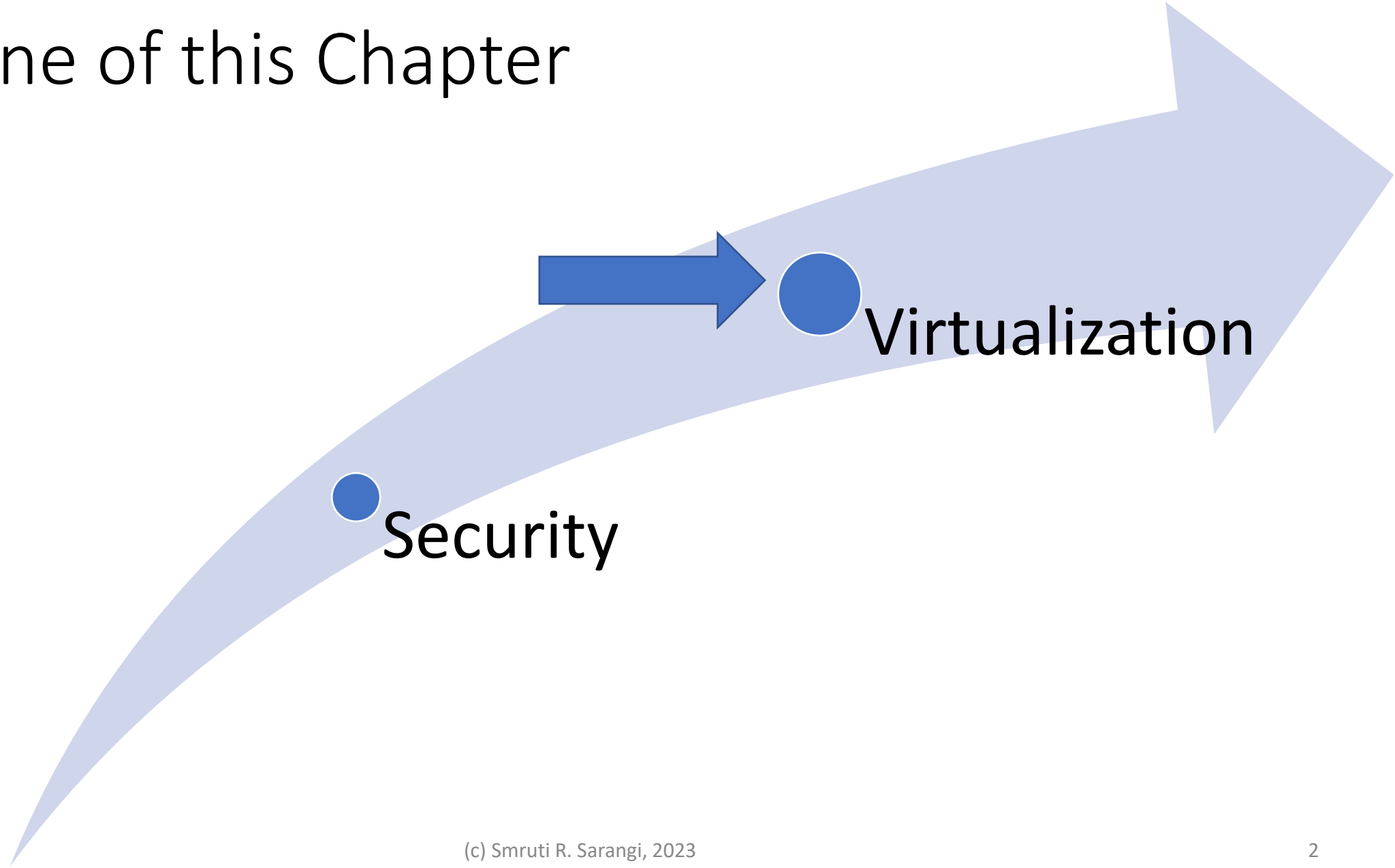
Chapter 8: Security and Virtualization

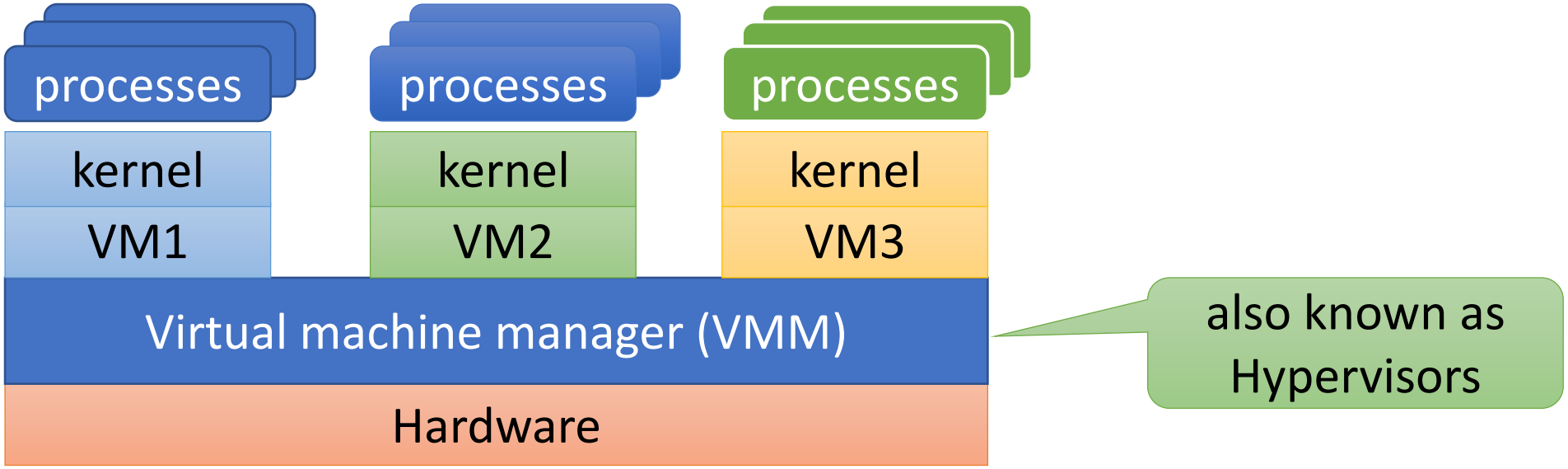
Smruti R Sarangi

IIT Delhi

Outline of this Chapter



Run an OS as an Application



Types of Hypervisors

Type	Explanation	Examples
Type 0	Hardware or Firmware-based VMM. Minimal software support	IBM LPAR, Oracle LDOM
Type 1	An OS that allows us to run other guest OSes. Provides native support and maps guest addresses to physical memory directly.	Linux KVM, VMWare ESX, Citrix XenServer
Type 2	Run a guest OS as a regular application	VMPlayer, VirtualBox
Para-virtualization	The code of the guest OS is changed (unlike the previous cases) and made VM aware	Xen

Advantages of VMs in Cloud Environments

Create custom environments

- Create custom **environments** (OS + libraries + software) and run them on any machine on a cloud.

Virtualize the CPU, network, file system, memory, and hardware

- Transparently migrate the VM across different types of machines

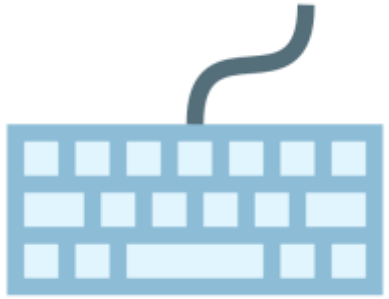
Server consolidation

- Live migration of VMs, efficiently heterogeneous server resources

Virtualization

CPU

Memory



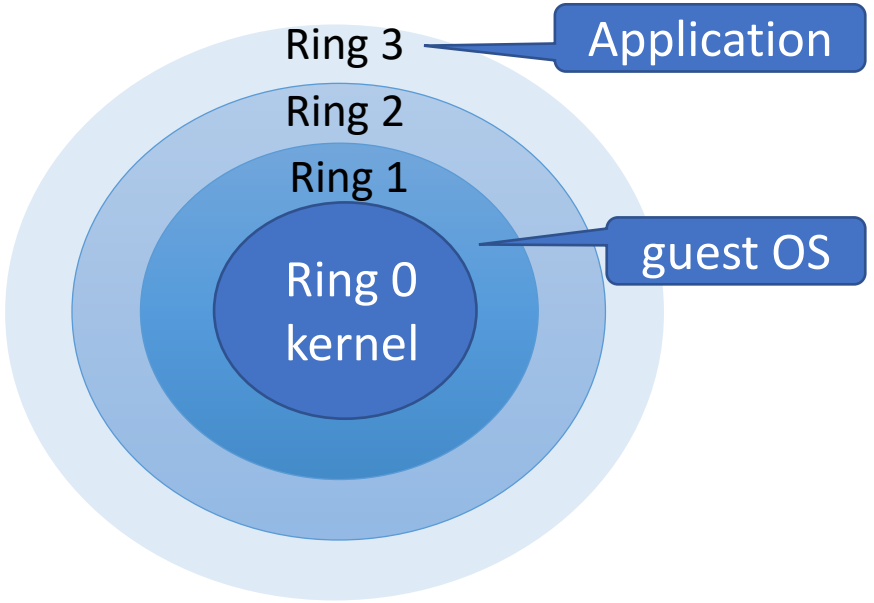
i/o

storage

The guest OS has no idea that it is running on a hypervisor.

The OS needs to think that it exclusively owns these devices

Virtualize the CPU



Regular instructions

Privileged instructions

Trap and emulate: The guest OS runs with user privileges. Invoking a privileged inst. leads to a trap. The hypervisor catches it and successfully emulates the instruction.

Instructions that execute differently based on the privilege level

Binary translation

Binary Translation

- The hypervisor follows a read-ahead scheme.
  - Whenever a new code page is accessed, a fault is generated (either the TLB entry is missing or the page is marked non-accessible)
  - The hypervisor reads the page and a few more subsequent pages
  - It scans all the instructions.
  - If there is a sensitive instruction (changes its behavior based on the privilege level), replace it with an alternative sequence
  - Make it generate a trap or make the behavior the same (guest & kernel)

What about memory?

Type 1 Hypervisors

TLB entry

GVA → HPA

Guest Application

Guest OS

Host OS

Type	Explanation
GVA	Guest virtual addr.
GPA	Guest physical addr.
HVA	Host virtual addr.
HPA	Host physical addr.

Nested paging

1. TLB miss → walk the guest and host page tables. (slow)
2. No need to track guest page table modifications.
3. VMM maintains only 1 page table.

Shadow paging

1. The VMM builds a GVA → HPA page table (one per guest process).
2. Keeps the shadow page table consistent with both page tables. TLB Miss: Access just the shadow page table (*fast*)
3. Track every guest page table update

I/O Virtualization

One network card appears as several virtual network cards (one for each VM)

Emulation: The hypervisor traps every I/O instruction and memory-mapped ld/st instruction (make pages non-accessible). Follows the trap-and-emulate method. Every guest OS uses emulated devices. [slow]

With H/W support (PCI-X devices): Use the SR-IOV and MR-IOV protocols. A single device exposes separate buffers (request queues), interrupts and DMA streams to each VM. MR-IOV allows sharing the resources with multiple computers.

Paravirtualization

Why not modify the guest operating system to make it VM aware?

- Xen [1] was the first popular para-virtualized hypervisor. Guest OSes communicate via system calls (also known as hypercalls).
- Avoid instructions that change their behavior based on the privilege level
- Ensure that either all privileged instructions either trap in VM mode or are replaced with non-privileged variants that do the same job (may invoke VMM routines)
- Use "fast exception handlers" that are directly registered in the CPU's exception and system call table. Page fault handlers need to be there with Xen (HW reasons 😳)
- Guest OSes can directly write into the CR3 register and change the current page table address (mapping: GVA → HPA). Updates to the page table need to go through the hypervisor. Xen maintains a shadow page table for each process and it updates it when the guest OS lets it know.
- Interrupts are sent as lightweight events to the guest OS (similar to signals)
- I/O: Create a bounded queue (request buffer) between the guest OS and Xen (aka I/O ring because it is a circular queue).

HW-Assisted Virtualization

Intel VT-x | AMD-V

Event	Action
Privileged instruction execution	Most of the time it is allowed like *syscall enter* and *exit* because in this case, VMs run in Ring 0 (albeit in non-root mode or VM mode). Some I/O instructions necessitate a VM exit. The device needs to be emulated.
Memory accesses	Use shadow page tables that are directly stored in HW (fast path). The CPU walks both the guest and host page tables on a TLB miss and updates the shadow page table (if there has been an update to either page table, slow path).
I/O	Use the SR-IOV and MR-IOV protocols
Storage	The hypervisor maintains a guest-to-host block table.

Bibliography

1. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). Association for Computing Machinery, New York, NY, USA, 164–177. https://doi.org/10.1145/945445.945462

srsarangi@cse.iitd.ac.in