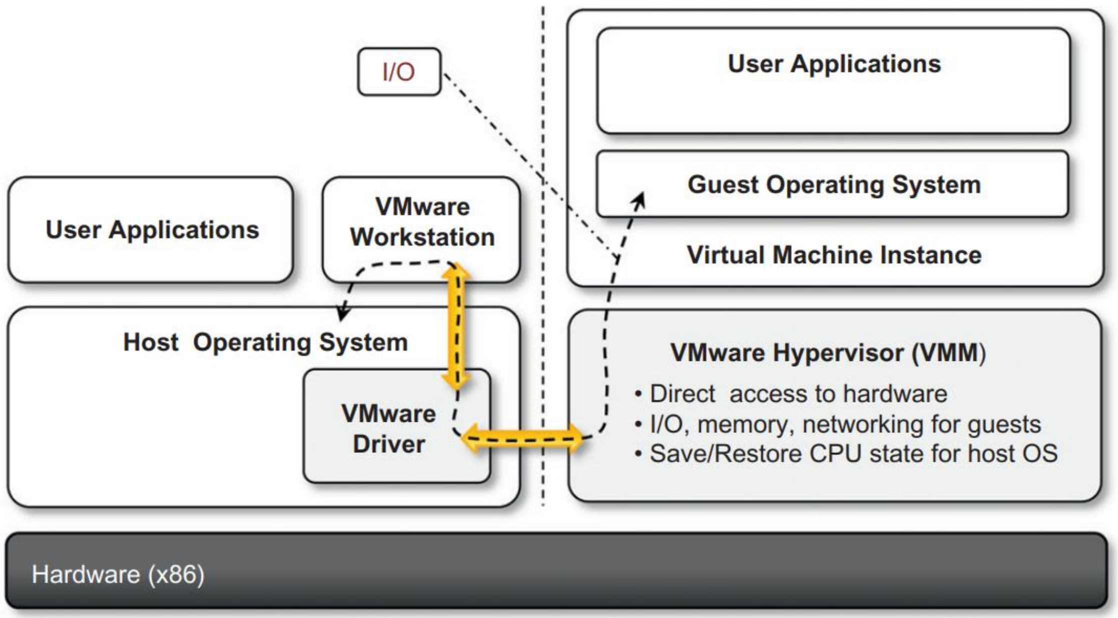


FIGURE 3.13

VMware workstation architecture.

Two main Services-

– It deploys a virtual machine manager that can run in privileged mode.

– It provides hooks for the VMware application to process specific I/O requests eventually by relaying such requests to the host operating system via system calls.

Server virtualization

- VMware provided solutions for server virtualization with different approaches over time.

Server virtualization

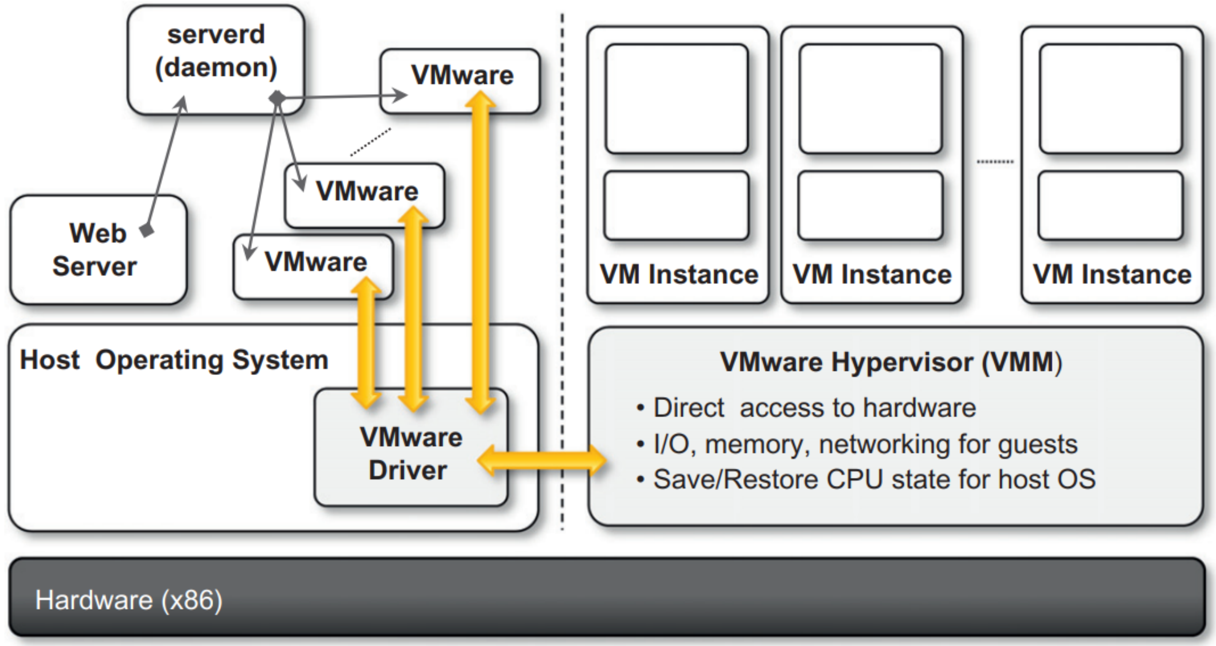


FIGURE 3.14

VMware GSX server architecture.

- Initial support for server virtualization was provided by VMware GSX server,
- VMware GSX server which replicates the approach used for **end-user computers and introduces remote management and scripting capabilities**

- It uses a client–server model, allowing remote access to virtual machines, at the cost of some graphical performance.

Server virtualization

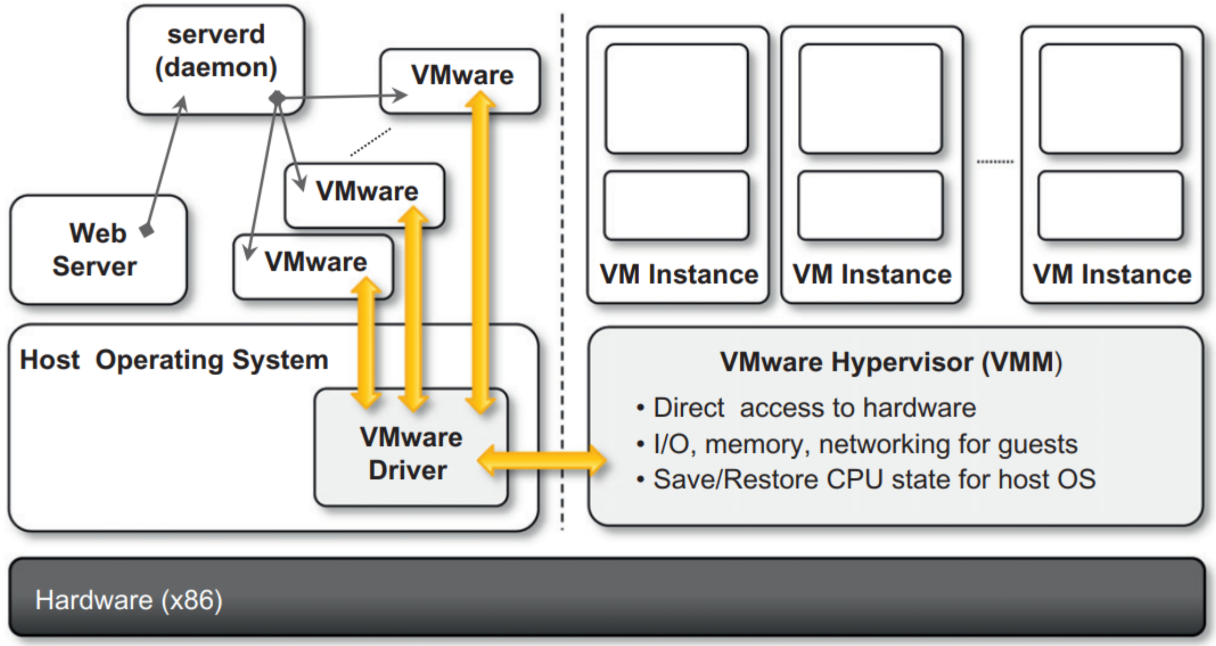


FIGURE 3.14

VMware GSX server architecture.

- The architecture is mostly designed to serve the virtualization of Web servers.
- **A daemon process, called serverd, controls and manages VMware application processes.**
- The remote host runs a VMware server authentication daemon

Server virtualization

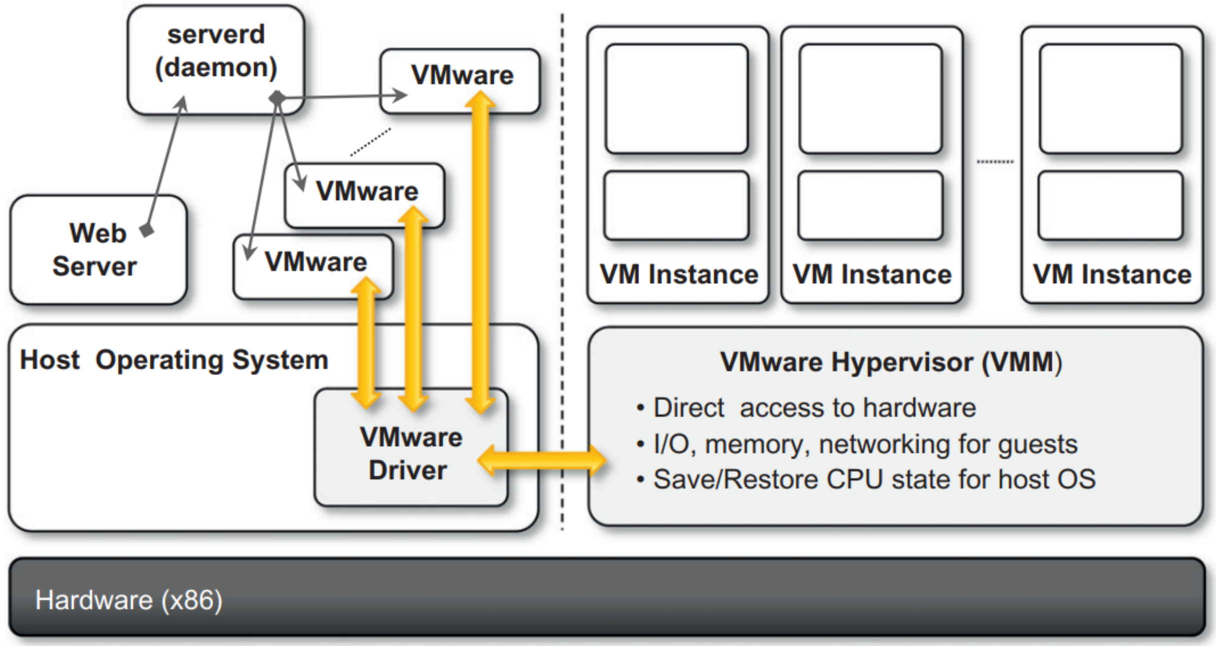


FIGURE 3.14

VMware GSX server architecture.

- **These applications are then connected to the virtual machine instances by means of the VMware driver** installed on the host operating system.
- **Virtual machine instances are managed by the VMM** as described previously.

Server virtualization

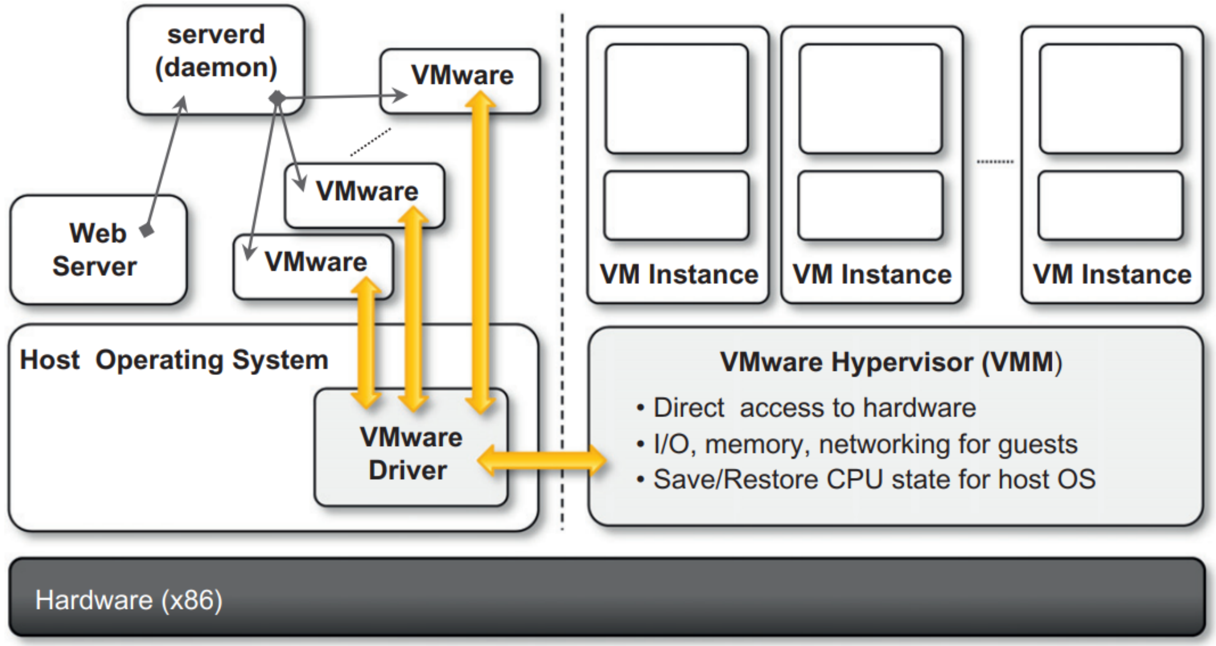


FIGURE 3.14

VMware GSX server architecture.

- **User requests for virtual machine management and provisioning are routed from the Web server through the VMM by means of serverd.**

Server virtualization

VMware ESX Server and its enhanced version, VMWare ESXi-

- These two solutions provide the same services but differ in the internal architecture, more specifically in the organization of the hypervisor kernel.
- **VMware ESX Server and its enhanced version, VMWare ESXi Server, are examples of the hypervisor-based approach.**
- **Both can be installed on bare metal servers and provide services for virtual machine management.**

Infrastructure virtualization and cloud computing solutions

- VMware provides a set of products covering the entire stack of cloud computing,
  - **from infrastructure management to Software-as-a-Service solutions hosted in the cloud.**

Infrastructure virtualization and cloud computing solutions

- Figure 3.16 gives an overview of the different solutions offered and how they relate to each other.

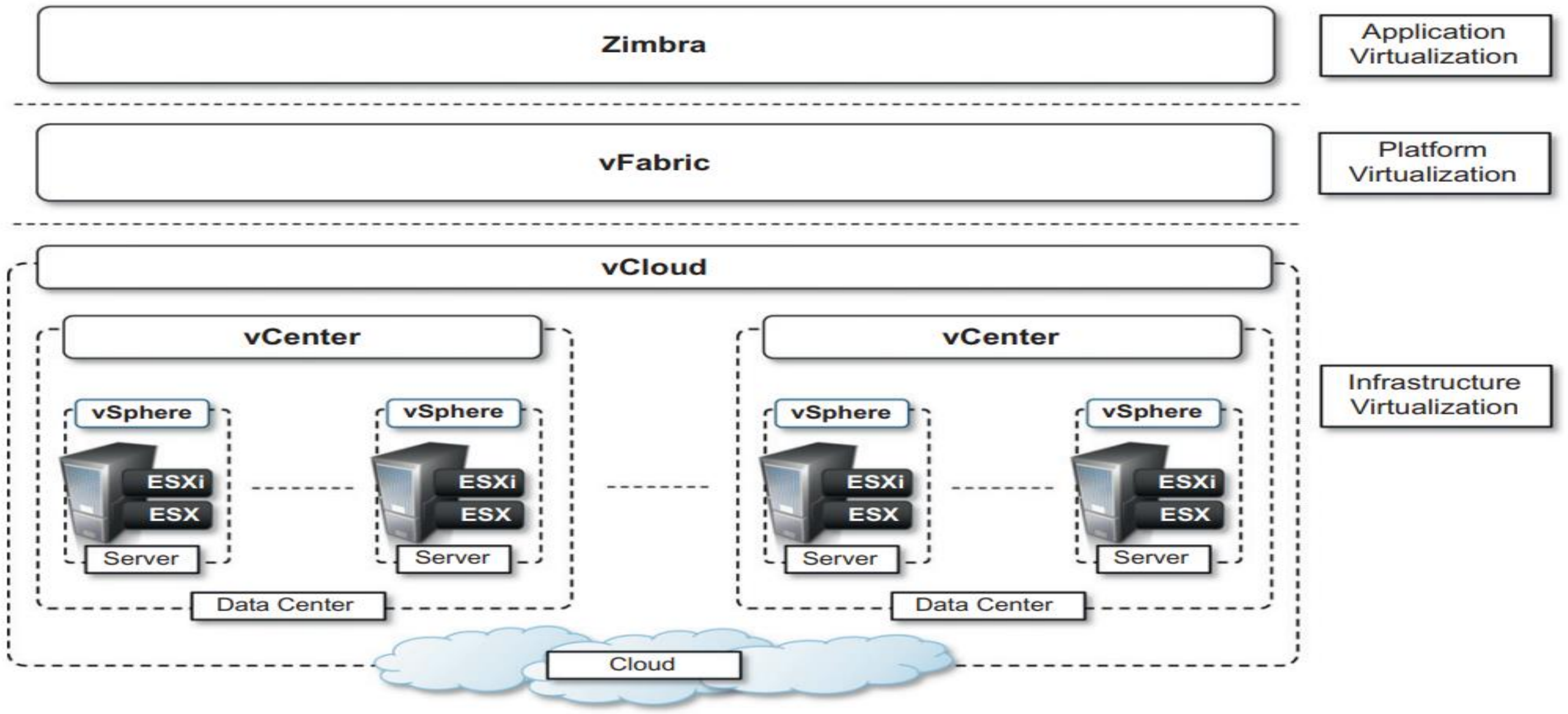


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- **ESX and ESXi constitute the building blocks** of the solution for virtual infrastructure management:

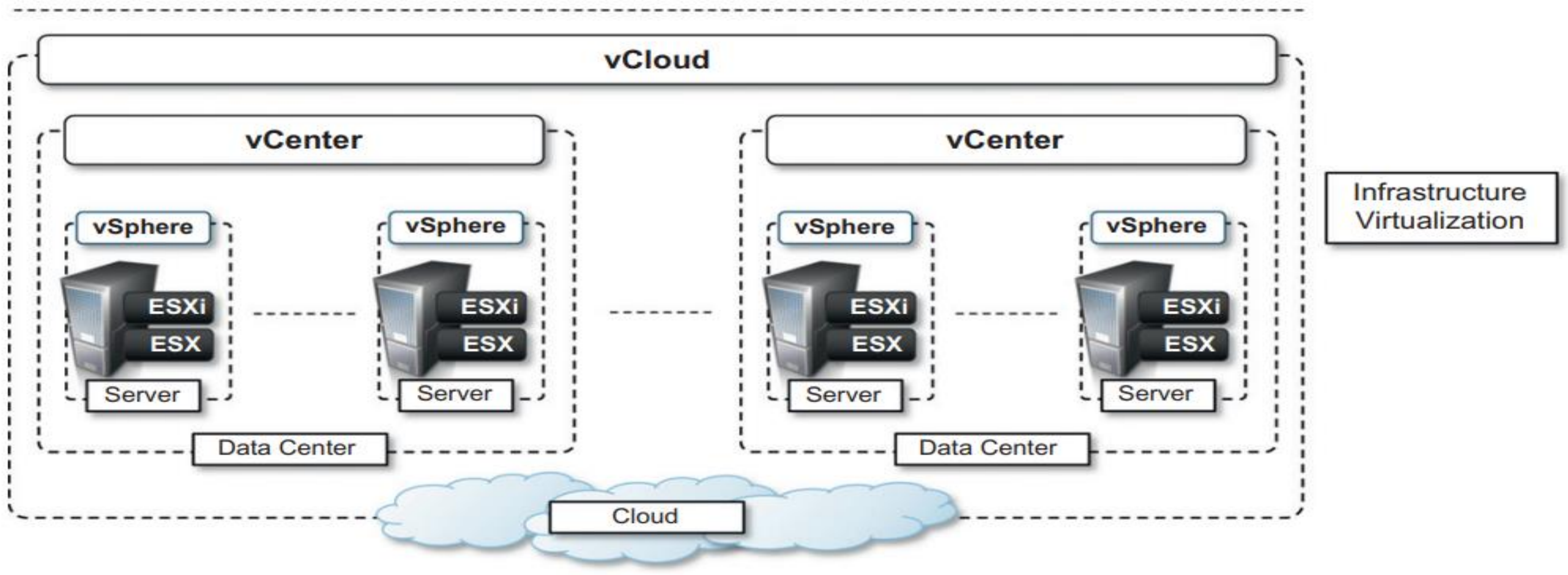


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- **A pool of virtualized servers is tied together and remotely managed as a whole by VMware vSphere.**

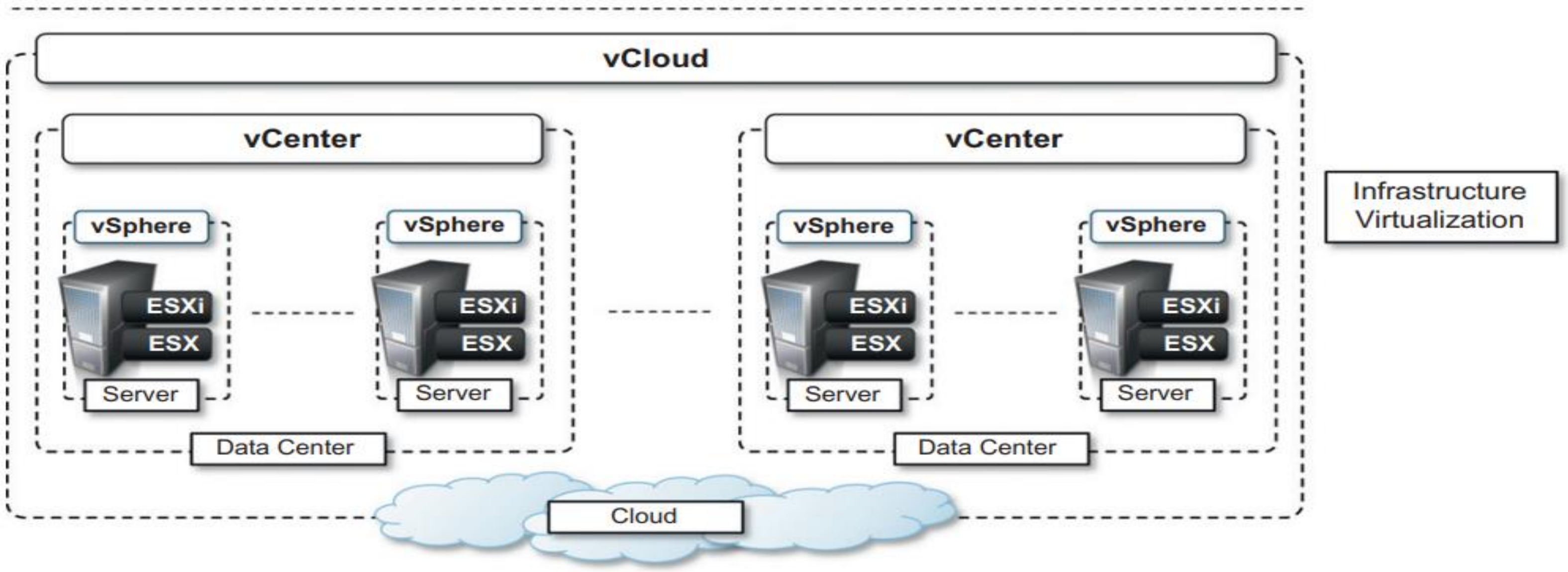


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- vSphere-
- As a virtualization platform it provides a set of basic services besides virtual compute services:
- **Virtual file system,**
- **virtual storage, and**
- **virtual network**
- constitute the core of the infrastructure;

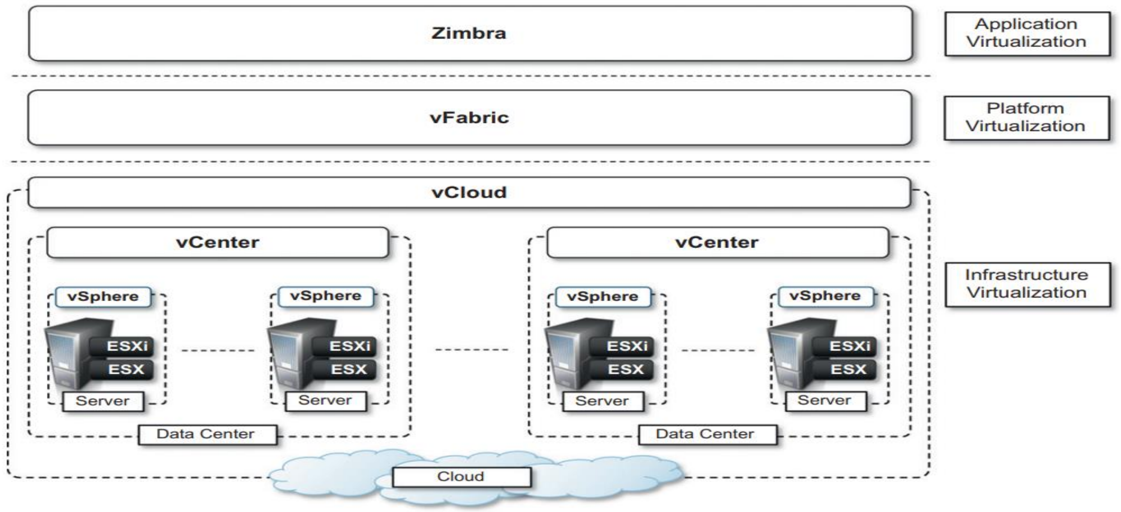


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- application services, such as
- **virtual machine migration,**
- **storage migration,**
- **data recovery, and**
- **security zones**, complete the **services offered by vSphere**.

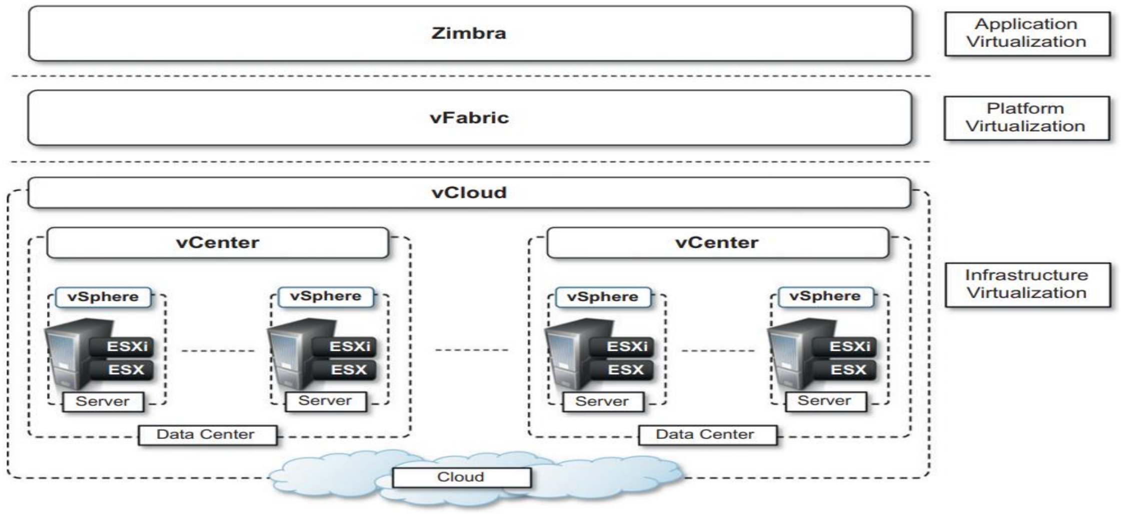


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- The management of the infrastructure is operated by VMware vCenter,
- which provides **centralized administration and management of vSphere installations in a data center environment**

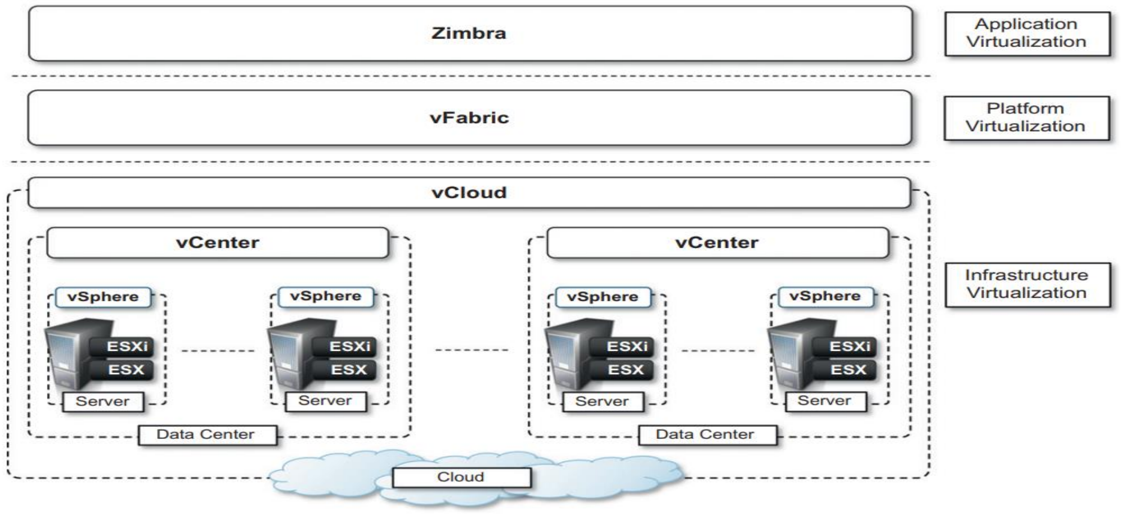


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- A collection of virtualized data centers are turned into a **Infrastructure-as-a-Service cloud by VMware vCloud,**
- which allows service providers **to make available to end users virtual computing environments on demand on a pay-per-use basis.**
- **IAAS**

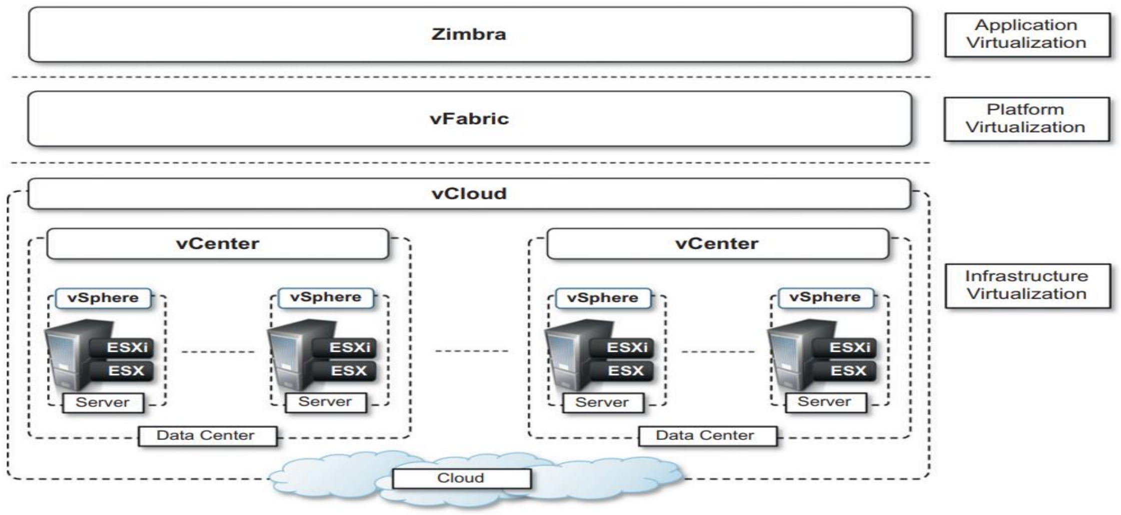


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- **A Web portal provides access to the provisioning services of vCloud,** and end users **can self provision virtual machines by choosing from available templates and setting up virtual networks** among virtual instances.

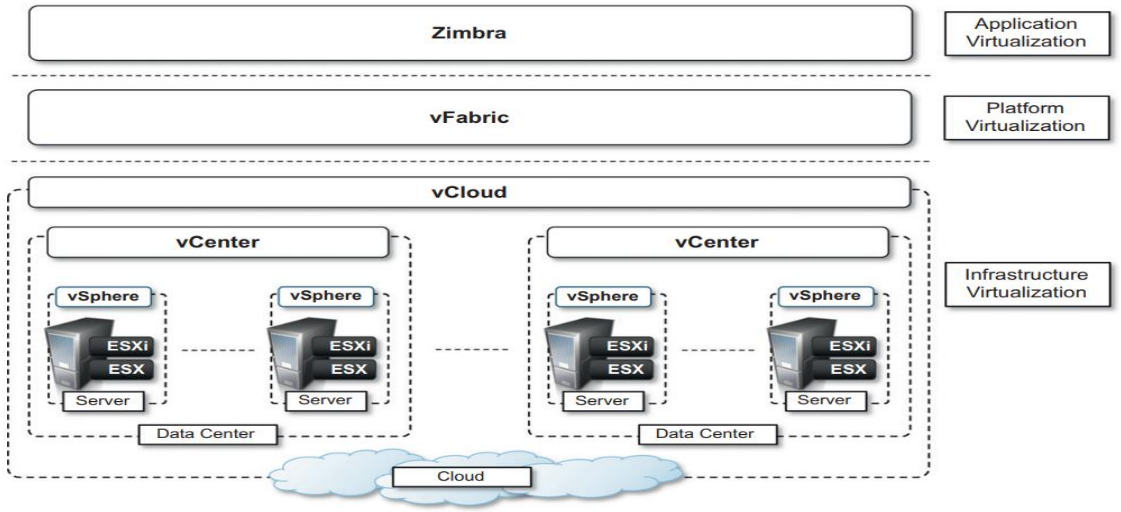


FIGURE 3.16
VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- VMware also provides a solution for **application development in the cloud with VMware vFabric,**
- vFabric is a set of components that facilitate the **development of scalable Web applications** on top of a virtualized infrastructure
- **PAAS**

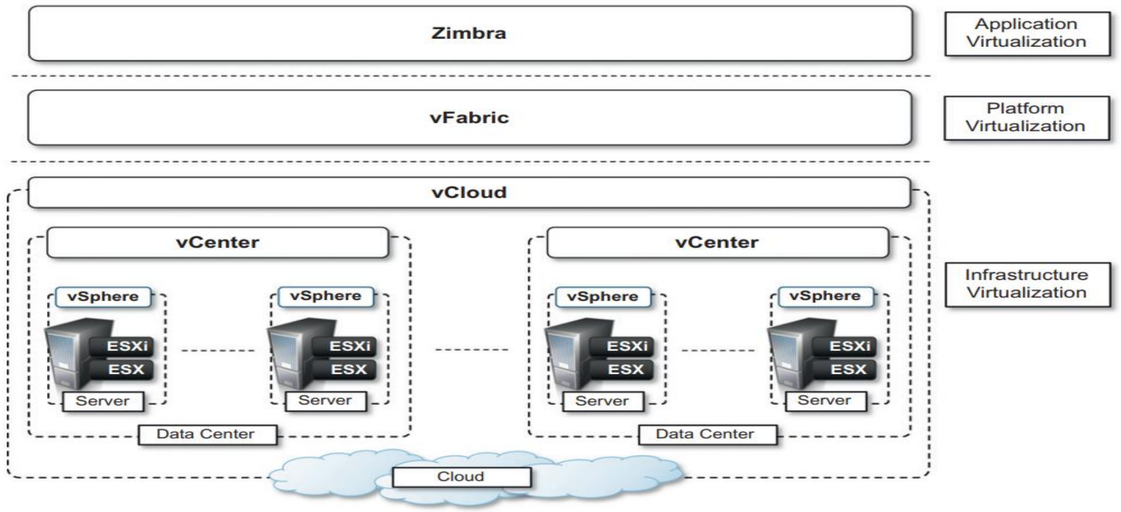


FIGURE 3.16
VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- vFabric is a collection of components for application monitoring, scalable data management, and scalable execution and **provisioning of Java Web applications**

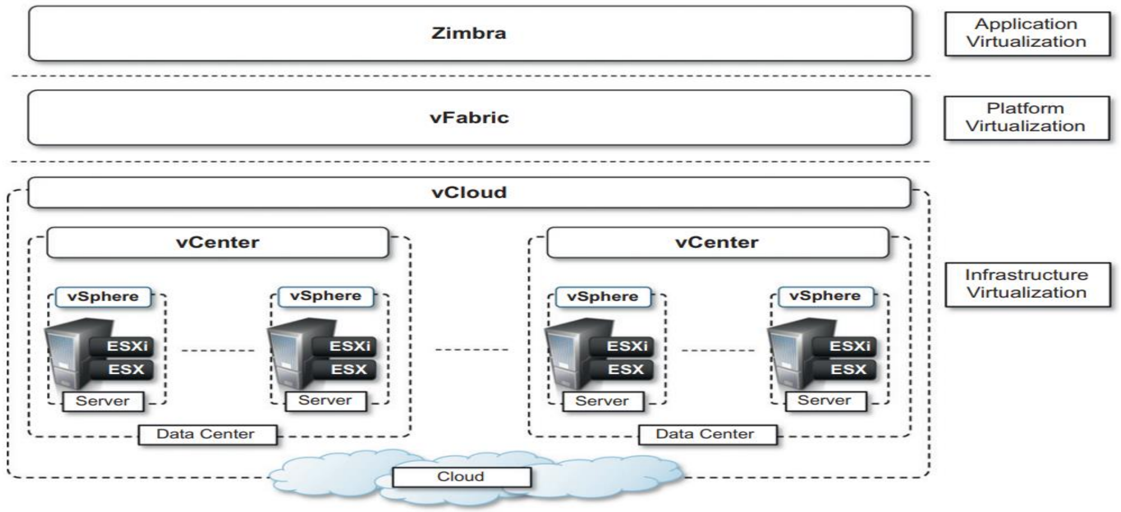


FIGURE 3.16
VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- Finally, at the top of the cloud computing stack, VMware provides Zimbra, **a solution for office automation, messaging, and collaboration that is completely hosted in the cloud** and accessible from anywhere.
- SaaS

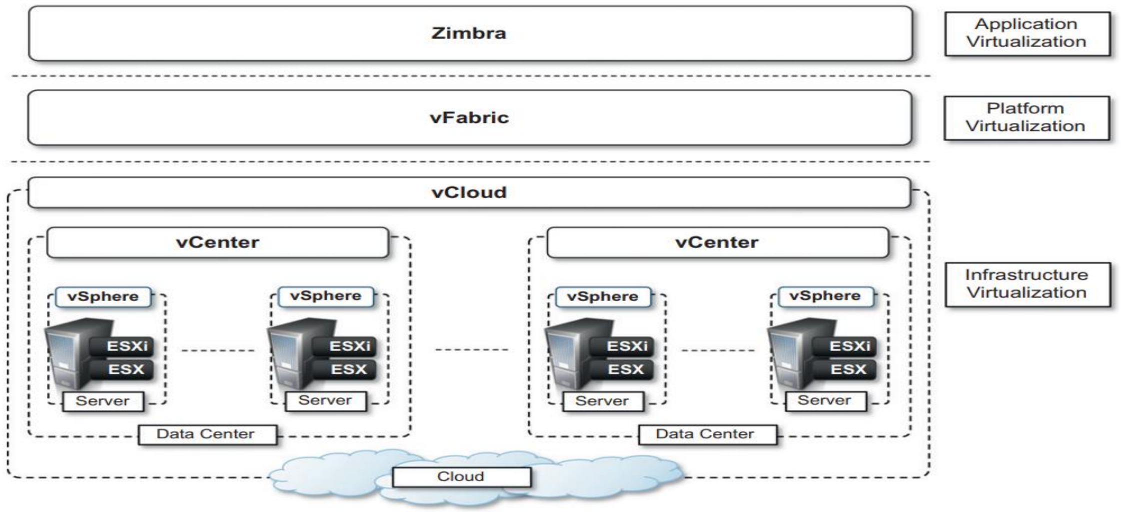


FIGURE 3.16

VMware Cloud Solution stack.

Infrastructure virtualization and cloud computing solutions

- Zimbra **is an SaaS solution that integrates various features into a single software platform providing email and collaboration management.**

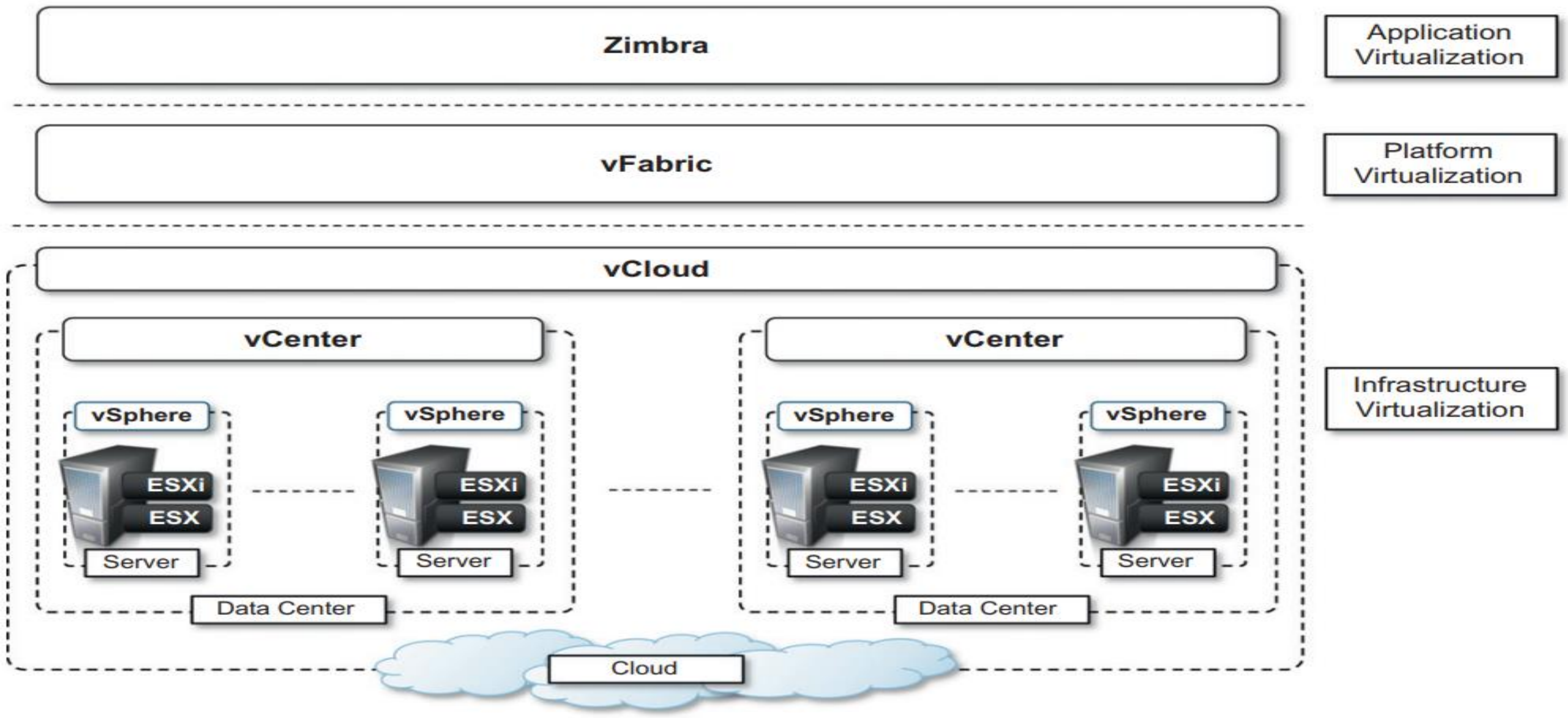


FIGURE 3.16

VMware Cloud Solution stack.

The cloud reference model

- Cloud computing **supports any IT service that can be consumed as a utility and delivered through a network**, most likely the Internet.
- Such characterization **includes quite different aspects: infrastructure, development platforms, application and services.**

Architecture

- It is possible **to organize all the concrete realizations of cloud computing into a layered view** covering the entire stack from hardware appliances to software systems.

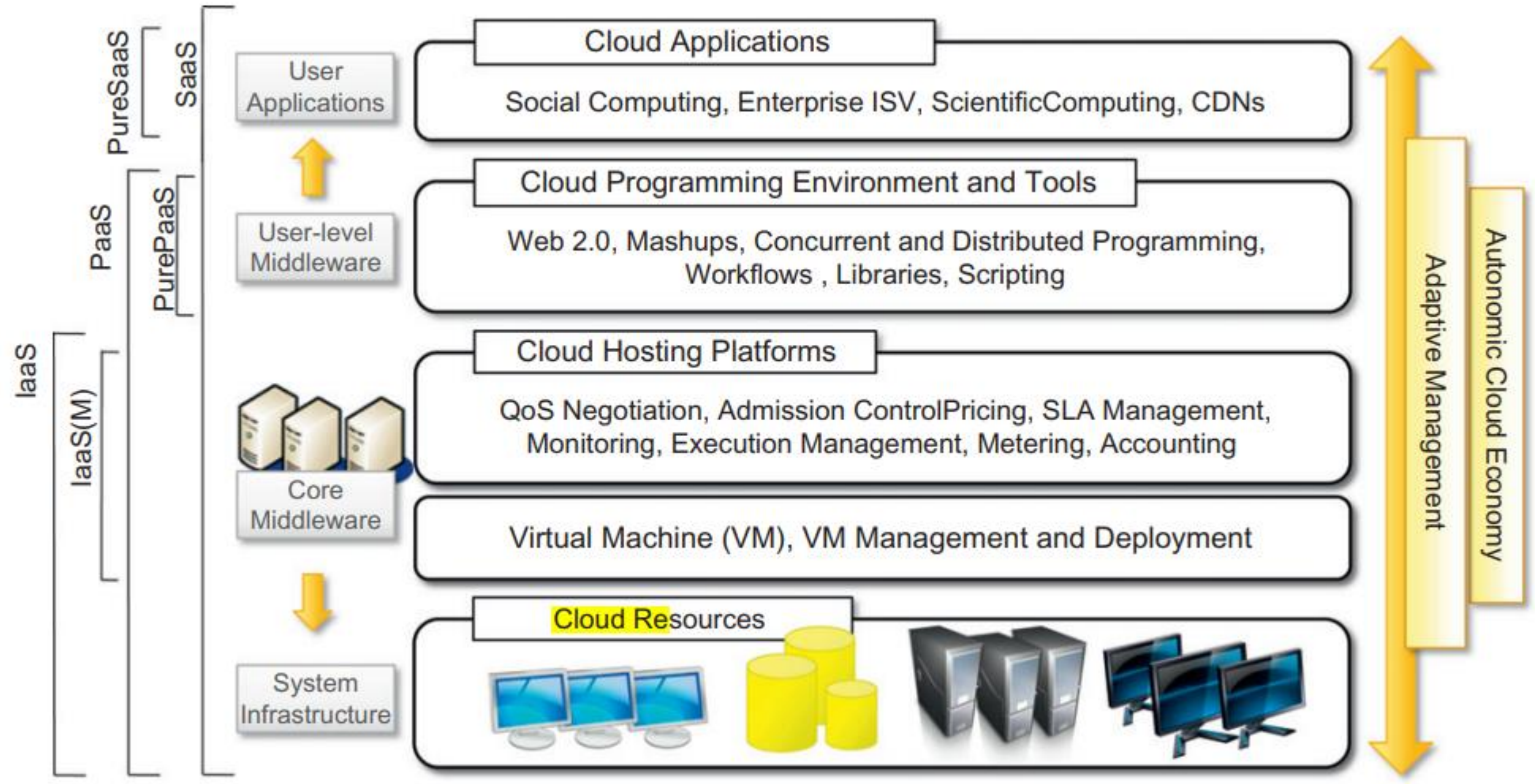
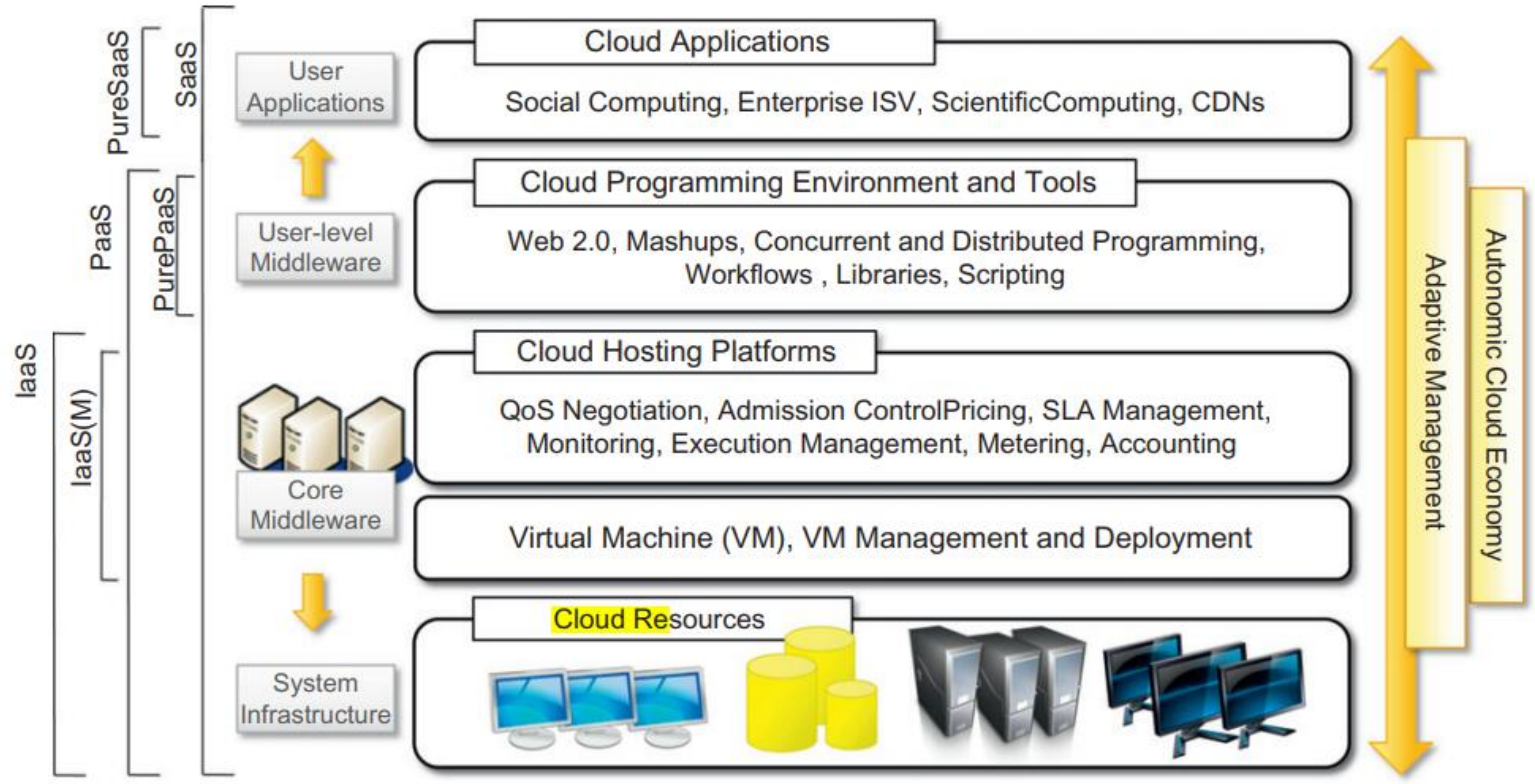

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- **Cloud resources -**are harnessed to offer "computing horsepower" required for providing services.
- Often, **this layer is implemented using a datacenter in which hundreds and thousands of nodes are stacked together.**



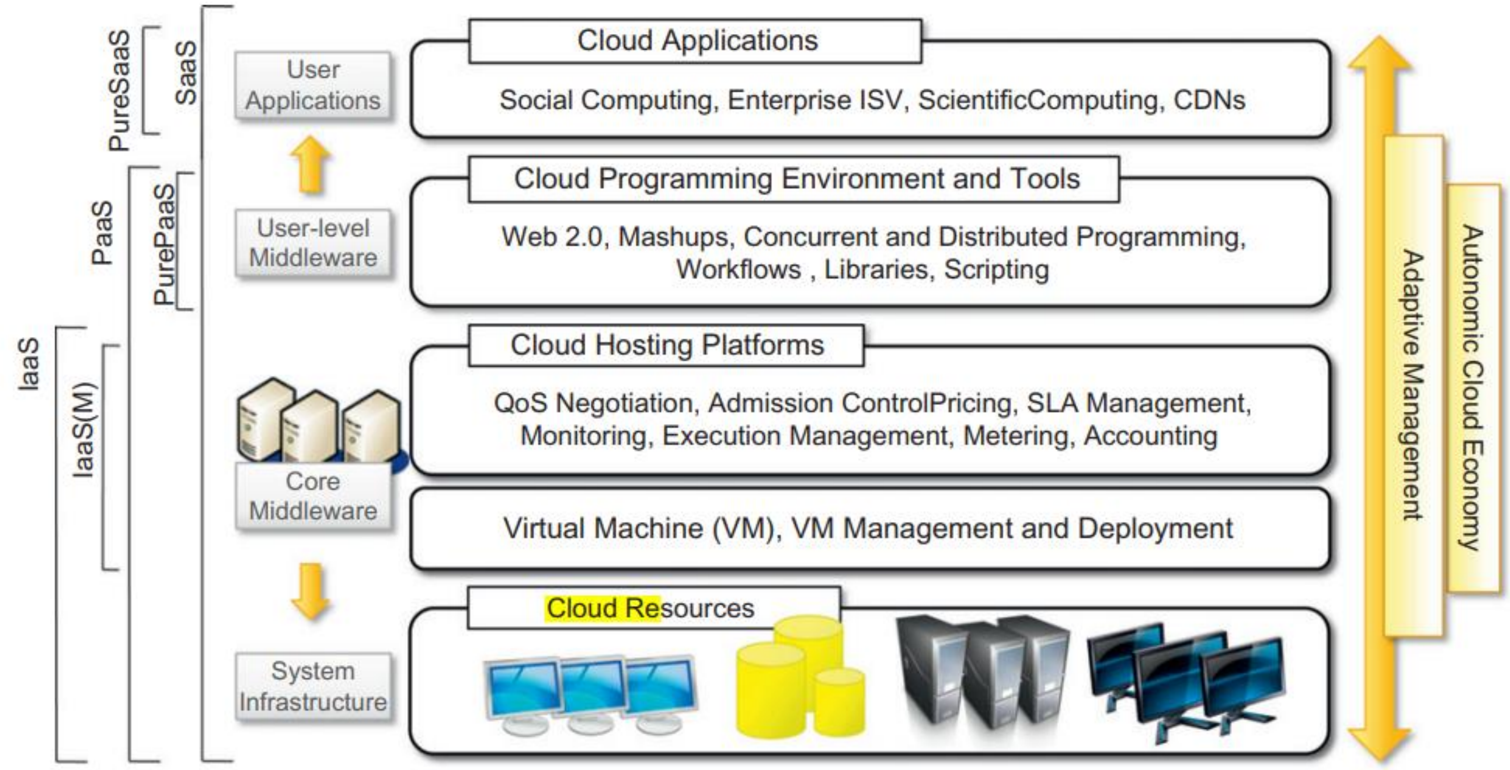
PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- Cloud infrastructure **can be heterogeneous in nature** because a variety of resources, such as clusters and even networked PCs, can be used to build it.
- Moreover, database systems and other storage services can also be part of the infrastructure



PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- The physical infrastructure is managed by **the core middleware,**
- The objectives of which are to provide an appropriate runtime environment for applications and **to best utilize resources**

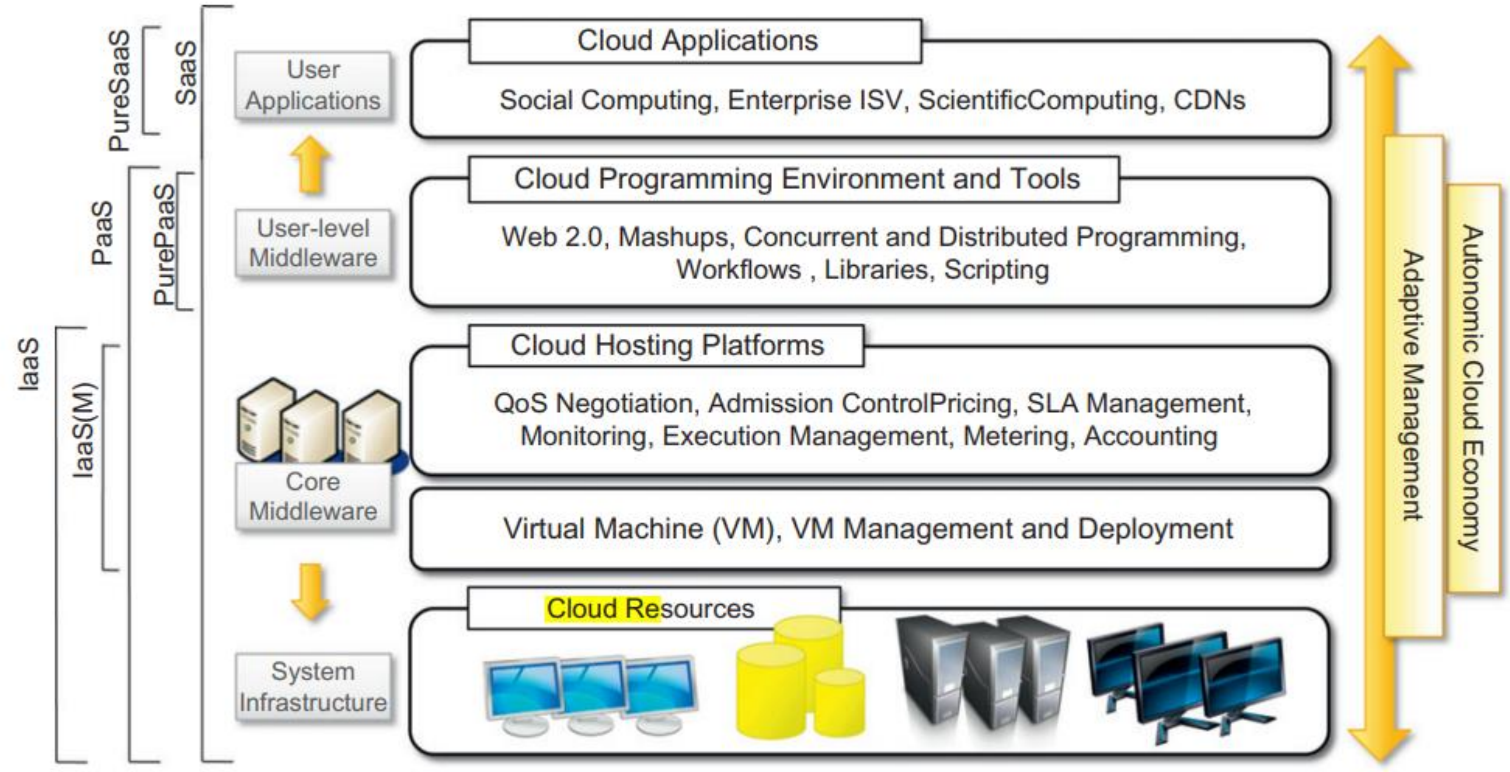

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- Core Middleware-
- At the bottom of the stack, virtualization technologies are used to guarantee runtime environment customization, **application isolation, sandboxing, and quality of service**.
- **Hypervisors** manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines.

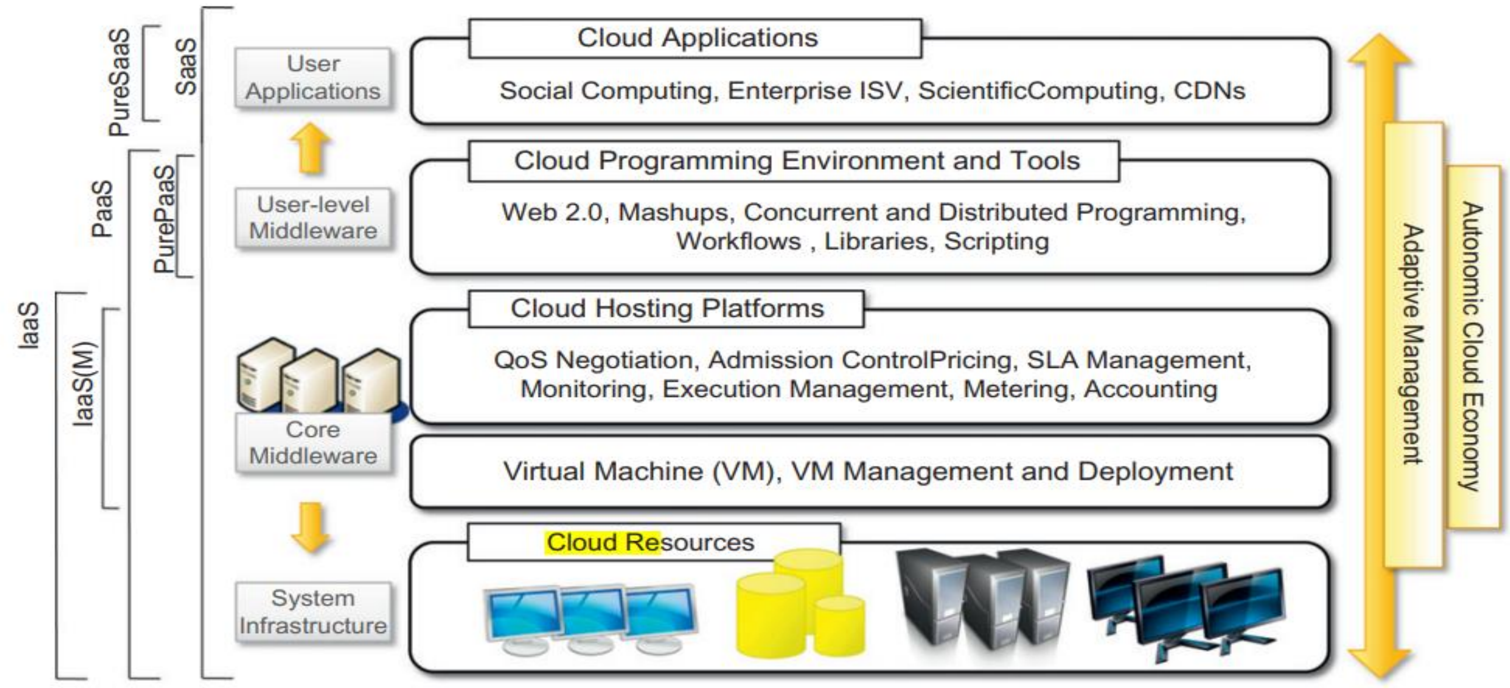
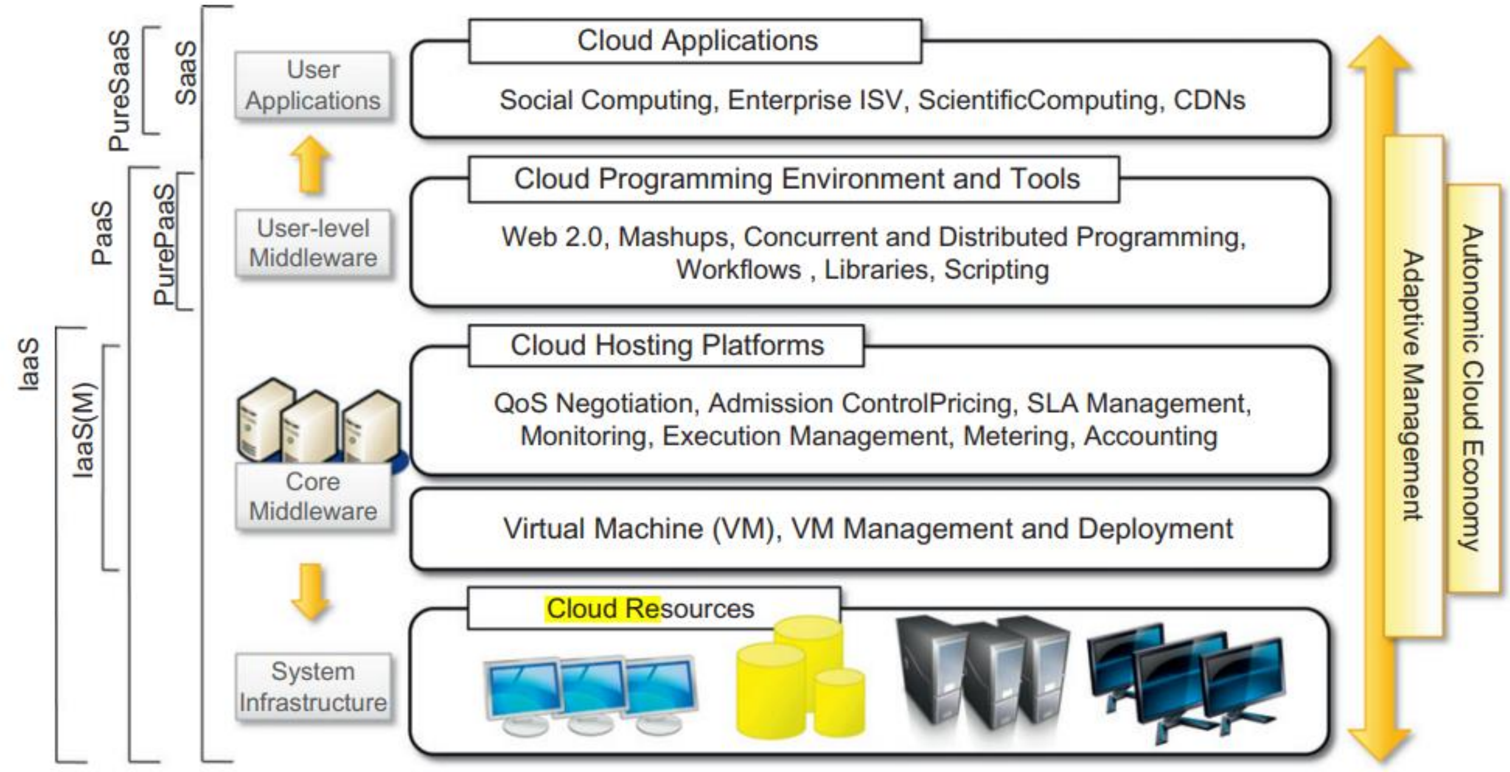

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- The combination of cloud hosting platforms and resources is generally classified as a Infrastructure-as-a-Service (IaaS) solution.



PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- We can organize the different examples of IaaS into two categories:
- **Some of them provide both the management layer and the physical infrastructure**;
- **others provide only the management layer (IaaS (M)).**

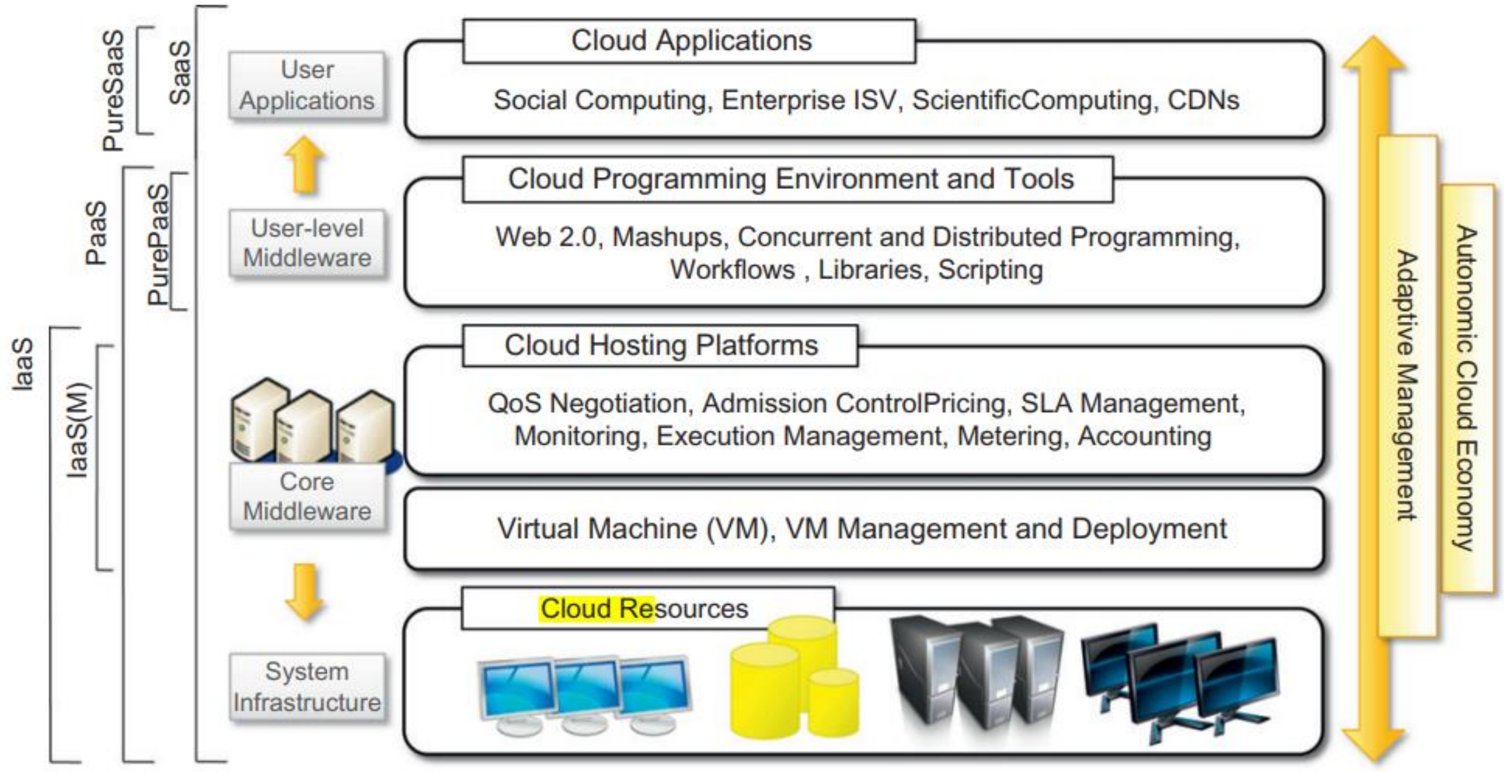
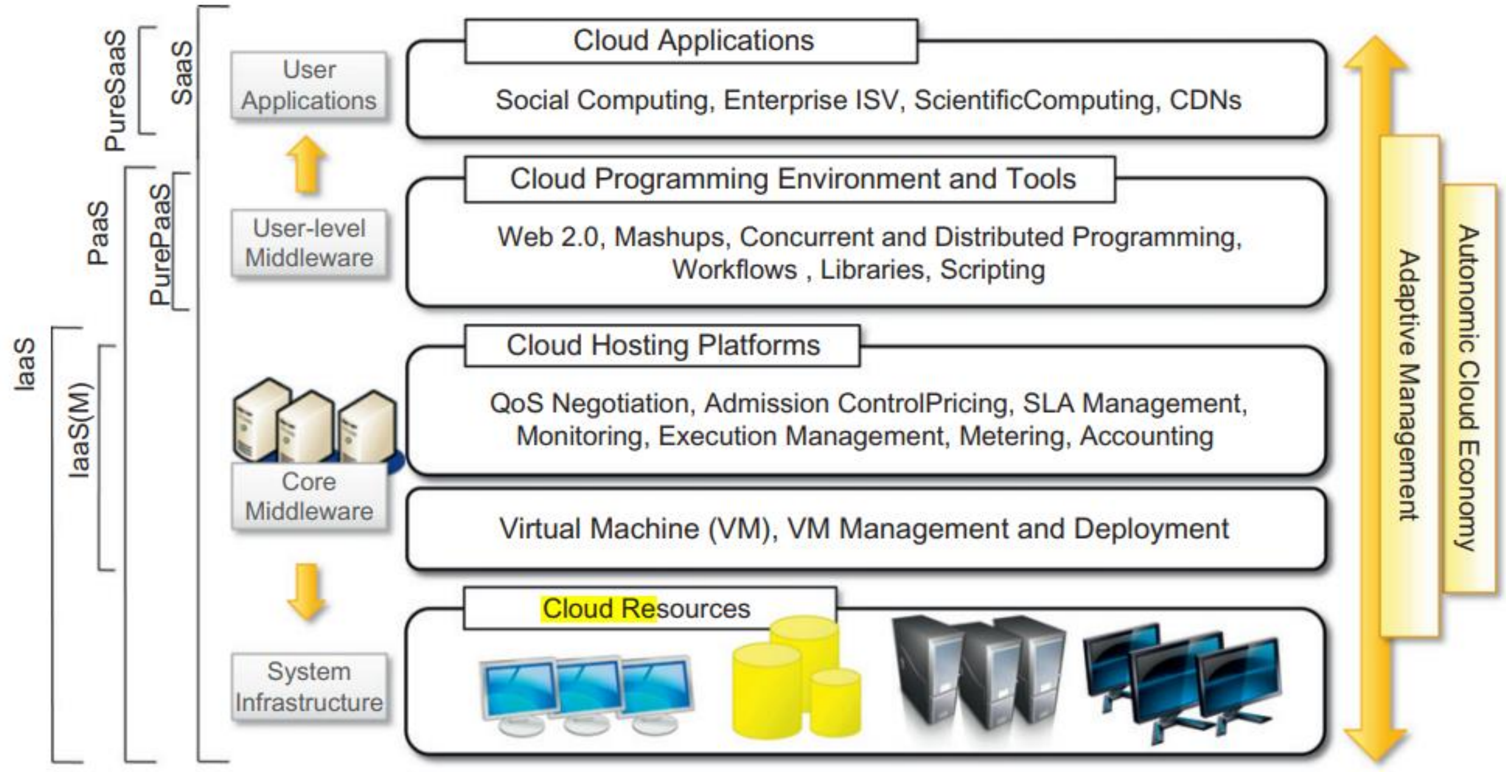

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- In this second case, the management layer is often integrated with other IaaS solutions that provide physical infrastructure and adds value to them.



PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

The cloud reference model

- A new layer is formed by cloud programming environments and tools, for offering users **a development platform for applications.**
- The range of tools include **Web-based interfaces, command-line tools, and frameworks for concurrent and distributed programming.**

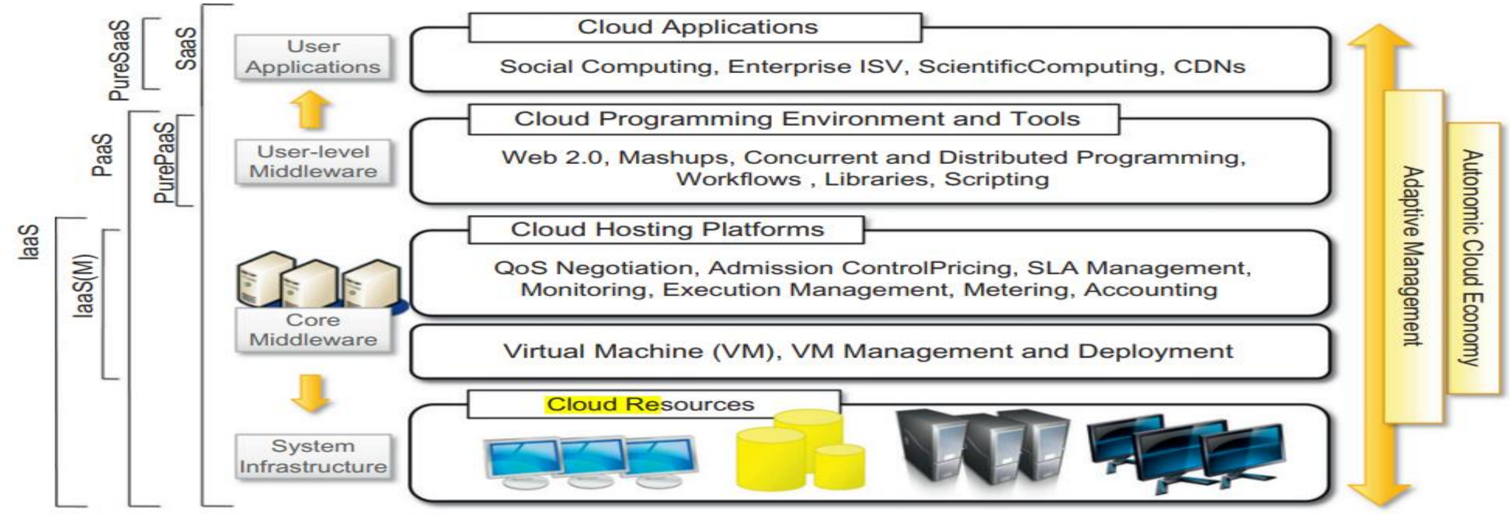

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1
The cloud computing architecture.

The cloud reference model

- Platform-as-a-Service (PaaS) because the service offered to the user **is a development platform rather than an infrastructure.**
- PaaS solutions generally include the infrastructure as well, which is bundled as part of the service provided to users.
- In the case of **Pure PaaS, only the user-level middleware is offered**, and it has to be complemented with a virtual or physical infrastructure.

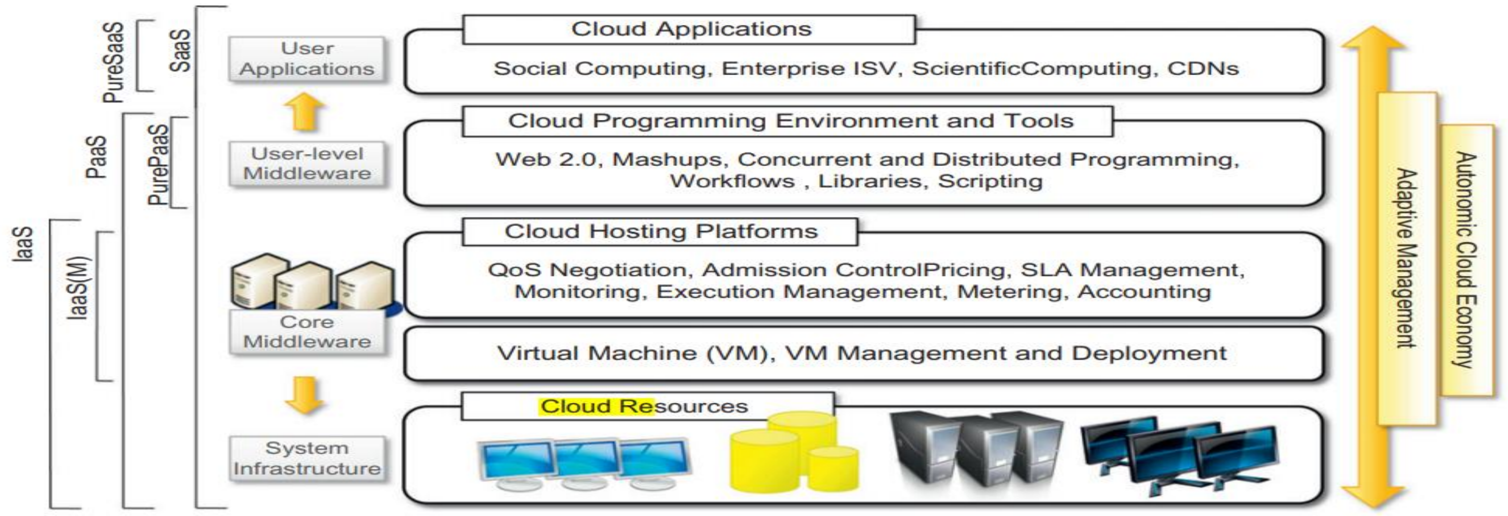
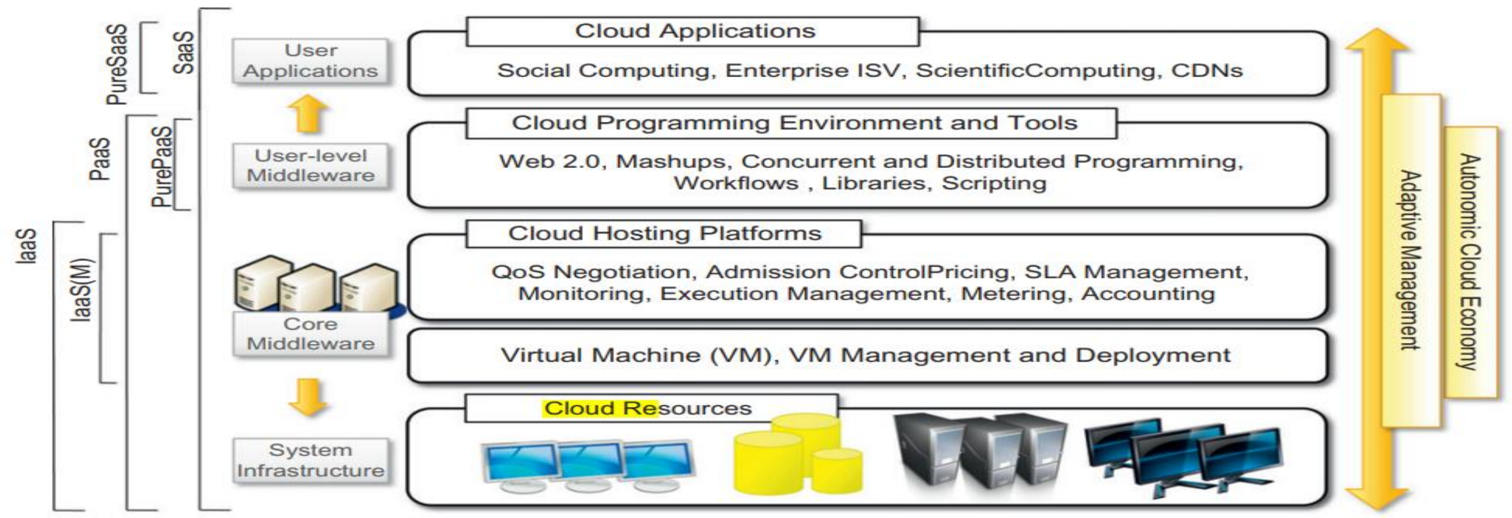

PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1
The cloud computing architecture.

The cloud reference model

- The top layer of the reference model depicted in Figure contains services delivered at the application level.
- These are mostly referred to as Software-as-a-Service (SaaS). In most cases these are **Web-based applications that rely on the cloud to provide service to end users.**



PureSaaS
SaaS
PaaS
PurePaaS
IaaS
IaaS(M)
User Applications
User-level Middleware
Core Middleware
System Infrastructure
Cloud Applications
Social Computing, Enterprise ISV, ScientificComputing, CDNs
Cloud Programming Environment and Tools
Web 2.0, Mashups, Concurrent and Distributed Programming, Workflows , Libraries, Scripting
Cloud Hosting Platforms
QoS Negotiation, Admission ControlPricing, SLA Management, Monitoring, Execution Management, Metering, Accounting
Virtual Machine (VM), VM Management and Deployment
Cloud Resources
Adaptive Management
Autonomic Cloud Economy

FIGURE 4.1

The cloud computing architecture.

Economics of the cloud

- The main drivers of cloud computing are **economy of scale** and simplicity of software delivery and its operation.
- In fact, the biggest benefit of this phenomenon is financial: **The pay-as-you-go model offered by cloud providers.**

Economics of the cloud

- In particular, cloud computing allows:
  - Reducing the **capital costs** associated to the IT infrastructure
  - Eliminating the **depreciation or lifetime costs** associated with IT capital assets
  - Replacing **software licensing with subscriptions**
  - Cutting the **maintenance and administrative costs of IT** resources

Capital Costs

- **A capital cost is the cost occurred in purchasing an asset that is useful in the production of goods or the rendering of services.**
- Capital costs are one-time expenses that are generally paid up front and that will contribute over the long term to generate profit.
- **The IT infrastructure and the software are capital assets because enterprises require them to conduct their business.**

Capital Costs

- **At present it does not matter whether the principal business of an enterprise is related to IT,**
- because the business will definitely have an IT department that is used to automate many of the activities that are performed within the enterprise:
- **payroll,**
- **customer relationship management,**
- **enterprise resource planning,**
- **tracking and inventory of products, and others.**

- Hence, **IT resources constitute a capital cost for any kind of enterprise .**

Capital Costs

- It is good practice to try **to keep capital costs as low as possible**
- As capital costs are associated with material things they are **subject to depreciation over time,**
- which in the end **reduces the profit of the enterprise because such costs are directly subtracted from the enterprise revenues.**

Capital Costs

- In the case of IT capital costs, the depreciation costs are represented
- **by the loss of value of the hardware over time and**
- **the aging of software products that need to be replaced because new features are required.**

- Before cloud computing diffused within the enterprise, the budget spent on IT infrastructure and software constituted a significant expense for medium-sized and large enterprises
- **Many enterprises owned a small or medium-sized datacenter that introduces several operational costs in terms of maintenance, electricity, and cooling.**
- Additional operational costs are occurred in maintaining an IT department and an IT support center.
- **Moreover, other costs are triggered by the purchase of potentially expensive software.**

- With cloud computing **these costs are significantly reduced** or simply disappear according to its penetration.

Capital Costs

- In terms of the pricing models introduced by cloud computing, we can distinguish three different strategies that are adopted by the providers:

1) **Tiered pricing**
2) **Per-unit pricing**
3) **Subscription-based pricing**

Tiered pricing-

- In this model, cloud services are offered in several tiers,
- Each of which offers **a fixed computing specification and SLA at a specific price per unit of time.**
- This model is used by Amazon for pricing the EC2 service, which makes available **different server configurations in terms of computing capacity (CPU type and speed, memory) that have different costs per hour.**

Per-unit pricing-

- This model is more suitable to cases where the principal source of revenue for the cloud provider is determined in terms of **units of specific services, such as data transfer and memory allocation.**
- In this scenario customers can configure their systems more efficiently according to the application needs.
- This model is used, for example, by **GoGrid, which makes customers pay according to RAM/hour units for the servers deployed in the GoGrid cloud.**

Subscription-based pricing-

- This is the model used mostly by SaaS providers in which users pay **a periodic subscription fee for use of the software or the specific component services that are integrated in their applications**

Open challenges

- Still in its infancy, cloud computing presents many challenges for industry and academia.

Open challenges

- Still in its infancy, cloud computing presents many challenges for industry and academia:-

1) Cloud definition
2) Cloud interoperability and standards
3) Scalability and fault tolerance
4) Security, trust, and privacy
5) Organizational aspects

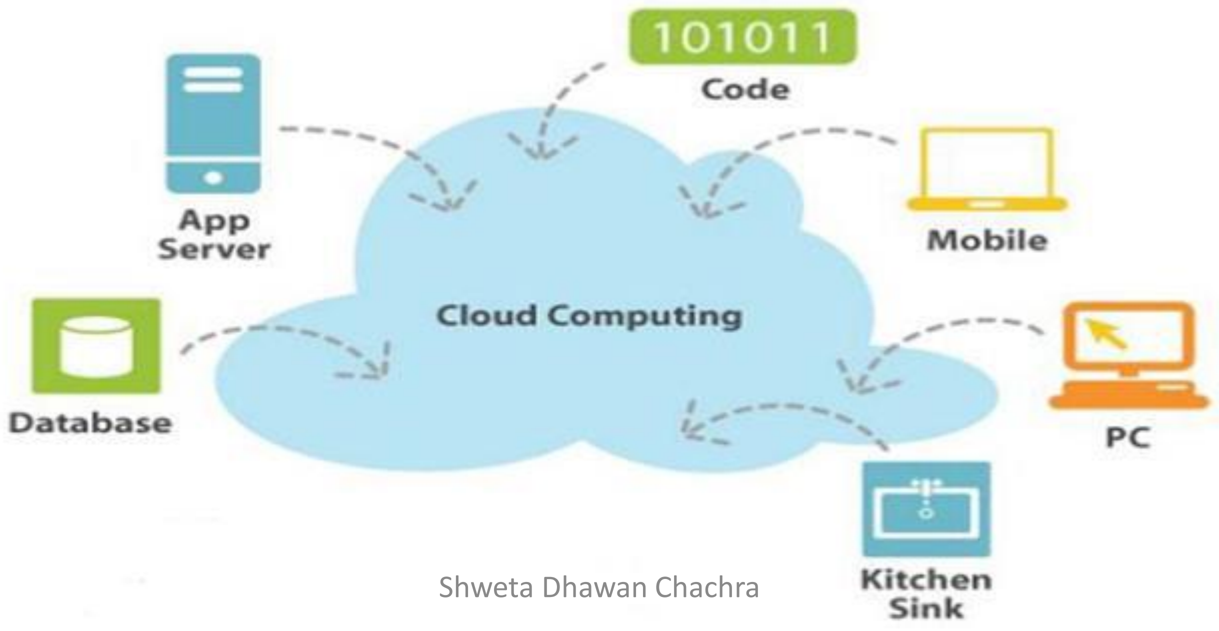
Cloud definition

- There have been several attempts made to define cloud computing

Cloud definition

- One of the most comprehensive formalizations is noted in the NIST working definition of cloud computing .
- It characterizes cloud computing as on-demand self-service, broad network access, resource-pooling, rapid elasticity, and measured service; classifies services as SaaS, PaaS, and IaaS; and categorizes deployment models as public, private, community, and hybrid clouds.
- The view is shared by many IT practitioners and academicians.

- US **National Institute of Standards and Technology (NIST)** defines Computing as:
- " Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. "



Cloud definition

- Despite the general agreement on the NIST definition, there are alternative taxonomies for cloud services.
- David Linthicum, founder of BlueMountains Labs, provides a more detailed classification, **which comprehends 11 different classes and better suits the vision of cloud computing within the enterprise.**

Cloud definition

- **David S. Linthicum mentions 11 major categories or patterns of cloud computing technology.**

David Linthicum, Managing Director – Chief Cloud Strategy Officer | Deloitte US

Managing Director | Chief Cloud Strategy Officer

+1 703 216 6676 dlinthicum@deloitte.com

- **David S. Linthicum mentions 11 major categories or patterns of cloud computing technology.**

1. Storage-as-a-service
2. Database-as-a-service
3. Information-as-a-service
4. Process-as-a-service
5. Application-as-a-service
6. Platform-as-a-service
7. Integration-as-a-service
8. Security-as-a-service
9. Management/governance-as-a-service
10. Testing-as-a-service
11. Infrastructure-as-a-service

- **David S. Linthicum mentions 11 major categories or patterns of cloud computing technology.**

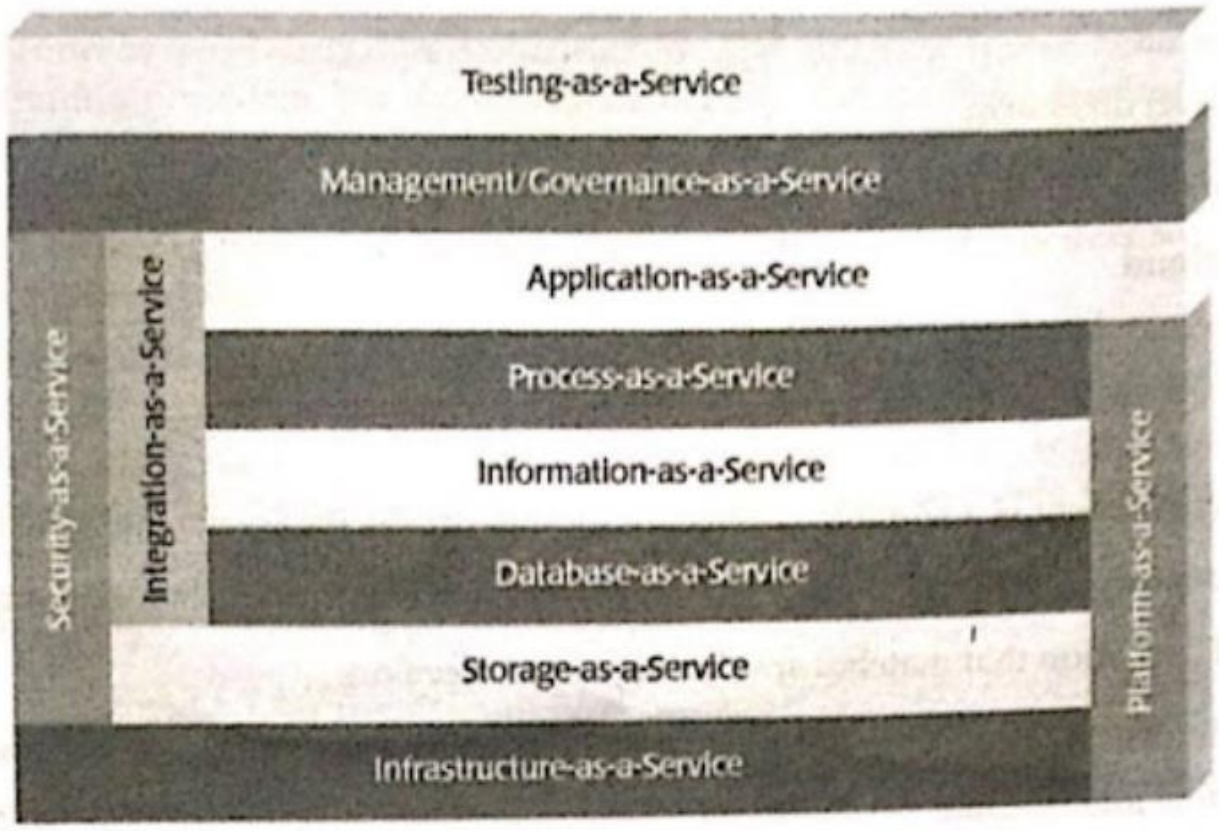


Figure 1.6 Patterns of cloud computing technology

Cloud definition

- A different approach has been taken at the University of California, Santa Barbara (UCSB), **which departs from the XaaS concept and tries to define an ontology for cloud computing.**

Cloud definition

- **In their work the concept of a cloud is dissected into five main layers:**
  - **applications,**
  - **software environments,**
  - **software infrastructure,**
  - **software kernel, and**
  - **hardware.**
- Each layer addresses the needs of a different class of users within the cloud computing community and most likely builds on the underlying layers

Cloud interoperability and standards

- Introducing standards and allowing interoperability between solutions offered by different vendors are objectives of fundamental importance.
- **Vendor lock-in** constitutes one of the major strategic barriers against the seamless adoption of cloud computing at all stages.

Cloud interoperability and standards

- **Vendor lock-in can prevent a customer from switching to another competitor's solution, or**
- **when this is possible, it happens at considerable conversion cost and requires significant amounts of time.**
- This can occur either because **the customer wants to find a more suitable solution for customer needs or because the vendor is no longer able to provide the required service.**

Cloud interoperability and standards

- The current state of standards and interoperability in cloud computing **resembles the early Internet era,**
- **when there was no common agreement on the protocols and technologies used and each organization had its own network.**

Cloud interoperability and standards

- Yet the first steps toward a standardization process have been made, and the following organizations are leading the path
- Cloud Computing Interoperability Forum (CCIF),
- Open Cloud Consortium, and
- DMTF Cloud Standards Incubator

Scalability and fault tolerance

- The ability to scale on demand constitutes one of the most attractive features of cloud computing. **Clouds allow scaling beyond the limits of the existing in-house IT resources, whether they are infrastructure (compute and storage) or applications services.**
- To implement such a capability, the cloud middleware has to be designed with the principle of scalability along different dimensions in mind—for example, performance, size, and load.
- **The cloud middleware manages a huge number of resource and users, which rely on the cloud to obtain the horsepower that they cannot obtain within the premises** without bearing considerable administrative and maintenance costs

Security, trust, and privacy

- Security, trust, and privacy issues are major obstacles for massive adoption of cloud computing.
- The traditional cryptographic technologies are used to prevent data tampering and access to sensitive information.
- **The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable.**

Security, trust, and privacy

- On one side we need to decide **whether to trust the provider itself;**
- On the other side, **specific regulations can simply prevail over the agreement the provider is willing to establish** with us concerning the privacy of the information managed on our behalf.
- Moreover, cloud services delivered to the end user can be the result of **a complex stack of services that are obtained by third parties via the primary cloud service provider.**

Security, trust, and privacy

- In this case there is **a chain of responsibilities in terms of service delivery** that can introduce more vulnerability for the secure management of data, the enforcement of privacy rules, and the trust given to the service provider.
- In particular, when **a violation of privacy or illegal access to sensitive information is detected**, **it could become difficult to identify who is liable for such violations.**

Organizational aspects

- Cloud computing introduces a significant change in the way IT services are consumed and managed.
- More precisely, storage, compute power, network infrastructure, and applications are delivered as metered services over the Internet.
- **This introduces a billing model that is new within typical enterprise IT departments, which requires a certain level of cultural and organizational process maturity.**
- In particular, a wide acceptance of cloud computing will require a significant **change to business processes and organizational boundaries**

Organizational aspects

- In particular, the following questions have to be considered:
  - What is the **new role of the IT department in an enterprise that completely or significantly relies on the cloud?**
  - **How will the compliance department perform its activity when there is a considerable lack of control over application**
  - **What are the implications (political, legal, etc.) for organizations that lose control over some aspects of their services?**
  - What will be the perception of the end users of such services workflows?