

Sistemas Operativos

Virtualización

Curso 2025

Facultad de Ingeniería, UDELAR

Agenda

1. Introducción
2. Requerimientos para la virtualización
3. Virtualización de CPU
4. Virtualización de la memoria
5. Virtualización de la E/S
6. Seguridad en la Virtualización
7. Máquinas Virtuales vs Contenedores

Introducción

Introducción

- A efectos de aumentar la confiabilidad de un centro de cómputos es usual mantener varios servidores independientes (ej: correo, FTP, web, DBMS, etc).
- Permite mantener estabilidad en el centro de cómputos
- Si un servidor cae, no falla todo el sistema
- Altos costos de mantenimiento (hardware duplicado)
- Baja eficiencia en uso de recursos (CPU, discos, memoria, etc)
- El hardware evoluciona rápidamente y muchas veces se desea mantener en funcionamiento software diseñado sobre arquitecturas actualmente obsoletas
- En un equipo de desarrollo se desea testear software sobre un conjunto importante de sistemas operativos (y sus diferentes versiones)

Introducción

- Solución: **Virtualización**
- Permite que un único equipo (hardware) soporte N máquinas virtuales (VM, Virtual Machine)
- Cada máquina virtual es independiente
- Ejecuta su propio SO
- Maneja los recursos del sistema como si fuera una máquina independiente
- Permite migrar datos fácilmente de un equipo a otro (ej: balanceo de carga)
- Permite ejecutar aplicaciones heredadas (*legacy*) en sistemas operativos que no funcionan en el hardware actual.
- La virtualización puede ser de tipo completo (máquinas virtuales) o de tipo parcial (contenedores).

Introducción

- Virtualización introducida en 1972 por IBM en el sistema VM/370

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

- SO: CMS (Conversational Monitor System)
- Cada CMS/App ejecuta sobre una máquina 370 virtual
- Cuando CMS intenta utilizar una instrucción privilegiada, ésta es atrapada por el VMM (VM/370)

- **Hipervisor**: ejecuta en modo kernel y da soporte a las máquinas virtuales
- El sistema operativo que ejecuta en la máquina virtual se denomina Sistema operativo invitado (*Guest OS*)
- El hipervisor realiza las operaciones que requieren privilegios en nombre del SO invitado

Requerimientos para la virtualización

Requerimientos para la virtualización

- Una máquina virtualizada debe comportarse igual que una máquina real
- Debe tener dos modos: kernel y usuario
- **Instrucciones sensibles** (Popek, Goldberg):
 - Operaciones E/S
 - Configuración de MMU
 - Administración de interrupciones
- Las instrucciones sensibles **deben** ejecutarse en modo kernel
- **Instrucciones privilegiadas**: disparan una trap si son ejecutadas en modo usuario
- Para que un sistema pueda soportar virtualización las instrucciones sensibles deben ser un subconjunto de las instrucciones privilegiadas

Requerimientos para la virtualización

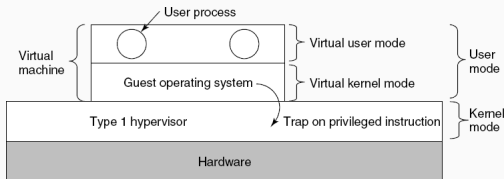
- Intel 386 (486, Pentium, etc hasta 2005) **NO** cumple dicha propiedad
 - Se ignoran instrucciones sensibles en modo usuario (POPF no modifica bit de habilitación de interrupciones)
 - Hay instrucciones que pueden leer el estado sensible en modo usuario sin producir una trap.
 - Ej: El SO virtualizado puede detectar que está en realidad en modo usuario
- Intel con tecnología VT (*Virtualization Technology*) agrega soporte para generar trap al ejecutar instrucciones sensibles.

Virtualización de CPU

- Se definen varios mecanismos de virtualización
 - Hipervisores de tipo 1 (y tipo 0)
 - Hipervisores de tipo 2
 - Paravirtualización
 - Emulación de plataforma
 - Virtualización de aplicaciones
 - Contenedores de aplicaciones

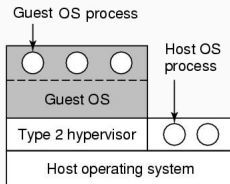
Virtualización de CPU - Hipervisores de tipo 1

- El hipervisor de tipo 1 ejecuta en modo kernel
- Cada VM se ejecuta como un proceso de usuario en modo usuario
 - Modo kernel virtual
 - Modo usuario virtual
- Cuando la VM ejecuta una instrucción sensible, se produce una trap que procesa el hipervisor.
 - Instrucción sensible ejecutada por SO invitado, se ejecuta a través del hipervisor
 - Instrucción sensible ejecutada por programa de usuario en VM, emula funcionamiento del hardware real



Virtualización de CPU - Hipervisores de tipo 2

- El hipervisor de tipo 2 ejecuta en modo usuario como un proceso más del SO anfitrión
- Permite la virtualización en arquitecturas que no cumplen con las hipótesis de Popek & Goldberg
- Aplican **traducción binaria**
 - El hipervisor analiza el flujo de ejecución (bloques de código) y “traduce” las instrucciones sensibles por llamadas al hipervisor.
 - Los bloques traducidos son ejecutados por la CPU directamente.



Virtualización de CPU - Hipervisores de tipo 2

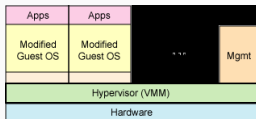
- Todas las instrucciones sensibles se sustituyen mediante llamadas a procedimientos que emulan estas instrucciones.
- Cada bloque de código es traducido y almacenado en un cache
 - Un único punto de entrada por bloque
 - Sin saltos ni llamadas ni interrupciones
 - Cada bloque termina en un salto, llamada o interrupción
 - Se inspecciona el bloque y se traducen las instrucciones sensibles por llamadas al hipervisor, y la instrucción final (salto, llamada o interrupción) por llamada al hipervisor

Virtualización de CPU - Hipervisores de tipo 1 vs 2

- Comparación de performance de hipervisores de 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)
 - **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 instrucción sensible
- Hipervisores de tipo 1 también pueden realizar traducción binaria
- Ejemplos
 - Tipo 1: VMware ESXi, Citrix XenServer y Microsoft Hyper-V
 - Tipo 2: VMware Workstation y Oracle VirtualBox

Virtualización de CPU - Paravirtualización

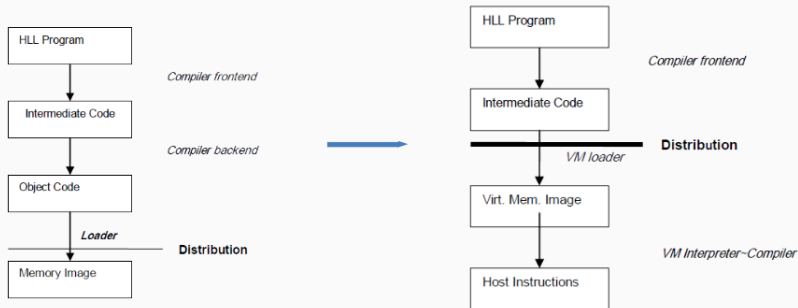
- Hipervisores de tipo 1 y tipo 2 funcionan con SO invitados sin modificar
- Si permitimos que se modifique el SO invitado para evitar las instrucciones sensibles (llamando, en cambio, al hipervisor) obtenemos paravirtualización.
- El SO invitado realiza llamadas al hipervisor en vez de ejecutar instrucciones sensibles a través de una API, ej VMI (Virtual Machine Interface)
- Ejemplo: VMI Linux
- Si el hipervisor solo soporta SO invitados paravirtualizados estamos ante un microkernel



Virtualización de CPU - Emulación de plataforma - Virtualización de aplicaciones

- Emulación de plataforma
 - Permite crear máquinas virtuales con un hardware distinto del nativo
 - Lento, debe traducir todas las instrucciones del hardware destino al nativo, emular periféricos, etc
 - Ejemplo: QEMU
- Virtualización de aplicaciones
 - Aumento de portabilidad (diferentes SO y arquitecturas de hardware)
 - Ej: Java VM, .NET CLR

Virtualización de CPU - Virtualización de aplicaciones



Virtualización de la memoria

Virtualización de la memoria

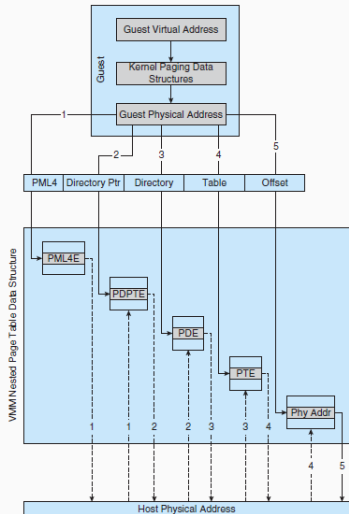
- Manejo de memoria virtual
 - Tabla de páginas (multinivel) en VM
 - Debe mapearse a tablas de páginas del hardware real
- Ej: SO invitado A crea tabla de páginas con la siguiente asignación
 - $5 \Rightarrow 10, 8 \Rightarrow 11, 2 \Rightarrow 12$
- SO invitado B ahora crea tabla de páginas con la siguiente asignación
 - $4 \Rightarrow 10, 5 \Rightarrow 11, 6 \Rightarrow 12$
- El VMM (hipervisor) debe mantener una tabla de páginas oculta (*shadow*) porque no puede darle los mismos marcos a dos VMs.

Virtualización de la memoria

- La creación de la tabla de páginas es una acción “sensible” (modifica MMU que genera trap), pero la posterior actualización de la tabla no es sensible.
- **Posible solución:** marcar las paginas de tabla como de solo lectura
- Caso SO paravirtualizado:
 - Se cambian las instrucciones de manejo de MMU por llamadas al hipervisor
 - Las actualizaciones sobre la tabla de páginas se pueden hacer en modo batch, llamando al hipervisor luego que el SO invitado ha realizado todos los cambios.

Virtualización de la memoria - soporte de hardware

- Implementa tabla de páginas anidada por hardware (*nested page table*)



Virtualización de la E/S

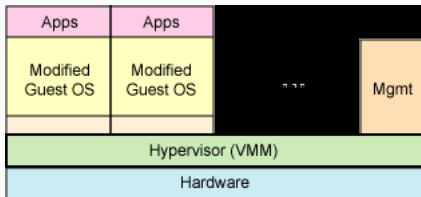
Virtualización de la E/S

- Cada VM piensa que tiene todo el hardware disponible para sí
 - Ej: discos, impresoras etc
- Las operaciones de E/S son sensibles por lo que son tratadas por el hipervisor
- Para algunos recursos es preferible virtualizar el dispositivo
 - Discos representados como un archivo en el FS del SO anfitrión
 - Permite utilizar nuevo hardware sobre SO que no saben manejarlo

- DMA utiliza direcciones absolutas (físicas), por lo que deben ser traducidas por el hipervisor antes de efectivizarse el DMA.
 - MMU de E/S virtualizada
- Direcciones de red (*IP address*)
 - Bridging
 - Network address translation (NAT)
- También hay que mapear las interrupciones al guest correspondiente
 - Con soporte de HW la interrupción se envía a un core que está corriendo un thread de la VM que corresponda.

Virtualización de la E/S

- Otro método es utilizar una de las máquinas virtuales para que refleje la E/S de todas las VM
- A esta VM a veces se le llama **dominio o**



- Mayor facilidad para este esquema en sistemas paravirtualizados
- Hipervisores de tipo 2 pueden utilizar los drivers del SO anfitrión
- Hipervisores de tipo 1 pueden utilizar los drivers del dominio o

Otros escenarios de uso para virtualización

- **Dispositivos virtuales** (*virtual appliances*)
 - Empaquetan un software junto con sus dependencias dentro de una VM
 - El usuario final solamente debe ejecutar la VM y con ella el programa que desea
- **Máquinas virtuales en CPUs multinúcleo**
 - Permiten definir multiprocesadores virtuales
 - El programador puede determinar cuantas CPUs necesita para su software y armar una configuración de VMs para dar soporte a dicha configuración

Seguridad en la Virtualización

- La virtualización puede mejorar la seguridad al aislar las máquinas virtuales entre sí.
- Sin embargo, también introduce nuevos riesgos, como la fuga de información a través del hipervisor.
- Es crucial mantener el hipervisor actualizado y protegido contra amenazas.

Máquinas Virtuales vs Contenedores

- Una Máquina Virtual (VM) es una emulación de un sistema de computadora.
- Las VMs se ejecutan en un software de hipervisor que proporciona la virtualización del hardware subyacente.
- Cada VM tiene su propio sistema operativo completo.
- Ejemplos: VMware, VirtualBox.

Beneficios y Desventajas de las Máquinas Virtuales

- Beneficios: aislamiento completo del sistema operativo, flexibilidad para ejecutar múltiples sistemas operativos, mayor seguridad
- Desventajas: mayor uso de recursos del sistema, puede requerir más tiempo de administración y mantenimiento, rendimiento ligeramente inferior debido a la emulación del hardware

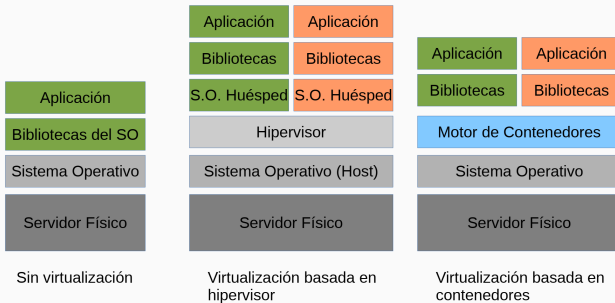
Contenedores

- Un contenedor es una unidad de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable de un entorno de computación a otro
- Solo hay una instancia del sistema operativo compartida por todos los contenedores
- Solo se pueden usar aplicaciones que funcionen en el mismo sistema operativo
- El hardware no está virtualizado pero si lo están los recursos que asigna el sistema operativo (cpu, memoria, red, etc.)

Beneficios y Desventajas de los Contenedores

- Beneficios: menor uso de recursos que las VMs, mayor portabilidad y eficiencia, más adecuados para microservicios y aplicaciones basadas en la nube, inicio más rápido
- Desventajas: menos aislamiento que una VM, dependencia del sistema operativo del host, seguridad ligeramente inferior debido al compartimiento del kernel

Comparación



Implementaciones de Máquinas Virtuales y Contenedores

- Máquinas Virtuales
 - VMware Workstation
 - Oracle VirtualBox
 - Microsoft Hyper-V
 - KVM (Kernel-based Virtual Machine)
- Contenedores
 - Docker
 - containerd (Kubernetes)
 - LXC (Linux Containers)

Implementaciones de Cloud Computing

- Amazon Web Services (AWS)
 - Máquinas Virtuales: Amazon Elastic Compute Cloud (EC2) (Xen y Nitro)
 - Contenedores: Amazon Elastic Container Service (ECS), Amazon Elastic Kubernetes Service (EKS) y AWS Fargate (Docker)
- Google Cloud Platform (GCP)
 - Máquinas Virtuales: Google Compute Engine (GCE) (KVM)
 - Contenedores: Google Kubernetes Engine (GKE)
- Microsoft Azure
 - Máquinas Virtuales: Azure Virtual Machines (VM) (Hyper-V)
 - Contenedores: Azure Kubernetes Service (AKS)