Universidad Nacional de La Matanza
Florencio Varela 1903 - San Justo - Buenos Aires - Argentina

Departamento de Ingeniería e Investigaciones Tecnológicas

Sistemas Operativos

Año 2015

Virtualización

Historia

La Virtualización no es un concepto nuevo, es una tecnología que comenzó a desarrollarse en la década del 60, en especial por IBM con la primera computadora diseñada específicamente para virtualización, que fue el mainframe IBM S/360 Modelo 67. Esta característica de virtualización ha sido un Standard de la línea que siguió (IBM S/370) y sus sucesoras, incluyendo la serie actual.
La idea de estos desarrollos era particionar los mainframes de gran tamaño a fin de mejorar su utilización, de esta forma se conceptualiza el término "Multitasking" (Ejecutar múltiples procesos concurrentemente)
Luego de unos años, en los 80´s y 90´s con la aparición de las computadoras personales (PC's) este particionamiento de la capacidad de cómputo prácticamente desaparece debido a la adopción de la arquitectura Cliente / Servidor donde parte de las cargas de trabajo se trasladaban a las computadoras de los usuarios.
Se tiende a la utilización de servidores Windows y Linux y se impone la idea de una aplicación por servidor físico; esta nueva era genera lo que se conoce como server sprawl (proliferación de servidores) acarreando las siguientes consecuencias:

Ventajas:

- ➢ Si falla un equipo físico sólo falla una aplicación o carga de trabajo.
- ➢ Se eliminan los conflictos entre aplicaciones dado que no residen en el mismo servidor.
- ➢ Se aplican actualizaciones y parches aisladamente.

Desventajas:

- ➢ Empieza un crecimiento descontrolado de servidores en el centro de cómputos.
- ➢ Se incrementa la complejidad de administración, consumo de energía y refrigeración.
- ➢ Sub utilización de la infraestructura (10% al 15% de capacidad utilizada)
- ➢ Se incrementa el costo del mantenimiento
- ➢ Cada vez que se tiene que desarrollar, probar o poner en producción una nueva aplicación es necesario la incorporación de un nuevo servidor físico (tal vez tan simple como simplemente instalarlo porque ya se tiene, o tan complejo como desde una empresa tener que verificar si se tiene en el presupuesto del ejercicio actual asignada una partida para la compra de nuevo hardware, generar la orden de pedido o pliego para su compra, esperar a que los diferentes proveedores realicen sus ofertas, adjudicar la compra a un proveedor, esperar a que éste entregue el hardware y recién ahí poder empezar a instalar la aplicación)

En la actualidad, se plantean para los servidores X86, los mismos problemas que se planteaban para los mainframes en la década del 60. A comienzos de los 2000 se comenzó a implementar la virtualización en los sistemas x86 como un medio para solucionar muchos de estos problemas y transformar los sistemas x86 en infraestructuras de hardware compartido de uso general que ofrecen un aislamiento completo, movilidad y opciones de elección del sistema operativo en los entornos de aplicaciones.

La frase de este párrafo que versa "…A comienzos de los 2000 se comenzó a implementar la virtualización en los sistemas x86" es mucho más profunda de lo que parece. En 1974, Gerald Popek y Robert Goldberg, publicaron un famoso artículo en el mundo de la virtualización

titulado “Formal Requirements for Virtualizable Third Generation Architectures”, cuyo nombre es español sería algo así como “Requerimientos Formales de Virtualización para las Arquitecturas de Tercera Generación”. Es un documento más que interesante de leer aunque el mismo sea de 1974, ya que ahí se propusieron por primera vez los requerimientos que se tenían que cumplir en una arquitectura de computadoras para que la misma fuese virtualizable. Debido a este artículo se decía que los microprocesadores de las familias x86 no eran virtualizables.
Se tuvieron que esperar casi 3 décadas para que se desarrollaran técnicas para poder llevar esto adelante. Por un lado VMware desarrolló una técnica de compilación dinámica para detectar lo que el artículo llaman “*Instrucciones Sensibles”* y por el otro Xen utilizó la paravirtualización que se basa en la modificación del sistema operativo guest.

La virtualización es definitivamente el primer paso para la llamada “Cloud Computing” o “Computación en la Nube”. Las compañías de todo el mundo tienen que pasar por un viaje en el que el primer paso es comenzar a virtualizar la infraestructura que ellos tienen, luego desplegar las aplicaciones de negocios críticas, y luego proporcionar automatización y autoservicios para convertir el centro de datos en una “nube privada”

Ventajas y desventajas de la virtualización

La virtualización de servidores X86 presenta las siguientes ventajas y desventajas.

Ventajas:

- Aislamiento: las máquinas virtuales son totalmente independientes, entre sí y con el hypervisor. Por lo tanto un fallo en una aplicación o en una máquina virtual afectará únicamente a esa máquina virtual. El resto de máquinas virtuales y el hypervisor seguirán funcionando normalmente.
- Seguridad: cada máquina tiene un acceso privilegiado (root o administrador) independiente. Por tanto, un ataque de seguridad en una máquina virtual sólo afectará a esa máquina.
- Protección: Facilita las soluciones de Recovery o Recuperación.
- Flexibilidad: podemos crear las máquinas virtuales con las características de CPU, memoria, disco y red que necesitemos, sin necesidad de “comprar” un ordenador con esas características. También podemos tener máquinas virtuales con distintos sistemas operativos, ejecutándose dentro de una misma máquina física.
- Balanceo dinámico: Los hipervisores permiten el balanceo dinámico de los recursos del hardware físico entre las distintas máquinas virtuales.
- Consumo óptimo: Se produce un consumo de recursos mucho más eficiente .
- Agilidad: la creación de una máquina virtual es un proceso muy rápido, básicamente la ejecución de un comando. Por tanto, si necesitamos un nuevo servidor lo podremos tener casi al instante, sin pasar por el proceso de compra, configuración, etc.
- Portabilidad: toda la configuración de una máquina virtual reside en uno o varios archivos. Esto hace que sea muy fácil clonar o transportar la máquina virtual a otro servidor físico, simplemente copiando y moviendo dichos ficheros que encapsulan la

máquina virtual. Algunos de los hipervisores que existen actualmente en el mercado permiten realizar la migración de una máquina virtual de un server físico a otro sin necesidad de apagarla.

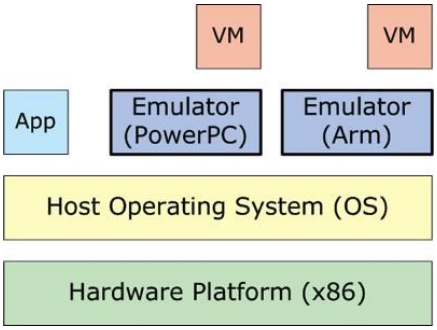
- ➢ Ahorro de costos: Podremos adquirir un solo servidor, aunque más potente, y no tener que comprar más servidores sino solamente ir creándolos en el gestor de máquinas virtuales. También permite ahorro en el costo de mantenimiento y en el de personal, además de ahorrar espacio y energía.

Desventajas:

- ➢ La avería del servidor anfitrión o host de virtualización afecta a todas las máquinas virtuales alojadas en él.
- ➢ Rendimiento inferior: Un sistema operativo virtualizado nunca alcanzará las mismas cotas de rendimiento que si estuviera directamente instalado en el servidor físico. Una aplicación generalmente correrá más despacio en una máquina virtual que en un servidor físico. La degradación dependerá de la tecnología de virtualización utilizada, de la configuración realizada a nivel hipervisor y de la propia aplicación. Por regla general, las aplicaciones que más repercuten la pérdida de rendimiento son las que realizan operaciones frecuentes de entrada/salida. No obstante, con el desarrollo de técnicas cada vez más trabajadas y con el advenimiento de la virtualización asistida por hardware en los microprocesadores esta pérdida de performance cada vez es menor haciendo que hoy en día aplicación de muy alta criticidad estén virtualizadas en la mayoría de las empresas.
- ➢ No es posible utilizar hardware que no esté soportado por el hipervisor.
- ➢ No se disponía de aceleración de vídeo por hardware, aunque ya en las últimas versiones de los hipervisores esto ya está soportado, permitiendo así que incluso esas aplicaciones que antes sí o sí debían residir en sistemas físicos debido a su alto uso de este tipo de hardware ahora ya se están de a poco comenzando a virtualizar. Los hipervisores modernos permiten presentarles a las máquinas virtuales una aceleración por software de video como así también planificar esas tareas en las tarjetas de video (GPU) que se instalan físicamente en el hardware del hipervisor.
- ➢ Proliferación de máquinas virtuales; al ser tan sencilla la creación de una máquina virtual y no implicar una compra de hardware, puede ocurrir que se creen máquinas virtuales indiscriminadamente. Esto aunque suene raro es extremadamente común en los ambientes virtualizados. Analizaremos más adelante entre las funcionalidades avanzadas de máquinas virtuales la clonación y templates.

Tipos de Virtualización

Emulación



La emulación o software emulador se diseña con el objeto de imitar una determinada arquitectura de hardware, de tal forma que los procesos y/o sistemas operativos diseñados para esta arquitectura puedan ejecutar como si efectivamente estuvieran sobre dicho hardware (arquitectura emulada).
La primera pregunta que surge es: ¿para qué podríamos necesitar hacer uso de un emulador?
La respuesta es simple: Podríamos ejecutar software preparado para determinada arquitectura aún en un equipo con una arquitectura completamente diferente, máquinas antiguas, sistemas embebidos o incluso para realizar pruebas de versiones de sistemas operativos, típicamente cambios en los kernels.
El uso más frecuente en esta época se da en la construcción de aplicaciones para teléfonos celulares, donde puede ser muy útil emular una arquitectura de determinado teléfono (por ejemplo, ARM), para luego instalar un SO compatible y poder desarrollar o probar aplicaciones.

Ventajas y Desventajas

Como se mencionó antes, podemos emular cualquier hardware, con los beneficios que eso supone.
Sin embargo, la emulación suele tener un rendimiento bajo, es decir, los tiempos de respuesta no son los mejores.
La siguiente pregunta obligada: ¿por qué?
La emulación implica la traducción de código de la máquina original a la máquina de destino, es decir, este software intermedio (emulador) se ocupa de interpretar las instrucciones recibidas desde la máquina emulada y luego generar las instrucciones necesarias para su ejecución en el hardware subyacente. Esa traducción conlleva un costo, y por eso se evidencia una baja performance en la mayoría de los casos.

Principales exponentes

Bochs (http://bochs.sourceforge.net/)

Se trata de un emulador altamente portable de IA-32 (x86) libre.
Codificado en C++, logra la emulación de la CPU Intel x86, dispositivos de entrada y salida comunes, y dispone de una BIOS configurable. El uso típico de Bochs es para proporcionar emulación completa de un PC x86, y puede ser compilado para emular 386, 486, Pentium/PentiumII/PentiumIII/Pentium4 o CPU x86-64 incluyendo opcionalmente instrucciones MMX, SSEx y 3DNow. En cuanto a sistemas operativos que soporta, Bochs permite la ejecución de Linux, DOS, Windows 95/98 y Windows NT/2000/XP o Vista.

Qemu (http://www.qemu.org/)

Qemu es sin duda una de las soluciones de virtualización más interesantes en la actualidad. Y es que aunque haya sido incluido en el presente apartado como un emulador, permite también virtualización completa de equipos. No en el sentido de virtualizar una infraestructura completa de servidores, ya que no proporciona funcionalidades complejas como puede tratarse de la migración de máquinas virtuales, y sí en el de satisfacer las condiciones necesarias para la ejecución de máquinas virtuales diversas en arquitectura hardware, sistemas operativos y aplicaciones, por lo que en la mayoría de los casos en los que queremos virtualizar de una manera más modesta Qemu será la solución más adecuada, ya que no será necesario llevar a cabo complejas configuraciones –como por ejemplo recompilar el kernel de Linux.

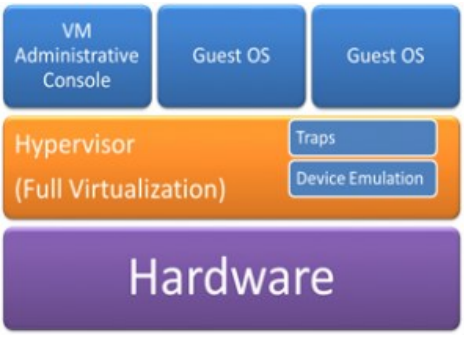
MAME (http://www.mamedev.org/)

Este es un ejemplo de uno de los emuladores libres más conocidos por quienes gustan de los videojuegos.

MAME (Multiple Arcade Machine Emulator), permite emular las famosas máquinas recreativas de los años ochenta y noventa, y que nos permite volver a disfrutar de aquellos juegos ahora en sistemas operativos como Windows, Linux o MacOS sin la necesidad de disponer de la máquina original. El número de sistemas emulados por MAME es realmente impresionante, lo que nos permite acceder a un sinfín de juegos emulados.

En este mismo sentido, se han desarrollado varios proyectos de emulación, no sólo de videojuegos, sino también de computadoras antiguas y hasta calculadoras. Uno de los más conocidos y emparentados con MAME es MESS (Multi Emulator Super System).

Hipervisores tipo 1 o Bare-Metal (Virtualización completa)



La virtualización completa, también llamada nativa, es un modelo que incluye un hipervisor con código específico para emular el hardware subyacente. Pero, ¿qué es un hipervisor?

El *hipervisor* o *hypervisor* es un pequeño monitor de bajo nivel de máquinas virtuales que se inicia durante el arranque, antes que las máquinas virtuales, y que normalmente corre justo sobre el hardware (denominado en alguna bibliografía como native o bare-metal), aunque también lo puede hacer sobre un sistema operativo (llamado en este caso hipervisor hosted, que se analizará más adelante).

Su objetivo fundamental es la gestión y administración de las instancias de las máquinas virtuales, por lo que lo habitual es que incluyan diversas herramientas de gestión y monitorización.

Con este modelo, el hipervisor sólo podrá gestionar máquinas virtuales con sistema operativo, bibliotecas y utilidades compiladas para el mismo hardware y juego de instrucciones que el de la máquina física.

Al introducir código en el hipervisor (para la emulación del hardware), por lo general en la mayoría de los casos la virtualización completa obtendrá un rendimiento relativamente menor que la paravirtualización (que analizaremos más adelante). Para optimizar el rendimiento es posible la utilización de soporte de hardware específico para virtualización. Actualmente, el uso de este soporte es prácticamente de carácter obligatorio.

El reto principal de la virtualización completa es la interceptación y simulación de operaciones que son privilegiadas, como por ejemplo las instrucciones de entrada/salida. Por otro lado, las instrucciones generadas por máquinas virtuales que no afectan ni acceden a otras máquinas virtuales ni a la máquina anfitriona suelen ser ejecutadas directamente por el hardware, sin simulación alguna.

En el gráfico que se puede observar más arriba se puede apreciar la arquitectura general del modelo de virtualización completa. Nótese que se menciona una máquina virtual "Administrativa".
Esta máquina tiene como objetivo la gestión y administración de todo el entorno virtual. Dependiendo del producto, puede implementarse como una máquina virtual específica (como se ve en el gráfico) o bien como una consola dentro del mismo hipervisor.

Ventajas y Desventajas

Una de las principales ventajas de la virtualización completa es la posibilidad de disponer de máquinas virtuales con sistemas operativos sin modificar, es decir, no hace falta alterar los sistemas operativos invitados (guests).
Por otra parte, al tener un hipervisor con código de emulación de hardware, generalmente se dispone de un rendimiento menor, comparado con la técnica de paravirtualización.
Sin embargo, a partir del uso de soporte para hardware (Intel VT / AMD-V), muchas instrucciones privilegiadas pueden ejecutarse directamente en el procesador, y esto está en continua mejora por parte de los fabricantes de microprocesadores, por lo que la penalización en los procesos de traducción se reduce cada vez más.
Otra desventaja de los hipervisores de este tipo es que al tratar directamente con el hardware subyacente deben implementarse los drivers o manejadores de dispositivos para cada hardware subyacente donde se quiera ejecutar el hipervisor. Por ejemplo, si se quiere ejecutar el hipervisor en una máquina que posee una placa de red marca Broadcom se deberá desarrollar un driver específico para esa placa. Debido a que los hipervisores han cobrado una importancia relevante hoy en día en los principales centros de cómputos de la mayoría de las compañías, son los fabricantes de hardware los que se encargan del desarrollo de dichos drivers para los hipervisores más importantes, y son ellos los que también llevan adelante las pruebas de homologación de dicho hardware, ya que si no lo hiciesen quedarían casi automáticamente excluidos del mercado comercial masivo.

Principales exponentes

VMWare (http://www.vmware.com)

VMware es sin duda alguna uno de los gigantes de la virtualización. Fue pionero a principios de los años noventa a la hora de ofrecer excelentes y completas soluciones para sistemas operativos tanto Microsoft Windows como Linux, en equipos cuyos recursos hardware en

teoría hasta entonces no eran suficientes para la implementación de este tipo de actividades. Ver artículo "Formal Requirements for Virtualizable Third Generation Architectures" de Popek y Golberg del año 1974.
Si bien hemos incluido a VMWare dentro del apartado de Virtualización Completa, es importante aclarar que la empresa ofrece soluciones de virtualización prácticamente a todos los niveles y para todas las necesidades.
El producto por excelencia, en cuanto a virtualización completa es el denominado VMware ESXi.

Citrix XenServer (http://www.citrix.com/products/xenserver/overview.html)

La empresa Citrix, históricamente dedicada a soluciones de acceso remoto, se adentraba en el mundo de la virtualización allá por el año 2007, adquiriendo a la empresa XenSource, originalmente formada por ex miembros del proyecto original del hipervisor Xen de la Universidad de Cambridge, dedicada por completo al desarrollo y soporte en el uso de Xen.
Tomando el hipervisor Xen como punto de partida, los equipos de XenSource y posteriormente de Citrix desarrollaron sobre él una completa plataforma de virtualización, proporcionando consolidación de servidores efectiva. Este producto se denomina Citrix XenServer.
En la actualidad, Citrix XenServer requiere de una máquina host de 64 bits y la administración de la plataforma virtual debe realizarse con un cliente que corre sobre Microsoft Windows (si bien ya están apareciendo productos de terceros para realizar estas gestiones que funcionan bajo otros sistemas operativos).
Por otra parte, la versión open source de este hipervisor: "Xen" a secas (http://www.xenproject.org), puede ser ejecutado en una máquina con hardware de 32 bits sin la necesidad de disponer de un cliente Windows para administrarlo. El soporte de hardware para virtualización en los procesadores es necesario cuando queremos utilizar sistemas operativos no modificables (por ejemplo alguno de la familia Microsoft Windows).

z/VM (http://www.vm.ibm.com)

Esta solución es la ofrecida por IBM en el sector de la virtualización. Puesto en el mercado a finales del año 2000, supuso un paso muy importante en una empresa que desde la década de los sesenta ha creado los mainframes más eficientes y ha trabajado con técnicas relacionadas con virtualización. Así, representó y sigue representando la evolución de las soluciones basadas en los conceptos y tecnologías desarrollados bajo la experiencia de IBM, con origen en el CP/CMS en los System/360-67 de IBM.
z/VM es utilizado en los zSeries de IBM, los computadores System z9 y System z10. Soporta como sistemas operativos en las máquinas virtuales Linux, z/OS, z/OS.e, TPF (Transaction Processing Facility) y z/VSE. También es posible instalar el propio z/VM, lo que nos permite anidar máquinas virtuales.
El uso más habitual hoy día es la ejecución de múltiples sistemas Linux sobre mainframes, aprovechando la flexibilidad y potencia que genera esta combinación.

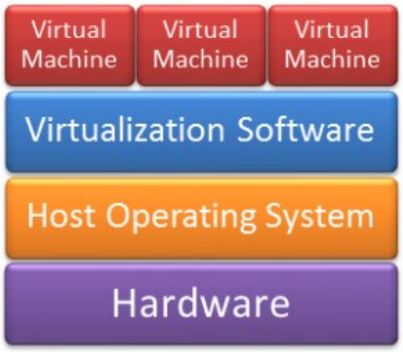
Microsoft Hyper-V (https://technet.microsoft.com/en-us/virtualization/cc150660.aspx)

Hyper-V es la solución de virtualización de Microsoft. Se ofrece en dos modalidades:

a) Como parte del sistema operativo Windows (Windows Server 2008 en adelante).
b) Como servidor stand-alone (hipervisor de tipo 1).

Este producto ha evolucionado notablemente en los últimos tiempos, al punto de competir prácticamente de igual a igual con algunos de los antes mencionados. Posee una arquitectura basada en un micro kernel, con distintas capas de abstracción y particiones donde corre cada máquina virtual, que ofician de separadores lógicos entre ellas.
En cuanto al soporte de sistemas operativos guests, posee algunas restricciones, al igual que ciertas deudas en cuanto al manejo de cierto hardware como audio, gráficos y dispositivos usb, aunque como se mencionó antes está en continua evolución.

Hipervisores tipo 2 ("host-based")



Un hipervisor tipo 2 es, en realidad, una aplicación de virtualización, que corre sobre un sistema operativo de la misma forma a como lo hace cualquier otra aplicación. Gestiona y administra las máquinas virtuales, controla el acceso a los recursos disponibles en el equipo físico e intercepta y trata cualquier instrucción privilegiada emitida por las máquinas virtuales.

En este modelo de virtualización, la gestión final con los recursos de hardware queda en manos del sistema operativo base, mediante las llamadas generadas por la aplicación de virtualización (hipervisor tipo 2), que en definitiva es una aplicación más que compite por los recursos junto con otros procesos.

Ventajas y Desventajas

Una de las ventajas evidentes de este modelo es la posibilidad de instalar sistemas operativos guests en una computadora personal de un usuario, pudiendo "encender" y "apagar" esas máquinas virtuales en cualquier momento. Esto significa una gran ventaja para el uso esporádico de máquinas virtuales, o con fines muy puntuales y a la vez implica ciertos riesgos. Por ejemplo: Si tengo un equipo con muchas aplicaciones, y entre ellas un software de virtualización con 3 máquinas virtuales en ejecución, podría ocurrir que esas aplicaciones hagan uso intensivo de ciertos recursos y por ende se vería afectada la performance de los tres equipos virtuales, o viceversa.
Debido a que el software de virtualización (hipervisor de tipo 2) es un proceso usuario más, estará sometido a las políticas de planificación del sistema operativo principal, y competirá por los recursos como cualquier otro proceso.
Por otra parte, este modelo tiene mayor exposición a caídas de todo el sistema. Por ejemplo: Si un bug en el driver de video provoca la caída de todo el sistema durante la ejecución de una herramienta de procesamiento de imágenes, tal situación dejaría fuera de servicio también a las máquinas virtuales.
Ahora bien, ¿eso significa que un hipervisor de tipo 1 no se puede "colgar"? No, no confundir. Un hipervisor de tipo 1 sigue siendo software y también está expuesto a fallas o bugs, sin embargo al ser un producto exclusivamente dedicado a su función de virtualizador, se reducen las posibilidades de que esto ocurra. Esto básicamente ocurre por el hecho que los kernel de los

hipervisores tipo 1 son mucho más reducidos que los kernel de sistemas operativos genéricos donde corren los hipervisores tipo 2. Al ser más chicos se requieren de menos líneas de código, y como se sabe, los errores o bugs en un sistema son proporcionales a la cantidad de las mismas.
En cuanto a la performance, hay que considerar que se tiene una capa de traducción adicional respecto al modelo de virtualización completa, por lo que se notará un rendimiento menor.

Por los motivos expuestos, generalmente se utilizan hipervisores tipo 2 en computadoras personales o bien en instalaciones muy reducidas, donde no se justifica contar con un hardware dedicado que sirva de host "anfitrión" de máquinas virtuales. Sin embargo, en centros de datos sí es común encontrar equipos con hipervisores de tipo 1, por cuestiones de performance, seguridad y estabilidad.

Principales exponentes

VMware Workstation (http://www.vmware.com)

Como mencionamos antes, **VMWare** es uno de los referentes en el mundo de la virtualización. De hecho, esta empresa ofrece varios productos, con distintos fines y para casi todos los gustos y presupuestos.
En lo que respecta al modelo de hipervisor tipo 2, los productos destacados son **VMware Workstation** y **VMware Player** y **VMware Fusion** (para sistemas operativos base Mac OS).
Permiten a la mayoría de los usuarios adentrarse en el mundo de la virtualización, ofreciendo una gran flexibilidad en la creación y utilización de las máquinas virtuales, con una interfaz muy intuitiva y potente, por lo que son muy utilizados en entornos educativos con fines de prueba y depuración de sistemas operativos y aplicaciones.
El producto denominado **VMware Server**, también entra en esta categoría de virtualizadores, éste último posee características adicionales al Workstation. No obstante, si bien aún se puede utilizar, es un producto discontinuado.

Oracle VM Virtualbox (https://www.virtualbox.org/)

Esta herramienta de virtualización fue originalmente desarrollada por la empresa alemana Innotek Gmbh. Posteriormente fue adquirida por Sun Microsystems y posteriormente por Oracle Corporation. Por este motivo, la herramienta fue renombrada varias veces: **Innotek VirtualBox, Sun xVM VirtualBox, Sun VirtualBox y finalmente, y por ahora, Oracle VM VirtualBox, o simplemente VirtualBox.**
Este producto no debe confundirse con **Oracle VM Server**, ya que ésta es una herramienta de Paravirtualización; a diferencia de **VirtualBox**, que funciona como hipervisor tipo 2, es decir, requiere un sistema operativo base o anfitrión.

Parallels Desktop (http://www.parallels.com)

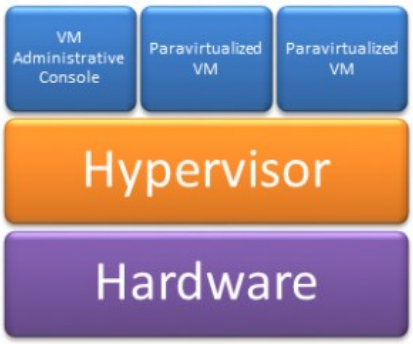
Se trata del producto de virtualización tal vez más conocido en el mundo de Macintosh. Funciona sobre el sistema operativo Mac OS X y permite la ejecución de máquinas virtuales con sistemas operativos x86 (32 y 64 bits), lo que lo hace muy atractivo para los usuarios que utilicen ambas plataformas.
La solución que compite directamente con **Parallels** y posee prestaciones muy similares a **VMWare Fusion**.

Windows Virtual PC (http://www.microsoft.com/es-ar/download/details.aspx?id=3702)

Este producto, antes denominado **Microsoft Virtual PC**, es un software de gestión de virtualización desarrollado por la empresa Connectix y comprado por Microsoft.
Actualmente, este software sólo soporta **Microsoft Windows** (anfitrión e invitados), ya que si bien hay iniciativas para dar soporte a sistemas operativos GNU/Linux, éstos aún no están oficialmente soportados.
Por este motivo, el uso más común de este producto es la utilización de máquinas virtuales de Microsoft Windows anteriores al sistema operativo anfitrión. Por ejemplo, un equipo con Windows 7 corriendo una máquina virtual con Windows XP.

Paravirtualización

Este modelo también hace uso de un hipervisor (puede ser de tipo 1, que es lo más frecuente, o de tipo 2) como capa de virtualización, para administrar el hardware subyacente.
Un sistema de paravirtualización no incluye emulación de hardware, pero sí requiere modificaciones en los sistemas operativos invitados (guests), haciendo que éstos estén al tanto del proceso de virtualización, comunicándose directamente con el hipervisor.



Ventajas y Desventajas

Los sistemas operativos guests deben ser modificables lógicamente. Eso excluye a todos los sistemas operativos privativos, como Microsoft Windows(*).
Nos detenemos un instante acá, para preguntarnos: ¿eso es una ventaja o una desventaja?
Bien, como ocurre muchas veces en cuestiones de tecnología, la respuesta a esa pregunta puede variar según el punto de vista que se analice. A priori, pensar en que un sistema de virtualización altere a nuestro sistema operativo guest suena a peligroso, y poco recomendable para un entorno productivo. ¿cómo podría garantizarse que el funcionamiento del kernel no se verá afectado?
Pero por otro lado, podemos hacer la siguiente lectura: Si mi objetivo es virtualizar servidores utilizando el mismo sistema operativo en cada máquina virtual y ese sistema operativo es absolutamente compatible con el virtualizador, ¿por qué no hacerlo?
El tiempo fue demostrando que este modelo no sólo funciona, sino que también fue adoptado, aunque parcialmente, por fabricantes de soluciones de Virtualización Completa. Más adelante mencionaremos algunos ejemplos.
(*) Aclaración importante: Si utilizamos un entorno de paravirtualización usando un hardware con soporte específico para virtualización, entonces sí podremos ejecutar sistemas operativos no modificables, ya que todas las operaciones privilegiadas/protegidas serán manejadas directamente por el hardware. Por este motivo, hoy podemos encontrar soluciones de paravirtualización que permiten ejecutar diversos sistemas operativos.

Otra ventaja es la mejora notable del rendimiento final, prácticamente similar a la que obtendríamos en un sistema no virtualizado. Esta mejora en el rendimiento se logra ya que se elimina la necesidad de capturar las instrucciones de las máquinas guests en el hipervisor, dado que los sistemas operativos cooperan en el proceso.

Principales exponentes

Xen (http://www.xenproject.org)

Se trata de una solución completamente open source surgida en la Universidad de Cambridge en el año 2003, pionera en la implementación del modelo de paravirtualización.
Como se mencionó antes, además de esta versión libre, existen otras versiones comerciales, con características adicionales, incluso hasta ofreciendo modelos de virtualización completa e híbridos.
Muchas empresas líderes en informática han apostado por Xen como solución de virtualización, como IBM, Oracle, SuSe, etc.

Con Xen, si no disponemos de procesadores con tecnología de virtualización sólo podremos instanciar máquinas virtuales con sistemas operativos modificables (Linux, Minix, Plan 9, NetBSD, FreeBSD, OpenSolaris), algo que desde el punto de vista Linux es un compromiso totalmente razonable debido a las mejoras en el rendimiento que se obtienen respecto a otros modelos. En cambio, si se requiere un soporte más amplio, ello puede presentarse como un inconveniente. Si disponemos de la nueva generación de chipsets AMD-V o Intel VT, podremos entonces ejecutar lo que Xen denomina Hardware Virtual Machine, o virtualización acelerada, que nos permitirá la creación y ejecución de dominios con instancias de sistemas operativos no modificados.

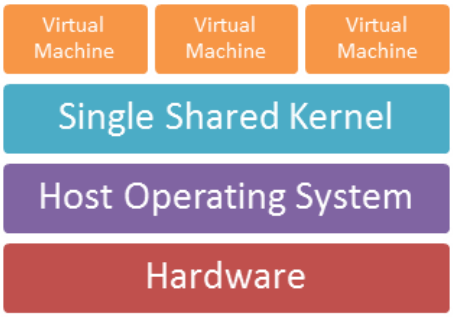
OracleVM (www.oracle.com/us/technologies/virtualization/oraclevm/overview/index.html)

Como se mencionó antes, Oracle es una de las empresas que basa su solución de virtualización en el hipervisor Xen. Soporta muchas de las funcionalidades que se buscan hoy día en un entorno virtual (consolidación de servidores, migración de máquinas virtuales, clonación, etc), posee una consola web de administración y ofrece beneficios en el licenciamiento para clientes que ya posean productos de Oracle.
No confundir con el anteriormente mencionado **Oracle VM Virtualbox**.

Sun xVM Server

Es un *hipervisor bare-metal* basado en el proyecto Xen, sobre entorno Solaris x86-64, para arquitectura Sparc.
A partir de la adquisición de Sun por parte de Oracle, el producto fue renombrado como **Oracle VM Server for Sparc**. Para más referencias, ver tecnologías LDOM de Solaris.

Virtualización a nivel S.O.



En este tipo de virtualización, a las máquinas virtuales se las denomina *"servidores virtuales".*
Esta diferencia no es sólo una cuestión de nombres. Existe una diferencia conceptual en la concepción del modelo.
Cada uno de estos servidores virtuales no requiere un kernel propio, sino que lo comparten con el sistema anfitrión que los aloja. También se los suele denominar contenedores.
Con la virtualización basada en contenedores, no existe la sobrecarga asociada con tener a cada huésped ejecutando un sistema operativo completamente instalado, y por otra parte sólo hay un sistema operativo encargado de la interacción con el hardware.

En definitiva, este modelo plantea la idea de que un sistema operativo tenga la capacidad de generar espacios lógicos independientes, para ejecutar los servidores virtuales sin ningún tipo de interacción entre ellos. Pero, ¿cómo lo logran? El concepto que está detrás de esta idea es el denominado **chroot** (http://es.wikipedia.org/wiki/Chroot).
Partiendo de este concepto, la virtualización a nivel del sistema operativo permite iniciar servidores virtuales, que trabajan de manera aislada e independiente dentro de los límites de su propio sistema de archivos root, lo que aporta seguridad en lo que a accesos a sistemas de archivos ajenos se refiere, por ejemplo, en el caso de que uno de los servidores virtuales se encuentre infectado por un virus.

Uno de sus rasgos principales es, como podemos ver, la ausencia de una capa intermedia de virtualización. Esto proporciona, prácticamente a partes iguales, tanto ventajas como desventajas, que analizaremos a continuación.

Ventajas y Desventajas

En primer lugar, este modelo ofrece un rendimiento muy próximo al que lograría cada servidor virtual en forma nativa. ¿Por qué? Al no tener un sistema operativo invitado que se comunica con un hipervisor, que luego se comunica con el hardware (o incluso con otro sistema operativo, dependiendo del tipo de hipervisor), sino que hay un único sistema operativo, claramente estaremos ganando tiempo.
Por esta misma razón, podríamos tener un número de servidores virtuales superior al número de máquinas virtuales completas que tendríamos, por ejemplo, al implementar virtualización completa.

Por otra parte, la ausencia de esta capa de virtualización intermedia implica que haya que realizar diversas modificaciones en el núcleo o sistema operativo anfitrión, sobre todo para hacer creer a los servidores virtuales que son ejecutados en un entorno exclusivo, por lo que un requisito indispensable es que el núcleo o kernel del sistema anfitrión se pueda modificar (como sabemos, esto no es posible en sistemas operativos privativos).

Otra desventaja importante es que una falla en el kernel puede provocar la caída de la totalidad de los servidores virtuales alojados (único punto de falla).

Por último, otra cuestión no menor: Debido a que estos servidores virtuales utilizan el kernel del sistema operativo anfitrión, no pueden correr sistemas operativos que difieran del instalado en éste, aunque en algunos casos es posible que ejecuten diferentes versiones de la misma distribución o incluso distintas distribuciones Linux.

Por lo general, los entornos corporativos evitan la virtualización basada en contenedores, prefiriendo los hipervisores y la opción de tener muchos sistemas operativos guests distintos. Un entorno virtual basado en contenedores, sin embargo, es una muy buena opción para empresas que ofrecen servicios de alojamiento en la nube, que necesitan una manera eficiente y segura para ofrecer sistemas operativos donde los clientes ejecuten sus servicios en ellos. En ese caso, no importa la diversidad de S.O. sino la consolidación de servidores y la seguridad.

Principales exponentes

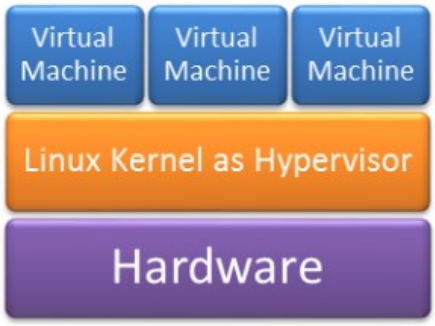
OpenVZ (https://openvz.org)

Es considerada una de las mejores opciones a la hora de consolidar servidores Linux sobre anfitriones Linux.
Permite la creación y mantenimiento de servidores virtuales aislados al introducir importantes modificaciones a nivel del kernel Linux anfitrión, creando una completa infraestructura virtual que parece real para los sistemas hospedados.
Dentro de la terminología OpenVZ los servidores virtuales son también llamados servidores privados virtuales (vps).
El producto es software libre, y se distribuye bajo la licencia GNU GPL. El alcance de este proyecto no queda ahí, sino que es la base de otra importante solución de virtualización a nivel del sistema operativo: Parallels Virtuozzo Containers, comercializada por Parallels, que por lo tanto también se dedica a apoyar OpenVZ.

Linux V-Server (http://linux-vserver.org/)

Esta es otra de las soluciones de virtualización a nivel de sistema operativo de gran importancia. El kernel Linux es utilizado para la creación de múltiples VPS (servidores privados virtuales) independientes en área de usuario. Esto se consigue gracias a diversas modificaciones del núcleo y las estructuras internas de datos, sumado a la implementación de conceptos como routing segmentado, cuotas extendidas, chroot y contexto.
Linux V-Server no sólo permite el aislamiento de los servidores privados virtuales, sino que además es posible especificar y asignar límites de uso y manipulación de los recursos disponibles como espacio en disco, consumo de memoria y procesador, adaptadores de red, etc.
Linux V-Server es la más antigua de las soluciones de virtualización a nivel del sistema operativo para Linux, estando soportado desde el núcleo 2.4 y en muy distintas arquitecturas: x86, x86_64, SPARC, PowerPC, MIPS, o ARM.

Virtualización a nivel de kernel



Este último modelo es aplicable en el mundo GNU/Linux y consiste en delegar la gestión de las máquinas virtuales en el kernel del sistema operativo, que hace las veces de hipervisor, y ejecuta a las máquinas virtuales del mismo modo en que lo hace con otros procesos del usuario.
Es importante no confundir este concepto con el modelo visto anteriormente (Virtualización a nivel Sistema Operativo), ya que en éste último existe un único sistema operativo que subdivide lógicamente los recursos, aislando a cada contenedor o vps (Virtual Private Server). En cambio, con la virtualización a nivel de kernel, un usuario podrá instalar múltiples sistemas operativos invitados, y la capa de gestión y acceso al hardware estará dentro del propio kernel del sistema operativo anfitrión.

¿Pero esto no se parece al modelo de hipervisores tipo 2, donde tenemos un sistema operativo y luego una capa de virtualización encima?
En realidad, no. Justamente en el caso de hipervisores tipo 2, éstos interceptan las llamadas al sistema, las interpretan, traducen, emulan y/o derivan al sistema operativo anfitrión, según el caso. Y en la virtualización a nivel de kernel, cada máquina virtual se comunica directamente con el kernel del sistema anfitrión, tal como lo hace cualquier otro proceso usuario.
Si se trata de máquinas virtuales con sistemas operativos no modificados, entonces esta tecnología requiere asistencia de hardware (Intel VT / AMD-V)

Ventajas y Desventajas

Una de los motivos por los que se popularizó tanto este modelo es que las funciones de hipervisor quedan cubiertas por el kernel de Linux, que se sabe está absolutamente probado y estabilizado a lo largo de los años.
Otra ventaja importante es que no se requiere hacer modificaciones en los sistemas operativos de las máquinas virtuales, aunque eso implica la dependencia de la asistencia por hardware.
Esto último prácticamente ya no se considera una desventaja, porque los fabricantes de procesadores incluyen esto desde hace varios años y cada vez con mayores prestaciones.
En términos de performance, las comparativas dejan muy bien parados a los sistemas de este tipo, de hecho se cree que en el futuro las soluciones más importantes de virtualización estarán encaminadas en esta dirección.
Pero eso se sabrá con el tiempo.

Principales exponentes

User-Mode Linux (http://user-mode-linux.sourceforge.net/)

Desde hace ya un tiempo se encuentra disponible en el kernel de Linux, aunque debe ser activado antes de usarse.
User-mode Linux, también conocido como UML, no requiere de software administrativo de manera separada para ejecutar o administrar las máquinas virtuales, sino que pueden ser ejecutadas directamente desde la línea de comandos.

UML permite al sistema operativo Linux ejecutar otros sistemas operativos Linux en el espacio de usuario, como si se tratara de procesos en el sistema operativo anfitrión, por lo que deben estar compilados para tal efecto. De esta forma, varios núcleos Linux pueden ejecutarse en el contexto de un único núcleo. El acceso por parte de las máquinas virtuales a los recursos hardware disponibles es totalmente configurable, pudiendo asignar solamente los dispositivos que consideremos estrictamente necesarios.
El uso más frecuente de este producto se da en entornos educativos y académicos y también en entornos de desarrollo y testing de aplicaciones.

Kernel-based Virtual Machine (http://www.linux-kvm.org/page/Main_Page/)

Fue incorporado en la versión 2.6.20 del kernel de Linux. Hace uso de un módulo en el kernel del sistema anfitrión (kvm.ko) que proporciona la infraestructura de virtualización. Requiere soporte de virtualización por hardware en el procesador (Intel VT o AMD-V) porque hace uso también de un módulo específico del procesador (kvm-intel.ko o kvm-amd.ko) y además utiliza un proceso Qemu ligeramente modificado, aunque está previsto que en el futuro ya no sea necesario.
KVM se inició como un proyecto Open Source por la empresa israelí Qumranet, que fue adquirida en el año 2008 por Redhat. Justamente Redhat cambió la estrategia de virtualización en sus distribuciones reemplazando a Xen por KVM. Lo mismo está ocurriendo con otras distribuciones como SuSe y Ubuntu.
El enfoque general de KVM es utilizar el módulo incluido en el kernel para todo lo referente a la gestión de las máquinas virtuales, dejando todo el resto (la gestión típica de recursos) a un S.O. probado como Linux.
El módulo KVM introduce un nuevo modo de ejecución en el núcleo Linux adicional a los modos kernel y user que de por sí aporta el kernel de forma estandarizada, llamado guest, utilizado para la ejecución de todo el código generado por las máquinas virtuales.
Proporciona funcionalidades que permiten implementar de manera sencilla migración de máquinas virtuales, tanto de manera offline como en caliente, y la toma y recuperación de imágenes o snapshots de las máquinas virtuales.
La importancia de KVM va más allá de su uso como virtualizador: fue la primera tecnología de virtualización que pasó a formar parte del núcleo Linux. Las razones por las que eso ocurrió fue su limpia implementación, que no requiere la inclusión de un hipervisor y por lo tanto es una solución puramente Linux. Uno de los principales beneficios de KVM, por ejemplo frente a Xen, es que no requiere modificación del kernel de Linux, siendo eso mismo lo que produce que no pueda obtener los rendimientos experimentados por las técnicas de paravirtualización.

Consideraciones finales sobre los tipos de virtualización

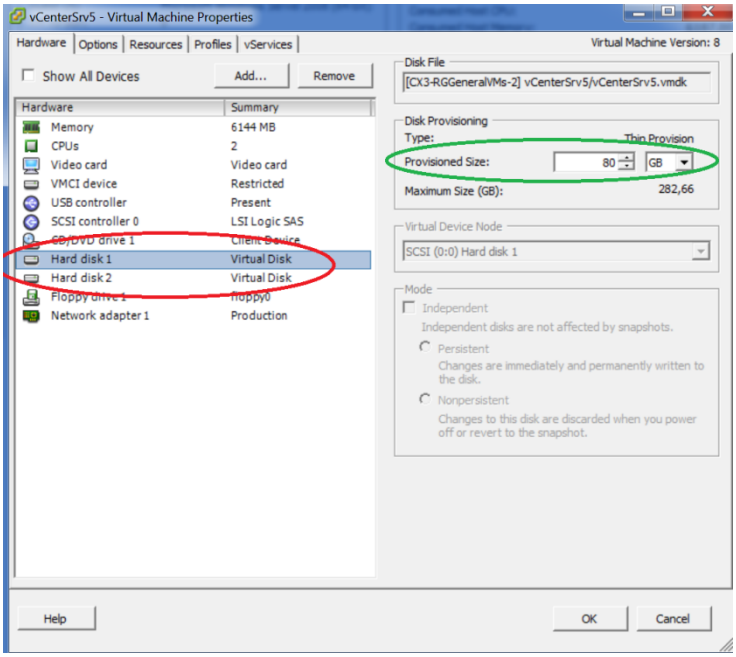
¿Cuál de los modelos expuestos es el mejor?
Esta pregunta, como tantas otras en el ámbito informático, no tiene una respuesta directa y determinante. Como vimos, cada modelo tiene sus ventajas y desventajas.
Dependiendo del tipo de instalación que se requiera, de la disponibilidad de hardware, del presupuesto económico, de los sistemas operativos que se deseen virtualizar, entre otros puntos, será más o menos adecuada la adopción de uno de estos productos.

Por ejemplo, si se pretende virtualizar un centro de datos, minimizando los costos económicos de las licencias del software de virtualización, probablemente debamos descartar los productos de VMware, aunque vale la aclaración que esta empresa ofrece productos gratuitos, pero limitados en cuanto a las funcionalidades que ofrecen.
Siguiendo con el ejemplo, podríamos pensar en un modelo con hipervisor tipo 1, por ejemplo Xen, o bien por un modelo basado en Kernel, como KVM. En estos casos, si bien el aspecto económico ya no sería un punto de conflicto, debemos pensar que el grado de madurez de estos productos es muy inferior al que puede tener VMware, sobre todo comparado a KVM.
Si nuestro centro de datos utiliza totalmente o en su mayoría sistemas operativos de la familia Windows, probablemente sea interesante adquirir el producto de Virtualización de Microsoft, que está optimizado para estos sistemas operativos.
En fin, como se puede ver, hay que hacer un análisis minucioso del proyecto de virtualización que se esté por encarar y de esa forma se estarán tomando mejores decisiones.

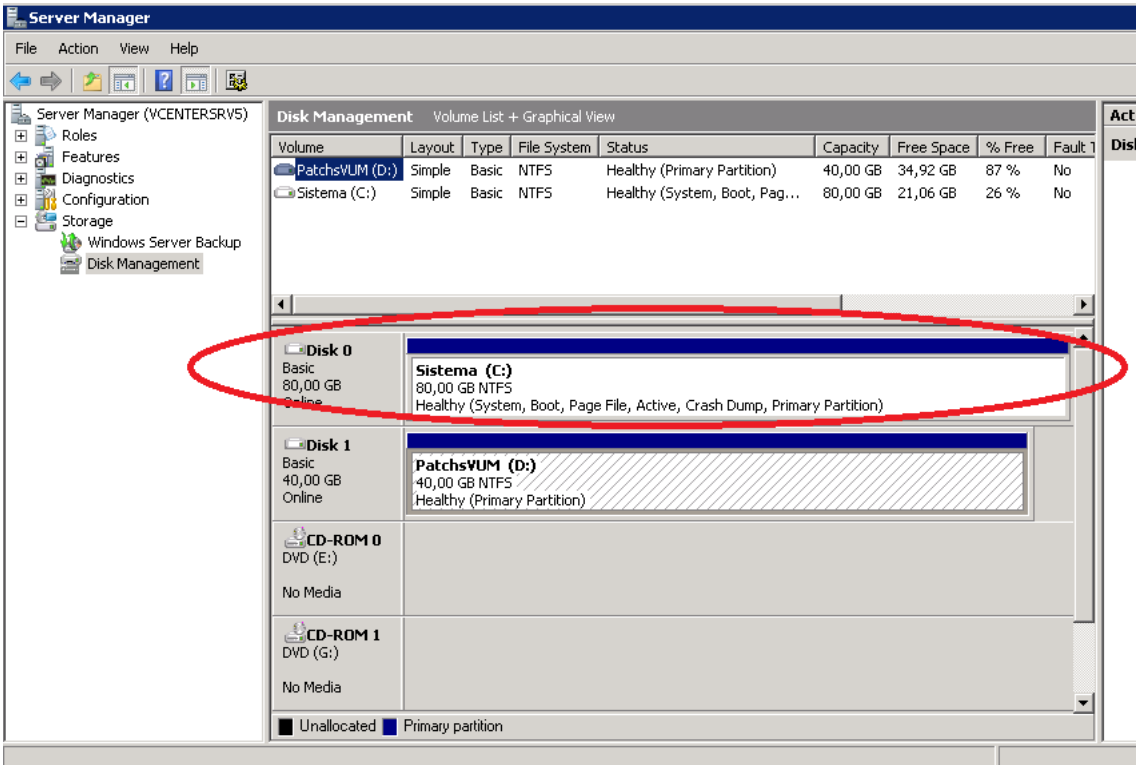
Por último, ¿cuál es la tendencia actual en términos de modelos de virtualización?
La tendencia es la suma de los aportes de cada tecnología.
Hoy en día, los principales proveedores de soluciones de virtualización utilizan conceptos de cada uno de los modelos. Por ejemplo, VMware hace uso de paravirtualización de drivers, que en definitiva es tomar una idea y adaptarla a su modelo. Xen, por su parte, implementa paravirtualización y también virtualización completa, según el caso, sólo por mencionar un par de ejemplos. Y mientras tanto, la virtualización asistida por hardware sigue en continua evolución, lo que nos garantiza un interesante futuro en el rubro.

Funciones Especiales

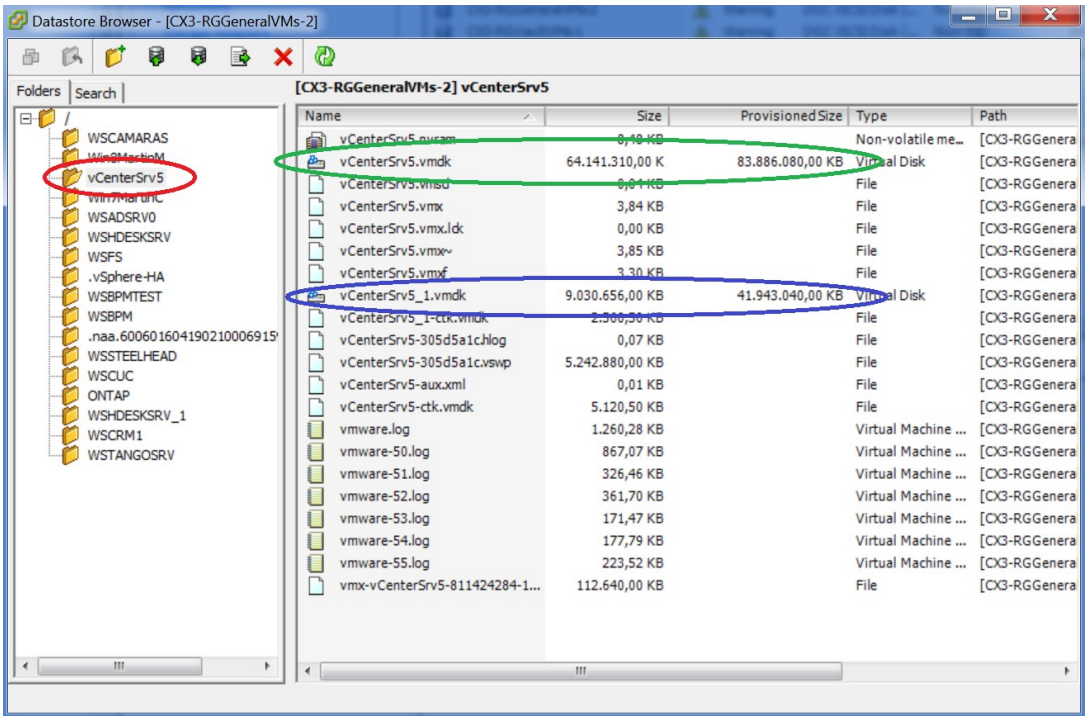
En este apartado trataremos las funciones especiales con las que se pueden trabajar cuando estamos frente a un sistema de virtualización.
Para poder entender esta parte del capítulo es muy importante que recordemos el concepto de "Encapsulación" de la máquina virtual (de ahora en más, VM). Es importante recordar que las VMs, miradas desde el punto de vista del hipervisor como sistema operativo, se encuentran encapsuladas en archivos. Según el hipervisor utilizado, estos archivos serán más o menos, pero siempre habrá archivos que representen a la VM en dicho sistema operativo. Y generalmente, uno de esos archivos (a veces más, como es el caso de VMware ESXi) representará al disco de la VM. Es decir, cuando dentro del sistema operativo de la VM (sistema operativo guest) veamos que hay un disco (por ejemplo en el administrador de discos de Windows), ese disco en realidad estará representado en el sistema operativo del hipervisor simplemente como un archivo.
Veamos un ejemplo de cómo se observa esto en el hipervisor de VMware ESXi para una VM corriendo Windows dentro de él.



En este caso podemos observar desde la aplicación de administración del hipervisor (para el caso de VMware ESXi llamada vSphere Client) las propiedades de la VM. Podemos observar, resaltado en rojo, que dicha VM posee dos discos. A modo de ejemplo, en la pantalla se ha seleccionado el primero de ellos (Hard Disk 1) y podemos ver, resaltado en verde a la derecha, las propiedades del mismo, en este caso haciendo hincapié en su tamaño de 80 GB. Es decir, desde el hipervisor le configuramos a la VM que va a tener dos discos, de los cuales el primero será de 80 GB.
Veamos cómo se ven estos dos discos desde dentro del sistema operativo guest, en este caso un Windows.



Como se podrá observar, desde dentro del administrador de discos de Windows podemos ver que el sistema operativo guest "ve" a dos discos, y el primero, resaltado en color rojo, posee como una de sus propiedades un tamaño de 80 GB.
Por último, miremos cómo se ve esta VM desde el hipervisor. Recordemos que la misma vista desde el hipervisor está "encapsulada" en archivos. Veamos pues dichos archivos:



En esta pantalla hay varios detalles a analizar. Recordemos que estamos analizando la situación "parados" dentro del hipervisor. Lo que estamos haciendo en esta pantalla es viendo los archivos que tiene el hipervisor como sistema operativo (sólo estamos viendo los archivos que

representan VMs, no los archivos que conforman al hipervisor en sí como sistema operativo). Seguimos utilizando la misma herramienta de administración del hipervisor.
Como se puede ver, esta ventana es muy parecida a la que pudiera ser un explorador de archivos de un Windows. A la izquierda tenemos el árbol de directorios de la unidad del hypervisor donde están almacenadas las VM. Podemos observar a simple vista como hay un directorio asignado a cada VM. Esto sería equivalente a cualquier sistema operativo genérico que dedica un directorio o carpeta a cada una de las aplicaciones.
En este caso está seleccionado el directorio que contiene los archivos de la VM llamada "vCenterSrv5" (resaltado en color rojo).
A la derecha observamos el contenido de dicho directorio, es decir, todos los archivos que representan a la VM dentro del hipervisor. Hemos resaltado dos archivos en particular, uno en verde llamado "vCenterSrv5.vmdk" y el segundo en rojo llamado "vCenterSrv5_1.vmdk". Estos dos archivos son los que representan en el hipervisor a los dos discos que la VM ve desde su sistema operativo guest (Windows). Como podemos observar para el primero, en la columna "Provisioned Size", su tamaño es de 80 GB, lo que se condice con el tamaño que el Windows "veía" desde dentro de la VM.
Tener este concepto de encapsulamiento claro será muy importante para entender el resto de este apartado que tratará sobre las funcionalidades especiales. En particular trataremos la problemática de ***suspender e hibernar VMs***, la ***clonación y templates***, los ***snapshots***, la ***migración*** de las VMs y el ***aprovisionamiento delgado o thin provisioning***.

Suspender e Hibernar en ambientes virtuales

Comencemos hablando acerca de qué es la suspensión. Olvidémonos por unos instantes del mundo virtual. ¿Qué significa "suspender" una PC o un servidor? ¿Qué es lo que sucede con el sistema operativo y con el hardware?
Cuando optamos por la opción de suspender una máquina (recordemos que estamos ahora hablando del mundo físico y el virtual lo hemos hecho a un lado) lo que sucede es que el sistema operativo de dicha máquina envía la orden al hardware para que entre en un estado de reposo. Esto es, los discos mecánicos dejar de girar, la CPU entra en un estado de reposo, y algunas otras cosas más. El objetivo de la suspensión es "aquietar" el sistema para que el consumo eléctrico sea el menor posible.
¿Pero qué pasa con la memoria? Cuando el sistema operativo y la máquina estaba en funcionamiento, en la memoria se encontraban gran parte de todos los procesos activos en el sistema, por lo menos, todas las estructuras necesarias para que éstos puedan ejecutarse (los PCB, etc.) además de todas las estructuras que administra el sistema operativo (tablas de directorio, tablas necesarias para que el hardware pueda funcionar correctamente, por ejemplo GDT, LDTs, etc.). La gran pregunta que aparece es qué pasa con toda esa información cuando el sistema operativo envía la orden de suspenderse.
La respuesta es que toda esa información permanece en la memoria. Sí, el estado de la memoria durante la suspensión de una máquina física se mantiene en la memoria mientras ese estado dure. ¿Pero cómo es que la máquina está apagada y la memoria aún puede contener datos? Bueno, en realidad, cuando una máquina física se suspende, su motherboard o placa madre sigue enviando pulsos eléctricos a los integrados o microchips que conforman la memoria principal del sistema para que ésta pueda mantener los datos. Recordemos de arquitectura de computadoras que la mal llamada memoria RAM (mal llamada porque lo que se quiere hacer alusión cuando se dice RAM es que la memoria es volátil, no que la memoria es de acceso aleatorio, ya que la mayoría de las ROMs utilizadas en las computadoras también son RAM) es una memoria de tipo dinámica. Esto quiere decir que la memoria constantemente

necesita que se “refresquen” sus datos mediante una serie de pulsos eléctricos ya que si no, todo el contenido se pierde. Debido a estos pulsos eléctricos constantes el hardware, por más que esté en reposo, requiere de un consumo de energía, mínimo pero consumo al fin. Es por esto que si agarramos una máquina que está en reposo y la desconectamos totalmente de la fuente de energía eléctrica, el sistema se apagará por completo y cuando lo encendamos nos daremos cuenta que el sistema operativo iniciará desde cero (posiblemente indicándonos con algún error que el sistema fue apagado de manera incorrecta llevando a cabo una serie de chequeos preventivos) en lugar de reanudar su ejecución desde el punto en que estaba cuando el mismo fue suspendido.

No hay que pensar que la suspensión funciona de manera diferente en las máquinas de escritorio que en las notebooks debido a que estas últimas no requieren que estén conectadas a una fuente de energía eléctrica durante la suspensión. La fuente eléctrica en las notebooks siempre está disponible ya que es la batería de las mismas. Sin embargo, si uno suspende una notebook y la deja sin conectar a un enchufe en la pared dándole suficiente tiempo a que la batería se agote, ocurrirá lo mismo que ocurre con una máquina de escritorio a la cual se la desconecta de la fuente eléctrica.

¿Y qué es lo que sucede con la hibernación? ¿Acaso es lo mismo que la suspensión pero con otro nombre? Definitivamente no lo es. En la hibernación el proceso es parecido pero no el mismo. El objetivo de la misma es el mismo que en la suspensión, bajar el consumo eléctrico. Pero en este último caso, el sistema operativo además de apagar todo el hardware para que éste consuma menos, transfiere el contenido total de la memoria a un archivo alojado en el disco rígido. De esta manera el sistema operativo puede apagar *todo* el hardware, incluida la memoria, ya que el contenido de esta permanece en una memoria secundaria no volátil. Por lo tanto, el consumo de energía en un sistema hibernado es nulo.

Cuando un sistema en esta última condición es reanudado (operación de *resume*) el sistema operativo lo primero que hace es transferir el contenido del archivo de hibernación a la memoria y restaurar su ejecución desde el punto en el que se encontraba cuando fue hibernado.

Una cosa a hacer notar son los tiempos en que se incurren con una u otra operación. En la suspensión de las máquinas físicas los datos permanecen en la memoria y no hay transferencia alguna entre ésta y algún archivo. Por lo tanto el tiempo entre que se da la orden de suspender y el momento en que la máquina física queda suspendida es casi inmediato. En cambio en la hibernación como es necesario transferir todo el estado de memoria a un archivo el tiempo que transcurre entre que se da la orden de hibernar y que la máquina física realmente se apaga depende de la velocidad de transferencia total al disco, siendo éste directamente proporcional al tamaño de la memoria física.

Habiendo detallado el marco teórico de la suspensión e hibernación, procederemos ahora a describir estas dos funcionalidades en los ambientes virtuales.

Se tendería a pensar que la suspensión no debería presentar mayores diferencias entre el ambiente físico y virtual. Sin embargo no hay que olvidarse que en los sistemas operativos virtuales, los hipervisores, normalmente hay varias VMs ejecutándose en el mismo hardware (recordemos que la consolidación fue el principal motivador de la tecnología virtual). Por lo tanto, un hipervisor al suspender una VM no se puede dar el lujo de “aquietar” todo el hardware apagando los discos, la CPU, etc., ya que en el mismo hardware hay otras VMs que no fueron suspendidas y todavía necesitan de los recursos del hardware para poder seguir trabajando. Por lo tanto, el hardware físico donde se ejecuta el hipervisor no sufre cambios de estado al suspender una VM.

Por lo tanto, en los sistemas operativos virtuales el hipervisor lo que hace cuando una VM se suspende es bajar o transferir todo el contenido de la memoria física que está ocupando la VM

a un archivo administrado por él con dicho contenido y luego deja a la VM en un estado suspendido, haciendo que ésta ya no demande recursos del hardware subyacente y liberando todos los recursos que ésta utilizaba, los cuales podrán ser utilizados por las demás VMs que quedan en ejecución en la misma máquina física. De esta manera, cuando el hipervisor reanuda la operación (*resume*) de la VM, realiza la operatoria inversa y transfiere el contenido de dicho archivo a la memoria principal de la máquina física continuando la VM su ejecución desde el punto en que se encontraba al suspenderse.

Para el caso de la hibernación, a nivel de sistema operativo guest (el que se ejecuta dentro de la VM) el proceso no cambia con respecto a una máquina física. El sistema operativo guest se encarga de bajar todo el contenido de "su memoria física" (recordemos que lo que el sistema operativo guest "ve" como memoria física puede no serlo en el mundo real) a un archivo administrado por él (normalmente en Windows llamado "hibernate.sys", situado en la raíz del disco "C:") y luego envía la señal para apagar su hardware (hardware virtual). Aquí hay que hacer notar que lo que se apaga es el hardware virtual, es decir, lo que el hipervisor ve es que la VM se apaga, pero el hardware físico que es administrado por el hipervisor permanece encendido dando servicio a las demás VMs que en él se ejecutan.

Pero entonces, si en los sistemas operativos virtuales la suspensión de la VM transfiere todo el contenido de la memoria que la VM está ocupando a un archivo (por el problema que comentábamos antes que no se puede apagar todo el hardware) y la hibernación también hace lo propio, ambos dejando el hardware físico encendido, ¿qué diferencia hay entre suspender e hibernar una VM? La diferencia radica en dónde se encuentra o "vive" el archivo que contiene toda esa información del estado de la memoria de la VM. En el caso de la suspensión, la operatoria de suspender es llevada a cabo por el hipervisor y el archivo al que se transfiere todo el contenido de la memoria de la VM reside en el hipervisor, es decir, normalmente es un archivo más que se encuentra en el mismo directorio de la VM dentro del File System del hipervisor, y que pasa a ser uno de los tantos archivos que representan a la VM, en este caso una VM suspendida.

En cambio cuando hibernamos una VM es archivo con el contenido de la memoria de la VM reside o "vive" dentro del File System del sistema operativo guest de la VM. Y yendo un poco más lejos, físicamente, los bloques que ocupa ese contenido pasan a ser bloques que físicamente están dentro del archivo que representa al disco virtual de la VM que se encuentra en el File System del hipervisor.

Vayamos más despacio con la última parte del párrafo anterior. Dentro del sistema operativo guest de la VM (tomemos como ejemplo que es un Windows) se da la orden de hibernar el sistema. El sistema operativo comienza a transferir toda la información que se encuentra en "su memoria física" a un archivo llamado "hibernate.sys" que reside en la raíz del disco "C:". Pero la acción de que el sistema operativo guest escriba en el disco "C:" en realidad lo que está haciendo es que el hipervisor escriba dichos datos en el archivo que representa a ese disco "C:" el cual es administrado por el hipervisor y reside en su File System (si hay dudas al respecto de esto volver a leer con detenimiento la primera parte de esta sección). En cambio en la suspensión, el hipervisor transfería todo el estado de memoria de la VM a un archivo también administrado por él pero el sistema operativo guest ni se enteraba de esto.

Lo que sí tiene que quedar claro es que tanto en la suspensión como en la hibernación, se tiene que transferir el contenido total del estado de la memoria de la VM a un archivo (ya sea que resida dentro del sistema operativo guest (suspensión) o fuera es éste (hibernación)), y por lo tanto el tiempo para llevar a cabo las dos operatorias serán similares.

Clones y Templates de Máquinas Virtuales

Como su nombre lo indica, un clon de una VM es una copia exacta de la misma. Imaginemos lo que pasaría si quisiéramos hacer un clon de una máquina física. Lo primero que tendríamos que tener es un hardware idéntico a la máquina a clonar.
¿Por qué este requerimiento? Cuando un sistema operativo se instala sobre cualquier hardware, el instalador del sistema operativo automáticamente detecta (si no lo hiciese el usuario debería hacerlo manualmente) el hardware sobre el cual se está instalando. En base a esta auto detección el instalador del sistema operativo escoge los drivers indicados para manejar cada componente instalado. Por ejemplo, selecciona el driver adecuado para trabajar con la placa de red, con la controladora de discos, con la placa de video y así con cada componente.
Si luego quisiéramos hacer una copia exacta del sistema operativo (lo que se conoce en la jerga como "hacer una imagen") e instalar dicha imagen en otro hardware físico, deberíamos cerciorarnos que dicho hardware sea exactamente igual al primero, ya que la imagen en cuestión tiene todos los drivers para trabajar con los componentes del primer hardware. Además deberíamos contar con un software adecuado para poder "sacar" esa imagen del sistema físico original (por ejemplo "Ghost", "Acronis", etc.). Luego, si utilizáramos dicha imagen en un hardware con diferentes componentes muy posiblemente veríamos pantallas de error al iniciar el sistema operativo en el nuevo hardware indicándonos que no se encuentran los drivers para trabajar con los nuevos componentes.
Cuando intentamos hacer lo propio con las VMs en un sistema operativo virtual el problema anterior no existe, ya que los sistemas operativos virtuales presentan siempre el mismo hardware a las VMs, sin importar el hardware físico subyacente donde corre el hipervisor. Por lo tanto, los clones de las VMs no sólo sirven si la VMs clonadas se ejecutarán en el mismo servidor o máquina física sino también si lo harán en otro servidor, siempre y cuando se tenga instalado en el segundo servidor el mismo hipervisor.
¿Pero cómo se realiza un clon de una VM? ¿Necesitamos contar también con el software de imágenes que necesitábamos en el mundo físico? La respuesta es no. Si recordamos nuevamente del principio el concepto de "encapsulación", podemos darnos cuenta que clonar una VM simplemente es entrar en la consola de administración del hipervisor y hacernos de una copia de todos los archivos que componen la VM que queremos clonar. Recordemos una vez más que dichos archivos son la fiel representación de la VM en el hipervisor, y por tanto, si podemos copiarlos entonces vamos a estar clonando dicha VM.
El proceso sería similar a si tuviéramos una aplicación cualquiera en un sistema operativo de uso general la cual constara de un par de archivos. Si en ese sistema operativo copiáramos ese par de archivos, estaríamos haciendo un clon de esa aplicación. Por lo tanto, dichos archivos de la aplicación son al sistema operativo de uso general lo que los archivos que representan a la VM son a un hipervisor.
Hablemos ahora un poco acerca de los *templates*. Un template es simplemente un molde de una VM que se utiliza para crear nuevas VMs a partir de él. Es por definición también una VM, nada más que generalmente es una VM que fue instalada una vez y luego no se enciende más, o sólo lo hace para actualizarse y quedar apagada. De hecho, en algunos hipervisores como por ejemplo VMware ESXi cuando una VM se convierte en template la misma ya no se puede volver a encender más. Para poder volverla a encender, por ejemplo para actualizarla, hay que convertirla primero en VM, luego encenderla, actualizar lo que haya que actualizar, apagarla y finalmente volverla a convertir en template.
La idea de los templates es que sean lo más genérico posibles. ¿Qué se debería instalar en una VM que se convertirá luego en template? Básicamente se instalará el sistema operativo guest,

las últimas actualizaciones de éste; las aplicaciones comunes a todas las VMs, esto es, compresores, antivirus, etc.; en fin, todo aquello que estará presente en todas las VMs del sistema. Pero no se debería instalar en un template aplicaciones como base de datos, servidores de correo, etc., ya que la idea es que una vez que se tenga el template se cree una nueva VM a partir del mismo y sobre esa nueva VM ahí instalar por ejemplo un motor de base de datos.
El principal objetivo de los templates es que el administrador de la infraestructura virtual no pierda tiempo en instalar dentro de cada nueva VM el sistema operativo de éste, sus actualizaciones y todo aquello que tienen todas las VMs en común. Luego, si el administrador necesita crear una nueva VM, por ejemplo para instalar un servidor de correo electrónico, lo que haría teniendo un template es crear la nueva VM a partir del template diferente, y una vez que se tiene la misma ahí instalar el servidor de correo.
En definitiva, *clones y templates* son casi lo mismo, es crear nuevas VMs a partir de otras. La gran diferencia entre ambos radica en que en los clones, tanto la VM original como su clon continúan su "vida" o ejecución independientemente y cada una se va "ensuciando" con el correr del tiempo típico del uso de cualquier máquina. En cambio con los templates uno siempre tiene el template vacío, "limpio", recién instalado, ya que el mismo no tiene ejecución, quedó tal cual estaba cuando se lo convirtió en template, y esto permite disponer siempre de una VM base recién instalada para crear nuevas VMs a partir de ella.
Pero si tanto los clones como los templates son una fiel copia de la VM original, ¿esto no acarrea problemas con las nuevas VMs creadas a partir de ellos (pensando en que van a tener los mismos nombres de hosts, las mismas direcciones IP, etc.)? Bueno, los sistemas operativos, como Windows o Linux, que corren dentro de las VMs poseen herramientas para "customizar" los datos que le dan identidad a ellos (como el nombre de host, direcciones IP, dominio al que pertenecen, password de adminstrador, etc.). En el caso de los sistemas operativos Windows esa herramienta es *sysprep* (de System Preparation). Los hypersores al clonar una VM o al crear una nueva a partir de un template permiten la automatización en la ejecución de dichas herramientas de tal forma que cuando se están creando las nuevas VMs (tanto desde un clon como desde un template) asignarle todos esos nuevos datos a la nueva VM basados en la entrada y opciones seleccionadas por el administrador.
A modo de ejemplo, supongamos que tenemos una VM llamada VM_A la cual tiene instalado: un Windows; como nombre de host, "vm_a"; como IP, la dirección 192.168.1.20 y como password de adminstrador, "pepito". Al clonar dicha VM, el administrador le dirá al hipervisor que la nueva VM se llamará VM_B, su nombre de host será "prueba" y su nueva dirección IP será 172.16.1.35. Al terminar de ingresar todos los datos el hipervisor comenzará haciendo una copia de todos los archivos que representan y encapsulan a la VM_A. Luego de terminar la tarea, el hipervisor encenderá la nueva VM (VM_B) y en el proceso de arranque del Windows en la VM_B se disparará el proceso de *sysprep* que "inyectará" todo los datos que había ingresado el administrador dándole la nueva identidad al Windows de la nueva VM_B.

Snapshots

Los snapshots son una de las mejores funcionalidades con lo que podemos trabajar en el mundo de los sistemas operativos virtuales. Es una tecnología que no es exclusiva de éstos, sino que se utilizan en varios rubros, como por ejemplo los sistemas de almacenamiento modernos.
Un snapshot es como la palabra lo indica una foto de una VM en un determinado momento. Un snapshot también se lo puede ver como un punto de restauración.
Los snapshots permiten ser tomados en un determinado momento y luego, de ser necesario, poder volver a ese punto en el tiempo.

Imagínense la siguiente situación. Un administrador de una VM tiene que instalar dentro de la misma un parche para una aplicación recién salido al mercado. Lo que el administrador no sabe es si la instalación del parche afectará a la producción de la aplicación o de la VM. Trabajando y valiéndose de los snapshots lo que el administrador puede hacer es tomar uno antes de la instalación del parche. Luego instalar el parche y dejar pasar un tiempo prudencial para ver qué sucede. Si la instalación del parche efectivamente afecta a la aplicación que corre dentro de la VM, entonces el administrador puede volver el tiempo hacia atrás, al momento en que se había tomado el snapshot y que el parche aún no estaba instalado y así esperar a que salga un nuevo parche que no afecte a la aplicación. De lo contrario, como ocurre normalmente, si la instalación del parche no afecta a la aplicación, luego de pasado el tiempo prudencial el administrador puede por optar eliminar el snapshot y todo continúa normalmente.

¿Pero cómo es que funciona esta funcionalidad? ¿Qué es lo que hace posible poder volver el tiempo atrás en las VMs? Otra vez acá es esencial comprender bien el concepto de encapsulación de las VMs en archivos, sobre todo el concepto de que los discos de una VM están representados por uno o más archivos en el hipervisor.

Lo que sucede en el hipervisor cuando en una VM se toma un snapshot es que los archivos que representan al disco de la VM en cuestión queda congelado (de hecho, en read-only o sólo lectura). A partir de ese momento, cuando dentro de la VM, en el sistema operativo guest, se realiza una escritura en disco, esa escritura es realizada en el hipervisor en un nuevo archivo que representa ahora al disco de la VM llamado archivo *delta* o *redolog*. ¿Por qué ese nombre *delta*? Porque justamente lo que está guardando a partir de ese momento son todas las modificaciones (el delta) que la VM fue haciendo desde que se tomó el snapshot. Es importante notar que ahora, el hipervisor tendrá que analizar de donde leer los datos o bloques cada vez que la VM ejecuta un comando de lectura en disco, ya que tiene dos opciones: que el dato leído nunca se modificó desde que se tomó el snapshot, con lo que tendrá que realizar la lectura desde el archivo original que ahora se encuentra en estado de sólo lectura o congelado; o bien si el bloque o dato a leer es un bloque que se modificó luego de que el snapshot de tomó, el bloque deberá ser leído desde el archivo delta.

Hay algunas consideraciones a tomar en cuenta cuenta cuando se utilizan snapshots.

La primera es que dentro de la VM el sistema operativo guest nunca se entera que por “afuera” de él alguien tomó un snapshot.

Segundo es que al sacar snapshots se incurre en una penalización en la performance, ya que ahora por cada comando de lectura que la VM ejecuta el hipervisor debe decernir de qué archivo leer la información.

Tercero, por definición, el archivo delta que se genera cuando se toma un snapshot puede crecer hasta un tamaño igual que el archivo base que queda congelado. Por lo tanto, trabajar con snapshots me hace aumentar el espacio en disco utilizado en el hipervisor. Esto es muy importante, porque uno piensa que si tiene una VM creada con un disco virtual de 80 GB, en el hipervisor el espacio en disco utilizado por la misma nunca será mayor a 80 GB más lo que consumen los demás archivos que conforman la VM. Sin embargo, por cada snapshot que se toma en la VM, se está creando en el mismo un archivo delta que potencialmente puede crecer hasta 80 GB. Lo que se pensaba que nunca podía ocupar más de 80 GB puede ahora ocupar 160 GB!.

Cuarto, y casi más importante que cualquier otro punto con respecto al uso de los snapshots, **los snapshots NO SON UN BACKUP.** Este punto es extremadamente importante ya que hay muchísimos administradores con muchos años de experiencia en la administración de ambientes virtuales que todavía piensan que tomar snapshots en una o varias VMs equivale a tener backups de la misma. Si uno se pone simplemente a analizar el funcionamiento de los snapshots, donde cada vez que se toma un snapshot se congela el archivo base que representa

al disco virtual de la VM y se crea uno nuevo que es un delta con las modificaciones, si se pierde el archivo base **sólo me quedan las modificaciones**, con lo que no puedo seguir trabajando con la VM. Por lo tanto, un snapshot **no es un clon**. Sin embargo, el uso de snapshots hace posible la toma de backups de las VMs con éstas encendidas. ¿Por qué? Cuando una VM se encuentra encendida el hipervisor mantiene un bloqueo o lock sobre los archivos que representan al disco virtual de la VM. Esto es debido a que el hipervisor está escribiendo en dicho archivo todo lo que esa VM envía a escribir en su disco dentro de la misma. Si uno quisiera tomar un backup de la VM copiando desde el hipervisor los archivos que la representan, no podría porque existe ese bloqueo del hipervisor sobre los mencionados archivos que me imposibilita leerlos desde otra aplicación. En cambio, si antes de tomar el backup uno le saca un snapshot a la misma, los archivos que representan a los discos virtuales quedan liberados del bloqueo ya que quedan en modo de sólo lectura. Es ahí donde entra en juego la aplicación de backup que ahora sí puede leer completamente esos archivos y guardarlos en una ubicación segura. Por lo tanto, los snapshots no son backups de las VMs, pero sí son una herramienta o funcionalidad que me permiten tomar backups.

¿Qué sucede cuando uno quiere hacer uso de la función de volver el tiempo atrás al momento de haber tomado el snapshot? Simplemente lo que hace el hipervisor es descartar todos los cambios que se grabaron en el archivo delta (eliminando el archivo) y volver a trabajar directamente sobre el archivo base, restaurando el bloqueo o lock sobre el mismo y haciendo que nuevamente las escrituras de la VM se realicen sobre el mismo.

¿Y qué sucede por otro lado si lo que se quiere es mantener el estado de la VM después de haber tomado un snapshot pero quiere descartar al snapshot en sí? Es decir, se pretende continuar con la ejecución de la VM en forma normal sin snapshot alguno. Lo que hace el hipervisor es hacer un “commit” o volcado de todos los datos que se encuentran en el archivo delta (que mantiene todos los cambios luego del snapshot y que son cambios que no se quieren perder) al archivo base.

Muchas veces ayudan mucho las analogías para entender algunos conceptos un poco más complejos. En este caso una buena analogía a los snapshots podría ser la lectura de un libro. Imaginemos que el transcurso del tiempo de una VM es el equivalente a pasar las hojas de un libro. El tiempo de una VM comienza en la página 1 y va avanzando por las páginas 2, 3 y así sucesivamente. Llegando un tiempo “T” cualquiera, supongamos que se está en la página 10, se saca un snapshot a la VM. El equivalente en nuestra analogía del snapshot sería poner en esa página 10 un señalador que me permita en algún momento futuro volver atrás. Pero la “vida” de la VM continúa como si nada hubiera pasado (recordar la primera consideración de trabajar con snapshots que habíamos tomado en cuenta en algunos párrafos anteriores). Continúa así y las páginas del libro siguen pasando: 11, 12,…,20. En la página 20 uno toma una de dos decisiones. Si la decisión es volver el tiempo atrás al momento de tomar el snapshot, lo que hace es ir a la página señalada por el señalador (página 10) y como verán, todo lo que se había avanzado en el libro desde la página 10 a la 20 se pierde y se tiene que volver a releer. Fíjense que el señalado aún está y aunque nuevamente comience a avanzar por las páginas a partir de la 10 siempre tendremos la posibilidad de volver hacia atrás a la página marcada por el señalador.

Si en cambio la decisión tomada en la página 20 del párrafo anterior es eliminar el punto de restauración, es decir, eliminar el snapshot, en nuestra analogía este acto se vería reflejado como simplemente sacar el señalador de la página y continuar la lectura normalmente por la página 21, 22 y así sucesivamente.

Es muy importante entender el funcionamiento de los snapshots, porque normalmente los hipervisores que se utilizan a menudo hacen referencia a esa eliminación de los snapshots como la opción “borrar snapshot” o “delete snapshot”. Esto que es natural, porque lo que se

busca es borrar el snapshot, al momento de administrar una infraestructura real el administrador ve la palabra “delete” y piensa que lo que se van a borrar son los datos que se modificaron en la VM luego de tomarse el snapshot, y no el snapshot en sí.
Por lo tanto, hay que ser consciente de que borrar un snapshot significa hacer un “commit” de los cambios guardados en el archivo delta al archivo base y lo que se está borrando es el punto de restauración o la posibilidad de volver hacia atrás, pero ni los datos ni los cambios son eliminados en dicha operación.

Migración de Máquinas Virtuales

La migración de las máquinas virtuales simplemente se refiere al “movimiento” de las mismas.
Mirado desde este punto de vista, como simplemente “movimiento”, parecería ser que la migración de las VMs no es ninguna función especial. De hecho el movimiento de archivos es una función de casi todos los sistemas operativos.
Sin embargo esta migración de las VMs va un poco más allá del simplemente movimiento de archivos. Analicemos un poco mejor la situación a continuación.
Dijimos al principio que una VM estaba compuesta o “encapsulada” en archivos, de la misma manera que una aplicación común y corriente está también compuesta o encapsulada en archivos. De hecho esos archivos son el “programa” de la aplicación.
Ahora bien, todos sabemos que cuando el sistema operativo ejecuta un programa éste se transforma en un proceso.
Lo mismo ocurre cuando un sistema operativo virtual “ejecuta” una VM (se dice que “enciende” la VM). Ahora la VM no sólo es ese conjunto de archivos que la representan, sino que también es uno o varios procesos en ejecución.
Es muy importante entender que una máquina virtual encendida es dos cosas: la primera un conjunto de archivos y la segunda un conjunto de procesos. Lo mismo ocurre con cualquier programa en ejecución. Una aplicación cualquiera que se ejecuta en un sistema operativo es el archivo del programa (o los archivos si son varios) y uno o varios procesos en ejecución.
Por supuesto, al igual que los programas, cuando una máquina virtual se encuentra apagada simplemente es ese conjunto de archivos que la representan.
Volviendo a nuestro tema de migraciones entonces, cuando decimos que vamos a migrar a mover una máquina virtual tenemos que ver bien qué “cosa” de la máquina virtual vamos a mover. ¿Vamos a mover los archivos o los procesos que la representan?
Por otro lado no debemos olvidar que la máquina virtual tiene estados, encendido, apagado, etc. Por lo tanto, ¿vamos a mover a la máquina virtual apagada o encendida? ¿o habrá otro estado?
Mover una máquina virtual apagada implícitamente nos dice que vamos a estar moviendo en el sistema operativo virtual (el hipervisor) los archivos que la representan desde la ubicación actual a una nueva, ya que al estar apagada no existen procesos que la representen todavía.
¿Pero acaso dijimos “mover una máquina virtual encendida”? ¿Y esto qué significaría? Mover sus archivos sería lo mismo que cuando está apagada, es decir, llevarlos desde la ubicación actual hasta una nueva, sumándole alguna dificultad extra al proceso ya que ahora la máquina virtual está encendida y está realizando cambios en los mismos (recordemos que todo lo que el sistema operativo guest de la máquina virtual esté escribiendo en realidad va a estar modificando en el hipervisor el archivo que represente al disco rígido de la VM).
Pero ahora también está en juego los procesos de la máquina virtual en el hipervisor. ¿Acaso se pueden mover estos procesos? ¿Y a dónde se moverían? Porque los procesos “viven” o se ejecutan en el hipervisor.

Bueno, migrar los procesos de una máquina virtual encendida es algo que hoy la mayoría de los hipervisores del mercado lo soportan y significa mover dichos procesos de una máquina física (donde está ejecutándose el hipervisor) a otra máquina física donde esté ejecutándose otra instancia o instalación del mismo hipervisor.
Por lo tanto, tenemos que entender que la migración de las máquinas virtuales puede involucrar por un lado la migración de los archivos que la representan desde una ubicación a otra en el hipervisor y por el otro la migración de los procesos que la representan de un hipervisor a otro en diferentes máquinas físicas. Es importante aclarar que uno no implica el otro. Es decir, son dos tipos de migraciones por separado e independientes entre sí y que generalmente se realizan por separado.
Por otro lado, es muy importante entender el estado de la máquina virtual al momento de efectuarse la migración. La misma puede estar apagada, encendida o suspendida. En la jerga de virtualización se suelen asociar a las migraciones con las VM apagada el nombre de migración en frío o "cold migration" y a las migraciones con la VM encendida como migración en caliente o "live migration".

Problemáticas y puntos a tener en cuenta para las migraciones

Las migraciones pueden ser tan sencillas como simplemente copiar archivos de una ubicación a otra (por ejemplo la migración en frío de los archivos que representan a una cierta máquinas virtual) o tan complejo como continuar la ejecución desde la siguiente instrucción en una máquina física totalmente diferente.
Los diferentes hipervisores del mercado terminan resolviendo los problemas que se suscitan con las migraciones de diferentes maneras. No es objeto de este material ilustrar cómo se llevan a cabo las diferentes migraciones en cada uno de los hipervisores, pero sí hacerle pensar al lector acerca de los diferentes problemas con los que cualquier hipervisor se topa al momento de realizar las migraciones.

- ***Migraciones en frío****:* Este tipo de migraciones no representan mayores problemas. Simplemente se trata de migraciones de los archivos que representan a la VM a migrar de una ubicación del hipervisor a otra. Nótese que dicha ubicación de archivos puede ser local al hypervisor (discos locales residentes en la máquina física donde se ejecuta el hipervisor) o remota (archivos accedidos por el hipervisor a través de una red SAN como ya se vio en el módulo de entrada/salida en el apartado de tecnologías de almacenamiento).
  Estas migraciones pueden entonces ser llevadas a cabo entre diferentes ubicaciones locales, remotas o entre ambas. Por ejemplo se podría querer migrar una VM que reside en un disco local del hipervisor a un volumen de datos que se encuentra en un sistema de almacenamiento externo accedido por Fibre Channel.

- ***Migraciones en caliente****:* Estas son las migraciones más complejas a llevarse a cabo. De hecho muchos hipervisores han tenido esta tecnología funcionando correctamente en los últimos años (2008). Las migraciones que se pueden realizar en este estado son de archivos (entre dos ubicaciones accesibles al hipervisor) o de procesos (los procesos que representan a la VM migran de una máquina física a otra). Es importante aclarar aquí para que no haya confusiones que cuando se dice que se migran los procesos que representan a una VM de una máquina física a otra se está migrando la VM completa, es decir, se migra la ejecución de una VM que se está ejecutando en el servidor A al servidor B sin que haya tiempo sin servicio (downtime). No debe bajo ningún punto de vista entenderse esta migración como que se migran los procesos que se ejecutan dentro del sistema

operativo guest a otro sistema operativo guest en otra VM en otra máquina física. Lo que se migra es la VM completa.

- *Migración de archivos:* En este caso teniendo la VM encendida el hipervisor inicia una migración de los archivos que representan a la VM de una ubicación en su File System a otra (recordemos que dicho File System puede hacer referencia a archivos locales o remotos). En este caso uno de los problemas con que se encuentra el sistema al realizar dicha operatoria es que mientras se están copiando todos los archivos que representan a la VM (entre ellos los que representan a los discos virtuales), la VM sigue realizando operaciones de entrada y salida, y cada vez que el sistema operativo guest de la VM escribe un dato de la VM dicho dato se escribe en el archivo que representa al disco en el hipervisor. Por lo tanto la pregunta es, ¿qué hago con los cambios que se van realizando en la VM mientras dura el proceso de copia?

  Para poner en contexto esta situación daremos un ejemplo. Supongamos que tenemos una VM que tiene un disco de 50 GB. El sistema operativo guest "ve" una máquina con un disco rígido de 50 GB. Por debajo, el hipervisor representa a dicho disco rígido como simplemente un archivo de 50 GB.

  Dentro de la VM existe un archivo en el "/home/usuario" llamado "log.txt" que guarda estadísticas de la cantidad de paquetes recibidos por la placa de red una vez cada 30 segundos.

  En determinado momento el administrador de la infraestructura virtual decide migrar dicha VM desde la ubicación actual donde se encuentra a una nueva ubicación para liberar espacio en el volumen actual. La tarea comienza y en un momento el sistema comenzará a copiar el archivo que representa al disco rígido de la VM (el que ocupa 50 GB). Como es un archivo bastante grande posiblemente va a estar varios minutos, incluso horas, copiandolo. ¿Qué pasa con los datos que se van agregando dentro de la VM al archivo "log.txt"? Porque se supone que la tarea de migración es totalmente transparente al usuario, y por supuesto, no debería haber pérdida de ningún dato.

  El hipervisor va por debajo copiando bloque por bloque el archivo de 50 GB que representa al disco rígido de la VM. En determinado momento copiará los bloques donde se encuentra físicamente el contenido del archivo "log.txt" (notar que el hipervisor no ve qué hay dentro del archivo que representa al disco rígido virtual, no "ve" dentro del archivo, para él simplemente es un archivo más; pero claramente dentro de ese archivo está la información que el sistema operativo guest está viendo dentro de su disco rígido). Continuará copiando los bloques uno a uno (incluido los que tienen el contenido del archivo "log.txt") y en determinado momento, dentro de la VM, el sistema operativo guest volverá a escribir las estadísticas de la placa de red. Por lo tanto, ¿qué contenido tendrá el archivo que representa al disco rígido de la VM en la nueva ubicación, más precisamente, los bloques que representaban al archivo "log.txt" dentro del guest?

  Es más, en ese momento mientras se está llevando a cabo la copia, ¿la VM desde dónde estará leyendo los datos cuando el SO Guest realiza una lectura? ¿Desde el archivo que representa al disco en la ubicación original o en la nueva?

  Estas son las problemáticas con las que se encuentran los hipervisores al momento de realizar una migración de archivos de una VM en caliente.

  A modo de ejemplo, simplemente para que vean cómo se pueden solucionar estos problemas explicaremos cómo lo hacía el hipervisor de VMware ESXi hace un tiempo y cómo lo hace ahora.

En versiones anteriores, cuando el administrador iniciaba una migración de archivos de una VM de un volumen a otro, el hipervisor inmediatamente tomaba una especie de snapshot del archivo que representaba al disco de la VM e iniciaba la copia leyendo el archivo original y copiandolo a la nueva ubicación. Mientras tanto, todos los cambios que la VM iba introduciendo (comandos de escritura dentro del SO guest) se iban escribiendo en un archivo delta (tal cual pasa con los snapshots ya explicados). Cuando el sistema terminaba de copiar el archivo original (en nuestro ejemplo anterior el archivo de 50 GB), en el archivo delta se encontraban todas las modificaciones que la VM había realizado desde el momento en que se había iniciado la copia. Por lo tanto procedía a copiar el contenido de dicho archivo delta introduciendo dichos cambios directamente en el archivo de destino de 50 GB. De esta manera, en el archivo de destino ahora tenía todos los datos del archivo original más los cambios que se habían realizado mientras había durado la copia. Pero mientras que se realizaba la copia del contenido del archivo delta, la VM otra vez volvía a introducir cambios. Como el tamaño del archivo delta era mucho más chico que el del disco original, no daba lugar a que mientras durara el copiado de estos nuevos datos la VM pudiera introducir muchos cambios nuevos, pero igualmente algunos cambios se realizaba. Por lo tanto estos nuevos cambios realizados en esta segunda fase de copiado se volvían a escribir en un nuevo archivo delta. Al terminar esta segunda fase el sistema volvía a iniciar una nueva fase copiando ahora el contenido del archivo delta más pequeño y volviéndose a repetir el proceso de crear un nuevo archivo delta que debería ser cada vez menor. Cuando se llegaba a un determinado umbral de tamaño de este archivo delta (muy pequeño) el sistema suspendía las escrituras temporalmente de la VM, copiaba el último de los archivos delta al destino y reanudaba las escrituras de la VM que ahora comenzaba a trabajar sobre el archivo que representaba al disco rígido virtual en la nueva ubicación.

El problema que tenía este método era que si la tasa de cambios (la tasa de escritura) de la VM era más elevada que la tasa de transferencia entre la ubicación del archivo original y el destino, el proceso nunca terminaba, ya que los archivos delta eran tan grandes que nunca llegaban al umbral para “freezar” las escrituras.

Por este motivo VMware cambió la forma de realizar esta copia en caliente. En la actualidad, el hipervisor al iniciar una migración de los archivos de una VM comienza directamente a copiar todos los bloques del archivo que representa al disco rígido virtual. Mientras tanto la VM continúa su ejecución, ejecutando lecturas y escrituras sobre su disco. Si la VM ejecuta una escritura en un bloque que aún no se copió por debajo en el hipervisor a la nueva ubicación, dicha escritura se escribe directamente en el archivo original, ya que en algún momento el hipervisor llegará hasta dicho bloque y lo copiará al destino. En cambio si la VM escribe un bloque que ya fue escrito por el hipervisor a la nueva ubicación, entonces el sistema “divide” la escritura en dos (se dice que se realiza un “Split” del IO) y se generan dos comandos de escritura con igual contenido, pero uno se escribe en el archivo original y el otro en el archivo de destino.

Este cambio en la forma de realizar la migración permitió que, de una sola pasada, cuando el hipervisor termina de copiar el archivo desde su ubicación de origen al destino, en este último ya se encuentra toda la información actualizada sin necesidad de realizar nuevas pasadas. La contrapartida es una penalización en la performance del proceso, ya que todos los comandos de escritura que se realizan sobre bloques ya copiados al destino se duplican, pero se asegura un éxito del proceso.

- *Migración del proceso de la VM:* Esta fue la funcionalidad que cuando se anunció por primera vez todo el mundo quería probar. Parecía casi imposible observar una VM ejecutándose en una máquina física y de repente, sin que el usuario de la VM lo notara, la VM estuviera ejecutándose en una nueva máquina, tal vez incluso a kilómetros de distancia. Para poder llevar esta migración a cabo generalmente se define una red LAN dedicada entre los servidores físicos que ejecutan los hipervisores para transportar todo el estado de ejecución de las VMs a migrarse de uno a otro servidor físico.

  El proceso generalmente se basa en comenzar a migrar todo el estado de ejecución de la VM a migrarse a través de la red dedicada para tal fin y, como la VM sigue en producción y funcionando, un proceso en el hipervisor va tomando cuenta de todas las páginas de memoria que se van modificando (formando una especie de archivo de bitmap) mientras la migración se lleva a cabo. Cuando la primera fase de la migración de datos se completa se freeza la VM temporalmente para copiar las páginas modificadas y se reanuda por último la ejecución desde la siguiente instrucción en el servidor de destino.

  Si bien está fuera del alcance de este curso, una cosa interesante que pasa luego de que la migración se ha completado es un punto relacionado con la red o networking. Cuando la VM se estaba ejecutando en el servidor de origen, los switches físicos de la red tenían cargados en sus tablas la MAC Address de la placa de red virtual de la VM asociada a un cierto puerto (el puerto donde está conectado el servidor físico donde la VM se estaba ejecutando). Al migrarse la VM y pasar su ejecución al servidor de destino, los switches todavía "ven" a la VM en el viejo puerto. Es decir, ellos tienen asociado la MAC de la VM en el puerto del servidor original, pero la VM ahora se está ejecutando en el servidor de origen. Por lo tanto, todos los mensajes de red dirigidos a la VM serán erróneamente dirigidos (forwardeados) al servidor que ya no contiene a la VM. Es por esto que una de las primeras tareas que se realizan una vez migrada la VM es enviar o forzar un ARP gratuito conteniendo como dirección de origen a la MAC de la VM. De esta forma, los switches tienen la obligación de actualizar sus tablas de direcciones y asociar así la MAC de la VM al nuevo puerto (puerto donde está conectado el servidor de destino donde ahora se ejecuta la VM).

  Pero pensemos un poco desde el punto de vista del sistema operativo virtual (el hipervisor) los problemas con que se encuentra al tratar de realizar dicha operación.

  - Primer punto a observar. Si una VM cambiará su ejecución de una máquina física a otra, ¿acaso la nueva máquina física no tiene que tener acceso a los archivos que representan a dicha VM? Absolutamente sí. Para que una nueva máquina física pueda continuar con la ejecución de una VM que venía ejecutándose en otro lado, la nueva máquina física tiene que tener acceso sí o sí a los archivos de la VM. Si no, ¿desde dónde leerá o escribirá cuando el SO guest realice operaciones de E/S? Por lo tanto una de las primeras condiciones que se tienen que cumplir para que se pueda llevar a cabo la migración de una VM de un servidor físico a otro es que los archivos de la VM en cuestión se encuentren en un espacio de almacenamiento compartido, es decir, una SAN. Si bien ahora algunos hipervisores permiten la migración de la ejecución de una VM que se encuentra en discos locales a otro servidor, lo que hacen es primero una migración de los archivos de dicha VM al disco local del servidor de destino y luego realizan la migración de la ejecución.
  - Segundo punto. Cuando una VM migra a un nuevo servidor físico, la misma pretende seguir ejecutando de la misma manera que lo venía haciendo. En realidad, la VM nunca se entera ni que es una VM ni que se migró su ejecución. Por eso es que ella no hará

ningún cambio en su ejecución y deberán estar dadas las condiciones para que ello se pueda cumplir.

Con seguir ejecutando de la misma manera nos referimos a que la VM seguirá ejecutando las mismas instrucciones que venía ejecutando antes de ser migrada. Por lo tanto, deberá haber una compatibilidad entre las CPU del servidor físico de origen y el de destino.

Veamos este punto con un poco más de detalle. Cuando una VM arranca realiza una serie de consultas al procesador físico para saber con qué set de instrucciones cuenta. Esta consulta se realiza al en el momento de booteo de la VM y luego no se realiza más. Por lo tanto, si una VM viene ejecutando con un determinado set de instrucciones y de repente, sin que nadie le avise, cambia su ejecución a un nuevo servidor físico con otra CPU, la VM pretenderá seguir trabajando con el mismo ser de instrucciones con el que venía haciéndolo en el servidor físico anterior.
Por lo tanto, otra condición que se tiene que dar es que las CPUs físicas de los servidores involucrados en la migración sean compatibles.

Ahora bien, ¿qué significa que dos CPUs sean compatibles o no? Dependiendo del hipervisor estas restricciones de compatibilidad serán más laxas en algunos y más estrictos en otro. Generalmente las características que no implican incompatibilidad son la velocidad de las CPUs y los tamaños de las cache internas.

En cambio dentro de las características que definen la compatibilidad o no de dos CPUs se encuentran la marca del procesador (generalmente AMD e Intel no son compatibles para migraciones en caliente), set de instrucciones y soporte o no de virtualización asistida por hardware.

- Tercer punto. Si una VM está utilizando algún dispositivo de hardware físico del servidor donde se encuentra ejecutando, dicha VM generalmente tendrá imposibilitada la opción de ser migrada a otro servidor físico, ya que en el servidor de destino no se tendrá acceso directo al hardware físico de otro servidor. Cuando se utilizan este tipo de configuraciones, esos usos del hardware directo actúan como si fuera un ancla que le imposibilita la migración fuera de ese servidor específico.
  El uso más común de dispositivos físicos para una VM viene dado cuando se le configura a la misma que utilice el CDROM del servidor físico. Cuando esto ocurre, generalmente el administrador de la infraestructura virtual elimina el uso de dichos dispositivos (si es que puede hacerlo con la VM encendida) para posibilitar así la migración.

- ***Migraciones en estado suspendido**:* Si bien cuando una VM suspendida está apagada, el estado real es “suspendido”.
  ¿Por qué merece un análisis la migración de las VMs en este estado si la VM está apagada? ¿No se podría tratar a la misma como una migración en frío? En realidad la respuesta es no. Pensemos que la VM se suspendió y cuando se encienda nuevamente (operación de “resume”) la misma va a pretender continuar con la tarea que estaba realizando en ese momento. La VM no vuelve a bootear y “redescubrir” en qué procesador está ejecutando para saber qué set de instrucciones está disponible, sino que va a continuar trabajando con el mismo que estaba utilizando al momento de suspenderse. Por tanto, para la migración de la VM de un servidor físico a otro se tendrán que cumplir las mismas condiciones que para la migración en caliente.

Aprovisionamiento delgado o Thin Provisioning

El aprovisionamiento delgado es una funcionalidad que no sólo se limita a los ambientes de máquinas virtuales, sino que por ejemplo también se utiliza en sistemas de almacenamiento. No obstante en este sección sólo nos dedicaremos a estudiar cómo se utiliza con máquinas virtuales.
Recordemos nuevamente el concepto de encapsulamiento donde en el hipervisor vemos un archivo que representa al disco rígido virtual dentro de la VM.
Hasta ahora habíamos dicho que si a una VM la configurábamos para que tenga un disco de, por ejemplo 50 GB, en el hipervisor íbamos a ver un archivo de ese tamaño donde se guardaba toda la información que el sistema operativo guest escribía en su "disco rígido".
Bueno, en realidad ese tamaño del archivo dentro del hipervisor depende de la modalidad o tipo de aprovisionamiento que se utilice para ese archivo.
Existen básicamente dos modos o tipos de aprovisionamiento: el *aprovisionamiento gordo, grueso, o thick provisioning*; y el *aprovisionamiento delgado, o thin provisioning.*
En el *thick provisioning* lo que ocurre es lo que explicábamos en el párrafo de arriba. Un administrador de una infraestructura virtual crear una VM y le asigna un disco de determinado tamaño e inmediatamente en el hipervisor se crear un archivo de ese mismo tamaño. No importa si el sistema operativo guest dentro de la VM escriba o no dato alguno. Siquiera importa si en la VM incluso se ha instalado o no un sistema operativo guest, ya que el hipervisor automáticamente crea un archivo del tamaño del disco virtual inmediatamente cuando la VM es creada o el nuevo disco es agregado a una VM ya existente.
En el *aprovisionamiento delgado o thin provisioning* sin embargo ocurre otra cosa. Cuando la VM es creada y asignada a la misma un disco, para seguir con el ejemplo de 50 GB, el hipervisor inmediatamente crea el archivo que representa a ese disco. Sin embargo en un principio dicho archivo tiene tamaño cero (ya que dentro de la VM todavía no se ha escrito ningún dato en dicho disco). A medida que la VM vaya escribiendo datos dentro de ese disco, entonces esos datos se irán escribiendo en el archivo subyacente del hipervisor y por lo tanto el mismo irá creciendo bajo demanda. Esto quiere decir que cuando la VM escriba su primer Giga Byte, el archivo tendrá 1 Gb, y a medida que se vayan escribiendo más y más datos el tamaño del archivo en el hipervisor irá creciendo. Por eso se dice que el tamaño de dicho archivo crece bajo demanda, porque cuando se demanda más espacio el hipervisor le asigna los bloques de disco necesarios a dicho archivo.
Lo que tiene que quedar en claro de ambos tipos de aprovisionamiento es que el sistema operativo guest ***SIEMPRE*** "ve" un disco rígido del tamaño aprovisionado, en nuestro ejemplo un disco de 50 GB. Es decir, si se utiliza thin provisioning para asignarle a una VM un disco de 50 GB, el sistema operativo guest de la VM ve que tiene un disco de 50 GB, no importa si la ocupación de ese disco es de 20 GB y por ende, en el hipervisor el archivo que representa a dicho disco también ocupa 20 GB. El guest siempre ve que tiene un disco de 50 GB.
Podemos ver el concepto en el siguiente gráfico:

En el dibujo podemos ver ambas modalidades. El dibujo intenta mostrar el archivo que representa a un disco virtual visto desde el hipervisor (cada uno de los cilindros), y en color azul muestra el tamaño ocupado.
A la izquierda podemos ver un archivo de tipo thick provisioning, donde todo el cilindro está pintado de azul. Esto quiere decir que no importa lo que la VM haya escrito o no en su disco, el tamaño del archivo en el hipervisor es constante y su tamaño concuerda con el tamaño configurado para el disco en la VM.

Los tres cilindros de la derecha representan a tres archivos de tipo thin provisining. Los tres discos virtuales para la VM tienen el mismo tamaño asignado (representado por la altura de los tres cilindros). Sin embargo, en el primero disco la VM sólo ha ocupado un 25% de su espacio, en el segundo un 50% y en el tercero un 75%. Como se puede observar, en el hipervisor según el caso, el tamaño del archivo (mostrado de color azul) es sólo de un 25% del total, en el segundo un 50% y en el último un 75%.

Por todo lo dicho, se podría decir que el *thin provisioning* es para las VMs una "promesa" de espacio por parte del hipervisor. Es decir, este último les "promete" o les hace ver a las VMs que tienen un espacio asignado que no es real. Es un espacio lógico más que físico. El espacio se vuelve real o físico recién cuando la VM utiliza por completo dicho espacio.

Por lo tanto, si se presta bien atención en lo escrito, el *thin provisioning* es una herramienta que le permite al hipervisor sobre suscribir el espacio físico, o dicho de otra manera, "prometer" más espacio del que realmente hay.

Veámoslo con un ejemplo para que quede claro. Supongamos que el hipervisor tiene un disco rígido donde guarda las VMs de 100 GB disponibles para tal fin.

Si el tipo de aprovisionamiento utilizado es de tipo *thick provisioning*, el hipervisor podrá por ejemplo crear hasta 10 VMs con un disco de 10 GB cada una (tomando como que los demás archivos que representan a la VM no consumen espacio, cosa que en la realidad no es así). Al querer crear la onceava VM se presentará un error en el hipervisor que indicará que ya no hay espacio suficiente para almacenar la nueva VM.

Si ahora en lugar de utilizar el tipo de aprovisionamiento *thick* utilizamos el aprovisionamiento *delgado o thin*, el hipervisor podrá crear si quiere 200 VMs donde a cada una se le asigna un disco de 10 GB, ya que al crearse la VM el tamaño del archivo en el hipervisor que representa al disco es cero. Es más, se podrá arrancar por ejemplo una VM, instalarle su sistema operativo guest y si el mismo recién instalado ocupa 4 GB, el tamaño del archivo en el hipervisor será de 4 GB.

Por lo tanto, en este caso, el hipervisor "prometió" a las VMs un espacio a cada una de 10 GB, y por ende "prometió" en total 2 TB!!!!.

Mientras que las VMs en su conjunto no escriban o utilicen más de 100 GB no habrá problemas, pero los mismos se suscitarán tan pronto cualquiera de las VMs quiera escribir un nuevo bloque que demande más allá de los 100 GB físicos en el hipervisor.

En consecuencia, la funcionalidad de *thin provisioning* es una herramienta extremadamente útil en los sistemas virtuales, porque nos permiten ir consumiendo el espacio físico a medida que realmente lo necesitamos, pero mal administrado o mal utilizado se puede convertir en un verdadero problema. Por eso en cuanto comenzamos a utilizar esta funcionalidad debemos prestar constante atención al espacio físico utilizado y disponible. Es por esto que las consolas de administración de este tipo de sistemas poseen la función de crear alarmas que permiten monitorear tanto el espacio libre disponible como el espacio aprovisionado. El espacio aprovisionado es el espacio que se "prometió". Por lo tanto, cuando el espacio aprovisionado pasa del 100% del espacio físico, comienza el riesgo de quedarnos sin espacio. Cuanto más allá del 100% se encuentre este espacio, más riesgo habrá de quedarnos sin espacio.