8. Virtualización

8.1. Conceptos

Tradicionalmente, las aplicaciones se han ejecutado directamente en un sistema operativo que ejecuta en un computadora personal o en una estación de trabajo (workstation) o un mainframe; y estos equipos sólo ejecutaban un sólo sistema operativo a la vez. Por lo tanto, el desarrollador de aplicaciones tenía que reescribir partes de estas para cada sistema operativo y plataforma en la que se ejecutarían y recibirían apoyo posterior, lo que aumentaba el tiempo de comercialización de nuevas características y funciones, aumentaba la probabilidad de defectos, los esfuerzos de pruebas de calidad y, por lo general, conducía a un aumento del precio. Para dar apoyo a varios sistemas operativos, los desarrolladores de aplicaciones necesitaban crear, gestionar y dar apoyo a diversas infraestructuras de hardware y sistemas operativos, un proceso costoso e intensivo en recursos. Una estrategia eficaz para tratar este problema se conoce como **virtualización** de hardware. La tecnología de virtualización permite que un solo PC o servidor ejecute simultáneamente varios sistemas operativos o varias sesiones de un solo sistema operativo. Una máquina con software de virtualización puede alojar numerosas aplicaciones, incluidas las que se ejecutan en diferentes sistemas operativos, en una única plataforma. En esencia, el sistema operativo del anfitrión (o *host*) puede soportar varias máquinas virtuales (*virtual machine*, VM), cada una de las cuales tiene las características de un sistema operativo en particular y, en algunas versiones de virtualización, las características de una plataforma de hardware en particular. Una VM también se denomina máquina virtual de sistema, enfatizando que es el hardware del sistema el que está siendo virtualizado. Podemos imaginarnos la posición que ocupa si observamos la Figura 8.1.

> En Ciencias de la Computación, la virtualización se refiere al acto de crear a través de software una versión virtual (en lugar de real) de algo, incluyendo plataformas de hardware de computadoras virtuales, dispositivos de almacenamiento y recursos de red de computadoras.

Popek y Goldberg (1974) definieron originalmente una «máquina virtual» como «un duplicado eficiente y aislado de una máquina de computadora real». Pero, para Smith y Ravi Nair (2005), el uso actual incluye máquinas virtuales que no tienen correspondencia directa con ningún hardware real. El hardware físico del «mundo real» que ejecuta la VM generalmente se conoce como el «*host*» (anfitrión), y la máquina virtual emulada en esa máquina generalmente se conoce como el «invitado». Un *host* puede emular varios invitados, cada uno de los cuales puede emular diferentes sistemas operativos y plataformas de hardware.

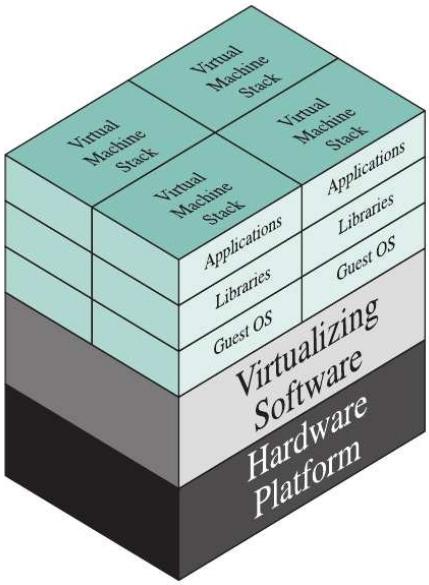


Figura 8.1.: Concepto de máquina virtual

Existen diferentes tipos de máquinas virtuales, cada una con diferentes funciones:

- Las **máquinas virtuales del sistema** (también denominadas máquinas virtuales de **virtualización completa**) proporcionan un sustituto de una máquina real. Proporcionan la funcionalidad necesaria para ejecutar sistemas operativos completos. Un **hipervisor** utiliza la ejecución nativa para compartir y administrar hardware, lo que permite múltiples entornos que están aislados entre sí, pero que existen en la misma máquina física. Los hipervisores modernos usan virtualización asistida por hardware, hardware específico de virtualización, principalmente de las CPU host.
- Las **máquinas virtuales de proceso** están diseñadas para ejecutar programas informáticos en un entorno independiente de la plataforma.

Algunas máquinas virtuales, como QEMU, están diseñadas para emular también diferentes arquitecturas y permitir la ejecución de aplicaciones de software y sistemas operativos escritos para otra CPU o arquitectura. La virtualización a nivel del sistema operativo permite que los recursos de una computadora se particionen a través del núcleo. Los términos no son universalmente intercambiables.

La virtualización comenzó en la década de 1960, como un método para dividir lógicamente los recursos del sistema proporcionados por los ordenadores centrales entre diferentes aplicaciones. Desde entonces, el significado del término se ha ampliado.

La serie CP/CMS de IBM -que ejecutaban sobre equipos IBM System/360-67 como los que vemos en la Figura 8.2-, fueron los primeros sistemas que permitieron la virtualización completa, implementando el tiempo compartido al proporcionar a cada usuario un sistema operativo de un solo usuario, el Sistema de Monitor de Conversación (*Conversational Monitor System*, CMS).

A diferencia de la memoria virtual, una máquina virtual del sistema autoriza al usuario a escribir instrucciones privilegiadas en su código. Este enfoque tenía ciertas ventajas, como agregar dispositivos de entrada y salida no permitidos por el sistema estándar.

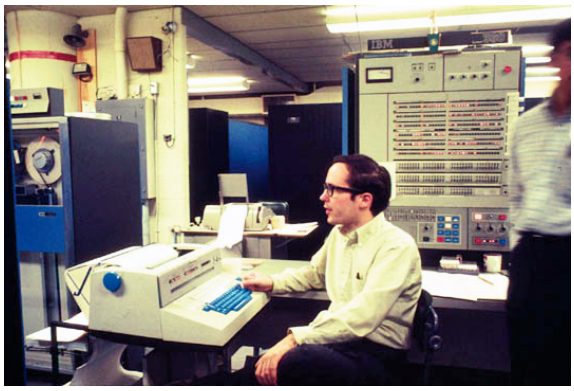


Figura 8.2.: IBM System/360

A medida que evoluciona la tecnología de memoria virtual para propósitos de virtualización, se pueden aplicar nuevos sistemas de **sobrecompromiso de memoria** (en inglés, *memory overcommitment*) para administrar el uso compartido de memoria entre varias máquinas virtuales en el sistema operativo de un ordenador.

> El sobrecompromiso de memoria (*memory overcommitment*) es un concepto en computación que cubre la asignación de más memoria a dispositivos de computación virtual (o procesos) de la que realmente tiene la máquina física en la que están alojados o en la que se ejecutan.

Es posible compartir páginas de memoria que tienen contenidos idénticos entre varias máquinas virtuales que ejecutan en la misma máquina física, lo que puede dar como resultado asignarlas a la misma página física mediante una técnica denominada **fusión de la misma página del núcleo** (en inglés, *kernel same-page merging*, o KSM). Esto es especialmente útil para páginas de sólo lectura, como las que contienen segmentos de código, como es el caso de varias máquinas virtuales que ejecutan el mismo software o similares, bibliotecas de software, servidores web, componentes de middleware, etc. Los sistemas operativos invitados no necesitan compatibles con el hardware del *host*, por lo que es posible ejecutar diferentes sistemas operativos en el mismo equipo (por ejemplo, Windows, Linux o versiones anteriores de un sistema operativo) para admitir software futuro.

8.2. Hipervisores

La virtualización es una forma de abstracción. Al igual que un sistema operativo abstrae los comandos de entrada y salida del disco de un usuario mediante el uso de capas de programa e interfaces, la virtualización abstrae el hardware físico de las máquinas virtuales que soporta. El monitor de máquina virtual (en inglés, *virtual machine monitor*, VMM) o **hipervisor** es el software que proporciona esta abstracción. Actúa como un corredor, o policía de tráfico, actuando como un gestor (o *proxy*, en inglés) para los invitados (VMs) a medida que solicitan y consumen los recursos del anfitrión físico.

Una máquina virtual es una construcción de software que imita las características de un servidor físico. Está configurado con un cierto número de procesadores, una cierta cantidad de RAM, recursos de almacenamiento y conectividad a través de los puertos de red.

Una vez creada la VM, se puede encender como un servidor físico, cargar con un sistema operativo y soluciones de software, y utilizarla como un servidor físico. A diferencia de un servidor físico, este servidor virtual sólo ve los recursos con los que ha sido configurado, no todos los recursos del propio *host* físico. Este aislamiento permite que una máquina *host* ejecute muchas máquinas virtuales, cada una de las cuales ejecuta las mismas o diferentes copias de un sistema operativo, compartiendo RAM, almacenamiento y ancho de banda de red, sin problemas. Un sistema operativo de una máquina virtual accede al recurso que le presenta el hipervisor. El hipervisor facilita la traducción y la entrada y salida de la máquina virtual a los dispositivos del servidor físico y de vuelta a la máquina virtual correcta. De esta manera, ciertas instrucciones privilegiadas que un sistema operativo «nativo» estaría ejecutando en el hardware de sus *hosts* quedan atrapadas y son ejecutadas por el hipervisor como un gestor para la máquina virtual. Esto crea una cierta degradación del rendimiento en el proceso de virtualización, aunque con el tiempo las mejoras tanto en el hardware como en el software han minimizado esta sobrecarga.

Una instancia de VM se define en los archivos. Una máquina virtual típica puede consistir sólo en algunos archivos. Existe un archivo de configuración que describe los atributos de la aplicación virtual máquina. Contiene la definición del servidor, cuántos procesadores virtuales (vCPUs) son asignado a esta máquina virtual, cuánta RAM está asignada, a qué dispositivos de entrada y salida la VM tiene acceso, cuántas tarjetas de interfaz de red (en inglés, *network interface card*, NIC) hay en el servidor virtual, y más. También describe el almacenamiento al que puede acceder la máquina virtual. A menudo ese almacenamiento se presenta como discos virtuales que existen como archivos adicionales en el sistema de archivos físico. Cuando una máquina virtual está encendida o instanciada, se crean archivos adicionales para registro de eventos (en inglés, *logging*), para paginación de memoria y otras funciones. Dado que una VM consiste esencialmente de los archivos, algunas funciones en un entorno virtual se pueden definir de forma más sencilla y rápida que en un entorno físico. Desde los primeros días de los equipos, las copias de seguridad ha sido una función crítica. Dado que las VMs ya son archivos, copiarlas produce no sólo una copia de seguridad de los datos, sino también una copia de todo el servidor, incluido el sistema operativo, y la propia configuración del hardware.

Un método común para desplegar rápidamente nuevas máquinas virtuales es mediante el uso de plantillas. Una plantilla proporciona un grupo estandarizado de configuraciones de hardware y software que se pueden utilizar para crear nuevas VMs configuradas con esas características. La creación de una nueva VM a partir de una plantilla consiste en proporcionar identificadores únicos para la nueva VM, y hacer que el software de aprovisionamiento construya una VM a partir de la plantilla y añadir los cambios de configuración como parte de la implementación.

8.2.1. Funciones del hipervisor

Las principales funciones que desempeña un hipervisor son las siguientes:

- Gestión de la ejecución de las máquinas virtuales: Incluye la programación de las

VM para su ejecución, la gestión de la memoria virtual para garantizar el aislamiento de las VM de otras VM, el cambio de contexto entre varios estados del procesador. También incluye el aislamiento de las máquinas virtuales para evitar conflictos en el uso de recursos y la emulación de mecanismos de temporización e interrupción.

- Emulación de dispositivos y control de acceso: Emular todos los dispositivos de red y de almacenamiento (por bloques) que esperan los diferentes controladores nativos de las máquinas virtuales, mediando el acceso a los dispositivos físicos por parte de las diferentes máquinas virtuales.
- Ejecución de operaciones privilegiadas por parte del hipervisor para las máquinas virtuales huéspedes: Algunas operaciones invocadas por los SOs huéspedes, en lugar de ser ejecutadas directamente por el hardware del host, pueden tener que ser ejecutadas en su nombre por el hipervisor, debido a su naturaleza privilegiada.
- Gestión de las máquinas virtuales (también llamada gestión del ciclo de vida de las máquinas virtuales): Configuración de las máquinas virtuales invitadas y control de los estados de las máquinas virtuales (por ejemplo, Inicio, Pausa y Parada).
- Administración de la plataforma de hipervisor y del software de hipervisor: implica la configuración de parámetros para las interacciones del usuario con el host del hipervisor, así como software de hipervisor.

8.2.2. Hipervisor tipo 1

Existen dos tipos de hipervisores, que se distinguen por la existencia de un sistema operativo entre el hipervisor y el *host*. Un hipervisor de tipo 1 (véase la Figura 8.3) se carga como una capa de software directamente en un servidor físico, de forma muy parecida a como se carga un sistema operativo. El hipervisor de tipo 1 puede controlar directamente los recursos físicos del *host*. Una vez instalado y configurado, el servidor es capaz de soportar máquinas virtuales como invitados.

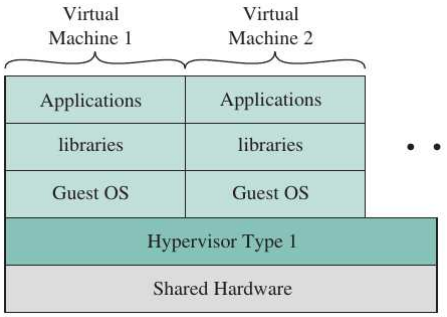


Figura 8.3.: Hipervisor tipo 1

En entornos maduros, donde los *hosts* de virtualización son agrupados para aumentar la disponibilidad y el balanceo de carga (load balancing), un hipervisor puede ser monta-

do en un nuevo anfitrión. Luego, ese nuevo *host* se une a un *clúster* existente, y los VMs pueden moverse al nuevo host sin interrupción del servicio. Algunos ejemplos de hipervisores tipo 1 son VMware ESXi, Microsoft Hyper-V y las diversas variantes de Xen.

8.2.3. Hipervisor tipo 2

Un hipervisor de tipo 2 explota los recursos y funciones de un sistema operativo host y se ejecuta como un módulo de software «encima» del sistema operativo (consulte la Figura 8.4). Depende del sistema operativo para manejar todas las interacciones de hardware en nombre del hipervisor. Algunos ejemplos de hipervisores de tipo 2 son VMware Workstation y Oracle VM Virtual Box.

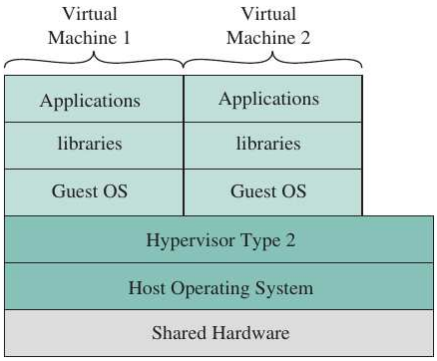


Figura 8.4.: Hipervisor tipo 2

Las diferencias clave entre los dos tipos de hipervisores son las siguientes:

- Normalmente, los hipervisores de tipo 1 funcionan mejor que los de tipo 2. Porque un hipervisor de tipo 1 no compite por recursos con un sistema operativo, hay más recursos disponibles en el host y, por extensión, más máquinas virtuales pueden estar alojado en un servidor de virtualización utilizando un hipervisor de tipo 1.
- Los hipervisores de tipo 1 también se consideran más seguros que los de tipo 2. Las máquinas virtuales de un hipervisor de tipo 1 realizan solicitudes de recursos que se gestionan de forma externa a ese invitado y no pueden afectar a otras máquinas virtuales ni al hipervisor con el que cuentan. Esto no es necesariamente cierto en el caso de las máquinas virtuales en un hipervisor de tipo 2, y un huésped malicioso podría afectar potencialmente a más que a sí mismo.
- Los hipervisores de tipo 2 permiten a un usuario aprovechar la virtualización sin necesidad de dedicar un servidor sólo a esa función. Los desarrolladores que necesitan ejecutar múltiples entornos como parte de su proceso, además de aprovechar el espacio de trabajo productivo personal que proporciona un sistema operativo para PC, pueden hacer ambas cosas con un hipervisor tipo 2 instalado como aplicación en su escritorio LINUX o Windows. Las máquinas virtuales creadas y

utilizadas pueden migrarse o copiarse de un entorno de hipervisor a otro, lo que reduce el tiempo de implementación. y aumentar la precisión de lo que se despliega, reduciendo el tiempo de lanzamiento al mercado de un proyecto.

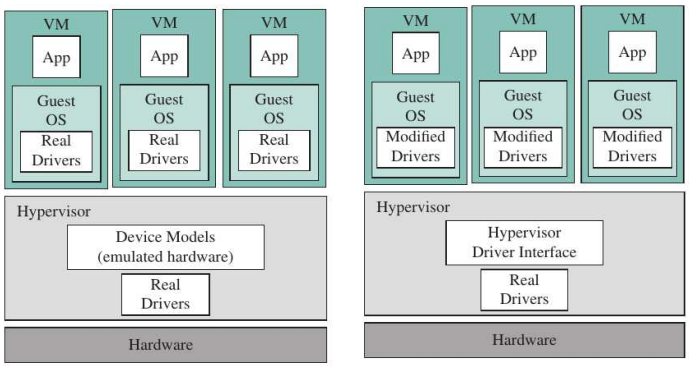
- Los hipervisores de tipo 2 permiten a un usuario aprovechar la virtualización sin necesidad de dedicar un servidor sólo a esa función. Los desarrolladores que necesitan ejecutar varios entornos como parte de su proceso, además de aprovechar el espacio de trabajo productivo personal que proporciona un sistema operativo para PC, pueden hacer ambas cosas con un hipervisor tipo 2 instalado como aplicación en su escritorio LINUX o Windows. Las máquinas virtuales creadas y utilizadas pueden migrarse o copiarse de un entorno de hipervisor a otro, lo que reduce el tiempo de implementación y aumentando la precisión de lo que se implementa, reduciendo el tiempo de comercialización de un proyecto.

8.2.4. Paravirtualización

A medida que la virtualización se hizo más frecuente en las empresas, tanto los proveedores de hardware como de software buscaron formas de proporcionar aún más eficiencia. Como era de esperar, estos caminos condujeron tanto a la virtualización asistida por software como a la virtualización asistida por hardware. La **paravirtualización** es una técnica de virtualización asistida por software que utiliza APIs especializadas para enlazar las máquinas virtuales con el hipervisor y optimizar su rendimiento. El sistema operativo de la máquina virtual, Linux o Microsoft Windows, tiene soporte especializado de paravirtualización como parte del núcleo, así como controladores específicos de paravirtualización que permiten que el sistema operativo y el hipervisor trabajen juntos de forma más eficiente con la sobrecarga de las traducciones del hipervisor. Esta técnica asistida por software ofrece soporte de virtualización optimizado en servidores con o sin procesadores que proporcionan extensiones de virtualización. El soporte de paravirtualización se ha ofrecido como parte de muchas de las distribuciones generales de Linux desde 2008.

Aunque los detalles de este enfoque difieren entre las distintas ofertas, a continuación se presenta una descripción general (véase la Figura 8.5).

Sin paravirtualización, el SO huésped puede ejecutarse sin modificaciones si el hipervisor emula el hardware. En este caso, las llamadas de los controladores del SO huésped al hardware son interceptadas por el hipervisor, que realiza la traducción necesaria para el hardware nativo y redirige la llamada al controlador real. Con la paravirtualización, el código fuente de un sistema operativo se modifica para que se ejecute como sistema operativo invitado en un entorno de máquina virtual específico. Las llamadas al hardware se sustituyen por llamadas al hipervisor, que es capaz de aceptar estas llamadas y redirigirlas sin necesidad de modificar los controladores reales. Este arreglo es más rápido con menos sobrecarga que un configuración no paravirtualizada.



(a) Hipervisor tipo 1

(b) Con SO huéspedes paravirtualizados

Figura 8.5.: Paravirtualización

8.2.5. Virtualización asistida por hardware

Del mismo modo, los fabricantes de procesadores AMD e Intel añadieron funcionalidad a sus procesadores para mejorar el rendimiento con hipervisores. AMD-V y VT-x de Intel designan las extensiones de virtualización asistida por hardware que los hipervisores pueden aprovechar durante el procesamiento. Los procesadores Intel ofrecen un conjunto de instrucciones adicionales denominado **extensiones de máquina virtual** (*Virtual Machine Extensions*, en inglés, VMX). Al tener algunas de estas instrucciones como parte del procesador, los hipervisores ya no necesitan mantener estas funciones como parte de su base de código (*codebase*), el código en sí puede ser más pequeño y más eficiente, y las operaciones que soportan son mucho más rápidas ya que se producen completamente en el procesador. Esta apoyo asistido por hardware no requiere un sistema operativo huésped modificado, a diferencia de la paravirtualización.

8.2.6. Dispositivo virtual

Un dispositivo virtual (o *virtual appliance*, en inglés) es un software independiente que puede distribuirse como una imagen de máquina virtual. Por lo tanto, consiste en un paquete de aplicaciones y un sistema operativo invitado. Es independiente de la arquitectura del hipervisor o del procesador, y puede ejecutarse tanto en un hipervisor tipo 1 como en un tipo 2.

La implementación de un dispositivo de aplicación preinstalado y preconfigurado es mucho más fácil que preparar un sistema, instalar la aplicación y configurarla. Los dispositivos virtuales se están convirtiendo en un medio *de facto* de distribución de software y han generado un nuevo tipo de negocio: el proveedor de dispositivos virtuales.

Además de muchos dispositivos virtuales útiles orientados a la aplicación, un desarrollo relativamente reciente e importante es el dispositivo virtual de seguridad (o *Security*

Virtual Appliance, SVA, en inglés). El SVA es una herramienta de seguridad que realiza la función de monitorear y proteger las otras máquinas virtuales (VM de usuario) y se ejecuta fuera de esas máquinas virtuales en una VM especialmente reforzada (*hardened*) para la seguridad. El SVA obtiene su visibilidad en el estado de una VM (incluidos el estado del procesador, los registros y el estado de la memoria y los dispositivos de entrada y salida), así como el tráfico de red entre VM y entre las VM y el hipervisor, a través de la API de introspección de máquina virtual del hipervisor. El documento NIST SP 800-125 (Recomendaciones de seguridad para la implementación del hipervisor, octubre de 2014) señala las ventajas de esta solución. Específicamente, el SVA es:

- Independiente de la configuración de la red virtual y no es necesario volver a configurarlo cada vez que la configuración de la red virtual cambia debido a la migración de máquinas virtuales o al cambio de conectividad entre las máquinas virtuales residentes en el host del hipervisor.
- No es vulnerable a una falla en el sistema operativo invitado.

8.2.7. Virtualización de contenedores

Un enfoque relativamente reciente para la virtualización se conoce como **virtualización de contenedores** (*container virtualization*, en inglés). En este enfoque, el software, conocido como contenedor de virtualización, se ejecuta sobre el núcleo del sistema operativo *host* y proporciona un entorno de ejecución aislado para las aplicaciones. A diferencia de las máquinas virtuales basadas en hipervisor, los contenedores no tienen como objetivo emular servidores físicos. En cambio, todas las aplicaciones en contenedores en un *host* comparten un núcleo de sistema operativo común. Esto elimina los recursos necesarios para ejecutar un sistema operativo separado para cada aplicación y puede reducir en gran medida la sobrecarga.

8.2.7.1. Grupos de control de kernel

Gran parte de la tecnología para contenedores que se usa hoy en día se desarrolló para Linux y los contenedores basados en Linux son, con mucho, los más utilizados. Antes de pasar a una discusión sobre contenedores, es útil presentar el concepto de **grupo de control** del kernel de Linux.

A partir del trabajo de Menage y Inc (2007), la API de proceso estándar de Linux se amplió para incorporar la contenedorización del entorno del usuario para permitir la agrupación de varios procesos, permisos de seguridad del usuario y gestión de recursos del sistema. Inicialmente conocidos como contenedores de proceso, a fines de 2007, la nomenclatura cambió a grupos de control (*cgroups*) para evitar la confusión causada por múltiples significados del término contenedor en el contexto del kernel de Linux, y la funcionalidad de los grupos de control se fusionó con la línea principal del kernel de Linux en kernel versión 2.6.24, lanzado en enero de 2008.

El espacio de nombres de proceso de Linux es jerárquico, en el que todos los procesos son hijos del proceso del momento de arranque que tienen en común llamado «`init`». Esto forma una jerarquía de proceso único. El grupo de control del kernel permite que coexistan varias jerarquías de procesos en un solo SO. Cada jerarquía se adjunta a los recursos del sistema en el momento de la configuración.

Los *cgroups* proporcionan:

- Limitación de recursos: los grupos se pueden configurar para que no excedan un límite de memoria determinado.
- Priorización: algunos grupos pueden obtener una mayor proporción de utilización de CPU o rendimiento de entrada y salida de disco.
- Contabilidad: mide el uso de recursos de un grupo, que puede usarse, por ejemplo, para fines de facturación.
- Control: congelación de grupos de procesos, su verificación (*checkpoint*) y reinicio.

8.2.7.2. Conceptos de contenedores

La Figura 8.6 compara las pilas de software del contenedor y del hipervisor. Para los contenedores, sólo se requiere un motor de contenedor pequeño como soporte para los contenedores. El motor de contenedor configura cada contenedor como una instancia aislada al solicitar recursos dedicados del sistema operativo para cada contenedor. Cada aplicación contenedor utiliza directamente los recursos del sistema operativo *host*.

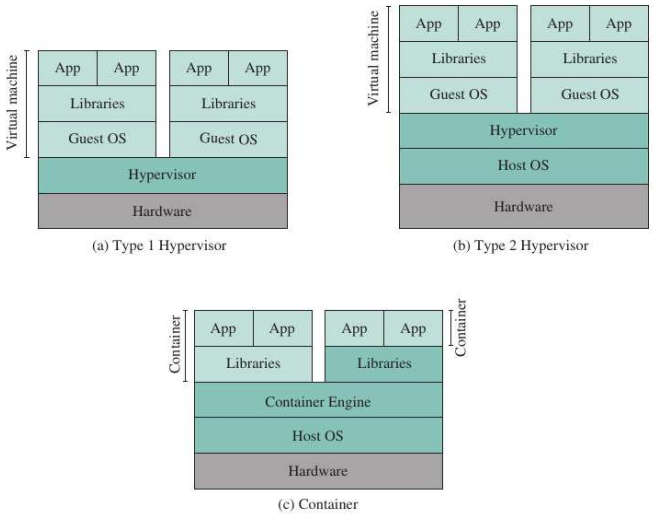


Figura 8.6.: Comparación entre VM y contenedores

Aunque los detalles difieren de un producto contenedor a otro, las siguientes son tareas típicas realizadas por un motor de contenedor:

- Mantener un entorno de tiempo de ejecución ligero y una cadena de herramientas (*toolchain*) que gestione contenedores, imágenes y compilaciones.
- Crear un proceso para el contenedor.
- Administrar puntos de montaje del sistema de archivos.
- Solicitar recursos del núcleo, tales como memoria, dispositivos de entrada y salida y direcciones IP.

Un ciclo de vida típico de los contenedores basados en Linux se puede entender a través de diferentes fases de los contenedores de Linux:

- Preparación: la fase de preparación (*setup*) incluye el entorno para crear e iniciar los contenedores de Linux. Un ejemplo típico de la fase de configuración es el kernel de Linux habilitado con banderas o paquetes instalados para permitir la partición del espacio del usuario. La configuración también incluye la instalación de la cadena de herramientas y las utilidades (por ejemplo, lxc *-linux containers-*, utilidades de puente *-bridge utils-*) para crear una instancia del entorno del contenedor y la configuración de red en el sistema operativo *host*.
- Configuración: los contenedores están configurados para ejecutar aplicaciones o comandos específicos. La configuración del contenedor de Linux incluye parámetros de red (por ejemplo, dirección IP), sistemas de archivos raíz, operaciones de montaje y dispositivos a los que se les permite acceder a través del entorno del contenedor. En general, los contenedores están configurados para permitir la ejecución de una aplicación en recursos controlados del sistema (como el límite superior en el acceso a la memoria de la aplicación).
- Administración: una vez que un contenedor está preparado y configurado, debe administrarse para permitir un arranque (*start up*) sin problemas y el apagado del contenedor. Por lo general, las operaciones administradas para un entorno basado en contenedor incluyen iniciar (*start*), detener (*stop*), congelar (*freeze*) y migrar (*migrate*). Además, hay metacomandos y cadenas de herramientas que permiten la asignación controlada y administrada de contenedores en un solo nodo para el acceso del usuario final.

Debido a que todos los contenedores en una máquina se ejecutan en el mismo kernel, compartiendo así la mayor parte del sistema operativo base, una configuración con contenedores es mucho más pequeña y liviana en comparación con una disposición de máquina virtual de SO hipervisor / invitado. En consecuencia, un sistema operativo puede tener muchos contenedores ejecutándose sobre él, en comparación con el número limitado de hipervisores y sistemas operativos invitados que pueden ser compatibles.

Los contenedores virtuales son factibles debido al control de recursos y a los aislamientos de procesos como se explica usando técnicas como el grupo de control del núcleo. Este enfoque permite compartir recursos del sistema entre varias instancias de contenedores aislados. Los *cgroups* proporcionan un mecanismo para administrar y monitorear los recursos del sistema. El rendimiento de la aplicación es cercano al rendimiento del sistema nativo debido al núcleo único compartido entre todas las instancias de contenedor de

espacio de usuario, y la sobrecarga es sólo para proporcionar un mecanismo para aislar los contenedores a través de *cgroups*. Los subsistemas de Linux particionados que usan primitivas de grupo de control incluyen a: sistemas de archivos, espacio de nombres de proceso, pila de red, nombre de host, IPC y usuarios.

Para comparar máquinas virtuales con contenedores, considere la operación de entrada y salida durante una aplicación con proceso P en un entorno virtualizado. En el entorno clásico de virtualización del sistema (sin soporte de hardware), el proceso P se ejecutaría dentro de una máquina virtual invitada. La operación de entrada y salida se enruta a través de la pila del sistema operativo invitado al dispositivo de E/S invitado emulado. La llamada de E/S es interceptada por un hipervisor que la reenvía a través de la pila del sistema operativo host al dispositivo físico. En comparación, el contenedor se basa principalmente en el mecanismo de indirección proporcionado por las extensiones del marco del contenedor que se han incorporado al núcleo principal. Aquí, un solo núcleo se comparte entre varios contenedores (en comparación con el núcleo del sistema operativo individual en las máquinas virtuales del sistema). La Figura 8.7 ofrece una visión general del flujo de datos de máquinas virtuales y contenedores.

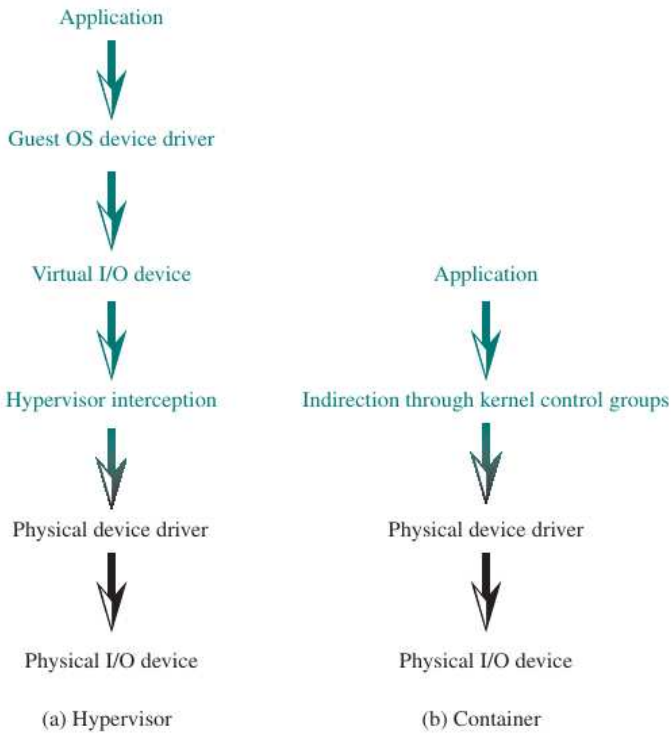


Figura 8.7.: Flujo de datos para el funcionamiento de E/S mediante hipervisor y contenedor

Las dos características notables de los contenedores son las siguientes:

1. No hay necesidad de un SO huésped en el entorno del contenedor. Por lo tanto, los contenedores son livianos y tienen menos gastos generales en comparación con las máquinas virtuales.

2. El software de gestión de contenedores simplifica el procedimiento para la creación y gestión de contenedores.

Debido a que son livianos, los contenedores son una alternativa atractiva a las máquinas virtuales. Una característica atractiva adicional de los contenedores es que proporcionan portabilidad de aplicaciones. Las aplicaciones en contenedores se pueden mover rápidamente de un sistema a otro. Estos beneficios de contenedor no significan que los contenedores sean siempre una alternativa preferida a las máquinas virtuales, como lo demuestran las siguientes consideraciones:

- Las aplicaciones de contenedor solo son portátiles en los sistemas que admiten el mismo núcleo del sistema operativo con las mismas funciones de soporte de virtualización, lo que generalmente significa Linux. Por lo tanto, una aplicación de Windows en contenedor solo se ejecutaría en máquinas con Windows.
- Una máquina virtual puede requerir una configuración de núcleo única que no sea aplicable a otras máquinas virtuales en el host; Este requisito se aborda mediante el uso del SO huésped.
- La virtualización con VM funciona en el borde del hardware y el sistema operativo. Es capaz de proporcionar un sólido aislamiento del rendimiento y garantías de seguridad con la interfaz estrecha entre máquinas virtuales e hipervisores. La contenedorización, que se encuentra entre el sistema operativo y las aplicaciones, incurre en una sobrecarga menor, pero potencialmente introduce mayores vulnerabilidades de seguridad.

Un caso de uso potencial, citado en Kerner (2016), gira en torno a Kubernetes, una tecnología de orquestación de contenedores de código abierto construida por Google pero ahora administrada por la Cloud Native Computing Foundation (CNCF). La fundación en sí opera como un proyecto colaborativo de la Fundación Linux. Como ejemplo, si un administrador dedica 500 Mbps a una aplicación particular que se ejecuta en Kubernetes, entonces el plano de control de red puede participar en la programación de esta aplicación para encontrar el mejor lugar para garantizar ese ancho de banda. O, al trabajar con la API de Kubernetes, un plano de control de red puede comenzar a crear reglas de firewall de entrada que conozcan las aplicaciones de contenedor.

8.2.8. Sistema de archivos de contenedores

Como parte del aislamiento de un contenedor, cada contenedor debe mantener su propio sistema de archivos aislado. Las características específicas varían de un producto contenedor a otro, pero los principios esenciales son los mismos.

Como ejemplo, nos fijamos en el sistema de archivos contenedor utilizado en OpenVZ. Esto se representa en la Figura 8.8. El planificador «`init`» se ejecuta para planificar aplicaciones de usuario y cada contenedor tiene su propio proceso «`init`», que desde la perspectiva de los nodos de hardware es sólo otro proceso en ejecución.

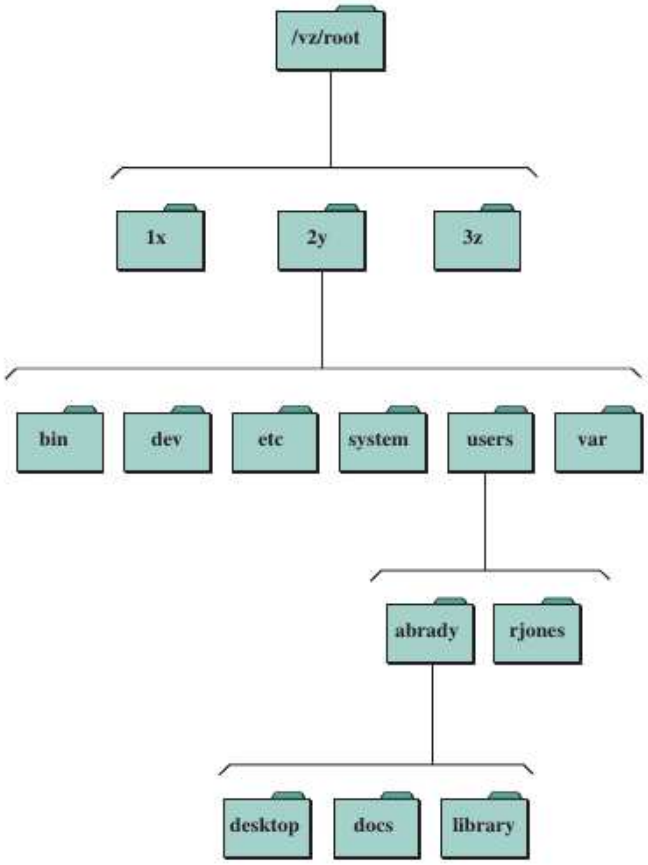


Figura 8.8.: Esquema de archivos en OpenVZ

Es probable que los múltiples contenedores en un host ejecuten los mismos procesos, pero cada uno de ellos no tiene una copia individual a pesar de que el comando «`ls`» muestra que el directorio «`/bin`» del contenedor está lleno de programas. En cambio, los contenedores comparten una plantilla, una característica de diseño en la que todas las aplicaciones que vienen con el sistema operativo, y muchas de las aplicaciones más comunes, se agrupan como grupos de archivos alojados por el sistema operativo de la plataforma y se enlazan simbólicamente a cada contenedor. Esto también incluye archivos de configuración, a menos que el contenedor los modifique; cuando eso sucede, el sistema operativo copia el archivo de plantilla (llamado copia al escribir, ***copy-on-write***), elimina el enlace simbólico virtual y coloca el archivo modificado en el sistema de archivos del contenedor. Al usar este esquema de uso compartido de archivos virtuales, se logra un considerable ahorro de espacio, con sólo archivos creados localmente que realmente existen en el sistema de archivos del contenedor.

A nivel de disco, un contenedor es un archivo y se puede ampliar o reducir fácilmente. Desde el punto de vista de la verificación de virus, el sistema de archivos del contenedor está montado bajo un punto de montaje especial en el nodo de hardware para que las herramientas del sistema a nivel de nodo de hardware puedan verificar de forma segura

cada archivo si es necesario.

8.2.9. Microservicios

Un concepto relacionado con los contenedores es el de microservicio. El documento NIST SP 800-180 (Definición NIST de microservicios, contenedores de aplicaciones y máquinas virtuales del sistema, febrero de 2016) define un microservicio como un elemento básico que resulta de la descomposición arquitectónica de los componentes de una aplicación en patrones acoplados débilemente (*loosely coupled*) que consisten en servicios autocontenidos que se comunican entre sí utilizando un protocolo de comunicaciones estándar 219 y un conjunto de API bien definidas, independientes de cualquier proveedor, producto o tecnología.

La idea básica detrás de los microservicios es, en lugar de tener una pila de aplicaciones monolíticas, cada servicio específico en una cadena de entrega de aplicaciones se divide en partes individuales. Al usar contenedores, las personas están haciendo un esfuerzo consciente para dividir su infraestructura en unidades más comprensibles. Esto abre una oportunidad para que las tecnologías de red tomen decisiones en nombre del usuario que antes no podían tomar en un mundo centrado en la máquina.

Dos ventajas clave de los microservicios son las siguientes:

- Los microservicios implementan unidades desplegables mucho más pequeñas, lo que permite al usuario enviar actualizaciones o realizar funciones y capacidades mucho más rápidamente. Esto coincide con las prácticas de entrega continua, donde el objetivo es expulsar unidades pequeñas sin tener que crear un sistema monolítico.
- Los microservicios también admiten escalabilidad precisa. Debido a que un microservicio es una sección de una aplicación mucho más grande, se puede replicar fácilmente para crear múltiples instancias y distribuir la carga sólo para esa pequeña parte de la aplicación en lugar de tener que hacerlo para toda la aplicación.

8.2.10. Docker

Históricamente, los contenedores surgieron como una forma de ejecutar aplicaciones de una manera más flexible y ágil. Los contenedores de Linux permitieron ejecutar aplicaciones livianas, dentro del sistema operativo Linux directamente. Sin la necesidad del hipervisor y las máquinas virtuales, las aplicaciones pueden ejecutarse de forma aislada en el mismo sistema operativo. Google ha estado utilizando contenedores Linux en sus centros de datos (*data centers*) desde 2006. Pero el enfoque de contenedores se hizo más popular con la llegada de los contenedores Docker en 2013. Docker proporciona una forma más simple y estandarizada de ejecutar contenedores en comparación con versiones anteriores de contenedores. El contenedor Docker también se ejecuta en Linux. Pero Docker no es la única forma de ejecutar contenedores. Linux Containers (LXC) es otra

forma de ejecutar contenedores. Tanto LXC como Docker tienen raíces en Linux. Una de las razones por las que el contenedor Docker es más popular en comparación con los contenedores de la competencia, como LXC, es su capacidad para cargar una imagen de contenedor en un sistema operativo host de una manera simple y rápida. Los contenedores Docker se almacenan en la nube como imágenes y los usuarios los ejecutan cuando los necesitan y de una manera simple.

Docker consta de los siguientes componentes principales:

- Imagen de Docker: las imágenes de Docker son plantillas de solo lectura de las que se crean instancias de contenedores de Docker.
- Cliente Docker: un cliente Docker solicita que se use una imagen para crear un nuevo contenedor. El cliente puede estar en la misma plataforma que un host Docker o una máquina Docker.
- Docker host: una plataforma con su propio sistema operativo host que ejecuta aplicaciones en contenedores.
- Motor Docker: este es el paquete de tiempo de ejecución ligero que construye y ejecuta los contenedores Docker en un sistema host.
- Máquina Docker: la máquina Docker puede ejecutarse en un sistema separado de los hosts Docker, que se utiliza para configurar los motores Docker. La máquina Docker instala el motor Docker en un host y configura el cliente Docker para hablar con el motor Docker. La máquina Docker también se puede usar localmente para configurar una imagen Docker en el mismo host que la máquina Docker.
- Registro de Docker: un registro de Docker almacena imágenes de Docker. Después de compilar una imagen de Docker, puede insertarla en un registro público como Docker Hub o en un registro privado que se ejecute detrás de su firewall. También puede buscar imágenes existentes y extraerlas del registro a un host.
- Docker hub: esta es la plataforma de colaboración, un repositorio público de imágenes de contenedores de Docker. Los usuarios pueden usar imágenes almacenadas en un centro que son aportadas por otros y aportan sus propias imágenes personalizadas.