Virtualisation

Summer student lecture 2008

Håvard Bjerke

CERN openlab

Virtualisation is old...

- IBM mainframes in the '60s supported it
- Then PCs became commodity
  - Commodity computers became cheaper
  - Cheap computers became fast
  - PC architecture became de facto
- Result: Virtualisation was forgotten
  - Efficient virtualisation became hard

The resurrection

- Resurrection
  - VMWare and a few other vendors
  - Made virtualisation feasible on x86, but relatively inefficient
- Revolution
  - Xen: Para-virtualisation
  - Efficient: close to native performance
  - Forget MS Windows – since Linux is open, we can hack it to support para-virtualisation

Overview

- Virtual machines
- Benefits of virtualisation
- Computer architecture
  - Memory management
  - Privilege separation
  - Interrupts
- Hardware Virtualisation
- Para-virtualisation
- The future

Virtual Machines

- Software level
  - Example: Java
  - Offers software compatibility across platforms
- Hardware level
  - Example: VMWare
  - Multiple OS instances on a single physical machine

Important Concepts

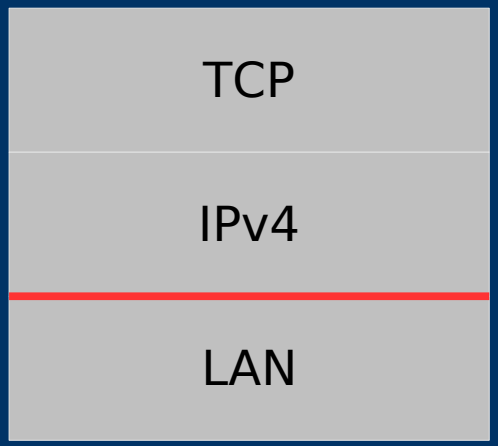
- Encapsulation
  - The Virtual Machine Monitor (VMM) *encapsulates* the VM
  - i.e. it knows everything that's happening inside the VM
  - It can control and optimize execution of the VM
- Isolation
  - The execution of one VM domain should not adversely affect execution of another domain

Stack Abstraction

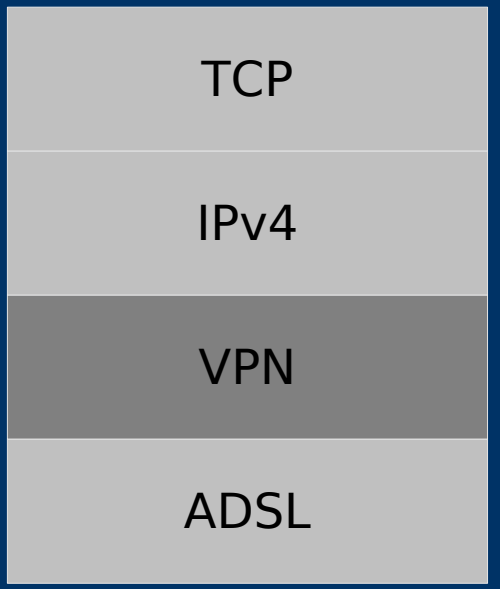
- Stack abstraction example: The OSI model
- Each layer is independent and can be implemented differently by different vendors

Abstraction vs Virtualisation

- Abstraction
  - TCP/IP stack
  - Replaceable layers
  - But: Friction between layers

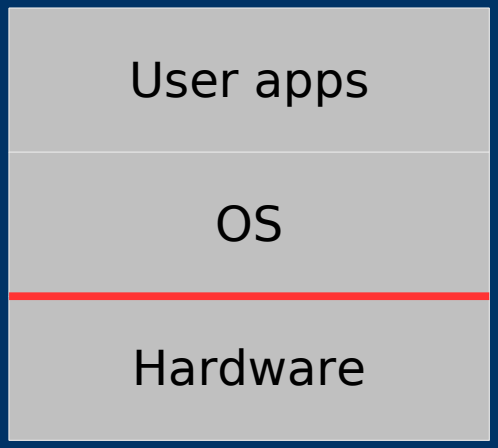


- Virtualisation
  - Virtual Private Networking (VPN)



Abstraction vs Virtualisation

- Computer abstraction layers



- Computer virtualisation

Benefits of HW virtualisation

- General application:
  - Server consolidation
- Specific for shared computing infrastructure, e.g. Grid and HPC:
  - Software flexibility
    - Let each user manage their own OS
    - And satisfy their own software dependencies
  - Flexible allocation of SMP and multi-core resources
  - Secure isolation between users
  - Migration between nodes
  - Checkpointing
  - Time sharing, scavenging of idle resources

How?

- Difficult engineering task
- Several aspects of hardware need to be virtualised
  - CPU
  - Memory management
    - Virtual memory
    - Page directories and tables
    - Legacy memory modes
      - Segmentation
      - Real mode
    - Physical memory
  - I/O

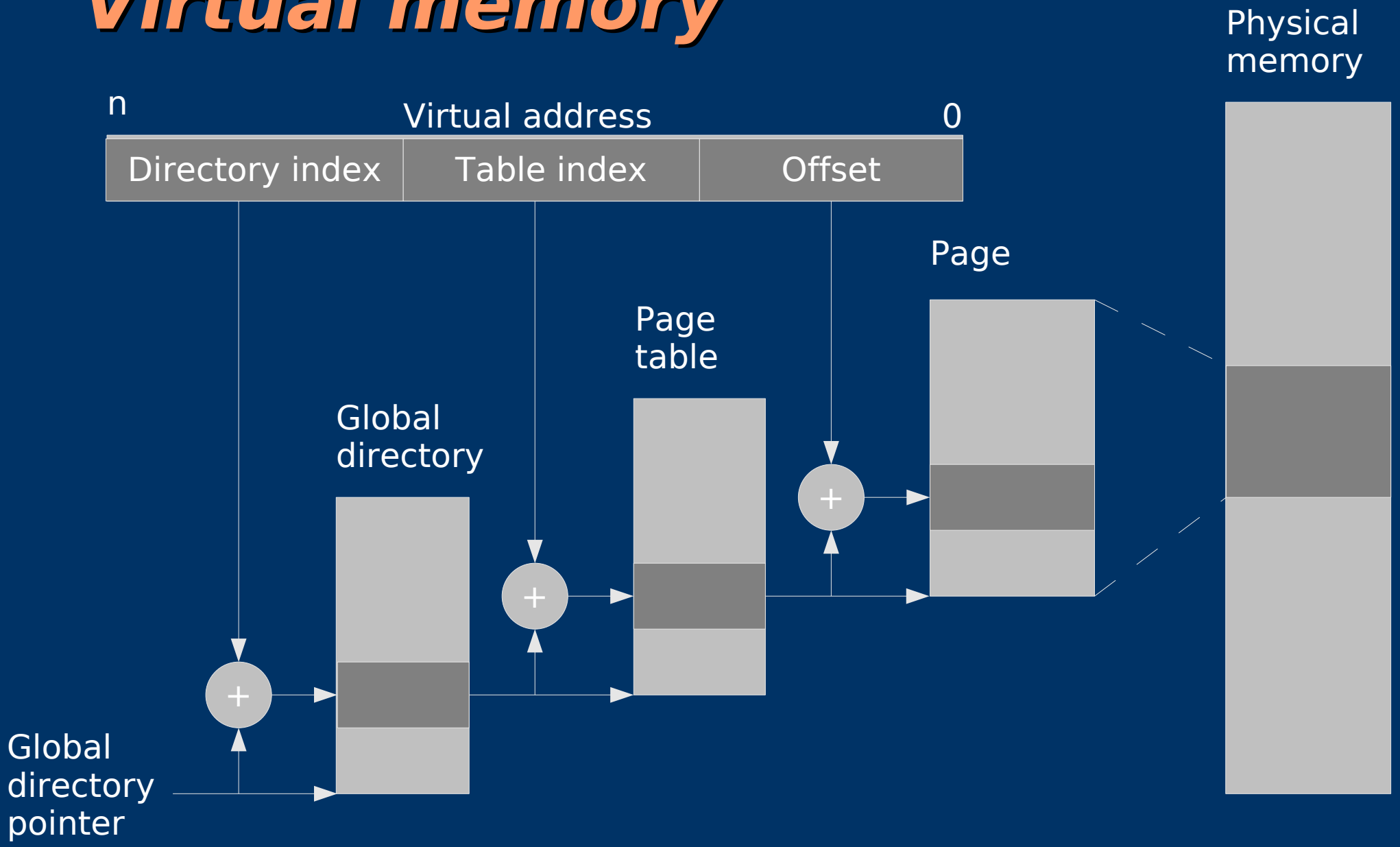
Computer architecture

- Commodity architectures: Intel-32, Intel-64, IA-64
- Virtual memory
- Translation Lookaside buffer
- Privilege separation
- Interrupts and exceptions

Virtual memory

- Simplifies memory management for application programmers
  - Single flat address space per process
  - Memory management is handled by the kernel
    - Mapping to physical memory
    - Protection
- Allows overcommit by swapping

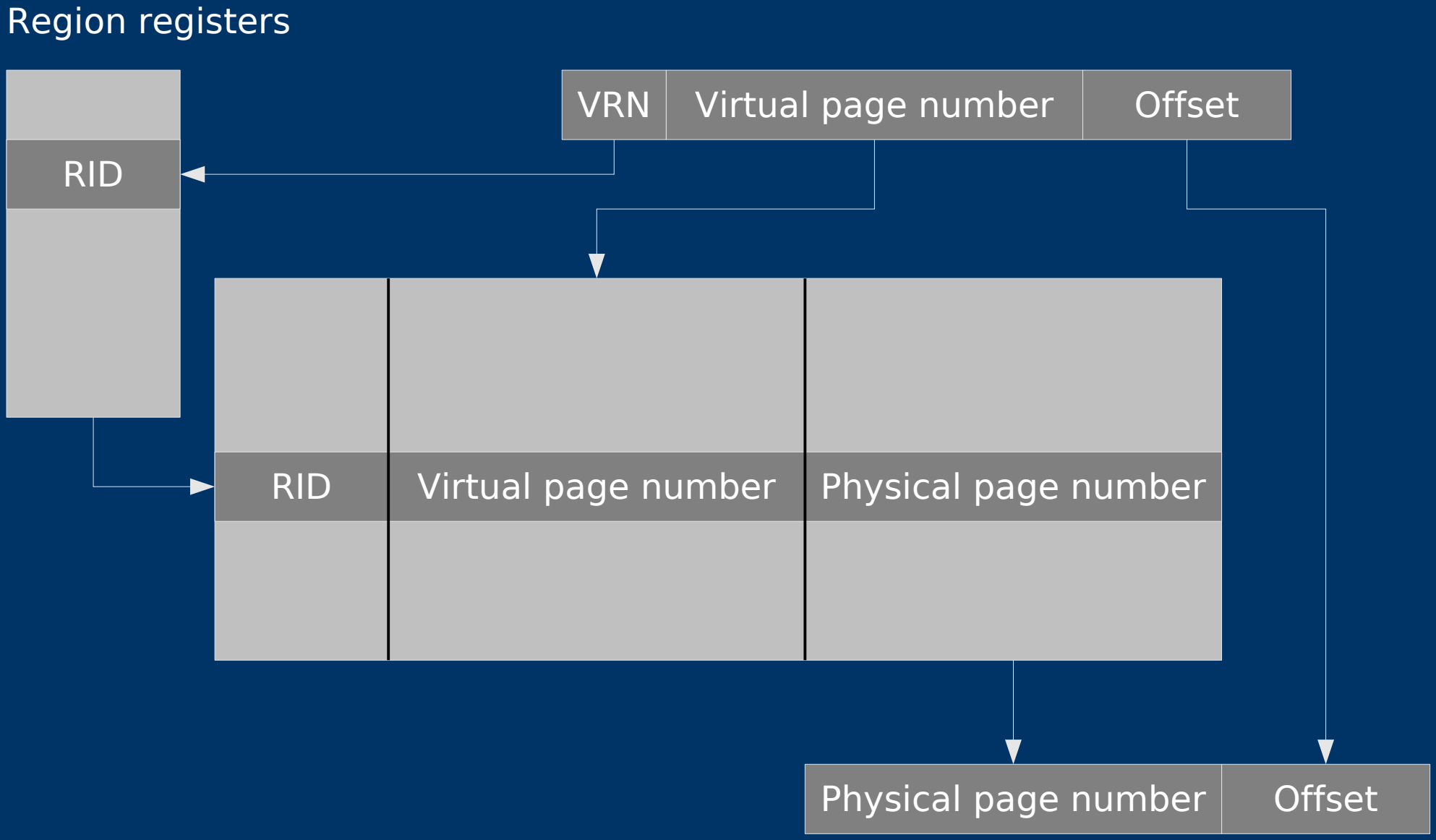
Virtual memory



Translation Lookaside Buffer

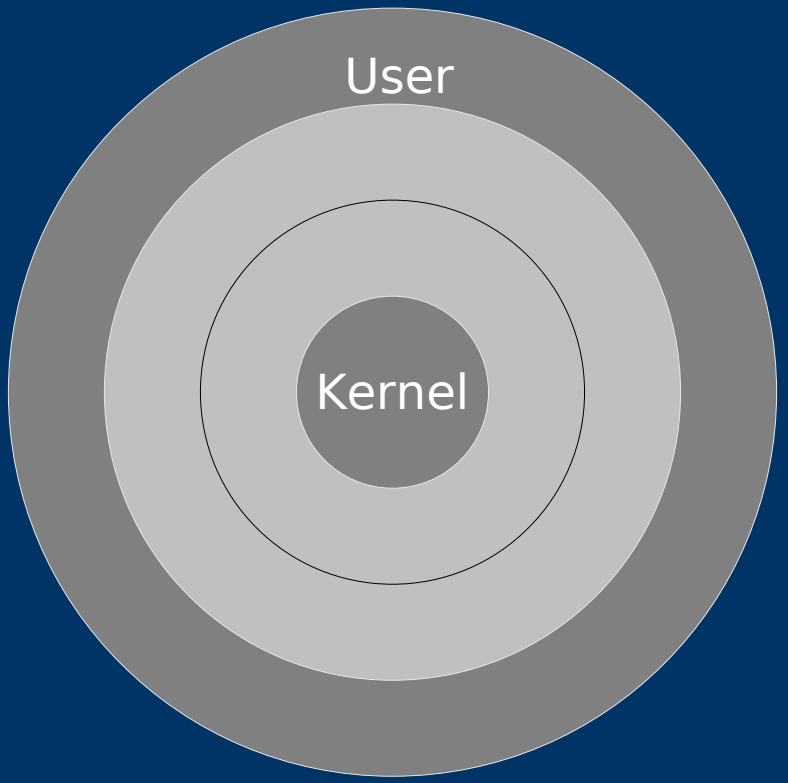
- Accelerates the translation from virtual address to physical address
- Implemented on-die
  - very low latency
- Caches only a subset of mappings
  - On x86 the scope of the whole buffer is only valid for one process: Expensive flush necessary each process switch
  - IA-64 tags entries to make each entry valid process-wise

Translation Lookaside Buffer



Protection rings

- Protect kernel from faulty or malicious code
- Protection of
  - Privileged state
  - Privileged instructions
  - Privileged pages or segments



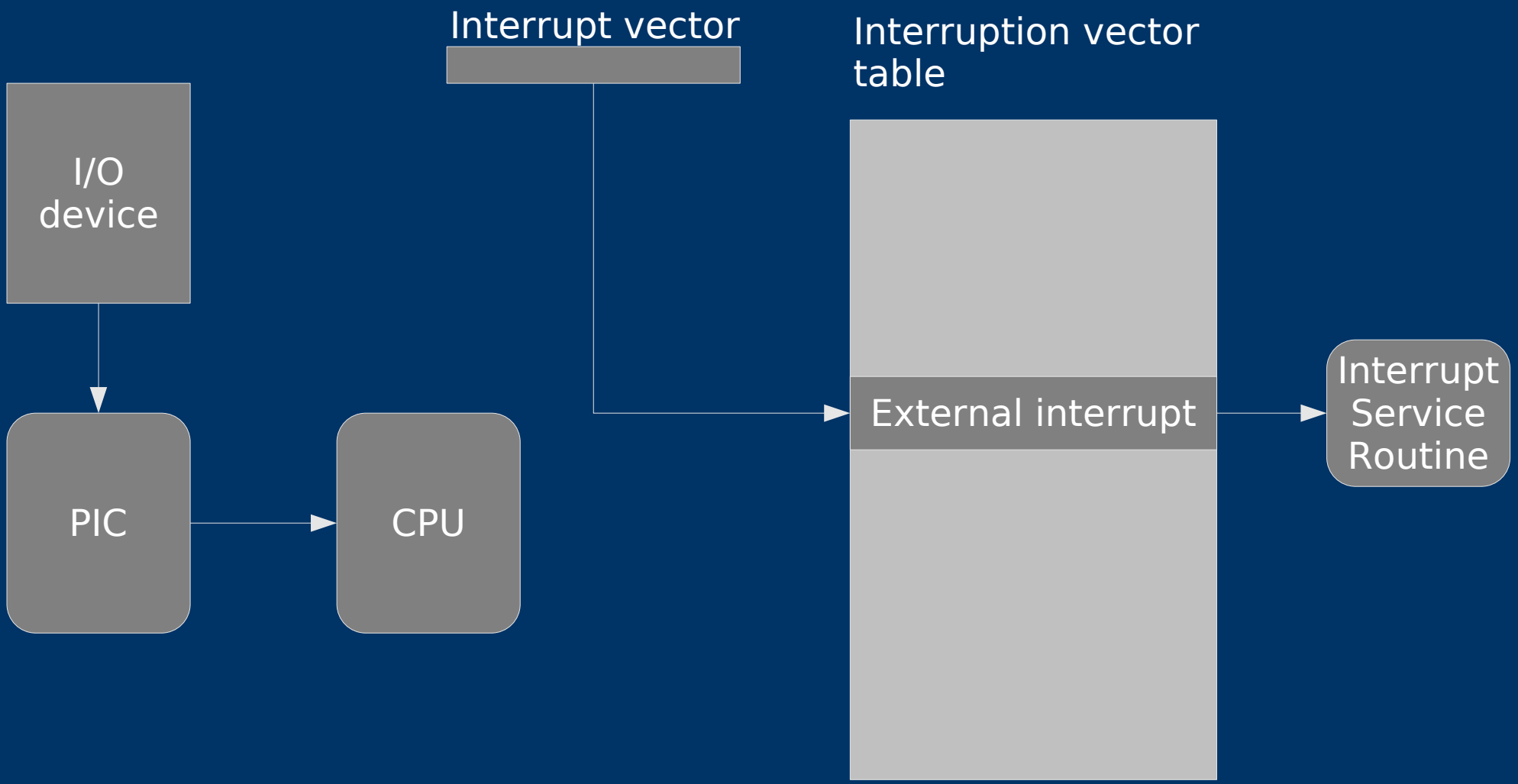
Kernel entry

- From ring 3 to ring 0 – From User space to Kernel space
- System calls
- Interrupt Service Routines
- Device access

Interrupts and exceptions

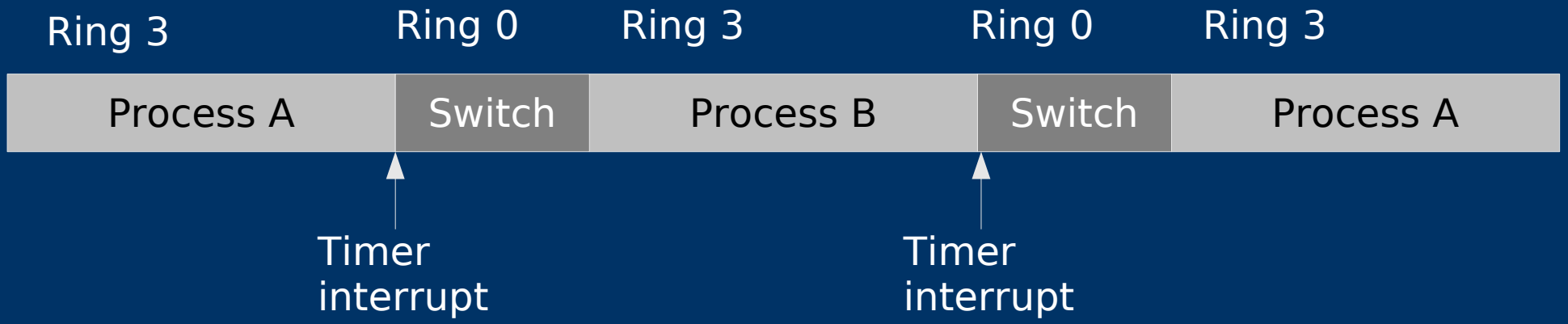
- Kernel entry
  - Exceptions
    - General protection fault
    - Segmentation fault
    - Page fault
    - Divide-by-zero
  - External interrupts
    - Keyboard
    - DMA finished
    - Packet on network
    - Timer

Interrupts and exceptions



Processes

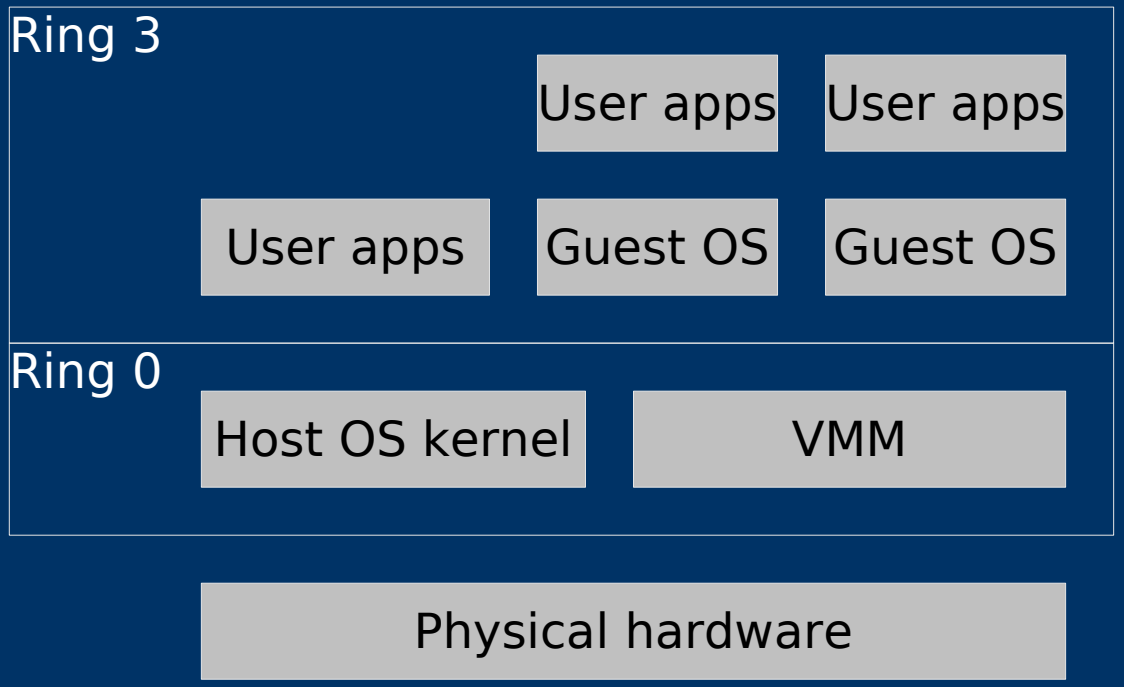
- Multitasking



Hardware Virtualisation

Hardware Virtualisation

- The Guest OS must think it is running on a real machine
- What happens if it is not run in ring 0?
- Need to intercept or remove some of the guest OS's operations

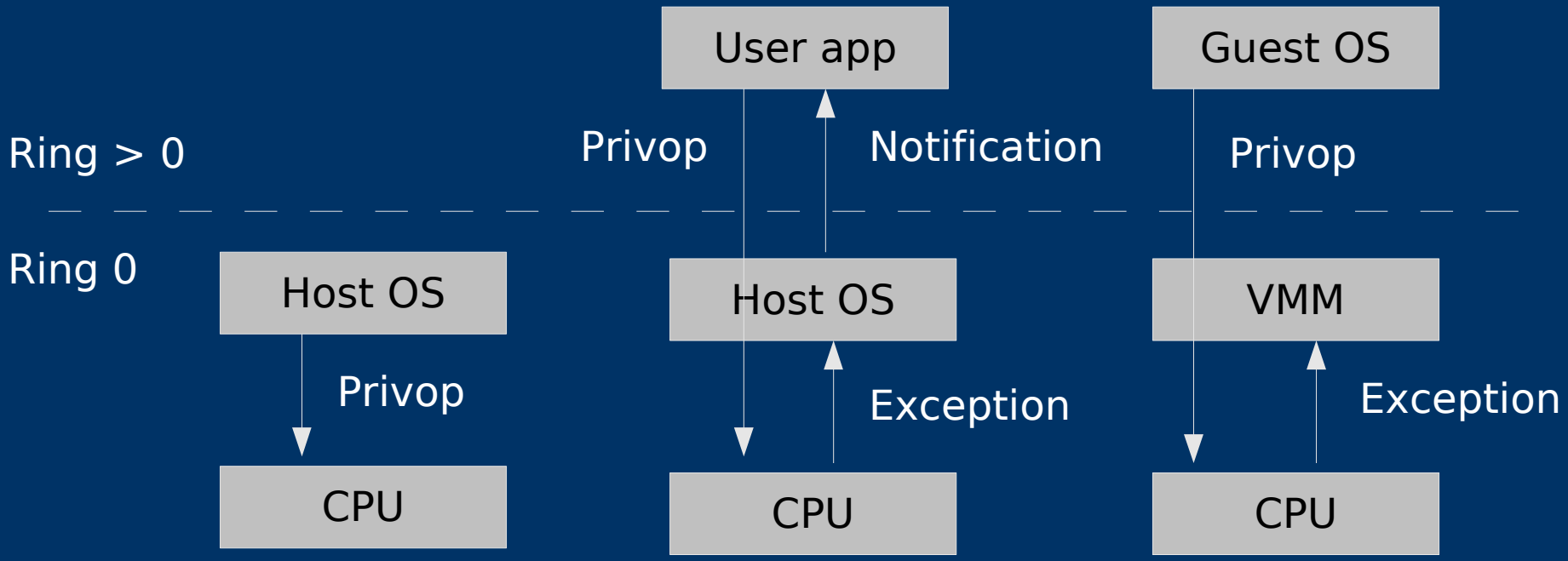


Hardware Virtualisation

- Three main approaches:
  - Interpretation (slow)
  - Binary patching or translation (faster)
    - Privileged operations
    - Privilege-sensitive operations
    - At runtime (VMWare) or compile-time (L4Ka Afterburning)
  - Source patching (Xen para-virtualisation) (fastest)

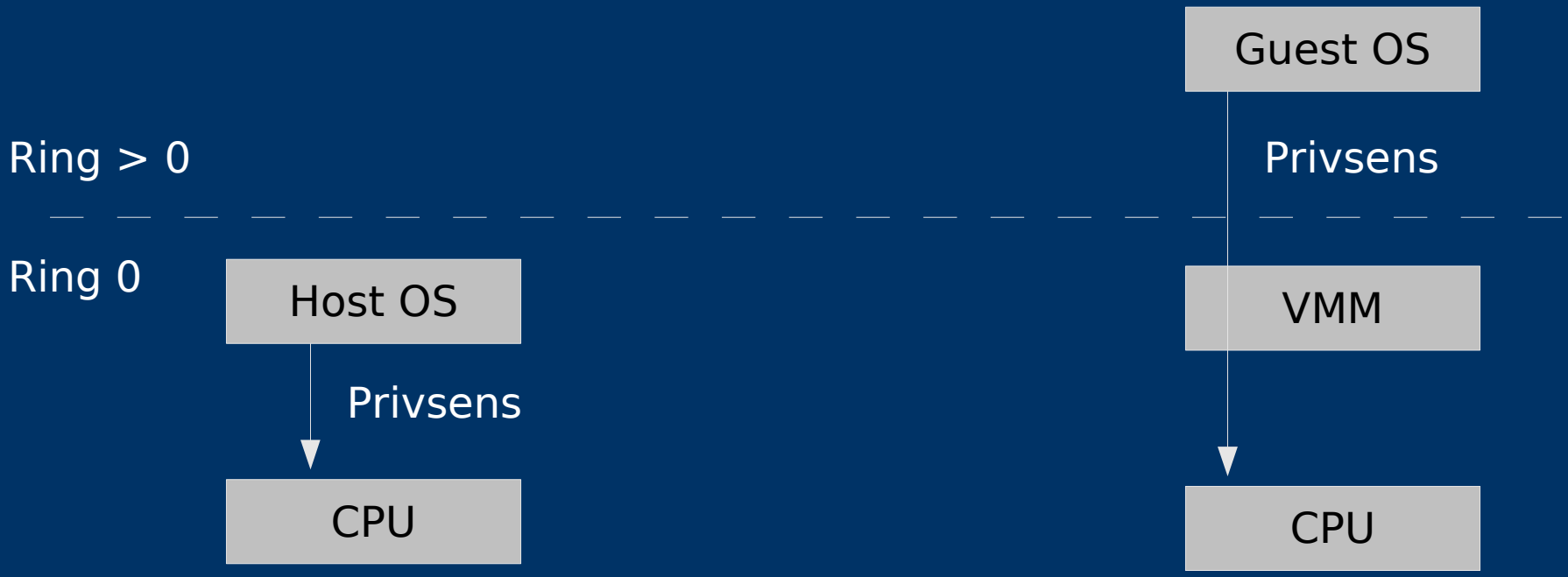
Privileged operations

- The guest OS must *think* that it is privileged



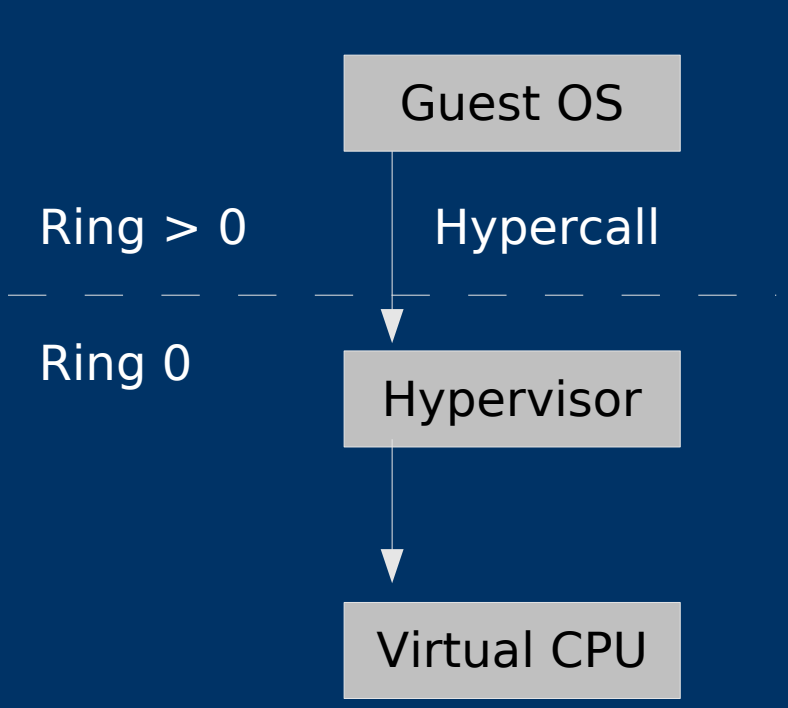
Privilege-sensitive operations

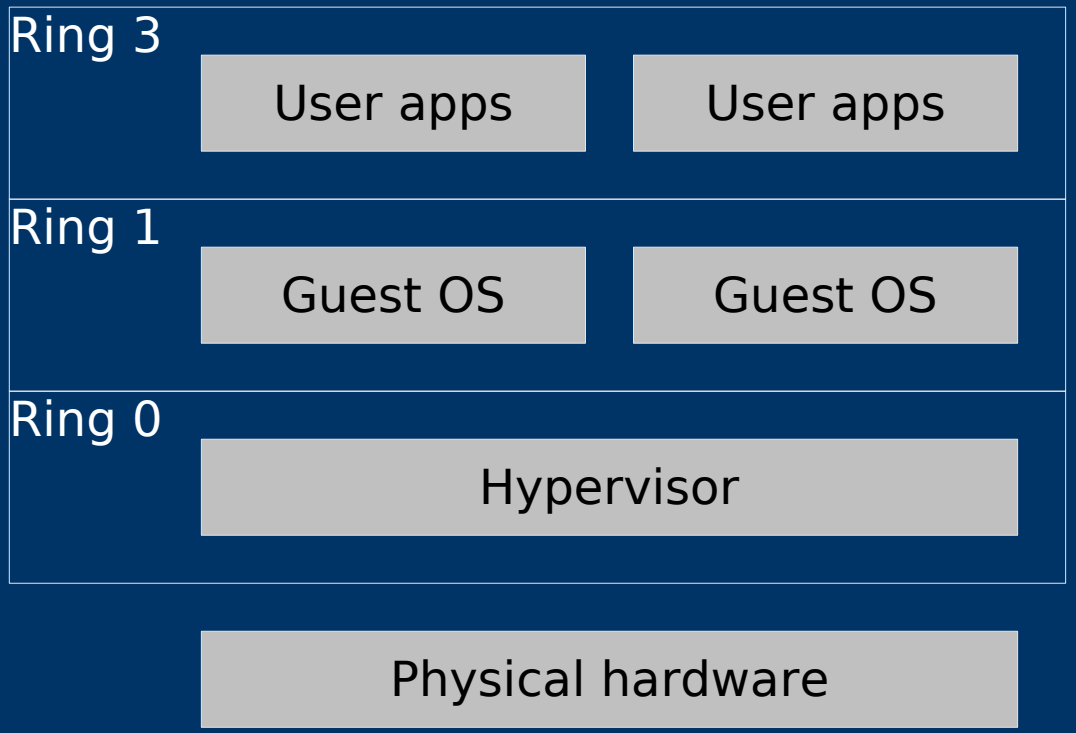
- Operations that are not protected, but
  - Access privileged state or
  - Whose results depend on CPL



Para-virtualisation

- Replace sensitive operations with calls to the Hypervisor - *hypercalls*





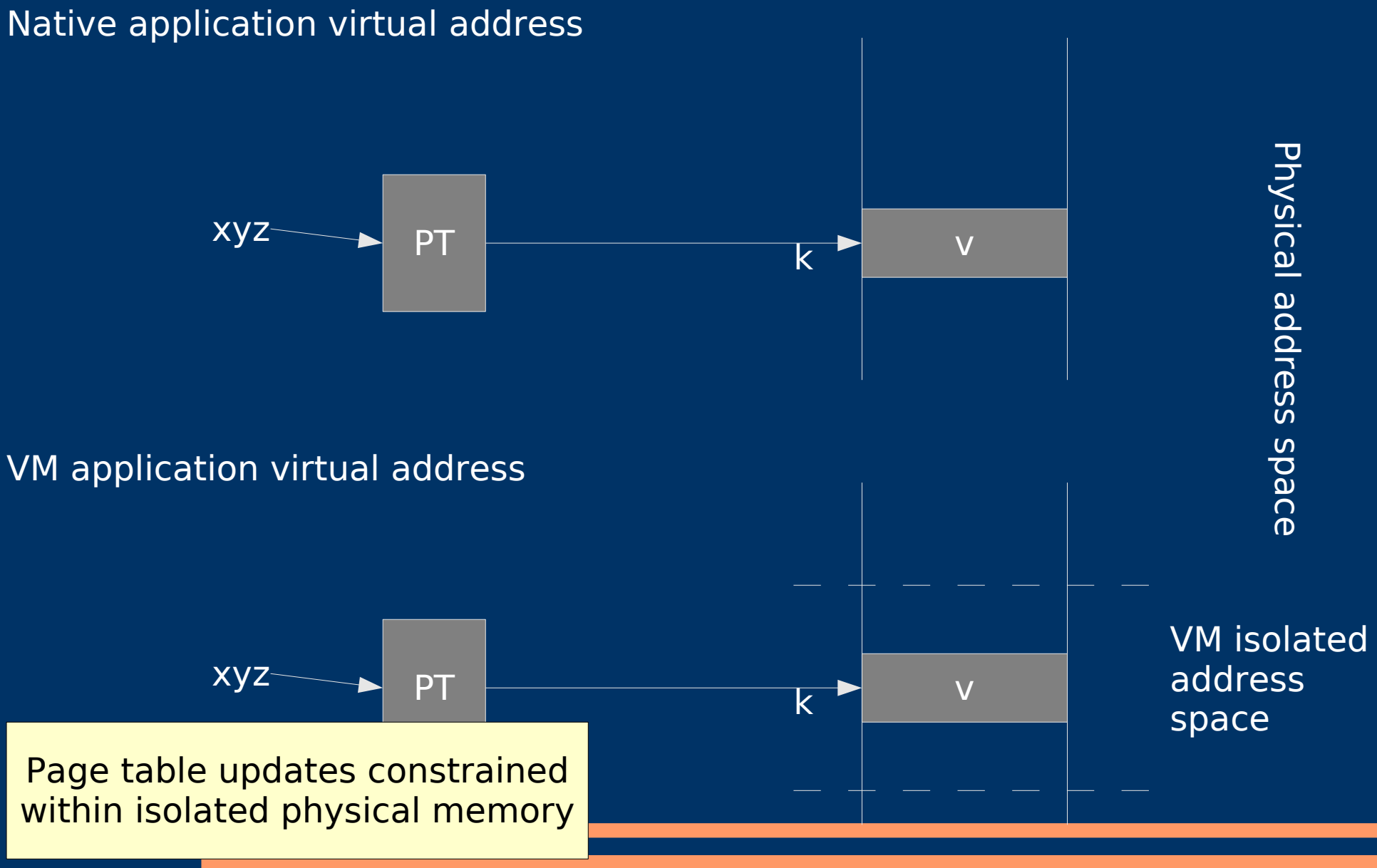
Xen memory management

- Page table updates through hypercalls
  - VMs use the same page table structures as the native MMU
  - Pages are marked read-only by the VMM
    - Any write by VM will fault to VMM
    - Efficient read by VM
    - Costly exit to VMM on page fault
- Multiple updates bundled

Xen memory management

- x86
  - Direct mapping between physical and virtual memory space
- IA-64
  - Logically separated address spaces using RIDs
  - Physical memory space has its own RID

Xen memory management

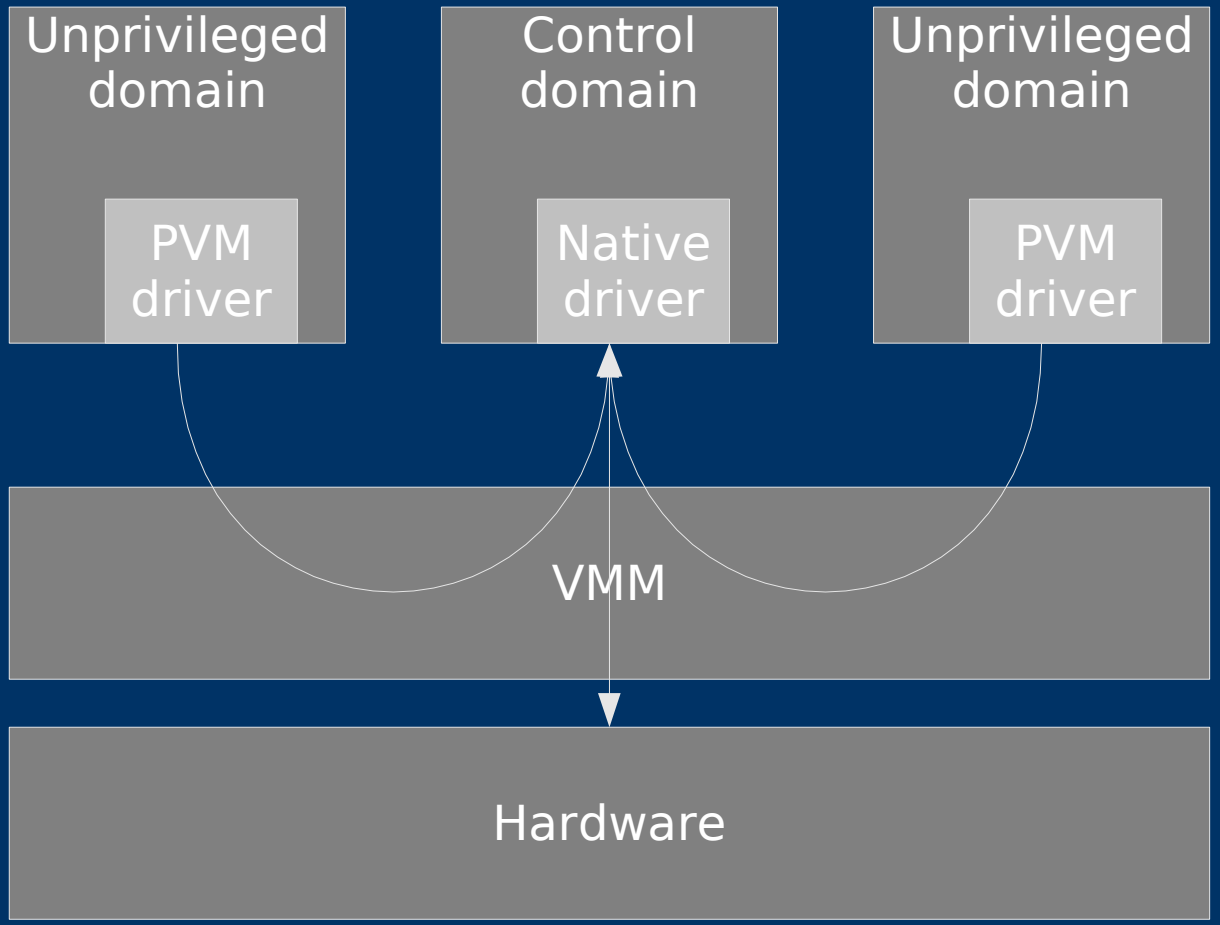


I/O virtualisation

- Protect I/O ranges of disk access, NICs, etc.
  - We don't want different VMs to write to the same device
  - Isolation dictates that a VM shouldn't be allowed to read another VM's volumes
- Solutions
  - Direct assignment: assign the whole device to a VM
  - Multiplexing: allows VMs to share the same device

Device multiplexing - Xen

- Context switch needed on each I/O operation

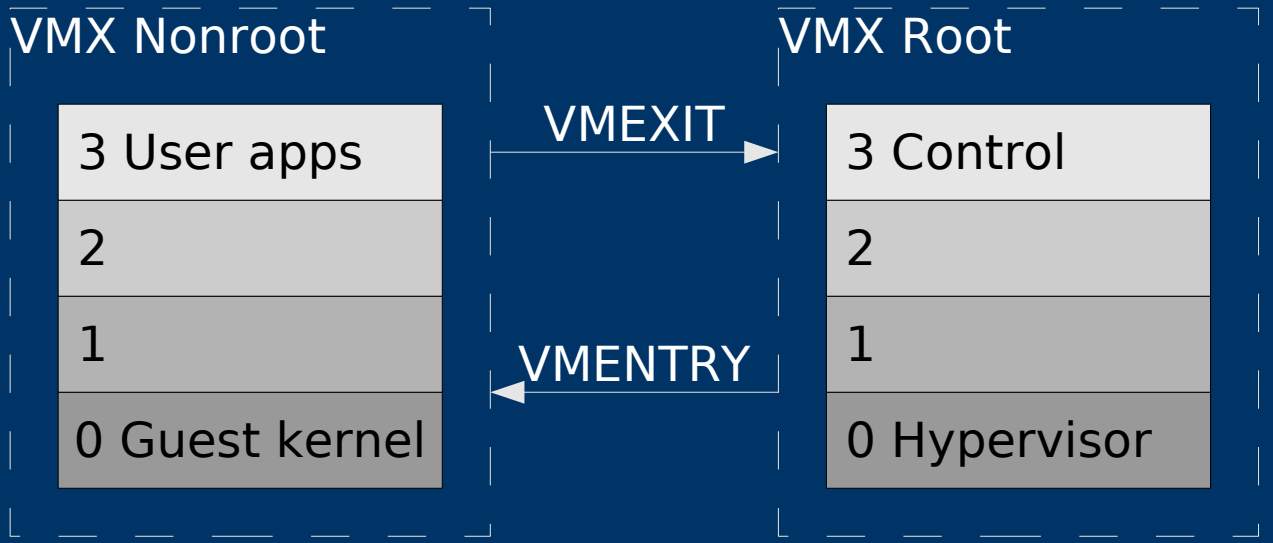


The Future

- Full hardware virtualisation without performance loss is not possible with conventional x86 architecture
- Extra facilities are needed in the hardware

Vanderpool (VT)

- X86: “VTx”
- IA-64: “VTi”
- Already mainstream

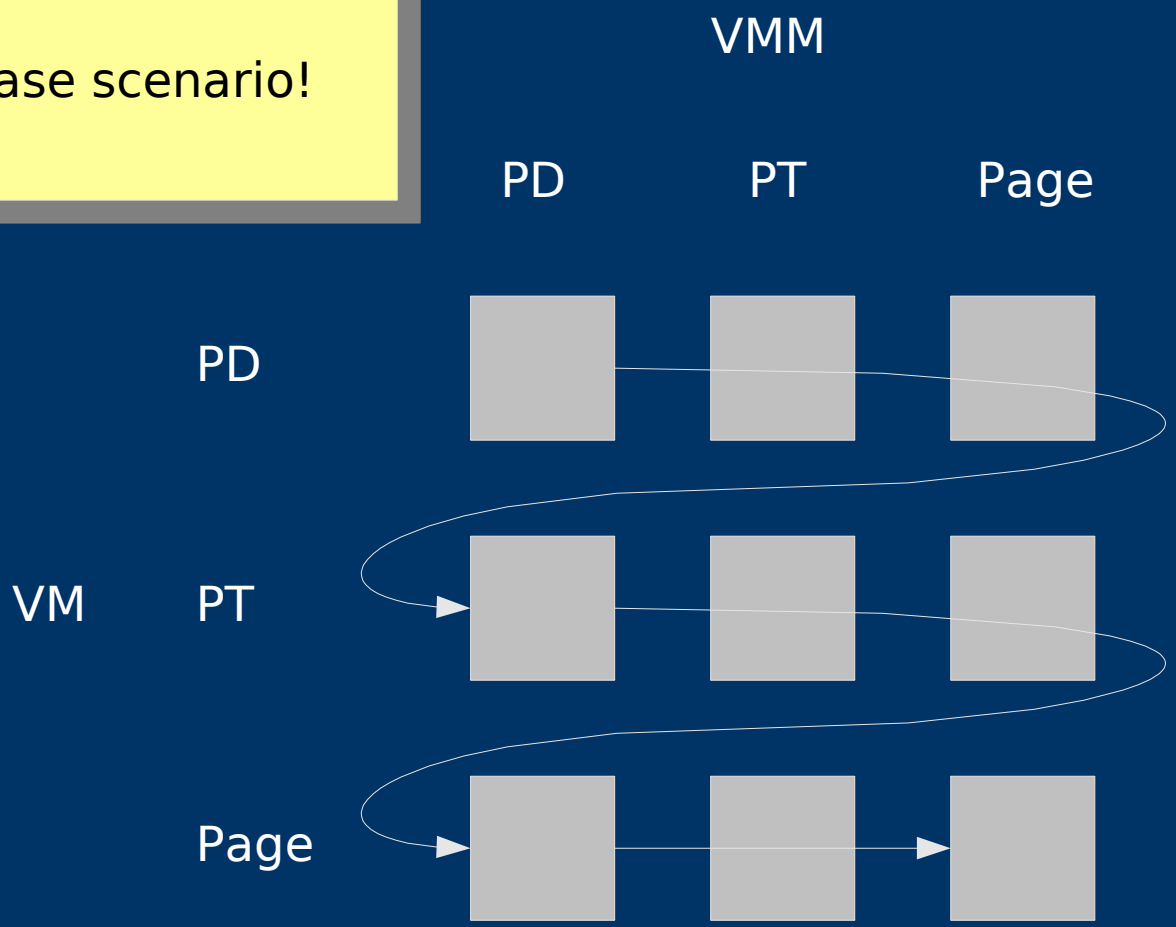


Extended Page Tables

- Aka “Nested Page Tables”
- A virtual address in a VM's address space resides in one of the VMM's pages!
- -> all of the VM's page table datastructures reside in the VMM's pages
- Eliminates VMEXIT, but:

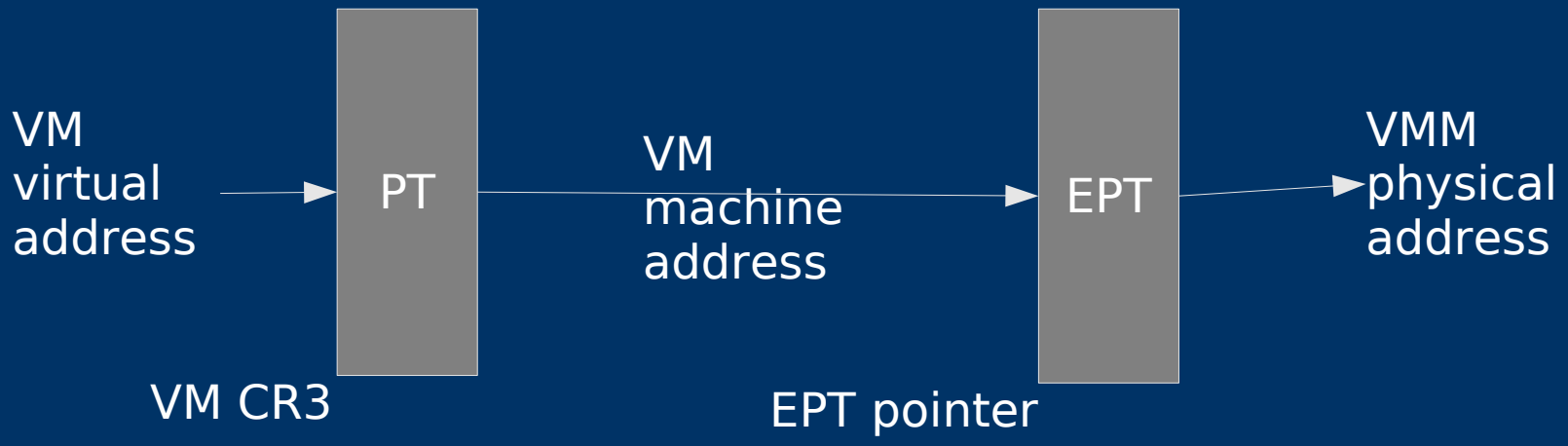
Extended Page Tables

Worst-case scenario!



Extended Page Tables

- VMX CR3 register points to VM's page directory
  - No VMEXIT needed on CR3 access or page fault



VT-d

- Device addresses protected by hardware
  - Allocated to VMs – protection domains
- DMA interrupts are assigned to the corresponding domain's protection domain
- The DMA's page table walk is assigned to the VM's page tables