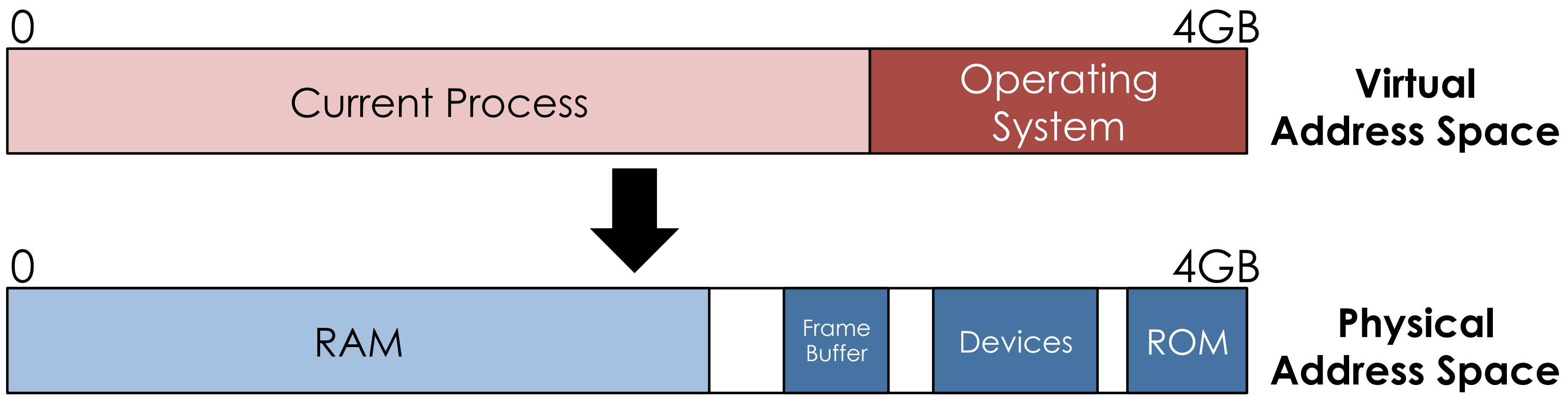
UNIVERSITÀ DI PISA

Memory Virtualization

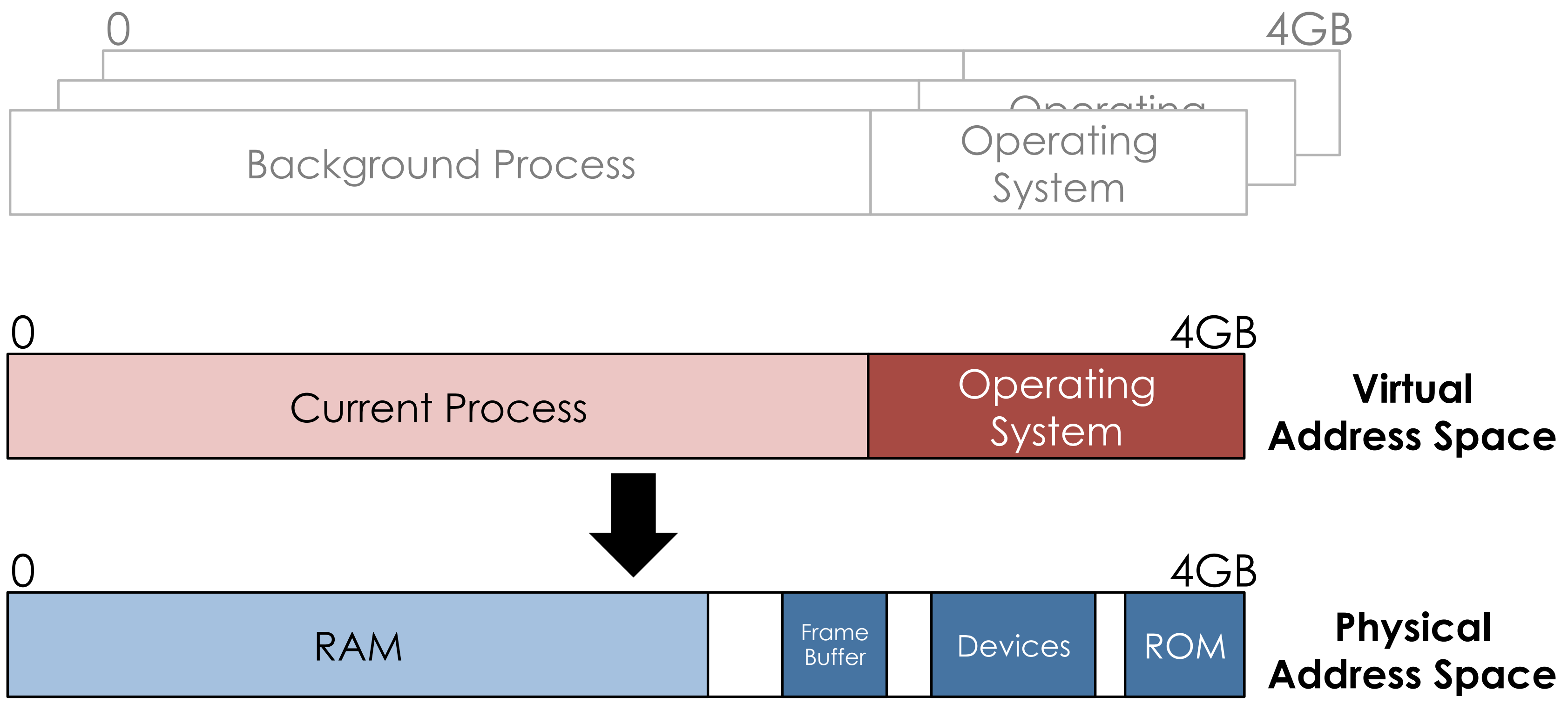
(based on Scott Devine slides by VMWare)

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Traditional Address Spaces



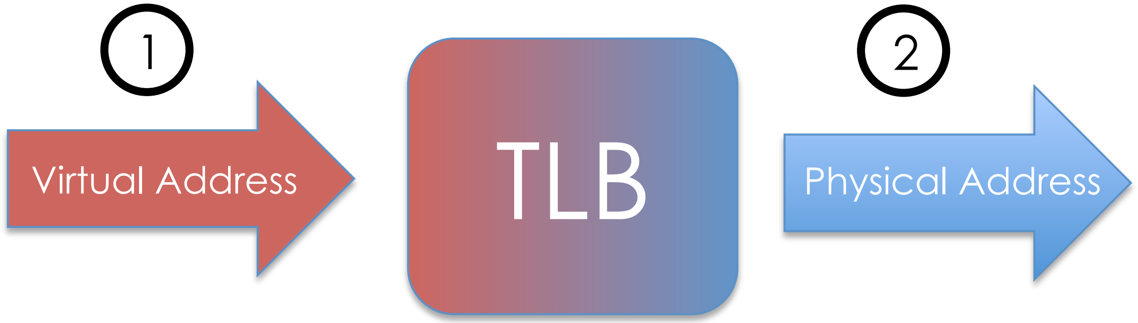
Traditional Address Spaces



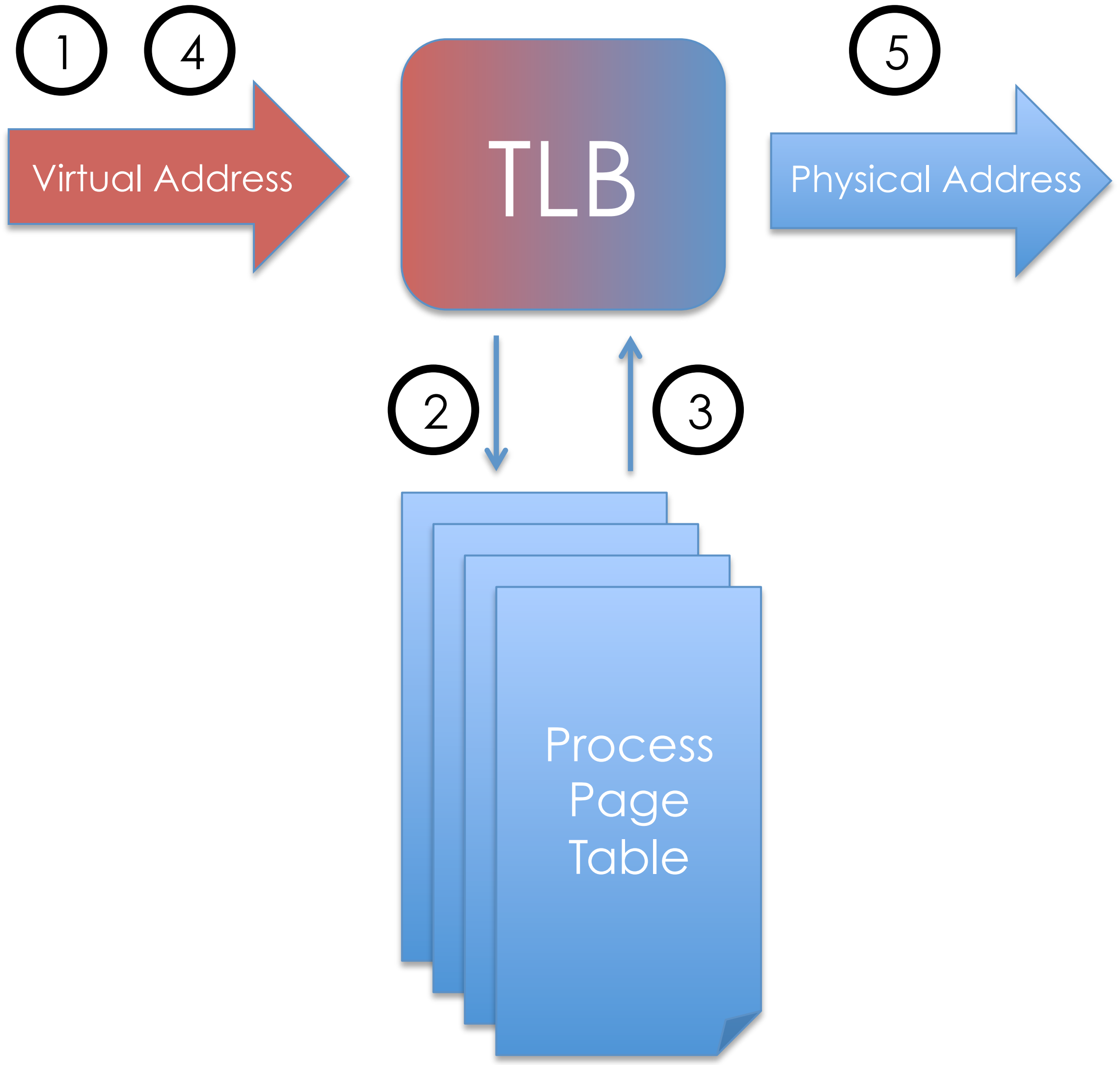
Memory Management Unit (MMU)

- Virtual Address to Physical Address Translation
  - Works in fixed-sized pages
  - Page Protection
- Translation Look-aside Buffer
  - TLB caches recently used Virtual to Physical mappings
- Control registers
  - Page Table location
  - Current ASID
  - Alignment checking

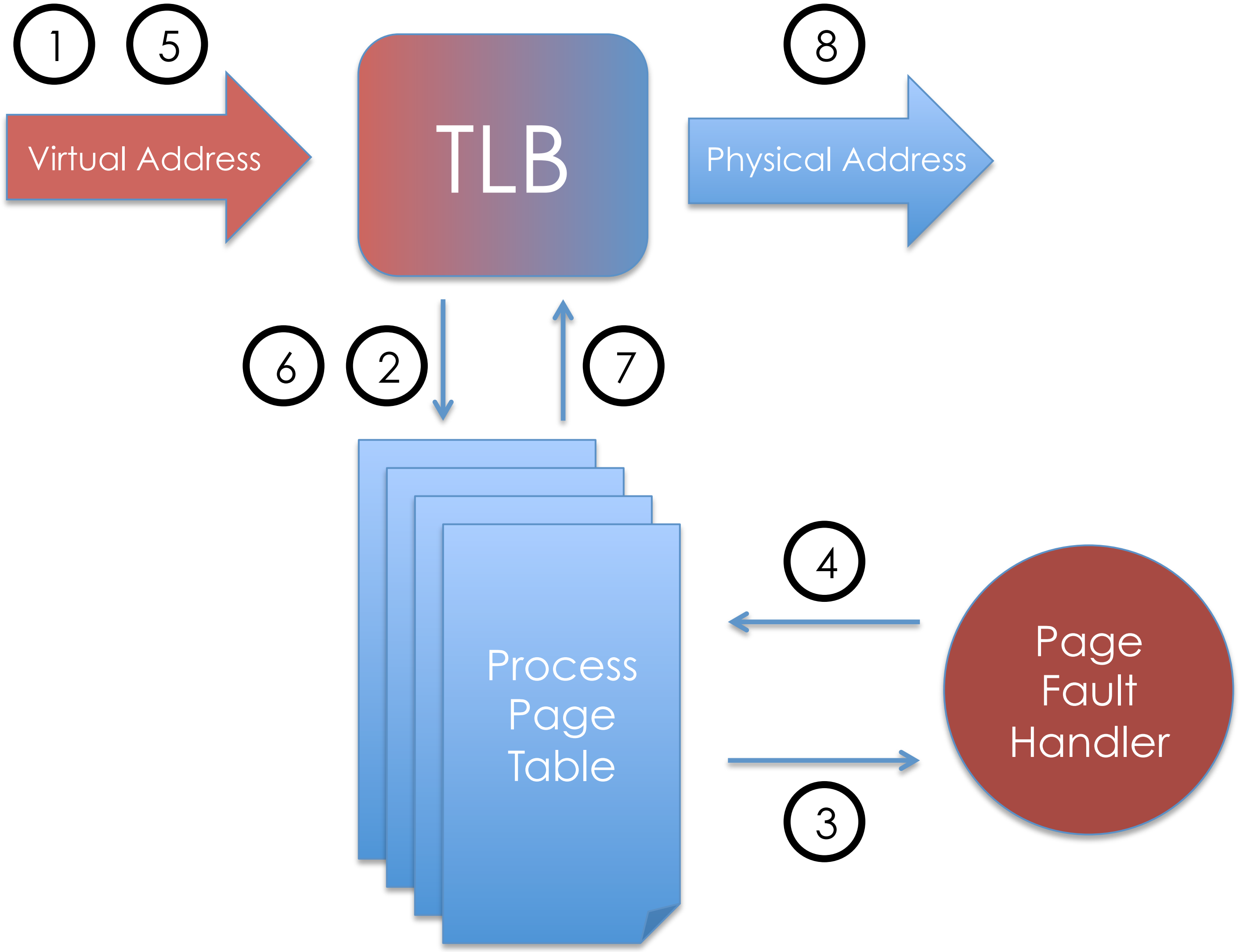
Traditional Address Translation (I)



Traditional Address Translation (II)

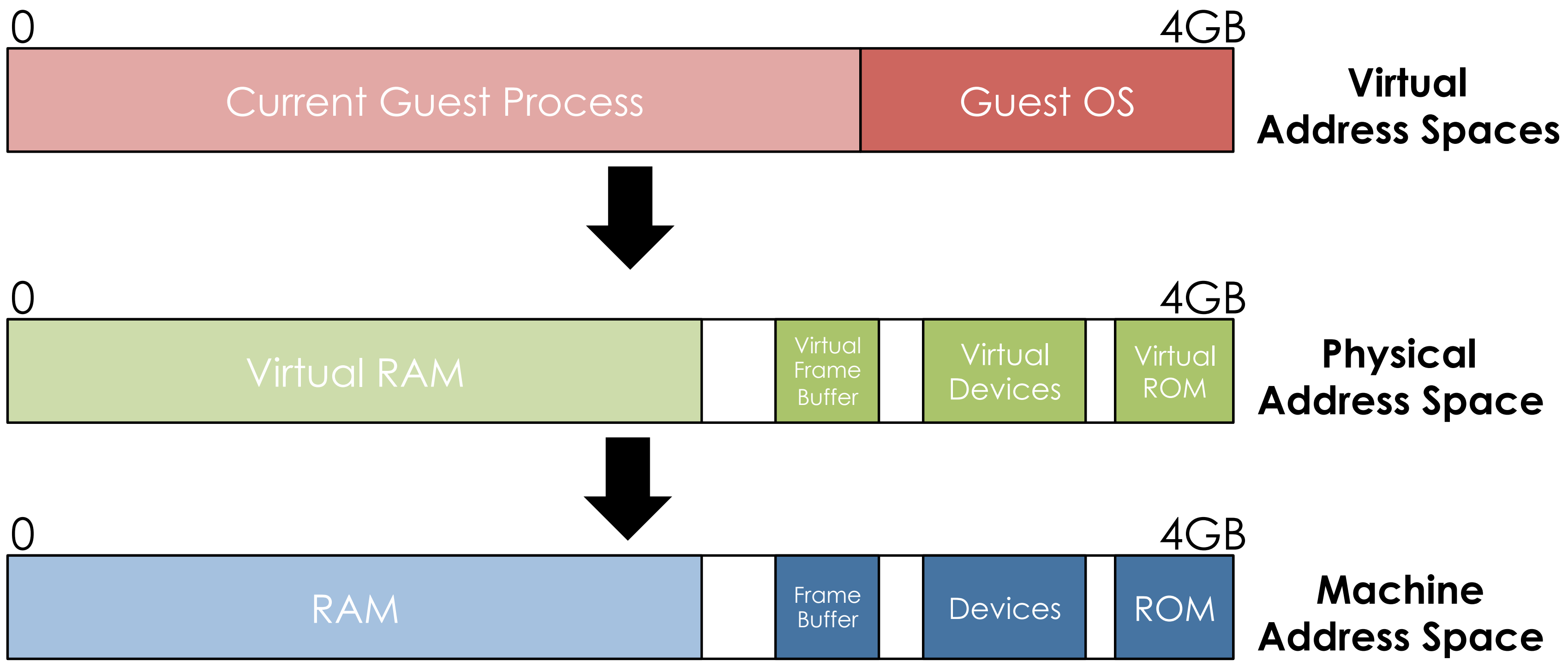


Traditional Address Translation (III)

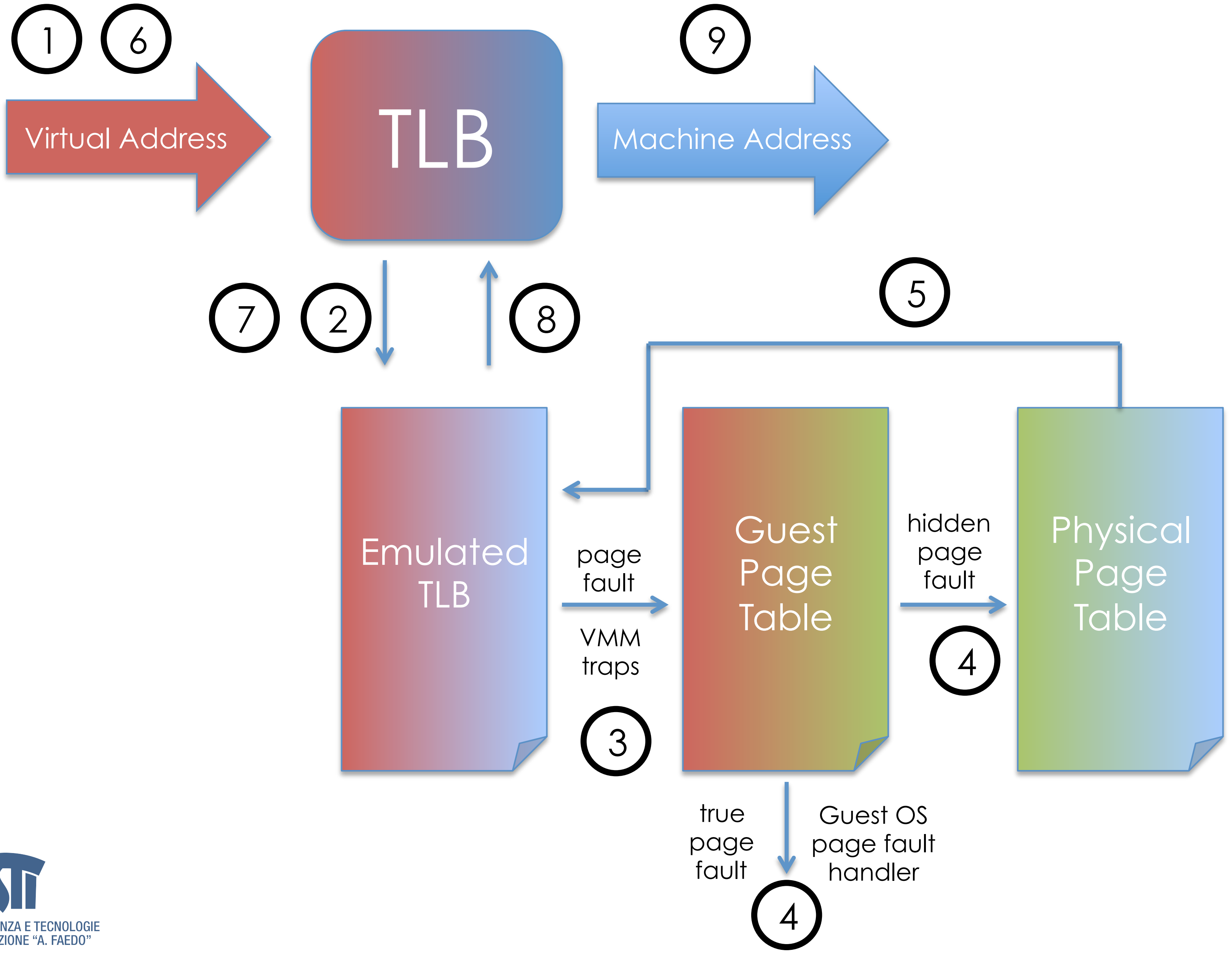


ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Virtualized Address Spaces



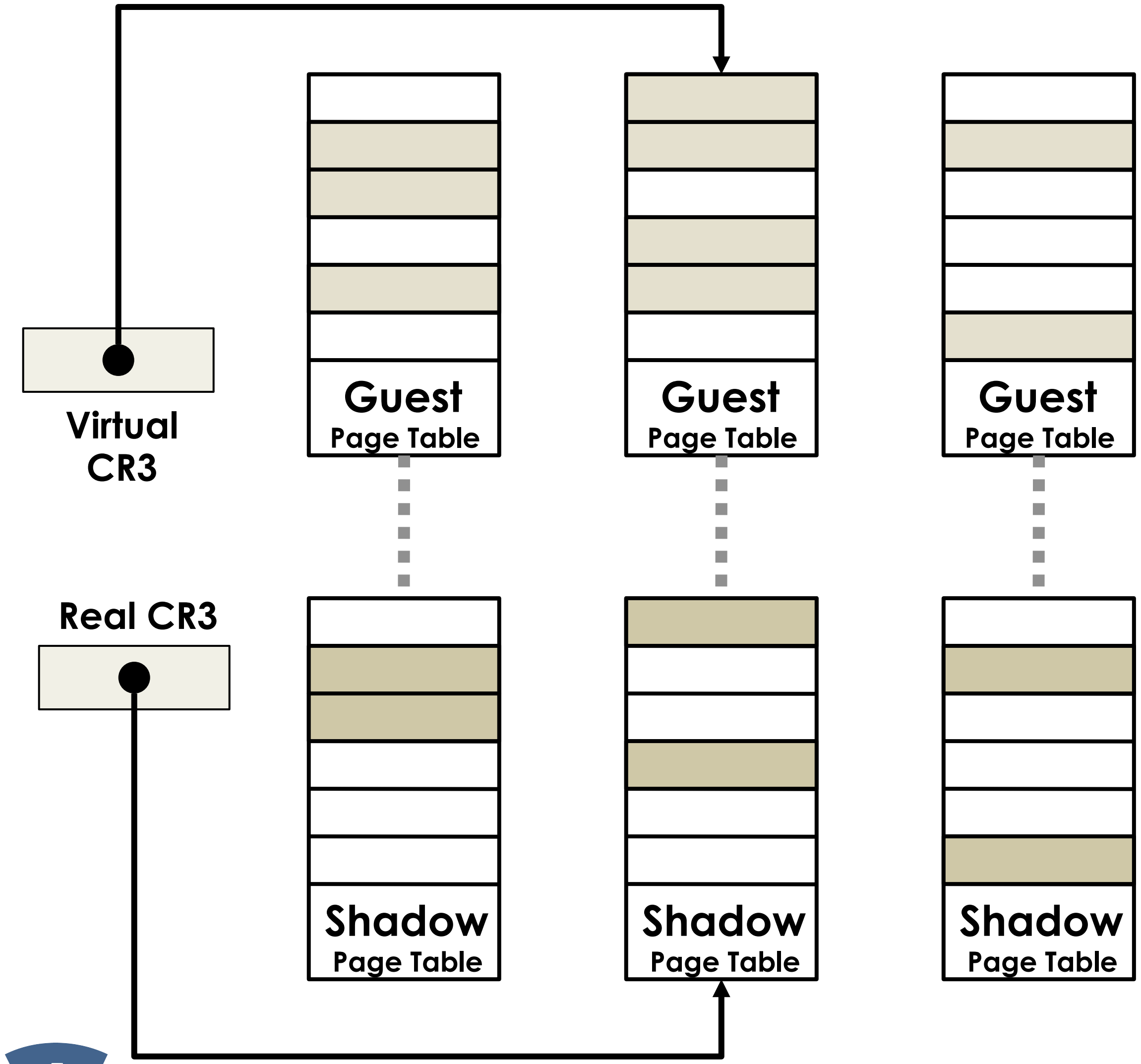
Virtualized Address Translation: TLB Emulation



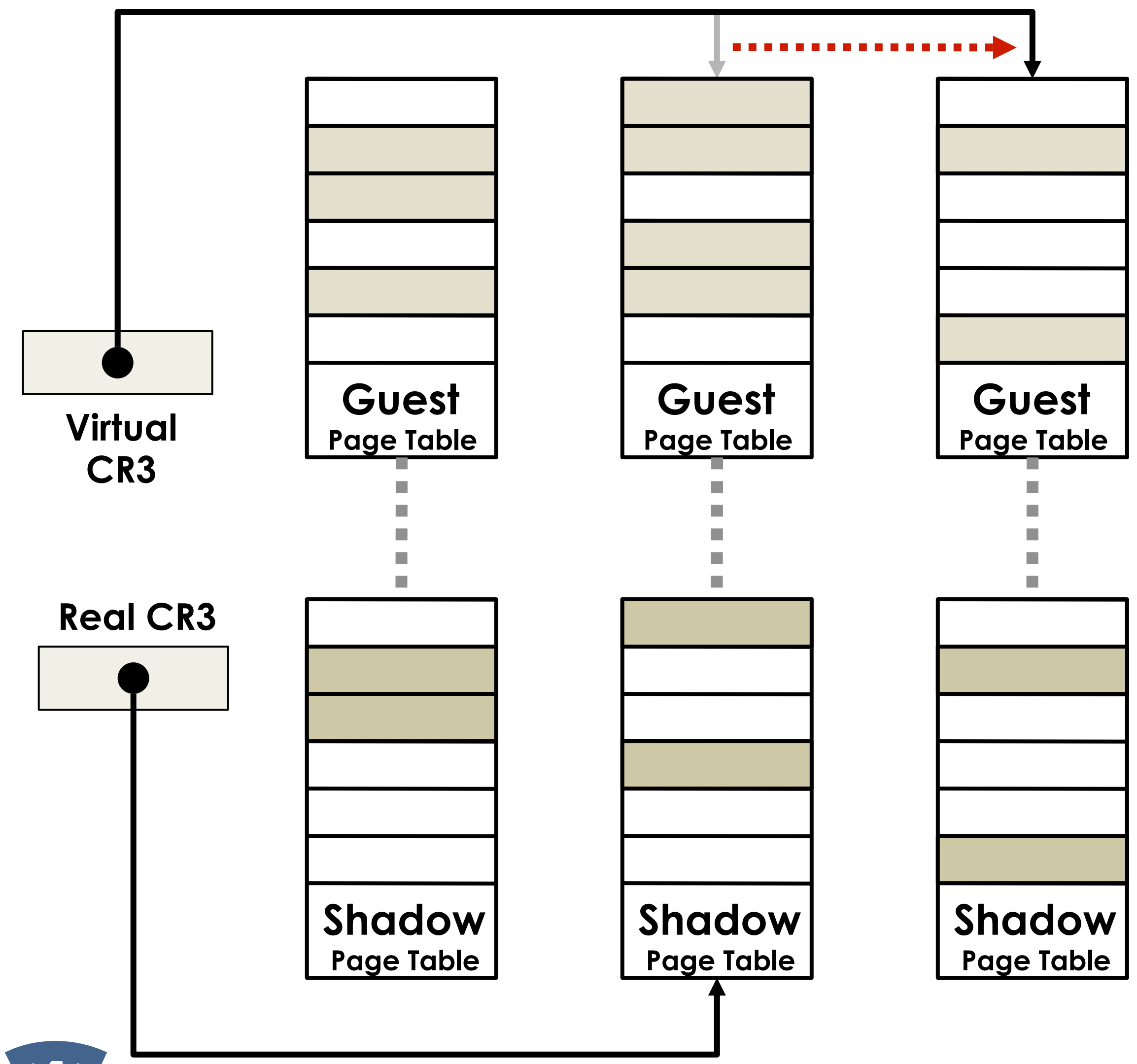
Issues

- Guest page table consistency
  - What happens when the guest changes an entry in its page table?
  - What happens when the guest switches to a new page table on a process context switch?
- Performance
  - Guest context switches flush entire software TLB
  - Minimize hidden page faults
  - Aggressive flushing will cause flood of hpfs every guest context switch
  - Keep one shadow page table per guest process

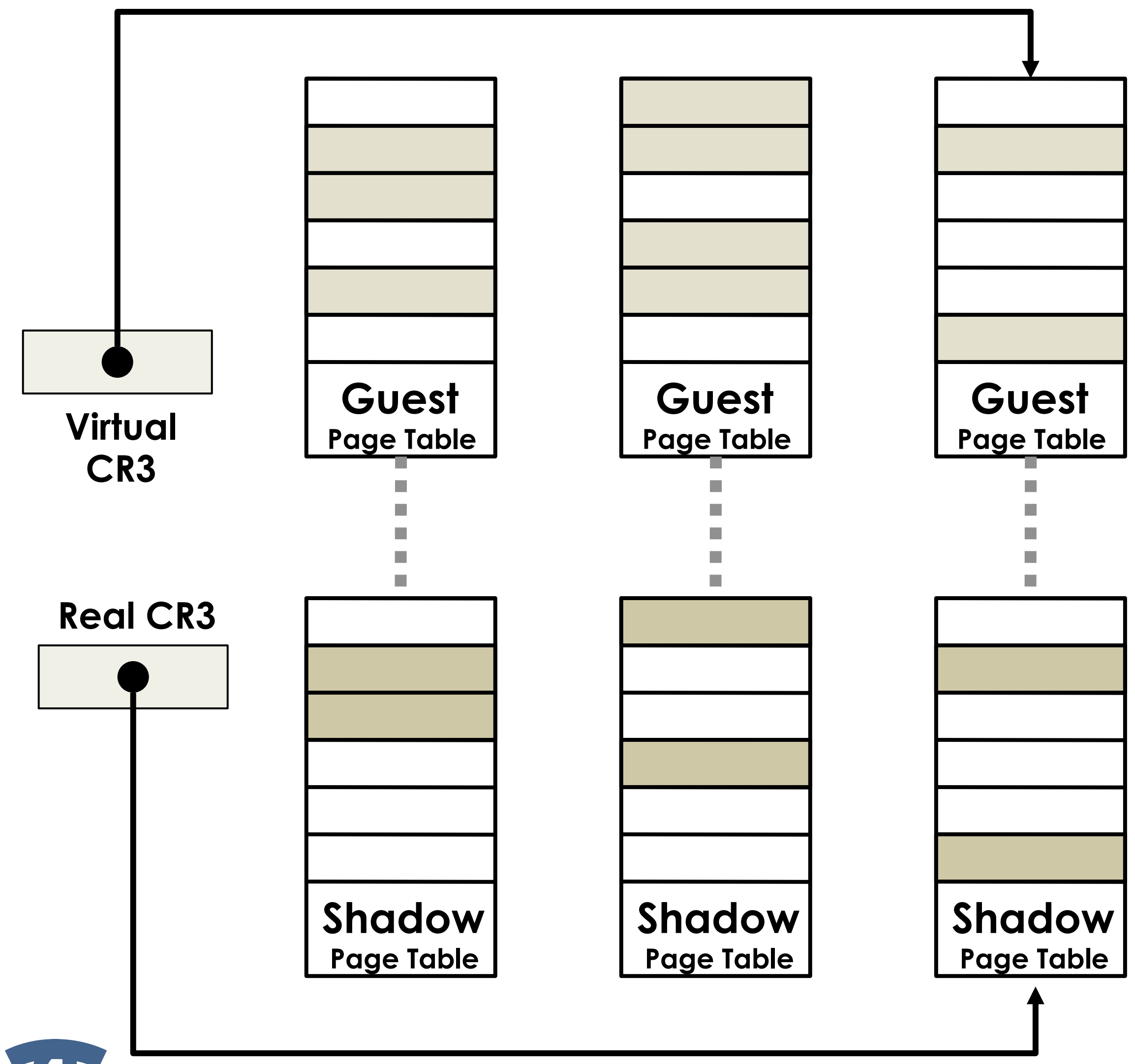
Virtualized Address Translation: Shadow Page Tables



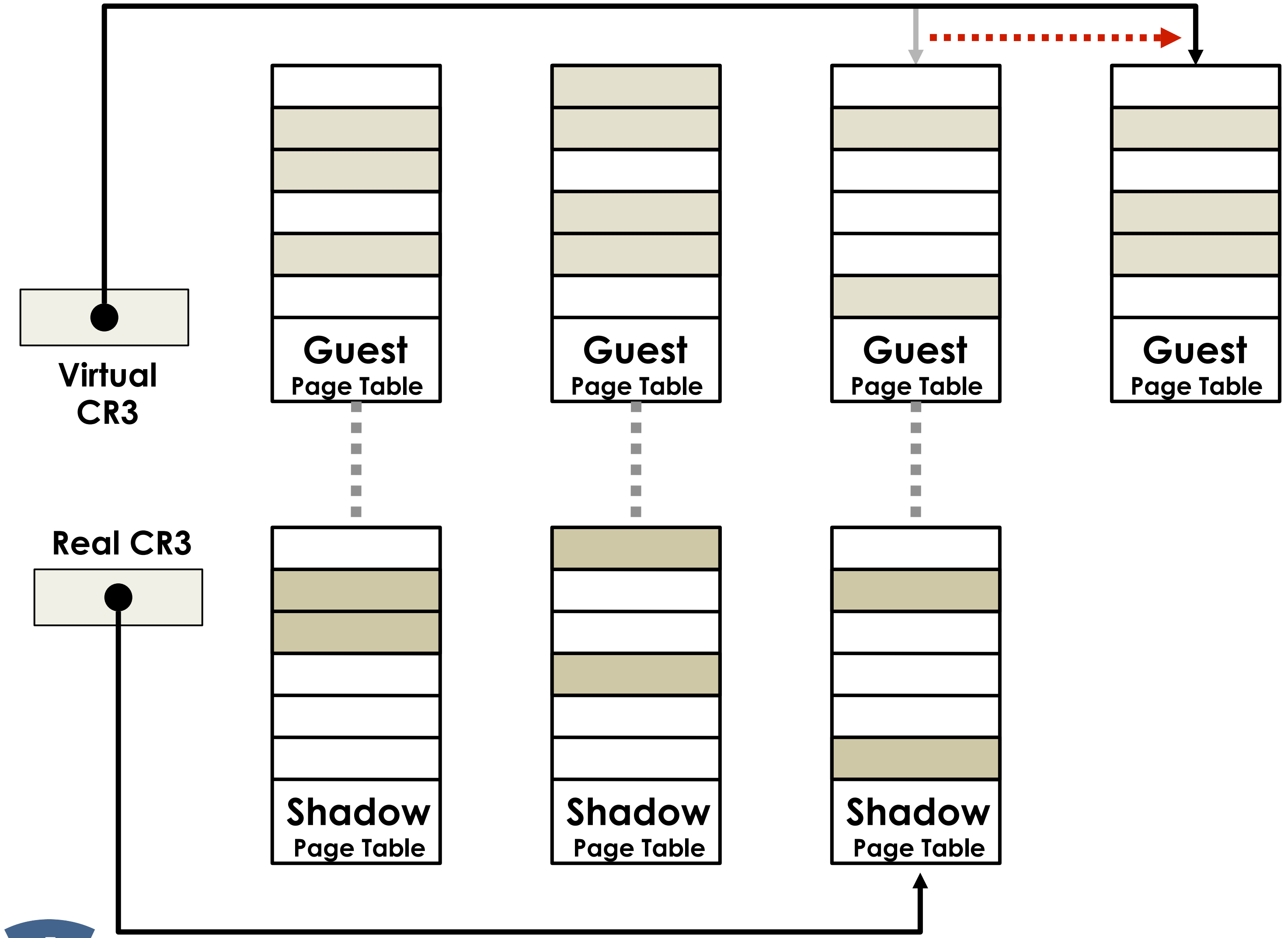
Guest Write to CR3



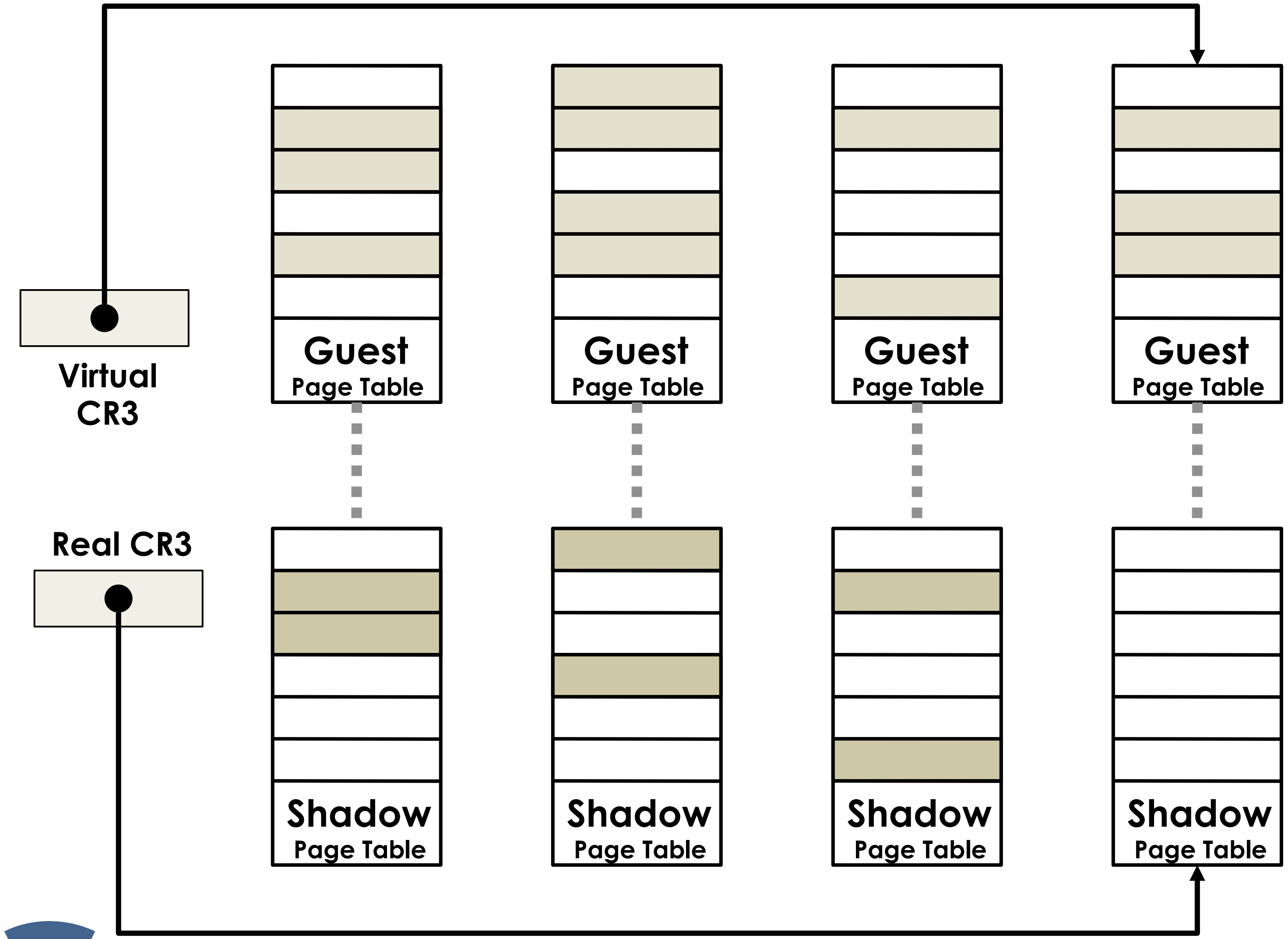
Guest Write to CR3



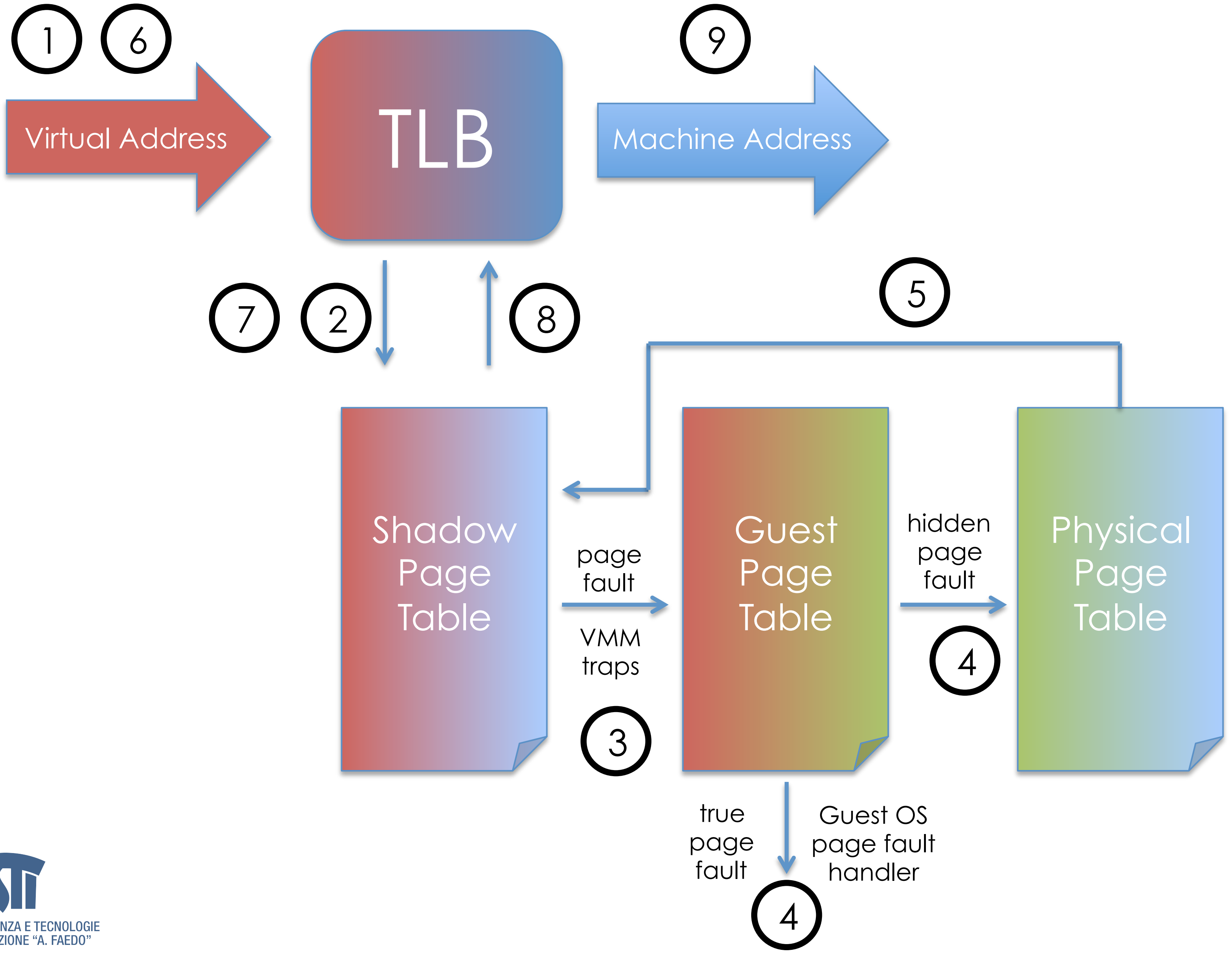
Undiscovered Guest Page Table



Undiscovered Guest Page Table



Virtualized Address Translation: Shadow Page Table



Issues

- Benefits
  - Handle page faults in same way as Emulated TLB
  - Fast guest context switching
- Page Table Consistency
  - Guest may not need invalidate TLB on writes to off-line page tables
  - Need to trace writes to shadow page tables to invalidate entries
- Memory Bloat
  - Caching guest page tables takes memory
  - Need to determine when guest has reused page tables

Hardware-assisted Virtualization: Nested Page Tables

- Nested paging uses an additional **nested page table** (nPT) to translate guest physical addresses to system physical addresses

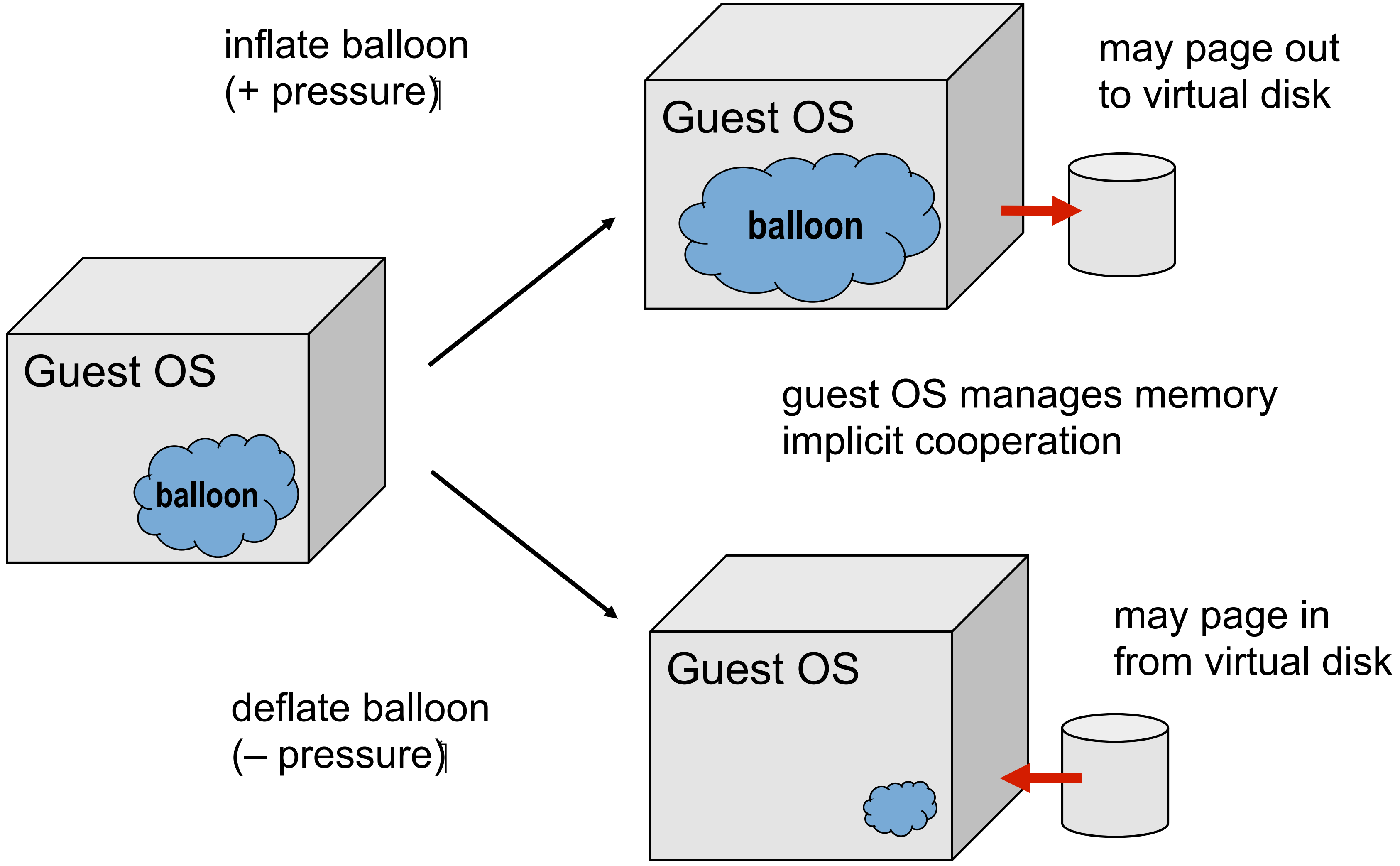
- The gPT maps guest linear addresses to guest physical addresses. Nested page tables (nPT) map guest physical addresses to system physical addresses.

- Guest and nested page tables are set up by the guest and hypervisor respectively. When a guest attempts to reference memory using a linear address and nested paging is enabled, the page walker performs a 2-dimensional walk using the gPT and nPT to translate the guest linear address to system physical address.

- Nested paging removes the overheads associated with shadow paging. Unlike shadow paging, once the nested pages are populated, the hypervisor does not need to intercept and emulate guest's modification of gPT.

- However because nested paging introduces an additional level of translation, the TLB miss cost could be larger.

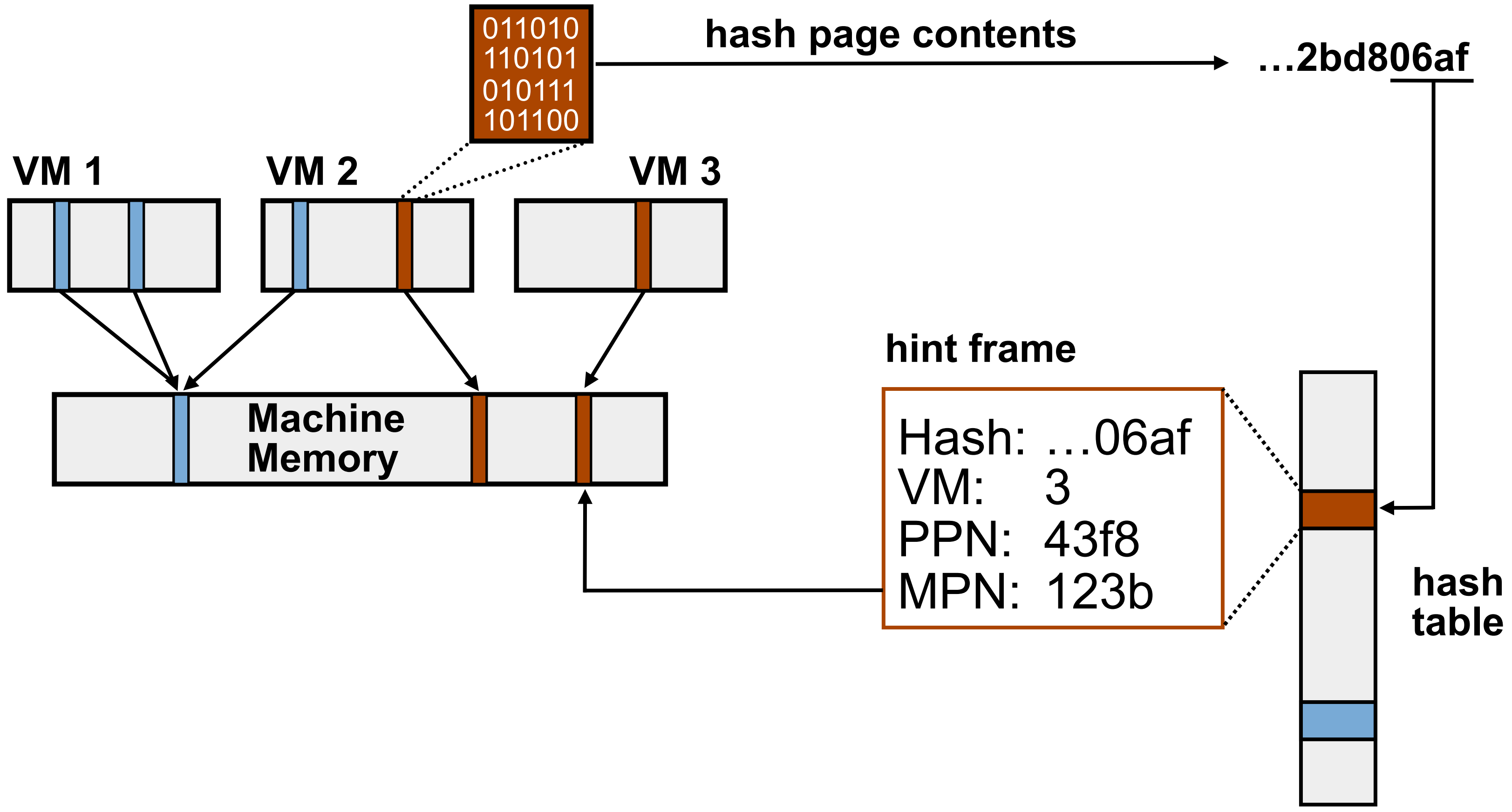
Reclaiming Memory: ballooning



Page Sharing

- Motivation
  - Multiple VMs running same OS, apps
  - Deduplicate redundant copies of code, data, zeros
- Transparent page sharing
  - Map multiple PPNs to single MPN copy-on-write
  - Pioneered by Disco [Bugnion et al. SOSP '97], but required guest OS hooks
- VMware content-based sharing
  - General-purpose, no guest OS changes
  - Background activity saves memory over time

Page Sharing: Scan Candidate PPN



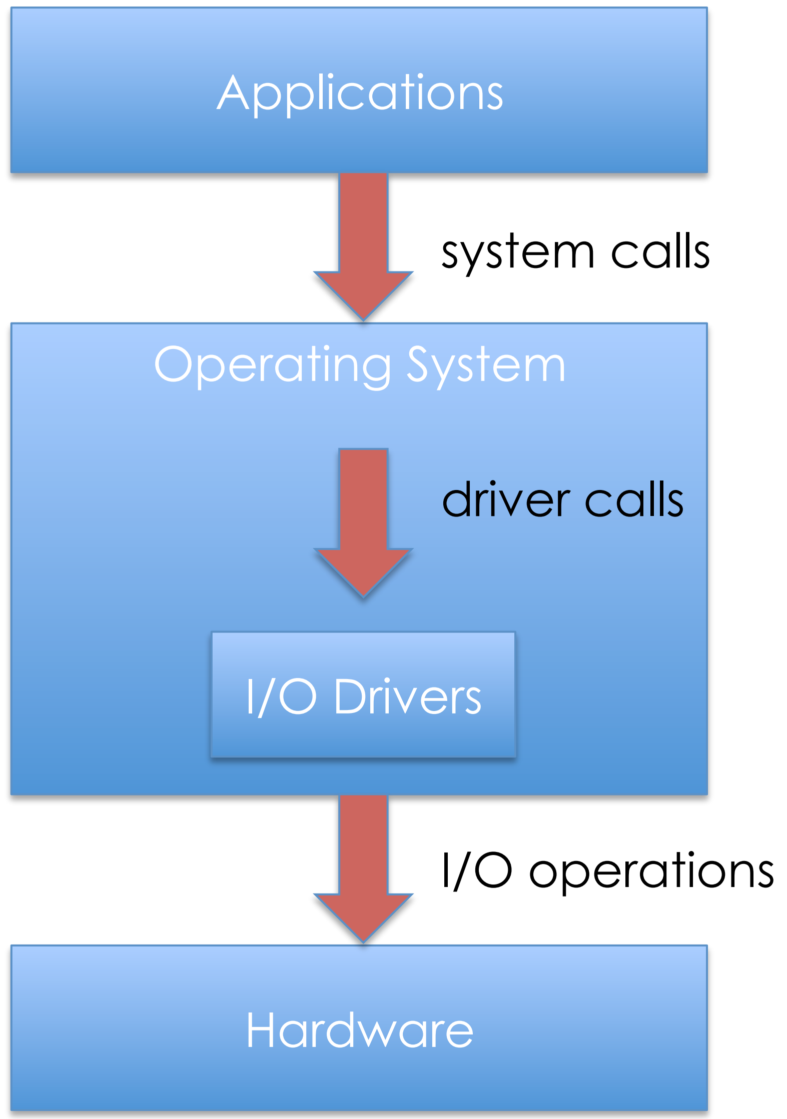
I/O Virtualization

Type of devices

- Dedicated Devices
  - Monitor, keyboard, mouse
  - No virtualization required, but VMM routing because guest OS runs in user mode
  - Interrupt handled by VM on activation by VMM
- Partitioned Devices
  - Disks
  - VMM maintains a map of parameters and re-issues the requests to physical devices
- Shared Devices
  - Network adapter
  - VMM translates through a virtual device drivers
- Spooled Devices
  - Printer
- Nonexistent Devices
  - Comm network

ISTITUTO DI SCIENZA E TECNOLOGIE
DELL'INFORMAZIONE "A. FAEDO"

Performing I/O



Virtualizing I/O

- I/O Operations level
  - I/O runs in privileged mode
  - Trap in user mode
  - Difficult to reverse engineer a complete I/O action
- Device drivers level
  - Needs virtual device drivers
  - VMM intercepts calls to virtual device drivers
  - Must know guest OS device driver implementation
  - Real drivers needed for native VMMs
- System calls level
  - Most efficient
  - Must know guest OS ABI to I/O and rewrite it taking care of emulation of everything else not directly related to I/O.