La Arquitectura de las Máquinas Virtuales.

La virtualización se ha convertido en una importante herramienta en el diseño de sistemas de computación, las máquinas virtuales (VMs) son usadas en varias subdiciplinas, desde sistemas operativos a lenguajes de programación y hasta en la arquitectura de procesadores. Liberando así a desarrolladores y a usuarios de la interfaz tradicional y limitaciones de recursos.
Una máquina virtual puede soportar procesos individuales o un sistema completo según el nivel de abstracción de la virtualización.
Las VMs surgieron del producto de diversos grupos con objetivos diferentes. Por consiguiente, es útil considerar la variedad de arquitecturas de VMs, y describirlas de un modo unificado, poniendo en consideración tanto la noción de virtualización como los tipos de VMs.

ABSTRACCIÓN y VIRTUALIZACIÓN.

A pesar de su increíble complejidad, los sistemas de computación existen y siguen evolucionando porque son diseñados con interfaces bien definidos que separan los distintos niveles de abstracción. La utilización de interfaces bien definidas facilita el desarrollo independiente de subsistemas tanto por parte de los grupos de trabajos de hardware como de software. La simplificación dada por la abstracción que esconde los detalles de realización, reduce la complejidad del proceso de diseño.
Un ejemplo de abstracción es el almacenaje de disco. El sistema operativo abstrae el direccionamiento del disco duro, que consiste en sectores y pistas, de modo que el disco aparece para el programa como un conjunto de archivos con tamaño variable. Los programas de aplicación pueden crear, escribir, y leer archivos sin saber la construcción del disco duro y la organización física.
Una computadora con una arquitectura de set de instrucciones (ISA) claramente ejemplifica las ventajas de las interfaces bien definidas. Las interfaces bien definidas permiten el desarrollo de subsistemas de computación que se relacionan no sólo en diferentes organizaciones sino también en tiempos diferentes, a veces años diferentes. Por ejemplo, Intel y AMD desarrollan microprocesadores que implementan el set de instrucciones Intel IA-32 (x86), mientras los desarrolladores de Microsoft escriben el software que es compilado para ese mismo set de instrucciones.
Como ambos grupos satisfacen la especificación de ISA, puede esperarse que el software se ejecute correctamente en cualquier computadora personal construida con un microprocesador IA-32.
Lamentablemente, las interfaces también tienen sus limitaciones. Los subsistemas y los componentes diseñados para las especificaciones de una interfaz no trabajarán con aquellos diseñados para otras.
Esta carencia de la interoperabilidad es limitante, sobre todo en un mundo de computadoras conectadas a una red, donde es ventajoso mover el software tan libremente como los datos.
La virtualización de un sistema o componentes, como un procesador, memoria, o un dispositivo de entrada salida, mapea los recursos visibles dentro de la interfaz en los recursos del sistema real posiblemente diferente. Por consiguiente, el verdadero sistema aparece como un sistema virtual diferente o como sistemas virtuales múltiples.
A diferencia de la abstracción, la virtualización no necesariamente pretende simplificar o esconder detalles. Por ejemplo, la virtualización transforma un disco grande en dos discos virtuales más pequeños, cada uno de los cuales parece tener sus propias pistas y sectores.
El software de virtualización usa la abstracción de archivo como un paso intermedio para proporcionar una correlación entre los discos virtuales y verdaderos. La escritura en un disco virtual es convertida en una escritura de un archivo (y por lo tanto en una verdadera escritura en disco). Notando que el nivel de detalle proporcionado en la interfaz del disco virtual, la dirección del sector/pista, no es diferente de un verdadero disco; entonces no ocurre una abstracción.

LAS MÁQUINAS VIRTUALES.

El concepto de virtualización puede ser aplicado no sólo a subsistemas como discos, sino también a una máquina entera. Para poner en práctica una máquina virtual, los desarrolladores añaden una capa de software a una verdadera máquina para soportar la arquitectura deseada. Una VM puede emular la compatibilidad de una máquina real y los recursos de hardware.

Arquitectura de las Interfaces.
Una consideración principal en la construcción de una VM es la fidelidad con la cual esta pone en práctica la arquitectura de las interfaces. La arquitectura, aplicada a sistemas de computadoras, se refiere a una especificación formal de una interfaz en el sistema, incluso en el comportamiento lógico de recursos manejados vía esa interfaz. Los niveles de abstracción corresponden a capas de implementación, tanto en hardware como en software, cada uno asociado con su propia interfaz o arquitectura.
Las tres más importantes interfaces y capas de implementación de un sistema típico de computación para la construcción de una VM son: la arquitectura de set de instrucciones, la interfaz de aplicación binaria, y la interfaz de programas de aplicación.
La arquitectura de set de instrucción: El ISA marca la división entre el hardware y el software. Incluye aquellos aspectos visibles por un programa de aplicación o puede incluir aquellos aspectos visibles sólo por el sistema operativo.
Interfaz binaria de aplicación: El ABI da el acceso de los programas a los recursos del hardware y a los servicios disponibles en un sistema. No incluye instrucciones de sistema, todos los programas se relacionan con los recursos de hardware indirectamente invocando los servicios del sistema operativo vía la interfaz de llamadas al sistema. Las llamadas al sistema proporcionan un camino al sistema operativo para realizar operaciones de parte de un programa de usuario después de validar su autenticidad y seguridad.
Interfaz de programas de Aplicación: El API da el acceso de los programas a los recursos de hardware y servicios disponibles en un sistema. El acceso se da a través de lenguajes de alto nivel (HLL). Cualquier llamada de sistema es por lo general realizada a través de bibliotecas. La utilización de un API permite al software de aplicación ser transportada fácilmente, a través de la recompilación, a otros sistemas que soporten el mismo API.

Procesos y sistemas de las VMs.
Para entender que es una máquina virtual, es necesario primero considerar el significado de "máquina" tanto de la perspectiva del sistema como del proceso. De la perspectiva de un proceso ejecutando un programa de usuario, la máquina consiste en un espacio lógico de direcciones de memoria asignado al proceso junto con instrucciones de nivel del usuario y registros que permiten la ejecución del código que pertenece al proceso.
La entrada-salida de la máquina es visible sólo por el sistema operativo, y el único modo que el proceso puede relacionarse con el sistema de entrada-salida es por las llamadas al sistema operativo. Así el ABI define la máquina vista como un proceso. Del mismo modo, el API especifica las características de la máquina vista como programas de aplicaciones.
Un sistema es un completo entorno de ejecución que puede soportar numerosos procesos simultáneamente. Estos procesos comparten un sistema de archivo y otros recursos de entrada-salida. El sistema asigna memoria real y recursos de entrada-salida a los procesos, y permite que los procesos se relacionen con sus recursos. De la perspectiva del sistema, las características del hardware definen la máquina; y es el ISA que proporciona el interfaz entre el sistema y máquina.
Un proceso VM es una plataforma virtual que ejecuta un proceso individual. Este tipo de VM existe únicamente para soportar el proceso; es creado cuando el proceso es creado y se termina cuando el proceso se termina.
En contraste, un sistema VM proporciona un entorno de desarrollo completo, que soporta un sistema operativo junto con sus muchos procesos de usuario. Este provee el sistema operativo de usuario con acceso a recursos de hardware virtuales, incluyendo gestión de redes, entrada-salida, y una interfaz de usuario gráfica junto con un procesador y memoria.
El proceso o el sistema que corre en un VM es el guest (software o sistema operativo de la VM), mientras la plataforma que soporta la VM es el host (sistema operativo de la máquina real). El software de virtualización que implementa un proceso VM a menudo es llamado run-time abreviación de "run-time software". El software de virtualización en un sistema VM es típicamente referido como el monitor de la máquina virtual (VMM).
En un proceso VM, el software de virtualización está en el nivel ABI o en el API, encima de la combinación Sistema Operativo/Hardware.
En el tiempo de ejecución se emula tanto instrucciones del nivel de usuario como del sistema operativo o las llamadas al sistema. En un sistema VM, el software de virtualización está entre el host, hardware de la máquina y el usuario, el software.
El VMM emula el hardware ISA de modo que el software usuario pueda ejecutar potencialmente ISA diferentes del que está implementado en el host. Sin embargo, en muchas aplicaciones de sistema de VM,

el VMM no realiza la emulación de instrucciones; mejor dicho, su papel principal es proporcionar recursos del hardware virtualizado.

LOS PROCESOS DE LAS MÁQUINAS VIRTUALES.

Los procesos de VMs proporcionan una ABI virtual o un entorno API para aplicaciones de usuario. En sus varias implementaciones, los procesos de VMs ofrecen replicación, emulación, y optimización.

Sistemas multiprogramados.

La mayoría de los sistemas operativos pueden soportar simultáneamente múltiples procesos de usuario por la multiprogramación, que da a cada proceso la ilusión de tener una máquina completa. Cada proceso tiene su propio espacio de direcciones, registros, y estructura de archivo. El sistema operativo comparte los tiempos del hardware y maneja los recursos para hacer esto posible. En efecto, el sistema operativo proporciona una réplica del estado del proceso de la VM para cada una de las aplicaciones que se ejecutan simultáneamente.

Los emuladores y los traductores dinámicos.

Un problema más complicado para el nivel del proceso de las VMs es el de soportar programas compilados con un set de instrucciones diferentes del que se ejecuta en el host.

El modo más real de realizar emulación es por la interpretación. Un programa intérprete descifra, y emula la ejecución de instrucciones de guest individuales. Este puede ser un proceso relativamente lento, requiriendo decenas de instrucciones para cada instrucción del programa fuente.

La mejor interpretación puede ser obtenida por la traducción dinámica, que convierte bloques de instrucciones de guest en bloques de instrucciones del host. La ejecución repetida de instrucciones amortiza el relativamente alto tiempo de la traducción.

Optimizador binario de Igual ISA.

Para reducir pérdidas de interpretación, los traductores dinámicos binarios a veces realizan optimizaciones de código durante la traducción. Esta capacidad conduce naturalmente a VMs en donde los set de instrucciones que utilizan el guest y el host son los mismos, con el único objetivo de la optimización.

El optimizador dinámico binario de igual ISA utiliza la información del perfil coleccionado durante la interpretación o fase de traducción para optimizar el programa binario en ejecución.

Lenguaje de alto nivel de las VMs.

Para el proceso de las VMs, la portabilidad de plataforma es un objetivo clave. Sin embargo, la emulación de una arquitectura convencional en otra, es aplicable sólo en caso básico y requiere un esfuerzo de programación considerable.

La portabilidad de plataforma se consigue más fácilmente diseñando un proceso VM como parte de un completo entorno para el desarrollo de aplicaciones de alto nivel. El HLL VM que resulta no corresponde directamente a ninguna plataforma verdadera, es diseñado para facilitar la portabilidad.

En un sistema convencional, un compilador primero genera el código intermedio que es similar al código máquina, pero más compacto.

Entonces, un generador de código usa el código intermedio para generar un binario que contiene el código máquina para la ISA específica y el sistema operativo. Este archivo binario es distribuido y ejecutado en plataformas que soportan la combinación de esa ISA y Sistema Operativo.

En un HLL VM, un compilador genera el código de máquina abstracto en el ISA virtual que especifica el interfaz de la VM. Este código ISA virtual, junto con la información de estructura de datos asociada, es distribuido para la ejecución en plataformas diferentes.

Cada plataforma host implementa una VM con la capacidad de cargar y ejecutar un ISA virtual y un set de librerías especificadas por el API. En su forma más simple, el VM contiene a un intérprete y en el más sofisticado, compilan el código máquina abstracto en el código máquina del host para la ejecución directa.

Una ventaja de un HLL VM consiste en que el software de aplicación es fácilmente portable una vez que la VM y las librerías son puestas en práctica en la plataforma del host. Aunque la realización de una VM toma un poco de esfuerzo, es mucho más simple que el desarrollo de un compilador auténtico para una plataforma y portar cada aplicación por la recompilación.

Sun Microsystems Java VM architecture y Microsoft Common Language Infrastructure (.NET), son ejemplos extensamente usados de HLL VMs. Los ISAs en ambos sistemas son basados en pilas para eliminar exigencias de registro y utilizar una especificación de datos abstracta y un modelo de memoria que se apoya en la programación orientada al objeto.

LAS MÁQUINAS VIRTUALES DEl SISTEMA.

Un sistema VM proporciona un ambiente completo en el cual un sistema operativo y muchos procesos, posiblemente de usuarios diferentes, pueden coexistir. Usando el sistema VM, una plataforma de hardware puede soportar simultáneamente múltiples guest.

El Sistema VM surgido durante los años 1960 y comienzos de los 70s era el origen del término máquina virtual. Entonces, los sistemas de computadoras centrales eran muy grandes, caros, y por lo general se compartían entre numerosos usuarios; con la tecnología VM, los grupos de usuarios diferentes podían utilizar sistemas operativos diferentes en el hardware compartido.

Cuando el hardware se hizo más barato la mayor parte se migró a los escritorios, perdiendo interés este sistema original de VM. Hoy, sin embargo, sistemas VMs se utilizan cuando los sistemas de computadoras son servidores o grupo de servidores compartidos por muchos usuarios.

Quizás la aplicación común más importante de la tecnología de VM es la independencia que esto proporciona entre múltiples sistemas que corren simultáneamente en la misma plataforma de hardware. Si la seguridad en un sistema de guest está comprometida o si sufre una falla, el software que corre en otros guest no es afectado.

En un sistema VM, el VMM principalmente proporciona la réplica de la plataforma. La cuestión central es dividir los recursos del hardware entre entornos de sistemas operativos múltiples (un ejemplo es la virtualización del disco). El VMM tiene el acceso, y la administración de todos los recursos del hardware.

Un sistema operativo guest y sus procesos de aplicación son manejados por el control del VMM. Cuando un sistema operativo guest realiza una instrucción privilegiada u operación que directamente actúa con recursos compartidos del hardware, el VMM intercepta la operación, la verifica y la ejecuta. El guest es inconsciente de esta forma de trabajo.

VMs Clásicas.

De la perspectiva del usuario, la mayor parte de las VMs proporcionan esencialmente la misma funcionalidad, pero se diferencian en sus detalles de realización. El VMM se ejecuta en el modo de mayor privilegio, mientras todos los sistemas de guest tienen privilegios reducidos de modo que el VMM pueda interceptar y emular todas las acciones de sistema operativo del guest que tendrían acceso normalmente o que manipularían recursos críticos del hardware.

Hosted VMs.

Una implementación alternativa de un sistema VM es aplicar el software de virtualización encima del sistema operativo del host, resultando un hosted VM. Una ventaja del hosted VM consiste en que un usuario lo instala junto como el programa de aplicación. El software virtualizado puede confiar en el sistema operativo del host para proporcionar controladores de dispositivos y otros servicios de nivel inferior

Sistemas completos VMs.

En el sistema convencional VMs, todo el software del host y del guest, así como también la aplicación usan la misma ISA que el hardware. En algunas situaciones, sin embargo, los sistemas de guest y de host no tienen el mismo ISA. Por ejemplo, los dos sistemas de escritorio más populares de hoy, ordenadores personales de Windows y Apple, usan ISAs diferente (y sistemas operativos diferentes). El sistema completo VMs trabaja con la virtualización de todo el software, incluso el sistema operativo y las aplicaciones.

Como los ISAs son diferentes, el VM debe emular tanto el código de sistema operativo como el de la aplicación.

La virtualización del multiprocesador.

Una forma interesante del sistema de virtualización ocurre cuando la plataforma del host es un multiprocesador con una gran memoria para compartir. Aquí, el objetivo importante es dividir el sistema en múltiple sistemas más pequeños, distribuyendo los recursos del hardware.

Con la división física, los recursos que usan los sistemas virtuales son desvinculados de aquellos usados por otros sistemas virtuales. La división física proporciona un alto grado de aislamiento, de modo que ni problemas de software ni fallas de hardware en una partición afecten a programas en otras particiones. Con la división lógica los recursos del hardware son multiplexados en el tiempo entre las diferentes particiones, mejorando la utilización de los recursos del sistema. Sin embargo, se pierden algunas ventajas del aislamiento del hardware.

Las VMs de código.
La funcionalidad y portabilidad son los objetivos de la mayor parte de las VMs que son puestos en práctica en el hardware ya desarrollado para algún estándar ISA. Las VMs de código implementan un nuevo ISA, tratando de mejorar la performance y eficacia. El ISA del host puede ser completamente nuevo, o puede ser una extensión del ISA existente.
Una VM de código no es una aplicación ISA nativa. En cambio, el VMM es la parte de la implementación del hardware; su único objetivo es emular al ISA del guest. Para mantener esta ilusión, el VMM reside en una región de memoria oculta de todo el software convencional. Esto incluye a un traductor binario que convierte instrucciones del guest en secuencias optimizadas de instrucciones del host ISA y los almacena en la región de memoria oculta.

CLASIFICACIÓN DE LA MÁQUINA VIRTUAL.
Dada esta a amplia cantidad de VMs, con objetivos y realizaciones diferentes, es provechoso organizarlas. Pueden ser primero divididas en de procesos o en de sistemas. Dentro de estas dos categorías principales de las VMs, pueden ser divididas según si ellos usan la misma ISA o una diferente. La base para esta diferenciación es que la emulación de ISA es un rasgo dominante en aquellos VMs que la soportan. Entre el proceso VMs que no realizan la emulación de ISA son los sistemas multiprogramados, que la mayor parte de ordenadores de hoy ya la soportan. También incluidos están los optimizadores binarios dinámicos de igual ISA, que emplean muchas de las técnicas de la emulación del ISA.
Los procesos VMs con diferentes guest y host ISAs incluyen traductores dinámicos, con la interfaz de la máquina típicamente definida en el nivel ABI, y con el HLL VMs en la interfaz del nivel API. Las VMs de sistema consisten en el sistema clásico de VMs así como las hosted VMs que proporcionan replicación y entornos de sistemas aislados. La diferencia principal entre el sistema clásico y el hosted VM es la implementación del VMM más que la función que ellos proporcionan al usuario.
En un sistema completo VMs, en donde el guest y el host son ISAs diferentes, la performance es secundaria a la funcionalidad correcta. Cuando la eficacia o performance se hace importante, como es el caso de las VMs de código, la interfaz de implementación de la VM puede ser más cercana al hardware del procesador.

Los sistemas de computadoras modernos son estructuras complejas que contienen numerosos componentes que se relacionan estrechamente tanto en el software como en el hardware. Dentro de este universo, la virtualización actúa como una interconexión de esa tecnología. Interponer un software de virtualización entre las capas de abstracción cerca de la interfaz HW/SW forman una máquina virtual que permite que subsistemas incompatibles trabajen juntos.
Además, la replicación por virtualización permite el uso más flexible y eficiente de los recursos del hardware.
Las VMs son usadas ampliamente para permitir la interoperabilidad entre el hardware, el software de sistema, y el software de aplicación.
Considerando la confianza basada en estándares y la consolidación que ocurre en la industria, es probable que un nuevo ISA, o un sistema operativo, o un lenguaje de programación estén basados en la tecnología VM. En el futuro, las VMs deberían ser vistas como una disciplina unificada al mismo grado que el hardware, los sistemas operativos, y el software de aplicación lo son hoy.