

Virtualización



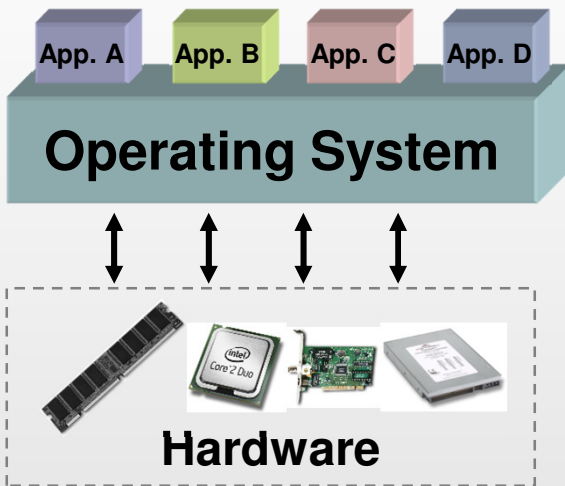
Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

Que es virtualizar?

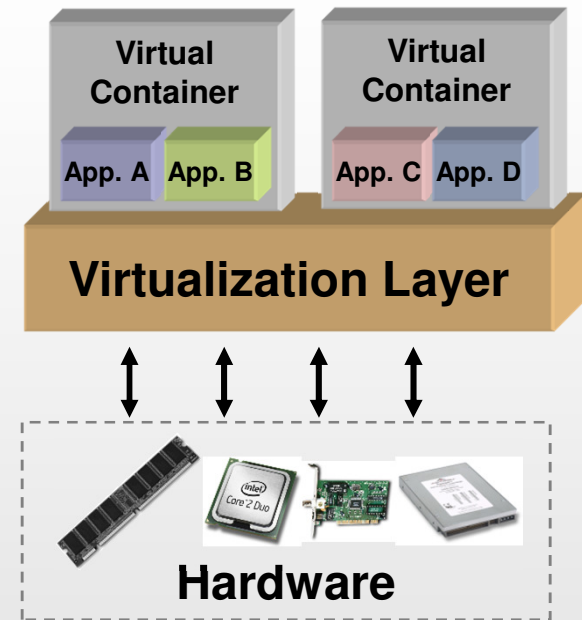
- Es una técnica que permite realizar “abstracción” de recursos de la computadora. Hoy en día también se virtualizan las redes, datacenters, etc...
- Permite ocultar detalles técnicos a través de la encapsulación
- Permite, entre otras cosas, que una computadora pueda realizar el trabajo de varias a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



¿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			



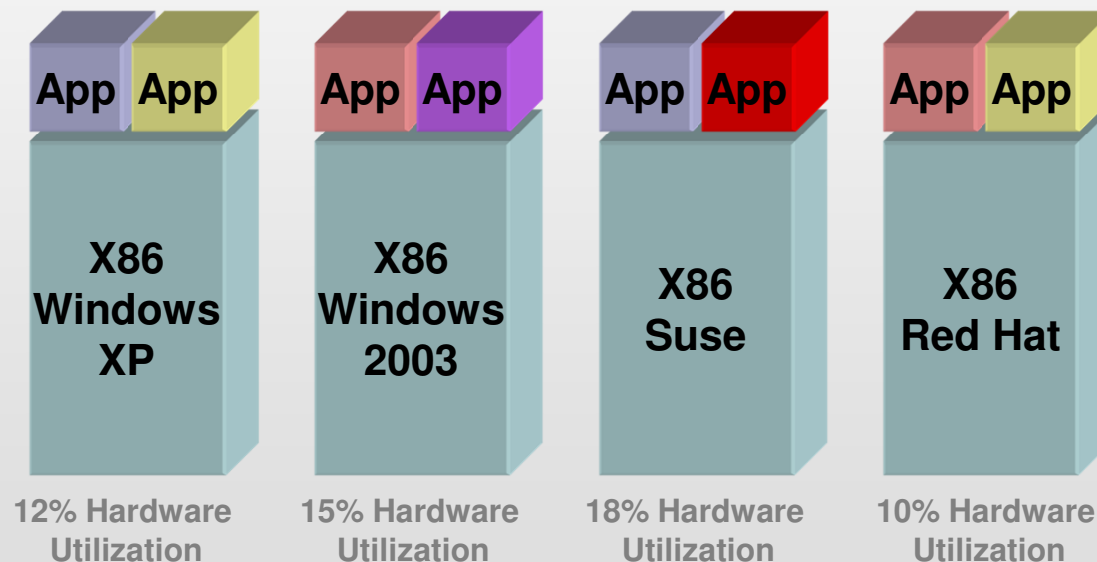
¿Cómo trabaja CMS y VM/370?

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



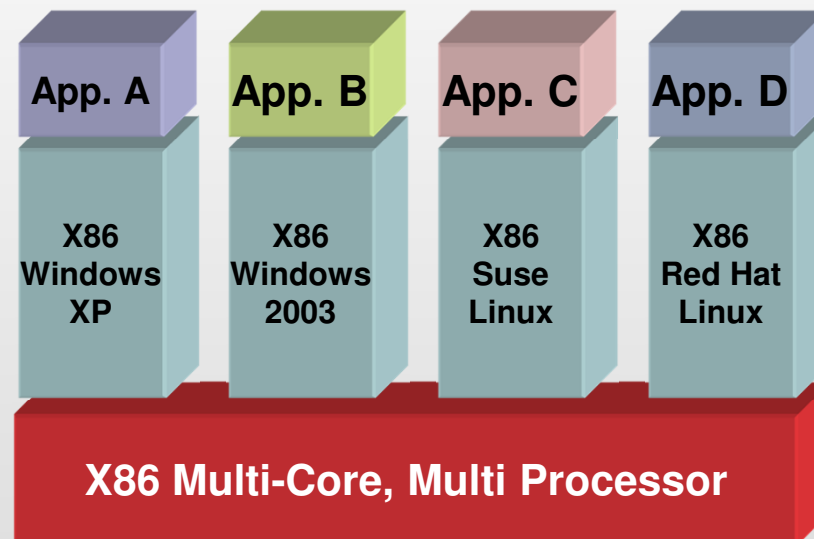
El esquema antes de la virtualización

- 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

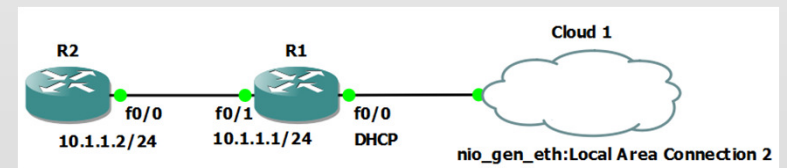
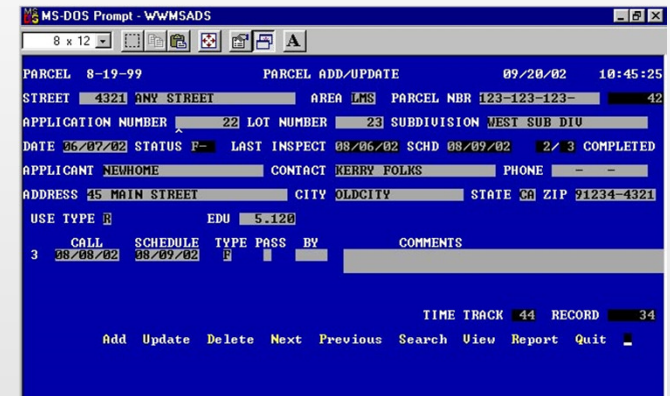
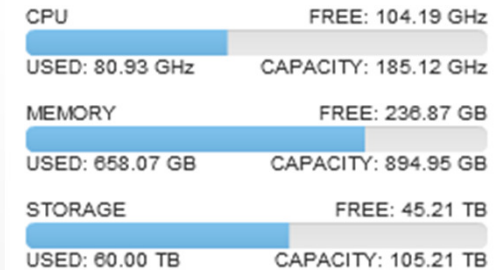


70% Hardware Utilization



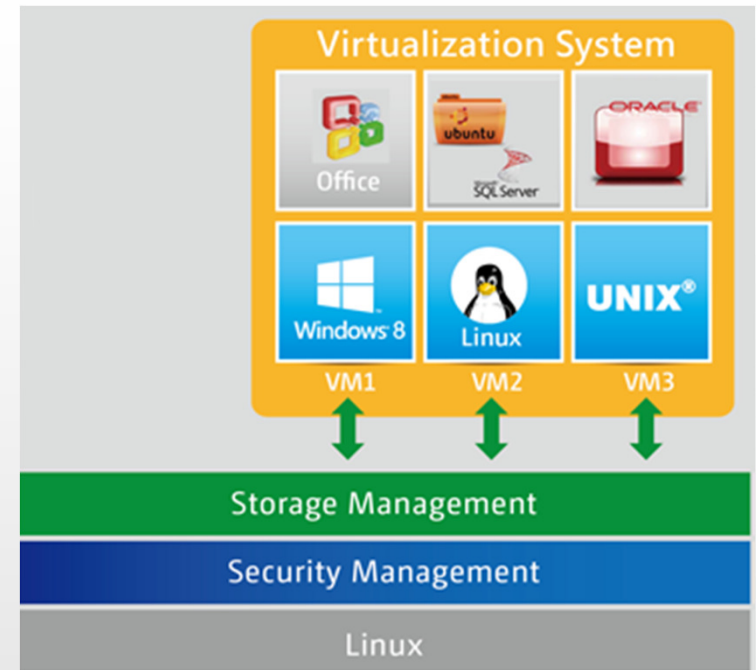
¿Por qué Virtualizar?

- 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 entornos 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 (MV) o entornos virtuales (EV), con distintos (o iguales) sistemas operativos corriendo aisladamente.
- Cada MV tiene su propio conjunto de hardware virtual (RAM, CPU, NIC, etc.) sobre el cual se carga el SO “guest”.
- El SO “guest” ve un conjunto consistente de hw, no el hw real (aunque a través de ciertas configuraciones).



Virtualización

- VM están representadas y son encapsuladas en archivos dentro de un File System.
- Fácil de almacenar, copiar.
- Sistemas completos (aplicaciones ya configuradas, so, hw virtual) pueden moverse de un servidor a otro, rapidamente.



Componentes: software host y guest

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



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) debería que 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.



Emulación y virtualización

- Algunas maquinas virtuales en realidad son emuladas.
- Son técnicas parecidas, pero tienen diferencias
- **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
 - Tiende a ser lenta.
- **Virtualización:** se trata de particionar un procesador físico en distintos contextos, donde cada uno de ellos corre sobre el mismo procesador.
 - Es más rápida que la emulación.



Repasemos algunos
conceptos ya vistos....



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

Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

Interrupciones

- Evento que altera la secuencia de instrucciones ejecutada por un procesador
- **Interrupciones enmascarables:** son ignoradas por la unidad de control por el tiempo que permanezca en ese estado.
- **Interrupción 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
- Algunas son detectadas por el procesador y otras son programadas
- Entre los tipos de excepción están los traps (los otros son fallos o abortos)



Modo usuario y supervisor

- En modo supervisor se puede ejecutar cualquiera de las instrucciones que acepta el procesador
- En modo usuario, no



Modo usuario/supervisor

Instrucciones privilegiadas

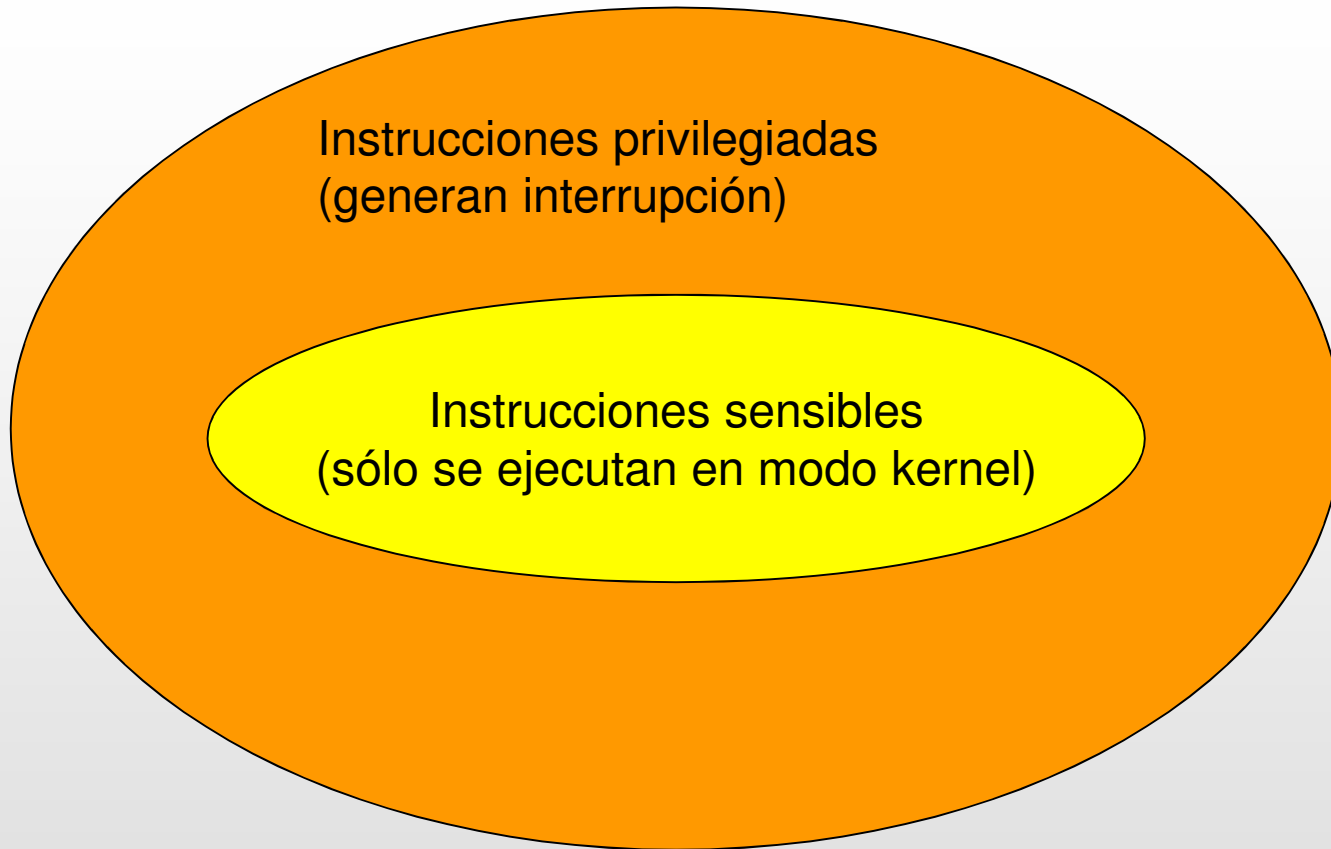
- Cuando estoy virtualizando, el guest (corriendo en modo usuario) emitirá una instrucción privilegiada que NO debe ser ignorada.
- INTEL 386 ignoraba las instrucciones privilegiadas que se invocaban en modo usuario.



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 (es decir, eficientemente).
- Una arquitectura es virtualizable si todas las instrucciones sensibles son privilegiadas.





Modo usuario



Modo supervisor



Hypervisors

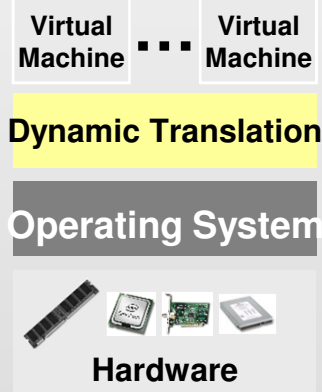
- También conocidos como VMM (Virtual Machine Monitor) es una porción de software que separa las “aplicaciones/SO” del hardware subyacente
- Proveen una plataforma de virtualización que permite múltiples SO corriendo en un host al mismo tiempo.



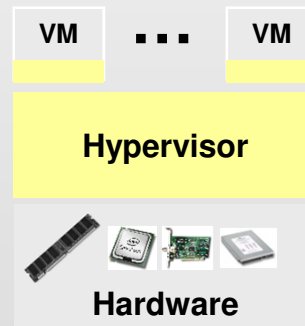
Hypervisors

- Interactúan con el HW
- Realizan la multiprogramación
- Ofrece varias MV hacia arriba

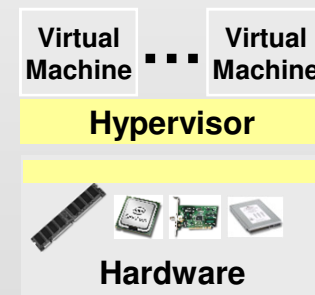
*1 ° Generacion
Basada en Software*



*2 ° Generacion
Paravirtualización*



*3 ° Generacion
Asistida por Hardware
(sin modificacion en el guest)*



Time

Virtualization Logic

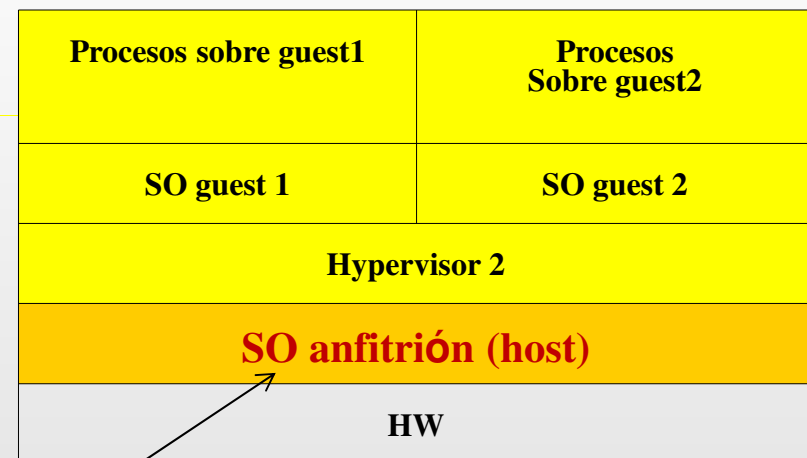
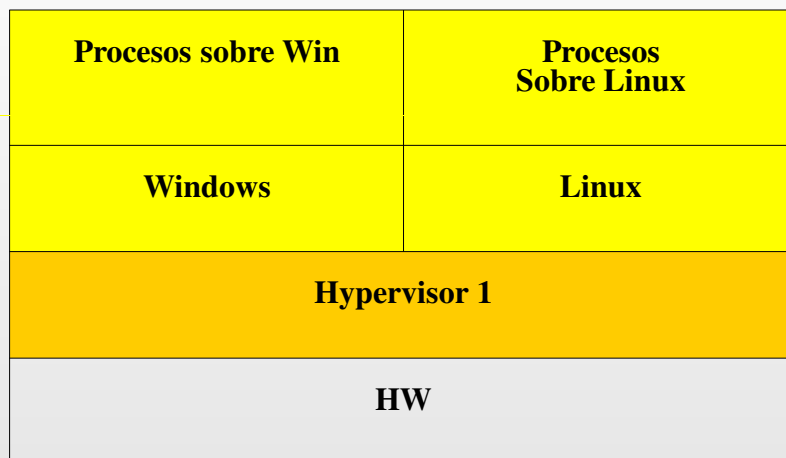


CPUs “virtualizables”

- El principal problema es la ejecución de excepciones desde los guest, ya que deben ser “controladas”. El hypervisor es quien realiza esta tarea
- Se necesitan CPU “virtualizables” (aptas para virtualizar)
- Intel Core2 la llama VT (Tecnología de virtualización)
- AMD Pacific la llama SVM (máquina virtual segura)
- Se usa VT en forma genérica



Hypervisor 1 e Hypervisor 2

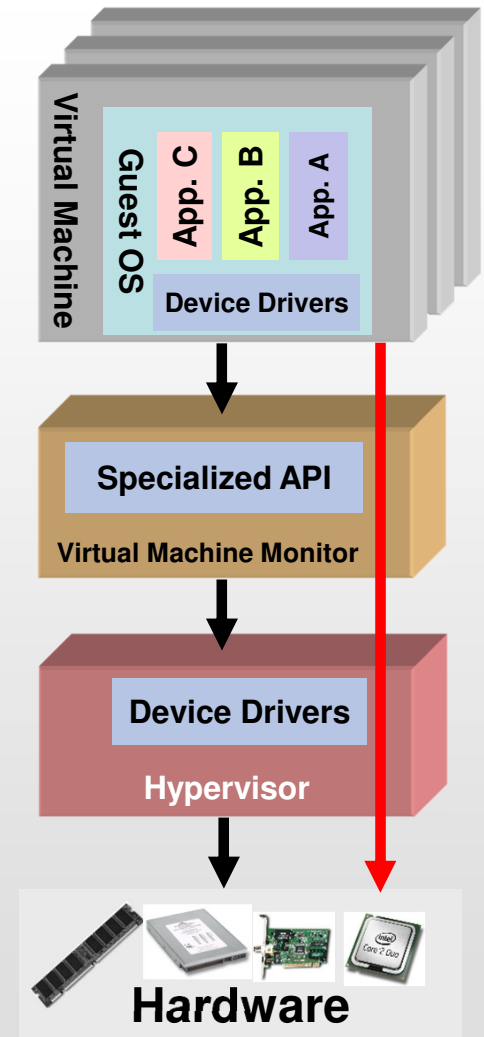


La diferencia mas importante es la existencia o no de un SO host

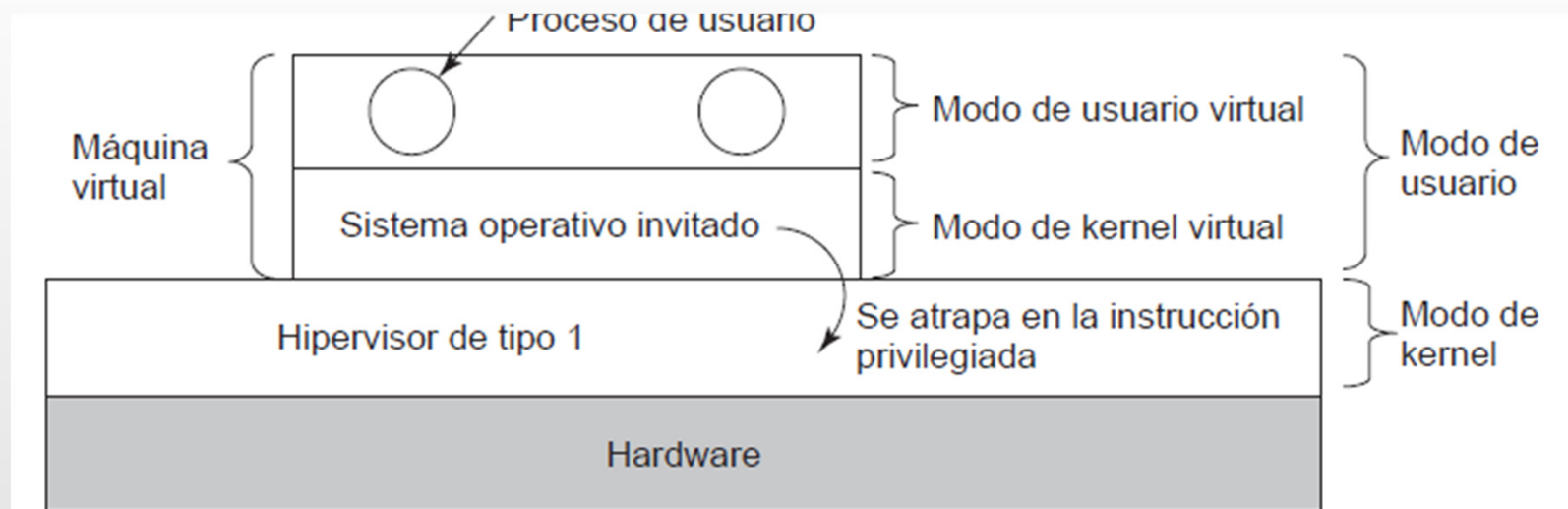


Hypervisor tipo 1

- Se ejecuta en modo kernel
- Cada VM se ejecuta como un proceso de usuario en modo usuario
- Existen un modo kernel virtual y modo usuario virtual
- Cuando la VM ejecuta una instrucción sensible, se produce una trap que procesa el hypervisor.
- Debe tener asistencia de Hardware Siempre

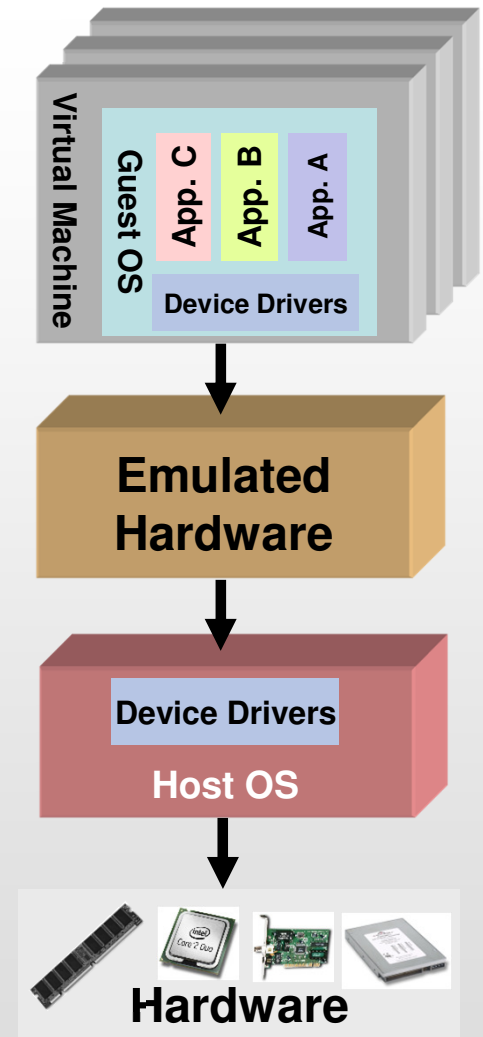


Hypervisor tipo 1



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 es 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



Hypervisors - Comparación

Procesos sobre Win	Procesos Sobre Linux
Windows	Linux
Hypervisor 1	
HW	

Procesos sobre guest1	Procesos Sobre guest2
SO guest 1	SO guest 2
Hypervisor 2	
SO anfitrión (host)	
HW	

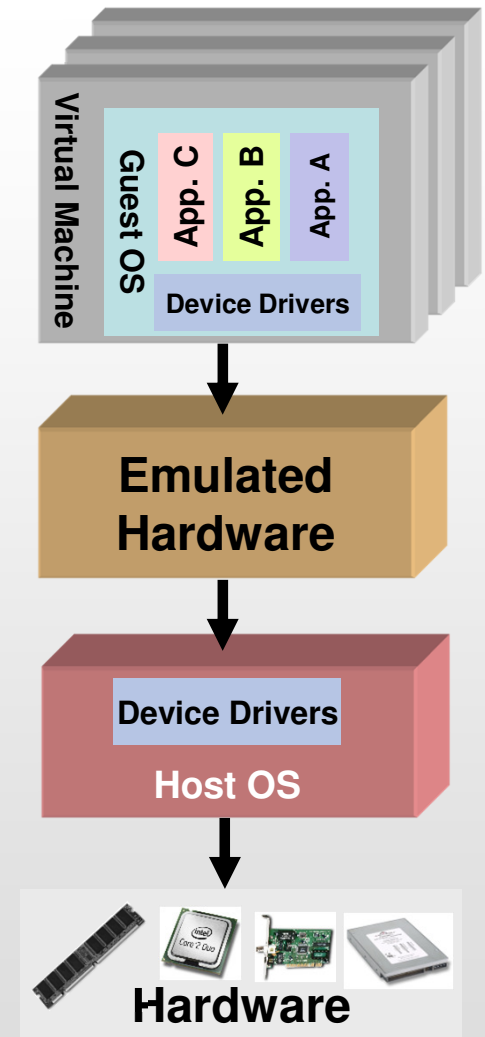
- Cuando un SO guest emite una instrucción sensible:
 - CPU no VT => ignora
 - CPU con VT => hay un trap al kernel
 - Kernel analiza: si la emitió el proceso usuario, emulará el HW sobre el guest.

Se ejecuta como un programa de usuario sobre un SO host.
Arriba de él, están los SO guests



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

- Porque 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



Traducción binaria

- El hypervisor continuamente analiza en runtime el flujo de ejecución (bloques de código) de los SO guest
- EL VMM modifica en runtime la imagen binaria del SO guest con el fin de poder tener control sobre el guest cuando éste intenta ejecutar una instrucción sensible
- Cuando se produce una instrucción sensible la intercepta, la convierte en una llamada al host y luego devuelve el control al guest.



VMware Workstation



- Explora código buscando bloques básicos (instrucciones seguidas que no cambien el program counter).
- Si hay instrucciones sensibles, sustituye cada una x 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



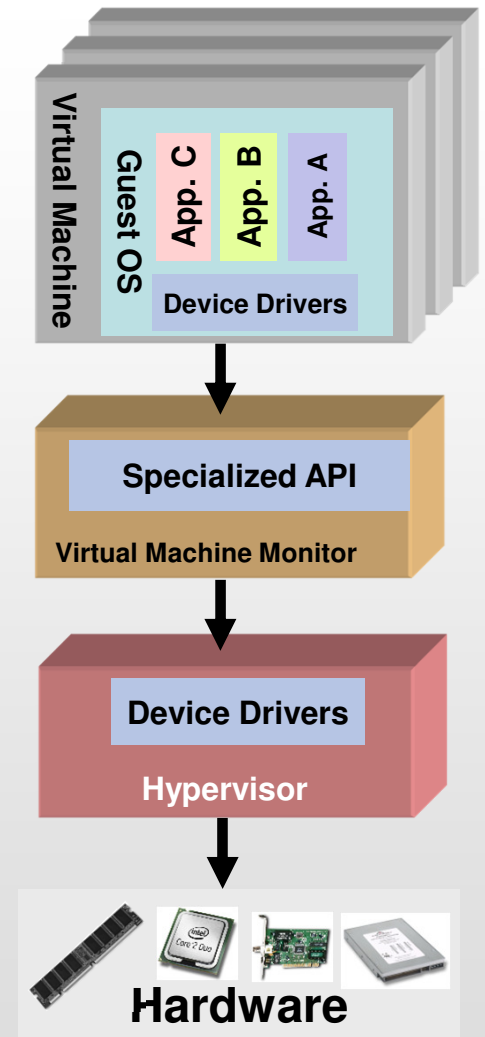
Performance – Tipo 1 vs Tipo 2

- Tipo 1: todas las instrucciones sensibles generan una trap que debe atender el hipervisor (arruina “localidad”, caché de CPU, TLBs, predicción de bifurcación, etc desde el guest). El hypervisor maneja estos conceptos.
- Tipo 2: todo el flujo de instrucciones debe ser traducido (costo inicial alto) pero luego quedan en cache y no se produce cambio de contexto al ejecutar una instrucción sensible
 - Hipervisores de tipo 1 también pueden realizar traducción binaria



Paravirtualización

- Los Hypervisores tipo 1 y 2 ejecutan SO guest 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 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 son un conjunto de llamadas a procedimientos.
- Los guests invocan estas llamadas.
- El hypervisor se transforma en un microkernel.
- Decimos que un SO está paravirtualizado cuando se han eliminado, intencionalmente, algunas instrucciones sensibles



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



Virtualización vs. Paravirtualización

virtualización “completa”

paravirtualización

Procesos usuario			Procesos usuario
Windows sin modificar	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	Linux modificado
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.
- Para virtualización parcial en drivers podría ser una solución intermedia



KVM (Kernel-based Virtual Machine)



- Infraestructura de virtualización para el kernel de Linux.
- 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



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
- También conocido como Viridian fué introducido en Windows 2008 server en el año 2008
- Es un hypervisor nativo
- Permite correr SO guest Windows, GNU/Linux, BSD y otros



VMware

- Infraestructura de virtualización propietaria
- Porvee 2 versiones:
 - Workstation: Es un hypervisor tipo 2 y requiere un procesador con extensión para virtualización
 - ESXi para servers: Es un hypervisor tipo 1
- Virtualiza plataformas X86 de 32 y 64 bits
- Virtualiza redes, firewalls, almacenamiento



vmware®



vmware®
ESXi



Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

Xen

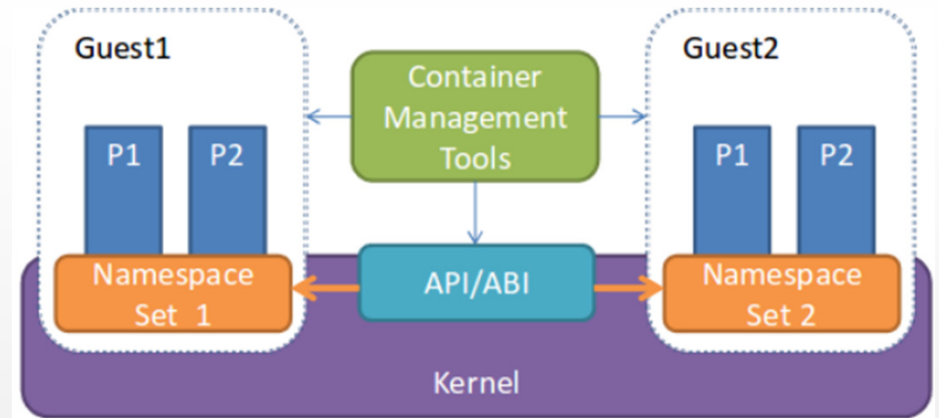


- Infraestructura de virtualización de código abierto desarrollada por la Universidad de Cambridge
- Requiere un procesador con extensión para virtualización.
- Virtualiza plataformas X86 de 32 y 64 bits
- Es un hypervisor nativo y también soporta paravirtualización
- Permite correr SO guest Windows,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



Linux Containers

- Grupos de procesos corriendo sobre el mismo kernel pueden aislarse en entornos que parecen máquinas separadas.
- No son una máquina virtual
- Son entornos virtuales, con su propio espacio de procesos y de nombres
- Se basan en cgroups (funcionalidad propia del kernel Linux)
- Algunos le llaman “virtualización por sistema operativo”



Linux Containers

- Algunos usos:
- Controlar asignación de recursos. Puedo asignarle a un container más ancho de banda de red, o tiempo de CPU
- Se puede “freezar” un container (todos los procesos del container se suspenden. Util para migraciones.



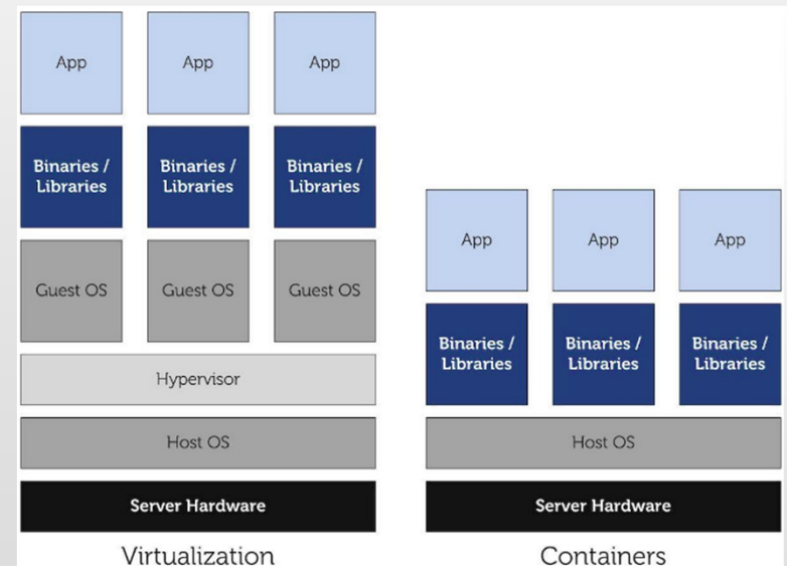
Máquinas virtuales vs containers

MV

- Existe una instancia del kernel
- Puedo disponer de distintos sistemas operativos
- Mayor Consumo de recursos por cada máquina virtual
- Ofrecen mayor separación y seguridad.

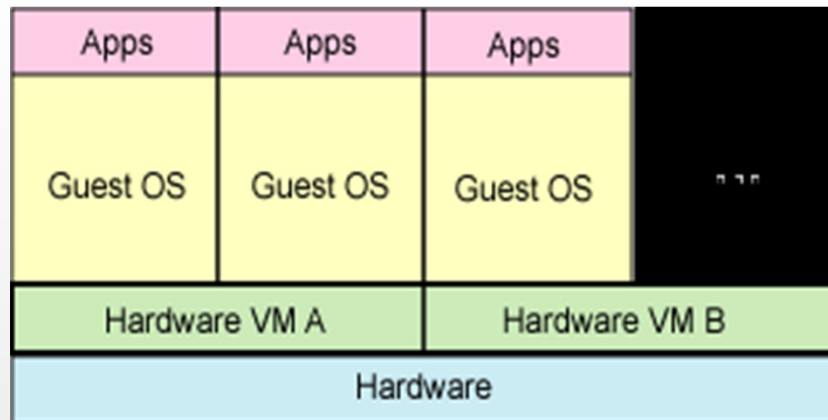
Containers

- No es necesario
- La virtualización se puede hacer por SO y por aplicación
- La creación y destrucción de contenedores es menos costosa

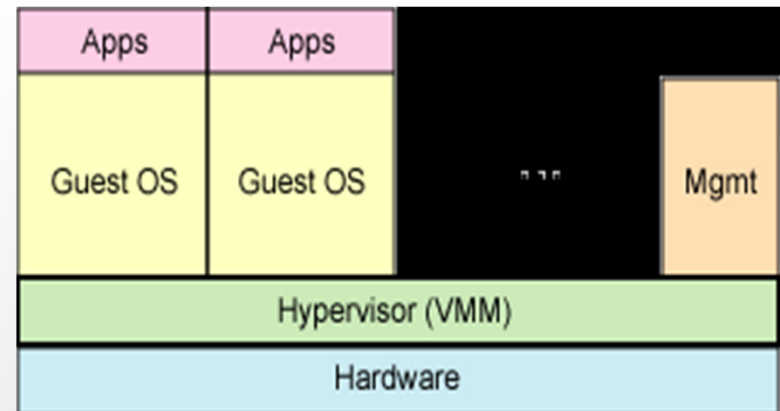


Resumiendo...

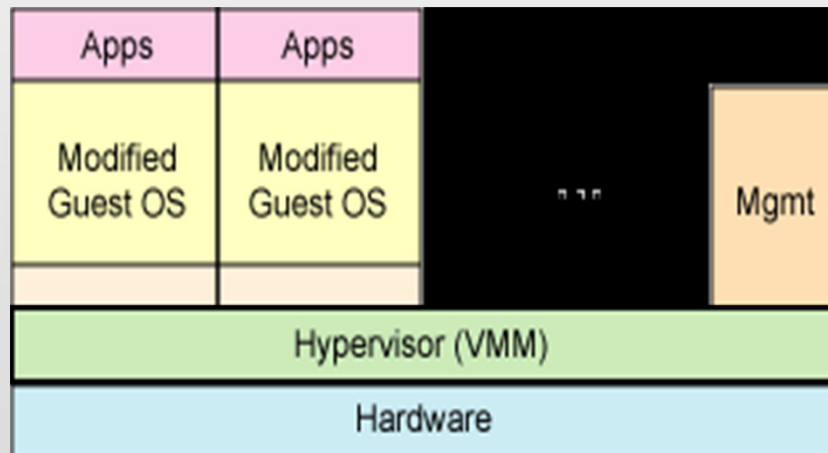
Emulación



V. completa



Paravirtualización



A nivel de SO

