Resumen Virtualización

Tomás Felipe Melli

Noviembre 2024

Índice

1 Virtualización

La virtualización es una tecnología que permite abstraer recursos físicos para que se perciban como múltiples instancias de recursos lógicos, optimizando su uso y administración. Pensemos lo siguiente, tenemos un servidor físico, pero necesitamos correr 3 SO diferentes, en vez de comprar 3 compus, se le instala un **hypervisor** que permite crear y ejecutar máquinas virtuales, cada una con su propio SO y configuraciones, compartiendo los recursos del HW físico.

1.1 Los objetivos de la virtualización

- **Portabilidad** : Permite ejecutar software en diferentes sistemas sin modificar el código. Como por ejemplo un programa en Java puede correr en Windows, Linux o Mac gracias a la JVM.
- **Simulación/Testing** : Se pueden crear entornos virtuales para probar software sin afectar el sistema real.
- **Aislamiento** : Se limita el acceso de una aplicación o usuario a una parte del sistema.
- **Particionar el hw** : Se divide un recurso físico en varias unidades lógicas independientes.
- **Agrupar funciones**(consolidation) : Se combinan varias cargas de trabajo en menos servidores para mejorar eficiencia.
- **Protejer ante fallas de hw** : Si un servidor falla, las máquinas virtuales pueden moverse automáticamente a otro.
- **Migración entre hw sin pérdida servicio** : Permite trasladar sistemas en ejecución a otro hardware sin afectar a los usuarios.

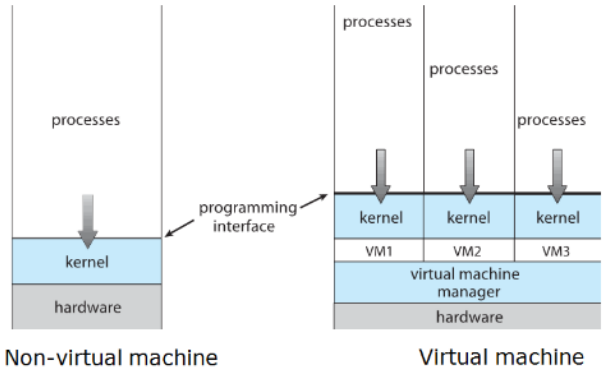
1.2 Hypervisor o VMM (Virtual Machine Monitor)

Mencionamos el **hypervisor** o **VMM**. *Pero, qué es esto?*
El **hypervisor** es un software o firmware que permite la creación y gestión de máquinas virtuales. Actúa como una capa entre el hardware físico y los sistemas operativos virtualizados, permitiendo que múltiples VMs se ejecuten de forma independiente sobre el mismo hardware. Existen dos tipos, **bare-metal** y **hosted**. El primero se ejecuta directamente sobre el hardware sin un sistema operativo intermedio y el segundo sobre un sistema operativo existente (como VirtualBox).

1.3 Cuándo un hypervisor es eficiente ?

En 1974, Popek y Goldberg establecieron tres características clave que debe cumplir un Hypervisor para ser considerado eficiente :

- **Fidelity** : Una máquina virtual debe comportarse de la misma manera que el sistema real cuando se ejecuta directamente en el hardware.
- **Performance** : La mayoría de las instrucciones deben ejecutarse directamente en el hardware sin intervención del VMM, evitando pérdidas de rendimiento significativas.
- **Safety** : El VMM debe tener control total sobre los recursos físicos y evitar que una VM interfiera con otras o con el hardware. Si una VM intenta acceder a memoria fuera de su espacio asignado, el VMM debe intervenir y bloquear la acción.



1.4 Simulación y Emulación

Una manera de lograr la virtualización es mediante la **simulación**. En un sistema **anfitrión**, se crea una representación artificial del sistema **huésped** mediante variables de estado. Cada instrucción del sistema huésped es interpretada y aplicada sobre este estado simulado. Un ejemplo de esto podría ser un simulador de procesador que imita el comportamiento de una arquitectura distinta a la del hardware real.
La simulación podría ser lenta, ya que cada instrucción debe analizarse e interpretarse antes de ejecutarse, lo que genera una sobrecarga significativa. También hay que simular eventos del hw como *interrupciones, accesos a memoria (DMA), concurrencia.*
En la **emulación de hw** el sistema huésped se ejecuta realmente en la CPU del anfitrión, sin interpretar cada instrucción individualmente. Se emulan los dispositivos de hardware, como tarjetas de red, discos y controladores de video. Cuando la máquina virtual (MV) cree que está accediendo a un dispositivo físico, en realidad está interactuando con un controlador de máquinas virtuales, que:

- Intercepta las operaciones de entrada/salida (E/S).
- Traduce esas operaciones para que se ejecuten en el hardware real o en una versión emulada del dispositivo.

La mayoría del código se corre con **traducción binaria** (una técnica para convertir instrucciones de un sistema en otro. Se usa principalmente cuando el código del sistema huésped no puede ejecutarse directamente en el hardware del anfitrión. Como QEMU que emula una CPU ARM en un procesador x86 (de orga II)).
La emulación de hw introduce ciertos problemas como:

- **Privilegios** : Toda la máquina virtual corre en modo usuario, lo que impide acceso directo a instrucciones privilegiadas.
- **Velocidad de acceso** : Los accesos a hardware reales pueden ser mucho más lentos que en un sistema físico. Esto es porque cada acceso a un dispositivo requiere pasar por el controlador de la máquina virtual.

1.5 Virtualización Asistida por HW

La virtualización asistida por hardware es una tecnología que permite ejecutar máquinas virtuales (VMs) de manera más eficiente mediante soporte directo en el procesador. Esta mejora soluciona varios problemas de la virtualización tradicional basada en software.
Qué problemas resuelve?

- **Ring Aliasing** : En un sistema normal, el kernel opera en un nivel de privilegio superior (ring 0), mientras que las aplicaciones lo hacen en un nivel inferior (ring 3). En una VM, el SO huésped cree estar en ring 0, pero en realidad se ejecuta en un nivel más bajo, lo que puede causar errores de permisos en ciertas instrucciones.

- **Address-Space Compression** : En un sistema sin virtualización, el sistema operativo tiene acceso total al espacio de direcciones de la memoria física. En una VM, el SO huésped cree que tiene control total de la memoria, pero en realidad es gestionada por el hipervisor. Como todas las VMs corren dentro del hipervisor, comparten el mismo espacio de direcciones desde la perspectiva del anfitrión. Si el hipervisor no controla correctamente el acceso a la memoria, una VM podría sobrescribir memoria perteneciente al hipervisor u otras VMs.
- **Non-Faulting Access to Privileged State** : Algunas instrucciones privilegiadas generan una excepción (trap) si se ejecutan en un nivel sin permisos, permitiendo que el hipervisor las capture y emule. Otras instrucciones no generan excepciones, simplemente fallan en silencio o devuelven resultados incorrectos, lo que complica su virtualización sin asistencia de hardware. Si el hipervisor no puede detectar que el SO huésped intentó ejecutar una instrucción privilegiada, no tiene forma de intervenir y emular su ejecución.
- **Interrupt Virtualization** : En una CPU física, las interrupciones son manejadas directamente por el sistema operativo. En una máquina virtual, el SO huésped cree que está manejando directamente las interrupciones, pero en realidad deben ser interceptadas por el hipervisor. Esto genera problemas, porque el SO huésped no puede interactuar directamente con el hardware real. Las interrupciones pueden llegar en momentos en que la VM no está ejecutándose. Sin virtualización asistida por HW, el hipervisor tiene que interceptar y reenviar manualmente todas las interrupciones a la VM. Esto introduce latencias y puede afectar el rendimiento del sistema.
- **Access to Hidden State** : Los procesadores tienen registros internos y estados que no son accesibles desde el software normal. Un hipervisor necesita conocer el estado completo del procesador para gestionar adecuadamente la VM. Sin acceso a estos registros ocultos, la virtualización puede ser inconsistente o generar errores difíciles de detectar.
- **Ring Compression**: Dado que el SO huésped y sus aplicaciones corren en el mismo nivel de privilegio, se pierde la separación de seguridad entre kernel y usuario.
- **Frequent Access to Privileged Resources**: Algunas instrucciones privilegiadas generan un trap cuando son ejecutadas por la VM. Esto significa que cada vez que la VM ejecuta una de estas instrucciones, el control se transfiere al hipervisor. Si estas instrucciones se usan con mucha frecuencia, esto puede causar una sobrecarga importante.

Para resolverlos, los fabricantes, en el caso de **intel** agregaron al procesar extensiones **VT-x** que proveen dos modos :

- **VMX root** : Se usa para ejecutar el hipervisor. Tiene acceso completo al hardware y puede controlar la ejecución de las VMs.
- **VMX non-root** : Se usa para ejecutar las máquinas virtuales (huéspedes). Parece un entorno normal de CPU, pero con ciertas restricciones. No puede ejecutar instrucciones privilegiadas directamente, evitando que interfiera con el hipervisor.

Se proveen 10 instrucciones para alternar entre ambos modos :

- VM Entry $\rightarrow$ Se usa para pasar el control desde el hipervisor a la máquina virtual.
- VM Exit $\rightarrow$ Se usa cuando la VM ejecuta una operación que requiere la intervención del hipervisor (como acceder a un recurso privilegiado).

Se agrega a su vez la **Virtual Machine Control Structure** que es una estructura en memoria que almacena y gestiona el estado de una máquina virtual. Se compone de :

- **Campos de control** : Especifican qué operaciones están permitidas para la máquina virtual. Definen qué interrupciones puede recibir la VM y qué puertos de E/S puede acceder.

- **Estado completo del huésped** : Almacena registros, estado de la memoria y configuraciones del sistema operativo dentro de la VM. Permite que la ejecución de la VM se reanude en el mismo punto en que se detuvo.
- **Estado completo del anfitrión** : Contiene información sobre el hipervisor y el hardware real. Se usa cuando la CPU debe salir del contexto de la VM y regresar al hipervisor.

La idea es que cuando el huésped ejecuta una instrucción "prohibida" (intenta acceder a un recurso privilegiado del hardware) según la VMCS, esto genera un VM Exit. El control pasa al hipervisor.La CPU cambia automáticamente a VMX Root Mode. Luego, el hipervisor decide si emula, ignora o bloquea la operación. El hipervisor actualiza la VMCS y reanuda la ejecución : Si la operación es válida, el hipervisor la simula y devuelve el control a la VM. Si es inválida, puede bloquear la VM o generar un error.

1.6 Desafíos

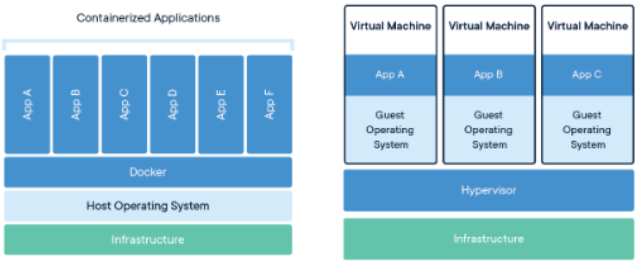
- **Acceso eficiente al disco** : Los sistemas operativos y los sistemas de archivos (FS) están diseñados para acceder al disco de manera eficiente, con cachés, planificadores de E/S y otras optimizaciones. Sin embargo, en un entorno virtual, el acceso al disco es indirecto, ya que la VM no accede al hardware real, sino a un disco virtual gestionado por el hipervisor.
- **Picos de carga en múltiples CPU** : Cuando varias VMs comparten un mismo procesador físico, pueden generarse picos de carga que afectan el rendimiento de todas las máquinas. Si el hipervisor no distribuye bien los recursos, una VM con alta carga puede ralentizar a otras. Esto es especialmente crítico en entornos con virtualización en la nube, donde los recursos son compartidos.
- **Único punto de falla (Single Point of Failure, SPOF)** : Si falla una pieza de hardware clave (como el servidor físico que aloja las VMs), todas las VMs en ese servidor caen. Esto es un gran riesgo en entornos de producción y puede afectar servicios críticos.
- **falla de una VM no afecta al hardware** : en un entorno virtual, una falla dentro de una VM (como un crash del SO huésped) no afecta al hardware real ni a otras VMs. Sin embargo, si una VM consume demasiados recursos o genera errores críticos, puede afectar indirectamente a otras VMs en el mismo servidor.

1.7 Escenarios de uso de la virtualización

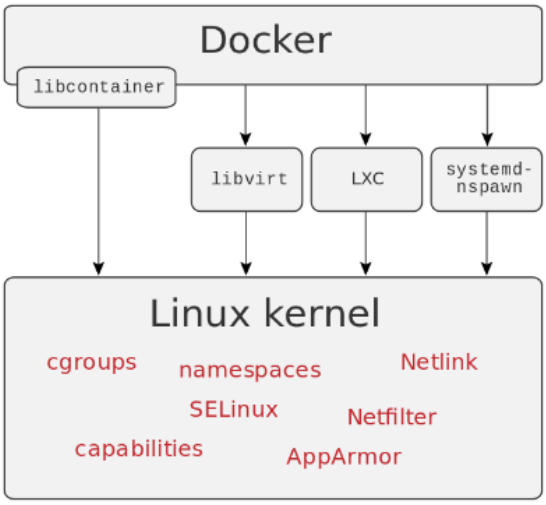
- Correr sistemas viejos
- Aprovechamiento de equipamente
- Desarrollo/testing/debugging
- Abaratar costos

2 Contenedores

A diferencia de las máquinas virtuales, los contenedores comparten el mismo kernel del sistema anfitrión. Existe una capa de administración de contenedores como **docker**, y sobre eso correr las distintas aplicaciones. La ventaja es que no hay $n - copias$ del so y por tanto, consume muchos menos recursos. Además, se puede iniciar una nueva aplicación de manera más rápida.



Para que esto funcione, el SO debe proveer de funcionalidades :



El kernel de Linux proporciona tecnologías clave para la virtualización a nivel de sistema operativo, lo que permite la creación y gestión de contenedores de manera eficiente.

- **Cgroups** : Los Cgroups permiten restringir y asignar recursos del sistema a procesos específicos (Memoria, CPU, I/O(cantidad de operaciones), red (ancho de banda)). Esto es fundamental para evitar que un contenedor consuma todos los recursos del sistema y afecte a otros procesos. Ejemplo : Un servidor con múltiples contenedores puede asignar más CPU a un servicio crítico (como una base de datos) y limitar la memoria de otros servicios menos importantes.
- **Namespaces** : Los Namespaces permiten que un proceso o grupo de procesos tenga su propia visión aislada del sistema operativo, como si fueran entornos independientes. Ejemplo : Un contenedor ejecutando una aplicación web puede creer que es el único proceso corriendo en la máquina, aunque en realidad hay varios contenedores ejecutándose en paralelo.

El kernel de Linux también provee funcionalidades específicas para gestionar redes, comunicación entre procesos (IPC) y permisos de privilegios

- **Netfilter** : es una infraestructura del kernel que permite el filtrado de paquetes de red, creando reglas de firewall, NAT (Network Address Translation) (modifica las direcciones IP de los paquetes, útil para el enmascaramiento de direcciones o la conexión de redes privadas a internet), entre otras. Es fundamental para la seguridad y la gestión del tráfico de red.
- **Netlink** : es un mecanismo de comunicación entre el kernel y los procesos de usuario, específicamente diseñado para interactuar con el sub-sistema de red del kernel. Se utiliza principalmente para obtener información sobre la configuración de la red, realizar cambios en ella y manejar eventos de red.

- **Capabilities** : permiten asignar permisos específicos a un proceso, otorgándole privilegios que normalmente estarían reservados al administrador (root), pero sin otorgar control completo del sistema. Esto es una forma de otorgar privilegios de manera más granular, lo que mejora la seguridad del sistema. Por ejemplo : Un proceso que necesita cambiar la configuración de la red puede recibir la capacidad CAP_NET_ADMIN(permite la administración de interfaces de red) sin necesidad de ser ejecutado como root, limitando así el riesgo de abusos de privilegios.

El kernel de Linux también provee otras tecnologías de seguridad tipo **MAC (Mandatory Access Control)**(es una forma de control de acceso donde las políticas de seguridad son impuestas por el sistema operativo, y no pueden ser modificadas por los usuarios o los programas que se ejecutan)

- **SELinux (Security-Enhanced Linux)** : es una implementación de MAC que proporciona un control de acceso más versátil y detallado, pero también es más complejo de configurar y gestionar.
- **AppArmor**
- **LSM (Linux Security Modules)**

2.1 Orquestación de Aplicaciones Contenerizadas

La orquestación de aplicaciones contenerizadas es **el proceso de automatizar la implementación, gestión y escalabilidad de aplicaciones dentro de contenedores**. Con el aumento del uso de contenedores como Docker, la necesidad de herramientas que administren múltiples contenedores y su interacción de manera eficiente ha dado lugar a plataformas especializadas.

- **Kubernetes** : es una plataforma de código abierto que proporciona una infraestructura robusta para la orquestación de contenedores.
- **OpenShift (y OKD)** : es una plataforma de orquestación basada en Kubernetes pero con algunas mejoras y funcionalidades adicionales, orientadas especialmente a empresas y desarrolladores.

3 Cloud

Algún entorno, propio o de terceros, donde tenemos la facilidad de correr servicios sin preocuparme por el hw que hay detrás.

3.1 Modelos

Existen diferentes modelos de implementación en la nube, como la nube pública, la nube privada y la nube híbrida

- **Pública** : es una infraestructura en la que los recursos informáticos (como servidores, almacenamiento, etc.) son proporcionados y gestionados por un proveedor externo (por ejemplo, Amazon Web Services, Google Cloud Platform, Microsoft Azure). Los usuarios tienen acceso a estos recursos de forma remota a través de internet.

- **Privada** : es una infraestructura de computación en la nube que es utilizada exclusivamente por una sola organización. Los recursos de la nube privada pueden estar en las instalaciones de la empresa o ser gestionados externamente por un proveedor de servicios, pero están dedicados únicamente a esa organización.