VIRTUALIZACION

Los hilos y procesos hacen la cosas mas simples, ya que construyen programas que aparentemente son ejecutados simultaneamente. Esto es erroneo ya que en un computador de procesador simple, seria una illusion que esto pasara, se debe a que los procesos son ejecutados a tiempo unico.

3.2 Virtualización

Los subprocesos y procesos pueden verse como una forma de hacer más cosas al mismo tiempo.
hora. En efecto, nos permiten construir (fragmentos de) programas que parecen ejecutarse simultáneamente. En una computadora de un solo procesador, esta ejecución simultánea es, por supuesto, una ilusión. Como solo hay una única CPU, solo se ejecutará una instrucción de un solo hilo o proceso a la vez. Al cambiar rápidamente entre hilos y procesos, se crea la ilusión del paralelismo.
Esta separación entre tener una sola CPU y poder pretender
hay más se puede extender a otros recursos también, lo que lleva a lo que se conoce como virtualización de recursos. Esta virtualización se ha aplicado durante muchas décadas, pero ha recibido un renovado interés a medida que los sistemas informáticos (distribuidos) se han vuelto más habituales y complejos, lo que lleva a la situación de que el software de la aplicación siempre sobrevive al software y al hardware de sus sistemas subyacentes.

Principio de virtualización

En la práctica, cada sistema informático (distribuido) ofrece una programación in-
terface a software de nivel superior, como se muestra en la Figura 3.6 (a). Hay muchos tipos diferentes de interfaces, que van desde el conjunto de instrucciones básicas ofrecido por una CPU a la vasta colección de interfaces de programación de aplicaciones que se envían con muchos sistemas middleware actuales. En esencia, la virtualización trata de ampliar o reemplazar una interfaz existente para imitar el comportamiento de otro sistema, como se muestra en la Figura 3.6 (b). Vamos a hablar sobre detalles técnicos sobre virtualización en breve, pero primero concéntrate en por qué la virtualización es importante.

Virtualización y sistemas distribuidos

Una de las razones más importantes para volver a introducir la virtualización en el
La década de 1970 fue para permitir que el software heredado se ejecutara en hardware costoso de computadora central. El software no solo incluía varias aplicaciones, sino también los sistemas operativos para los que fueron desarrollados. Este enfoque hacia el soporte de software heredado se ha aplicado con éxito en los mainframes IBM 370 (y sus sucesores) que ofrecían una máquina virtual a la que se habían trasladado diferentes sistemas operativos.
Como el hardware se volvió más barato, las computadoras se volvieron más poderosas y el
se redujo la cantidad de diferentes indicadores del sistema operativo, la virtualización

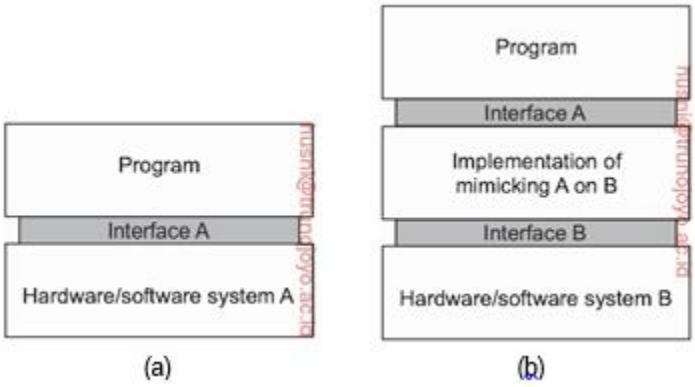

Figura 3.6: (a) Organización general entre un programa, interfaz y sistema. (b) Organización general del sistema de virtualización A en la parte superior de B.

se convirtió en un problema menor. Sin embargo, los asuntos han cambiado nuevamente desde fines de los años noventa. Primero, mientras que el hardware y el software de sistemas de bajo nivel cambian razonablemente rápido, el software en niveles más altos de abstracción (por ejemplo, middleware y aplicaciones) a menudo es mucho más estable. En otras palabras, nos enfrentamos a la situación de que el software heredado no se puede mantener al mismo ritmo que las plataformas en las que se basa. La virtualización puede ayudar aquí portando las interfaces heredadas a las nuevas plataformas y, de este modo, abriendo de inmediato las últimas para grandes clases de programas existentes.

Nota 3.4 (Discusión: ¿software estable?)
Aunque, de hecho, hay una gran cantidad de software heredado que puede beneficiarse de la estabilidad
interfaces para hardware subyacente que cambia rápidamente, es un error creer que el software para servicios ampliamente disponibles apenas cambia. Con el cambio creciente hacia la informática del lado del servidor en forma de software como servicio (SaaS), se puede mantener mucho software para una plataforma relativamente homogénea, propiedad en su totalidad de la organización que ofrece el servicio asociado. Como consecuencia, el mantenimiento de los productos de software puede ser mucho más fácil, ya que existe una necesidad mucho menor de distribuir cambios a potencialmente millones de clientes. De hecho, los cambios pueden suceder rápidamente entre sí después de los cambios en el hardware y la plataforma disponibles, pero sin que ningún cliente note realmente los tiempos de inactividad [Barroso y Hölze, 2009].

Igualmente importante es el hecho de que las redes se han vuelto completamente penetrantes. Es difícil imaginar que una computadora moderna no esté conectada a una red. En la práctica, esta conectividad requiere que los administradores de sistemas mantengan una colección grande y heterogénea de computadoras servidor, cada una con aplicaciones muy diferentes, a las que los clientes pueden acceder. Al mismo tiempo, los diferentes recursos deberían ser fácilmente accesibles para estos

aplicaciones La virtualización puede ayudar mucho: la diversidad de plataformas y máquinas se puede reducir esencialmente al permitir que cada aplicación se ejecute en su propia máquina virtual, posiblemente incluyendo las bibliotecas y el sistema operativo relacionados, que, a su vez, se ejecutan en una plataforma común.
Este último tipo de virtualización proporciona un alto grado de portabilidad y
fl exibilidad Por ejemplo, para realizar redes de distribución de contenido que puedan soportar fácilmente la replicación de contenido dinámico, Awadallah y Rosenblum [2002] han argumentado que la administración se vuelve mucho más fácil si los servidores de borde soportan la virtualización, permitiendo que un sitio completo, incluido su entorno ser copiado dinámicamente. Estos argumentos siguen siendo válidos y, de hecho, la portabilidad es quizás la razón más importante por la que la virtualización desempeña un papel tan importante en muchos sistemas distribuidos.

Tipos de virtualización
Existen muchas formas diferentes en que se puede realizar la virtualización. Un
Smith y Nair [2005a] describen una descripción de estos diversos enfoques. Para comprender las diferencias en la virtualización, es importante darse cuenta de que los sistemas informáticos generalmente ofrecen cuatro tipos diferentes de interfaces, a los tres
niveles diferentes:

1. Una interfaz entre el hardware y el software, conocido como el en-
arquitectura de conjunto de construcción (ISA), formando el conjunto de instrucciones de la máquina. Este conjunto está dividido en dos subconjuntos:
• Instrucciones privilegiadas, que solo pueden ejecutarse por
El sistema operativo.
• Instrucciones generales, que pueden ser ejecutadas por cualquier programa.
2. Una interfaz que consiste en llamadas al sistema como lo ofrece un sistema operativo.
3. Una interfaz que consiste en llamadas a bibliotecas, generalmente formando lo que se conoce como una interfaz de programación de aplicaciones (API). En muchos casos, las llamadas del sistema antes mencionadas están ocultas por una API.

Estos diferentes tipos se muestran en la Figura 3.7. La esencia de la virtualización es imitar el comportamiento de estas interfaces.
La virtualización puede llevarse a cabo de dos maneras diferentes. Primero, podemos construir un sistema de tiempo de ejecución que básicamente proporciona un conjunto de instrucciones abstracto que se usará para ejecutar aplicaciones. Las instrucciones se pueden interpretar (como es el caso del entorno de tiempo de ejecución de Java), pero también se pueden emular como se hace para ejecutar aplicaciones de Windows en plataformas Unix. Tenga en cuenta que en este último caso, el emulador también tendrá que imitar el comportamiento de las llamadas del sistema, que ha demostrado ser, en general, lejos de ser trivial. Este tipo de virtualización, que se muestra en la Figura 3.8 (a), lleva a lo que Smith y Nair [2005a] llaman una máquina virtual de proceso, haciendo hincapié en que la virtualización es solo para un solo proceso.

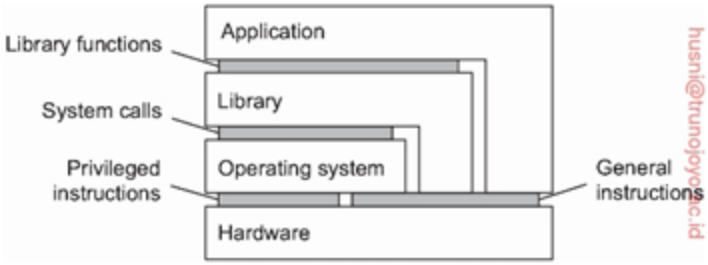

Figure 3.7: Various interfaces offered by computer systems.

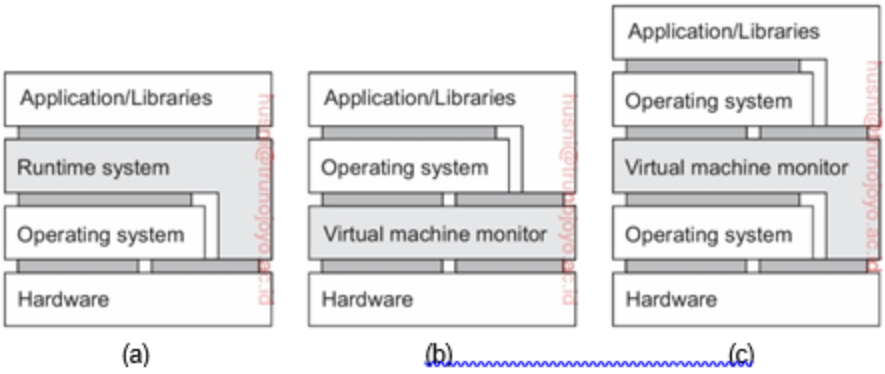

Figura 3.7: Diversas interfaces ofrecidas por los sistemas informáticos.
Figura 3.8: (a) Una máquina virtual de proceso. (b) Un monitor de máquina virtual nativo. (c) Un monitor de máquina virtual alojado.

Un enfoque alternativo hacia la virtualización, que se muestra en la Figura 3.8 (b), es proporcionar un sistema que se implementa como una capa que protege el hardware original, pero que ofrece el conjunto de instrucciones completo de ese mismo (u otro hardware) como una interfaz. Esto lleva a lo que se conoce como monitor de máquina virtual nativo. Se llama nativo porque se implementa directamente sobre el hardware subyacente. Tenga en cuenta que la interfaz ofrecida por un monitor de máquina virtual se puede ofrecer simultáneamente a diferentes programas. Como resultado, ahora es posible tener sistemas operativos invitados múltiples y diferentes que se ejecuten de forma independiente y simultánea en la misma plataforma.
Un monitor de máquina virtual nativo tendrá que proporcionar y regular acceso
a varios recursos, como almacenamiento externo y redes. Al igual que cualquier sistema operativo, esto implica que tendrá que implementar controladores de dispositivo para esos recursos. En lugar de hacer todo este esfuerzo de nuevo, un monitor de máquina virtual alojado se ejecutará encima de un sistema operativo de host de confianza como se muestra en la Figura 3.8 (c). En este caso, el monitor de la máquina virtual puede hacer uso de las instalaciones existentes proporcionadas por ese sistema operativo host. En general, debe ser dado privilegios especiales en lugar de ejecutarse como una aplicación de nivel de usuario. El uso de un monitor de máquina virtual alojado es muy popular en sistemas modernos distribuidos, como centros de datos y nubes.

Como argumentan Rosenblum y Garfinkel [2005], las máquinas virtuales son,
llegando a ser cada vez más importante en el contexto de la confiabilidad y seguridad para sistemas (distribuidos). Al permitir el aislamiento de una aplicación completa y su entorno, una falla causada por un error o un ataque de seguridad ya no debe afectar a una máquina completa. Además, como también mencionamos anteriormente, la portabilidad se mejora en gran medida a medida que las máquinas virtuales proporcionan una nueva disminución entre el hardware y el software, lo que permite trasladar un entorno completo de una máquina a otra. Regresamos a la migración en la Sección 3.5.

Nota 3.5 (Avanzado: sobre el rendimiento de las máquinas virtuales)
Las máquinas virtuales funcionan sorprendentemente bien. De hecho, muchos estudios muestran que
las máquinas virtuales modernas se ejecutan cerca de ejecutar aplicaciones directamente en el sistema operativo host. Echemos un vistazo más de cerca a lo que está pasando por debajo de las máquinas virtuales. Smith y Nair [2005b] proporcionan una cuenta detallada y completa de máquinas virtuales.

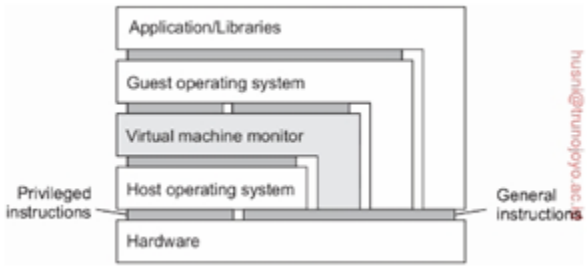

Figura 3.9: Aplicaciones, sistema operativo invitado, monitor de máquina virtual y sistema operativo host en una sola plataforma de hardware.
Parte de la respuesta al problema de rendimiento se muestra en la Figura 3.9, que forma una extensión de la Figura 3.8 (c): una gran parte del código que constituye un monitor de máquina virtual, un sistema operativo invitado y una aplicación se ejecuta de forma nativa en el hardware subyacente. En particular, todas las instrucciones de máquina generales (es decir, sin privilegios) son ejecutadas directamente por la máquina subyacente en lugar de ser interpretadas o emuladas.
Este enfoque no es nuevo y se basa en la investigación de Popek y Goldberg [1974] que formalizó los requisitos para la ejecución eficiente de máquinas virtuales. En pocas palabras, Popek y Goldberg asumieron que la máquina subyacente proporcionaba al menos dos modos de operación (sistema y modo de usuario), que un subconjunto de las instrucciones se podía ejecutar solo en el modo de sistema, y que el direccionamiento de la memoria era relativo (es decir, un la dirección física se obtuvo agregando una dirección relativa a un desplazamiento encontrado en un registro de reubicación). Se hizo una distinción entre dos tipos de instrucciones. Una instrucción privilegiada es una instrucción que se caracteriza por el hecho de que si y solo si se ejecuta en el usuario modo, provoca una trampa en el sistema operativo. Las instrucciones no privilegiadas son todas las demás instrucciones.

Dados estos supuestos formales, Popek y Goldberg definieron dos clases
de instrucciones especiales Una instrucción sensible al control es aquella que puede afectar la configuración de una máquina. Un ejemplo típico es una instrucción que afecta el diseño de la memoria, por ejemplo, al cambiar la compensación de la memoria almacenada en un registro de reubicación. Otro ejemplo son las instrucciones que afectan a la tabla de interrupciones que contiene punteros para interrumpir los manejadores.
Una instrucción sensible al comportamiento es aquella cuyo efecto está parcialmente determinado por el contexto en el que se ejecuta. Por ejemplo, los procesadores Intel x86 tienen instrucciones que pueden o pueden no afectar ciertos registros dependiendo de si esa instrucción se ejecuta en modo de sistema o en modo de usuario. Un ejemplo dado en [Smith y Nair, 2005b] es el de la instrucción POPF, que puede establecer una pestaña habilitada para interrupciones, pero solo cuando se ejecuta en modo sistema.
Ahora tenemos el siguiente resultado importante:

Para cualquier computadora convencional, se puede construir un monitor de máquina virtual si el conjunto de instrucciones sensibles para esa computadora es un subconjunto del conjunto de instrucciones privilegiadas.

Lo que esto dice es que mientras se capturan las instrucciones sensibles cuando se ejecutan en el modo de usuario, podemos ejecutar de forma segura todas las instrucciones no sensibles de forma nativa en el hardware subyacente. Esto también significa que cuando diseñamos conjuntos de instrucciones, si nos ocupamos de que se cumpla con el requisito anterior, no obstaculizaremos innecesariamente la virtualización eficiente de ese conjunto de instrucciones.
Desafortunadamente, no todos los conjuntos de instrucciones tienen instrucciones confidenciales solo de privilegio:
ciones, incluido quizás el más popular, concretamente el conjunto de instrucciones Intel x86. Como resultado, este conjunto tiene 17 instrucciones sensibles que no son privilegiadas [Robin e Irvine, 2000]. En otras palabras, cada una de estas instrucciones se puede ejecutar en modo de usuario sin causar una trampa en el sistema operativo, pero afecta la forma en que el sistema operativo administra sus recursos. En estos casos, hay esencialmente dos soluciones.
La primera solución es emular todas las instrucciones. Por supuesto, esto tendría
un efecto adverso grave en el rendimiento. Para sortear los problemas, un enfoque implementado en VMWare [Sugerman et al., 2001], es escanear el ejecutable e insertar el código alrededor de las instrucciones confidenciales no privilegiadas para desviar el control al monitor de la máquina virtual. Allí, la emulación apropiada se llevará a cabo, por ejemplo, al considerar el contexto en el que se iba a ejecutar la instrucción. El efecto es que puede tener lugar la virtualización completa, lo que significa que la ejecución puede llevarse a cabo sin cambiar el sistema operativo invitado ni la aplicación misma.
Una solución alternativa es aplicar paravirtualization, que requiere la
Sistema operativo invitado para ser modificado. En particular, el sistema operativo invitado se modifica de modo que se aborden todos los efectos secundarios de ejecutar instrucciones confidenciales no privilegiadas en el modo de usuario, que normalmente se ejecutarían en modo de sistema. Por ejemplo, el código se puede reescribir para que estas instrucciones simplemente ya no se produzcan, o si lo hacen, que su semántica es la misma independientemente de si

ejecutándose en modo usuario o sistema. La paralavitualización ha sido adoptada por Xen [Barham et al., 2003; Chisnall, 2007].

Aplicación de máquinas virtuales a sistemas distribuidos
Desde la perspectiva de los sistemas distribuidos, la aplicación más importante de
La virtualización se encuentra en la computación en la nube. Como ya mencionamos en la Sección 1.3, Los proveedores de la nube ofrecen aproximadamente tres tipos diferentes de servicios:

• Infraestructura como servicio (IaaS) que cubre la infraestructura básica
• Platform-as-a-Service (PaaS) que cubre servicios a nivel de sistema • Software-as-a-Service (SaaS) que contiene aplicaciones reales

La virtualización juega un papel clave en IaaS. En lugar de alquilar una máquina física, un proveedor de la nube alquilará una máquina virtual (monitor) que puede o no compartir una máquina física con otros clientes. La belleza de la virtualización es que permite un aislamiento casi total entre los clientes, quienes de hecho tienen la ilusión de que han alquilado una máquina física dedicada. El aislamiento, sin embargo, nunca se completa, aunque solo sea por el hecho de que los recursos físicos reales se comparten, lo que a su vez conduce a un menor rendimiento observable.
Para hacer las cosas más concretas, consideremos la Amazon Elastic Compute Cloud, o simplemente EC2. EC2 le permite a uno crear un entorno que consiste en varios servidores virtuales en red, formando conjuntamente la base de un sistema distribuido. Para simplificar la vida, hay una gran cantidad de imágenes de máquina preconfiguradas disponibles, conocidas como Imágenes de máquina de Amazon, o simplemente AMI. Un AMI es un paquete de software instalable que consiste en un kernel del sistema operativo junto con una serie de servicios. Un ejemplo de una AMI simple y básica es una imagen LAMP, que consiste en un kernel de Linux, el servidor web Apache, un sistema de base de datos MySQL y bibliotecas PHP. También están disponibles imágenes más elaboradas que contienen software adicional, así como imágenes basadas en otros kernels de Unix o Windows. En este sentido, un AMI es esencialmente el mismo que un disco de arranque (aunque hay algunas diferencias importantes a las que regresamos en breve).
Un cliente de EC2 debe seleccionar una AMI, posiblemente después de adaptarla o configurarla. A continuación, se puede iniciar una AMI dando como resultado una instancia de EC2: la máquina virtual real que se puede usar para alojar las aplicaciones de un cliente. Un problema importante es que un cliente casi nunca sabrá exactamente dónde se está ejecutando una instancia. Obviamente, se está ejecutando en una sola máquina física, pero donde se ubica esa máquina permanece oculta. El más cercano puede llegar a identificar la ubicación donde debería ejecutarse una instancia seleccionando una de las pocas regiones provistas por Amazon (EE. UU., América del Sur, Europa, Asia).

Para comunicarse, cada instancia obtiene dos direcciones IP: una privada que puede utilizarse para la comunicación interna entre diferentes instancias, haciendo uso de las instalaciones de red internas de EC2 y una dirección IP pública que permite que cualquier cliente de Internet se ponga en contacto con una instancia. La dirección pública se asigna a la privada usando la tecnología estándar de traducción de direcciones de red (NAT). Una forma simple de administrar una instancia es utilizar una conexión SSH, para la cual Amazon proporciona los medios para generar las claves apropiadas.

El entorno EC2 en el que se ejecuta una instancia proporciona diferentes
niveles de los siguientes servicios:

- CPU: permite seleccionar el número y el tipo de núcleos, incluidas las GPU
- Memoria: define cuánta memoria principal se asigna a una instancia
- Almacenamiento: define la cantidad de almacenamiento local asignado.
- Plataforma: distingue entre arquitecturas de 32 o 64 bits.
- Red: establece la capacidad de ancho de banda que se puede usar

Además, se pueden solicitar recursos adicionales, como una interfaz de red adicional. El almacenamiento local que viene con una instancia es transitorio: cuando la instancia se detiene, se pierden todos los datos almacenados localmente. Para evitar la pérdida de datos, un cliente deberá guardar explícitamente datos en una tienda persistente, por ejemplo, haciendo uso del Servicio de almacenamiento simple de Amazon (S3). Una alternativa es adjuntar un dispositivo de almacenamiento que se asigna al Almacén Elastic Block de Amazon (Amazon EBS). De nuevo, este es otro servicio más, pero que se puede usar en la forma de un dispositivo de bloque virtual que simplemente se monta como uno podría montar un disco duro adicional. Cuando se detiene una instancia, todos los datos que se almacenaron en EBS persistirán. Y tal como cabría esperar, un dispositivo EBS también puede montarse (re) en cualquier otra instancia.
Debería estar claro ahora que, sin haber entrado en ningún momento significativo
nivel de detalle, el IaaS ofrecido por EC2 permite al cliente crear una cantidad (potencialmente grande) de máquinas virtuales, cada una configurada con recursos según sea necesario, y capaz de intercambiar mensajes a través de una red IP. Además, se puede acceder a estas máquinas virtuales desde cualquier lugar a través de Internet (siempre que el cliente tenga las credenciales adecuadas). Como tal, Amazon EC2, al igual que muchos otros proveedores de IaaS, ofrece los medios para configurar un sistema distribuido completo que consiste en servidores virtuales en red y aplicaciones distribuidas distribuidas por el cliente. Al mismo tiempo, esos clientes no necesitarán mantener ninguna máquina física, lo que a menudo es ya una gran ganancia, como encontraremos en varias ocasiones a lo largo de este texto. De hecho, uno puede argumentar que la virtualización se encuentra en el núcleo de la computación en la nube moderna.