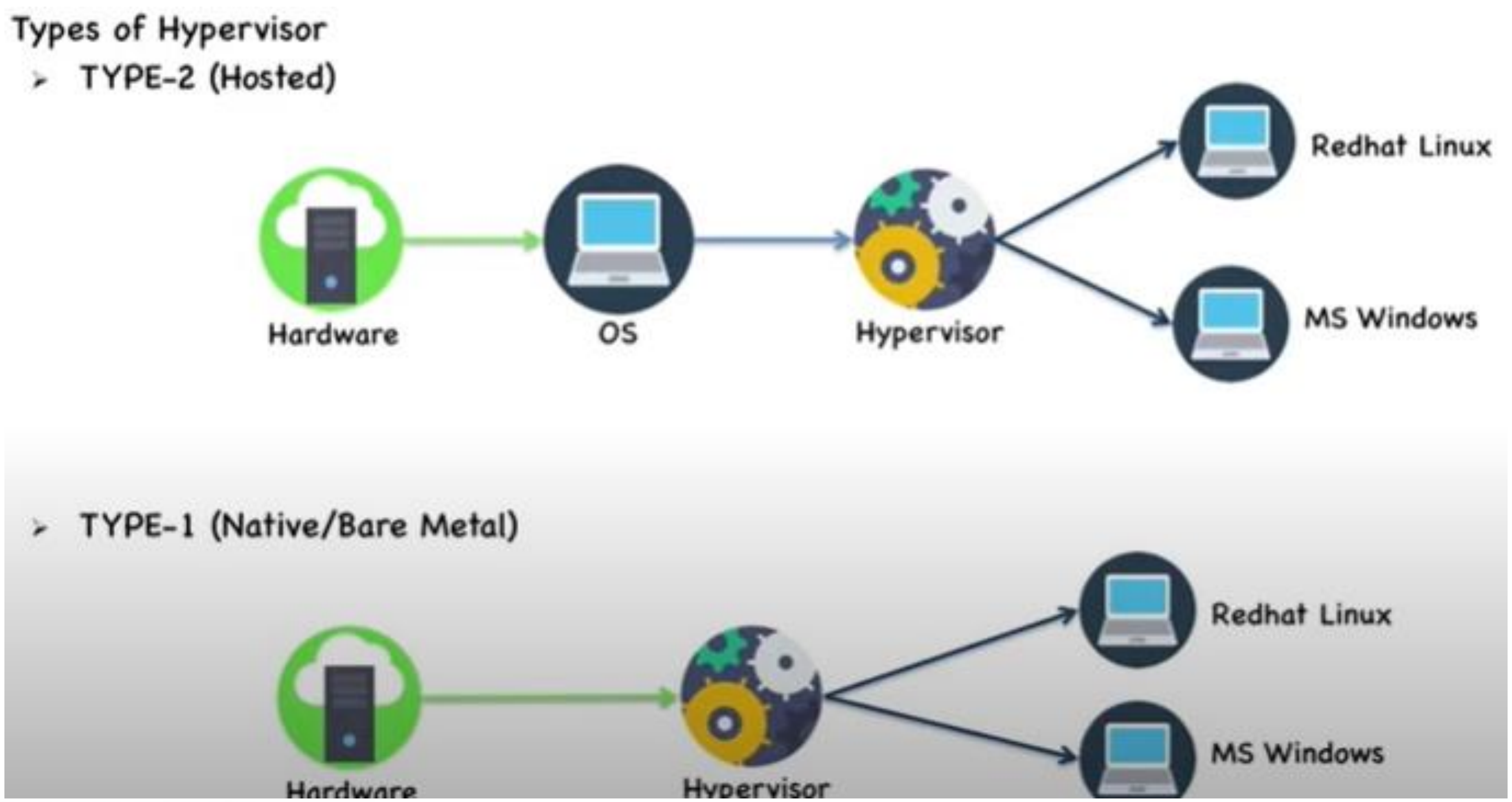
CPU Virtualization

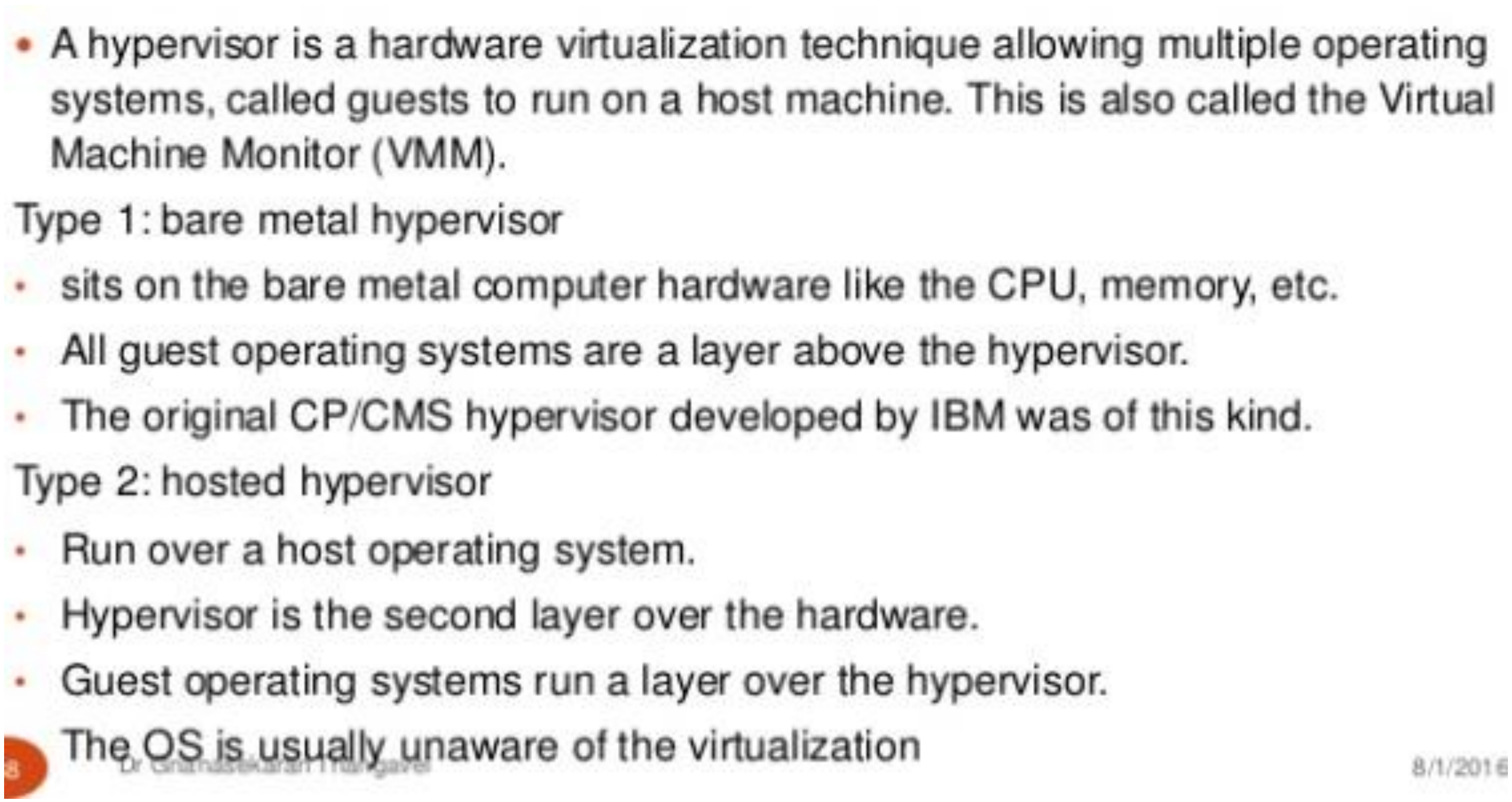
Hypervisors

- Virtualization and Cloud Computing Lecture 2: Review of OS concepts
  https://www.youtube.com/watch?v=ix9Ylli70yY
- Virtualization and Cloud Computing Lecture 4: Hardware-assisted CPU virtualization in KVM/QEMU
  https://www.youtube.com/watch?v=wiSwGRjJN-w
- Virtualization and Cloud Computing Lecture 5: Full Virtualization
  https://www.youtube.com/watch?v=MVxBxg_aNk0

Hypervisors



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).

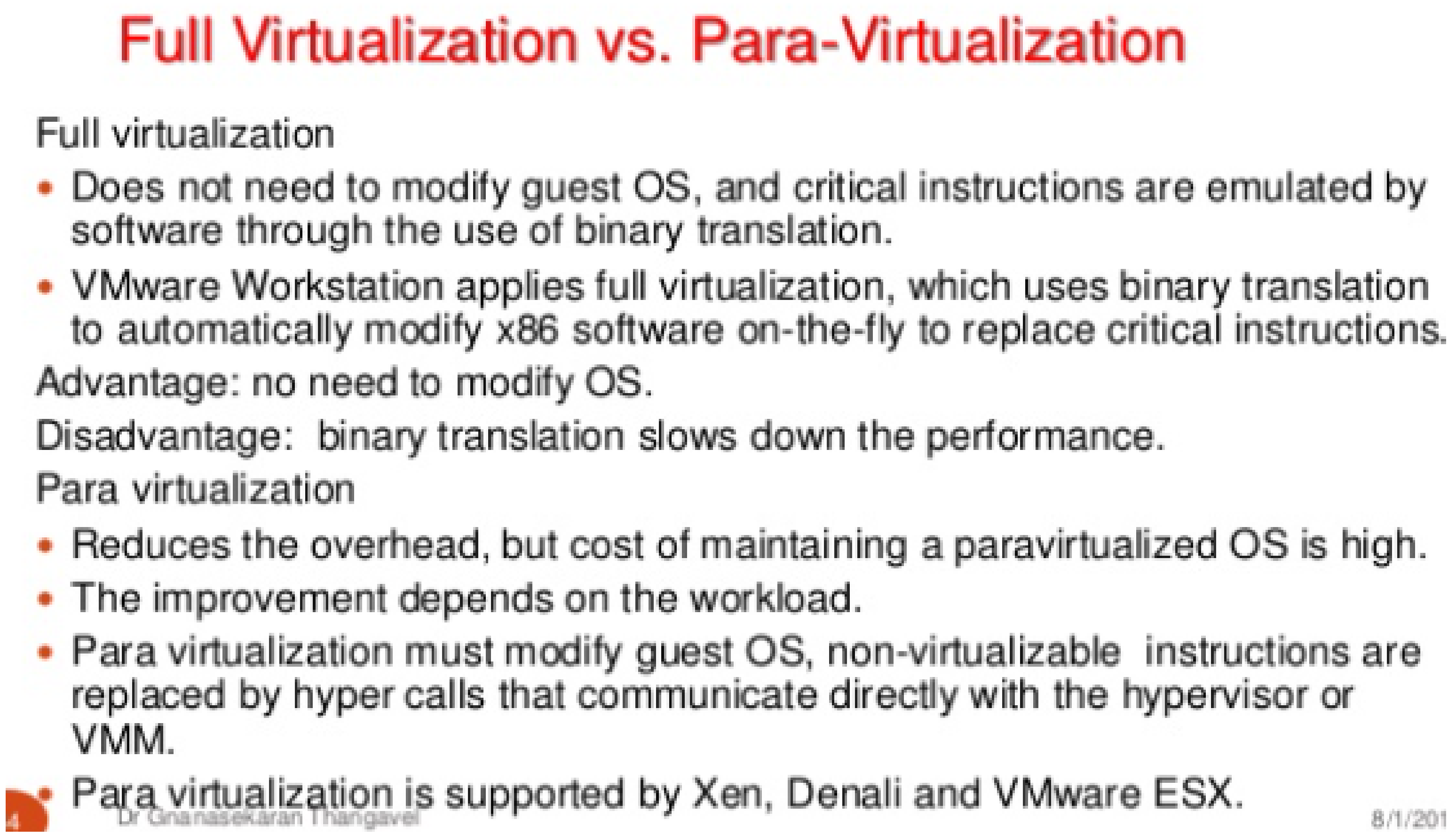
Type 1: bare metal hypervisor

- sits on the bare metal computer hardware like the CPU, memory, etc.
- All guest operating systems are a layer above the hypervisor.
- The original CP/CMS hypervisor developed by IBM was of this kind.

Type 2: hosted hypervisor

- Run over a host operating system.
- Hypervisor is the second layer over the hardware.
- Guest operating systems run a layer over the hypervisor.
- The OS is usually unaware of the virtualization

Hypervisors



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.
- VMware Workstation applies full virtualization, which uses binary translation to automatically modify x86 software on-the-fly to replace critical instructions.

Advantage: no need to modify OS.

Disadvantage: binary translation slows down the performance.

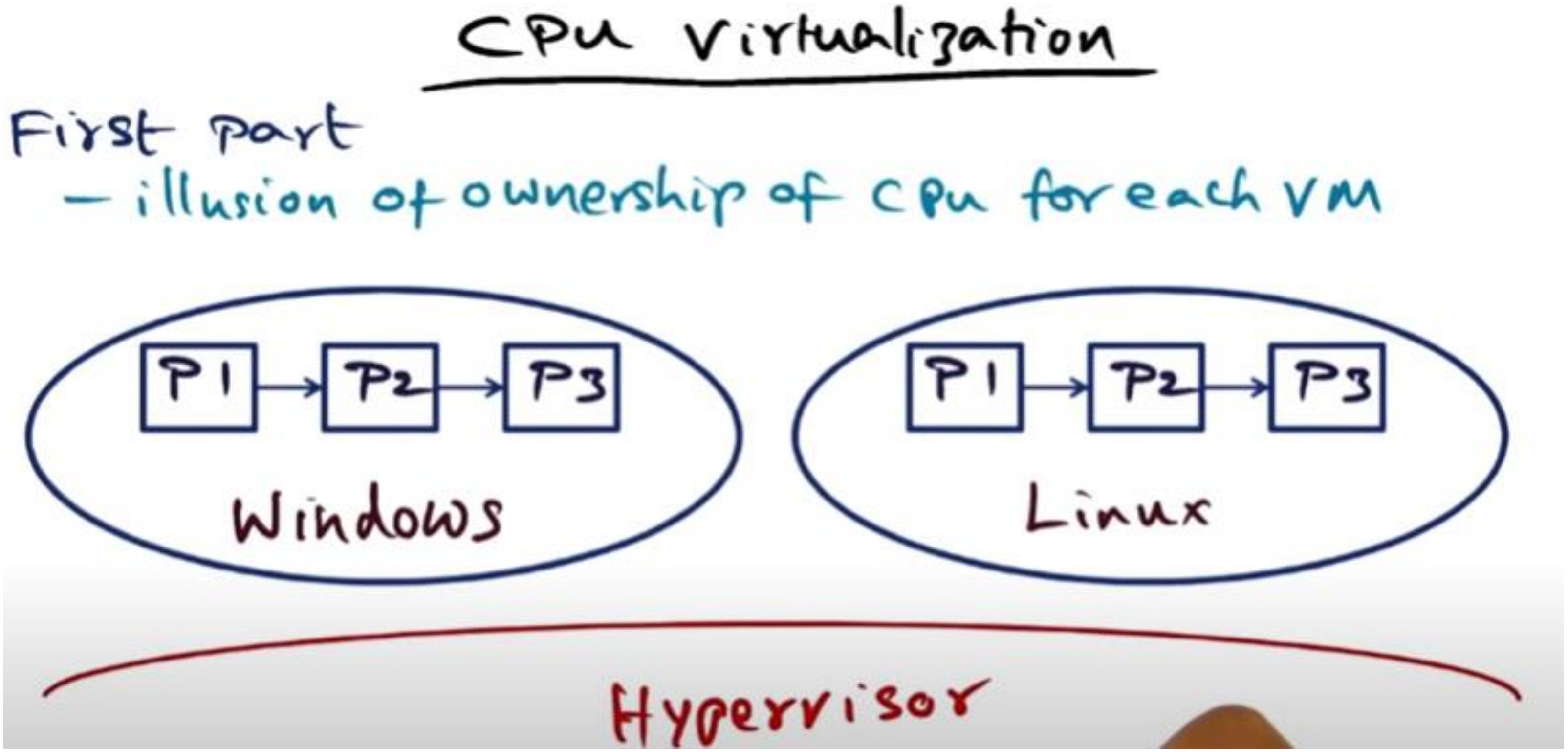
Para virtualization

- Reduces the overhead, but cost of maintaining a paravirtualized OS is high.
- The improvement depends on the workload.
- Para virtualization must modify guest OS, non-virtualizable instructions are replaced by hyper calls that communicate directly with the hypervisor or VMM.
- Para virtualization is supported by Xen, Denali and VMware ESX.

CPU Virtualization

- With CPU Virtualization, all the virtual machines act as physical machine and distribute their hosting resources just like having various virtual processors.
- Sharing of physical resources takes place to each virtual machine when all hosting services get the request.
- Finally, the virtual machines get a share of the single CPU allocated to it, being a single-processor acting as dual-processor.

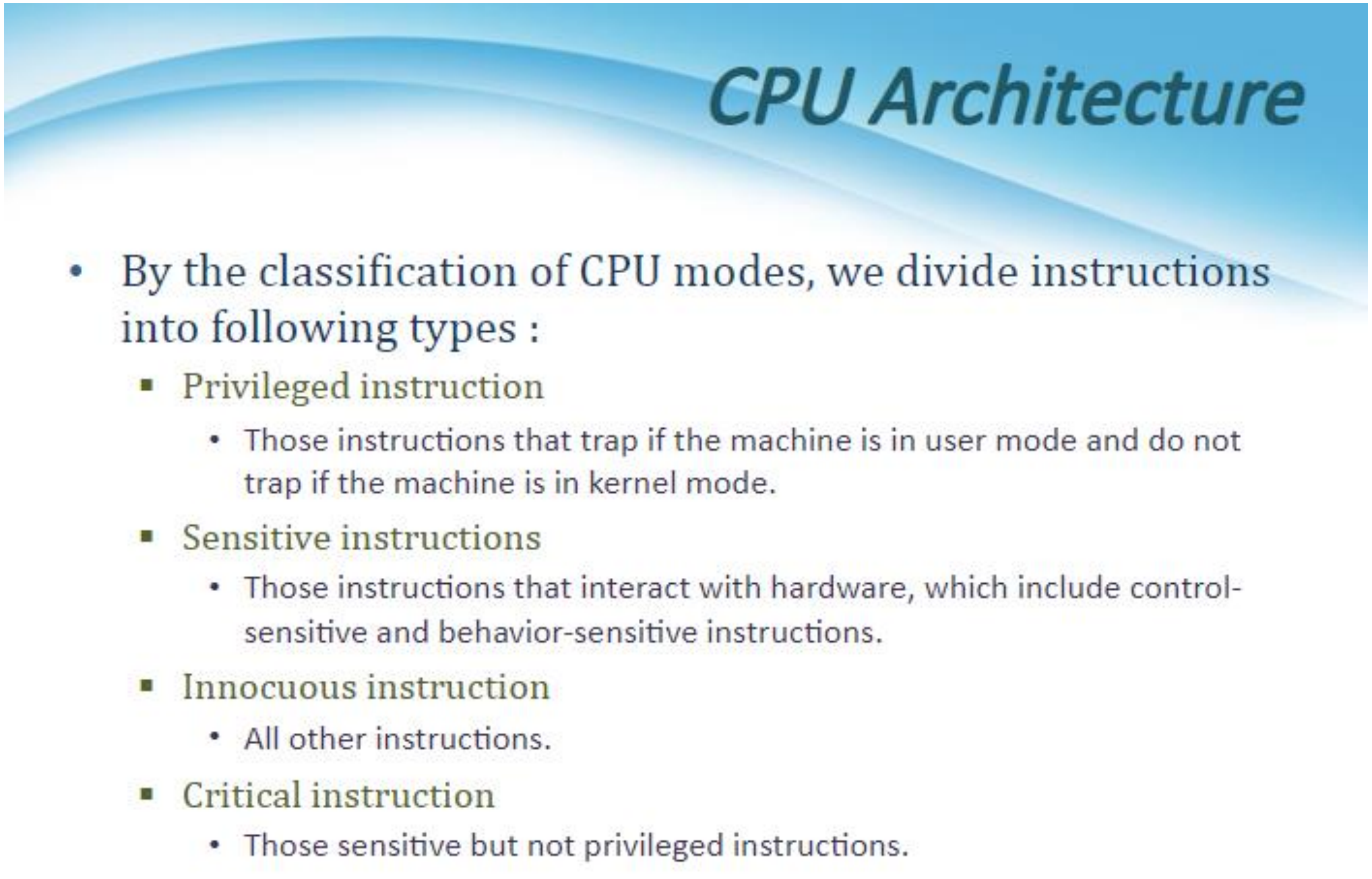
CPU Virtualization



CPU

Virtualization

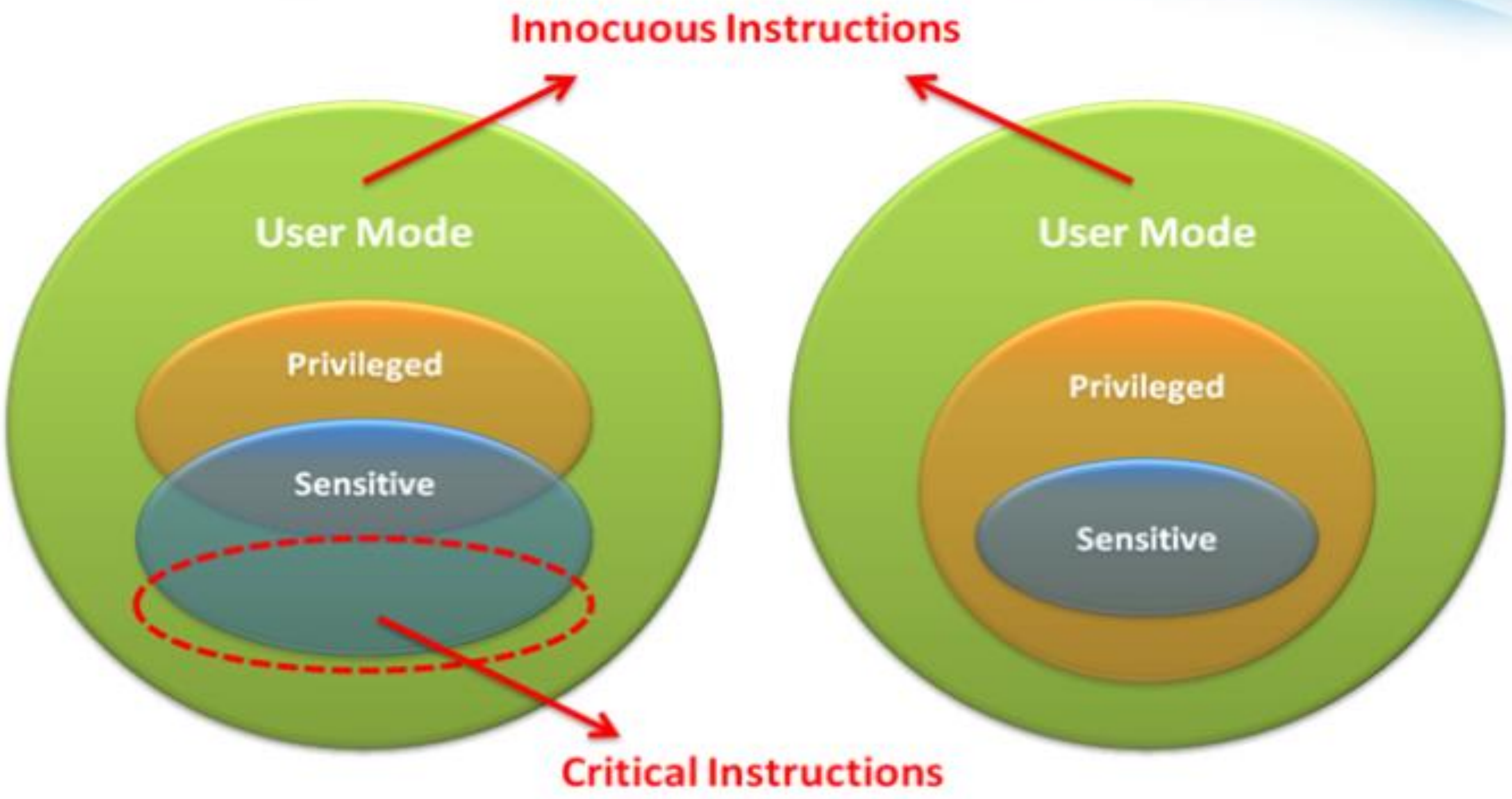
- All processors runs
  - User mode
    - Unprivileged instructions(UI).
    - UI of VMs run directly on the host machine for higher efficiency.
  - Supervisor mode
    - to ensure controlled access of critical hardware.
    - privileged instructions.



CPU Architecture

- By the classification of CPU modes, we divide instructions into following types :
  - Privileged instruction
    - Those instructions that trap if the machine is in user mode and do not trap if the machine is in kernel mode.
  - Sensitive instructions
    - Those instructions that interact with hardware, which include control-sensitive and behavior-sensitive instructions.
  - Innocuous instruction
    - All other instructions.
  - Critical instruction
    - Those sensitive but not privileged instructions.

CPU Architecture

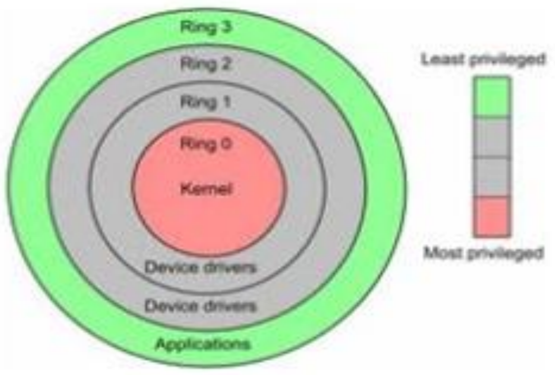


Instructions

- Unprivileged instructions
- Critical instructions
  - Privileged instructions
    - Execute in a privileged mode and will be trapped if executed outside this mode.
  - Control-sensitive instructions
    - Attempt to change the configuration of resources used.
  - Behavior-sensitive instructions.
    - Different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

CPU Architecture

- Modern CPU status is usually classified as several modes.
- In general, we conceptually divide them into two modes :
  - Kernel mode (Ring 0)
    - CPU may perform any operation allowed by its architecture, including any instruction execution, IO operation, area of memory access, and so on.
    - Traditional OS kernel runs in Ring 1 mode.
  - User mode (Ring 1 ~ 3)
    - CPU can typically only execute a subset of those available instructions in kernel mode.
    - Traditional application runs in Ring 3 mode.



Virtualization

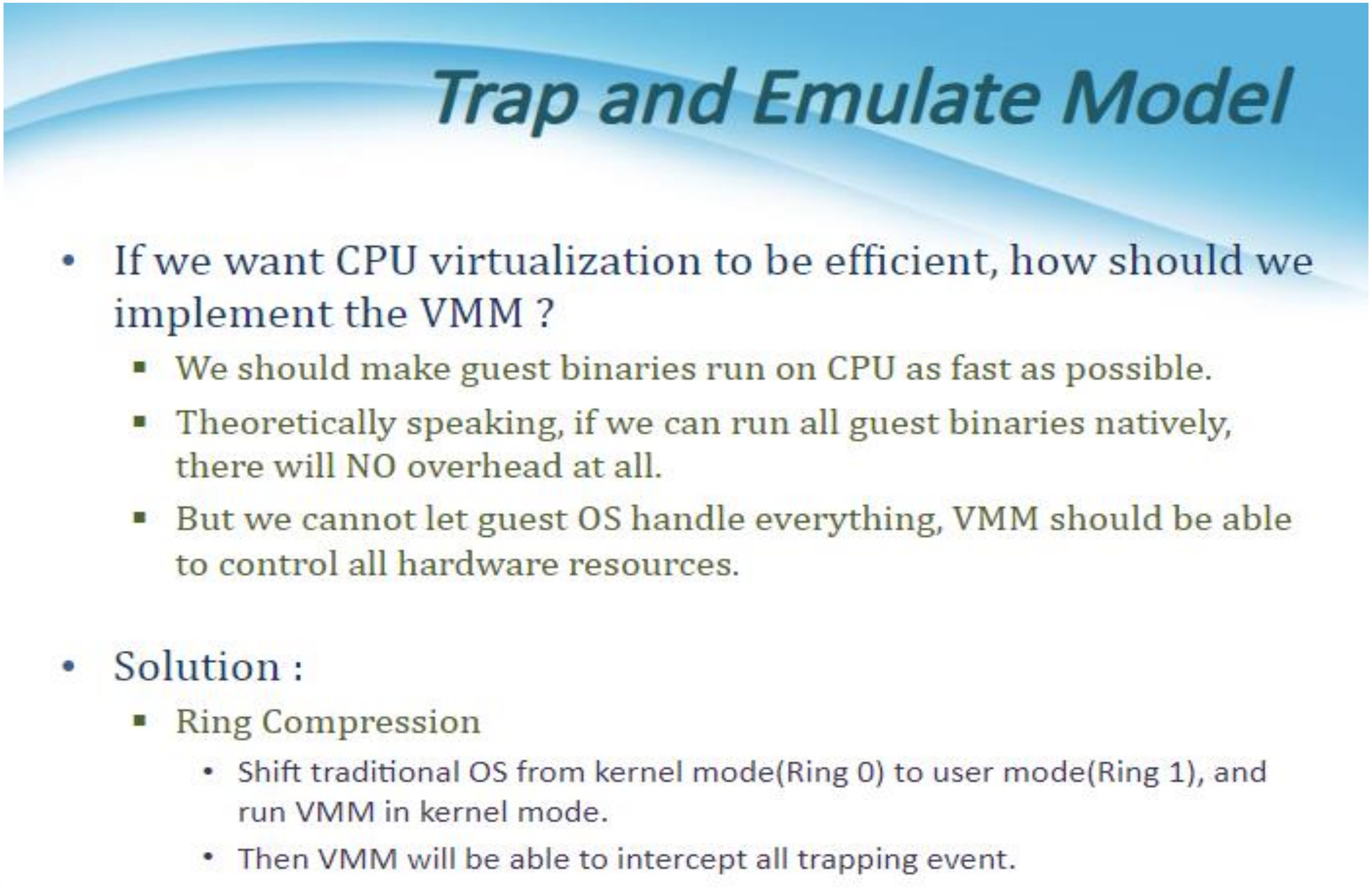
- A CPU architecture is virtualizable
- If it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.
- When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM.
- VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system.

Software based CPU Virtualization

- Software-based where with the help of it, application code gets executed on the processor and the privileged code gets translated first and that translated code gets executed directly on the processor.
- This translation is purely known as Binary Translation (BT).
- The guest programs that are based on unprivileged coding runs very smooth and fast.
- The code programs or the applications that are based on privileged code components that are significant such as system calls, run at a slower rate in the virtual environment.

CPU Architecture

- What is trap ?
  - When CPU is running in user mode, some internal or external events, which need to be handled in kernel mode, take place.
  - Then CPU will jump to hardware exception handler vector, and execute system operations in kernel mode.
- Trap types :
  - System Call
    - Invoked by application in user mode.
    - For example, application ask OS for system IO.
  - Hardware Interrupts
    - Invoked by some hardware events in any mode.
    - For example, hardware clock timer trigger event.
  - Exception
    - Invoked when unexpected error or system malfunction occur.
    - For example, execute privilege instructions in user mode.



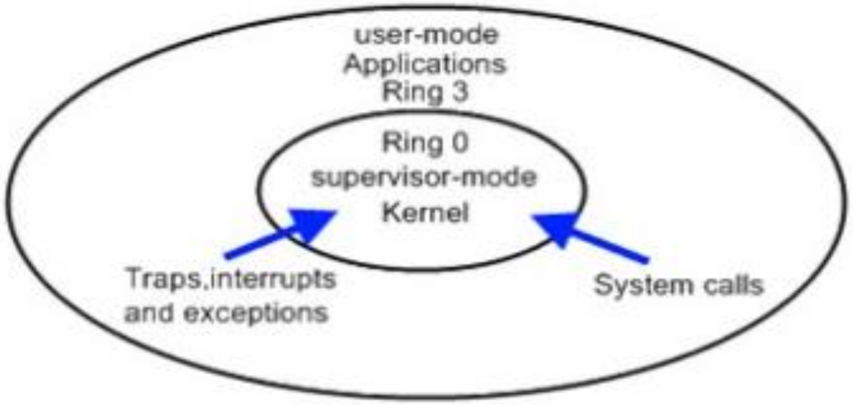
Trap and Emulate Model

- If we want CPU virtualization to be efficient, how should we implement the VMM ?
  - We should make guest binaries run on CPU as fast as possible.
  - Theoretically speaking, if we can run all guest binaries natively, there will NO overhead at all.
  - But we cannot let guest OS handle everything, VMM should be able to control all hardware resources.

- Solution :
  - Ring Compression
    - Shift traditional OS from kernel mode(Ring 0) to user mode(Ring 1), and run VMM in kernel mode.
    - Then VMM will be able to intercept all trapping event.

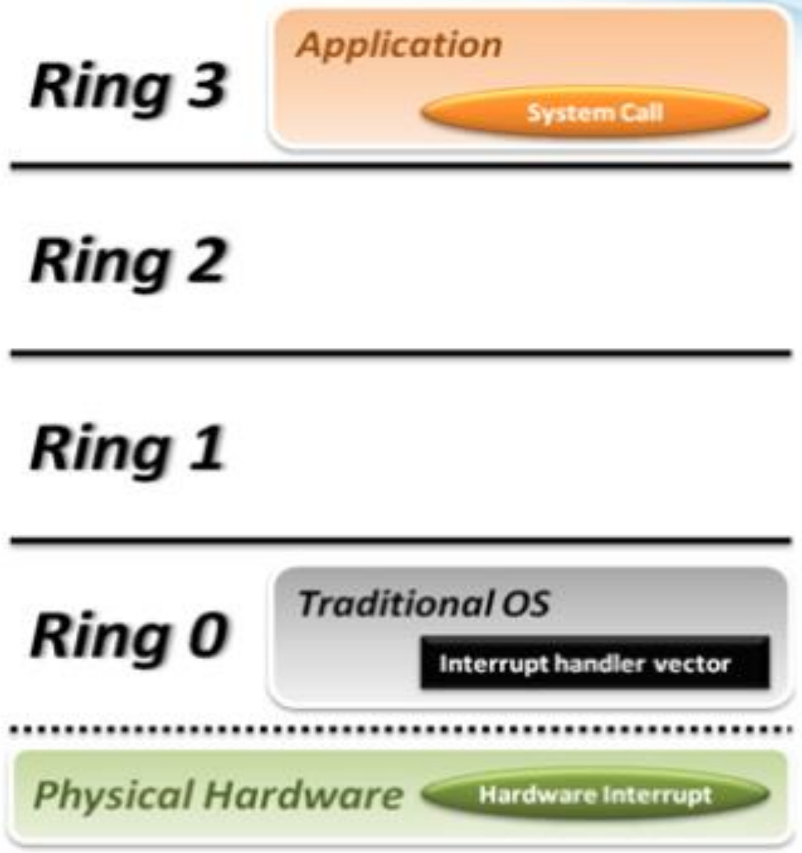
Trap and Emulate Model

- VMM virtualization paradigm *(trap and emulate)*:
  1. Let normal instructions of guest OS run directly on processor in user mode.
  2. When executing privileged instructions, hardware will make processor trap into the VMM.
  3. The VMM emulates the effect of the privileged instructions for the guest OS and return to guest.



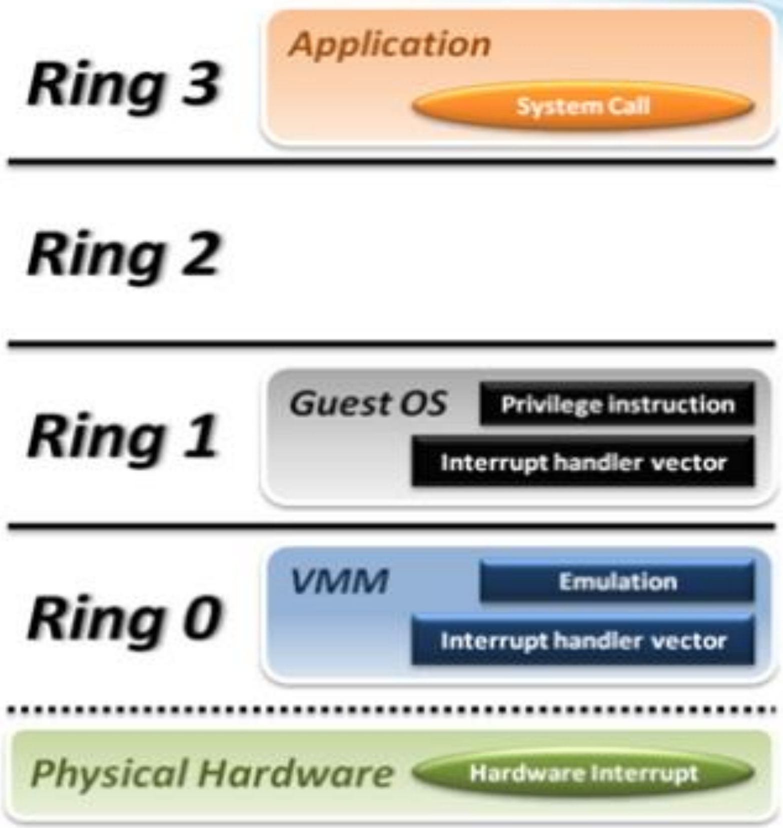
Trap and Emulate Model

- Traditional OS :



Trap and Emulate Model

- VMM and Guest OS :



Hardware-Assisted Virtualization

- There is hardware that gets assistance to support CPU Virtualization from certain processors.
- The guest user uses a different version of code and mode of execution known as a guest mode.
- The guest code mainly runs on guest mode.
- System calls runs faster than expected.
- Workloads that require updation of page tables get a chance of exiting from guest mode to root mode that eventually slows down the performance and efficiency of the program.

Hardware Solution

- Let's go back to trap model :
  - Some trap types do not need the VMM involvement.
    - For example, all system calls invoked by application in guest OS should be caught by gust OS only. There is no need to trap to VMM and then forward it back to guest OS, which will introduce context switch overhead.
  - Some critical instructions should not be executed by guest OS.
    - Although we make those critical instructions trap to VMM, VMM cannot identify whether this trapping action is caused by the emulation purpose or the real OS execution exception.

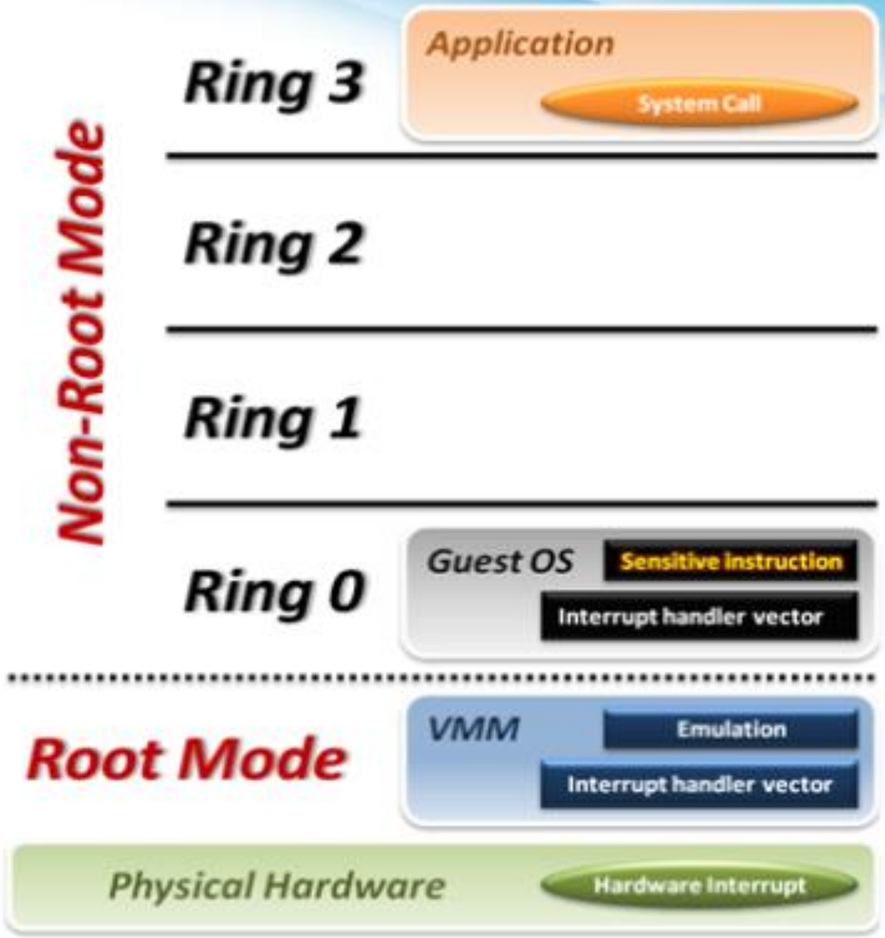
- Solution :
  - We need to redefine the semantic of some instructions.
  - We need to introduce new CPU control paradigm.

Intel VT-x

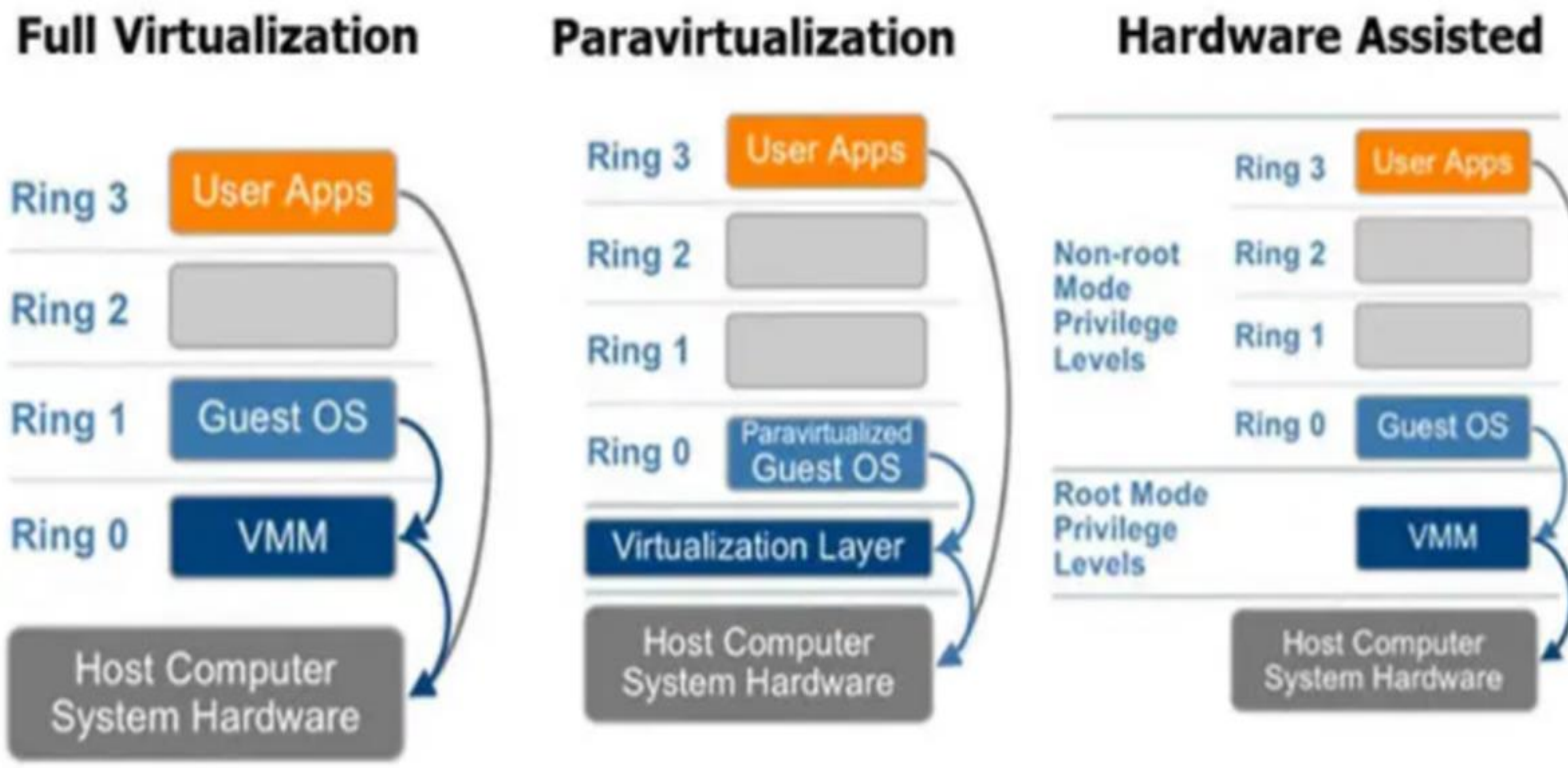
- In order to straighten those problems out, Intel introduces one more operation mode of x86 architecture.
  - VMX Root Operation (Root Mode)
    - All instruction behaviors in this mode are no different to traditional ones.
    - All legacy software can run in this mode correctly.
    - VMM should run in this mode and control all system resources.
  - VMX Non-Root Operation (Non-Root Mode)
    - All sensitive instruction behaviors in this mode are redefined.
    - The sensitive instructions will trap to Root Mode.
    - Guest OS should run in this mode and be fully virtualized through typical "*trap and emulation model*".

Intel VT-x

- VMM with VT-x :



Architectural Comparison



Benefits of CPU Virtualization

- Overall performance and efficiency are improved
- Security: The VM machines are also kept separate from each other and because of that any cyber-attack or software glitch unable to create damage to the system, as a single machine cannot affect another machine.
- Cost is very less
- It provides the best backup of computing resources since the data is stored and shared from a single system.
- It also offers great and fast deployment procedure options

Virtualization

- Server Virtualization:
  https://slideplayer.com/slide/5103233/
- https://www.educba.com/cpu-virtualization/
- http://www.brainkart.com/article/Virtualization-of-CPU,-Memory,-and-I-O-Devices_11337/
- https://www.slideshare.net/drgst/cs6703-grid-and-cloud-computing-unit-3