

Procesos

Prof. Mariela Curiel

Bibliografía

- A. Tanenbaum & M. Van Steen. "Sistemas Distribuidos. Principios y Paradigmas". 2da. Edición.
- Smith & Nair. The Architecture of Virtual Machines. IEEE Computer. 2005.

Threads y procesos

- La idea básica es construir procesadores virtuales (software) que se ejecuten sobre procesadores físicos.
- **Procesador:** Provee un conjunto de instrucciones y la capacidad de ejecutarlas.

Definiciones

Un **proceso** es una entidad que posee 2 características importantes:

- **Recursos:** Básicamente: un espacio de direcciones (programas, datos y pila y un PCB), archivos, memoria, etc. El SOP realiza la función de protección para evitar interferencias no deseadas entre procesos en relación con los recursos.
- **Planificación/Ejecución:** El proceso sigue una ruta de ejecución. Tiene un PC, un Estado de ejecución (Listo, bloqueado, ejecutándose, etc.) y una prioridad.

Definiciones

- Estas dos características son independientes y pueden ser tratadas como tales por los sistemas operativos. En algunos sistemas operativos **se le denomina a la unidad activa hilo (thread)** y a la unidad propietaria de recursos se le suele denominar **proceso o tarea**.

Definiciones

En un entorno multihilo **se le asocia a los procesos:**

- Un espacio de direcciones virtuales que soporta la imagen del proceso.
- Acceso protegido a procesadores, otros procesos, archivos y recursos de E/S

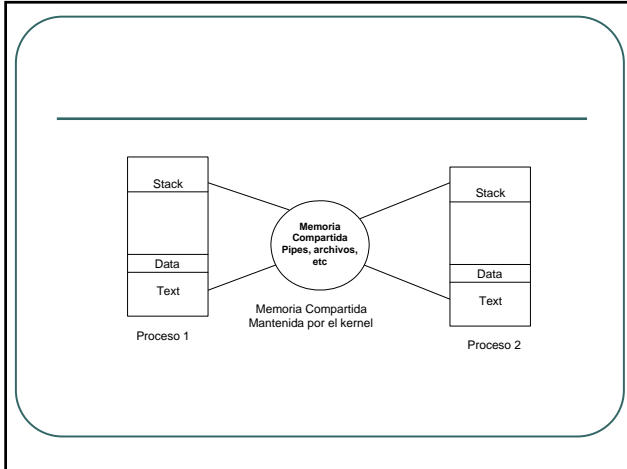
Definiciones

En un entorno multihilo **se asocian a cada hilo:**

- un estado de ejecución.
- Un PC, un contexto (conjunto de registros) que se almacena cuando no está en ejecución.
- Una pila.
- Un espacio de almacenamiento para variables locales.
- Acceso a la memoria y recursos del proceso.

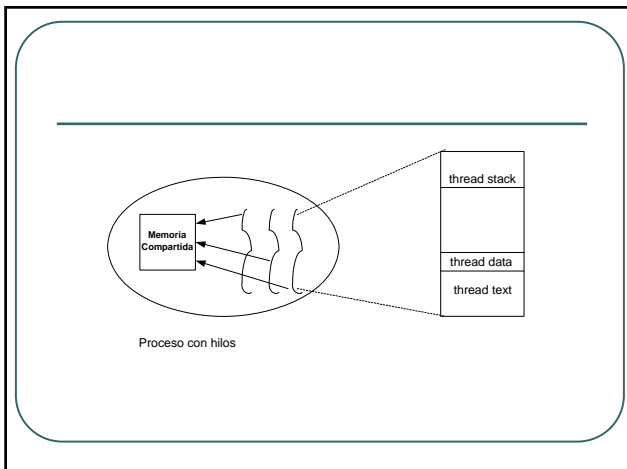
Definiciones

- Para que la comunicación entre procesos sea posible es necesario usar las **llamadas al sistema**.



Modelo de un Proceso Multithreaded

- Cada thread dentro de un proceso es una entidad independiente, pero dado que los threads forman parte de un mismo proceso o espacio de direcciones, la comunicación entre threads es mucho más rápida y simple.
- La comunicación entre varios threads se lleva a cabo a través de recursos del proceso usuario, no a través de los recursos del kernel.



Características de los threads

- Cada thread es independiente de otro thread no obstante, ciertas acciones que un thread lleve a cabo, afectarán a otros threads dentro del mismo proceso. Por ejemplo, si un thread abre un archivo, el archivo estará abierto para el resto de los threads dentro del mismo proceso. Si un thread termina usando la llamada al sistema `exit()`, esto provocará que todos los threads dentro del mismo proceso terminen.

Características de los threads

- No existe forma de predecir el orden de ejecución o el orden de culminación de varios threads.
- Los threads son invisibles fuera de los límites del programa o proceso que los contiene.

Cambios de Contexto

- Los threads comparten el mismo espacio de direcciones. Por lo tanto el cambio de contexto entre un thread y otro que pertenecen al mismo proceso pudiera hacerse de forma totalmente independiente del sistema operativo.
- Los cambios de contexto de procesos son más costosos. Implican salvar el contexto del proceso, cambio de modo (trap al sistema operativos, etc), otro cambio de modo, restaurar contexto del nuevo proceso.

Por qué Usar Threads?

Los mayores beneficios de los hilos provienen de las consecuencias del rendimiento:

- **Lleva mucho menos tiempo crear un hilo en un proceso existente, que crear un proceso totalmente nuevo. La creación de un hilo puede ser hasta 10 veces más rápida que la creación de un proceso en Unix.**
- Lleva menos tiempo finalizar un hilo que un proceso.
- Lleva menos tiempo el cambio de hilo que el cambio de proceso.
- Debido a que los threads son parte del mismo proceso, el compartir datos entre ellos es muy fácil; todo los threads tienen acceso a los datos globales. No se necesitan llamadas al sistema para su comunicación.

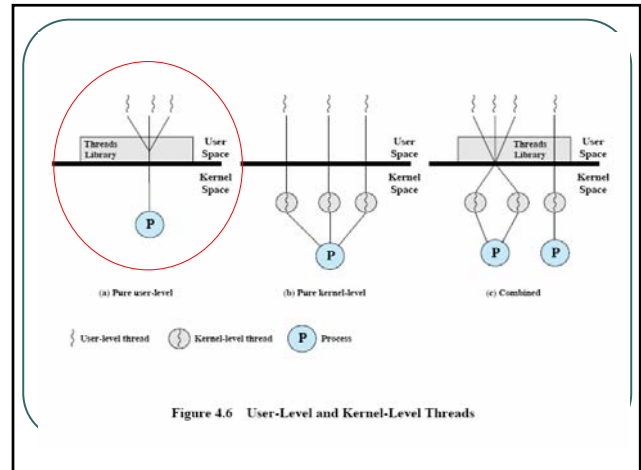
Por qué Usar Threads?

Los mayores beneficios de los hilos provienen de las consecuencias del rendimiento:

- Mayor eficiencia si se tienen un multiprocesador (y se soportan hilos a nivel del kernel). Paralelismo.
- Mayor concurrencia aún si se desconocen los hilos a nivel de kernel (si se logra a través de la librería que al bloquearse un hilo, pueda ejecutarse otro dentro del mismo proceso).
- Los programas que realizan diversas tareas o que tienen varias fuentes y destinos de E/S (servidor) se pueden diseñar e implementar fácilmente usando hilos.

Implementaciones

- Threads a nivel de usuario.
- Threads a nivel de kernel
- Procesos Livianos



Hilos a Nivel de Usuario (ULT)

- El kernel probablemente desconozca la existencia de los hilos.
- Este tipo de hilos es soportado por librerías (ej. C-threads de Mach o los pthreads de POSIX), las cuales ofrecen las funciones para la creación, sincronización y scheduling de threads.

Hilos a Nivel de Usuario (ULT)

- Como no se involucra al kernel, las operaciones sobre los threads son más rápidas. La librería actúa como un mini-kernel que controla los threads.
- Las operaciones de creación, destrucción, sincronización, etc., son llamadas a rutinas de librería, no son llamadas al sistema. Cuando un hilo se suspende, la librería toma el control, salva su contexto y permite la ejecución de otro hilo dentro del mismo proceso.

Hilos a Nivel de Usuario

- Cualquier aplicación puede programarse para ser multihilos a través del uso de una librería de hilos

Ventajas de los Hilos a nivel de Usuario

- El cambio de hilo se hace a nivel de proceso usuario, no hay un cambio de modo. Esto ahorra el overhead que producen los dos cambios de modo.
- Se pueden diseñar (si la librería lo permite) estrategias de planificación adecuadas a la aplicación.
- Se pueden ejecutar en cualquier sistema operativo.

Desventajas

- Si un hilo hace una llamada al sistema bloqueante, se bloquearán todos los hilos del proceso a menos que se use la técnica del revestimiento.
- No se obtienen ventajas con el multiprocesamiento.

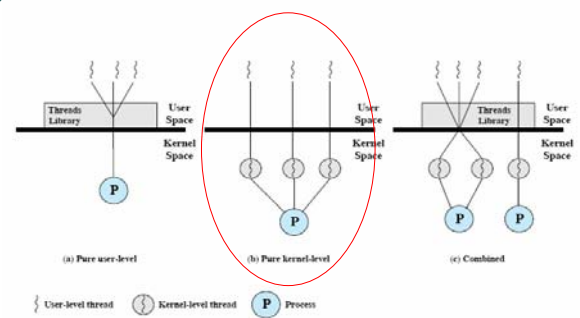


Figure 4.6 User-Level and Kernel-Level Threads

Threads a Nivel de Kernel

Hilos a Nivel del Kernel (KLT)

- El núcleo gestiona los hilos. No hay código de gestión de hilos en la aplicación, sólo la interfaz (API) para acceder a las utilidades de los KLT.
- Cualquier aplicación puede programarse para ser multihilo. Todos los hilos de una aplicación se mantienen en un solo proceso.
- El núcleo mantiene toda la información de cada hilo y realiza la planificación.

Ventajas

- Varios hilos de un mismo proceso se pueden planificar en distintos procesadores.
- Si se bloquea un hilo de un proceso, el SOP puede planificar otro hilo del mismo proceso.
- Las rutinas del núcleo pueden ser en si misma multihilos (demonios).

Desventaja

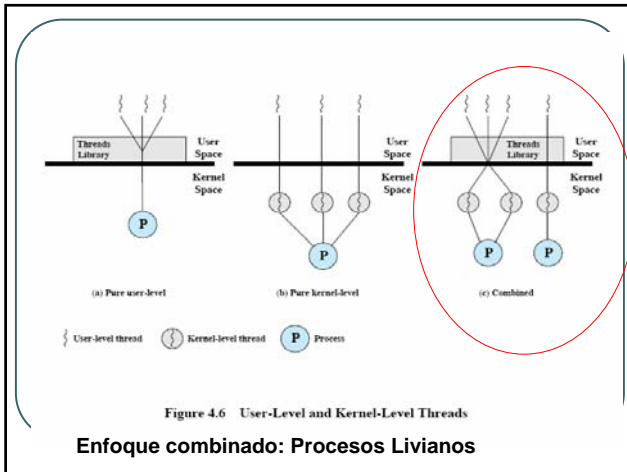
- La principal desventaja es que el cambio de hilo, requiere un trap al sistema operativo y dos cambios de modo.

operación	Hilos a nivel de usuario	Hilos a nivel de nucleo	Procesos
Crear Proceso Nulo	34	948	11300
Señalizar-esperar	37	441	1840

tiempos en microsegundos

Desventajas

- Si varios threads acceden el mismo conjunto de datos, este acceso debe ser sincronizado (semáforos, variables de condición, etc.). Tanto las llamadas de sincronización como las de creación y destrucción de threads del kernel requieren de llamadas al sistema.



Enfoque combinado

- El kernel sólo reconoce y gestiona a los procesos livianos. La librería a nivel de usuario multiplexa los threads del usuario en un determinado número de procesos livianos y se encarga del **scheduling, cambio de contexto y sincronización** entre threads sin involucrar al kernel.

Enfoque Combinado

- La librería seleccionará un thread a nivel de usuario para que se ejecute en un proceso liviano. La librería también garantiza la ejecución de threads que no estén bloqueados. Si un thread usuario que se está ejecutando llega a bloquearse, la librería elige a otro thread para ejecución.

Enfoque Combinado

- Un thread de usuario usará exactamente un proceso liviano (a nivel del kernel). Cuando el thread usuario concluye, el proceso liviano se preserva y puede ser utilizado por otro thread usuario del mismo proceso o de otro proceso usuario.

Enfoque combinado

- Este concepto ha sido virtualmente abandonado, se habla básicamente de threads a nivel de kernel o de threads a nivel de usuario.

Hilos en Sistemas Distribuidos

- Usando hilos, se puede permitir el uso de llamadas al sistema bloqueantes sin necesidad de “bloquear” todo el proceso. Esta propiedad vuelve a los hilos particularmente atractivos para su uso dentro de sistemas distribuidos. Concentrémonos en la *arquitectura Cliente-Servidor*.

Cientes Multihilos

- Sirven para esconder la latencia de comunicación a través de la red.
- Ejemplo 1: Navegadores WEB
 - En muchos casos una página WEB consiste de un texto plano con múltiples figuras.
 - Con frecuencia el navegador, establece la conexión con el servidor, recupera y comienza a desplegar la página HTML (incluso se permite al usuario el desplazamiento dentro de la página) mientras el navegador continúa recuperando otros archivos que conforman la página.

Cientes Multihilos

Ejemplo 1: Navegadores WEB

- Desarrollar navegadores multihilos simplifica este hecho de forma considerable. Tan pronto como llega la página principal se pueden activar hilos que se encarguen de recuperar las demás partes. Cada hilo establece su propia conexión con el servidor.
- Mientras tanto el usuario advierte el retardo en las imágenes pero puede ir explorando el documento.

Cientes Multihilos

- Si el servidor está saturado o es lento no se observarán mejoras notables en el rendimiento.
- Ejemplo 2: Servidores WEB Replicados
 - En muchos casos los servidores se replican en distintas máquinas y cada servidor proporciona el mismo conjunto de documentos WEB. Están localizados en el mismo sitio y se conocen por el mismo nombre.

Cientes Multihilos

- Ejemplo 2: Servidores WEB Replicados
 - Cuando entra una petición para una página WEB es re-enviada a uno de los servidores (usando round-robin u otra técnica de balanceo de carga)
 - Cuando se usan clientes multihilos cada conexión puede ir a una réplica diferente del mismo servidor. En este caso los distintos archivos se transmiten en paralelo asegurando que la página WEB completa se despliega en un tiempo más corto.

Servidores Multihilos

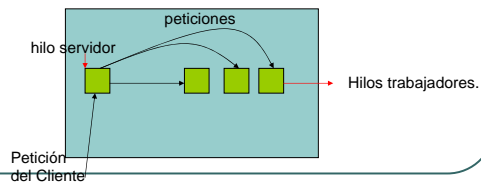
- El principal uso de la tecnología multihilos está del lado del servidor.
- Básicamente buscan mejorar el desempeño (aún en servidores monoprocesador) y la forma cómo se estructura el servidor.

Servidores Multihilos

- Ejm: por lo general un servidor de archivos espera una petición de entrada para una operación de archivo, posteriormente ejecuta la petición (operación bloqueante al disco) y luego envía la respuesta de regreso.

Servidores Multihilos

- Alternativas:
 - Modelo Servidor/Trabajador: Las peticiones son enviadas por los clientes hasta el servidor. Después de examinar la petición el hilo servidor elige un hilo trabajador sin utilizar y le encarga la petición.



Servidores Multihilos

- El hilo trabajador realiza la lectura, lo cual puede provocar que se suspenda hasta que los datos sean recuperados. Si el hilo se suspende, el procesador, selecciona otro para su ejecución

Servidores Multihilos

- Cómo se programa un servidor de archivos en la ausencia de hilos?
 - El servidor recibe a peticiones, las examina y las trata de resolver antes de recibir la siguiente petición. Mientras espera por el disco el servidor está ocioso y no procesa otra petición. Se procesan menos peticiones por segundo (throughput).
 - Con los hilos se gana un rendimiento considerable. Cada hilo se programa secuencialmente en forma tradicional.

Servidores Multihilos

- Supongamos que los hilos no están disponibles, pero los diseñadores del sistema consideran inaceptable el rendimiento debido al uso de un solo hilo y operaciones bloqueantes.
- Sol: uso de operaciones no bloqueantes (asíncronas).

Servidores Multihilos

- Cuando llega una petición se examina. Si se puede satisfacer desde el cache bien. Sino, se envía un mensaje al disco (no bloqueante). Se registra el estado de la petición actual en una tabla y se va al siguiente mensaje. El siguiente mensaje puede ser una petición para otro trabajo o una respuesta del disco relacionada con una operación previa. En este último caso se recupera la información relevante de la tabla y junto con los datos solicitados, se envía al cliente.

Servidores Multihilos

- En este caso la programación del servidor es más difícil y se necesita que el sistema operativo provea llamadas asíncronas.

Contenido

- Hilos
 - Repaso del concepto de hilos y procesos
 - Hilos en sistemas distribuidos.
- ▶ Virtualización
- Clientes
- Servidores
- Migración de Código

Virtualización

- Definiciones y rol de la virtualización en sistemas distribuidos.
- Diferentes tipos de máquinas virtuales.

Rol de la Virtualización en los Sistemas Distribuidos

- Una de las razones más importantes de introducir la virtualización en la década de los 60/70 fue la de permitir que los mainframes (muy costosos) fueran usados por múltiples aplicaciones (no había sop multiprogramados). Se particionaba la plataforma de Hw en una o más máquinas virtuales. Cada máquina era lo suficientemente similar al software hardware, para que las aplicaciones pudieran ejecutarse sin ningún cambio.

Rol de la Virtualización en los SD

- En los 80 y 90 se desarrollaron los sistemas de operación multitarea y simultáneamente se produjo una caída en los costos de Hw y los monitores de máquina virtual fueron desapareciendo.
- Pasando el 2000, las máquinas virtuales vuelven a ser un tópico importante en la academia (aunque algunos casos, como el de JAVA ya llevan algunos años)

Rol de la Virtualización en los SD

- El hw y sw de soporte (bajo nivel) cambian razonablemente rápido, las aplicaciones se hacen más estables. El software heredado (legacy) no se puede actualizar al mismo paso que las plataformas sobre las que se apoya. La virtualización puede ayudar a desarrollar interfaces heredadas en nuevas plataformas.

Rol de la Virtualización en los SD

- Las redes son ahora completamente masivas. Los administradores tienen que administrar una cantidad de recursos heterogéneos, ejecutando aplicaciones muy diferentes, a los cuales tienen acceso los clientes. Estos recursos, deberían ser fácilmente accesibles desde las aplicaciones.

Rol de la Virtualización en los SD

- En este caso la virtualización puede ayudar mucho:
 - reducir la heterogeneidad. Esto aumenta la portabilidad.
 - Al no tener las aplicaciones acceso directamente al Hw. pueden disminuir los problemas de seguridad.
 - Si falla una máquina virtual las demás se siguen ejecutando, esto permite mayor confiabilidad.

Definiciones

- **Abstracción:** “simplificar lo que está en niveles inferiores”. El SOP abstrae los detalles de un disco duro (pistas, cilindros, sectores) y presenta a los programas un conjunto de archivos. Los programadores de aplicación trabajan sobre los archivos sin preocuparse de los detalles del dispositivo físico.

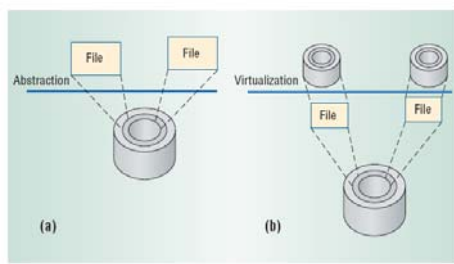


Figure 1. Abstraction and virtualization applied to disk storage. (a) Abstraction provides a simplified interface to underlying resources. (b) Virtualization provides a different interface or different resources at the same abstraction level.

Definiciones

- El conjunto de instrucciones de una arquitectura (ISA), ejemplifica las ventajas de abstracciones e interfaces bien definidas:
 - Por ejemplo la gente de Intel diseña microprocesadores que implementan el conjunto de instrucciones del arquitectura IA-32 (x86). Mientras los desarrolladores de Microsoft escriben software que se compila al mismo conjunto de instrucciones. Como todo el mundo cumple las especificaciones el software debe ejecutarse correctamente el cualquier PC que tenga un microprocesador IA-32.

Definiciones

- Las interfaces bien-definidas tienen sus limitaciones. Los sistemas y componentes diseñados según las especificaciones de una interfaz no trabajarán sobre otra interfaz o con componentes diseñados para otra interfaz. Por ejemplo programas compilados para una arquitectura no podrán ejecutarse en otra arquitectura. Esta falta de inter-operabilidad puede ser limitante especialmente en un mundo de redes y sistemas distribuidos donde trae sus ventajas mover tanto software como datos.

Definiciones

- Virtualizar un componente es “mapear” su interfaz y recursos encima de otro componente real, posiblemente diferente. El sistema real aparece como un sistema virtual diferente o múltiples sistemas virtuales. A diferencia de la abstracción, la virtualización no tiene como objetivos esconder detalles.

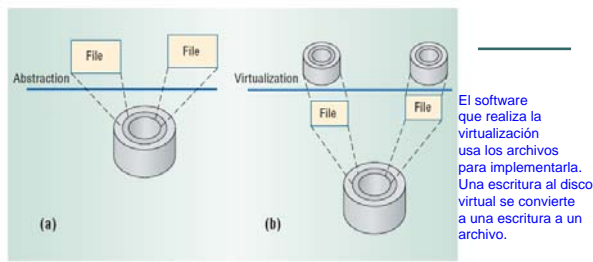
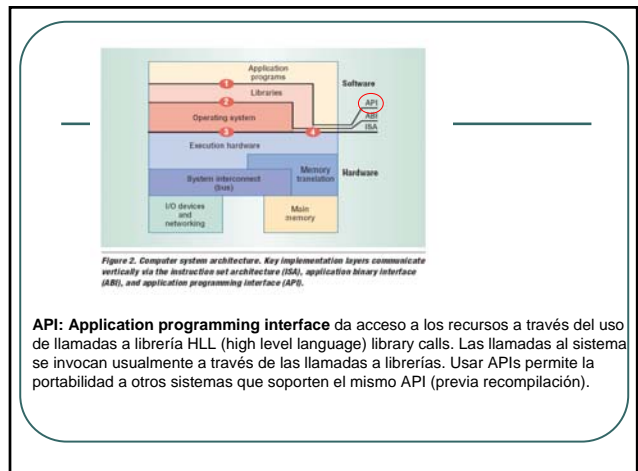
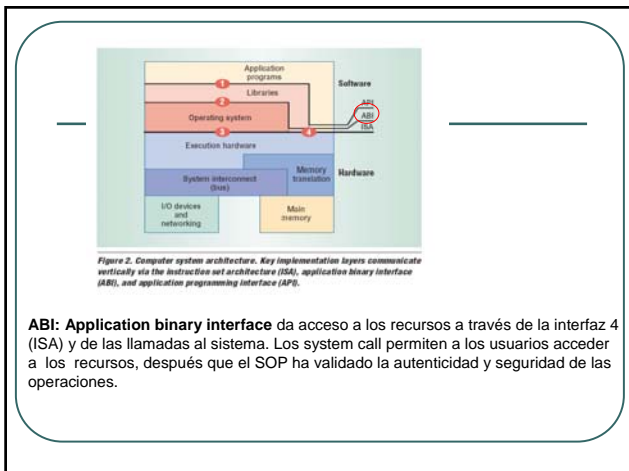
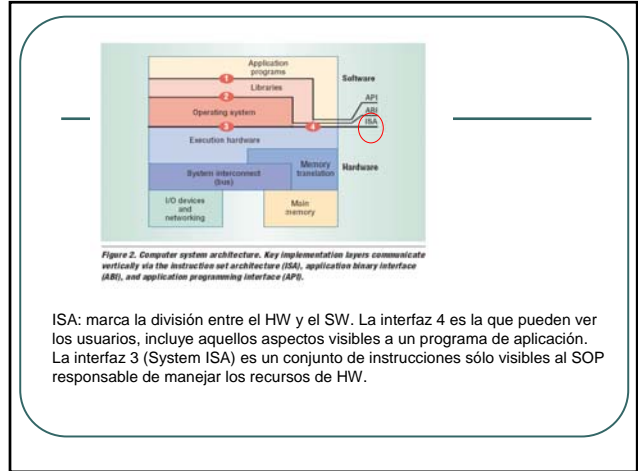
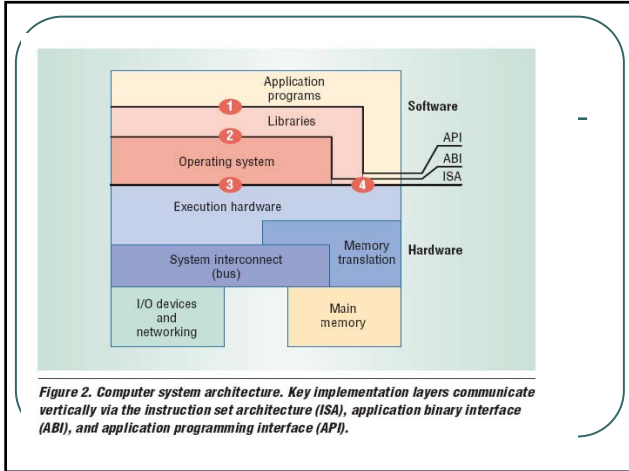


Figure 1. Abstraction and virtualization applied to disk storage. (a) Abstraction provides a simplified interface to underlying resources. (b) Virtualization provides a different interface or different resources at the same abstraction level.

Definiciones

- El concepto de virtualización se puede aplicar tanto a componentes por separado (discos, memoria) como a una máquina entera.
- La siguiente figura muestra algunos niveles importantes de abstracción e interfaces de los sistemas de computación. ISA (Instruction Set Architecture, Application Binary Interface y API) son bien importantes en la construcción de máquinas virtuales.



Tipos de Máquinas Virtual

- Desde la perspectiva de un proceso ejecutando un programa usuario, la máquina es el espacio de direcciones asignado al proceso, sus instrucciones y los registros. La E/S es sólo accesible a través del ABI. El ABI define la forma como el proceso "vé" los dispositivos reales.
- Desde la perspectiva del SOP, todo el sistema (sop, middleware, aplicaciones) se ejecuta en una máquina con ciertas características. Estas características son definidas por la ISA.

Tipos de Máquinas Virtual

- Existen máquinas desde la perspectiva de un proceso y desde la perspectiva de un sistema operativo.
- **Máquinas Virtuales de Proceso:** es una plataforma que ejecuta un proceso. Se crea cuando un proceso comienza y termina al igual que el proceso.
 - **Máquinas Virtuales de Sistema:** ofrecen un ambiente persistente que soporta la ejecución del sistema operativo y de los procesos.

Terminología

- El proceso que se ejecuta sobre una máquina virtual es el invitado "guest".
- La plataforma subyacente sobre la que se ejecuta la máquina virtual es el host.
- El software que implementa una máquina virtual de proceso es el runtime
- Y el software de virtualización en una máquina virtual de sistema se le denomina monitor de la máquina virtual VMM. Ejemplos típicos de este enfoque son VMware y XEN

Tipos de VM

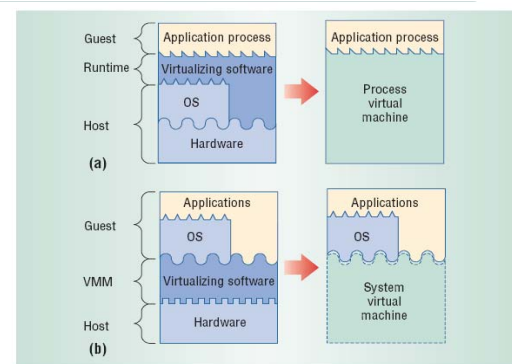


Figure 3. Process and system VMs. (a) In a process VM, virtualizing software translates a set of OS and user-level instructions composing one platform to those of another. (b) In a system VM, virtualizing software translates the ISA used by one hardware platform to that of another.

Máquinas Virtuales de Proceso

- **Sistemas Multiprogramados:** La mayoría de los SOP soportan múltiples procesos a través de la técnica de multiprogramación. Se le dá a cada proceso la ilusión de tener una “máquina propia”. (El código de todos los procesos va a la misma ISA)

Máquinas Virtuales de Proceso

-Emuladores y “dynamic binary translators”.

Se desea ejecutar un código binario que fue compilado para una arquitectura diferente en la arquitectura actual. Ejem: Intel IA32-El permite a las aplicaciones compiladas para Intel IA-32 correr en el procesador Itanium.

Máquinas Virtuales de Proceso

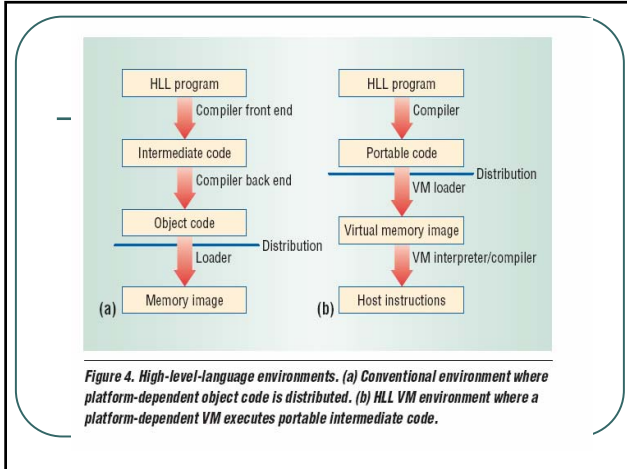
-Emuladores y “dynamic binary translators”.

La forma más fácil de emular (“imitar”) la arquitectura subyacente es a través de *interpretación*, a medida que se van ejecutando las instrucciones. Lento.

Dynamic binary translation: pasa o traduce las instrucciones del proceso invitado a las instrucciones del host (por bloque) y las deja en el cache para un posible reuso. Es una compilación Just-in-time, de código binario de una arquitectura a otra. (*Dynamic binary translation and optimization. Ebcioğlu, E. Altman, M. Gschwind, S. Sathaye. IBM T. J Watson Research Center.*)

Máquinas Virtuales de Proceso

- **High-Level-Language Virtual Machines:** se diseña una maquina virtual a nivel de proceso como parte de un ambiente de desarrollo de aplicaciones. La máquina virtual no corresponde a una plataforma específica sino que está diseñada para ejecutarse en muchas plataformas (portabilidad). La siguiente figura muestra las diferencias entre compilar un código de forma convencional y hacerlo en un ambiente HLL VM.
 - Java VM de Sun y la infraestructura Microsoft Common Lenguaje son ejemplos de HLL VM.
 - El software es portado fácilmente una vez que la máquina virtual y las librerías se implementan en la plataforma host.



Máquinas Virtuales de Sistema

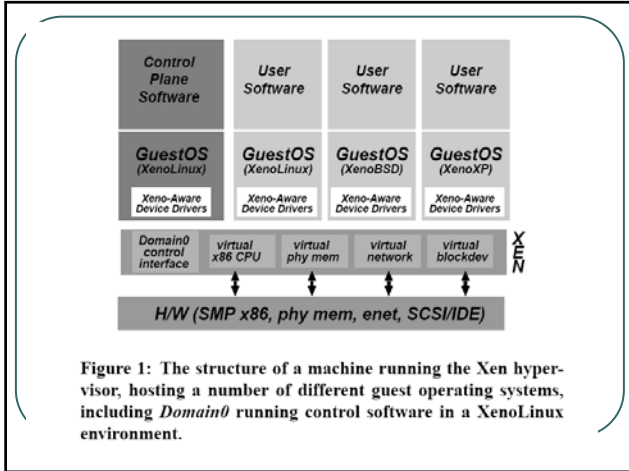
- Proporcionan un ambiente completo en el cual pueden coexistir múltiples sistemas operativos y procesos. Se pueden soportar muchos tipos de sops invitados.

Implementaciones

- **Clásicas:** VMM se coloca justo por encima del hardware y el resto del software por encima de él. EL VMM se ejecuta con los privilegios más altos mientras que los sistemas invitados se ejecutan con privilegios menores. El VMM tiene que tomar el control cuando los sistemas de operación invitados realizan acciones que manipulan los recursos críticos.

Clásicas

- El VMM provee una visión uniforme del hw. Hace que máquinas de diferentes vendedores con diferentes subsistemas de E/S luzcan similares. Así se puede ver un SD, como un pool de recursos dispuesto a satisfacer demandas de servicio.
- Una arquitectura es virtualizable si soporta la técnica de Ejecución Directa: el VMM debe tener siempre el control final del CPU. El procesador debe, entre otras cosas soportar más de dos modos de ejecución privilegiada.



Implementaciones

- **Hosted VM:** se coloca el software virtualizante por encima del sistema operativo o al mismo nivel. Ventaja: se instala como una aplicación más del sistema. El software virtualizante puede confiar en el sop subyacente para proveer los servicios de acceso a los dispositivos. Un ejemplo de este sistema es VMware GSX server

Implementaciones

- **VM del sistema completo:** se emula no sólo el Hw sino el sistema operativo. Ejemplo: Windows PC y los sistemas basados en Apple Power PC tienen diferentes ISA y diferentes SOP. Si quiero ejecutar una aplicación windows sobre un Apple Power-PC se tiene que emular la ISA y el SOP. Ejemplo Virtual PC: un sistema windows se ejecuta en una plataforma Macintosh.

Contenido

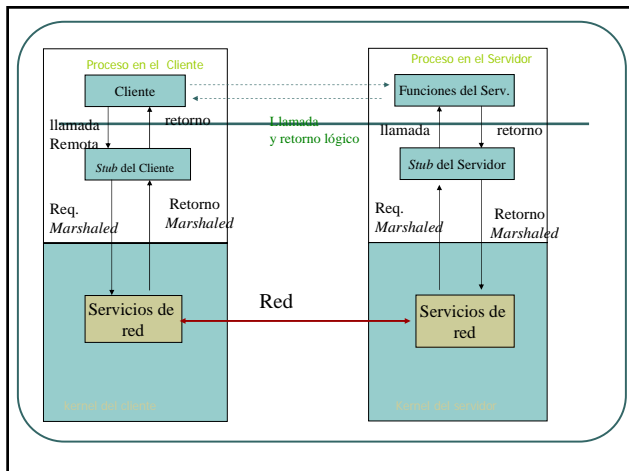
- Hilos
 - Repaso del concepto de hilos y procesos
 - Hilos en sistemas distribuidos.
- Virtualización
- ▶ Clientes
- Servidores
- Migración de Código

Cientes

- Proporcionan los medios necesarios para que los usuarios interactúen con servidores remotos.
- Existen dos maneras para soportar la interacción:
 - Por cada servicio remoto la máquina cliente tiene un servicio por separado que puede contactar el servicio en el servidor. Ejem: Bases de datos locales y remotas en dispositivos cliente especiales.
 - Los clientes sólo ofrecen la interfaz. La máquina cliente sólo se utiliza como terminal sin necesidad de almacenamiento local. Se denominan "clientes livianos". Las interfaces actuales son bien sofisticadas. El sistema de ventanas suele ejecutarse también en el servidor.

Cientes

- Software del lado del Cliente para la transparencia de acceso: ocultan al usuario los detalles de comunicación con procesos remotos, ejemplo: los stubs del lado del cliente de RPC.
- Transparencia de replicación: Si se tiene un servidor replicado, el software del lado del cliente puede enviar la misma petición a cada réplica, luego recopilar todas las respuestas y pasar sólo una respuesta a la aplicación cliente.



Servidores

- Un servidor es un proceso que implementa un servicio específico en una dirección (puerto) específica.
- Existen diversas formas de organizar servidores:
 - Servidores iterativos: el propio servidor lee la petición, la manipula y si es necesario devuelve una respuesta al cliente.

Servidores

- Existen diversas formas de organizar servidores:
 - Concurrentes: El servidor recibe la petición y se la pasa a otro thread o proceso.

Servidores

- Normalmente cada servidor o servicio está asociado a un puerto específico. Los puertos se asignan de manera global para servicios muy conocidos. Por ejemplo los servidores que atienden peticiones ftp están asociados al puerto 21. Un servidor HTTP para WWW siempre atenderá el puerto 80. Estos puertos fueron asignados por IANA (Internet Assigned Numbers Authority).
- Con los puertos asignados el cliente debe conocer únicamente la dirección de la máquina donde corre el servidor.

Servidores

- Otra opción: es muy común tener múltiples servidores en ejecución simultánea la mayoría de los cuales está esperando de forma pasiva hasta que entra la petición del cliente. En este caso lo mejor es tener un super-servidor que atiende varios puertos y cuando entra un requerimiento por uno de estos puertos se crea un sub-proceso que maneje la petición.

Servidores y Estado

- Aspectos del diseño de servidores:
 - Sin estado: no mantiene información del Edo. de sus clientes. Ejem: Un servidor WEB
 - Con estado: Mantiene información del estado de sus clientes. Ejem: un servidor de archivos que permite a los clientes mantener una copia local de un archivo, sobre la cual puede hacer incluso operaciones de actualización. Mejora el desempeño, no obstante puede haber problemas si el servidor falla.

Servidores y Estado

Servidores sin estado ("stateless")

- Cuando un cliente envía una solicitud a un servidor, éste la lleva a cabo, envía la respuesta y elimina de sus tablas internas toda la información relativa a dicha solicitud.
- No guarda información del cliente entre solicitudes. No mantiene un registro de las operaciones que van dejando los clientes
- Cada solicitud debe ser autocontenida.

Servidores y Estado

- Consecuencias:
 - Servidores y clientes son completamente independientes.
 - Se reducen los estados inconsistentes debido a caídas de clientes y/o servidores
 - Posible pérdida de performance, ej. Un servidor no se puede anticipar al comportamiento del cliente.
 - "El estado" del servidor no crece con más clientes

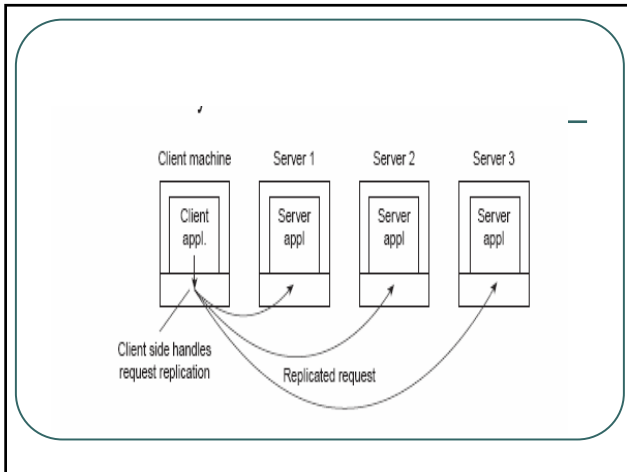
Servidores y Estado

Servidores con estado

- Los servidores guardan información del estado de los clientes entre solicitudes: tabla que asocia los descriptores de archivos con los archivos propiamente dichos.
- Pueden saber qué datos están en el cache del cliente (permiten al cliente mantener copias locales de datos compartidos).

Servidores y Estado

- Consecuencias:
 - Mejor performance: las actualizaciones se hacen localmente, mensajes mas cortos.
 - Si el servidor falla y se pierde la tabla que lleva las actualizaciones que realizan los clientes a cada archivo. No se puede garantizar que se han procesado las actualizaciones más recientes.



Cliente

- Software del lado del cliente para la transparencia de fallos. Se hace a través del middleware, se pueden hacer varios reintentos de conexión o intentar con otro servidor. También se puede, ante una falla devolver los datos del cache tal y como lo hacen algunos navegadores.