



Virtualización de los recursos HW v.20231017

Arquitecturas Virtuales

Depto. de Arquitectura de Computadores
Universidad de Málaga

© 2013-23 Guillermo Pérez Trabado, Eladio Gutiérrez Carrasco, Julián Ramos Cózar



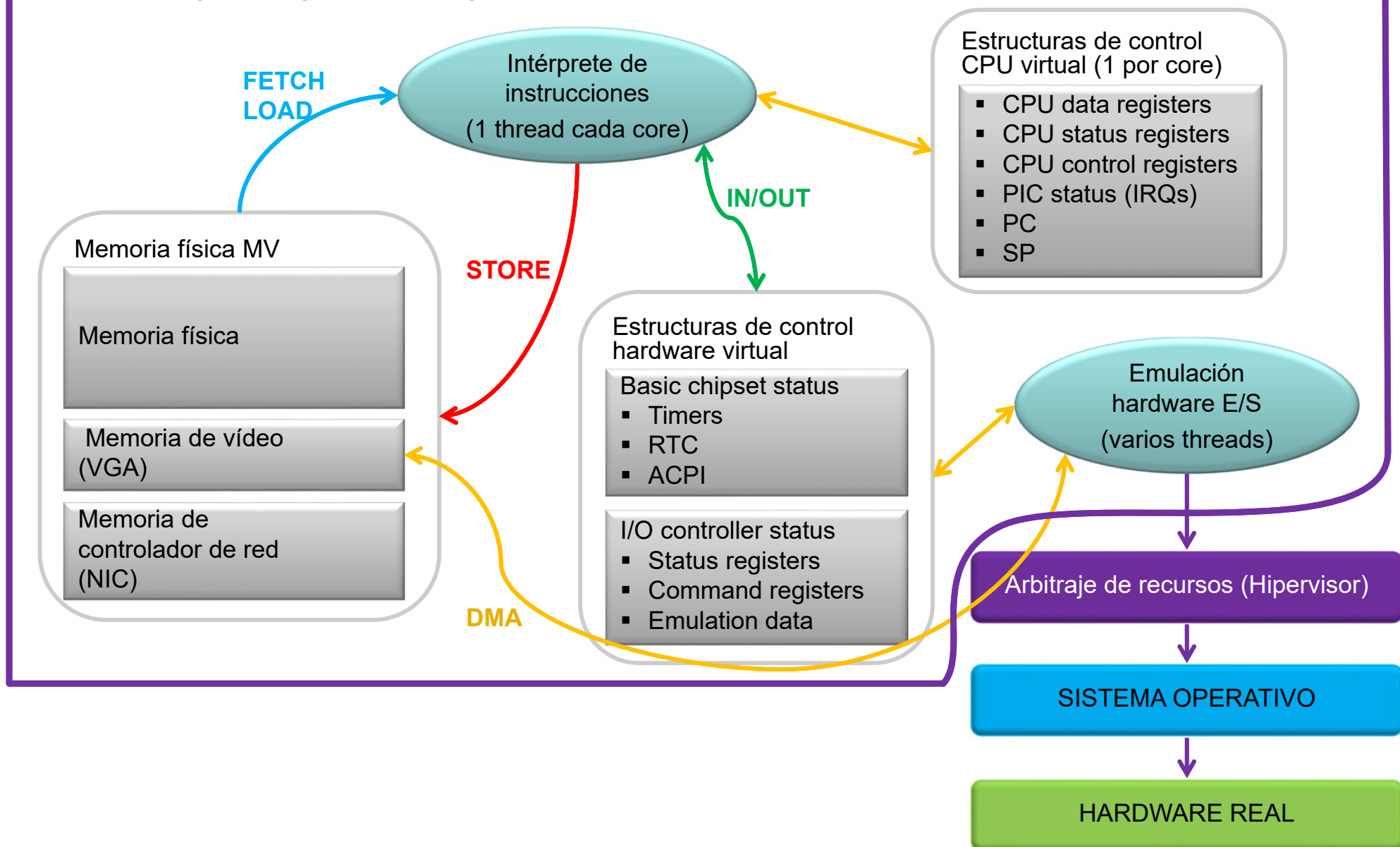
Implementación del Hypervisor

● ● ● | Implementación Software

- **Emulación** (CPU real y virtual distintas)
 - El emulador es un programa intérprete que decodifica y simula la ejecución una a una las instrucciones de la CPU virtual
- **Virtualización nativa** (CPU real y virtual idénticas)
 - **Traducción binaria** en tiempo de ejecución
 - Las instrucciones de la CPU virtual se analizan (precompilan) y se traduce a un código seguro para la CPU real sin instrucciones privilegiadas y que incluye código de emulación por software del hardware de E/S virtual.
 - La traducción lleva cierto tiempo adicional, aunque el código traducido puede ejecutarse indefinidamente.
 - **Trap & Emulate**
 - Se deja ejecutar a la CPU real el código de la CPU virtual pero en modo no privilegiado. Cuando usa instrucciones privilegiadas genera excepciones en la CPU real que debe emular dichas operaciones por software.
 - Más rápido
 - El modo privilegiado de la CPU virtual es emulado por software para que el kernel virtual no tenga privilegios sobre el hardware real.

Implementación hipervisor

Proceso del Hipervisor para cada máquina virtual



● ● ● | Intérprete de CPU

```
function core_emulation(core_status status, BYTE *mem)
{
    PC=CPU_RESET_ADDRESS;
    while (true)
    {
        /** Emulate jump to vector for external IRQ */
        if (status.pic_status.IRQ_pending!=0 && status.status_flags.IRQEnabled=
        {
            status.status_flags.priv_mode=true;
            ...save status flags and PC in Stack...
            PC=status.IRQ_vectors[current_IRQ];
        }
        else /* emulate next instruction at PC */
        {
            opcode=mem[PC];
            switch (opcode)
            {
                case ADD: ... emulate add..
                case SUB: ... emulate sub..

                /* Emulate enter PRIV mode through SYSCALL (TRAP) */
                case TRAP:
                    mem[--status.SP] = status.PC;
                    mem[--status.SP] = status.status_flags;
                    status.status_flags.priv_mode=true;
                    PC=status.IRQ_vectors[opcode.argument];
                /* Emulate exit fromPRIV mode through RTI */
                case RTI:
                    if (status.status_flags.priv_mode==false)
                        ...jump to PRIVILEGED_OPCODE exception...
                    else
                    {
                        status.PC=mem[status.SP++]
                        status.status_flags=mem[status.SP++];
                    }
                /* Emulate dialog with I/O controllers */
                case IN | OUT:
                    if (status.status_flags.priv_mode==false)
                        ...jump to PRIVILEGED_OPCODE exception...
                    else
                    {
                        get I/O port from instruction and emulate that port of corresponding controller
                    }
            }
        }
    }
}
```

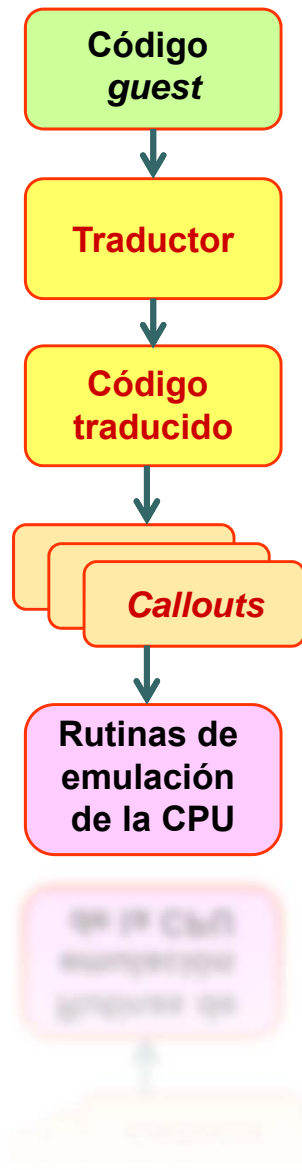
Data control structures

```
struct VM_control_data {
    Core_status    cores[MAX_CORES];
    PhyMem_status  mem_status;
};

struct Core_status {
    Core_data_registers      data_registers;
    Core_control_registers   control_registers;
    Core_status_flags        status_flags;
    Core_PIC_status          pic_status;
    register                 PC;
    register                 SP;
};

struct Core_status_flags {
    bool    Priv_mode;
    bool    Overflow;
    bool    Zero;
    bool    Carry;
    ...
};
```

Runtime Binary Translation



- El hipervisor revisa el código de la máquina virtual buscando las instrucciones privilegiadas y las sustituye por código de emulación y llamadas a librerías de simulación de hardware virtual en su lugar.
- El código resultante se ejecuta en modo no privilegiado simulando por software el modo privilegiado de la CPU virtual.
- Instrucciones de acceso a Entrada/Salida (IN/OUT):
 - Se sustituyen por llamadas a rutinas que realizan la E/S mediante llamadas al sistema del S.O. subyacente.

Runtime Binary Translation

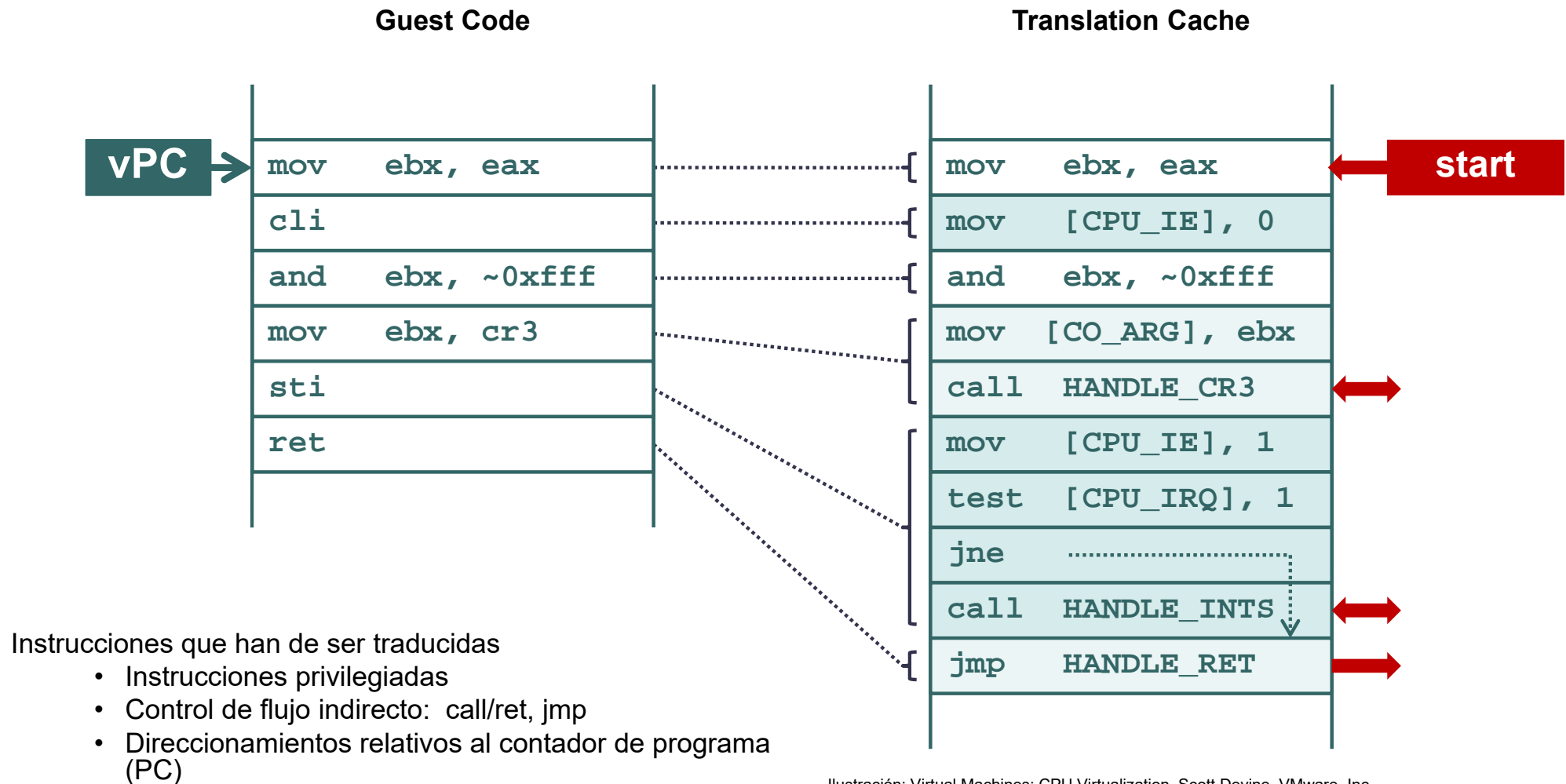


Ilustración: Virtual Machines: CPU Virtualization. Scott Devine. VMware, Inc.

● ● ● | Algunas cuestiones a considerar con la traducción binaria

- Códigos que se **auto-modifican**
- Códigos que **auto-referencian** (cargan datos de la zona de código)
- **Valor del estado (incluyendo PC) a preservar si hay interrupciones durante la ejecución de código traducido**

● ● ● | *Trap & Emulate*

- El S.O. de la MV se “deprivilegia” (*guest OS de-privileging*)

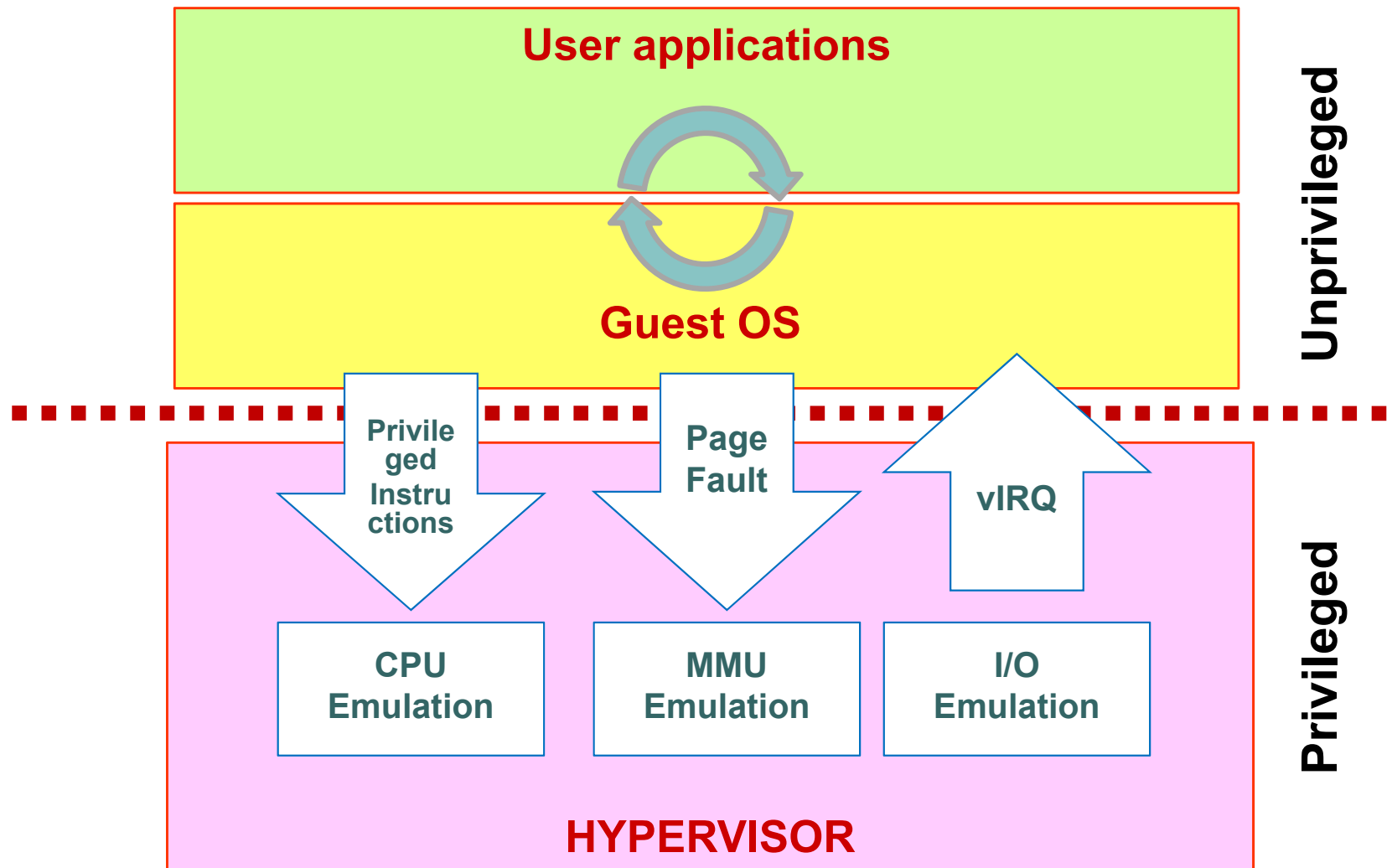


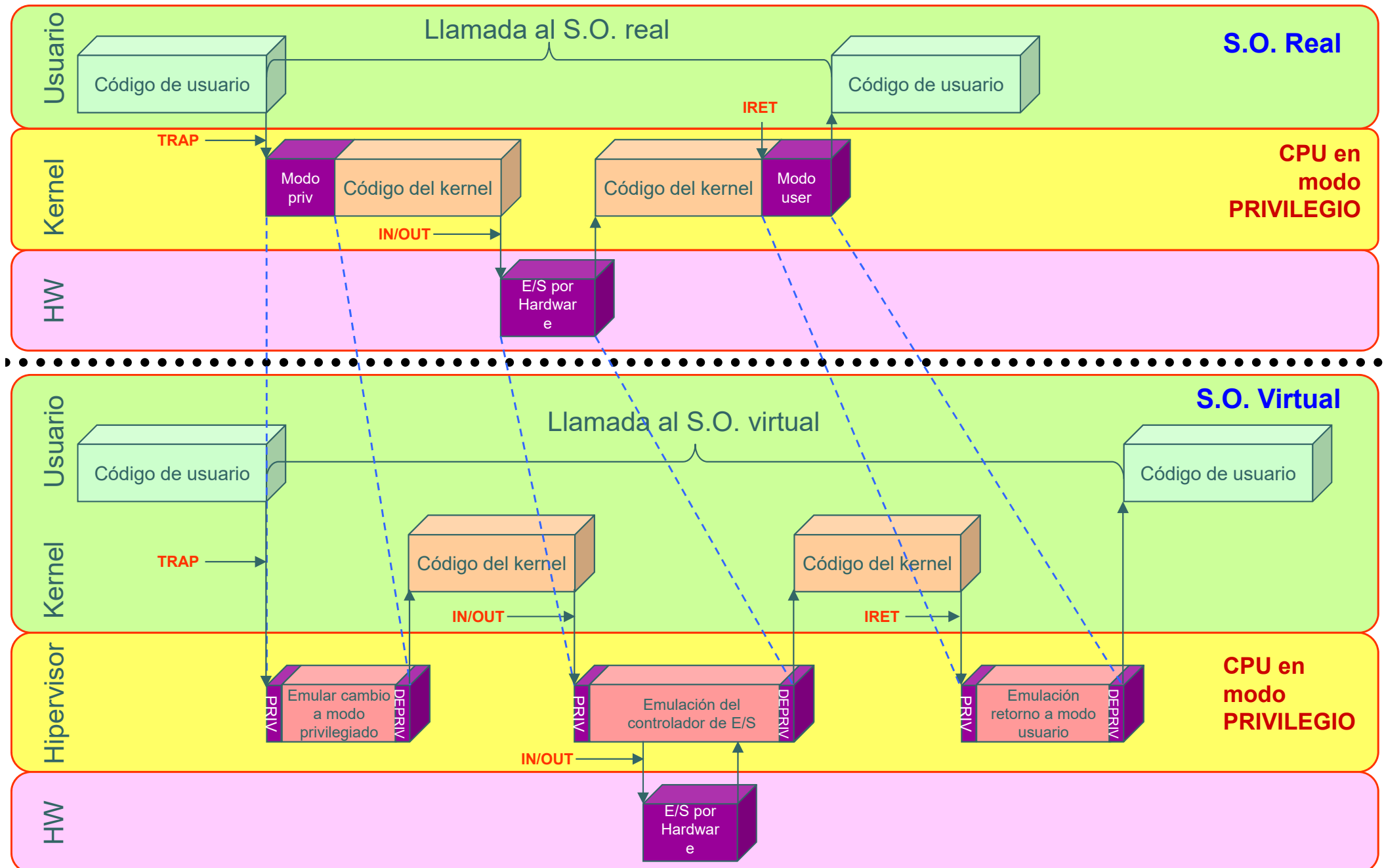
Ilustración: Virtual Machines: CPU Virtualization. Scott Devine. VMware, Inc.



Trap & Emulate

- Todas las operaciones privilegiadas generan excepciones que saltan al código del hipervisor
 - La máquina virtual es un proceso ejecutado en modo usuario del procesador.
 - La ejecución de una instrucción privilegiada genera una excepción por falta de privilegio que llama a una subrutina en el hipervisor que simula el efecto de la instrucción que ha fallado.
 - Entrada/Salida: Operar sobre los registros de los controladores con IN/OUT provoca excepciones que llaman a rutinas del hipervisor que simula el funcionamiento de dichos controladores.





● ● ● | *Trap & Emulate*

○ Desventajas

- Hay procesadores con instrucciones con efectos distintos en modos usuario y supervisor (no se genera un TRAP en modo usuario).
 - CPUs NO virtualizables en sentido clásico, ej. Intel x86 (IA32)
- El proceso de excepciones puede añadir muchísimo *overhead*.
- El hipervisor debe emular por software todo el estado de CPUs y chipset reservado al modo privilegiado mediante estructuras de datos ocultas (*shadow registers*):
 - Registros de control
 - MMU
 - Controlador de interrupciones (PIC), Timers, Real Time Clock
 - Tabla de vectores de interrupción



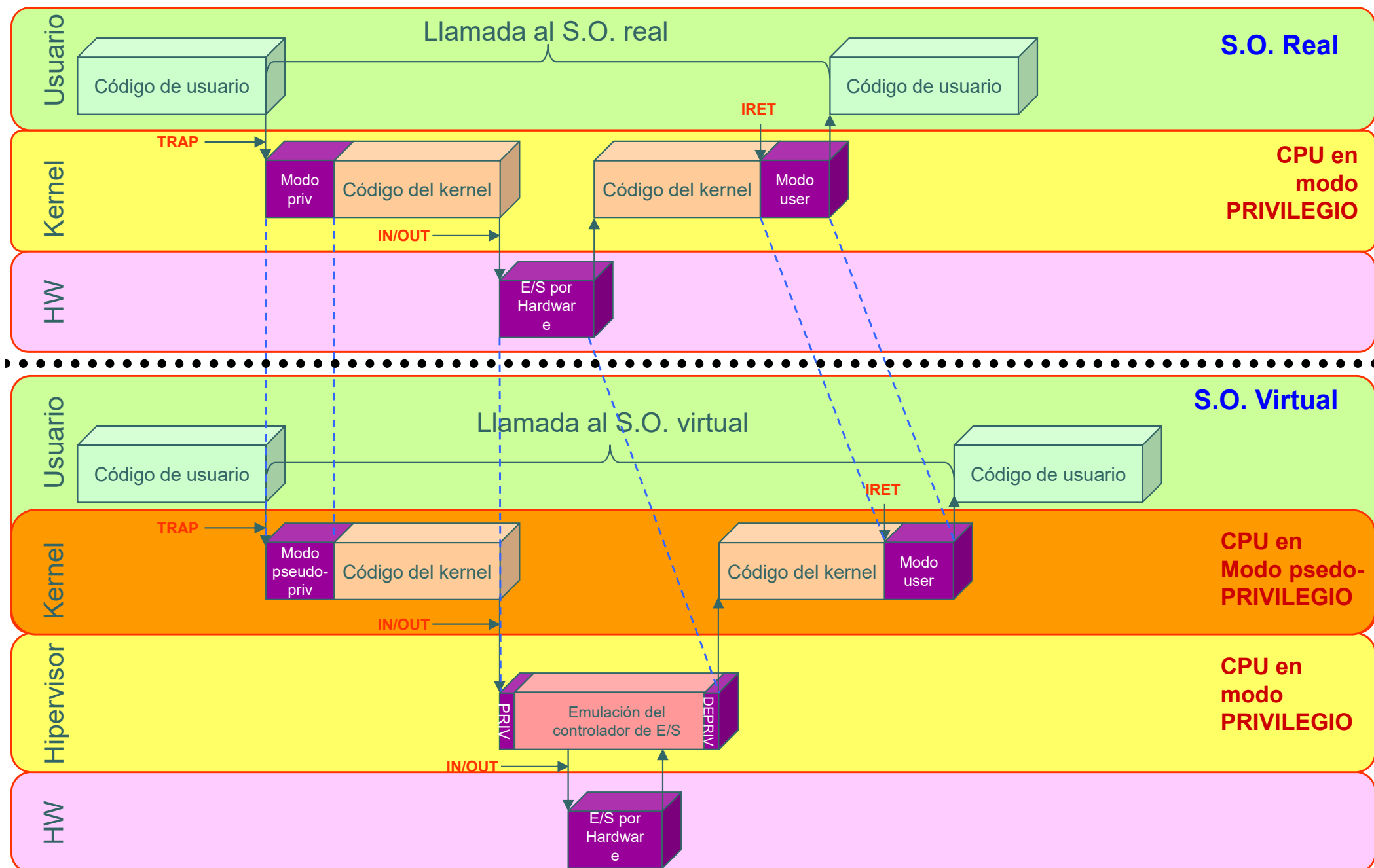
Implementación Hardware

- Virtualización nativa
 - Ejecución directa del código sobre la CPU.
- Se añade un nuevo modo pseudo-supervisor que simula la CPU virtual en modo privilegiado pero sin privilegios sobre el hardware real.
 - Las instrucciones IN/OUT para acceder a los registros de los controladores generan excepciones que llaman a rutinas en el hipervisor para simular los controladores.
- La ventaja fundamental es no tener que simular el paso a modo privilegiado dentro de la máquina virtual.





Virtualización por Hardware





Implementación de los recursos hardware virtuales

CPU, RAM, Red / Almacenamiento / Entrada/Salida
Checkpoints, Snapshots y Clones

● ● ● | Virtualización de la CPU

- Simulación de la CPU virtual mediante técnicas software o soporte hardware
 - Soporte hardware del modo virtual (Intel VT-x / AMD-V)
 - Trap & Emulate
 - En todos los casos se simula un modo privilegiado para la MV y otro para el hipervisor.
- Visibilidad de la CPU
 - El hipervisor puede simular un modelo de CPU distinto a la real.
 - Requiere simular registros de control (*shadow registers*) para ocultar los de verdad.
 - Simulación de una CPU más antigua:
 - Ofrece un modelo más sencillo para garantizar la portabilidad.
 - Evita que fallen programas antiguos por incompatibilidad con las nuevas CPUs.
 - Visibilidad total de la CPU real:
 - El software puede detectar la presencia de instrucciones avanzadas para aumentar la velocidad de proceso.
 - Se aprovechan todas las mejoras de cada CPU.
 - El software debe ser compatible con el modelo exacto de CPU presente.



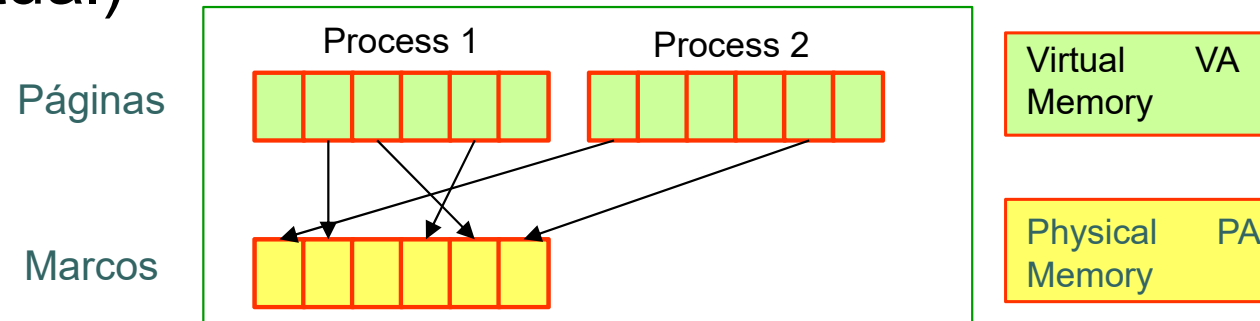
Multiplexación de la CPU entre MVs

- Muchos servidores suelen estar inactivos casi todo el tiempo.
- El hipervisor aplica una planificación de la CPU tipo *round-robin* entre máquinas virtuales.
 - El S.O. invitado detiene la CPU cuando no se usa.
 - El hipervisor detecta la instrucción HALT y aprovecha para pasar el control a otra máquina virtual.
 - Si una MV no bloquea la CPU dentro de un cierto tiempo, el hipervisor interrumpe su ejecución para pasar el control a otra máquina.
- En los sistemas virtuales se observan tiempos de espera inexplicables cuando el hipervisor planifica otra máquina virtual.
- El hipervisor puede definir políticas para controlar el reparto de tiempo de CPU entre máquinas virtuales.



Virtualización de la RAM

- Sin virtualización (soporte nativo de memoria virtual)



- Con virtualización

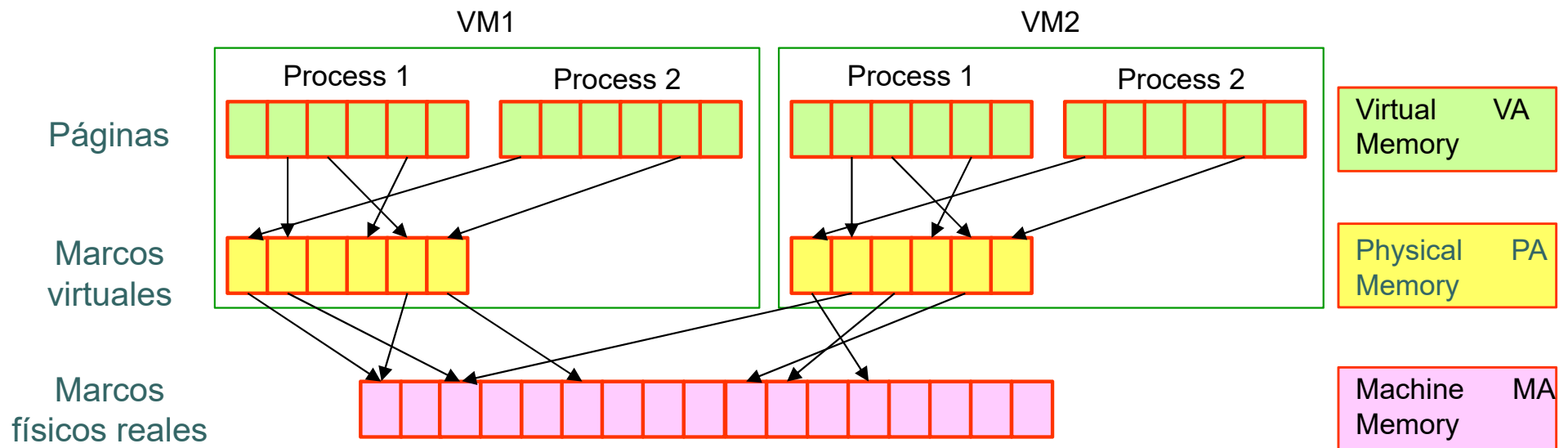


Ilustración: VMware and hardware assist Technology (Intel VT and AMD-V), Jack Lo, VMWORLD06



Virtualización de la RAM

- La máquina virtual requiere la simulación de dos recursos hardware
 - Su espacio físico de memoria RAM
 - La unidad de gestión de memoria (MMU) del procesador
- El hipervisor requiere controlar el uso de la memoria física por parte de las máquinas virtuales
 - Control de acceso (permisos, validez, presencia, *swapping*)
 - Traducción por páginas (libertad para asignar memoria no continua a cada máquina virtual y no asignar las páginas no usadas)
- Son necesarias dos MMUs:
 - Una MMU privada a cada máquina virtual
 - Una MMU real en el hipervisor.

● ● ● | Paginación virtual

- La simulación introduce **dos niveles de traducción**
 - Dentro de una máquina virtual
 - Espacio Lógico Virtual (ELV) -> Espacio Físico Virtual (EFV) (S.O. Invitado)
 - En el hipervisor
 - Espacio Físico Virtual (EFV) -> Espacio Físico Real (EFR) (Hipervisor)
- La MMU virtual se implementa mediante:
 - **Simulación software (*Shadow Page Tables*)**
 - Se interceptan las instrucciones que programan la MMU virtual y se programa la MMU física con la traducción directa ELV→EFR
 - Requiere interceptar todo el acceso del S.O. invitado a la MMU emulada y guardar dos versiones de la TP (la del S.O. invitado –solo para interceptar su gestión- y la del hipervisor –en uso en la MMU-).
 - Overhead de memoria y de proceso.
 - **Soporte Hardware (*Nested Paging / Extended Page Table*)**
 - La MMU convencional traduce de ELV a EFV.
 - Un segundo nivel hardware de la MMU traduce las referencias EFV a EFR.
 - El S.O. invitado programa las tablas del primer nivel. El hipervisor gestiona las tablas del segundo nivel de traducción.
 - La TLB hace caching de la traducción final de ELV a EFR.

● ● ● | *Shadow Page Tables*

- Mantener un segundo set de tablas de páginas en el hipervisor
- Este set está oculto (*shadow*) a las MVs
- Detectar los cambios en las TP del S.O. *guest*:
 - Modificación del PTBR (context switch), invalidaciones de la TLB, creación de páginas, modificación de las TP, ...

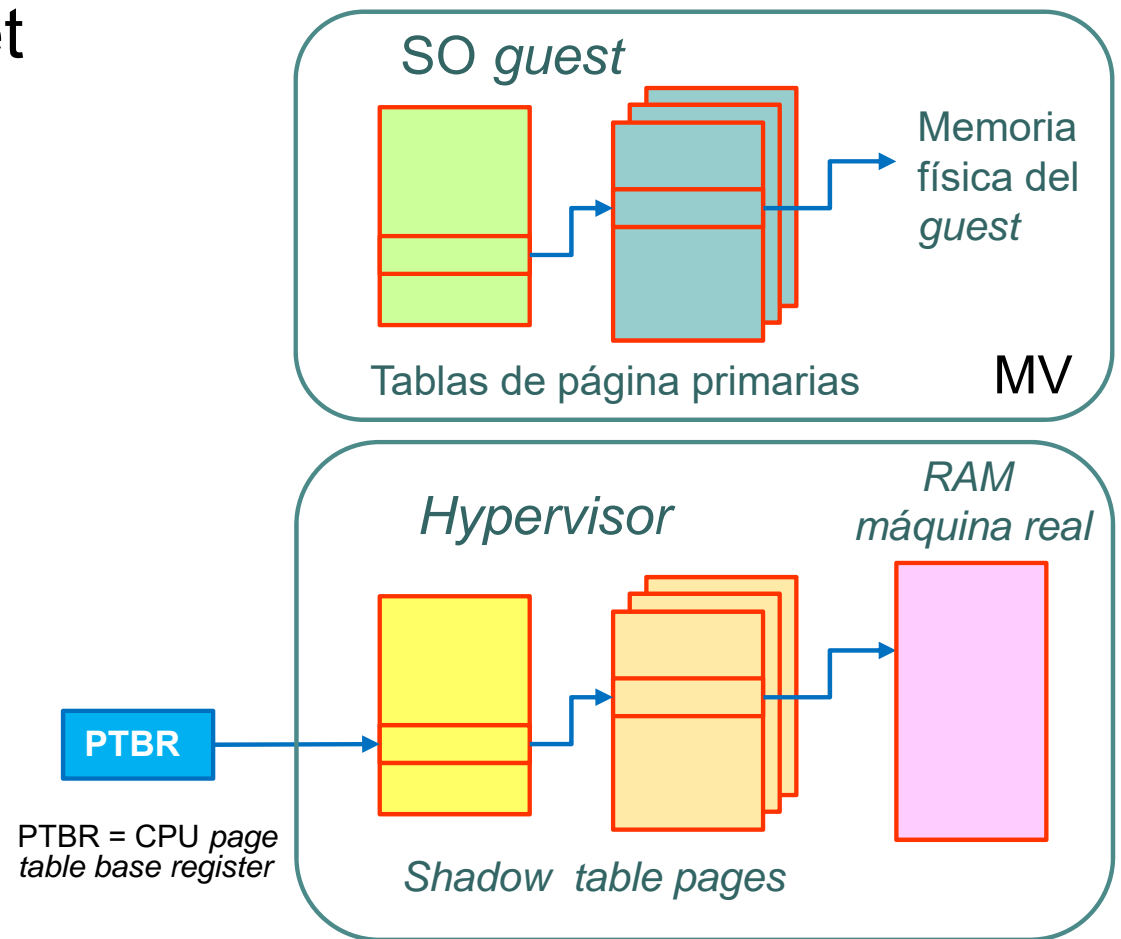
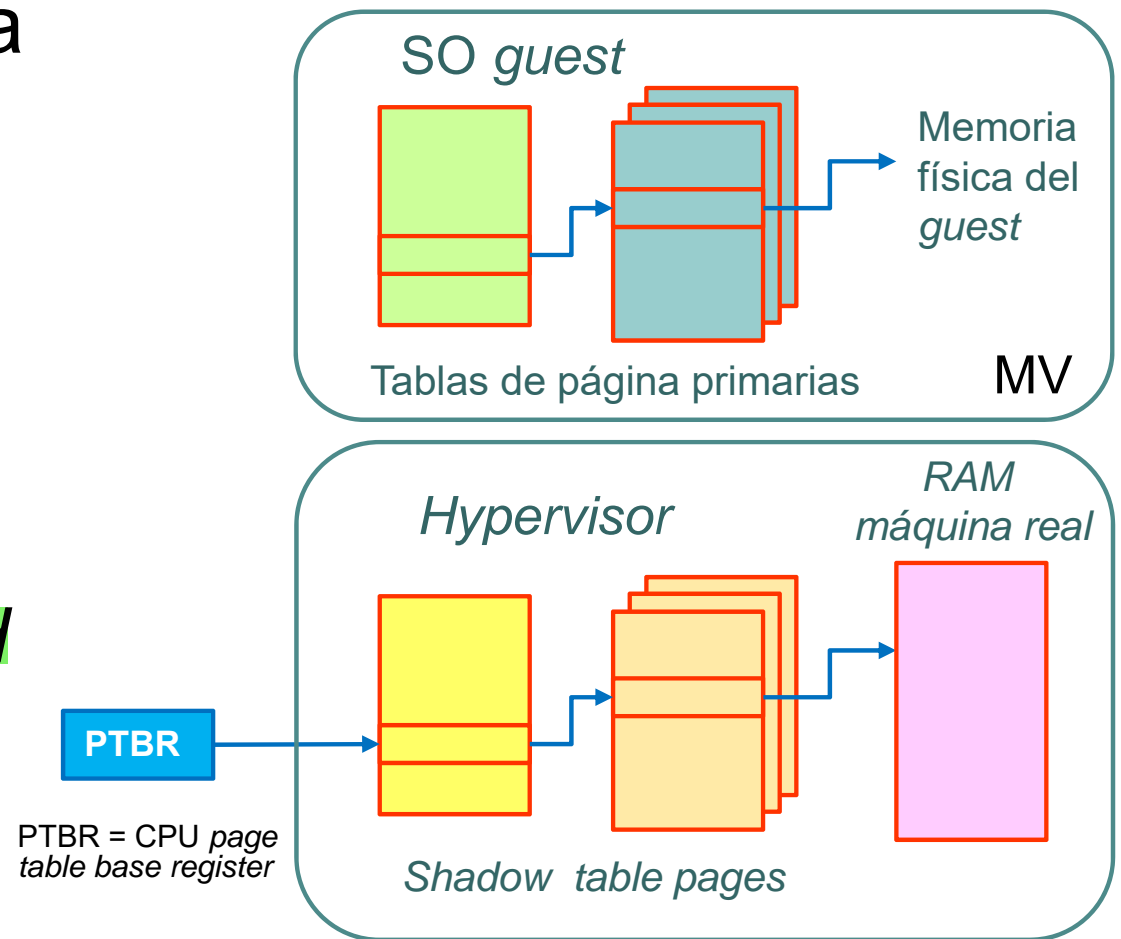


Ilustración basada en otras de
<http://www.cs.toronto.edu/~delara/courses/ece1779/handouts/Virtualization.pdf>
<https://corensic.wordpress.com/2011/12/05/virtual-machines-virtualizing-virtual-memory/>

● ● ● | *Shadow Page Tables*

- El hipervisor actualiza las *shadow page tables* de forma adecuada
- Ventajas: el *guest* no requiere modificación
- Desventajas: *overhead* de memoria y procesamiento



Nested Paging / Extended Page Tables

- Soporte HW
 - AMD: *Nested paging* (NPT), (también denominado *Rapid Virtualization Indexing* (RVI))
 - Intel: *Extended Page Tables* (EPT)
 - Ambos conceptualmente análogos
- Dos juegos de tablas de páginas
 - Uno ELV -> EFV (guest) y otro EFV->EFR (hipervisor)
 - Dos punteros PTBR: *guest* y *nested* (ej. CR3, nCR3)
 - La MMU tiene que recorrer estos dos niveles de tablas

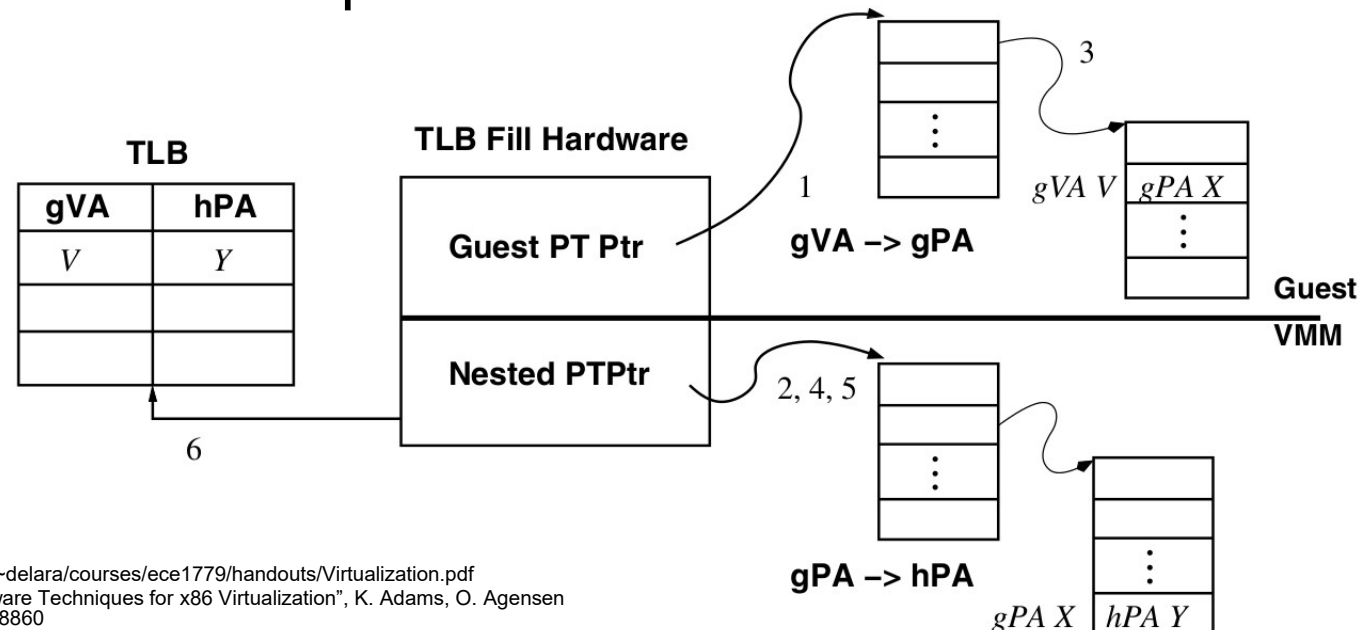


Ilustración: <http://www.cs.toronto.edu/~delara/courses/ece1779/handouts/Virtualization.pdf>
"A Comparison of Software and Hardware Techniques for x86 Virtualization", K. Adams, O. Agensen
<http://dx.doi.org/10.1145/1168857.1168860>

Nested Paging / Extended Page Tables

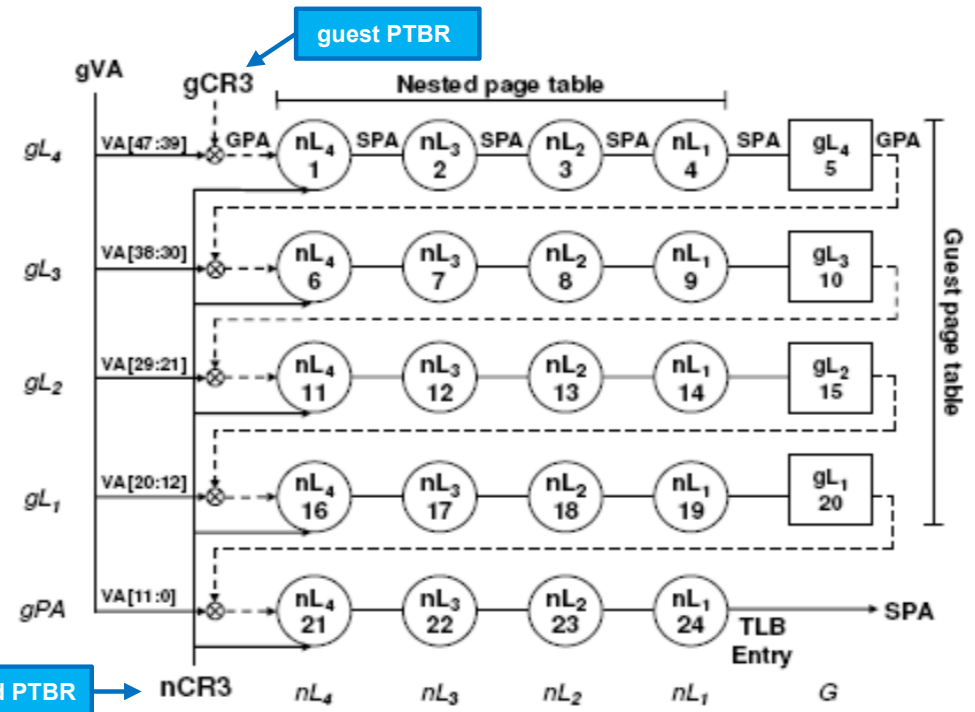
- Todo acceso por parte de la MV a memoria física requiere la traducción anidada:
 - Esto incluye los accesos a marcos de páginas del *page walk* del *guest*

- Número de accesos total:

$$(G+1)*(N+1)$$

para un *guest page walk* con paginación de G niveles y un *nested page walk* de N niveles

Ilustración: AMD-V™ Nested Paging White Paper, Advanced Micro Devices, Inc., 2008.



- El tamaño de la tabla de páginas *nested* se reduce con respecto a la *shadow page table*
 - Sólo una tabla *nested* para cada MVs
- La TLB cachea la traducción final de ELV a EFR
 - La TLB es un factor crítico del rendimiento



Multiplexación de la RAM del hipervisor entre MVs

- Muchos sistemas solo acceden a una parte de su memoria RAM durante un periodo de tiempo (localidad).
 - El hipervisor controla la memoria en uso (accedida) dentro de cada máquina.
 - Inicialmente la máquina va creciendo conforme usa memoria RAM de sus supuestos bancos de RAM (asignación bajo demanda).
 - El hipervisor controla qué memoria continua usándose al cabo de un rato dentro de la MV (“working set” de la MV).
- La suma de RAM en uso de todas las MVs puede exceder la RAM del sistema físico (*overcommitting*)
 - Si se agota la RAM libre del hipervisor puede liberar memoria no usada por una máquina para concederla a otra.
 - El contenido de una página ya en uso se guarda en un fichero de swap privado a cada MV.
 - El swap de hipervisor es invisible al S.O. invitado (sigue creyendo que sus páginas están en RAM).
- El hipervisor permite definir políticas para controlar el reparto de la RAM entre máquinas virtuales.

● ● ● | Virtualización de Discos

- La unidad de simulación es el volumen virtual de disco (SCSI LUN).
 - El volumen es un conjunto de bloques de tamaño uniforme (direccionamiento LBA).
 - La geometría no es relevante hoy en día.
- Cada volumen virtual se puede implementar de forma transparente sobre:
 - Un volumen físico real (*by-pass*).
 - Un fichero en un sistema de ficheros del hipervisor.
- En todos los casos se ofrece el mismo modelo de controlador al sistema virtual:
 - Independencia total de la implementación.
 - El mismo driver en el S.O. virtual para todas las implementaciones.

● ● ● | Rendimiento de los discos

- Volúmenes físicos (by-pass)
 - Menos retardo adicional en cada operación de acceso.
 - Menos portabilidad (atados a un volumen real).
 - Se usa para aplicaciones críticas o para migrar un servidor existente.
- Discos virtuales implementados sobre ficheros:
 - Menos velocidad de transferencia y más latencia.
 - Más portabilidad y flexibilidad. Es posible mover la ubicación del volumen a un nuevo almacenamiento sin modificar la MV.
 - Es posible no asignar el espacio libre del volumen. Esto es una ventaja y un riesgo a la vez. Puede provocar fragmentación e inanición por falta de disco real.

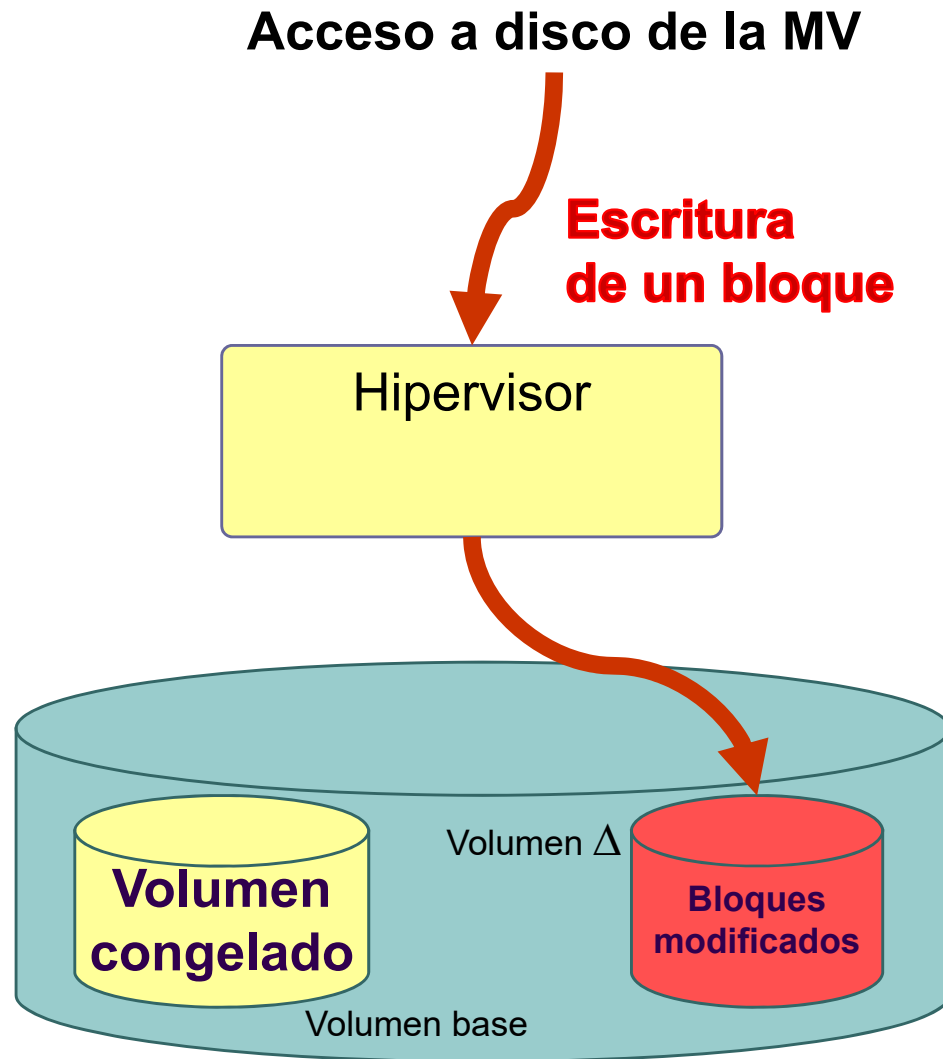
● ● ● | Snapshots

- Son estados “congelados” de un sistema informático y preservados en el tiempo para permitir la posibilidad de una “vuelta atrás” desde un estado posterior del sistema.
- Se puede congelar:
 - un dispositivo de almacenamiento individual
 - una máquina virtual completa con memoria y varios dispositivos de almacenamiento (*checkpoint*)

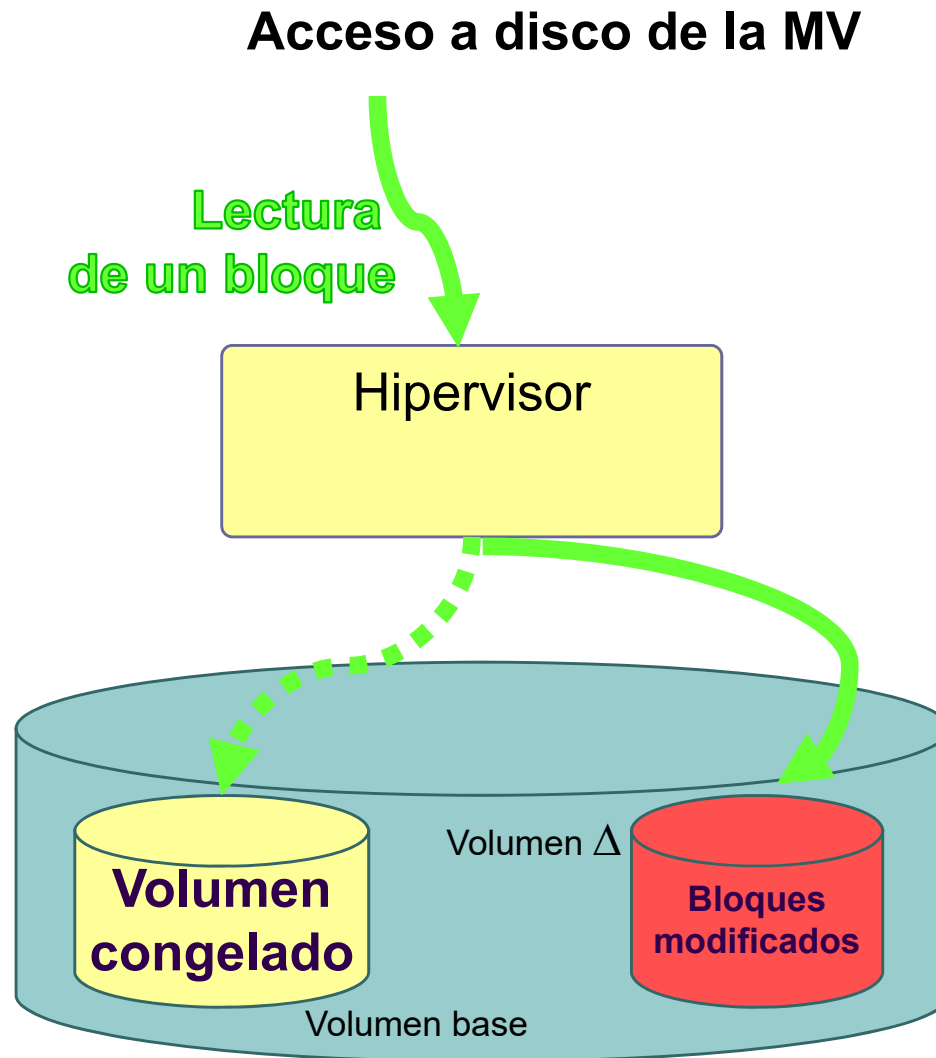
● ● ● | Snapshots de disco

- Permite preservar el estado de un disco virtual en un punto del tiempo.
- Implementación
 - Se bloquea el acceso de escritura al disco. Solo se permiten lecturas.
 - Se crea otro “volumen de snapshot” vacío que solo guarda los bloques modificados. Se permiten lecturas y escrituras.
 - Las escrituras se hacen en el “volumen de snapshot”.
 - Las lecturas van primero al “volumen de snapshot”. Si el bloque no está allí, se lee del volumen original.
- Disminuye la eficiencia
 - En escritura hay que gestionar la asignación de espacio para las modificaciones.
 - En lectura hay que consultar el índice de bloques modificados que contiene el snapshot.

● ● ● | Snapshots de disco



● ● ● | Snapshots de disco



● ● ● | *Snapshot* de una MV

- Recuerda el estado de una máquina completa:
 - Snapshot de todos sus discos +
contenido de la RAM +
configuración hardware
- Posibilidad de recuperar desastres de forma instantánea.
 - Se puede volver instantáneamente a un estado estable.
 - Desconfiguración, virus, updates inestables.
 - Deshacer pruebas tipo WHAT-IF?
- Implementación:
 - Se guarda el estado de la RAM si está encendida.
 - Se guarda un snapshot de disco



Operaciones para eliminar *snapshots*


- Deshacer (*rollback*)
 - Se vuelve al estado anterior. Es muy rápida.
 - Se carga el estado de RAM guardado como RAM actual de la MV.
 - Se elimina el fichero de snapshot con los cambios.
 - Se reanuda el acceso en escritura al disco original.

- Consolidar (*commit*)
 - Se continua en el estado actual pero funcionando sin snapshot.
 - Se elimina el estado de RAM guardado.
 - Se graban los bloques modificados sobre el disco original.
 - Se elimina el fichero de snapshot.
 - El tiempo necesario depende de la cantidad de información a consolidar.



Clones

- Son copias de máquinas virtuales ya existentes
 - Ahorran el tiempo de instalación y configuración de una nueva instancia de un S.O.
 - Los prototipos se pueden “bloquear” para convertirlos en una biblioteca de máquinas prefabricadas con un estado conocido del cual partir: el concepto se suele denominar comercialmente “plantilla” (“template”).



Tipos de clones

(según la implementación)

○ Full-clones

- Copia convencional de una máquina virtual
- Se crea un nuevo disco virtual mediante una copia completa del original

○ Linked-clones

- Es una máquina cuyo estado esta almacenado mediante técnicas “diferenciales”. Se guardan solo los cambios respecto a la plantilla.
- Se crea un nuevo disco virtual mediante un “snapshot” del disco de la plantilla. El snapshot solo almacena las escrituras o modificaciones que realiza la máquina “clon” durante su ejecución.



Linked-clones: ventajas y desventajas

- La creación de clones es casi instantánea al no copiar el almacenamiento.
- Reduce espacio de disco ocupado por cada nueva máquina.
- Mejora la concurrencia de E/S a disco si el hipervisor implementa una cache de disco para los bloques de los discos virtuales:
 - Reuso de las lecturas físicas entre máquinas virtuales
 - Aumenta el hit-ratio de la caché de disco del hipervisor al acceder al mismo bloque en varias máquinas
- Si varios clones arrancan a la vez, el uso de disco no se multiplica por el número de máquinas virtuales.
- No se puede actualizar la plantilla ya que habría que destruir los clones.

● ● ● | Uso de los linked-clones

- Los linked-clones son muy útiles para **escritorios virtuales**
 - Numerosas instancias idénticas a partir de una misma plantilla.
 - La vida de una máquina virtual es breve (una sesión de usuario de varias horas o días).
 - Los clones se destruyen cuando el usuario sale de la sesión.
 - En cada nueva sesión se usa una máquina virtual recién clonada.
- Se requiere una infraestructura de software (**broker** de MVs) para generar los clones “bajo demanda” cada vez que un usuario solicita abrir una sesión y destruir los que están inactivos. Ejemplos:
 - Leostream (<http://www.leostream.com>)
 - Orientado a entornos de negocios
 - vCenter Lab (<http://www.vmware.com/products/labmanager>)
 - Orientado a entornos virtuales de desarrollo de software

● ● ● | Virtualización de la red:

- Requiere simular
 - Uno o más interfaces de red (NIC) en cada máquina virtual
 - Una o más LANs virtuales (switches Ethernet) simulando el transporte de los paquetes bien entre máquinas virtuales o bien entre la LAN real y las LANs virtuales mediante los NICs reales del sistema host.

● ● ● | NICs virtuales

- Cada NIC simula un interfaz de red con su propia MAC única.
- Para cada NIC virtual el administrador especifica a qué LAN virtual se conecta.
- Se simula el interfaz de control (registros de control y estado) de NICs muy populares para que el sistema operativo invitado use un driver estándar (Intel e1000, PCnet).
- Algunos hipervisores simulan NICs virtuales optimizados para alto rendimiento que requieren un driver especial (p.e. vmnetII, vmnetIII en VMWare).

● ● ● | LANs virtuales en desktops

- El hipervisor simula 1 o más LANs virtuales.
- Cada LAN virtual puede estar asociada a un NIC real del host de varias formas:
 - Bridged: el NIC funciona como un bridge Ethernet (nivel 2) entre la LAN virtual y la LAN externa. Las MVs tienen visibilidad total a nivel 2 de la LAN externa.
 - NAT: se interpone un router IP con NAT entre la LAN virtual y la LAN externa usando como dirección externa la IP del Host real. Las MVs tienen visibilidad a nivel IP de la LAN externa con las limitaciones inherentes a NAT.
 - Host-Only: La LAN no está asociada a ningún NIC (aislada del exterior). Sirve para comunicar MVs entre sí.

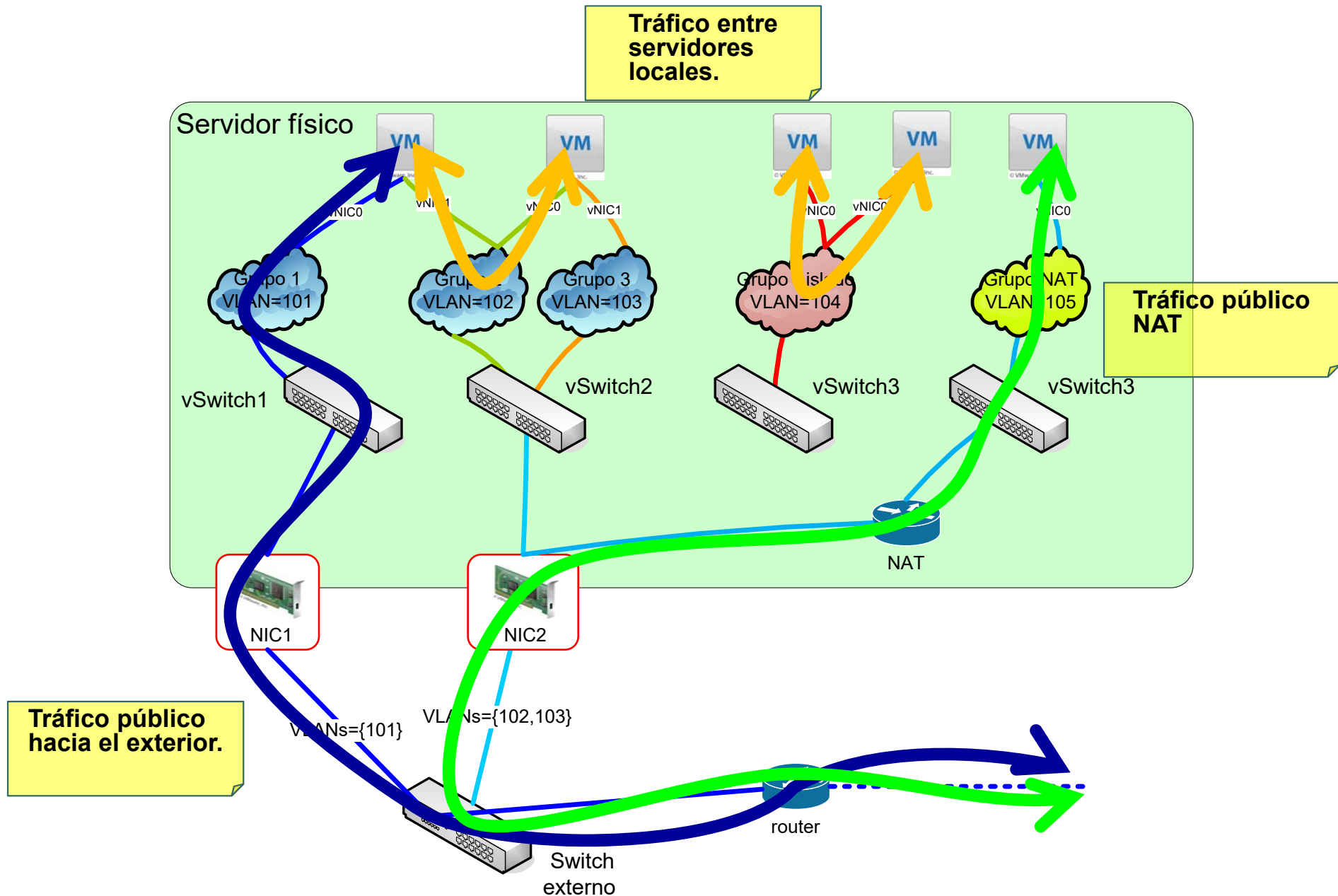
● ● ● | LANs virtuales en servidores

- El hipervisor simula uno o más **switches virtuales** Ethernet.
 - Un vSwitch tiene asociados 0 o más NICs reales.
 - Si tiene 0 está aislado del exterior.
 - Si tiene más de 1, se pueden usar simultáneamente con políticas de balanceo de carga o de fail-over.
 - Cada NIC real aparenta un enlace Ethernet punto a punto entre el switch virtual y un switch externo.
 - VLANs
 - Dentro de cada vSwitch se definen 1 o más VLANs (grupos de puertos).
 - Una MV puede tener 1 o más NICs virtuales conectados a distintos grupos de puertos.
 - A cada grupo de puertos se le asocia un tag de VLAN 802.1q.
 - Los NICs reales multiplexan el enlace trabajando en modo trunk 802.1q (identificando la VLAN o grupos de cada paquete mediante el identificador o tag 802.1q en la cabecera del paquete).
 - Los switches externos deben soportar también la multiplexación IEEE802.1q en los puertos a los que están conectados los NICs reales del hipervisor.

● ● ● | Implementación de LANs

- El paso de paquetes Ethernet entre NICs de máquinas virtuales conectadas al mismo switch virtual se hace mediante buffers de memoria del hipervisor.
 - La velocidad de paso de datos es muy elevada.
- Las comunicaciones con el switch exterior se hacen compartiendo el NIC exterior.
 - Cada MV usa su propia MAC para emitir.
 - La recepción se hace en modo promiscuo. Los paquetes recibidos se entregan a NIC con la MAC correspondiente.

LANs virtuales en servidores





Avances en el soporte de virtualización de las CPUs

- El soporte básico ayuda con la simulación del comportamiento y registros de la CPU de la MV en modo pseudo-supervisor.
- La traducción de la MMU se puede hacer programando hábilmente el hardware de paginación, pero existen agujeros de seguridad.
 - Las CPUs actuales también aportan un modo virtual para el hardware de paginación que no tiene permisos para acceder fuera de la memoria de la MV.
- El acceso a los dispositivos PCI se hace interceptando las instrucciones de acceso a los registros de control (IN/OUT).
 - Algunas CPUs recientes incorporan dispositivos PCI virtuales que pueden ser simulación por software o bien un acceso controlado a un dispositivo PCI verdadero.



Resumen

- La simulación de una arquitectura requiere de un mecanismo para interceptar los accesos al hardware (instrucciones privilegiadas) y de un sistema software para simular todos los recursos virtuales.
- El hardware de virtualización solo ayuda a mejorar el rendimiento interceptando instrucciones privilegiadas.
- El rendimiento del hardware virtual depende de la implementación concreta de cada periférico y de las optimizaciones aplicadas.