

marinamuca01

www.wuolah.com/student/marinamuca01

14760

Tema1SO-EstructurasdeSOs.pdf

Resúmenes Teoría



2º Sistemas Operativos



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.





**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

ARQUITECTURAS MONOLÍTICAS, MICROKERNEL Y MÁQUINAS VIRTUALES

Arquitectura Monolítica

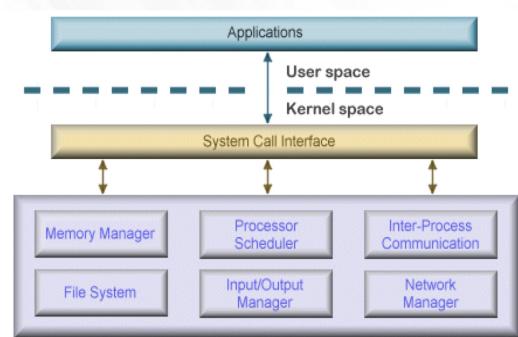
El SO es un único programa que se ejecuta en el modo más privilegiado del procesador (ring 0 en x86)

Las dependencias entre los distintos módulos son complejas salvo para algunos elementos bien establecidos
(¡NO oculta información!)
↳ entre módulos

El modelo de obtención de servicios es la llamada a procedimiento.

Problemas

- { La fuerte dependencia entre módulos provoca dificultades a la hora de comprender el código y de realizar modificaciones.
- Al ser un sólo programa cargado en memoria el fallo de un módulo puede provocar la caída del sistema.



Arquitectura Microkernel

Una pequeña parte de la funcionalidad del SO se implementa como Kernel y el resto como procesos de usuario.

El microkernel soporta memoria virtual de bajo nivel, creación de procesos (hebras) y, comunicación y sincronización.

El modelo de obtención de servicios es por paso de mensajes entre procesos.

La aplicación solicita un servicio al SO, send (Server, &m), y espera la resolución del servicio, receive (Server, &r).

El microKernel obtiene la solicitud de servicio y entrega el mensaje al servidor Server (si el server necesita el ↳ de una funcionalidad del SO).

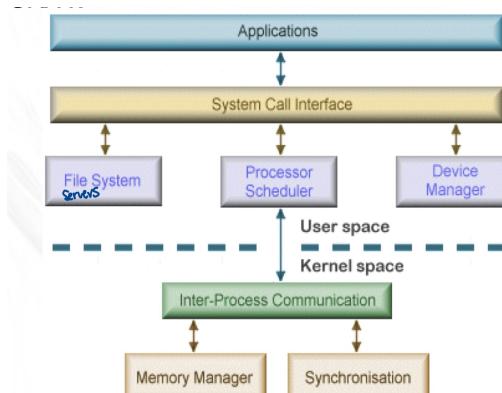
El servidor Server obtiene el mensaje m, genera el resultado y envía el resultado al Kernel para que éste lo devuelva a la aplicación, r.

Ventajas

- { Fiabilidad → un error en un módulo que se implemente como proceso de usuario sólo provoca una caída parcial del sistema que puede recuperarse levantando el proceso de servicio.
- Extensibilidad → Podemos incluir nuevos servicios como procesos de usuario.

Desventajas →

- Peor rendimiento → que la arquitectura monolítica porque para obtener un servicio se realizan más cambios de modo y de espacio de direcciones.



Virtualización

1960 → Virtualización se refiere a una máquina virtual = duplicado aislado de un computador real.

VM (Virtual Machine) = capaz de usar recursos HW (CPU, memoria, almacenamiento y E/S) virtualizados (abstracción de los recursos reales o físicos) por el VMH.

VMM (Virtual Machine Monitor) o hipervisor = software que proporciona la abstracción de la máquina real a las máquinas virtuales.

Requisitos para que un HW soporte virtualización

Equivalencia → programa ejecutándose sobre un hipervisor debe comportarse de forma idéntica a si se ejecutase sobre la máquina física real equivalente.

Control de recursos → El hipervisor es el único software que controla los recursos virtualizados.

Eficiencia → La mayor proporción de instrucciones deben ejecutarse sin la intervención del hipervisor.

Hipervisor y Máquinas virtuales → Capa software (hipervisor) que proporciona recursos virtuales (CPU, memoria, almacenamiento y dispositivos) a máquinas virtuales (SO, bibliotecas y aplicaciones).

OS-level virtualization → El Kernel permite múltiples instancias aisladas a nivel de usuario (containers). Un programa no verá todos los recursos del SO sino solamente los asignados al container.

Tíos de virtualización

Virtualización actualmente = capa de abstracción software situada entre el software a ejecutar y el hardware real.

Hipervisor y máquinas virtuales

Hipervisor crea entorno de trabajo, la VM, para el software a ejecutar.

El hipervisor puede ser software, firmware o hardware y el ordenador que el que se ejecuta se denomina guest machine. Más recientemente, el desarrollo y gestión de hipervisores y VMs se denomina platform virtualization o server virtualization.

La virtualización permite que un único ordenador ejecute distintos SOs o múltiples ejecuciones del mismo SO.

La máquina virtual dispone de una serie de recursos virtuales proporcionados por el hipervisor y sobre ella se ejecuta el SO (guest OS), las bibliotecas y las aplicaciones.

El ratio de consolidación indica el número de VM que proporciona un determinado hipervisor.

↳ Cuántas máquinas es capaz de tirar el HW a la vez.

Ejemplo: 7:1 indica 7 VMs en una máquina anfitriona (host machine).



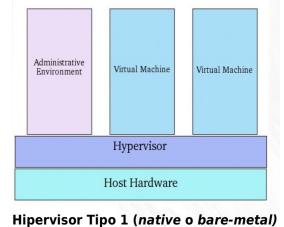
ENCENDER TU LLAMA CUESTA MUY POCO



Tipos de hipervisor

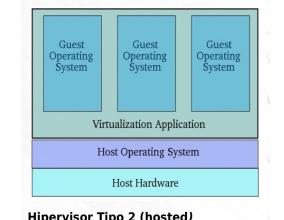
Tipo 1 (native o bare-metal) → Se ejecutan directamente en el host HW para proporcionar VM a los SO invitados.

Ejemplo: Xen, VMware ESX/ESXi; (servidor de virtualización)



Tipo 2 (hosted) → Se ejecutan sobre un SO convencional como el resto de programas y el SO invitado se ejecuta sobre la abstracción proporcionada por el hipervisor.

Ejemplo: VMware Workstation y VirtualBox.



¿Por qué virtualizar?

- Permite ejecutar diferentes SOs en el mismo HW = 1er objetivo.
- Consolidación de servidores → un HW servidor se virtualiza y tienes distintos SO actuando como servers en en distintas VM.
- Desarrollo de Kernels. VM es más fácil de controlar y observar que un HW real ("no tocas mucho").
- VM relocation. VM con aplicaciones se traslada completamente a otro HW real.
- Evaluación de distintos SOs.

Un poco sobre implementación

Hipervisor → traduce las peticiones de recursos virtuales que proporciona a peticiones sobre los recursos reales.

Traducción provoca degradación en el rendimiento.

Una VM se representa mediante archivos

Archivo de configuración: CPUs, memoria, dispositivos E/S accesibles, etc.

Archivo de almacenamiento: discos virtuales y su correspondiente soporte mediante archivos reales.

funciones del hipervisor

Gestión de la ejecución de VMs

Planeación de VMs para ejecución en CPU
Gestión de memoria virtual para aislar VMs
Cambio contexto y emulación de timer e interrupciones.

Emulación de dispositivos y control de acceso.
Ejecución de instrucciones privilegiadas
Gestión de Máquinas Virtuales.
Administración del hipervisor.

Hipervisores tipo 1 vs. tipo 2

- **Tipo 1 mejor rendimiento que tipo 2**
 - Dispone de todos los recursos para las VM.
 - Múltiples niveles de abstracción entre SO invitado y HW real en tipo 2 no permiten alto rendimiento.
- **Tipo 2** permiten realizar virtualización sin tener que dedicar toda la máquina a dicho fin.
Ejemplo: prueba de Kernels o puesta a punto de servidores.

HW-assisted virtualization

- 1er HW para asistir a la virtualización incluido en IBM System/370 de 1972 como soporte para VM/370.
- Arquitectura x86 no soportaba HW-assisted virtualization \Rightarrow se desarrollan soluciones software
 - ↳ problema: SO que se ejecutan sobre MV necesitan ejecutarse en ring 0
 - Soluciones
 - Paravirtualización
 - full virtualization
- Paravirtualización
 - Hipervisor proporciona API (hypercalls en Xen) y el SO que se ejecuta en una VM utiliza dicha API.
 - Implicaciones de esta técnica
 - Código del guest OS debe estar disponible.
 - Reemplazar instrucciones que necesitan ring 0 por hypercalls Ejemplo: cli por vm_handle_cli.
 - ↳ == cambiar instrucciones privilegiadas del kernel por las de la API
 - Recompilar guest kernel y usarlo.
- full virtualization
 - Técnica implementada en la primera generación de hipervisores en x86.
 - Idea = traducción binaria para atrapar y virtualizar la ejecución de instrucciones sensibles (ring 0).
 - Las instrucciones sensibles se detectan (estáticamente o dinámicamente) y se reemplazan con trampas al hipervisor.
 - Esta técnica puede provocar sobrecarga en el rendimiento con respecto a una VM que se ejecute en HW que soporte virtualización, IBM System/370.
 - Solución final: HW-assisted virtualization
 - ↳ "Kernel un ring por debajo del ring 0."

OS-level virtualization: containers

Containers proporcionan un entorno aislado para la ejecución de programas sobre el mismo SO.

Linux Kernel control group (cgroup). Permite la agrupación de procesos y gestión de recursos del sistema.
↳ cgroup permite que distintas jerarquías de procesos coexistan en el mismo SO.

A cada jerarquía se le asignan un conjunto de recursos del sistema:

- Memoria máxima
- Prioridad para CPU
- Prioridad para dispositivos E/S

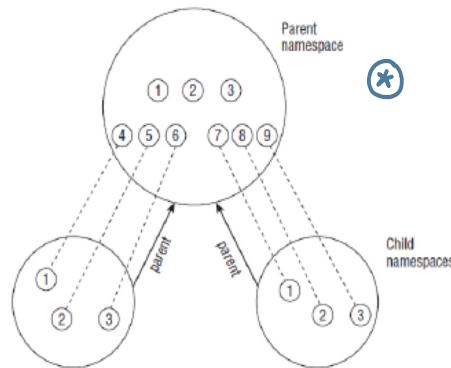
OS-level virtualization: Linux Namespaces

Espacio de nombres permite hacer visibles ciertos recursos del Kernel de forma única dentro de dicho espacio.

Algunos espacios de nombres incluidos actualmente:

- System V IPC (inter-process communication) y POSIX message queues.
- Mounts. Sistema de archivos montado.
- PID. Espacio de nombres para identificadores de procesos.
- Userid. Permite limitar los recursos por usuario.

* Ejemplo: 3 espacios de nombres y pid del proceso varía dependiendo del espacio de nombres utilizado



OS-level virtualization: UML y KVM

UML { SO anfitrión es un Kernel modificado que contiene extensiones para controlar múltiples máquinas virtuales cada una con su SO invitado.

KVM { Implementación full virtualization para Linux sobre HW x86 que contenga las extensiones correspondientes:
Intel VT o AMD-V.