La virtualizzazione

Corso di Sistemi Operativi

Proff. Cotroneo Domenico
Cinque Marcello
Natella Roberto

Indice e riferimenti

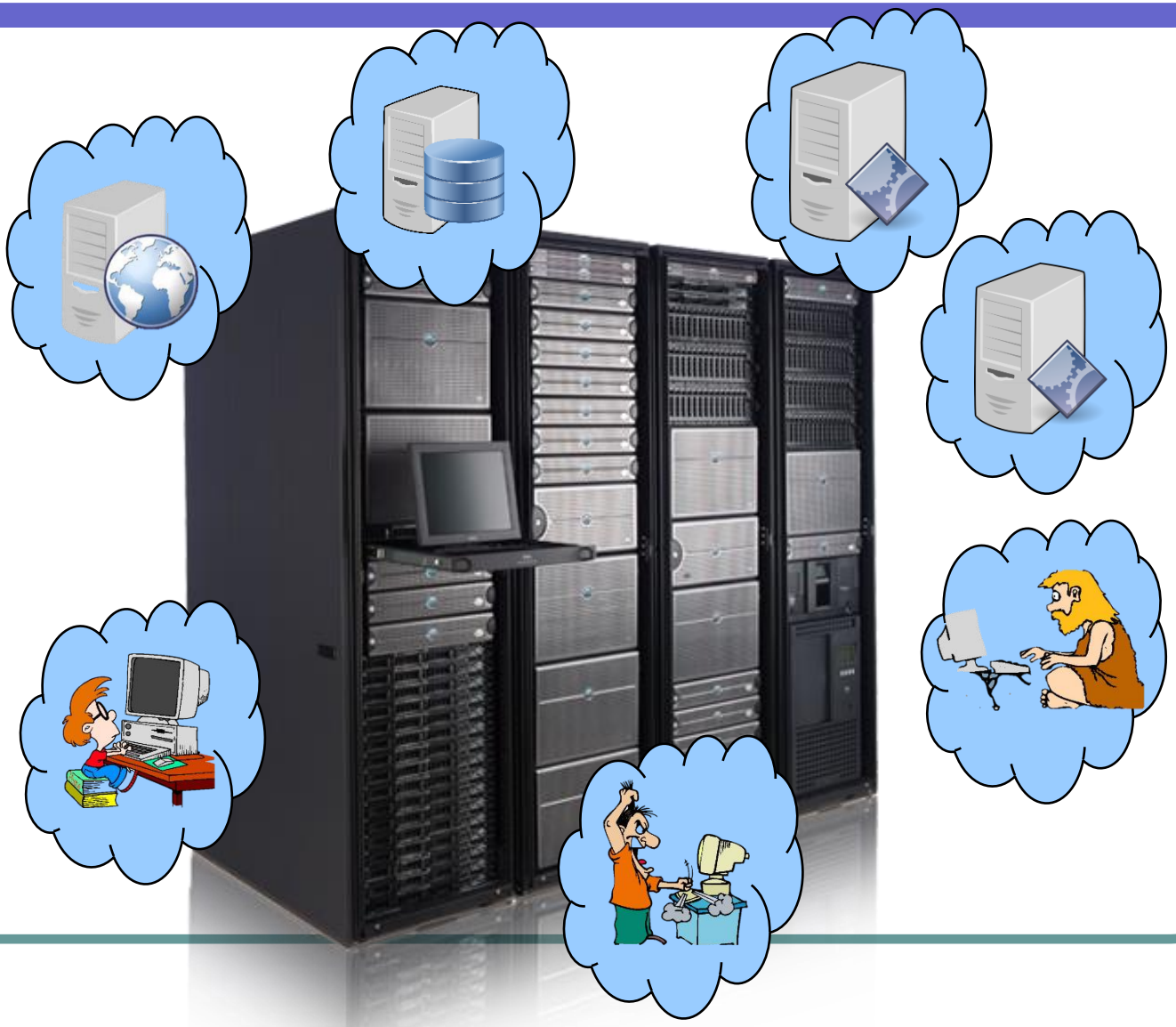
- Argomenti:
  - Utilizzi e benefici della virtualizzazione
  - Architetture e tecniche di virtualizzazione (CPU, memoria, I/O)
  - Approfondimento su VMware workstation
- Riferimenti:
  - A. Tanenbaum, H. Bos, "*Modern Operating Systems*," Pearson ed., 4ª edizione, 2015 (**Capitolo 7: Virtualization and the Cloud**)
  - Neiger et al., "*Intel® Virtualization Technology: Hardware Support for Efficient Processor Virtualization*", Intel Technology Journal, 10(3), 2006

Macchine virtuali

- Una **macchina virtuale** (VM) è una emulazione (mediante tecniche sw/hw) di un macchina reale
- Ogni macchina virtuale esegue il proprio **sistema operativo** ed applicazioni
- Le VM e risorse sono gestite da un **virtual machine monitor** (VMM), o **hypervisor**
- Più macchine virtuali possono condividere le risorse fisiche della macchina su cui eseguono

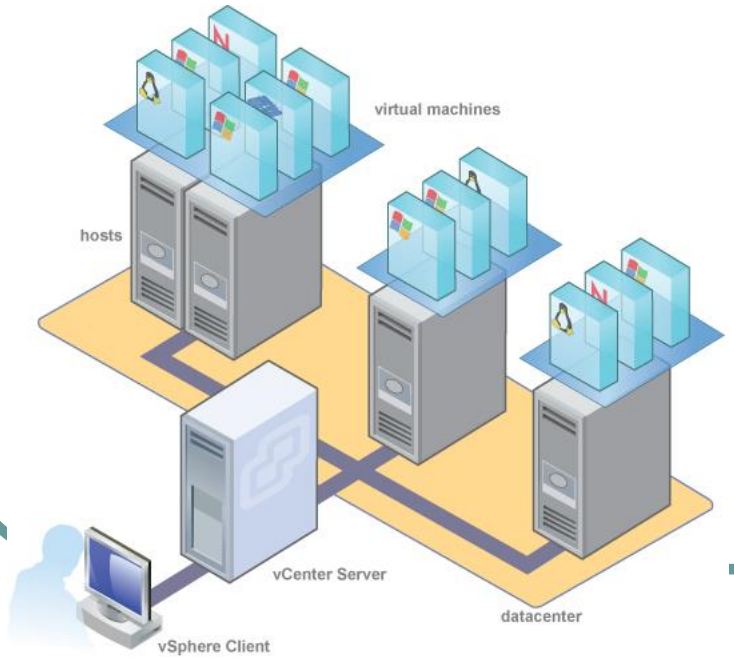
Macchine fisiche e virtuali

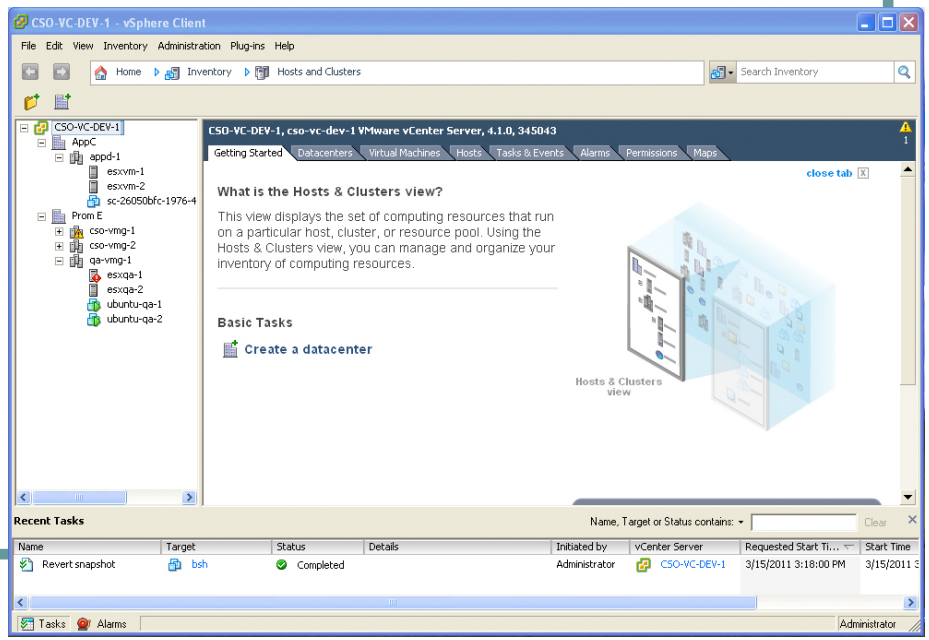
Sistemi virtualizzati



Gestione di sistemi virtualizzati

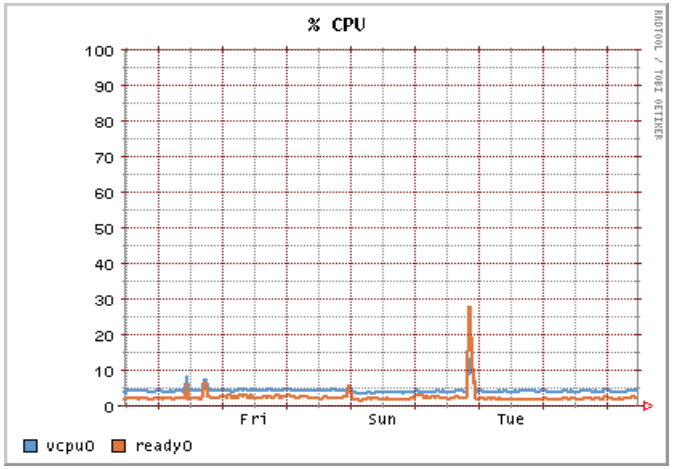
- Le macchine virtuali possono essere **velocemente** create, configurate, monitorate, **migrate**, ..., attraverso semplici strumenti software centralizzati
- Costi ridotti, manutenzione più semplice



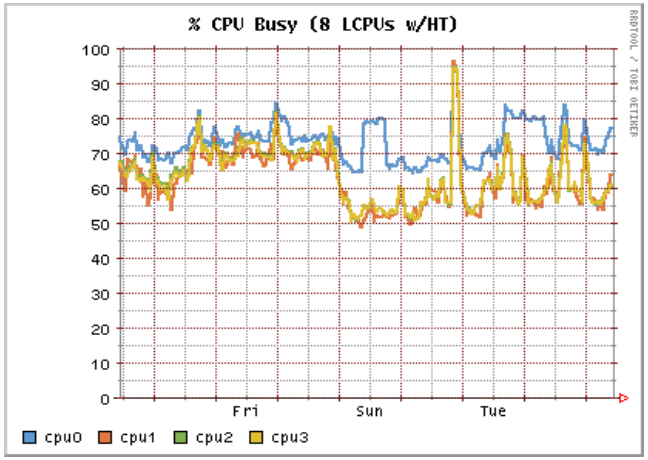


Efficienza

- Avere più VM su una stessa macchina fisica (**workload consolidation**) permette di sfruttare appieno la capacità dell'hardware e di ridurre i consumi energetici



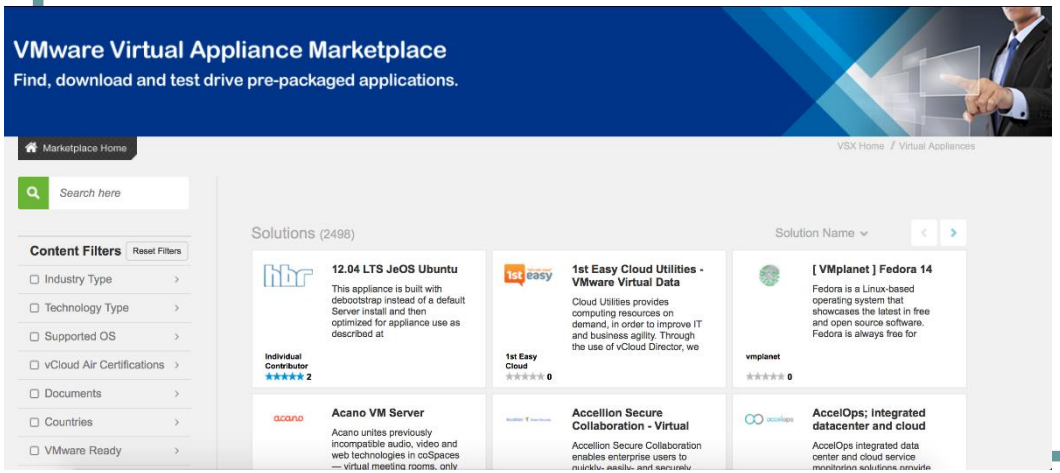
Un singolo server

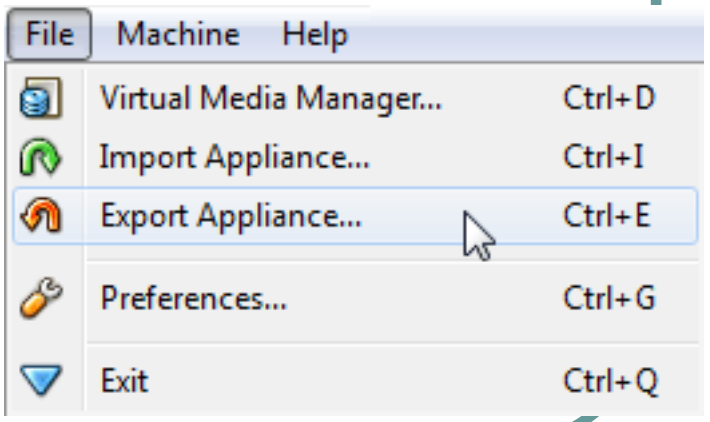


Server consolidati

Flessibilità

- Applicazioni **legacy**, basate su SO obsoleti e non più supportati, possono essere eseguite **allo stesso tempo e sullo stesso HW** delle applicazioni moderne
- Una applicazione (**virtual appliance**) può “portarsi dietro” il suo ambiente di esecuzione (versione di SO e librerie, configurazione, ...)



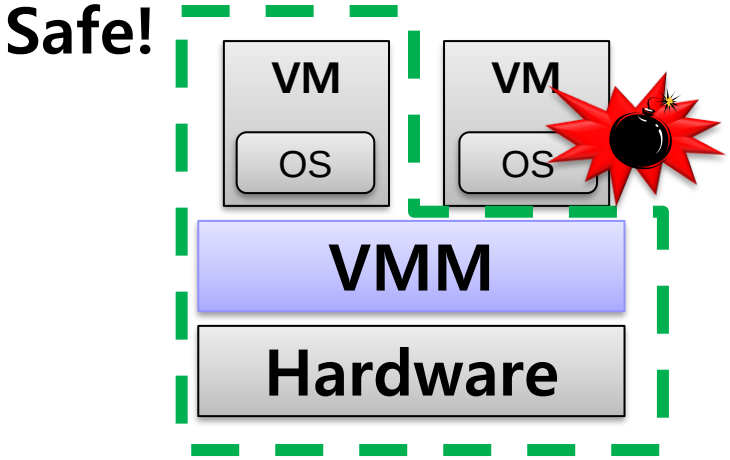


Virtualizzazione e cloud computing

- Il cloud computing è un nuovo modello per la fornitura di risorse hardware e software (***pay-per-use***)
- I fornitori hanno ampie risorse e personale specializzato, e possono condividere le risorse tra più organizzazioni (**multi-tenancy**) per realizzare economie di scala
- I clienti sono sollevati dai costi di acquisizione e manutenzione di un centro di calcolo

Affidabilità e sicurezza

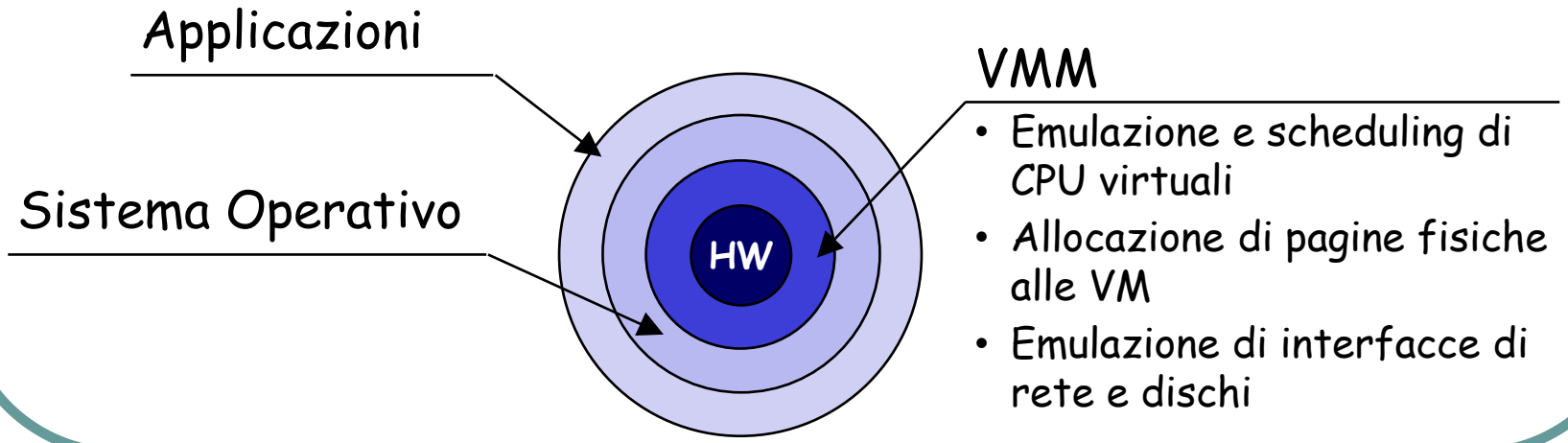
- L'uso di macchine virtuali garantisce un buon grado di affidabilità e sicurezza
- Il SO nella VM "vede" una macchina reale **isolata**, e non può danneggiare le altre VM o il VMM
- Il VMM è l'unico componente **privilegiato** che può gestire l'hardware fisico e le macchine virtuali



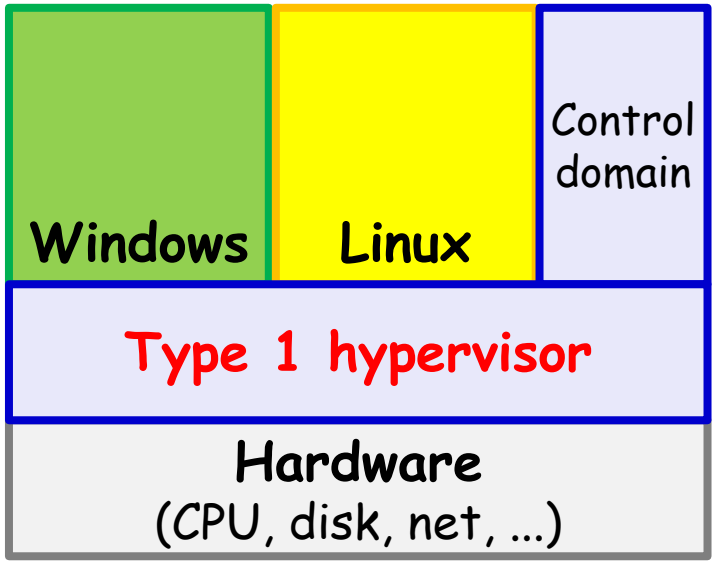
VMM e OS a confronto

- Entrambi gestiscono risorse hardware, e forniscono una astrazione della macchina

Risorse	Sistema Operativo	VMM
CPU	Processi, Thread	CPU virtuale
Memoria	Memoria virtuale	Memoria virtuale
Disco	File, Directory	Disco virtuale
Rete	Socket	Rete virtuale



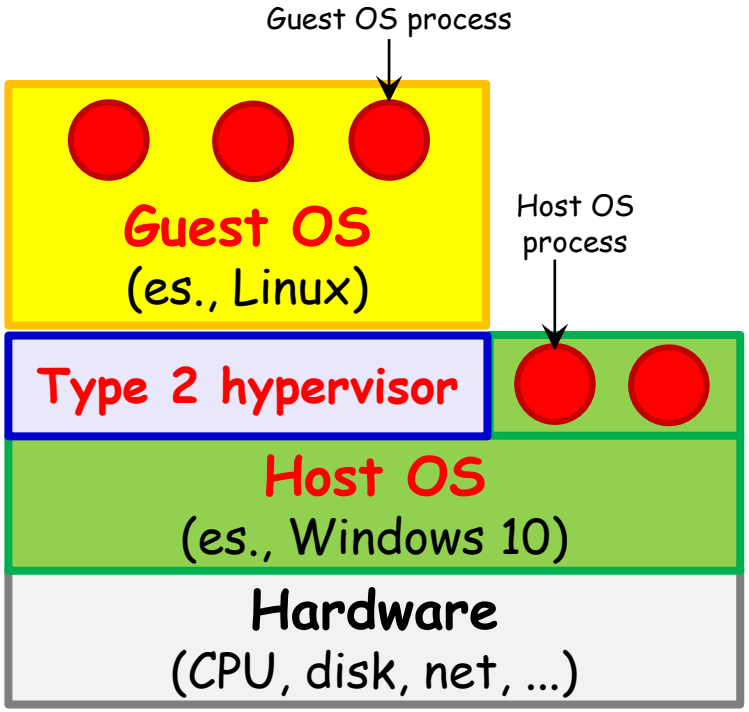
Architetture di virtualizzazione



***Hypervisor Tipo 1*:**

il VMM esegue su "hardware nudo"

(**bare-metal virtualization**)



***Hypervisor Tipo 2*:**

il VMM esegue su un SO tradizionale (es. Windows)

(**hosted hypervisor**)

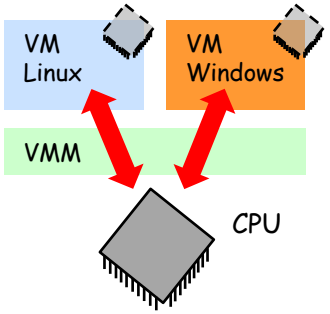
Architetture di virtualizzazione

- Le architetture di tipo 1 consentono le migliori prestazioni. È la soluzione più utilizzata in **ambito server**
  - VMware ESXi; Xen; Hyper-V
- Le architetture di tipo 2 facilitano l'integrazione tra i SO nelle VM (**guest OS**) con il SO di partenza dell'utente (**host OS**). È la soluzione più utilizzata in **ambito desktop**
  - VMware Workstation/Fusion; Oracle VM VirtualBox

VIRTUALIZZAZIONE DELLA CPU

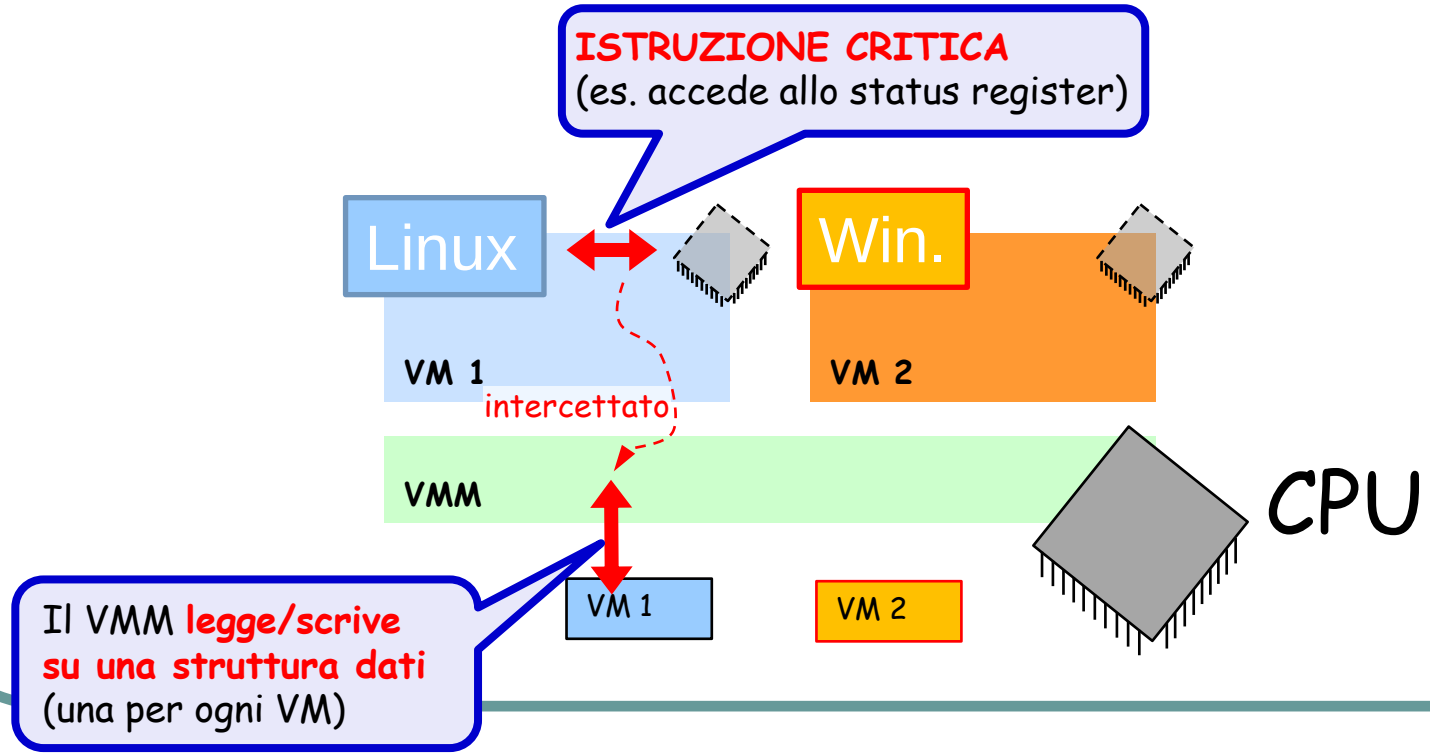
La “virtualizzabilità” della CPU

- I SO tradizionali sono sviluppati **assumendo** che essi abbiano accesso **esclusivo e privilegiato** a:
  - Lo stato privilegiato del processore (registri di controllo, etc.)
  - Eccezioni (page faults, machine check exceptions, ...)
  - Interrupts e interrupt masking
  - Traduzione degli indirizzi (tabelle delle pagine)
  - Accesso della CPU all’I/O (I/O ports e MemoryMapped I/O)
- La **coesistenza** di più SO e del VMM pone il problema delle **interferenze nell’uso della CPU**



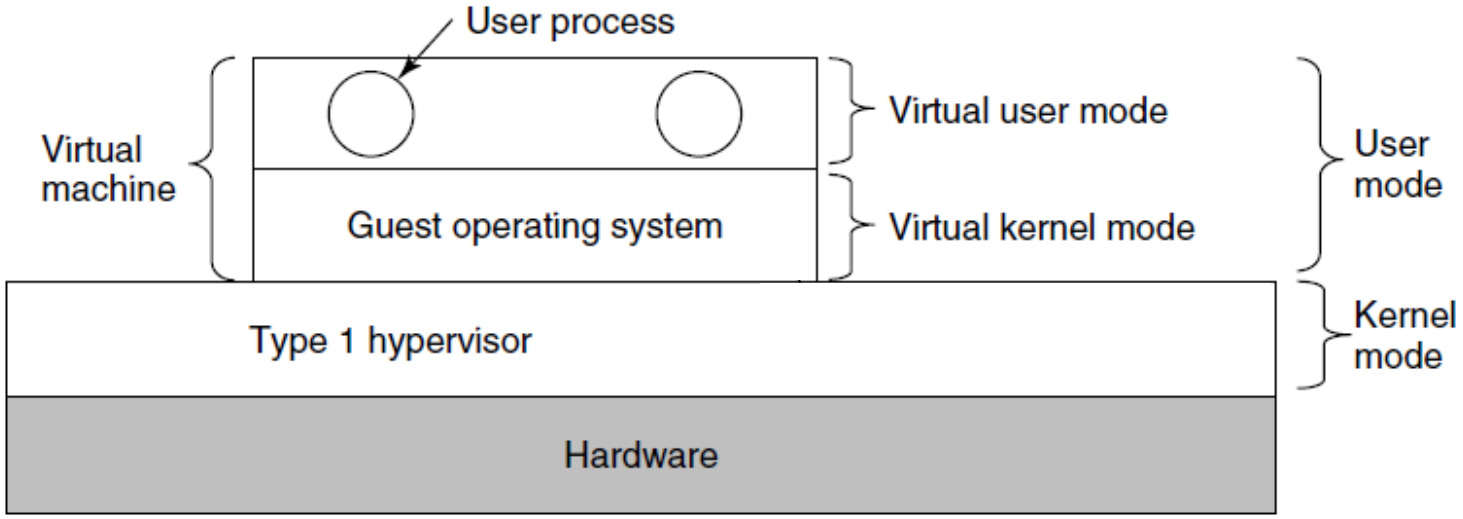
La “virtualizzabilità” della CPU

- Il VMM espone al guest OS una **emulazione** della CPU
- Intercetta l'esecuzione delle "istruzioni critiche” (**sensitive instructions**)



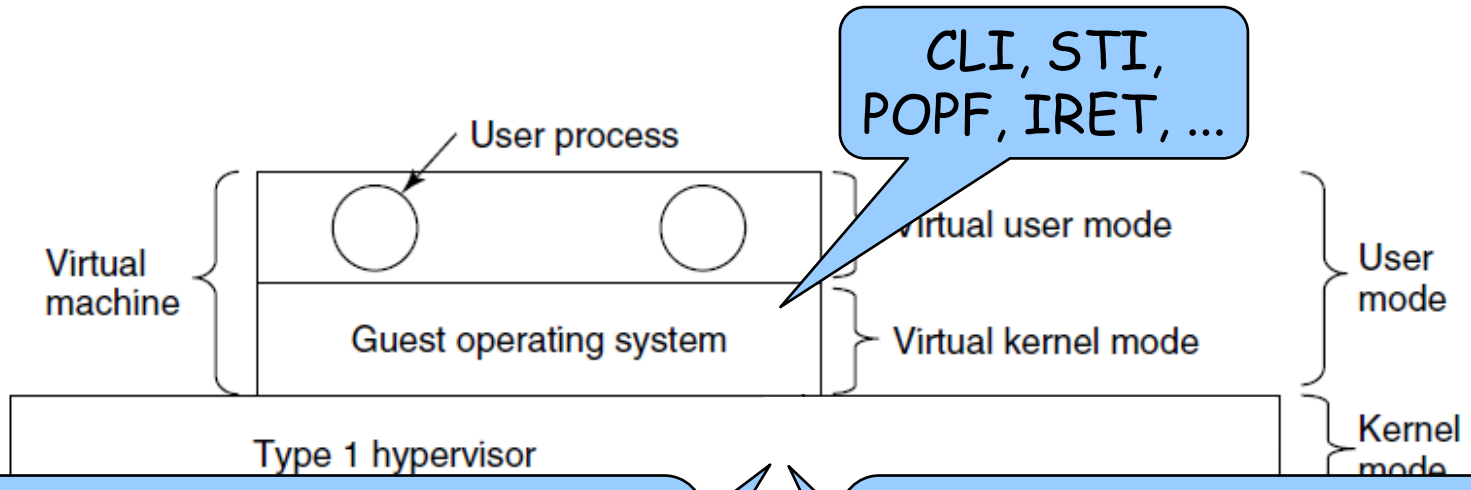
De-privileging

- **De-privileging**: i *guest OS* e il VMM coesistono, eseguendo a diversi **livelli di privilegio**
  - Il VMM occupa il ruolo precedentemente ricoperto dal SO (**kernel/supervisor mode**)
  - Le VM (guest OS+processi) sono in **user mode**, a sua volta distinto in **virtual kernel mode** e **virtual user mode**



Trap-and-emulate

- Il guest OS tenta di usare le istruzioni “sensibili” della CPU (**sensitive instructions**)…
  - Gestione di registri di I/O, MMU, interrupt, stato della CPU, ...
- …il VMM intercetta ed emula (**trap-and-emulate**) queste istruzioni, mantenendo il controllo della CPU fisica



La CPU genera una **trap** se il guest SO esegue una “**sensitive instruction**” **in user-mode**

Il VMM simula **l’effetto dell’istruzione** in software, o con hardware speciale

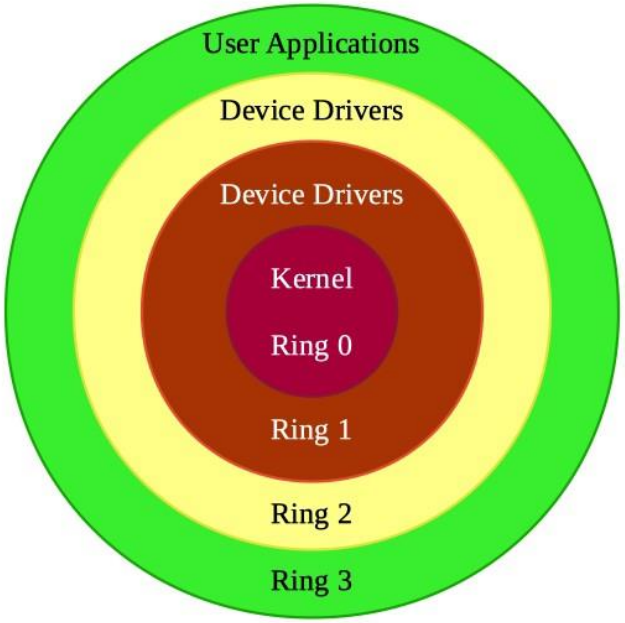
Esempio di trap-and-emulate

1. Il guest SO tenta di disattivare le interruzioni, eseguendo l'istruzione **CLI** ("clear interrupts")
2. Chiaramente, il guest SO **non deve poter privare** il VMM e le altre VM delle interruzioni!
3. Per questo motivo, al momento della CLI, il VMM viene **innescato da una trap** della CPU
4. Il VMM pone a 0 lo Interrupt Flag all'interno di una **struttura dati** dedicata alla VM, e non farà arrivare alcun interrupt alla VM
5. Si noti che lo Interrupt Flag nel **registro fisico** di stato della CPU rimane inalterato!

Nei processori Intel

Sono presenti quattro livelli di protezione (**PL o ring**) per isolare e proteggere i programmi utente da altri programmi e dal sistema operativo

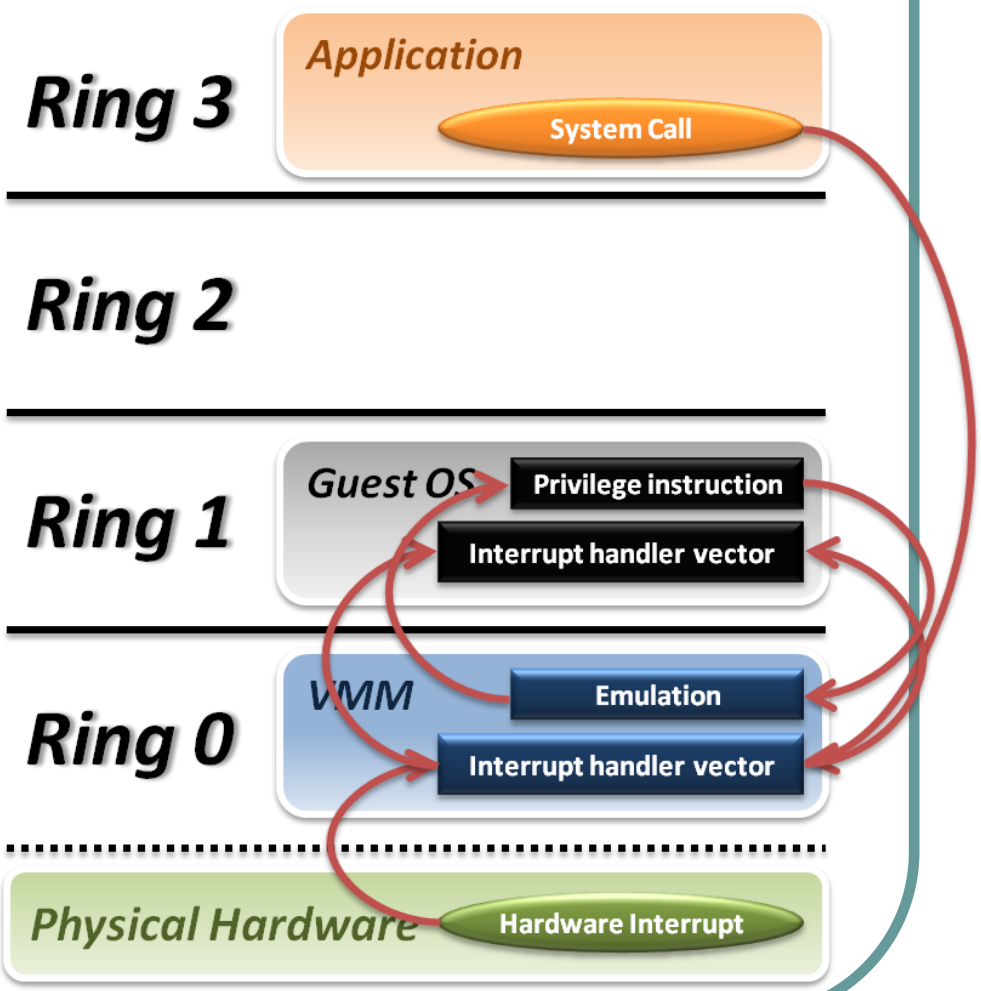
Ogni task che è eseguito dalla CPU è sempre classificato in base ad uno dei quattro livelli di privilegio. Il **CPL** (*Current Privilege Level*) specifica il livello di privilegio del task corrente, questo valore può essere variato solo mediante **istruzioni sensibili**.



Standard IA Protection Rings

Esempio nel caso di CPU Intel

- **System call**
  - La CPU genera una trap gestita dal VMM
  - Il VMM salta al guest OS
- **Interrupt hardware**
  - L'interrupt hardware viene seguito dalla ISR del VMM
  - Il VMM salta alla corrispondente ISR del guest OS
- **Istruzioni privilegiate**
  - L'esecuzione di istruzioni privilegiate nel guest OS causano trap gestite dal VMM
  - Il VMM emula l'istruzione e salta di nuovo al guest OS



La (non) “virtualizzabilità” in CPU Intel

- L’architettura x86 tradizionale **non è** “**virtualizzabile**” usando semplicemente trap-and-emulate
- In x86, purtroppo molte delle istruzioni sensibili **non generano alcuna trap**!
  - Se un processo utente o il guest OS tentano di eseguire l’istruzione sensitive, la CPU **ignora l’istruzione** (**senza innescare la trap e il VMM**)
  - Tali istruzioni sono dette “sensibili ma non-privilegiate”

Tecniche di virtualizzazione CPU

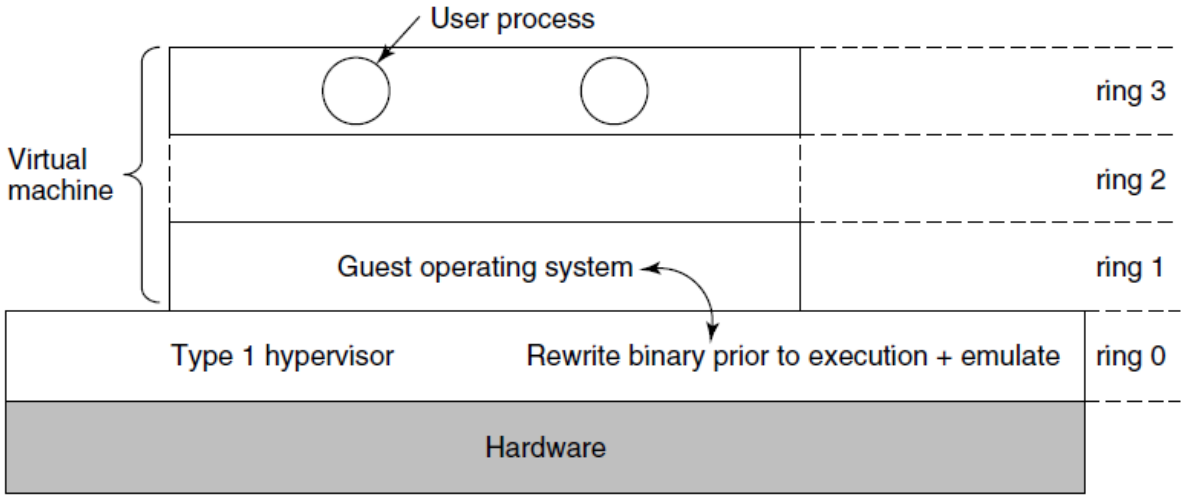
- **Full virtualization**, senza supporto hardware
  - Utilizza tecniche software (dynamic binary translation, shadow page tables, ...)
- **Para-virtualizzazione**
  - Il guest SO è “ri-scritto” per cooperare con il VMM
- **Full virtualization**, con **supporto hardware** (rispettivamente Intel VT e AMD SVM)
  - Migliori prestazioni e VMM più semplice

Virtualization & Cloud Computing

Virtualization Solutions	Cloud Providers
vmware Full Virtualization (hw assisted, sw)	terremark, CONTEGIX, bluemile, TIER 3, bluelock., DATAPIPE
Xen™ Para-Virtualization	amazon web services™, rackspace HOSTING, SOFTLAYER, CARPATHIA HOSTING, INC., GOGRID
KVM Full Virtualization (hw assisted)	IBM, DATAPIPE, Joyent, ElasticHosts Flexible servers in the cloud

Full virtualization, no supporto hw

- Nel 1999, l'hypervisor di VMware introdusse tecniche efficienti di full-virtualization per Intel x86
- Il **codice binario** del guest OS viene **riscritto** "al volo" dal VMM, sostituendo le istruzioni sensibili con del codice di emulazione



Dynamic binary translation (1)

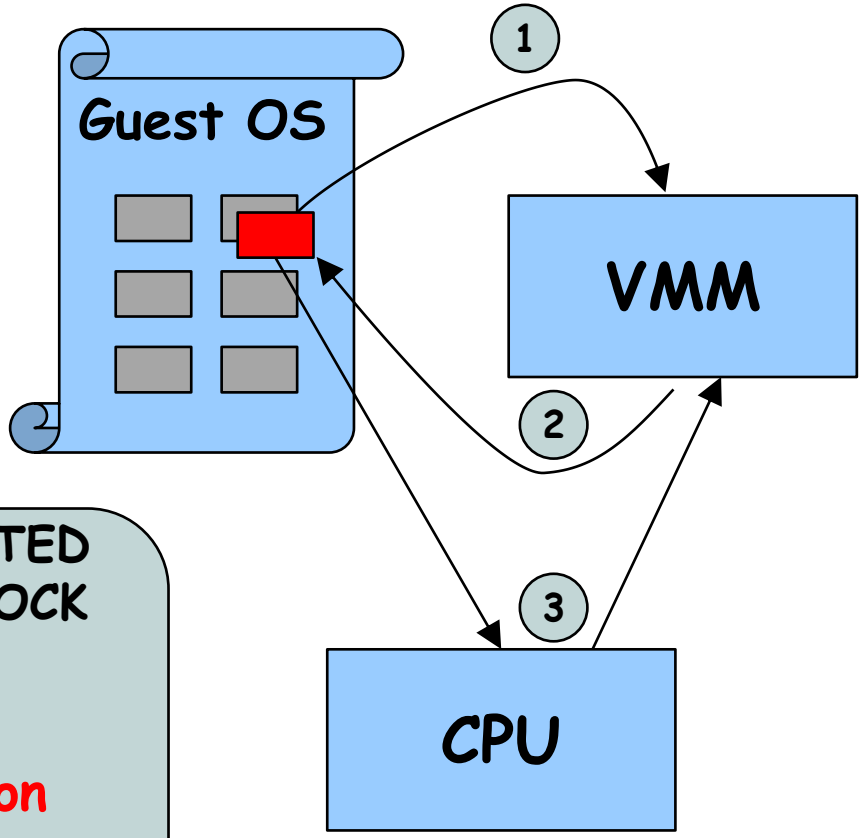
- Il VMM, sin dall'avvio della VM, analizza **a blocchi** il codice eseguito dal guest OS
- Il VMM **riscrive** le eventuali istruzioni privilegiate con codice di emulazione
- Inoltre, il **salto finale** viene sostituito con una chiamata al VMM, in modo che il VMM **possa avanzare il processo di traduzione** sul prossimo blocco

BASIC BLOCK

```
mov …
add …
cli
add …
…
jmp <guest>
```

*Un blocco è una breve **sequenza di istruzioni sequenziali** che termina con una **istruzione di salto***

Dynamic binary translation (2)



BASIC BLOCK

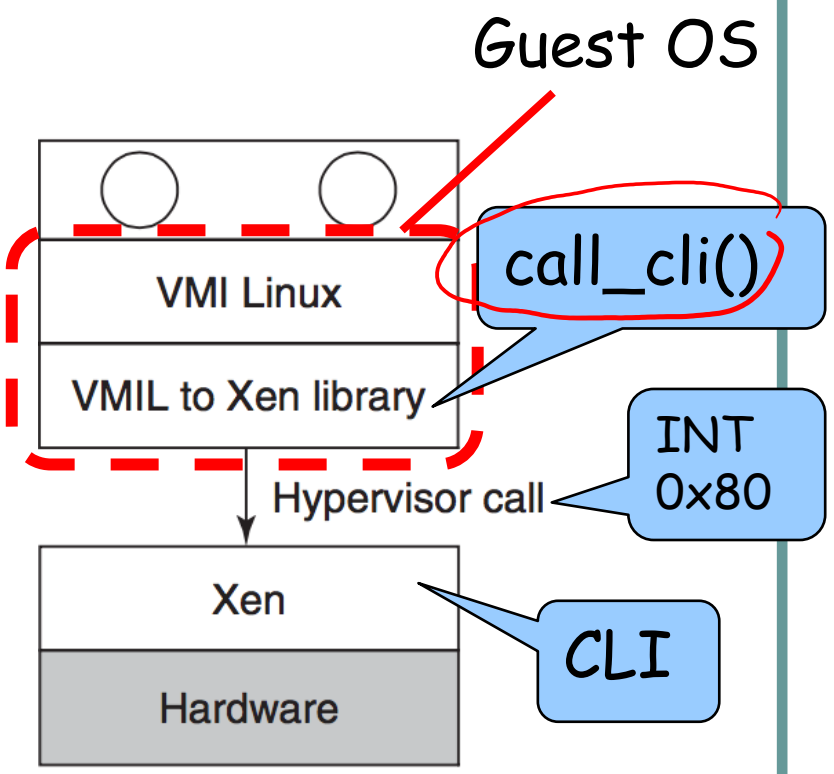
```
mov ...
add ...
cli
add ...
...
jmp <guest>
```

TRANSLATED BASIC BLOCK

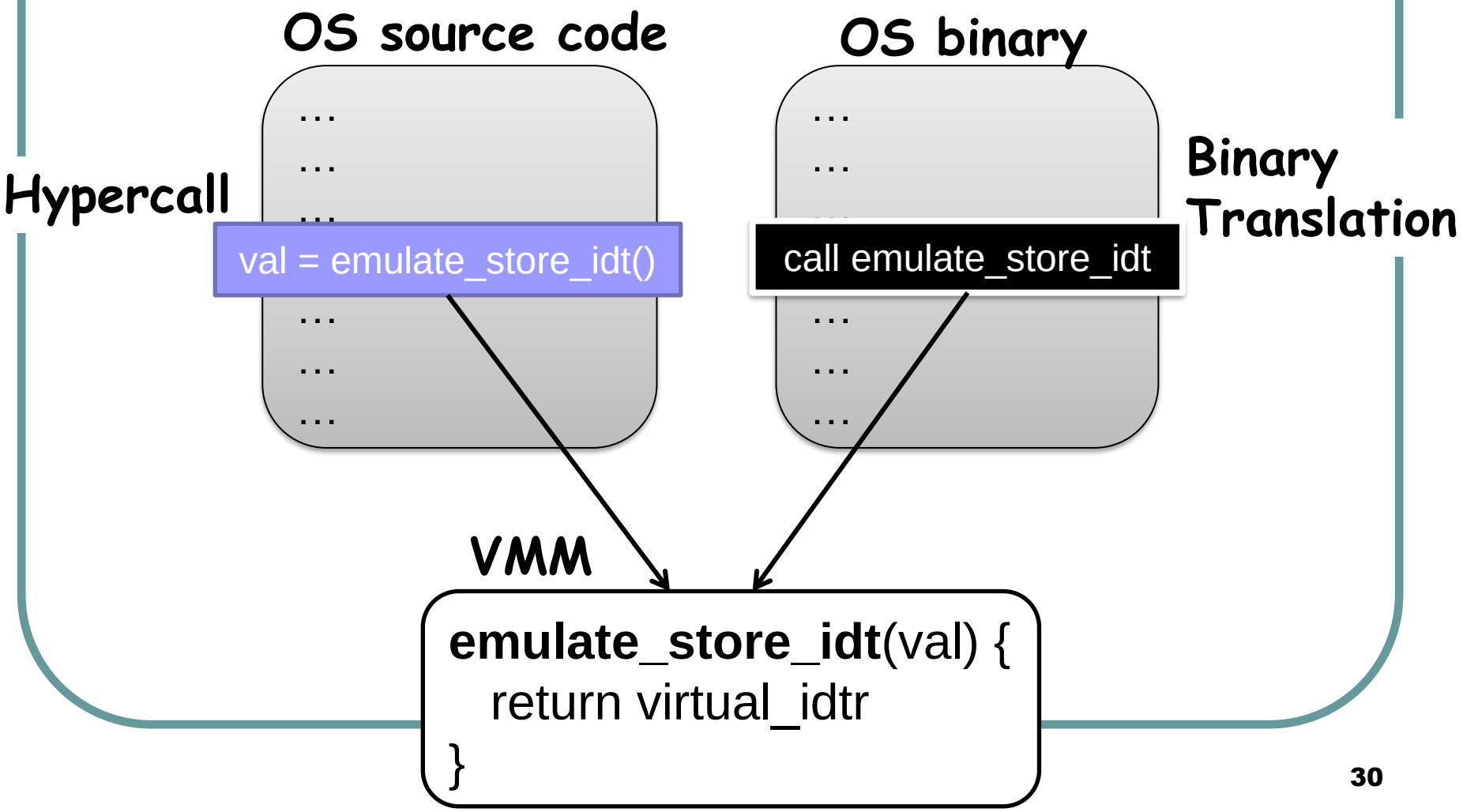
```
mov ...
add ...
cli emulation
add ...
...
jmp <VMM>
```

Para-virtualizzazione

- Il VMM **espone** al guest OS una interfaccia software (**hypercalls**), che viene usata dal guest OS al posto delle istruzioni sensibili
- Richiede che il guest OS sia "**scritto**" in modo speciale per eseguire sul VMM, invece che sull'hardware fisico
- Non applicabile per sistemi **legacy** oppure **proprietari**, come Windows



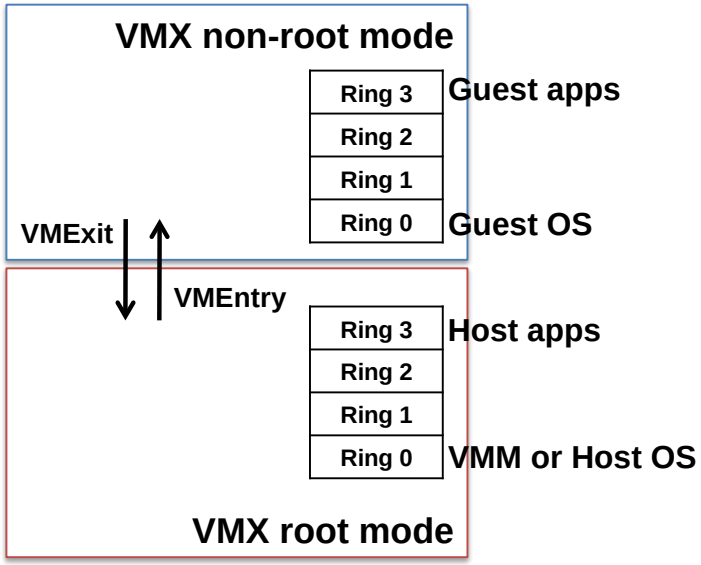
Hypercall e DBT a confronto

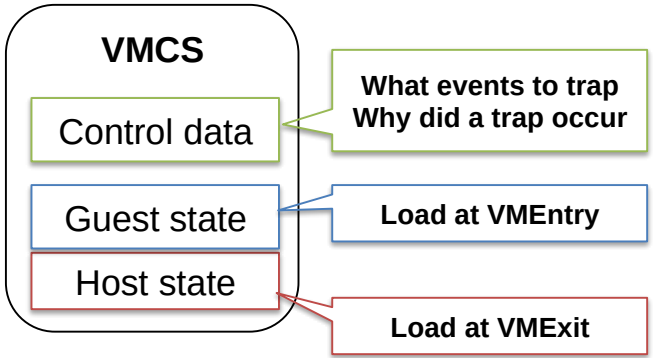


Supporto hardware per la virtualizzazione della CPU (1)

- Le CPU Intel VT introducono **VMX root/non-root modes** e **VMCS** (VM Control Structure)
  - **Semplificano** il VMM, buona parte delle operazioni di virtualizzazione è in hardware
  - Evitano il **ring de-privileging** del guest OS
  - **Configurazione flessibile** del meccanismo di trap, a livello di singola istruzione
  - Maggiore **efficienza** del cambio di contesto tra VM e VMM

Supporto hardware per la virtualizzazione della CPU



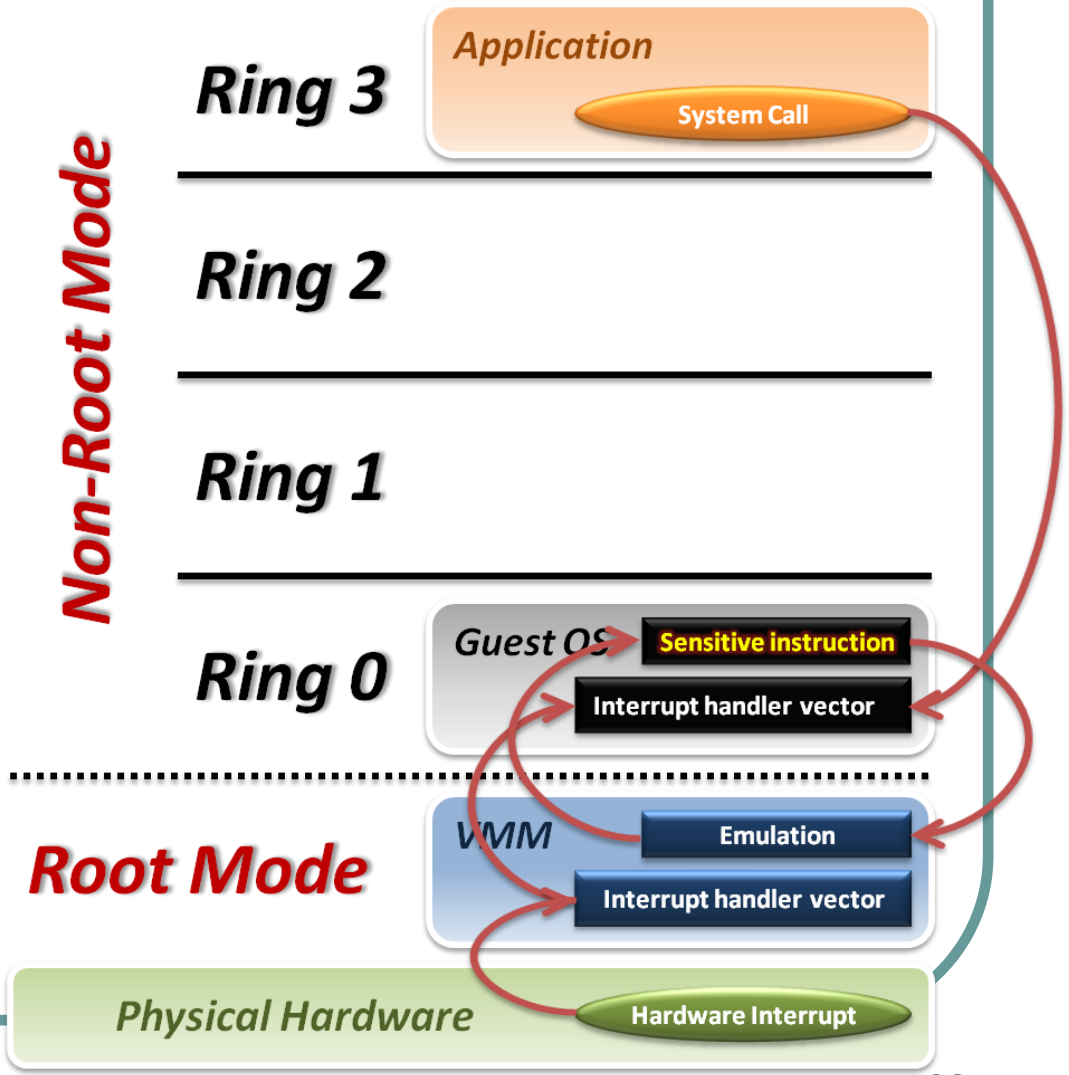


Struttura dati
(una per VM/CPU virtuale)

- **VMX root** mode: esecuzione del **VMM**
- **VMX non-root** mode: esecuzione del **guest OS**
  - Il comportamento delle istruzioni sensibili è **configurato dal VMM**
- **VM-exit**, **VM-enter**: eventi che causano il passaggio root/non-root (es., timer, page fault, ...)

Supporto hardware per la virtualizzazione della CPU (3)

- **System call**
  - La trap è gestita direttamente dalla ISR del guest OS
- **Interrupt hardware**
  - Gli eventi hardware sono sempre gestiti dal VMM, ma con ottimizzazioni
- **Istruzioni privilegiate**
  - Solo lo stretto necessario delle istruzioni privilegiate causano trap gestite dal VMM



VM Control Structure (VMCS)

- Il VMM usa la VMCS per indicare alla CPU le condizioni e le azioni di **VM-entry** e **VM-exit**
- Lo **stato della CPU** è salvato/caricato in **hw** nella VMCS
- Ogni **CPU virtuale** ha una apposita VMCS
- Solo **una VMCS** alla volta è attiva su una **CPU fisica**

VM-execution controls	Determines what operations cause VM exits	CR0, CR3, CR4, Exceptions, IO Ports, Interrupts, Pin Events, etc.
Guest-state area	Saved on VM exits Reloaded on VM entry	EIP, ESP, EFLAGS, IDTR, Segment Regs, Exit info, etc.
Host-state area	Loaded on VM exits	CR3, EIP set to monitor entry point, EFLAGS hardcoded, etc.
VM-exit controls	Determines which state to save, load, how to transition	Example: MSR save -load list
VM-entry controls	Determines which state to load, how to transition	Incl uding injecting events (interrupts, exceptions) on entry

VM-exit

- Esempi di cause di VM-exit:
  - Istruzioni sensibili
    - **CPUID**: riporta le CPU capabilities
    - **RDMSR**, **WRMSR**: legge/scrive i “model-specific registers”
    - **INVLPG**: invalida entry TLB
    - **RDPMC**, **RDTSC**: legge i registri di perf. monitoring e di timestamp
    - **HLT**, **MWAIT**, **PAUSE**: disattivazione del guest OS
    - **VMCALL**: nuova istruzione per invocare il VMM
  - Accessi a stato sensibile
    - **MOV DRx**: accessi ai debug register
    - **MOV CRx**: accessi ai control register
    - **Task switch**: accessi al CR3 (puntatore alla tabella della pagine)
  - Eccezioni ed eventi asincroni
    - Page fault, debug exceptions, interrupts, etc.

VM-entry

- Il VMCS consente di “iniettare” eventi (interrupt, eccezioni) nella VM, al momento del VM-entry
- L’iniezione è fatta dalla CPU in hardware, semplificando il VMM e migliorando le prestazioni

Esempio: Page faults

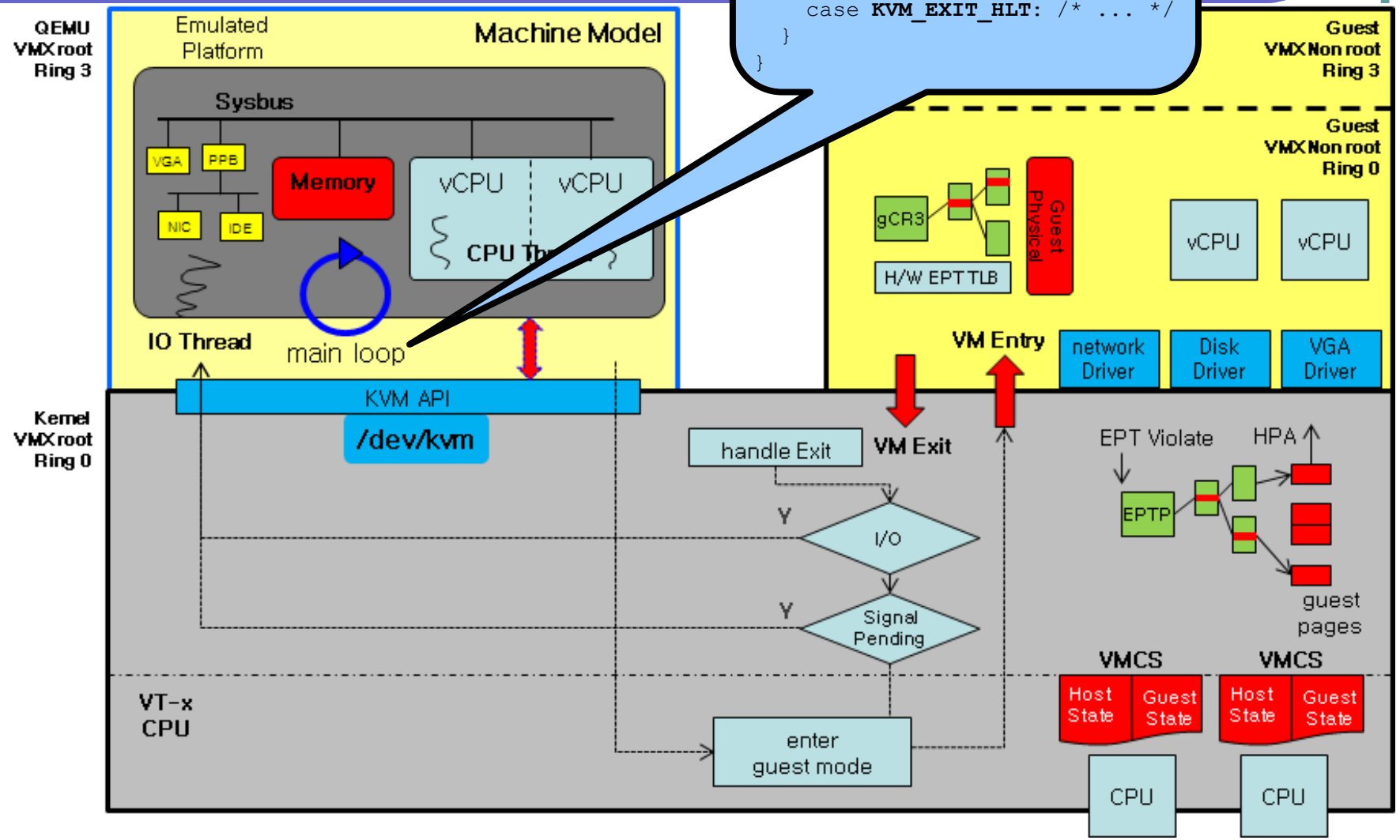
1. La CPU scrive nel VMCS le cause dell’eccezione
2. Il VMM sceglie se **gestire il page fault**, in modo trasparente al guest OS (es., paginazione a domanda), ...
3. ... oppure iniettare l’eccezione nel guest OS, utilizzando il campo VM-entry controls nel VMCS

Esempio: Interrupt masking

1. Il guest OS usa istruzioni CLI/STI (caso frequente)
2. La CPU apre/chiude la “interrupt window” della VM, **senza innescare** il VMM con delle VM-exit
3. Quando si verifica un interrupt, il VMM “accoda” l’interrupt alla VM
4. La CPU inietterà automaticamente l’interrupt alla apertura della interrupt window

KVM+QEMU

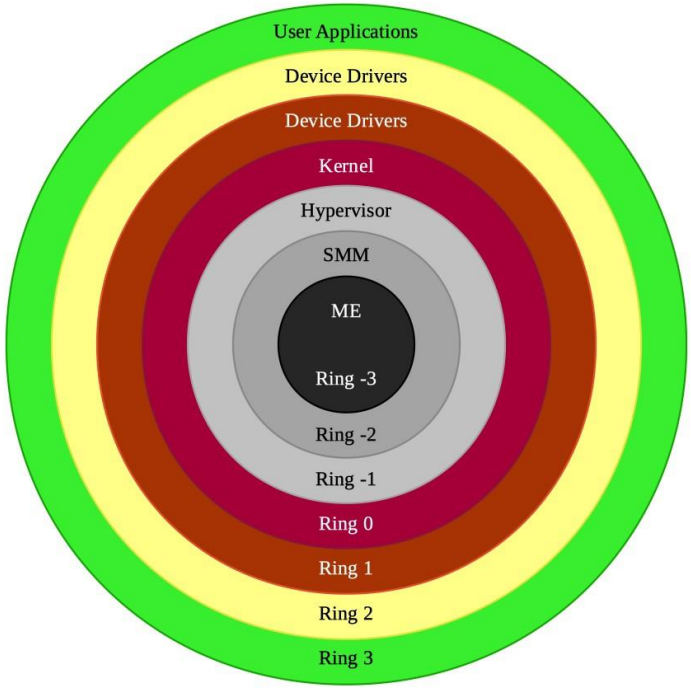
```
open("/dev/kvm")
ioctl(KVM_CREATE_VM)
ioctl(KVM_CREATE_VCPU)
for (;;) {
  ioctl(KVM_RUN)
  switch (exit_reason) {
    case KVM_EXIT_IO:  /* ... */
    case KVM_EXIT_HLT: /* ... */
  }
}
```



NEGATIVE RINGS

the complete view of the ring architecture becomes:

- Ring -3: Management Engine (ME) {Highest Privilege}
- Ring -2: System Management Mode (SMM)
- Ring -1: Hypervisor
- Ring 0: Kernel
- Ring 1: Device Drivers
- Ring 2: Device Drivers
- Ring 3: User Applications {Lowest Privilege}



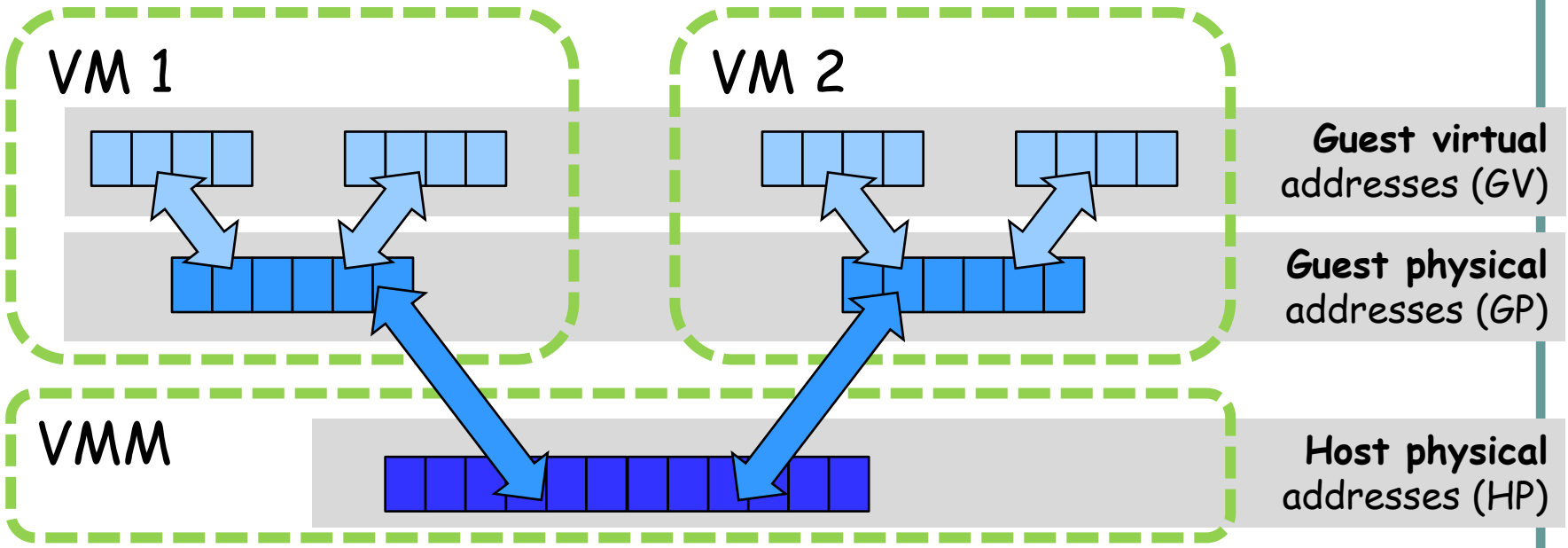
IA Negative Rings

VIRTUALIZZAZIONE DELLA MEMORIA

Virtualizzazione della memoria

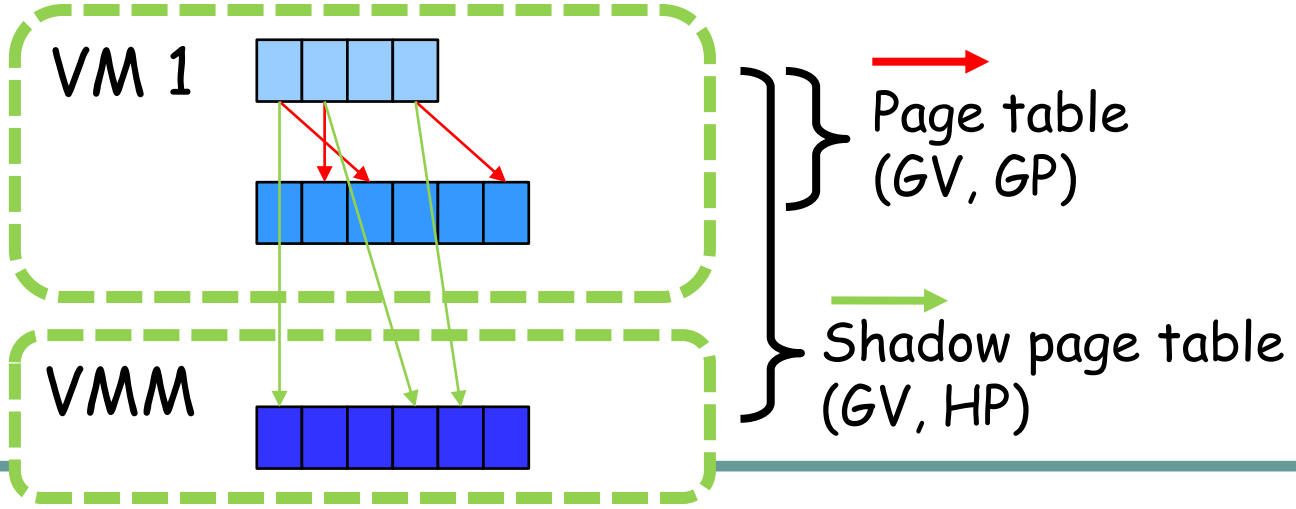
- All’interno di una VM, il guest OS gestisce la **“finta” memoria fisica della VM**, utilizzando i tradizionali meccanismi della **memoria virtuale**
- Il VMM amministra la **“vera” memoria fisica**, tramite paginazione a domanda, swapping, algoritmi di prelazione, ...
- Le VM sono **isolate** e non possono accedere alla memoria di altre VM (in modo simile all’isolamento dei processi nei SO)

Indirizzi virtuali e fisici



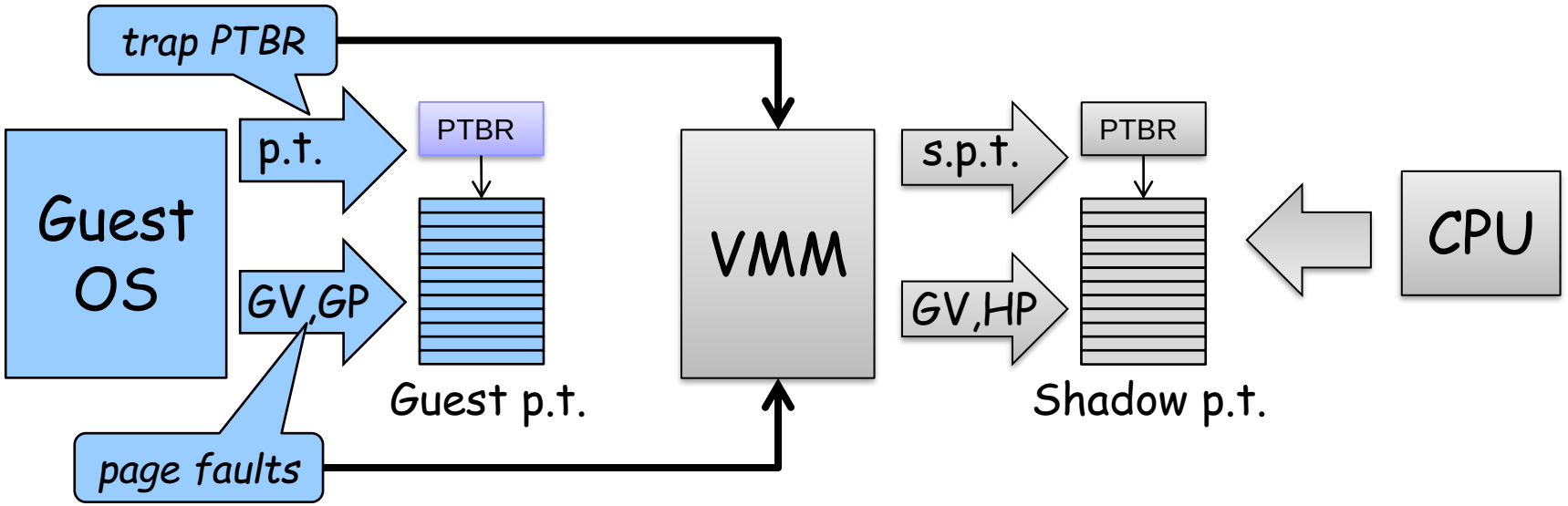
Shadow page tables (1)

- Per ogni **p.t. (page table)** del guest OS, il VMM crea una **shadow page table**
- Il VMM virtualizza il *page-table base register* (**PTBR**), ed intercetta il guest OS quando esso aggiorna una p.t.
- Il VMM decide la assegnazione delle pagine fisiche, e la scrive nella shadow p.t.
- La CPU fisica usa la shadow p.t., ed **ignora la p.t. del guest OS**



Shadow page tables (2)

1. Il guest OS scrive, con una **istruzione sensitive**, nel registro (virtuale) PTBR l'indirizzo della p.t.
2. Con trap-and-emulate, il VMM carica nel registro (fisico) PTBR l'indirizzo della **shadow page table**
3. Il VMM intercetta le modifiche alla p.t. (esso imposta le pagine della p.t. come **read-only**, in modo da causare **page fault**)
4. La **CPU usa la shadow page table** per tradurre gli indirizzi virtuali della VM

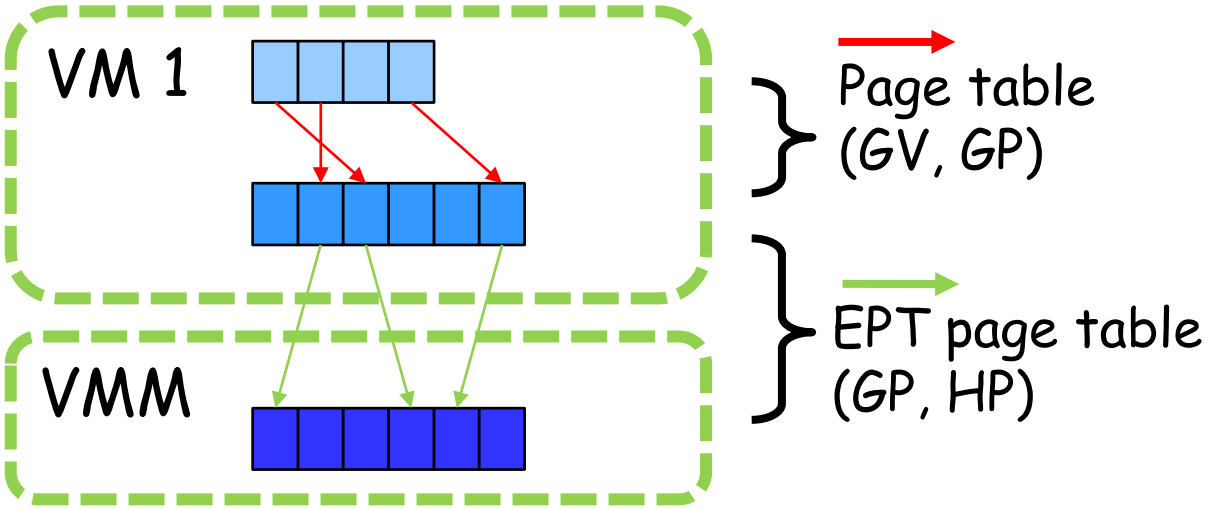


Shadow page tables (3)

- Le shadow page tables sono un meccanismo di virtualizzazione oneroso, poiché aumentano **il volume e l'overhead dei page fault**
  - **Hypervisor-induced page faults**: Page faults introdotti dall'hypervisor per allineare le shadow p.t. e le guest p.t.
  - **Guest-induced page faults**: Page faults "genuini" prodotti dalla VM. Sono intercettati dal VMM e "re-iniettati" nella VM.

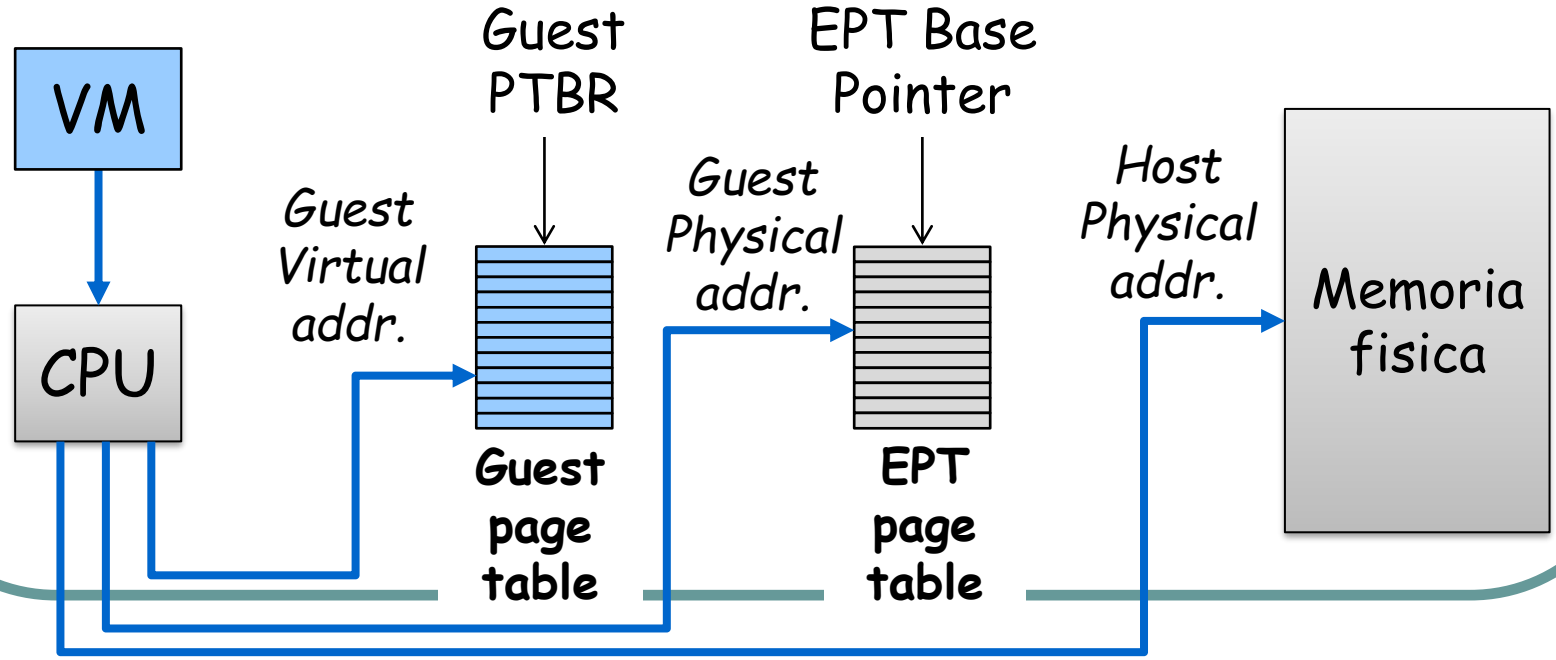
Supporto hardware per virtualizzazione della memoria

- Intel VT introduce un livello aggiuntivo di traduzione degli indirizzi mediante **Extended Page Tables** (EPT), gestite dal VMM
- La CPU accede in hardware ad entrambe le tabelle
- Si evitano hypervisor-induced page faults



Traduzione degli indirizzi

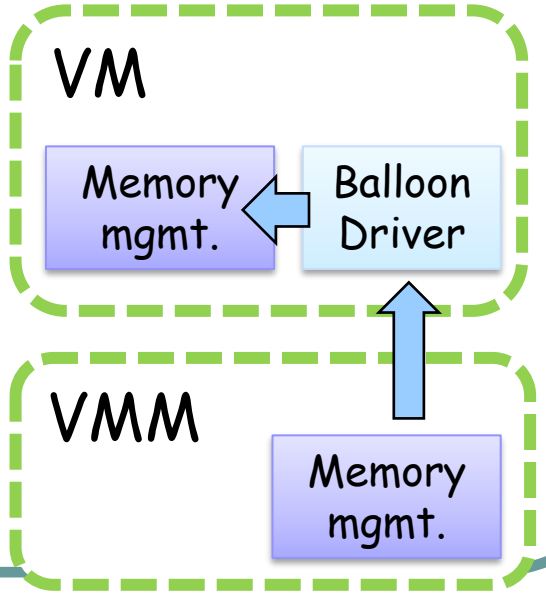
1. La CPU accede le guest page table del guest OS (da **guest virtual addr.** a **guest physical addr.**)
2. La CPU accede le extended page table del VMM (da **guest physical addr.** a **host physical addr.**)



Prelazione delle pagine (ballooning)

- Il VMM gestisce la memoria delle VM mediante **paginazione a domanda**
- Per prestazioni ottimali, è necessario che gli **algoritmi di sostituzione** delle pagine del VMM e del guest OS **cooperino**, tramite la **tecnica del balooning**

1. Il guest OS esegue un **modulo kernel** (**baloon driver**) che alloca pagine virtuali nella VM
2. Il guest OS fa **swap-out** delle pagine meno importanti
3. Il baloon driver comunica le pagine da esso allocate al VMM, che le può ri-utilizzare



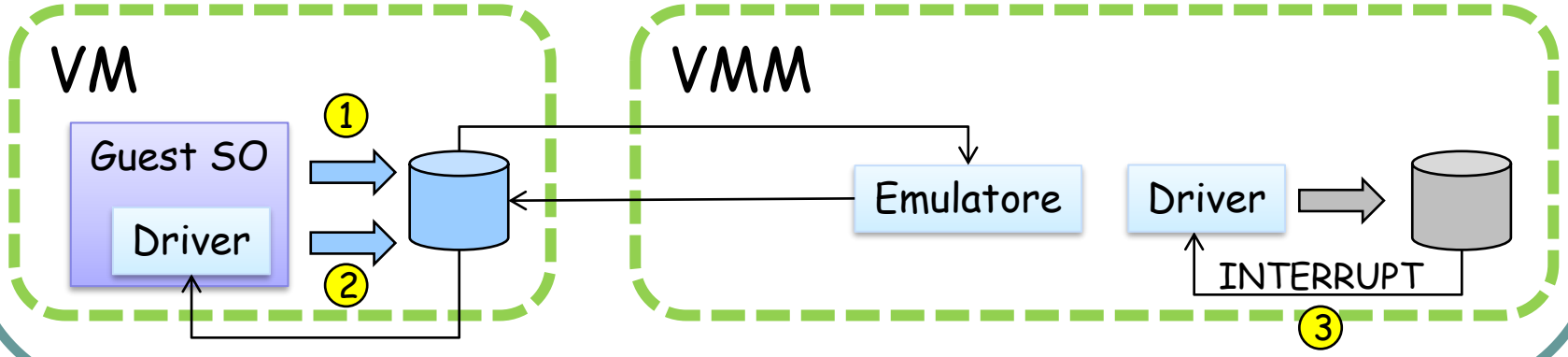
Virtualizzazione dell’I/O

Virtualizzazione dell'I/O

- Le istruzioni di I/O (programmed I/O, memory-mapped I/O, DMA) sono "virtualizzabili"
- Il VMM può intercettarle con *trap-and-emulate*, e simulare un dispositivo virtuale
  - **Disco virtuale**: il VMM decodifica il comando di lettura/scrittura, e lo effettua su un file su disco dedicato al disco virtuale
  - **NIC virtuale**: il VMM decodifica il frame e lo inoltra alla rete fisica o virtuale (**virtual switch**)

I/O full virtualization (1)

1. Il guest OS **interroga il bus** per trovare i dispositivi connessi; il VMM intercetta ed indica il modello di dispositivo virtuale
2. Il guest OS tenta di **leggere/scrivere** i registri del dispositivo virtuale; il VMM intercetta e copia i valori da/verso i registri hardware
3. Il dispositivo hardware genera una **interruzione**; il VMM la serve simulando una interruzione nel contesto della VM



I/O full virtualization (2)

- La I/O full virtualization permette di “mostrare” al guest OS un dispositivo differente da quello fisico
  - Sono spesso simulati dispositivi semplici e ben noti (es. *RTL8139*, *AC97*)
- Le prestazioni sono penalizzate dalla emulazione e dal passaggio attraverso il VMM

KVM+QEMU (vhost-net)

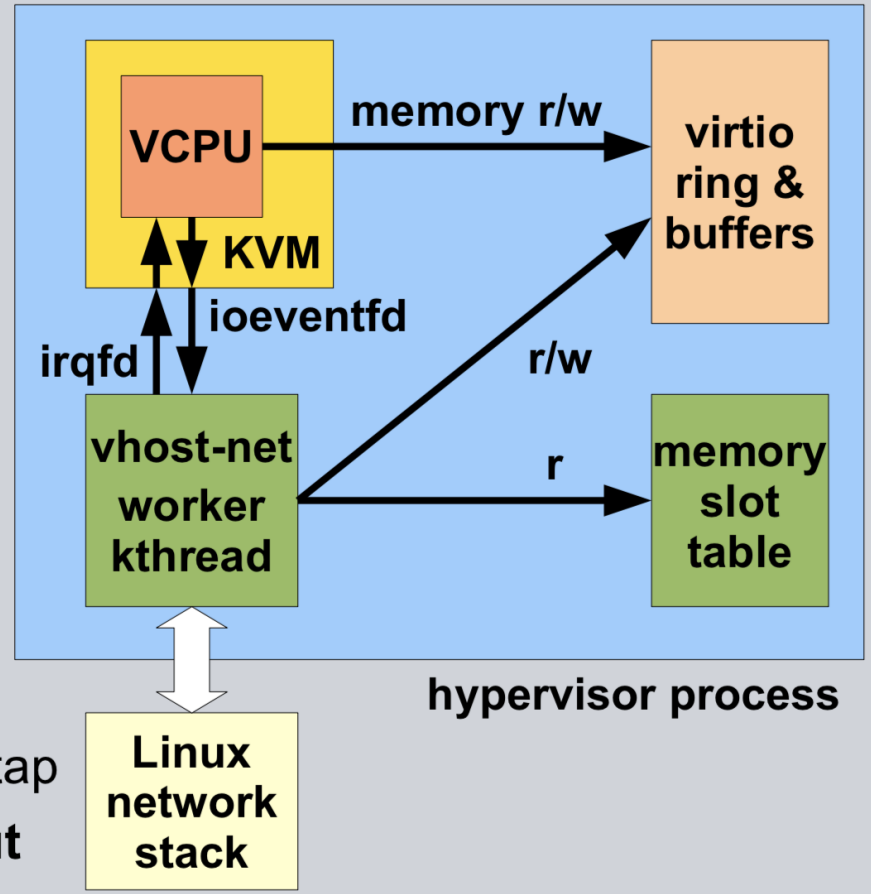
Goal: high throughput / low latency guest networking

- Avoid heavy exits
- Reduce packet copying
- No in-kernel QEMU, please!

The vhost-net model

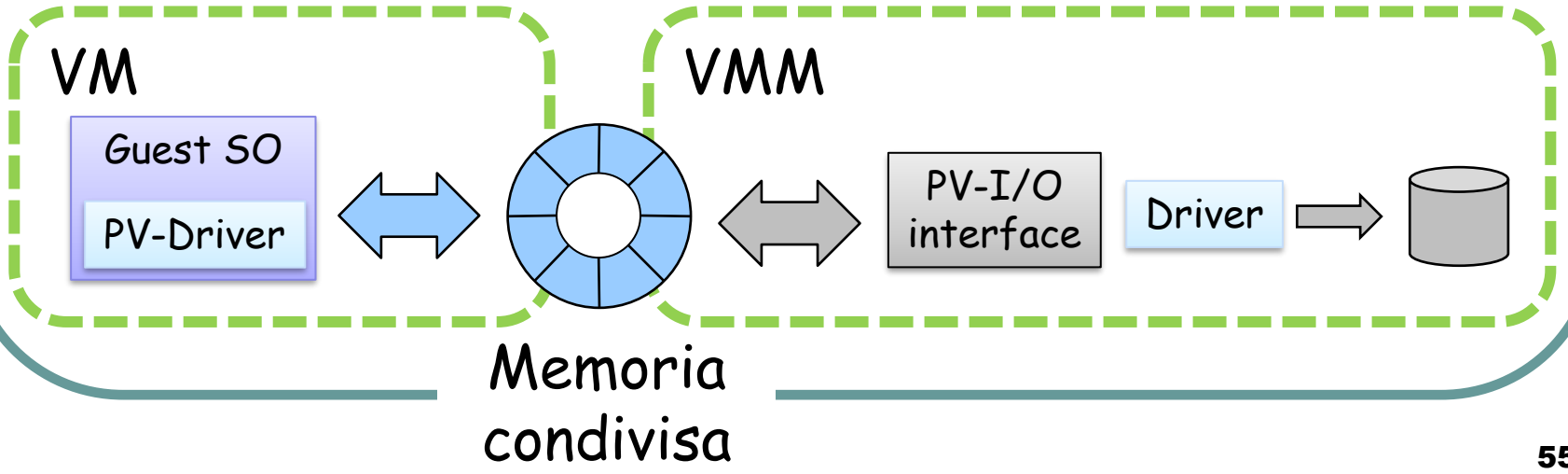
- Host user space opens and configures kernel helper
- virtio as guest-host interface
- KVM interface: eventfd
  - TX trigger → ioeventfd
  - RX signal → irqfd
- Linux interface vie tap or macvtap

Enables multi-gigabit throughput



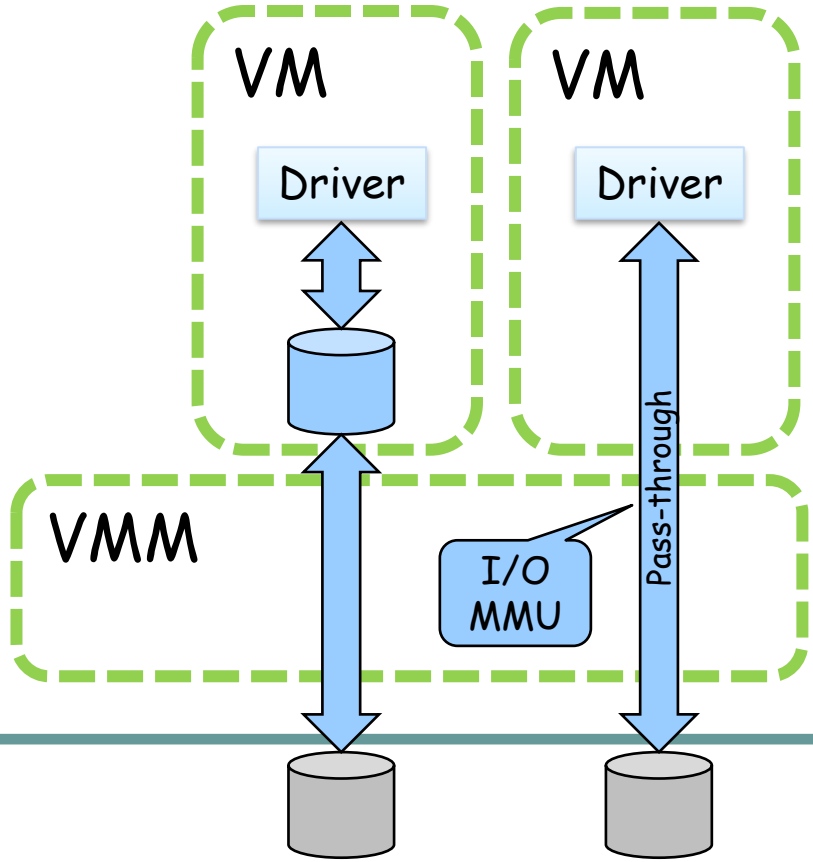
I/O paravirtualization

- Il VMM esporta al guest OS delle aree di **memoria condivisa** e istruzioni speciali per l'I/O
- Il guest OS è "consapevole" della virtualizzazione; esso carica un driver apposito per l'I/O paravirtualizzato
- Migliora le prestazioni perché evita l'emulazione di un dispositivo



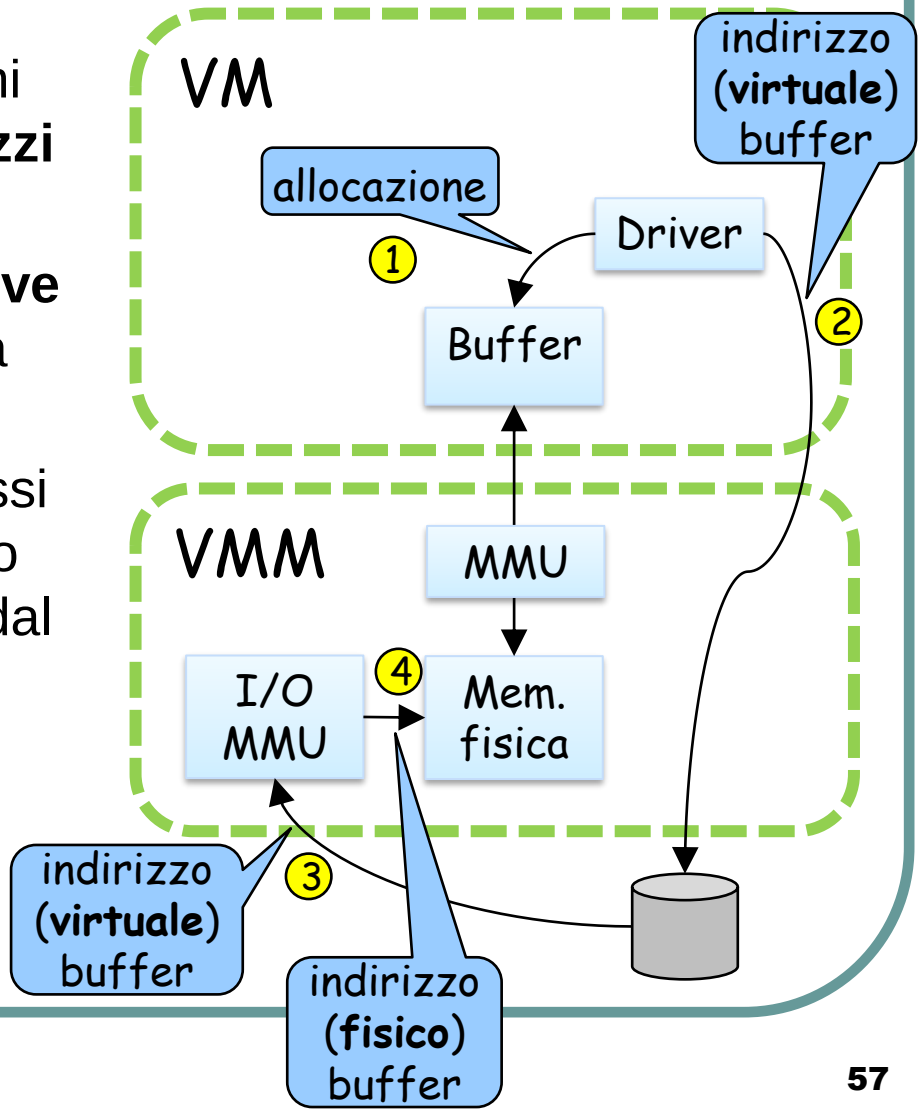
Device pass-through e I/O-MMU

- Il device pass-through assegna un dispositivo di I/O in **modo esclusivo** ad una VM, aumentando le prestazioni
- Il guest OS effettua **memory-mapped I/O** sul dispositivo



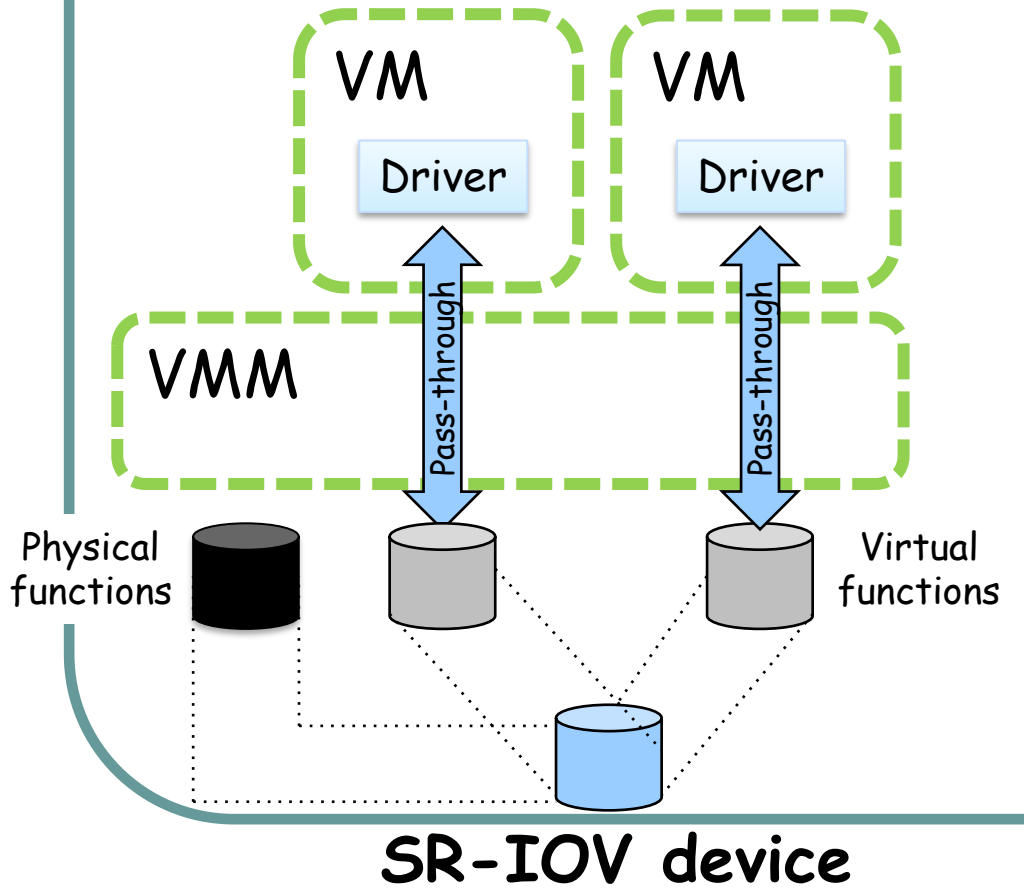
I/O-MMU

- Il guest OS effettua operazioni di I/O indicando i suoi **"indirizzi fisici"** al dispositivo
- Tuttavia, il dispositivo **non deve** usare gli indirizzi indicati dalla VM, ma quelli dal VMM
- La **I/O-MMU** traduce gli accessi di memoria fatti dal dispositivo verso gli indirizzi fisici decisi dal VMM
- La I/O-MMU fa **interrupt remapping** per inoltrare le interruzioni alla VM



Single Root I/O Virtualization (SR-IOV)

- In caso di numerose VM, **non è fattibile** assegnare in modo esclusivo un dispositivo ad ogni VM



- La **SR-IOV** (supportata dall'hardware **PCIe**) permette a più VM di **condividere il dispositivo** senza passare per il VMM
- Il controller fornisce più **interfacce virtuali** (virtual functions) distinte, con propri registri, DMA, interrupt
- Il controller condivide le risorse fisiche del dispositivo tra le interfacce virtuali

LE TECNOLOGIE VMWARE

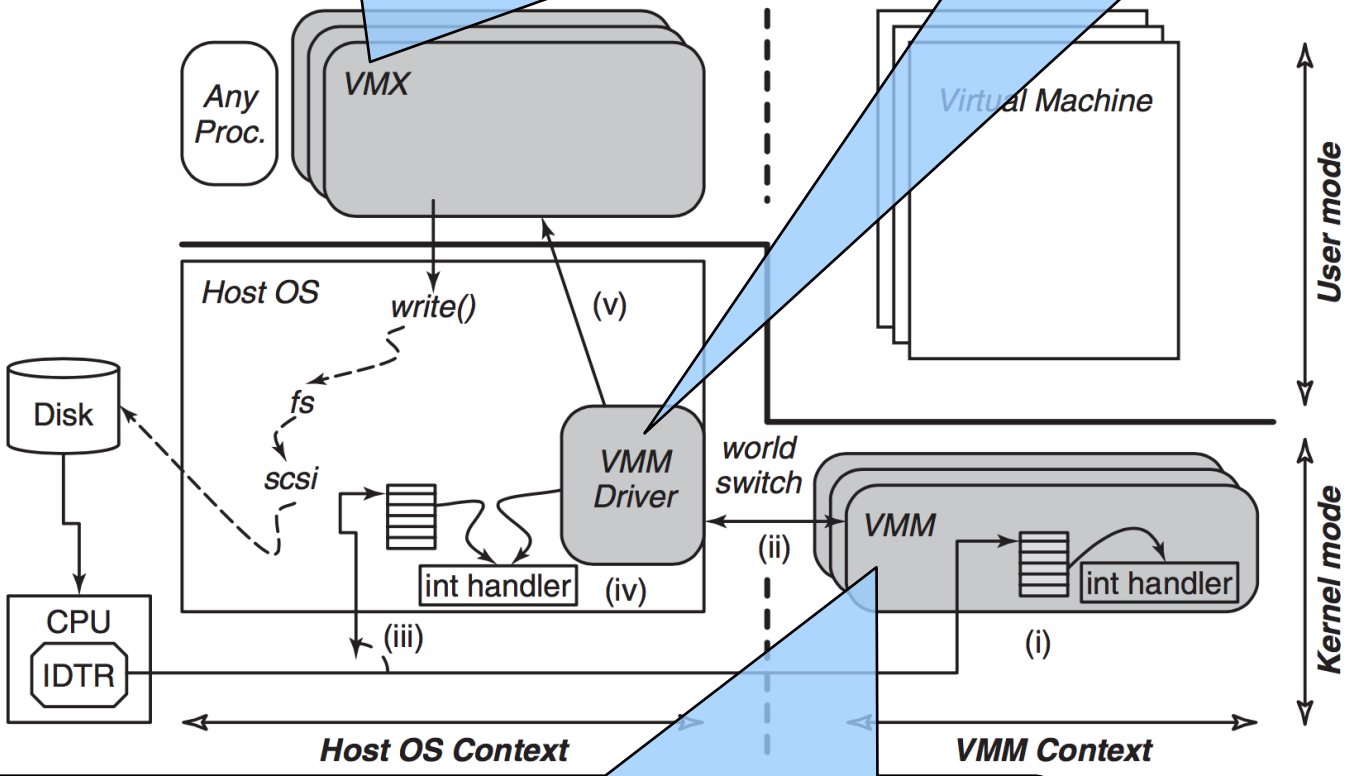
VMware Workstation

- La prima soluzione per piattaforme Intel x86
- Un hypervisor type-2 in grado di eseguire su Linux e Windows
- Full virtualization con dynamic binary translation e shadow page tables
- Si avvale dello **host OS** per accedere all'I/O, emulando semplici dispositivi legacy

VMware Hosted Architecture

- Programma user-space "percepito" dall'utente (un VMX per VM).
- Avvia la VM, emula la maggior parte dei device, effettua syscall sull'host OS

- Modulo kernel dell'host OS
- Permette al VMM di eseguire, sospendendo temporaneamente l'host OS



- Implementa i meccanismi di virtualizzazione (exception handlers, DBT, shadow p.t., ...)

Virtual Hardware Platform

	Virtual Hardware (front end)	*Back end*
Multiplexed	1 virtual x86 CPU, with the same instruction set extensions as the underlying hardware CUP	Scheduled by the host operating system on either a uniprocessor or multiprocessor host
	Up to 512 MB of contiguous DRAM	Allocated and managed by the host OS (page-by-page)

Emulated	PCI Bus	Fully emulated compliant PCI bus
	4x IDE disks 7x Buslogic SCSI Disks	Virtual disks (stored as files) or direct access to a given raw device
	1x IDE CD-ROM	ISO image or emulated access to the real CD-ROM
	2x 1.44 MB floppy drives	Physical floppy or floppy image
	1x VMware graphics card with VGA and SVGA support	Ran in a window and in full-screen mode. SVGA required VMware SVGA guest driver
	2x serial ports COM1 and COM2	Connect to host serial port or a file
	1x printer (LPT)	Can connect to host LPT port
	1x keyboard (104-key)	Fully emulated; keycode events are generated when they are received by the VMware application
	1x PS-2 mouse	Same as keyboard
	3x AMD Lance Ethernet cards	Bridge mode and host-only modes
	1x Soundblaster	Fully emulated

VMware ESXi

- VMware ESXi è un **hypervisor bare metal**
- Riusa il VMM da Workstation, l'host OS è sostituito dal **VMkernel**
- CPU e memory mgmt. ottimizzati per scalabilità e basso overhead
- Funzioni avanzate di migrazione, network/storage virtualization, ...