2. Introducción a la virtualización

Dado el creciente nivel de recursos de las máquinas que forman parte de los sistemas informáticos, se produce el hecho de que en determinados momentos estas máquinas no utilizan buena parte de su potencialidad. Para aprovechar esta situación aparece de nuevo un término ya antiguo en el mundo de la informática como es el de la *virtualización* .

De esta forma, por medio de la virtualización, se quieren aprovechar las máquinas para realizar la simulación de hardware, o sobre todo últimamente la instalación de varios sistemas operativos virtuales en una misma máquina.

Esta virtualización se hace de diferentes formas según qué técnica se utiliza, y últimamente han sido muchas las máquinas virtuales que han ido apareciendo y desarrollándose con nuevas versiones.

2.1. Introducción a la virtualización

La virtualización en el mundo de la informática arrancó ya en los años sesenta y tuvo diferentes aplicaciones que iban desde sistemas computacionales complejos hasta el trabajo de capacidades o componentes individuales.

En muchos momentos las capacidades de las computadoras no se utilizan al máximo, por lo que dejan un margen para el aprovechamiento de estos recursos. Pero la cuestión es cómo utilizar estos recursos en determinados ámbitos si en la máquina ya tenemos instalados el sistema operativo y una serie de softwares. Así de entrada parecería que sólo podemos optimizar los recursos de la máquina a base de instalar y ejecutar más software. La virtualización abre la puerta a aprovechar estos recursos de máquina en todo tipo de situaciones, incluso a tener varios sistemas operativos dentro de la misma y trabajando a la vez. La manera en que esto se puede materializar es lo que se conoce como *virtualización* .

> La **virtualización consiste** en la abstracción de los recursos de la máquina para poder utilizar los que sobran y crear máquinas virtuales que utilizan este hardware (hardware) *como si* estuviera perfectamente libre.

Entonces, a partir de la virtualización se puede compartir una máquina física para ejecutar varias máquinas virtuales, por lo que éstas comparten

trabajo. Además, la virtualización permite utilizar estas máquinas virtuales con independencia de su hardware. Esto se consigue ocultando los detalles técnicos mediante el método de la encapsulación .

Una máquina virtual permite tener varios ordenadores virtuales ejecutándose sobre el mismo ordenador físico. Esto se logra mediante *software* o software, una capa de software que crea una capa de abstracción entre el hardware de la máquina física o *host* y el sistema de la máquina virtual. Éste es el método para crear una versión virtual de un dispositivo o recurso tanto si es todo un servidor como simplemente un disco duro, una red o un sistema operativo.

Esta virtualización se produce mediante esta capa de software conocida como **máquina virtual** . Ésta se conoce como *software* ***anfitrión,*** en inglés *host* que simula un entorno de hardware capaz de alojar un software *guest* u huésped . Este software *guest* , que puede llegar a ser un sistema operativo completo, se ejecuta como si estuviera aislado en una plataforma de hardware autónoma.

> En informática, una **máquina virtual es** un software que emula a una computadora y puede ejecutar programas como si fuera una computadora real.

Una característica de las máquinas virtuales es que los procesos que ejecutan están limitados por recursos y abstracciones proporcionados por ellas.

Una de las utilizaciones más extendidas de las máquinas virtuales consiste en ejecutar sistemas operativos para probarlos. De esta forma podrá ejecutar un sistema operativo que desee probar, por ejemplo, un Linux, desde el sistema operativo habitual, por ejemplo un Mac OS X , sin necesidad de instalarlo directamente en su máquina y sin miedo a que se desconfigure el sistema operativo primario.

2.2. Arquitecturas. Tipo de máquinas virtuales

El apantallamiento del hardware de un ordenador como si fuera una máquina totalmente autónoma a partir del encapsulamiento proporcionado por la capa de software llamado **máquina virtual** puede producirse básicamente de dos maneras diferentes. Esta clasificación se produce según la funcionalidad de la máquina virtual y según su grado de equivalencia en una verdadera máquina autónoma.

Existen entonces dos tipos de máquinas virtuales: las **máquinas virtuales de proceso y** las máquinas **virtuales** de sistema .

virtuales de sistema permiten albergar, por ejemplo, todo un sistema operativo huésped gracias al software anfitrión de la máquina virtual.

2.2.1. Máquinas virtuales de proceso

La máquina virtual de proceso, también conocida como **máquina virtual de aplicación,** se ejecuta como un proceso normal dentro de un sistema operativo y soporta un solo proceso. La máquina se inicia automáticamente cuando se lanza el proceso que se desea ejecutar y se detiene cuando este proceso ha finalizado. El objetivo de esta máquina es proporcionar un entorno de ejecución **independiente de la plataforma de hardware y del sistema operativo, de** modo que esconda los detalles de la plataforma subyacente y permita que un programa se ejecute siempre de la misma manera sobre cualquiera tipos de plataforma.

Uno de los ejemplos más conocidos de este tipo de máquina virtual es la **máquina virtual de Java** . Otro ejemplo sería la del entorno **.Net de Microsoft,** que se llama *common* language runtime . Otros ejemplos de máquinas virtuales de proceso son los siguientes: Macromedia Flash Player, **SWF** , *Perl virtual machine,* Perl **y** un largo etcétera.

Muchos de estos ejemplos corresponden a máquinas virtuales destinadas a poder ejecutar varios procesos de distintos lenguajes de programación. La finalidad siempre es apantallar el hardware del proceso a ejecutar; de esta forma, se puede programar el proceso sin tener en cuenta en qué tipo de máquina se ejecutará. En todos ellos, la estructura de la máquina virtual es similar.

Aparte de las máquinas virtuales destinadas a ejecutar lenguajes de programación en diferentes entornos de hardware, también existen las máquinas virtuales de proceso que se dedican a simular hardware , de forma que permiten **que** determinadas aplicaciones pensadas para otras arquitecturas de procesador se puedan ejecutar sobre un hardware que en teoría no soportan.

JVM. Máquina virtual de Java

Una de las máquinas virtuales de proceso más utilizadas es la máquina virtual de Java debido a la cantidad de aplicaciones que utilizan ese lenguaje de programación. Java es un lenguaje de programación de la empresa **Sun Microsystems** que originariamente se llamaba Oak, que fue pensado y programado para utilizar en pequeños dispositivos electrodomésticos, pero sin éxito comercial. Sus creadores se encontraron entonces con un lenguaje con una tecnología fuerte, eficiente, orientada a objetos, independiente de la arquitectura pero que no servía para nada. Pero al poco tiempo en Sun se dieron cuenta de que estas características eran perfectas para las aplicaciones para Internet; de esta forma, con unos pequeños retoques, Oak se convirtió en Java. En el momento en que el navegador Netscape incorporó Java, el interés por este lenguaje se disparó

Entonces esta máquina virtual de Java introduce un nuevo nivel de abstracción y proyección entre la máquina y el software que se ejecuta. Esto todavía tiene un interés más especial cuando el código que está ejecutando puede provenir de alguien que lo ha programado en la otra punta del mundo.

La diferencia principal entre un lenguaje como Java que se ejecuta en la JVM y un lenguaje como *JavaScript es* la eficiencia.

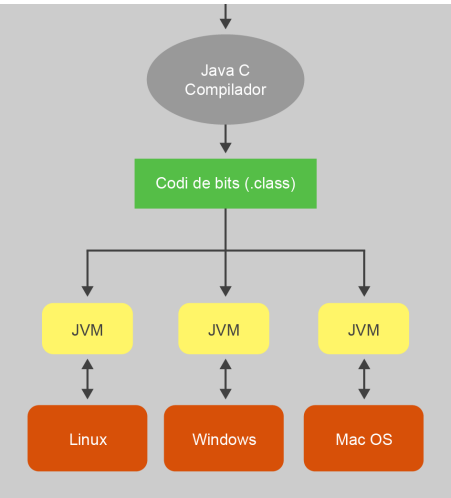
Para ejecutar un programa escrito en un lenguaje completamente interpretado, el intérprete debe realizar el análisis léxico y sintáctico en el momento de ejecutar el programa, lo que provoca una sobrecarga considerable en la ejecución de este programa.

En cambio, los lenguajes basados en una máquina virtual suelen ser más rápidos, ya que dividen la tarea de ejecutar un programa en dos partes: en la primera se realiza el análisis léxico y sintáctico del programa fuente, para generar el programa en instrucciones del procesador virtual (código intermedio), y en el segundo paso se da una iteración sobre el código intermedio para obtener la ejecución final del programa.

Entre las propiedades del lenguaje Java está la **portabilidad (es** posible ejecutar el mismo archivo de clase .class *en una* amplia variedad de arquitecturas de hardware y software), el **dinamismo (ya que** las clases son cargadas en el mismo momento de utilizarlas), la eficiencia **y la seguridad** .

La clave de muchas de estas características está en la JVM. Esta JVM es el núcleo del lenguaje de programación, y ahí se encuentra el motor que en realidad ejecuta el programa Java. Siempre que se ejecuta un programa de Java, las instrucciones de este programa no son ejecutadas por el hardware que hay debajo, sino que se pasan a un elemento de software intermedio, que es el encargado de que las instrucciones sean ejecutadas por el hardware . Es decir, el código Java no se ejecuta directamente sobre un procesador físico, sino sobre un **procesador virtual Java** . En la figura 2 .1 puede ver la capa de software que implementa en la máquina virtual de Java:

Figura 2.1. Estructura de la JVM en una máquina



Esta capa de software oculta los detalles inherentes a la plataforma, a las aplicaciones Java que se ejecutan sobre ella. Dado que la plataforma Java se diseñó pensando que se implementaría sobre una amplia gama de sistemas operativos y de procesadores, se incluyeron dos capas de software para aumentar su portabilidad. La primera, dependiente de la plataforma, se llama *adaptador,* mientras que la segunda, que es independiente de la plataforma, se llama *interfaz de portabilidad* . De esta forma, la única parte que debe escribirse para una plataforma nueva es el adaptador. El sistema operativo proporciona servicios de utilización de ventanas, red, sistema de archivos, etc.

Sun, la empresa de la JVM, utiliza el término *máquina virtual de Java* para referirse a la especificación abstracta de una máquina de software para ejecutar programas Java. La especificación de esta máquina virtual define elementos como el formato de los archivos de clase de Java (. *class*), así como la semántica de cada una de las instrucciones que componen el conjunto de instrucciones de la máquina virtual. Las implantaciones de esta especificación se conocen como "sistemas en tiempo de ejecución Java". Un sistema de tiempo de ejecución incluye típicamente la arquitectura siguiente que puede ver en la tabla 2 .1 .

Tabla 2.1. Sistema que utiliza Java JIT (just in time)

Elemento	Función
Motor de ejecución	El procesador virtual que se encarga de ejecutar el código generado por algún compilador de Java.
Gestionador de memoria	Encargado de obtener memoria para las nuevas instancias de objetos y realizar tareas de recogida de basura (desbloquea memoria de objetos instanciados que ya no se utilizan).
Gestionador de errores y excepciones	Encargado de generar, lanzar y atrapar excepciones.
Soporte de métodos nativos	Encargado de llamar métodos de C++ o funciones de C desde métodos Java o viceversa.
Interfaz multihilos	Encargada de proporcionar el soporte para hilos y monitores.
Cargador de clases	Su función es cargar dinámicamente las clases Java a partir de los archivos de clase (*.class*).

Instalación de la máquina virtual de Java, JVM

Aunque casi con toda seguridad en la instalación del sistema operativo de la máquina ya se incluirá la instalación de la máquina virtual de Java, en caso de tener que hacer usted una instalación de esta JVM, habrá que ir a la página oficial de Sun. Dado que la máxima utilidad de la JVM es que hace transparente el hardware para la ejecución de las aplicaciones de Java, se debe poder instalar la JVM para cualquier hardware y para cualquier sistema operativo, tanto del entorno privativo como de código libre. Por tanto, si busca la página oficial de Sun y vaya al apartado de descargas de la JVM, verá que debe elegir su hardware y su sistema operativo. Es necesario que mire que, junto a la descarga del archivo correspondiente de la máquina virtual, también haya un enlace con la ayuda para instalarla. Si se descarga la JVM correspondiente a su máquina y sistema operativo y siga correctamente las instrucciones que encontrará, verá cómo no hay ningún problema en instalarla.

Desventajas de las máquinas virtuales de proceso

Una de las razones por las que las máquinas virtuales no son la solución definitiva de la computación es que añaden una gran complejidad al sistema en tiempo de ejecución. Por ejemplo, la JVM espera que la computadora sobre la que corre soporte el estándar de **IEEE** para los números de punto flotante de 32 bits y 64 bits. La mayoría lo hacen, pero hay algunas que no, lo que implica un trabajo extra.

Pero la principal desventaja de los lenguajes basados en máquinas virtuales es que efectivamente son más lentos que los lenguajes completamente compilados, debido a la sobrecarga que genera tener una capa de software intermedia entre la aplicación y el hardware de la computadora.

2.2.2. Máquinas virtuales de sistema

Las máquinas virtuales de sistema, o también llamadas **máquinas virtuales de hardware,** permiten a la máquina física subyacente *multiplexarse* en varias máquinas virtuales, cada una ejecutando el propio sistema operativo.

> La capa de software que permite la virtualización se llama **monitor de máquina virtual,** MMV **o** también **hipervisor** .

Esta capa de software MMV gestiona los cuatro recursos principales de una computadora:

- CPU
- Memoria

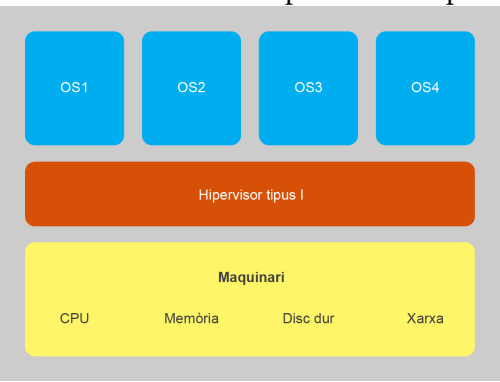
Con esta técnica de la emulación del hardware se consigue lo que se conoce como virtualización completa. Esto permite que cada máquina virtual pueda ejecutar cualquier sistema operativo soportado por el hardware de la computadora (en las técnicas más modernas ya no es necesario que el sistema operativo sea compatible con el hardware de la computadora); de esta forma los usuarios de la máquina pueden ejecutar simultáneamente dos o más sistemas operativos distintos en máquinas virtuales. Una máquina virtual de sistema permite tener varios ordenadores virtuales ejecutándose sobre el mismo ordenador físico.

Últimamente, estas máquinas virtuales han ganado mucha importancia, ya que las máquinas actuales tienen muchos recursos de hardware y permiten la implementación de varios sistemas operativos dentro de una misma máquina, pudiendo simular muchas situaciones reales sin necesidad de implementarlas realmente . Por ejemplo, en un PC de mesa o incluso en un portátil, podemos instalar un conjunto de sistemas operativos, uno de servidor y el resto como máquinas cliente, realizar la simulación de una red con un servidor de archivos y probar la gestión de los usuarios y sus privilegios sin necesidad de implementar realmente la red.

Un monitor de máquina virtual (MMV) admite varios tipos de arquitectura:

1. Clásica . En esta arquitectura, el monitor de la máquina virtual se ejecuta directamente sobre el hardware. Entonces los sistemas operativos huéspedes corren directamente sobre la capa de software que hay sobre el hardware, el MMV o el hipervisor. Este sistema se conoce como el *hipervisor de tipo I* , **hipervisor** I. En la figura 2 .2 puede ver la estructura de esta arquitectura:

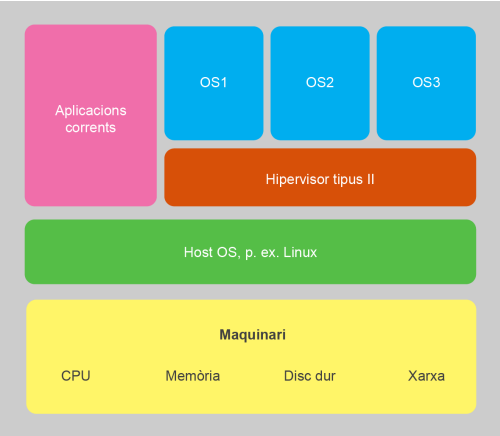
Figura 2.2. Ésta es la estructura de funcionamiento de un hipervisor del tipo I.



Esta arquitectura no carga excesivamente el sistema, permite una buena optimización de los recursos de la máquina, pero requiere que los sistemas operativos que instale en la máquina virtual sean compatibles con el software de la máquina. Como ejemplo de máquina virtual del tipo hipervisor I puede encontrar la versión **ESX** de **VmWare** .

hacerlo directamente sobre el hardware, y los sistemas operativos virtuales se ejecutan sobre el MMV . De esto viene el nombre de *indirecta* . También se conoce con el nombre de **hipervisor II** . Actualmente, esta arquitectura es una de las más utilizadas, y podrá encontrar muchas máquinas virtuales que la utilizan. En este sistema, se está ejecutando el sistema operativo anfitrión y sus aplicaciones y además se ejecuta la máquina virtual como una aplicación más, y los diferentes sistemas operativos son instancias de la máquina virtual, como puede ver en la figura 2 . 3 :

Figura 2.3. Ésta es la estructura de funcionamiento del hipervisor de tipo II.



En esta arquitectura, al igual que ocurre con la de hipervisor I, sólo se pueden instalar sistemas operativos virtuales que sean compatibles con la arquitectura del procesador y otros recursos de la máquina.

De ejemplos de máquinas virtuales que utilizan este sistema encontrará muchos, las más comunes son las siguientes: **VmWare** , **VirtualBox, Microsoft** Virtual PC, KVM y **un** largo etcétera. Algunas de estas máquinas virtuales son de pago, como VmWare, Microsoft Virtual PC y Oracle Virtual Box, y otras son de código libre, como la VirtualBoxOSE o la KVM. VmWare y Oracle ofrecen versiones gratuitas, aunque en el caso de VmWare, con menos prestaciones. Oracle, la propietaria actual de VirtualBox ofrece versiones de código libre como la VirtualBoxOSE o bien un uso personal restringido para la versión Oracle VirtualBox. Algunas de estas máquinas virtuales de código libre pueden correr en sistemas operativos privativos, como Windows, o en los de código libre como Linux.

Entre los inconvenientes de esta arquitectura, encontraríamos el hecho de que el hipervisor de tipo II es menos eficiente que el de tipo I, pero sobre todo que no se puede instalar un sistema operativo que no sea compatible con la arquitectura del procesador de la máquina.

3. Paravirtualización . El hecho de que en los sistemas de hipervisor I y II sólo se pueden instalar sistemas operativos en la máquina virtual que sean compatibles con la arquitectura del procesador (la mayoría de veces x86), y además comporta una disminución en el rendimiento en el

Ante este hecho aparece una técnica que se conoce como **paravirtualización, que** representa una solución intermedia en el camino de poder virtualizar cualquier sistema en cualquier hardware y hacerlo prácticamente sin penalización en el rendimiento del sistema operativo virtualizado, o con penalizaciones muy bajas de cerca de 2%-8%.

Para conseguir este funcionamiento, la máquina virtual se instala directamente sobre el hardware, haciendo lo que se conoce como *una instalación de bajo nivel* , muy cerca del mismo hardware. Esta técnica requiere entonces modificar los sistemas operativos huéspedes para ser instalados en la máquina virtual. Éste es el inconveniente principal de la paravirtualización, que necesita sistemas operativos modificados, aunque ofrecen las mismas prestaciones, lo que pasa desapercibido para el usuario, y reduce el abanico de posibilidades de los sistemas huéspedes.

Un ejemplo importante de la paravirtualización es **Xen** . Xen es una máquina virtual de código libre que proporciona virtualizaciones de alto rendimiento sobre máquinas de arquitectura x86, pero existe una limitación de sistemas operativos modificados aptos para ser instalados en la máquina virtual Xen. Actualmente existen algunos Linux, NetBSD y una versión inicial de Windows XP, pero que no se puede lanzar al público porque necesita licencia, como sistemas operativos disponibles para la máquina Xen.

Sin embargo, esta máquina virtual tiene sus utilidades y aplicaciones (IBM las utiliza en sus servidores y ordenadores centrales), por lo que las compañías principales fabricantes de procesadores, Intel y AMD , ya se han apresurado a **sacar** versiones **de** sus procesadores compatibles con la máquina virtual Xen por lo que no será necesario adaptar los sistemas operativos a la máquina virtual, lo que permitirá instalar cualquier sistema operativo con un alto rendimiento.

4. ***Full virtualization*** . El hecho de que los hipervisores I y II requieran que el sistema operativo a virtualizar sea compatible con el hardware de la computadora, y que el rendimiento de este sistema virtual se vea penalizado, salvo que se utilice la paravirtualización con la correspondiente limitación de los sistemas a virtualizar, hace que la virtualización tenga trabas para alcanzar la situación en la que se pueda virtualizar cualquier sistema operativo en cualquier arquitectura de hardware. Esto es lo que se conoce como **hoja virtualización o** *virtualización* completa . La cuestión es conseguir máquinas virtuales que apantallen el hardware y además lo emulen para poder instalar todo tipo de sistemas operativos en cualquier hardware, sea cual sea la arquitectura del procesador.

En esta arquitectura, la máquina virtual se instala directamente sobre el hardware, los sistemas operativos virtuales se ejecutan sobre la máquina

Actualmente los fabricantes de procesadores están realizando modelos de procesadores con la máxima compatibilidad con las máquinas virtuales (AMD-V, Intel VT-x) para facilitar la virtualización total o completa.

Como ejemplos de máquinas virtuales que pueden hacer virtualización completa se encuentran **Mac-on-Linux** , **Parallels Workstation,** algunas versiones de VmWare, como la **VmWare GSX Server, VirtualBox** y algunas otras. En el caso de Vmware, utiliza una técnica conocida como *traslación binaria* para modificar automáticamente las instrucciones de software x86 que simula una virtualización completa.

Entre las aplicaciones de las máquinas virtuales de sistema sobresalen las siguientes:

- **Arquitectura de instrucciones (ISA).** La máquina virtual puede proporcionar una arquitectura de instrucciones ISA que sea ligeramente distinta a la verdadera máquina; es decir, se puede **simular hardware** .
- **Coexistencia de sistemas operativos.** Varios sistemas operativos pueden coexistir sobre la misma máquina, sólidamente aislados unos de otros, por ejemplo, para probar un sistema operativo nuevo sin necesidad de instalarlo directamente. De esta forma el MMV ofrece un **entorno de ejecución completo,** es decir, ofrece la posibilidad de instalar un nuevo sistema operativo, con sus aplicaciones, sus usuarios, su gestión del disco, de la red , etc.
- **Consolidación de servidores.** Varias máquinas virtuales, cada una con el propio sistema operativo llamado *sistema operativo **huésped*** o *guest* , se pueden utilizar para **consolidar servidores** . Esto permite que servicios que normalmente deberían ejecutarse en computadoras diferentes para evitar interferencias, se pueden ejecutar en la misma máquina de forma completamente aislada y compartiendo los recursos de una única computadora. En muchas ocasiones, la consolidación de servidores contribuye a reducir el coste total de las instalaciones necesarias para mantener los servicios, ya que permiten ahorrar en hardware.
- **Optimización de recursos.** Hoy en día, la virtualización es una opción muy buena, ya que en la mayoría de los casos las máquinas actuales están siendo infrautilizadas, ya que disponen de una gran capacidad de disco duro, de memoria, etc. En muchos casos llegan a utilizar sólo en un 30% o 60% de la capacidad de sus recursos. Virtualizar la necesidad de nuevas máquinas en una ya existente permite un ahorro considerable de los gastos asociados como energía, mantenimiento, espacio físico, etc.

VMware

Dentro de las máquinas virtuales de sistema existe una máquina virtual, la VMware, que se encuentra dentro de las máquinas virtuales de pago, aunque también dispone de algunas distribuciones libres. Esta máquina virtual es un software de virtualización disponible para arquitectura x86 de procesadores. La VMware puede funcionar en Windows, Linux y Mac OS X que utilice procesadores Intel. Dentro de las versiones gratuitas se encuentran las VMware Player, VMware Server y VMware ESXi, y la versión comercial VMware ESX Server. En ese caso, la máquina virtual libre tanto para Windows como para Linux es la VMware Player.

procesador Intel), tanto de 32 bits como de 64 bits. Sin embargo, si tiene que instalar un sistema operativo virtual de 64 bits, primero tendrá que asegurarse de que su procesador es de 64 bits, y sobre todo que es del tipo AMD64-VT o Intel64-VT, es decir, un procesador con soporte específico para la virtualización. Si tiene un procesador de 64 bits pero no con la especificación VT, no podrá virtualizar sistemas operativos de 64 bits con esta máquina virtual, y deberá instalar la versión correspondiente a 32 bits. En caso de que no sepa si su procesador cumple este requisito o no, en la página de MVware tiene la opción de descargar una pequeña aplicación en Linux, una imagen ISO que debe grabar en un CD, y arrancar su ordenador con el CD, y entonces automáticamente le dará la información correspondiente de su procesador y sabrá si soporta o no la virtualización de sistemas de 64 bits por medio de VMware Player.

VirtualBox

Un ejemplo de las máquinas virtuales de sistema que utiliza la arquitectura de hipervisor del tipo II es VirtualBox. Actualmente esta máquina virtual pertenece a la empresa **Oracle,** pero sigue distribuyendo versiones de código libre VirtualBoxOSE e incluso distribuye una copia de la versión comercial para uso personal. VirtualBox es un software de máquina virtual para arquitecturas de procesador x86 y AMD64/Intel64 que permite instalar sistemas operativos huéspedes en la arquitectura de hipervisor del tipo II. Dentro de los sistemas operativos anfitrión en los que podemos instalar la máquina virtual VirtualBox, se encuentran los siguientes: GNU/Linux, Mac OS X, OS /2 Warp, Microsoft Windows, y Solaris/OpenSolaris. Dentro de estos sistemas podemos virtualizar FreeBSD, GNU/Linux, OpenBSD, OS /2 Warp, Windows, Solaris, MS-DOS y muchos otros.

2.2.3. Virtualización en el sistema operativo

Aparte de la virtualización por procesos, como es el caso de la JVM (máquina virtual de Java), que permite la ejecución de programas de Java en cualquier máquina, y de las máquinas virtuales de sistema (que virtualizan todo el hardware) que permiten instalaciones de sistemas operativos virtuales, existe un tercer nivel de virtualización. Este nivel de virtualización no es una capa de software que apantalla el hardware, sino que se trata de virtualizar el propio sistema operativo, el mismo *kernel* del sistema operativo.

En este sistema, el *kernel de* un sistema operativo permite crear diversas instancias de espacios de usuario, conocidas como *contenedores* totalmente aisladas y que permiten instalar un sistema operativo en cada una de ellas. Estos compartimentos también se conocen como *entornos virtuales* . Esto tiene aplicaciones como, por ejemplo, en las zonas de **Solaris (** Solaris *zonas)* en las que el sistema actúa como si realmente hubiera varios servidores ejecutándose en varias máquinas diferentes.

Un ejemplo de máquina virtual que trabaja sobre el *kernel* del sistema operativo es la **KVM (** kernel *-based virtual machine)* . Esta máquina

módulo del núcleo (kvm.ko) y herramientas en el espacio de usuario. Es totalmente de software libre. El componente KVM para el kernel está incluido en las distribuciones de Linux desde la versión 2.6.20 del kernel.

KVM permite ejecutar máquinas virtuales a partir de imágenes de discos de sistemas operativos sin modificar; entonces, cada máquina virtual tiene su propio **hardware virtualizado:** tarjeta de red, disco duro, tarjeta gráfica, etc.

Como requisitos, la KVM necesita un procesador x86 con soporte *virtualization thecnology (AMD* -V, Intel VT-x) y permite ejecutar sistemas operativos como, por ejemplo, GNU/Linux (32 bits y 64 bits) y Windows (32 bits).