

Sistemas Operativos

Virtualización



Sistemas Operativos

- ❑ Versión: Abril 2025
- ❑ Palabras Claves: Virtualización, Emulación, Paravirtualización, Traducción Binaria



Que es virtualizar?

- Es una técnica que permite realizar “abstracción” de recursos de la computadora. Hoy en día también se virtualizan las redes, storages, data centers, etc...
- Es una capa abstracta que desacopla el hardware físico del sistema.
- Permite ocultar detalles técnicos a través de la encapsulación.
- Permite, entre otras cosas, que una computadora pueda realizar el trabajo de varias, a través de la compartición de recursos de un único dispositivo de hardware

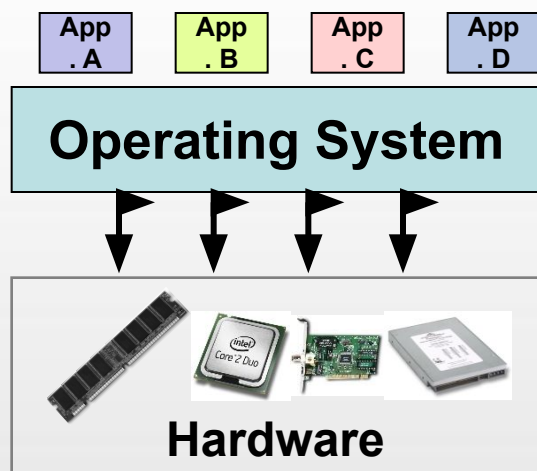


Tipos de virtualización

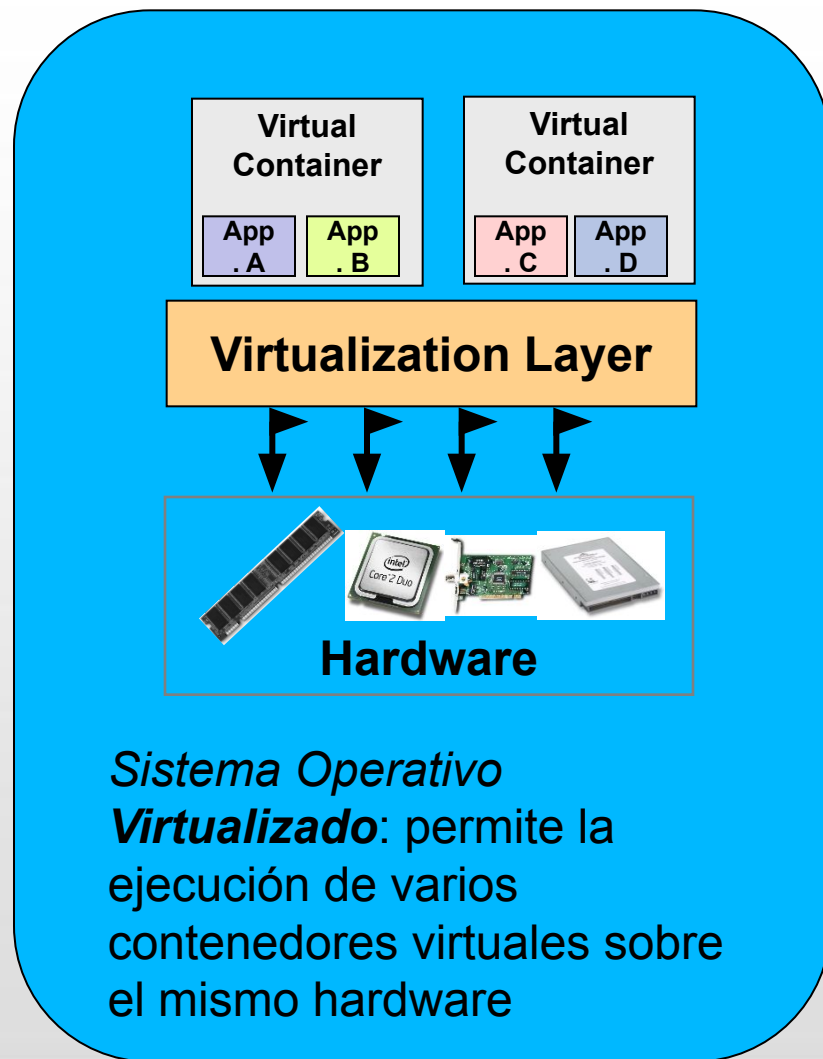
- Process Level: permite lograr portabilidad entre diferentes sistemas. Java Virtual Machine (JVM)
- Storage Level: presente una vista lógica del almacenamiento al usuario, quien no sabe donde están los datos guardados (RAID, LVM, etc.)
- Network Level: integra recursos de hardware de red con recursos de software
- Operating System Level: SO permite la existencia de varias instancias de espacio de usuario aisladas (containers)
- System Level: permite la creación de máquinas virtuales



¿Que es virtualizar?



*Sistema Operativo **No Virtualizado***: un único SO controla todos los recursos de hardware



*Sistema Operativo **Virtualizado***: permite la ejecución de varios contenedores virtuales sobre el mismo hardware



Ejemplo: CMS y VM/370

- VM/CMS es un sistema operativo orientado a máquinas virtuales que se lanzó en el año 1972 por IBM
- El SO corría en entornos mainframe con arquitectura System/370, System/390 y mas avanzados en zSeries
- Utiliza como núcleo a VMCP (Virtual Machine Control Program), lo que permite la ejecución de las VM y el control del hardware



App	App	App	Modo usuario
CMS	CMS	CMS	
VM/370 (VMM, Virtual Machine Monitor)			Modo protegido
Hardware 370			



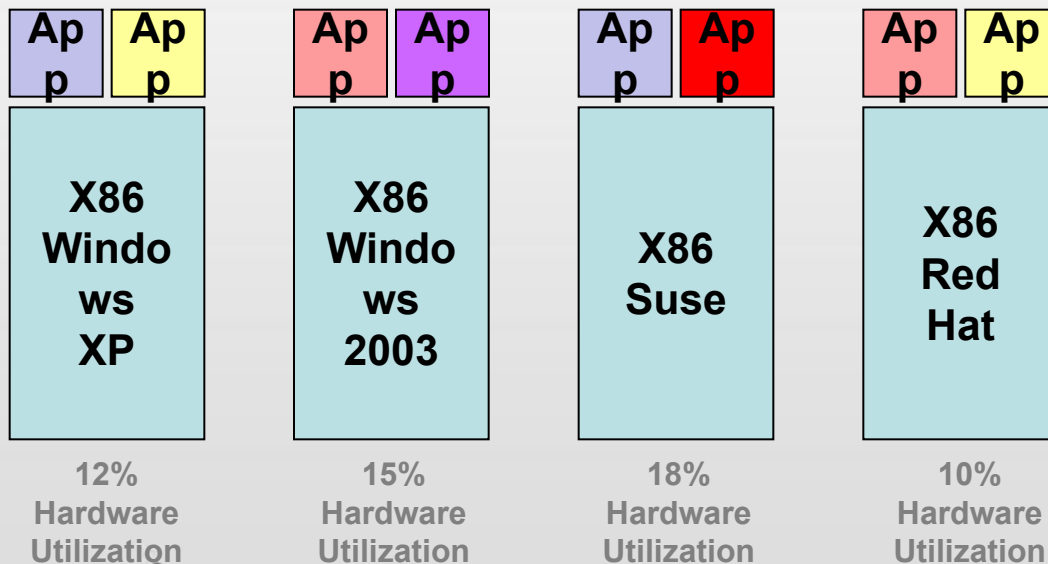
¿Cómo trabaja CMS y VM/370?

- En VM/370, hay una máquina CMS (Conversational Monitor System) para cada usuario, con “la ilusión” del hw completo.
- Una aplicación sobre el CMS hace una system call y la “atrapa” (es un trap) el CMS.
- Ejemplo: CMS emite instrucciones de I/O que son traps a VM/370



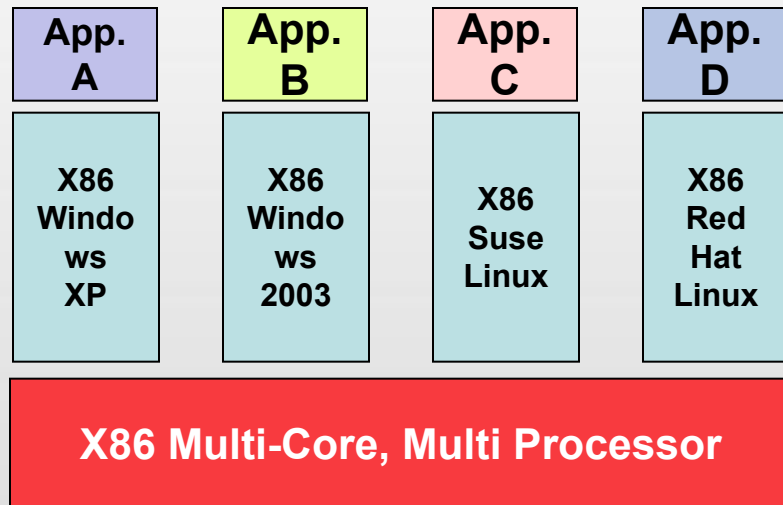
El esquema antes de la virtualización

- En el esquema de los años 2000, 1 máquina corría 1 SO junto con sus aplicaciones
- El aprovechamiento del hardware era bajo



El esquema con la virtualización

- Se aprovecha el hardware corriendo varios SO al mismo tiempo junto con sus aplicaciones
- Cada SO desconoce de la existencia de otros SO

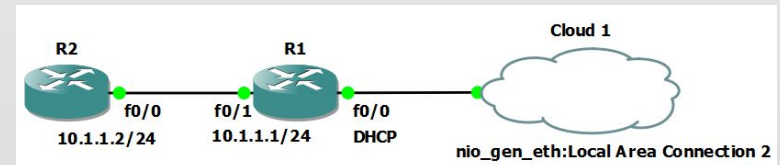
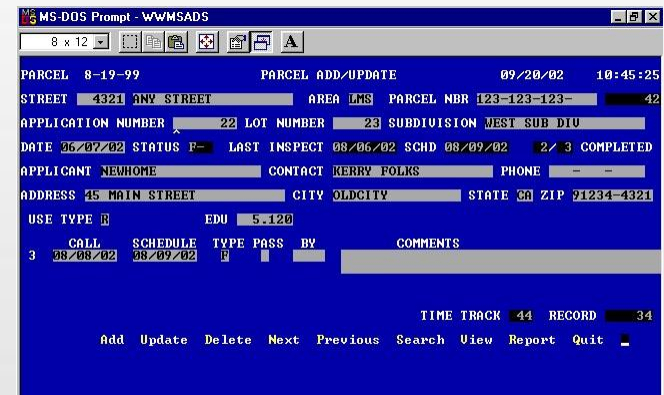
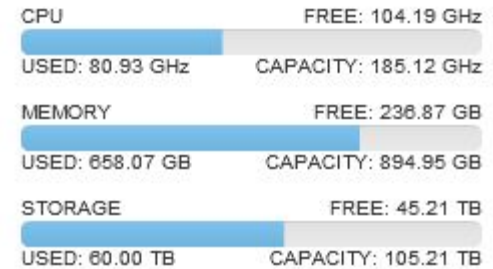


70% Hardware Utilization



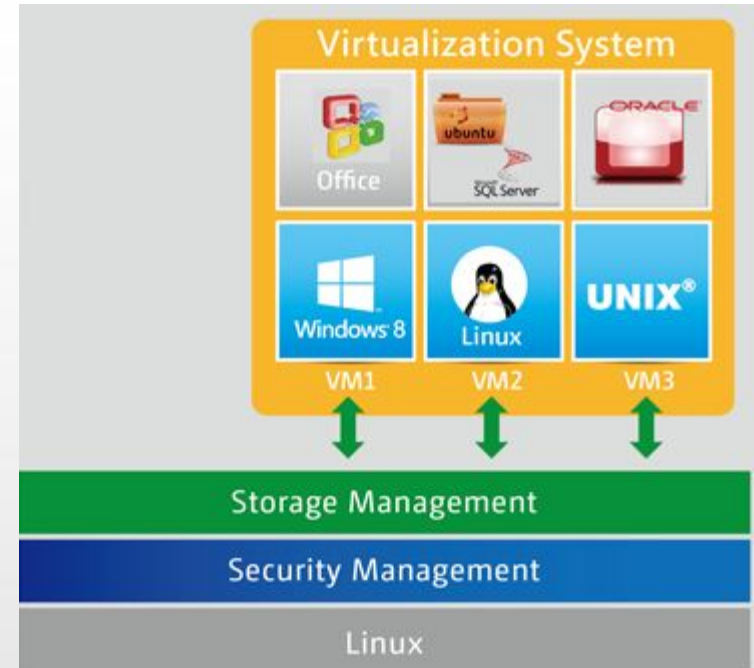
¿Por qué virtualizar?

- Tengo muchas máquinas servidores, poco usados.
- Tengo que correr aplicaciones heredadas (legacy) que no pueden ejecutarse en nuevo hw o SO.
- Tengo que probar aplicaciones no seguras.
- Tengo que crear un SO o entorno de ejecución con recursos limitados.
- Tengo que simular la computadora real, pero con un subconjunto de recursos
- Necesito usar un hw que no tengo (necesito “crear la ilusión” de hw).
- Necesito simular redes de computadoras independientes.



¿Por qué virtualizar? (cont)

- Tengo que correr varios y distintos SO simultáneamente.
- Necesito hacer testeo y monitoreo de performance
- Necesito que SOs existentes se ejecuten en ambientes multiprocesadores que comparten memoria
- Necesito facilidad de migración.
- Necesito ahorrar energía (tendencias de green IT o tecnología verde)



Virtualización. Ideas generales

- Es una capa de abstracción sobre el hw para obtener una mejor utilización de los recursos y flexibilidad.
- Permite que haya múltiples máquinas virtuales (VM), o entornos virtuales (EV), con distintos (o iguales) sistemas operativos corriendo de manera aislada.
- Cada VM tiene su propio conjunto de hardware virtual (RAM, CPU, NIC, etc.) sobre el cual se ejecuta el SO “**guest**”.
- El SO “**guest**” ve un conjunto consistente de hw, no el hw real (aunque a través de ciertas configuraciones podría ver parte del hardware real)



Virtualización

- VMs están representadas y son encapsuladas en archivos dentro de un File System.
- Fácil de almacenar, copiar.
- Fácil de hacer backup y restaurar.
- Simple de expandir y agregar recursos.
- Sistemas completos (aplicaciones ya configuradas, SO, hw virtual) pueden moverse de un servidor a otro rápidamente.



Componentes: software host y guest

- Existe un software **host** (que simula) y un software **guest** (lo que se quiere simular).
- **Guest** puede ser un sistema operativo completo



Monitor de Máquinas Virtuales (VMM)

- Programa que se corre sobre el hardware para implementar las máquinas virtuales
- Se encarga de controlar los recursos y de la planificación de los *guests*
- Consideraciones:
 - El VMM necesita ejecutarse en modo supervisor
 - El software *guest* se ejecuta en modo usuario
 - Las instrucciones privilegiadas en los *guests* implican traps al VMM
 - El VMM interpreta/emula las instrucciones privilegiadas



Características de una MV

(Popek y Goldberg, 1974)

“Formal Requirements for Virtualizable Third Generation Architectures”

- **Equivalencia / Fidelidad:** un programa ejecutándose sobre un **VMM** (Virtual Machine Monitor), también conocido como **Hipervisor**, debería tener un comportamiento idéntico al que tendría ejecutándose directamente sobre el hardware subyacente.
- **Control de recursos / Seguridad:** El VMM tiene que controlar completamente y en todo momento el conjunto de recursos virtualizados que proporciona a cada guest.
- **Eficiencia / Performance:** Una fracción estadísticamente dominante de instrucciones tienen que ser ejecutadas sin la intervención del VMM, o en otras palabras, directamente por el hardware.

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.4815&rep=rep1&type=pdf>



Repasemos algunos conceptos ya vistos....



Interrupciones

- Interrumpen el secuenciamiento del procesador durante la ejecución de un proceso, llamándose al manejador de la interrupción
 - **Por software:** cuando un usuario invoca, por ejemplo, una llamada al sistema.
 - **Por hardware:** cuando un dispositivo de hardware requiere la atención de la CPU para que ejecute su manejador. No son programadas
 - **Enmascarables:** son ignoradas por la unidad de control por el tiempo que permanezca en ese estado.
 - **No enmascarables:** siempre serán recibidas y atendidas por la CPU. Sólo unos pocos eventos críticos (como fallos del hardware) pueden generar interrupciones no enmascarables



Excepciones

- Causadas por errores de programación o condiciones anómalas que deben ser manejadas por el kernel
 - División por 0
 - Desbordamiento del Stack



Modo usuario y supervisor

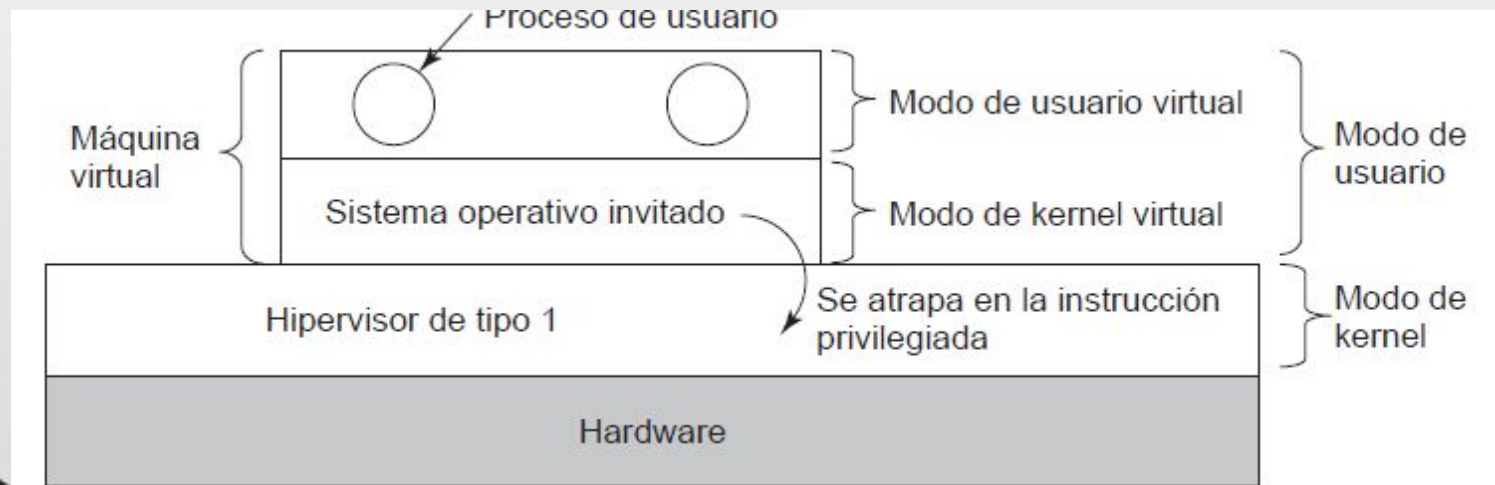
- En modo supervisor se puede ejecutar cualquiera de las instrucciones que acepta el procesador
- En modo usuario, solo se puede ejecutar un subconjunto de las instrucciones aceptadas



Modo usuario/supervisor

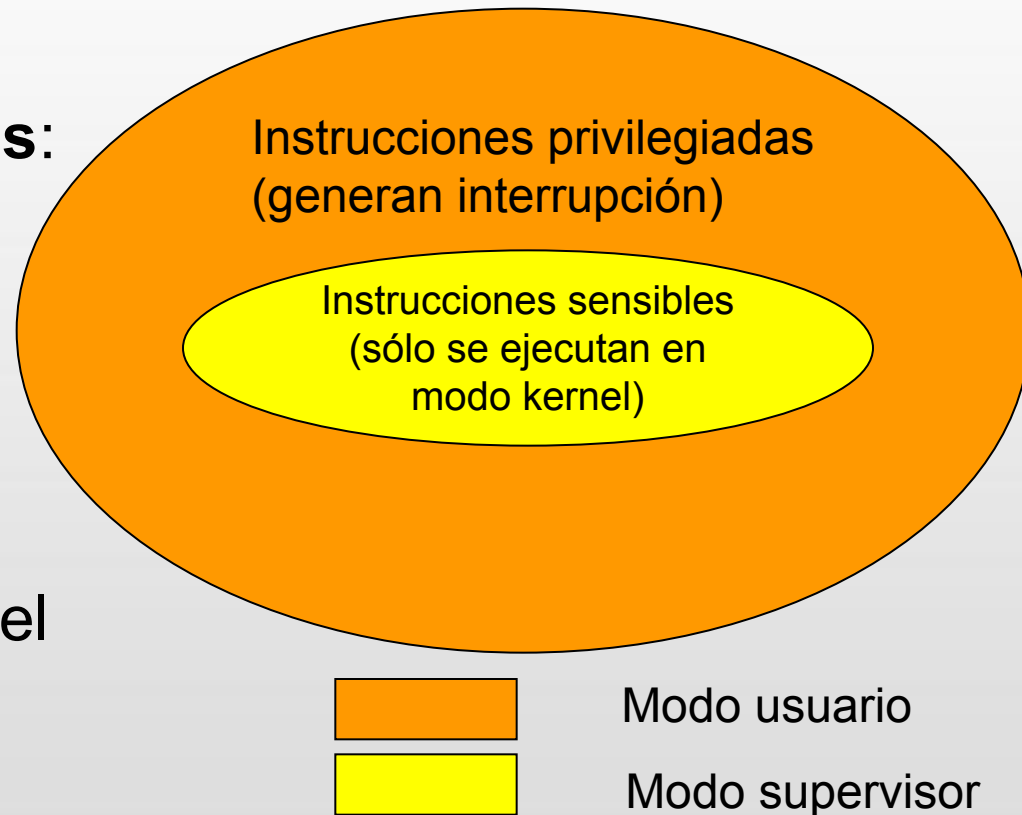
Instrucciones privilegiadas

- Cuando estoy virtualizando, el guest (corriendo en modo usuario) emitirá una instrucción privilegiada que NO debe ser ignorada, sino que debe generar un trap al SO.
- Mecanismo conocido como trap-and-emulate
- No aplicable en arquitectura x86



Instrucciones privilegiadas y sensibles

- **Instrucciones inocuas o no privilegiadas:** se ejecutan nativamente
- **Instrucciones privilegiadas:** provocan una interrupción al ser ejecutadas en modo usuario
- **Instrucciones sensibles:** deben ejecutarse en modo kernel (E/S, configuración del hard (MMU), administración de interrupciones)



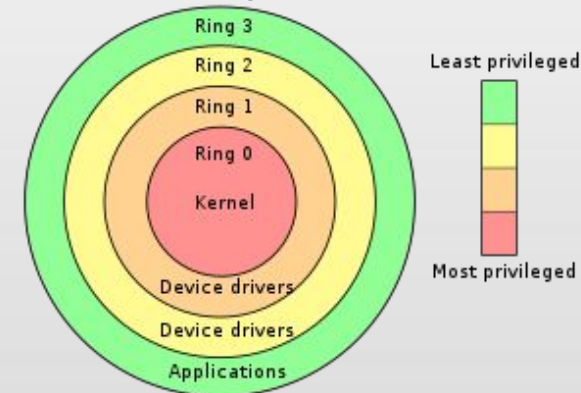
Instrucciones privilegiadas y sensibles

- Para construir un VMM es suficiente con que todas las instrucciones que podrían afectar al correcto funcionamiento del VMM (instrucciones sensibles) siempre generen una excepción y pasen el control al VMM.
- Las instrucciones no privilegiadas deben ejecutarse nativamente en el hardware (es decir, eficientemente).
- Para que un sistema soporte virtualización, las instrucciones sensibles deben ser un subconjunto de las privilegiadas



Anillos de privilegios en x86

- Los procesadores x86 proveen protección basada en el concepto de niveles o anillos (rings) de privilegios (0 = mas privilegio, 3 menos privilegio). Estos niveles se establecen en 2 bits del registro **CS**.
 - El software mas privilegiado se ejecuta en el Ring 0 (modo Kernel) y el menos privilegiado (modo usuario) en el 1, 2 o 3 (en particular en Intel se ejecuta en el 3)
- El nivel de privilegios determina si las instrucciones privilegiadas, que controlan la funcionalidad básica de la CPU, pueden ejecutarse sin generar una excepción
 - En Ring 0 se ejecutan directamente
 - En otros Rings se invocan a través de una excepción
- Se busca que los Hypervisores se ejecuten en el Ring 0 (o lo mas cercano) y los SO guest se ejecuten en el 1 o 3 (x86)



Trap and emulate

- Funciona similar a la emulación pero realiza una interpretación selectiva
- Aplicaciones y sistema operativo ejecutan en modo usuario
- Aplicaciones ejecutan nativamente en el hardware
- VMM ejecuta en modo privilegiado
- Cuando se ejecuta una instrucción privilegiada en el guest SO (en modo usuario) se produce un “trap” al VMM
- VMM ejecuta las instrucciones necesarias y retorna el control al guest SO
- No puede ser utilizado en todas las ISAs. Debe cumplir con el teorema de Popek and Goldberg
- x86 no cumple con el teorema: no todas las instrucciones sensitivas son privilegiadas (por ej. popf)



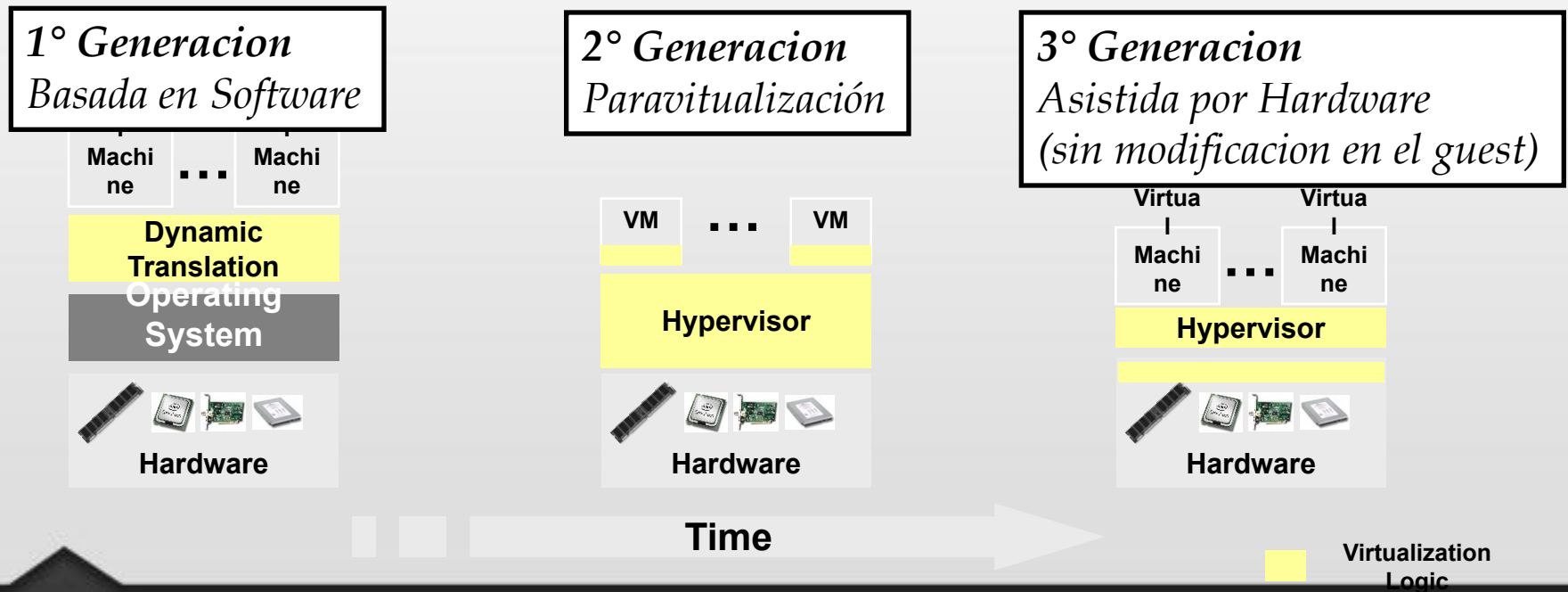
Hypervisors

- También conocidos como VMM (Virtual Machine Monitor) es una porción de software que separa las “aplicaciones/SO” del hardware subyacente
- Proveen una plataforma de virtualización que permite múltiples SO corriendo en un host al mismo tiempo.

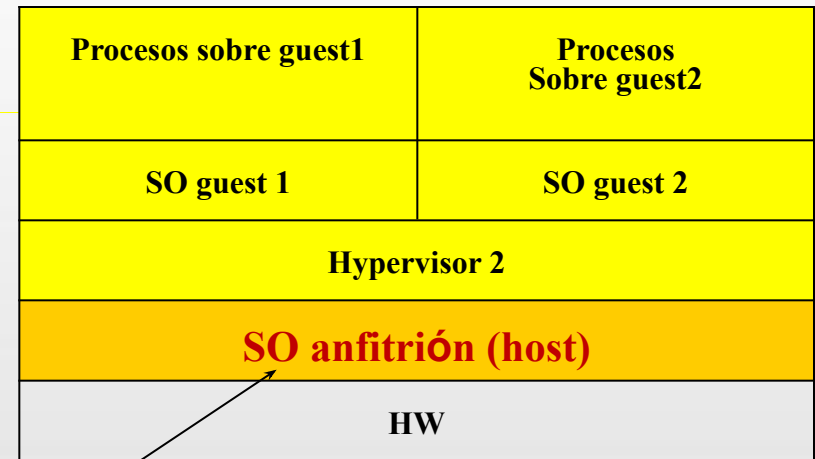
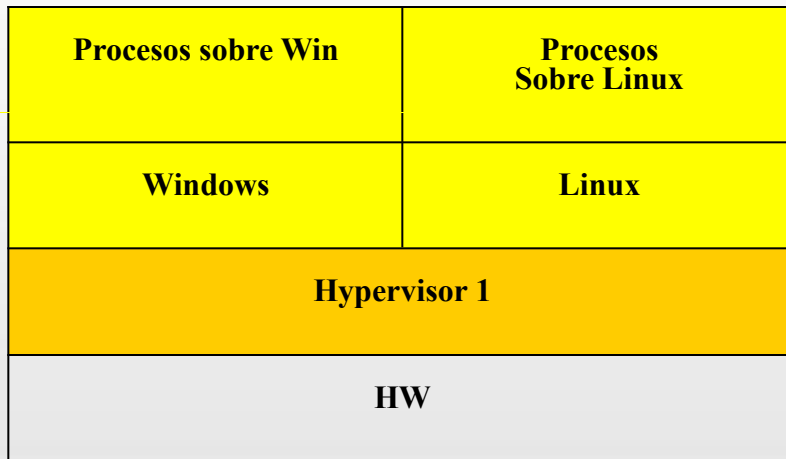


Hypervisors

- Interactúan con el HW en líneas generales (2° y 3° generación)
- Realizan la multiprogramación
- Ofrece varias MV hacia arriba



Hypervisor 1 e Hypervisor 2

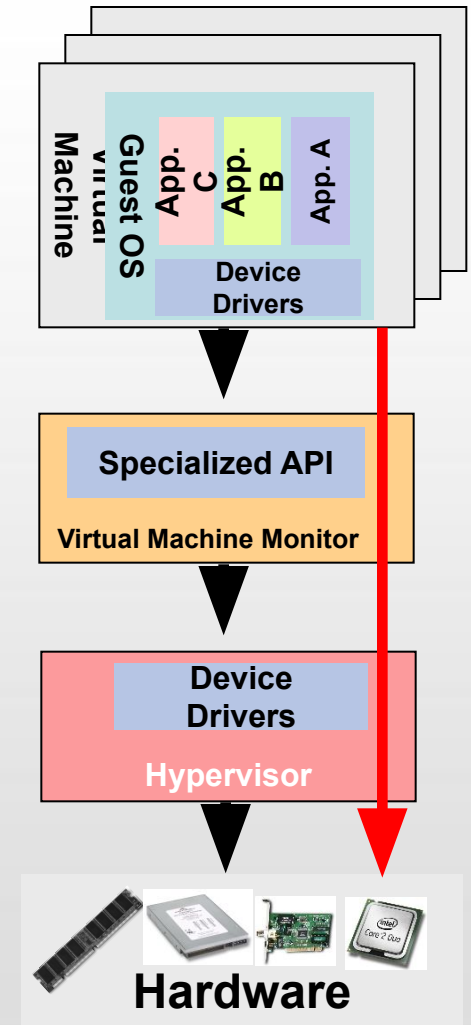


La diferencia mas importante es la existencia o no de un SO host



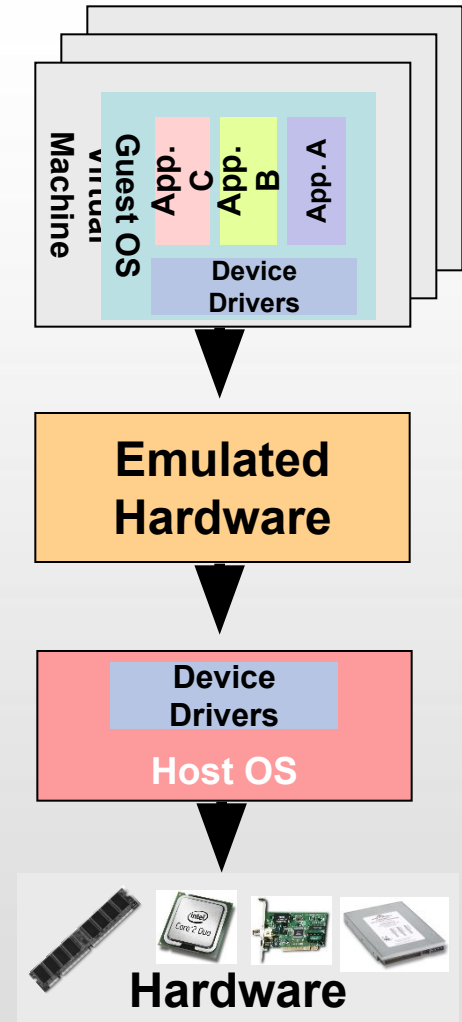
Hypervisor tipo 1

- Se ejecuta en modo kernel (Ring 0)
- Cada VM se ejecuta como un proceso de usuario en modo usuario (Ring 3)
- El SO guest no requiere ser modificado
- Existen un modo kernel virtual y modo usuario virtual
- Siempre que la VM ejecuta una instrucción sensible, se produce una trap que procesa el hypervisor. Algunos hipervisores introducen extensiones que le evitan tener que traducir todas las instrucciones
- Debe tener asistencia de Hardware Siempre



Hypervisor tipo 2

- Se ejecuta como un programa de usuario en un SO host
- Su función principal de interpretar un subconjunto de las instrucciones de hardware de la máquina sobre la que corre
- Debe emularse el hardware que se mapea a los SO guest



Hypervisors - Comparación

■ Tipo 1

- Se ejecuta sobre el HW
- Hypervisor se ejecuta en modo kernel real
- SO guest en modo kernel “virtual” (pero es modo usuario)

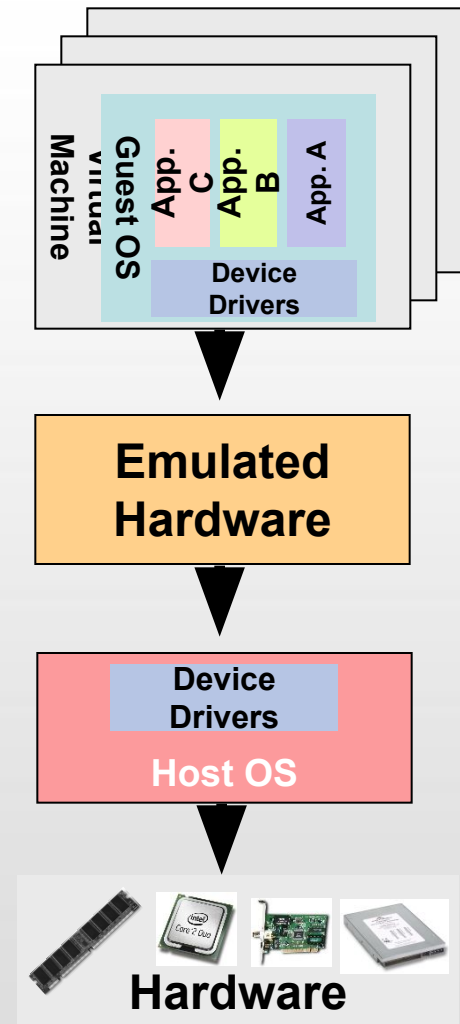
■ Tipo 2

- Se ejecuta como un programa de usuario sobre un SO host.
- Arriba de él, están los SO guests.
- Interpreta un conjunto de instrucciones de máquina.
- El SO host es quién se ejecuta sobre el HW



¿Por qué un hypervisor 2 funciona en CPU no VT?

- El hipervisor de tipo 2 ejecuta en modo usuario como un proceso más del SO anfitrión
- Las instrucciones sensibles se sustituyen por llamadas a procedimiento que emulan las instrucciones
- El verdadero HW NUNCA ejecuta las sensibles que emite el guest: las ejecuta como llamadas el hypervisor
- Todo el hardware es emulado, inclusive la CPU



Técnicas de Virtualización

- De acuerdo al modo en el que se implemente el VMM podemos nombrar:
 - Emulación
 - Asistida por Hardware
 - Paravirtualización



Emulación

- **Emulación:** provee toda la funcionalidad del procesador deseado a través de software (ej: QEMU, MAME).
 - Se puede emular un procesador sobre otro tipo de procesador
 - Aplicación/SO emulado ejecuta en modo usuario
 - Se reescribe el conjunto completo de instrucciones
 - Todas las instrucciones son capturadas por el emulador
 - Cada instrucción es interpretada y traducida a una (o varias) equivalente adecuada al hardware subyacente
 - Tiende a ser lenta



Full Virtualization

- **Full Virtualization:** se trata de particionar un procesador físico en distintos contextos, donde cada uno de ellos corre sobre el mismo procesador.
 - Los SO guest deben ejecutar la misma arquitectura de hardware sobre la que corren
 - No requiere que los guest se modifiquen
 - Es en general la técnica mas utilizada
 - El VMM analiza el flujo de ejecución
 - Los bloques que contienen instrucciones sensibles son modificados
 - Los bloques con instrucciones inocuas se ejecutan directamente en el hardware



Full Virtualization: Traducción binaria

- Introducida en 1998 por VMWare para permitir virtualizar sobre arquitecturas x86
- Se combina traducción binaria con ejecución directa
- Instrucciones no sensibles ejecutan directamente sobre el hardware
- El hypervisor continuamente analiza en runtime el flujo de ejecución (bloques de código) de los SO guest y “traduce” las instrucciones sensibles por llamadas al hypervisor
- Los bloques traducidos son ejecutados por la CPU directamente
- Permite mejorar el rendimiento al poner en cache los bloques traducidos



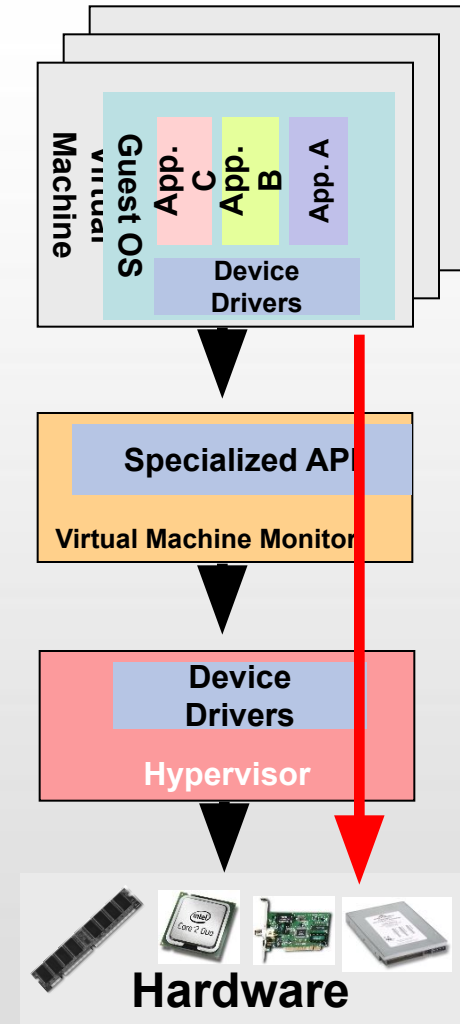
Asistida por hardware

- Se necesitan CPU “virtualizables”
- Intel la llama VT (Tecnología de virtualización) y AMD, SVM (máquina virtual segura)
- Root-Mode y Non-Root Mode en Intel. Host Mode y Guest Mode en AMD
- VMM ejecuta en root-mode, VMs en non-root mode. Ambos en ring-0
- Cada modo 4 anillos de seguridad con diferentes niveles de privilegio
- Ms generan traps al hypervisor cuando se ejecutan instrucciones sensibles (non-root mode)



Paravirtualización

- Los hypervisors tipo 1 y 2 ejecutan SO guests no modificados.
- Paravirtualización: se trata de tener SO guests modificados para mejorar el rendimiento.
- Cuando se quiere ejecutar una instrucción sensible, el SO guest la transforma en una llamada al VMM que expone una API específica.
- El VMM:
 - No realiza traducción binaria completa
 - No debe emular instrucciones de hardware, lo cual hace que las llamadas se resuelvan de modo mas sencillo
- El SO guest es como un proceso de usuario que hace llamadas al SO (el hypervisor)



Paravirtualización - Como se implementa

- El hypervisor cuenta con una API, que es un conjunto de llamadas a procedimientos.
- Los guests, en vez de invocar instrucciones sensibles, invocan a estas llamadas (hypercalls)
- Se eliminan las instrucciones sensibles de los guest y se reemplazan por llamadas a la API especializada.
- El hypervisor se transforma en un microkernel.
- Decimos que un SO está paravirtualizado cuando se han eliminado, intencionalmente, algunas instrucciones sensibles (si se eliminan todas es paravirtualización completa si solo se eliminan algunas, se la llama paravirtualización parcial)
- Si no se eliminan TODAS las instrucciones sensibles, el VMM deberá realizar traducción binaria



Paravirtualización - Implementaciones

- Existen 2 modelos:
 - **Recompilando el Kernel del sistema guest:**
 - Los drivers y la forma de invocar a la API residen en el kernel
 - Es necesario instalar un sistema operativo modificado/específico
 - **Instalando drivers paravirtualizados:**
 - La paravirtualización es parcial (para algunas funciones y dispositivos)
 - Generalmente utilizada para placas de red, o gráficas
 - Esta es la técnica que se utiliza hoy en día



Virtualización vs. Paravirtualización

virtualización “completa”

paravirtualización

Procesos usuario		Procesos usuario	
SO sin modificar (GNU/Linux, Windows)	Trap x instrucciones sensibles. Hypervisor emula y regresa.	Se accede a la API para todas o ciertas instrucciones. Se desacopla la función del hypervisor y VMM	SO modificado (GNU/Linux, Windows)
Hypervisor tipo 1		Microkernel	
HARDWARE			



Virtualización vs. Paravirtualización

- Virtualización completa o nativa puede generar problemas de performance ya que tiene que **emular la totalidad del hardware**.
- Paravirtualización completa en kernel tiene mejor performance, pero soporta pocos SO, pues necesita modificar el SO original.
- Paravirtualización parcial en drivers es una solución intermedia



Software y Características



ISO 2008 - Fac. de Informática -
U.N.L.P.

Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

VMware Workstation



- Es un hypervisor de tipo 2
- Explora código buscando bloques básicos (instrucciones seguidas que no cambien el program counter).
- Si hay instrucciones sensibles, sustituye cada una por una llamada a VMware.
- El bloque se pone en la cache de VMware, luego se ejecuta
- Si no hay instrucciones sensibles, el bloque se ejecuta tan rápido como si fuera nativa.



VMware Workstation

- La acción de atrapar instrucciones sensibles y emularlas se conoce como traducción automática.
- Generalmente tienen un costo en la performance:
 - -2% sobre CPU y RAM
 - Entre -8% y -20% para dispositivos de I/O



KVM (Kernel-based Virtual Machine)



- Infraestructura de virtualización para el kernel de Linux.
- Soportado nativamente en el kernel desde 2.6.20
- Se lo considera tipo 1 ya que está embebida en el kernel del SO
- Virtualiza plataformas X86 de 32 y 64 bits
- Requiere un procesador con extensión para virtualización.
- Sistemas Operativos guest que admite: versions de Linux, BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, AROS Research Operating System, Android 2.2, GNU/Hurd (Debian K16), Minix 3.1.2a, Solaris 10 U3, Darwin 8.0.1, entre otros
- Soporta paravirtualización de algunos dispositivos para Linux, OpenBSD, FreeBSD, NetBSD, Plan 9 y Windows mediante API

<https://www.redhat.com/es/topics/virtualization/what-is-KVM>



- No es en si un hypervisor, sino que es una distribución GNU/Linux basada en Debian que agrupa funcionalidades de virtualización:
 - Utiliza KVM como hypervisor y LXC para la gestión de contenedores. También utiliza QEMU para emular ciertas VMs
 - Provee una interfaz web y una CLI para la gestión de las VMs
 - Es de código abierto. Tiene versión gratis y paga (soporte)
 - Ya que usa KVM, aprovecha sus características como, por ejemplo, la migración en caliente
 - Soporta CEPH, que permite crear un Sistema de archivos distribuido entre todos los hosts
 - Se puede configurar en cluster (varios hosts corriendo proxmox y gestionarlos desde uno de ellos)
 - Su instalación es sencilla a través de una “.iso”. El instalador provee el SO y las herramientas.



ProxmoxVE



Hyper-V (Windows Server Virtualization)

- Infraestructura de virtualización propietaria de Microsoft
- Requiere un procesador con extensión para virtualización.
- Virtualiza plataformas X86 de 32 y 64 bits
- Se lo considera tipo 1, ya que está embebida en el Kernel del SO
- También conocido como Viridian fué introducido en Windows 2008 server en el año 2008
- Es un hypervisor nativo. La funcionalidad se agrega como un Nuevo “Rol” en versiones de Windows Server y Windows 10 Pro (no Home Edition)
- Permite correr SO guest Windows, GNU/Linux, BSD y otros
- Junto con VMWare lideran el Mercado de los sistemas de virtualización empresariales



VMware



- Infraestructura de virtualización propietaria
- Porvee 2 versiones:
 - Workstation: es un hypervisor tipo 2 y requiere un procesador con extensión para virtualización
 - ESXi o vSphere para servers: Es un hypervisor tipo 1
- Virtualiza plataformas x86 de 32 y 64 bits
- Virtualiza redes, firewalls, almacenamiento
- Incluye una importante suite de herramientas que permiten, de acuerdo a la licencia adquirida, utilizar un mayor o menos número de funciones
- ESXi, junto con Hyper-V, lideran el Mercado de los sistemas de virtualización empresariales





- Infraestructura de virtualización de código abierto desarrollada por la Universidad de Cambridge
- Requiere un procesador con extensión para virtualización.
- Virtualiza plataformas X86 de 32 y 64 bits
- Es un hypervisor nativo y también soporta paravirtualización
- Permite correr SO guest Windows, GNU/Linux, BSD y otros
- Permite la migración de máquinas virtuales en caliente entre miembros de un cluster XEN

<https://xenproject.org/>

