Cloud Computing Concepts

CS3132

Dr. Anand Kumar Mishra

NIIT University

Resolving the complication x86 Virtualization

- VMware resolved the challenge in 1998
- Developed - **binary translation techniques**
  - that allow the VMM to run in Ring 0 for isolation and performance,
  - while moving the OS to a user level ring with greater privilege than applications in Ring 3 but less privilege than the virtual machine monitor in Ring 0

Binary Translation Technique

- Binary translation involves dynamically translating binary instructions from the guest's architecture (source architecture) to the host's architecture (target architecture) as they are executed
- This technique is used in both Type 1 and Type 2
- Use Case:
  - Cross-Architecture Virtualization:
    - Running software built for one CPU architecture on a host with a different architecture
    - For example, running x86 applications on an ARM-based host
  - Legacy Software Support:
    - Allowing older applications or operating systems to run on modern hardware that may not natively support their architecture

Binary translation involves several steps and techniques

1. Decoding:
   - The binary code of the guest application or OS is decoded to understand the instructions.
2. Analysis:
   - The translator analyzes the decoded instructions to determine their purpose and potential dependencies.
3. Translation:
   - The translator converts the guest architecture instructions into equivalent host architecture instructions. This is often the most complex and critical step.
4. Optimization:
   - The translated code may undergo further optimization to improve performance. For instance, it can reduce the number of translations by caching already translated code.

Binary translation involves several steps and techniques

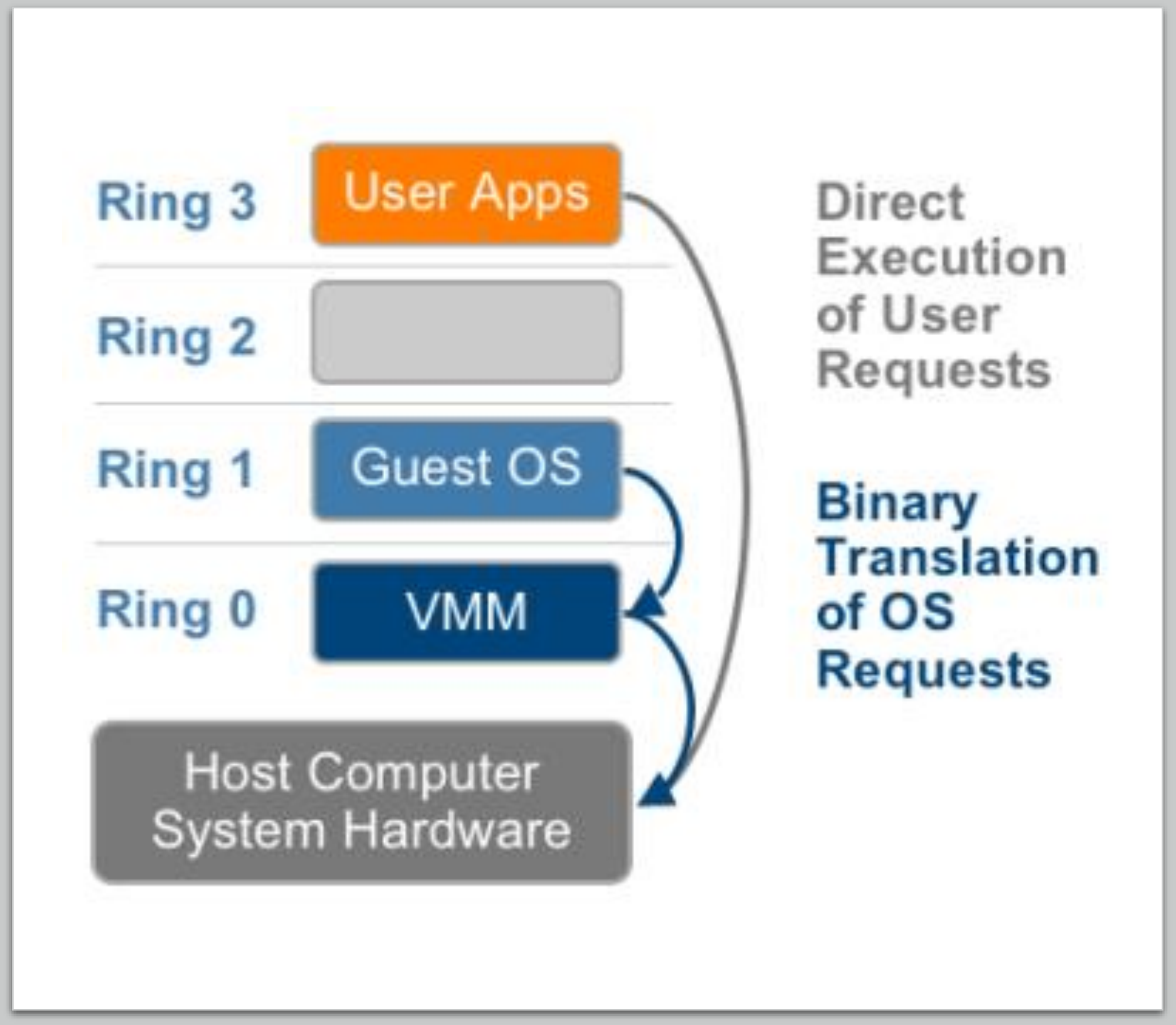
5. Execution:
   - The translated code is executed on the host machine, and the results are returned to the guest OS or application.
6. Handling Exceptions:
   - Special handling is required for exceptions and interrupts that may occur during translation to ensure correct operation.

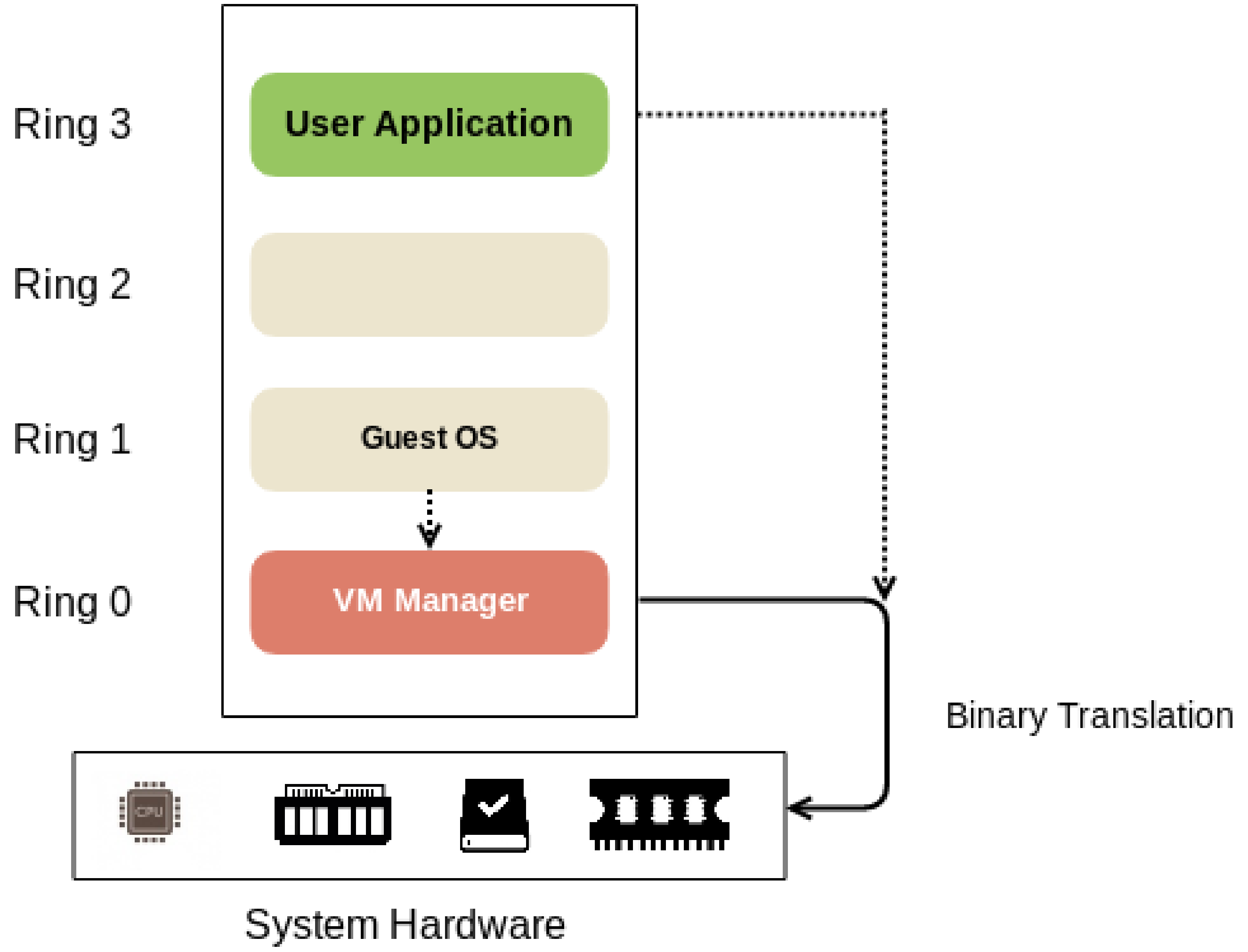
Techniques

- Alternative techniques for handling sensitive and privileged instructions to virtualize the CPU on the x86 architecture
  - Full virtualization using binary translation
  - OS assisted virtualization or paravirtualization
  - Hardware assisted virtualization (first generation)

Technique 1 – Full Virtualization using Binary Translation

- VMware can virtualize any x86 operating system using a combination of binary translation and direct execution techniques


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


Ring 3
User Application
Ring 2
Ring 1
Guest OS
Ring 0
VM Manager
Binary Translation
CPU
System Hardware

VMware: full virtualization

- VMware implements full virtualization:
  - either in the desktop environment, by means of Type II hypervisors,
  - or in the server environment, by means of Type I hypervisors
- In both cases, full virtualization is made possible by means of:
  - direct execution (for nonsensitive instructions) and
  - binary translation (for sensitive instructions)

With the new generation of hardware architectures and the introduction of hardware-assisted virtualization (Intel VT-x and AMD V) in 2006, full virtualization is made possible with hardware support,

but

before that date, the use of dynamic binary translation was the only solution that allowed running x86 guest operating systems unmodified in a virtualized environment.

Full virtualization and binary translation

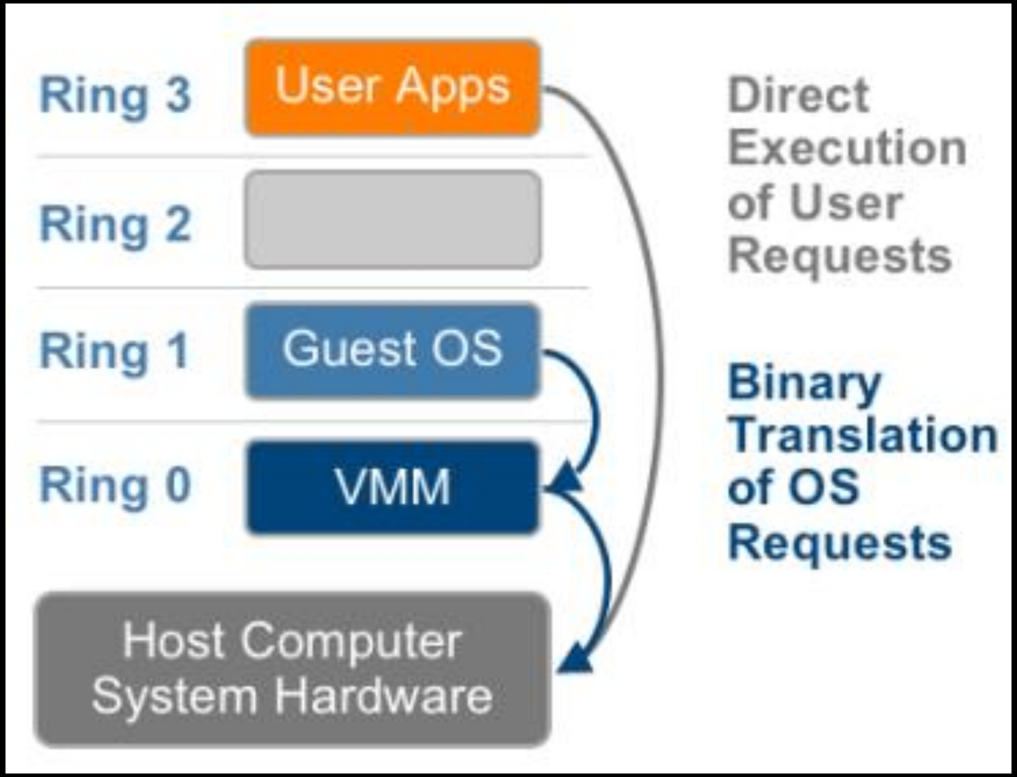
- With the new generation of hardware architectures and the introduction of hardware-assisted virtualization (Intel VT-x and AMD V) in 2006, full virtualization is made possible with hardware support,

but

- before that date, the use of dynamic binary translation was the only solution that allowed running x86 guest operating systems unmodified in a virtualized environment.
- x86 architecture design does not satisfy the first theorem of virtualization, since the set of sensitive instructions is not a subset of the privileged instructions
  - This causes a different behavior when such instructions are not executed in Ring 0

Technique 1 – Full Virtualization using Binary Translation

- **User level code** is directly executed on the processor for high performance virtualization
- **Translates kernel code** to replace nonvirtualizable instructions with new sequences of instructions that have the intended effect on the virtual hardware
- **Virtual Machine Monitor** provides each Virtual Machine with all the services of the physical system, including a virtual BIOS, virtual devices and virtualized memory management


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

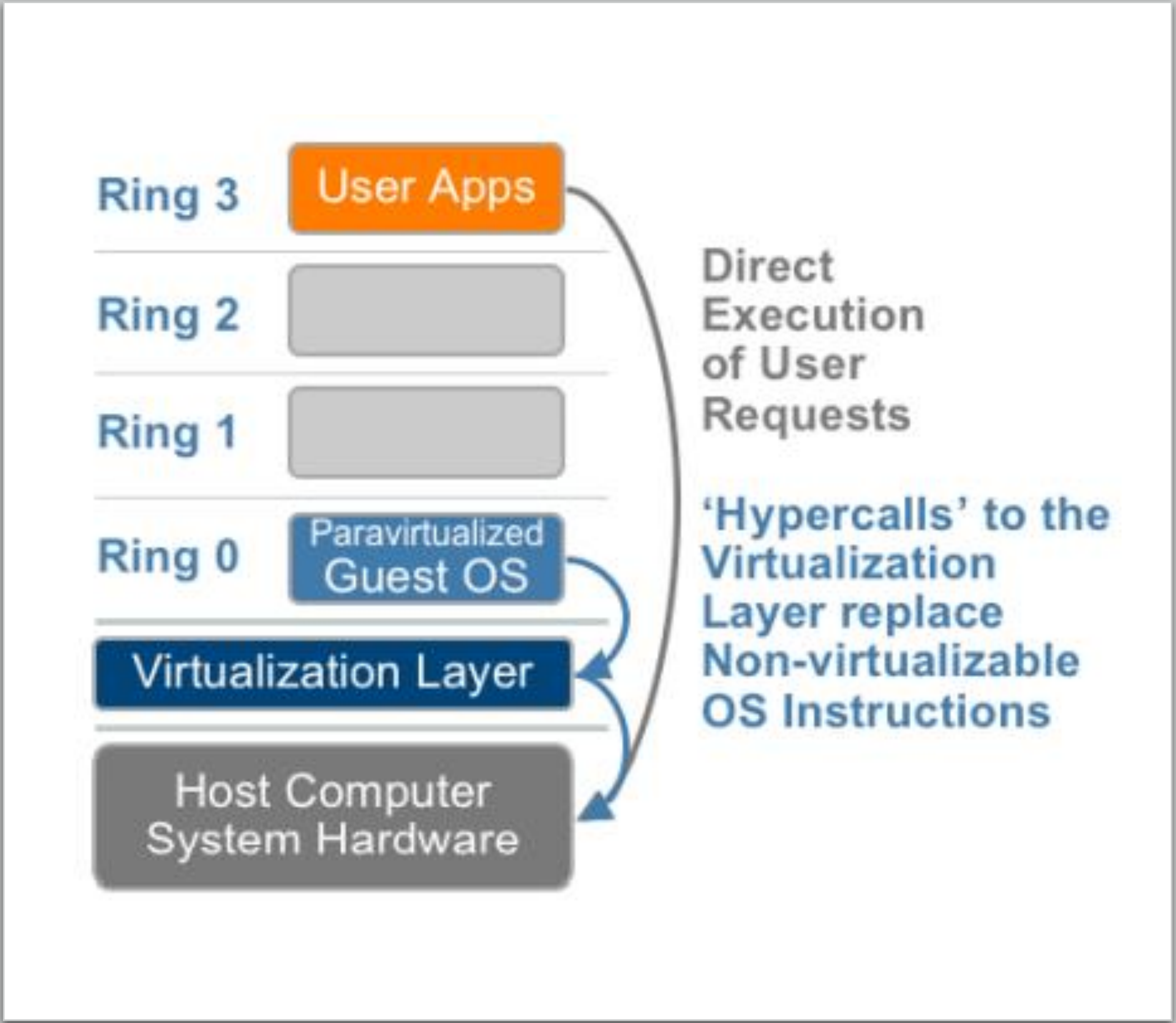
Technique 1 – Full Virtualization using Binary Translation

- This combination of binary translation and direct execution provides Full Virtualization as the guest OS is fully abstracted (completely decoupled) from the underlying hardware by the virtualization layer
- The **guest OS is not aware it is being virtualized and requires no modification**
- Full virtualization is the only option that requires no hardware assist or operating system assist to virtualize sensitive and privileged instructions

Technique 1 – Full Virtualization using Binary Translation

- The hypervisor translates all operating system instructions and caches the results for future use, while user level instructions run unmodified at native speed
- Full virtualization offers the best isolation and security for virtual machines, and simplifies migration and portability as the same guest OS instance can run virtualized or on native hardware
- VMware’s virtualization products are examples of full virtualization

Technique 2 – OS Assisted Virtualization or Paravirtualization


Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Paravirtualized Guest OS
Virtualization Layer
Host Computer System Hardware
Direct Execution of User Requests
'Hypercalls' to the Virtualization Layer replace Non-virtualizable OS Instructions

The Paravirtualization approach to x86 Virtualization

Technique 2 – OS Assisted Virtualization or Paravirtualization

- Paravirtualization involves modifying the OS kernel to replace nonvirtualizable instructions with hypercalls that communicate directly with the virtualization layer hypervisor
- The hypervisor also provides hypercall interfaces for other critical kernel operations such as memory management, interrupt handling and time keeping
- As paravirtualization cannot support unmodified operating systems (e.g. Windows 2000/XP), its compatibility and portability is poor

Technique 2 – OS Assisted Virtualization or Paravirtualization

- Instead of dynamically translating the sensitive instructions, leading to overhead, we can categorically change the kernel instructions to replace the sensitive instructions with hypercalls (call to the hypervisor) to get a modified OS with no sensitive instructions
- Every time a sensitive function needs to be executed, a hypercall is made instead
- Both Type 1 and 2 hypervisors work on unmodified OS
    - In contrast, para-virtualization modifies OS kernel to replace all sensitive instructions with hypercalls
    - Thus, the OS behaves like a user program making system calls and the hypervisor executes the privileged operation invoked by hypercall

Technique 2 – OS Assisted Virtualization or Paravirtualization

- Paravirtualization can also introduce significant support and maintainability issues in production environments as it requires deep OS kernel modifications
- The open source **Xen project** is an example of paravirtualization that virtualizes the processor and memory using a modified Linux kernel and virtualizes the I/O using custom guest OS device drivers

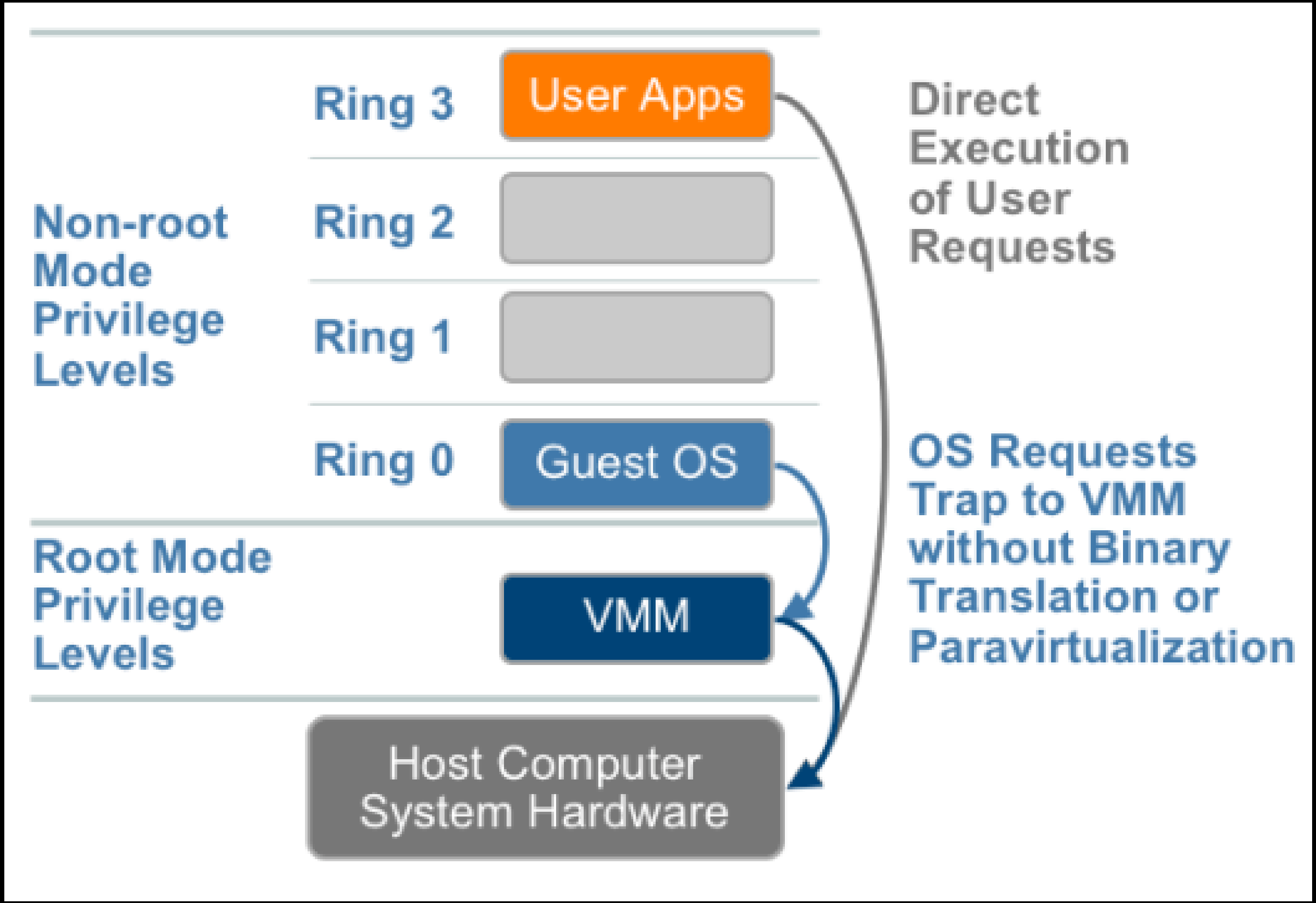
Technique 2 – OS Assisted Virtualization or Paravirtualization

- While it is very difficult to build the more sophisticated binary translation support necessary for full virtualization, modifying the guest OS to enable paravirtualization is relatively easy
- The VMware tools service provides a backdoor to the VMM Hypervisor used for services such as:
  - time synchronization
  - logging and
  - guest shutdown

Technique 2 – OS Assisted Virtualization or Paravirtualization

- Vmxnet is a paravirtualized I/O device driver that shares data structures with the hypervisor
  - It can take advantage of host device capabilities to offer improved throughput and reduced CPU utilization
- It is important to note for clarity that *the VMware tools service and the vmxnet device driver are not CPU paravirtualization solutions*
  - They are minimal, non-intrusive changes installed into the guest OS that do not require OS kernel modification
- Xen ran as a para-virtualized version of Linux when the hardware did not support Type 1 hypervisors
- Xen needed Linux to run in domain 0 (master partition) just like HyperV

Technique 3 – Hardware Assisted Virtualization


Ring 3
User Apps
Direct Execution of User Requests
Non-root Mode Privilege Levels
Ring 2
Ring 1
Ring 0
Guest OS
OS Requests Trap to VMM without Binary Translation or Paravirtualization
Root Mode Privilege Levels
VMM
Host Computer System Hardware

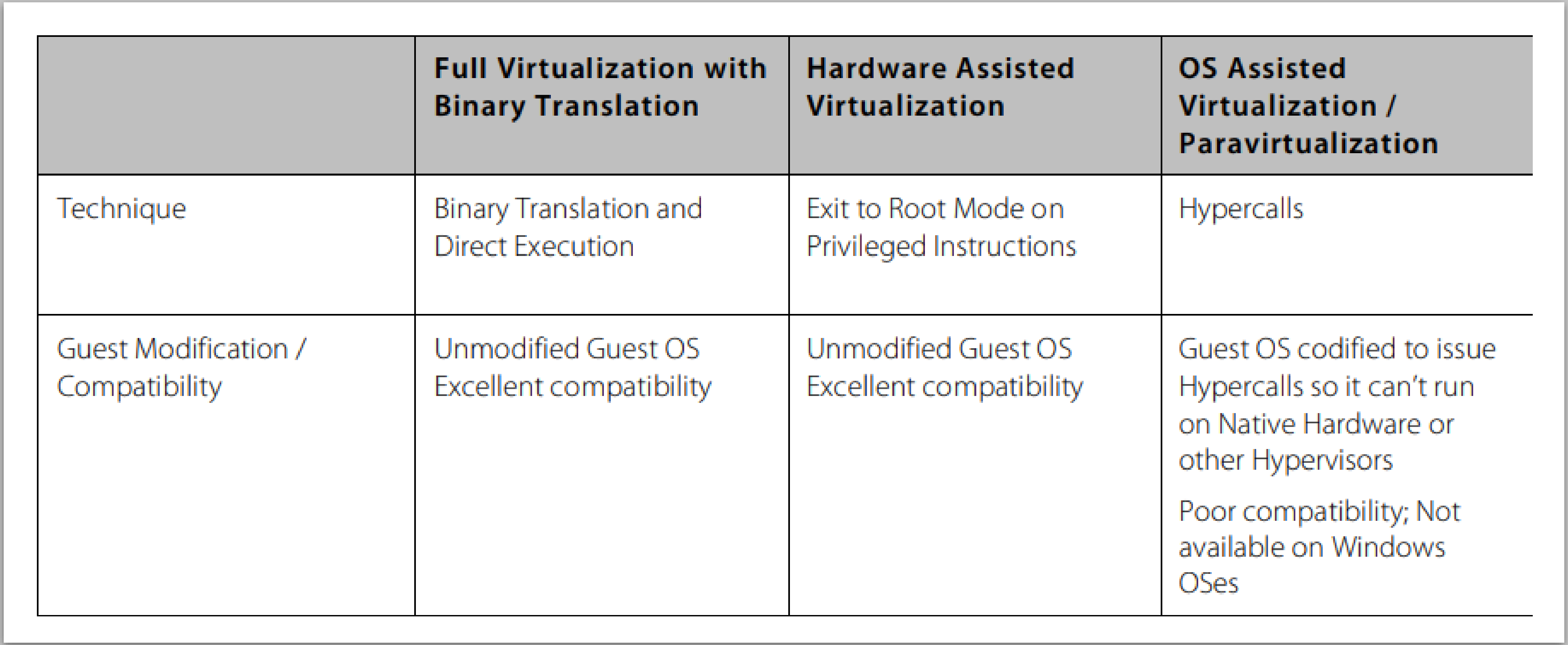
The hardware assist approach to x86 virtualization

Technique 3 – Hardware Assisted Virtualization

- Hardware vendors are rapidly embracing virtualization and developing new features to simplify virtualization techniques
- First generation enhancements include Intel Virtualization Technology (VT-x) and AMD’s AMD-V which both target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0
- Privileged and Sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or paravirtualization

Technique 3 – Hardware Assisted Virtualization

- The guest state is stored in Virtual Machine Control Structures (VT-x) or Virtual Machine Control Blocks (AMD-V)
- Processors with Intel VT and AMD-V became available in 2006, so only newer systems contain these hardware assist features



	Full Virtualization with Binary Translation	**Hardware Assisted Virtualization**	**OS Assisted Virtualization / Paravirtualization**
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes

	Full Virtualization with Binary Translation	**Hardware Assisted Virtualization**	**OS Assisted Virtualization / Paravirtualization**
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen

Home Appliance

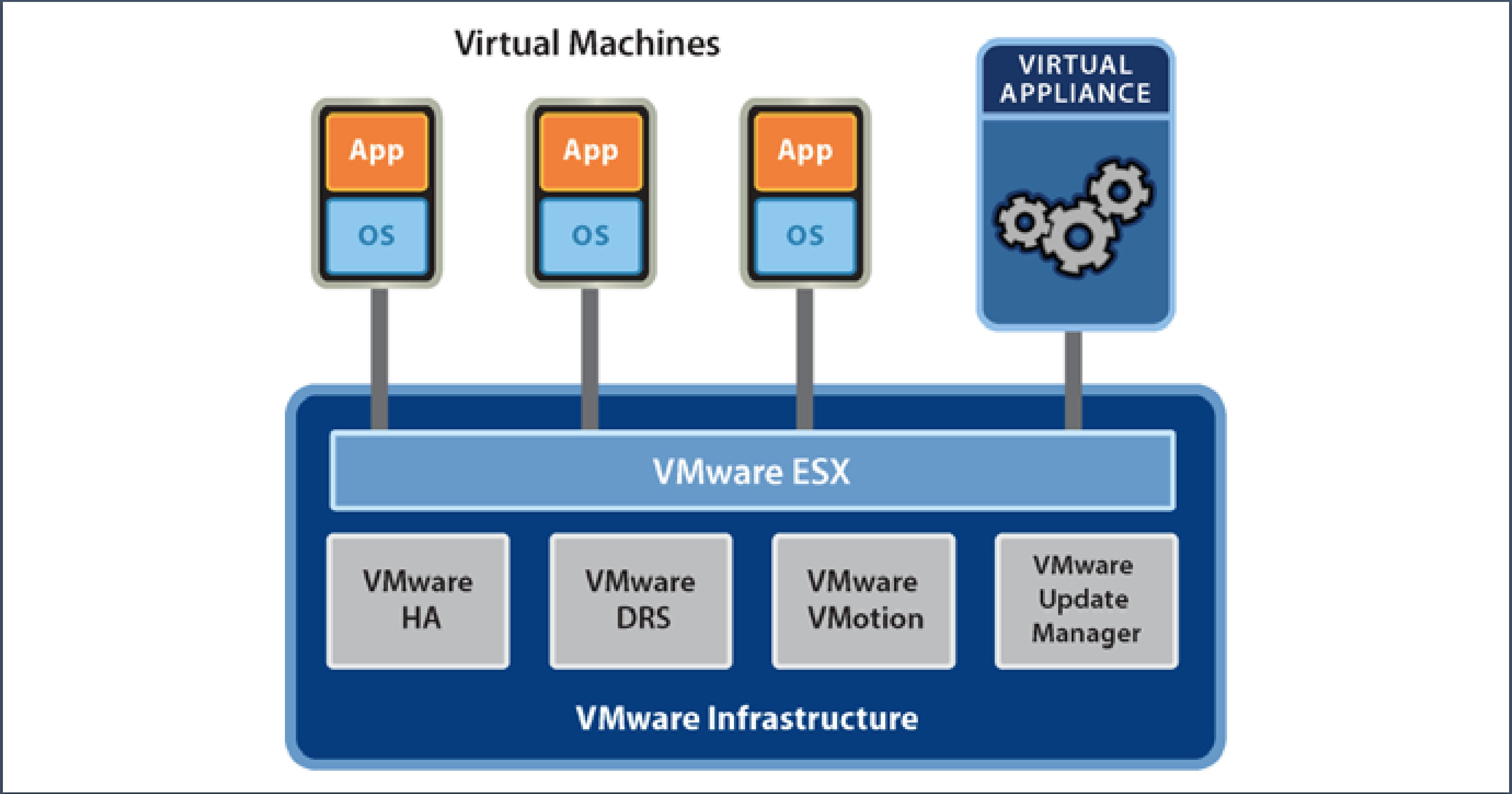
- Home appliances that many of us use in our kitchens and laundry rooms
- Just as a dishwasher **integrates** water sprayers, heating coils, sensors and timers as a prepackaged solution for washing and drying dishware

Virtual Appliance

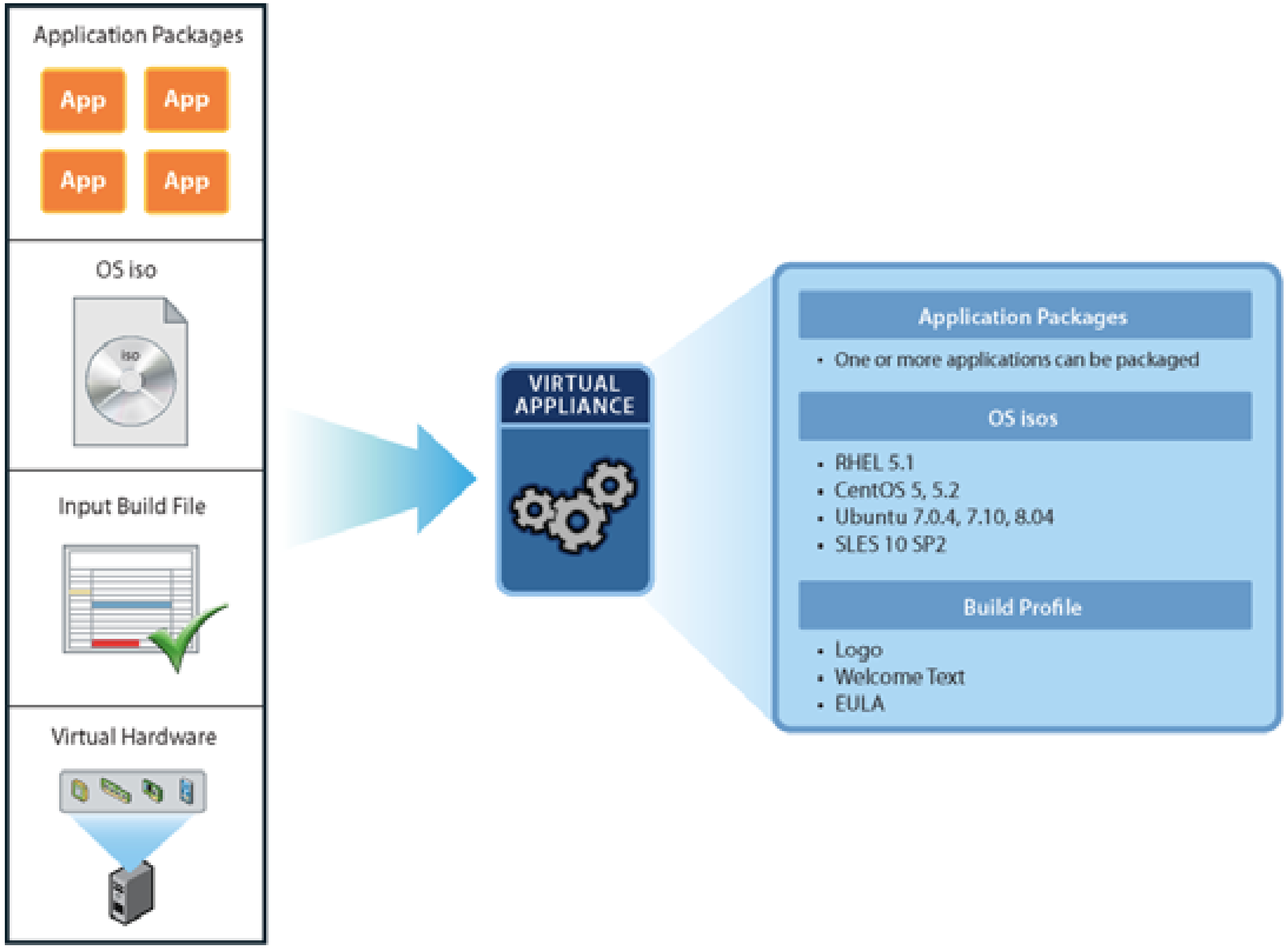
- Pre-configured Software Solution
- A new paradigm for software delivery
- Improves the experience for developers and customers
- By packaging -
  - Pre-configured,
  - Virtualization-ready solutions
- In a single software package that is -
  - Secure, easy to distribute, and easy to manage

Virtual Appliance

- A virtual appliance uses -
  - A pre-installed, pre-configured operating system and an application stack that is **customized to provide a specific set of functions**s
- A virtual appliance typically comes in the open virtualization format (OVF)



A Virtual Appliance Running on VMware Infrastructure



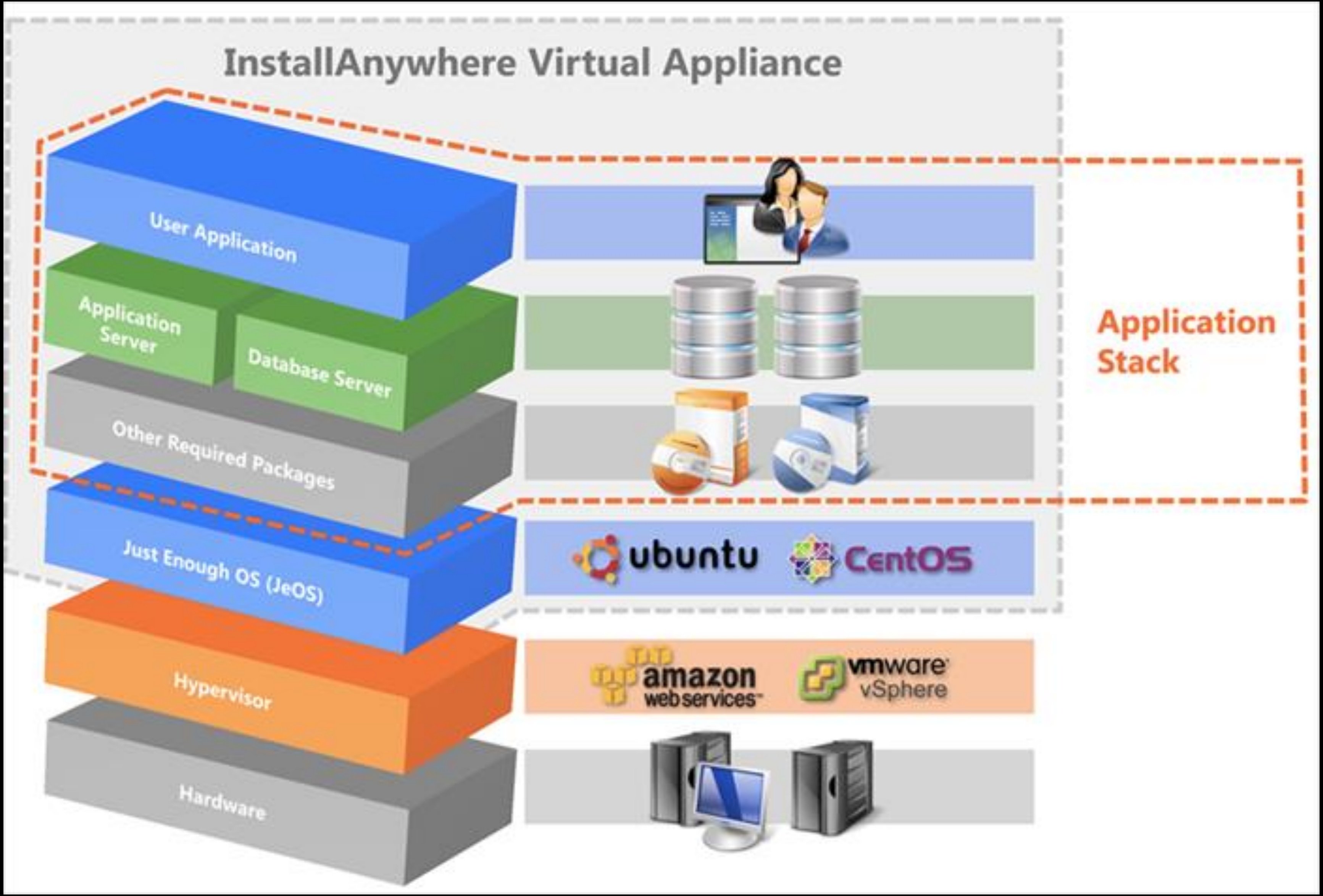
Virtual Appliance Packaging with VMware Studio

Virtual Appliance - Examples

- Virtual appliances in IaaS systems such as **Amazon's** Elastic Compute Cloud (EC2)
- **VMware** Ready virtual appliances created using VMware Studio run seamlessly on VMware products such as
  - VMware Infrastructure, VMware ESXi, VMware Workstation, VMware Fusion™ and VMware Server
  - Third-party virtualization products that support the OVF specification

Components of a Virtual Appliance

- A thin operating system layer called just enough operating system (JeOS)
- Required operating system packages
- Final application
- The self-contained virtual appliance can run directly on hypervisors such as VMware vSphere or Amazon EC2


InstallAnywhere Virtual Appliance
User Application
Application Server
Database Server
Other Required Packages
Just Enough OS (JeOS)
Hypervisor
Hardware
Application Stack
ubuntu
CentOS
amazon web services
vmware vSphere

Virtual Appliances vs. Virtual Machines

- Virtual appliances, like virtual machines, incorporate
  - an application, OS and virtual hardware
- Virtual appliances differ from virtual machines -
  - They are delivered to customers as preconfigured solutions
  - Simplify deployment for customers by eliminating the need for manual configuration of the VMs and OSs used to run the appliance

Virtual Appliance - Benefits

Group	Benefits
Developers and Appliance Vendors	a) Reduce development and distribution costs. b) Accelerate time to market and expand customer reach. c) Strengthen security and improve the customer experience
Customers	i. Reduce the cost of owning and operating software. ii. Accelerate evaluations, deployment and time to value. iii. Leverage integration with virtualization platforms.