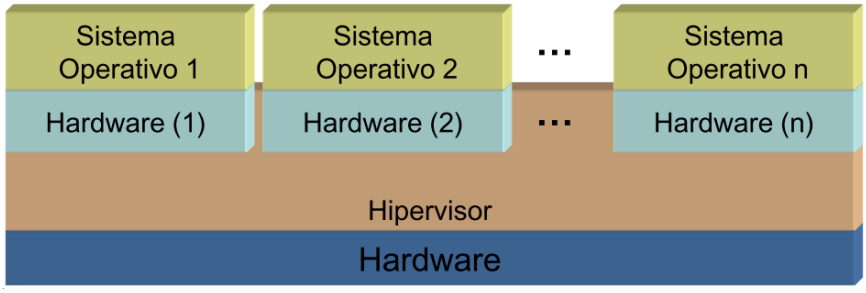
TEMA 3. VIRTUALIZACIÓN

Contenido

CONCEPTOS GENERALES

Un **hipervisor** es un software que simula una o varias máquinas (virtuales) idénticas (o no) a la máquina real.



Comentado [id1]: Sobre hipervisor se puede ejecutar en general cualquier programa que se ejecute sobre máquina real: en particular, un SO. Como se verá más abajo, esto es lo que se llama un hipervisor tipo 1

Utilidad de los hipervisores:

- Ejecución simultánea de varios SO en una misma máquina
- Explotación de Mainframes: múltiples máquinas adaptadas cada una a las necesidades de usuario (escalabilidad) -> Hardware as a Service
  Ejemplo: hospedaje web mediante servidores virtuales dedicados
- Computación en la nube
- Reducción consumo energético ya que no tenemos n hardwares, sino un hardware que simula esos n.
- Despliegue de aplicaciones dentro de máquinas virtuales

Tipos de hipervisores:

- Hipervisores tipo I
- Hipervisores tipo II
- Paravirtualizadores

HIPERVISORES TIPO I

El hipervisor se ejecuta directamente sobre el hardware anfitrión y tiene que resolver parte de la funcionalidad de un SO en el anfitrión:

- Arranque de la máquina
- Gestión de dispositivos, Interfaz de usuario

Para poder construir el hipervisor de tipo 1, el procesador deber ser **virtualizable**:

- Soportar virtualización por hardware
- Al ejecutar una instrucción privilegiada en modo usuario se debe generar una excepción

Comentado [id2]: VMCS = vCPU (VMCS, en intel)

Virtualización: mayor aprovechamiento de los recursos de computación

Sobre un solo servidor hay múltiples máquinas virtuales

El hipervisor **simula** memoria, dispositivos y procesadores para otras máquinas.

Ejemplos de hipervisores tipo 1: Microsoft Hyper-V, IBM z/VM, KMV

Simulación de la memoria

Cada máquina virtual simula un espacio físico virtual mediante paginación anidada.

Simulación interfaz de usuario

- Cada máquina virtual simula su **memoria de vídeo**: Cuando el operador quiere ver la pantalla de una máquina virtual, el hipervisor vuelca su memoria de vídeo sobre la memoria de video del anfitrión.
- El hipervisor dirige los eventos de teclado y ratón del anfitrión al procesador de la máquina virtual que el operador desea controlar

Simulación unidades de disco

Se **mapean** sobre:

- Una unidad o una partición en anfitrión
- Un **archivo estático** o **dinámico** en el anfitrión en el que se encuentra toda la información.

Comentado [id3]: •**Dinámicos**: el archivo está indexado por número de bloque, y los datos asociados a dicho número de bloques es el contenido de dicho bloque en el disco virtual. En este caso, lo normal es que el archive sea de crecimiento dinámico, es decir, que inicialmente sea un archive vacío que incremente su tamaño conforme el SO invitado añada información a la unidad virtual.
•**Estáticos**: el archivo tiene desde su creación el mismo tamaño que la unidad virtual a la que simula. En este caso, la organización puede ser indexada o más probablemente directa, lo que hace el acceso más eficiente

Simulación del adaptador de red

Podemos comunicar las máquinas virtuales entre ellas o con el afitrión. Podemos simular una red utilizando:

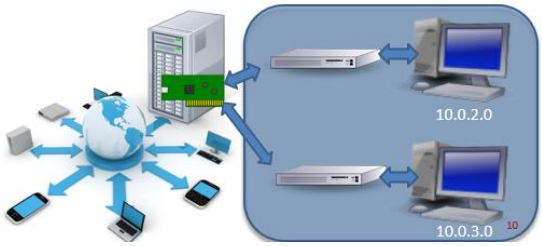
Bridged Networking

- Máquina virtual comparte adaptador de red de anfitrión
- Anfitrión y máquina virtual están conectadas a la misma red
- Máquina virtual y anfitrión usan diferente MAC y diferente IP

Network Address Translation (NAT)

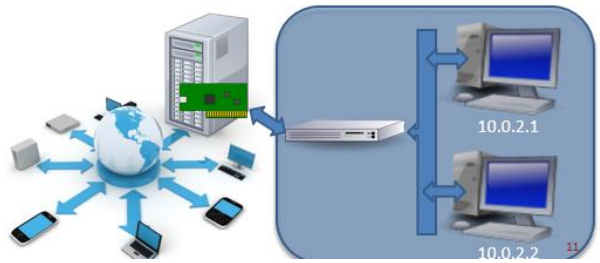
- Anfitrión simula un router a cada máquina virtual
- Máquinas virtuales usan direcciones IP privadas
- Cada máquina virtual en una red distinta

Comentado [id4]: Se conectan a la red mediante un router que asigna direcciones por DHCP. Reciben direcciones de la misma red.
Por tanto, las máquinas virtuales:
- •Pueden ver la red externa
- •Pueden ser vistas desde la red externa mediante port forwarding
- •Pueden ver el anfitrión?
- •Pueden verse entre ellas



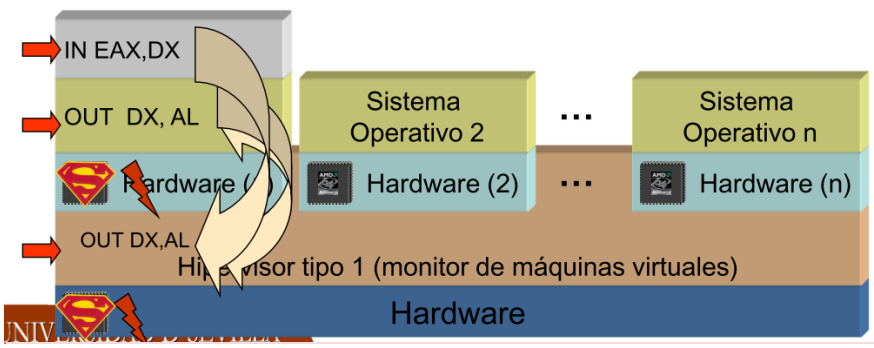
Red NAT

- Hipervisor simula una red privada de máquinas virtuales
- Máquinas virtuales usan direcciones IP privadas
- Hipervisor dispone de servidor DHCP



Simulación del procesador

- Reparte tiempo de procesador entre las máquinas virtuales
- Simula un estado de CPU por cada máquina virtual
- Es fundamental impedir el acceso al hardware del anfitrión
- **Instrucciones privilegiadas**: impide su ejecución en las maquinas virtuales, simulando los cambios de modo de ejecución



Soporte hardware para la virtualización: Intel VT-x

El procesador tiene dos **modos de funcionamiento:**

- **root**: modo para hipervisor, acceso irrestringido
- **non-root**: modo para sistema huésped. Algunas instrucciones provocan transición a modo root

Cada modo root y non-root tiene sus niveles de privilegio (supervisor, usuario e intermedios)

Para cada máquina virtual hipervisor crea una estructura **VMCS** (Virtual Machine Control Structure). VMCS contiene estado e información de control de cada máquina virtual. Si la máquina virtual tiene varios procesadores virtuales, se crea un VMCS para cada uno

En el VMCS de una máquina virtual se puede **definir**:

- Dispositivos a los que puede acceder la máquina virtual directamente-> **Device passhtrough**: requiere que procesador disponga de una unidad IOMMU (Input/Output Memory Management). Se utiliza para adaptadores de res, unidades de almacenamiento, GPU's
- Dispositivos que son simulados (al intentar usarlos, se transfiere control a hipervisor)

Funcionamiento:

Comentado [id5]: CPU REAL SÓLO ESTÁ EN MODO SUPERVISOR CUANDO EJECUTA CÓDIGO DEL HIPERVISOR

Supongamos que sobre sistema operativo de máquina 1 se **ejecuta un proceso de modo usuario**
- •Si NO se está ejecutando código del monitor: cpu real en modo usuario.
- •Si se está ejecutando código de un proceso: cpu virtual en **modo virtual usuario**.

Supongamos que proceso de modo usuario **ejecuta instrucción priviliegiada**
- •Esto provoca una **excepción** en el único procesador que realmente hay (HW real)
- •El **procesador real pasa a modo supervisor** y se transfiere el **control al hipervisor** (monitor de máquinas virtuales)
- •Éste comprueba qué máquina ha producido la excepción, y comprueba en qué estado está el procesador. Como está en modo virtual usuario, **simula que produjese una excepción**, lo cual tiene como efecto que pasa a **modo supervisor**, y se transfiere el control al SO de la máquina virtual que es quien deberá tratarlo. Dado que ya no se está ejecutando código del monitor, la **CPU real vuelve a pasar a modo usuario.**
- •Este SO virtual, podrá intentar ejecutar una instrucción privilegiada: esto provocará que el **procesador real, que está en modo usuario, provoque una nueva excepción**
- •Como consecuencia, nuevamente, el **procesador real pasa a modo supervisor**, y se transfiere **control al hipervisor** (monitor de máquinas virtuales)
- •Esta comprueba en qué estado está el procesador que ha causado la excepción, y al comprobar que está en modo supervisor**, SIMULA la ejecución de la instrucción**, devolviendo el control a la máquina virtual, que sigue ejecutando su sistema operativo, el cual se cree (el pobre) que está en modo supervisor ejecutando instrucciones privilegiadas
- •Será el sistema operativo de la máquina virtual quien decida qué hacer con su proceso de usuario

Comentado [id6]: Los procesadores modernos disponen de soporte hardware para la virtualización. En el caso de Intel, esta tecnología se llama VT-x y en el caso de AMD se llama AMD-V

Comentado [id7]: IOMMU (Input/Ouput Memory Management Unit): Es una unidad de manejo de memoria que conecta un bus de entrada/salida con la memoria principal.

- Hipervisor inicia modo VMX con instrucción MVXON
- Para cada máquina virtual crea los VMCS
- Lanza las máquinas virtuales (instrucción VMLAUNCH)
- Cada vez que un procesador virtual ejecuta una instrucción no permitida en su VMCS: transfiere control a rutina de hipervisor que la simula
- Hipervisor puede terminar en cualquier momento (instrucción VMXOFF)

Hipervisores tipo II

El hipervisor se ejecuta como un proceso más dentro de un sistema operativo anfitrión. Simula las máquinas virtuales interpretando las instrucciones que éstas deben ejecutar.

Ejemplos de hipervisores tipo 2: QEMU, VMWare, VirtualBox, BOCHS

Simulación memoria, dispositivos, interfaz de usuario

Mismas técnicas que **hipervisores tipo I**

Simulación del procesador

- Interpretación de instrucciones una a una: ¡**prohibitivo**!
- Hipervisor analiza bloques de código antes de su ejecución: Cada **bloque básico** (secuencia de instrucciones que no modifican el flujo de control (no contienen saltos, interrupciones,…) se procesa de la siguiente forma:
    - **Instrucciones no privilegiadas**: se permite su ejecución directa
    - **Instrucciones privilegiadas:** son sustituidas por llamadas al hipervisor, que las simula

  Los bloques básicos procesados se copian a cache de código del procesador

Hipervisores tipo 1 vs tipo 2

Ventajas hipervisores tipo 1: eficiencia, siempre y cuando hagan uso de aceleración hardware

Ventajas hipervisores tipo 2: flexibilidad, se instalan sobre un sistema ya en explotación

Comentado [id8]: Es discutible que un hipervisor tipo 1 sea necesariamente más eficiente que un tipo 2, pues posiblemente el tiempo necesario en atender las excepciones con las que se simulan las instrucciones privilegiadas (acceso a vectores de interrupción, cambios de contexto) sea mayor que el tiempo para simularlas mediante una rutina suficientemente optimizadas.

PARAVIRTUALIZADORES

¿Y si el sistema operativo invitado no tuviese instrucciones privilegiadas?

El paravirtualizador actúa como capa intermedia, modificando el código fuente del sistema operativo invitado sustituyendo instrucciones privilegiadas por llamadas al hipervisor (paravirtualizador). Así, el paravirtualizador ofrece un API a SO invitado y proporciona una interfaz para el hipervisor.

Comentado [id9]: El problema de la **eficiencia en los hipervisores** consiste en que se emplea tiempo o bien en atrapar y simular las instrucciones privilegiadas (hipervisores tipo 1) o bien en procesar los bloques básicos de código y reemplazar las instrucciones privilegiadas por llamadas al hipervisor en tiempo de ejecución.

Sin embargo, el sistema operativo invitado debe estar compilado para ejecutarse sobre paravirtualizador

¿Y si quisiéramos ejecutarlo sobre una maquina real?

Para ello, se ofrece una capa de abstracción del hardware. La interfaz entre el núcleo y el hardware se encapsula en una **capa VMI** (Virtual Machine Interface, VMWare, 2005). VMI ofrece un API único al núcleo. Su implementación puede enlazarse con bibliotecas que implementen sus funciones para diferentes paravirtualizadores, hardware real

Nueva interfaz: paravirt-ops (VMWare, XEN, IBM, RedHat)

- **Ventaja**: eficiencia.
- **Inconveniente**: no es transparente para SO huesped
- **Solución**: hipervisor híbrido capaz de actuar como paravirtualizador si el SO huésped lo permite

Ejemplo: Xen

HIPERVISORES HÍBRIDOS

Para solucionar el incoveniente del paravirtualizador, se utiliza un hipervisor híbrido que toma las mejores características del hipervisor tipo 1, 2 y del paravirtulizador:

Hipervisor tipo II:

- Opción más flexible: se instala sobre sistema operativo anfitrión
- No es estrictamente necesario un soporte hardware para virtualización ni siquiera que hardware sea virtualizable

Hipervisor tipo I:

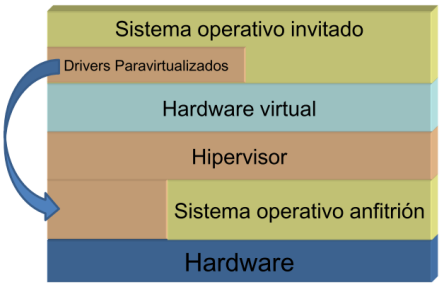
No obstante, el hardware hoy en día siempre soporta virtualización y con dicho soporte hipervisor tipo I es mucho más eficiente

Paravirtualizador:

Un virtualizador instalado sobre un sistema operativo anfitrión puede instalar parte del hipervisor como módulo en Linux o como driver en Windows. Y así, aprovechar el soporte hardware (como **hipervisor tipo I)**

El desarrollador del hipervisor puede crear drivers para los SO invitados que en lugar de intentar usar el hardware virtual usen directamente el hipervisor de esta forma, se añade paravirtualización.

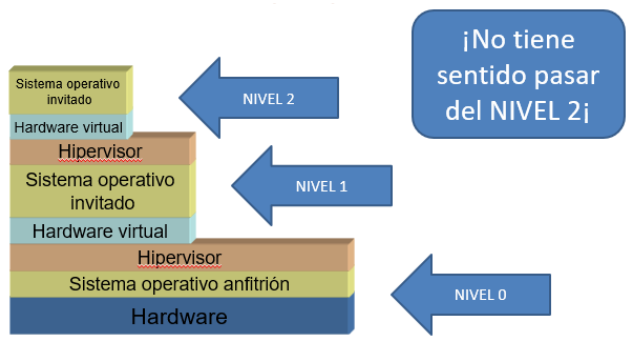
Ejemplos de hipervisores híbridos: VMWare (VMWare Tools), VirtualBox (Vitualbox Guest, Additions), KVM (virtio drivers)



Virtualización anidada

La **virtualización anidada** se da cuando si un hipervisor es capaz de virtualizar VMCS o IOMMU, una máquina virtual puede a su vez convertirse en máquina anfitrión. Una máquina virtual puede proporcionar también servicios de virtualización.

Cada sistema operativo se convierte en un "nivel"



Comentado [id10]: No tiene sentido pasar del nivel 2 pues a partir de este estaríamos montando más maquinas virtuales sobre otras máquinas virtuales, y habría que simular memoria, dispositivos, procesadores en todas las máquinas virtuales -> se necesitarían demasiados recursos

No todos los hipervisores soportan virtualización anidada

Ejemplos de ipervisores que la soportan: KVM, Hyper-V en versiones Windows Server 10 Datacenter, VMWare ESXi, XEN (a partir de versión 4.4), VirtualBox a partir de versión 6 y sólo con AMD-V