Virtualización y jerarquía de memoria

Arquitectura de Computadores

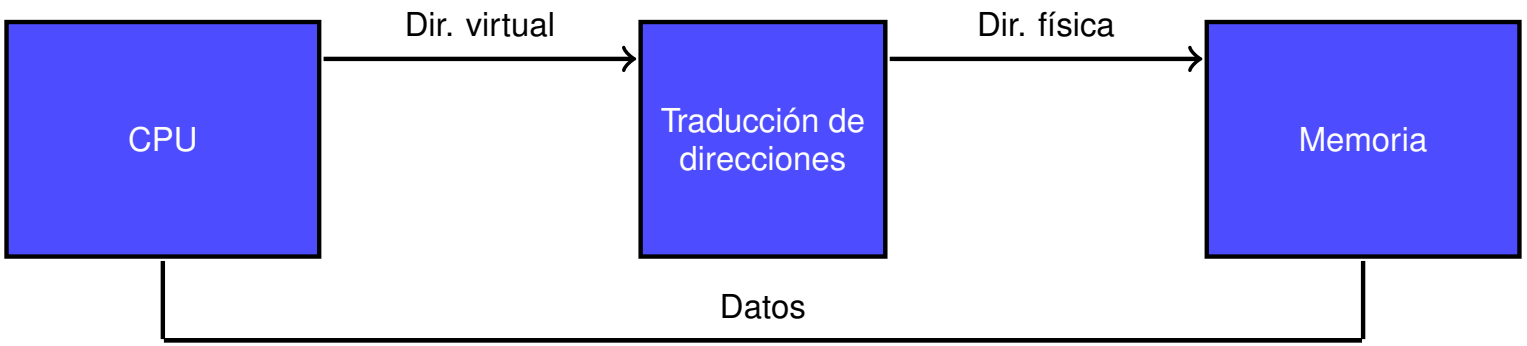
J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Límites del direccionamiento físico

- Todos los programas comparten un único espacio de direcciones.
  - **Espacio de direcciones físico**.
- No hay forma de prevenir que un programa acceda a un recurso.

Superando el límite físico



- Los programas se ejecutan en un **espacio de direcciones virtuales normalizado**.
- **Traducción de direcciones**:
  - Llevado a cabo por el hardware.
  - Gestionado por el SO.
- **Características soportadas**:
  - Protección, Traducción, Compartición.

Ventajas de la memoria virtual (I)

- **Traducción**:
  - Los programas pueden tener una **vista consistente** de la memoria.
  - Reduce el **coste** de aplicaciones **multi-hilo**.
  - Solamente hace falta tener en memoria el **conjunto de trabajo**.
  - Estructuras dinámicas usan solo la memoria física que **realmente necesitan** (p. ej. Pila).

Ventajas de la memoria virtual (II)

- **Protección**:
  - Permite **proteger** a unos procesos de otros.
  - Se pueden fijar **atributos a nivel de página**.
    - Solo lectura, ejecución, ...
  - Los datos del **núcleo** está protegidos de los programas.
  - Mejora la protección frente a **software malicioso**.

- **Compartición**:
  - Se puede **proyectar** una página a varios procesos.
    - P. ej.: Archivos proyectados en memoria.

Diferencias con caché

- **Remplazo**:
  - **Caché**: controlado por hardware.
  - **MV**: controlado por software.
- **Tamaño**:
  - Tamaño de caché independiente de longitud de dirección.
  - Tamaño de MV dependiente de longitud de dirección.

Parámetros

Parámetro	Caché L1	Memoria virtual
Tamaño de bloque	16 – 128 bytes	4096 – 65,536 bytes
Tiempo de acierto	1 – 3 ciclos	100 – 200 ciclos
Penalización de fallos	8 – 200 ciclos	10^6 – 10^7 ciclos
Tiempo de acceso	6 – 160 ciclos	$8 \cdot 10^5$ – $8 \cdot 10^6$ ciclos
Tiempo de transferencia	2 – 40 ciclos	$2 \cdot 10^5$ – $2 \cdot 10^6$ ciclos
Tasa de fallos	0.1% – 10%	0.00001% – 0.001%

Cuatro preguntas sobre la jerarquía de memoria

1. ¿Dónde se ubica un bloque en el nivel superior?
   - **Ubicación de bloque**.
2. ¿Cómo se localiza un bloque en el nivel superior?
   - **Identificación de bloque**.
3. ¿Qué bloque debe remplazarse en caso de fallo?
   - **Remplazo de bloque**.
4. ¿Qué ocurre en caso de escritura?
   - **Estrategia de escritura**.

Cuatro preguntas sobre la **memoria virtual**

1. ¿Dónde se ubica una **página** en la **memoria principal**?
   - **Ubicación de página**.
2. ¿Cómo se localiza un bloque en la **memoria principal**?
   - **Identificación de página**.
3. ¿Qué **página** debe remplazarse en caso de fallo?
   - **Remplazo de página**.
4. ¿Qué ocurre en caso de escritura?
   - **Estrategia de escritura**.

¿Dónde se ubica una página en memoria principal?

- Una página se puede ubicar en **cualquier marco de página** de la memoria principal.
  - Correspondencia totalmente asociativa.
- Gestión realizada por el sistema operativo.
- **Objetivo**: **Minimizar la tasa de fallos**.
  - No se puede hacer mucho con la penalización por fallo.
  - Penalización muy alta debida a lentitud de discos magnéticos.

¿Cómo se localiza una página en memoria principal?

- Se mantiene una **tabla de páginas por proceso** en memoria principal.
  - Tabla de **correspondencia** entre **identificador de página** e **identificador de marco de página**.

- Reducción de tiempo de traducción.
  - **TLB**: *Translation Lookaside Buffer*.
  - Evita accesos a la tabla de páginas de memoria principal.

¿Qué bloque debe remplazarse en caso de fallo de página?

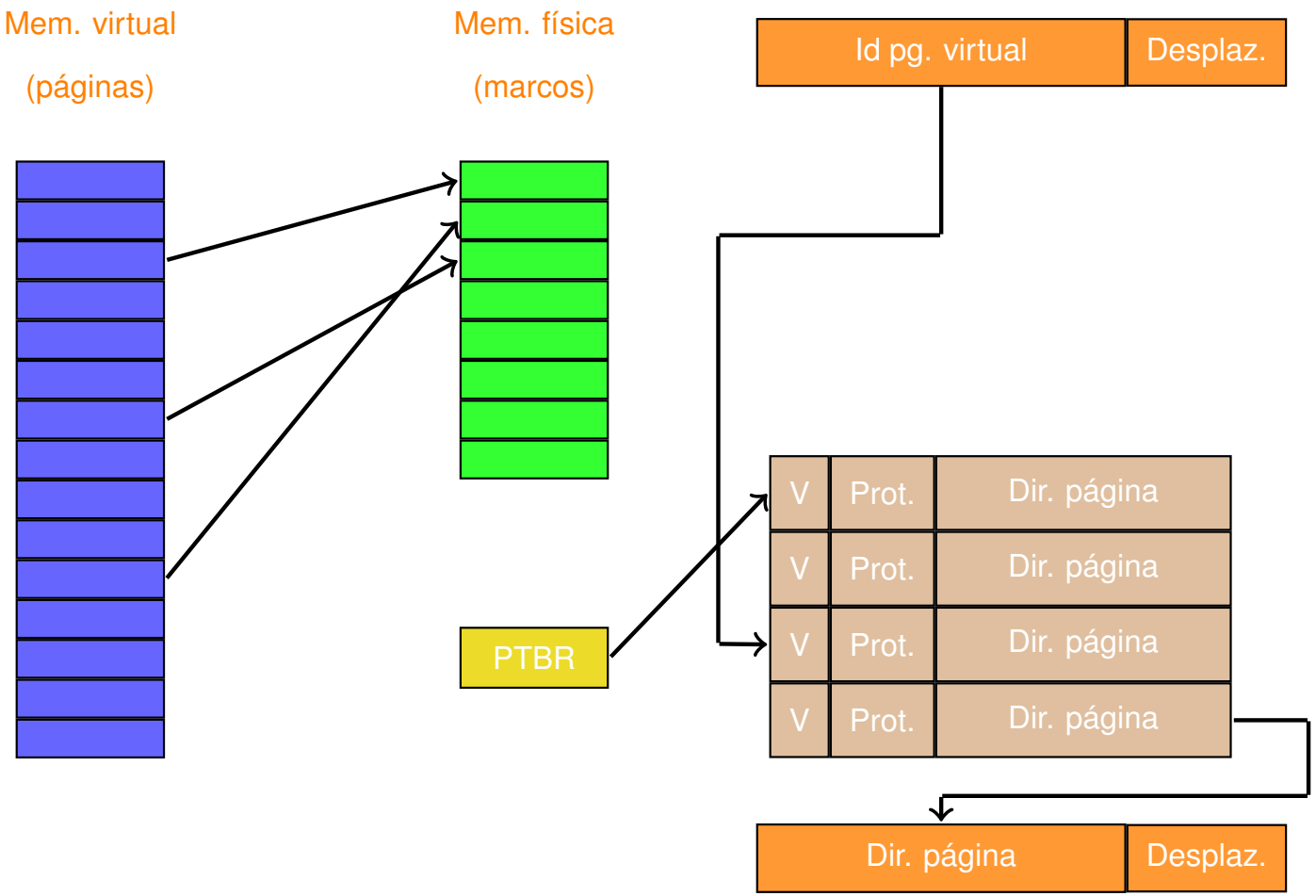
- Política de remplazo definida por el Sistema Operativo.
  - Típicamente **LRU** (*Least-recently used*).
- La arquitectura debe ofrecer soporte al Sistema Operativo.
  - **Bit de uso**: Activado cuando se accede a la página.
    - Cuando hay fallo en TLB.
  - El sistema operativo pone a cero este bit de forma periódica.
    - Registra valores más tarde.
    - Permite determinar páginas tocadas en un intervalo.

¿Qué ocurre en caso de escritura?

- La política de escritura es siempre **write-back**.
  - Nunca se ha construido un sistema de MV con write-through.
  - **¡No te sientas tentado!**

- Coste de escrituras en disco tremendamente alto.
  - Minimización de escrituras en disco.
  - Uso de **dirty bit** para indicar cuando se ha modificado una página.

Tabla de páginas



Tamaño de tabla de páginas

- Si se asume **direcciones virtuales** de **32 bits**, **páginas** de **4 KB** y **4 bytes** por **entrada en tabla**:
  - **Tamaño de tabla**:

$$\frac{2^{32}}{2^{12}} \times 2^2 B = 2^{22} B = 4MB$$

- **Alternativas**:
  - Tablas de páginas multi-nivel.
  - Tablas de páginas invertidas.
- **Ejemplo**: IA-64
  - Ofrece las dos alternativas al desarrollador del SO.

TLB: Translation Lookaside Buffer

- **Caso ideal**.
  - Cada acceso a memoria requiere dos accesos.
    1. Acceso a TP.
    2. Acceso a memoria.
  - Escenario peor en caso de tablas multinivel.

- **Solución**:
  - Usar caché de traducciones para evitar accesos a TP.
    - **Etiqueta**: Porción de dirección virtual.
    - **Datos**: Número de marco, bits de protección, bit de validez y dirty-bit.

Máquinas virtuales

- Desarrolladas a finales de los 60.
  - Usadas desde entonces en entornos *mainframe*.
  - Ignoradas en máquinas monousuario hasta finales de los 90.

- Popularidad recuperada debido a:
  - Importancia creciente de **aislamiento** y **seguridad** en sistemas modernos.
  - Fallos en **seguridad** y **fiabilidad** en sistemas operativos.
  - **Compartición** de un computador por varios usuarios.
  - Gran incremento en **prestaciones** de procesadores.
    - Sobrecarga de MMV's más aceptable.

Monitor de máquina virtual

A virtual machine is taken to be an efficient, isolated duplicate of the real machine. We explain these notions through the idea of a virtual machine monitor (VMM) . . .
. . . a VMM has threee essential characteristics.

- First, the VMM provides an environment for programs which is essentially identical with the original machine,
- second, programs run in this environment show at worst only minor decreases in speed;
- and last, the VMM is in complete control of system resources.

Fuente: Popek, G. y Goldberg, R. **Formal requirements for virtualizable third generation architectures**.

Communications of the ACM, Julio de 1974

Virtualización

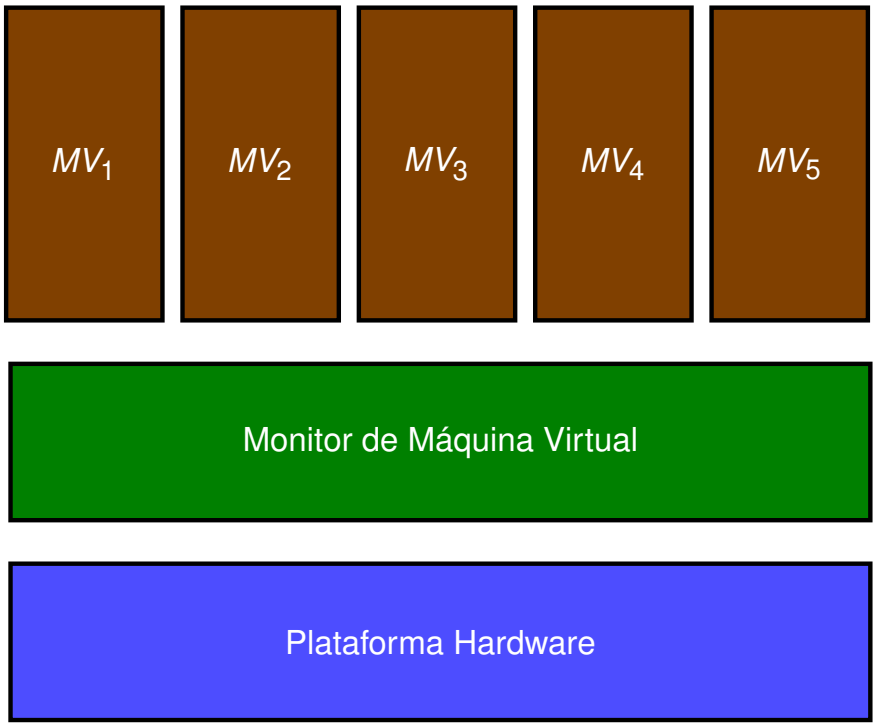
- **Definición general**: Cualquier método de emulación que ofrece una interfaz software estándar con la máquina física.
  - ¿Java VM?

- **Máquinas virtuales de sistema**: Ofrecen un entorno completo de sistema a nivel de ISA binaria.
  - Se suele asumir que ISA de MV e ISA de hardware son idénticas.
  - **Ejemplos**:
    - IBM VM/370.
    - VMWare ESX Server.
    - Xen.

Máquina virtual

- Ofrece la ilusión de que los usuarios tienen un **computador completo** a su disposición.
  - Incluyendo su copia del Sistema Operativo.
- Un computador ejecuta **varias máquinas virtuales**.
  - Puede soportar diversos sistemas operativos.
  - Todos los Sistemas Operativos comparten el hardware.
- **Terminología**:
  - **Host**: Plataforma hardware subyacente.
  - **Guest**: Máquinas virtuales que comparten recursos.

MV y VMM: Capas



- **MMV** → Capa de software de sistema.
  - El monitor se ejecuta sobre la plataforma hardware.
  - Permite la ejecución de varias máquinas virtuales sobre HW único.
  - Cada máquina virtual tiene su propio sistema operativo y aplicaciones.
  - Permite ejecutar aplicaciones sin modificarlas.

MMV

- **Monitor de máquina virtual** o **hipervisor**:
  - Software que soporta las máquinas virtuales.
- MMV determina la correspondencia entre **recursos virtuales** y **recursos físicos**.
- **Alternativas en compartición de recursos físicos**:
  - Compartición de tiempo.
  - Particionamiento.
  - Emulado por software.
- Un MMV es **más pequeño** que un SO tradicional.

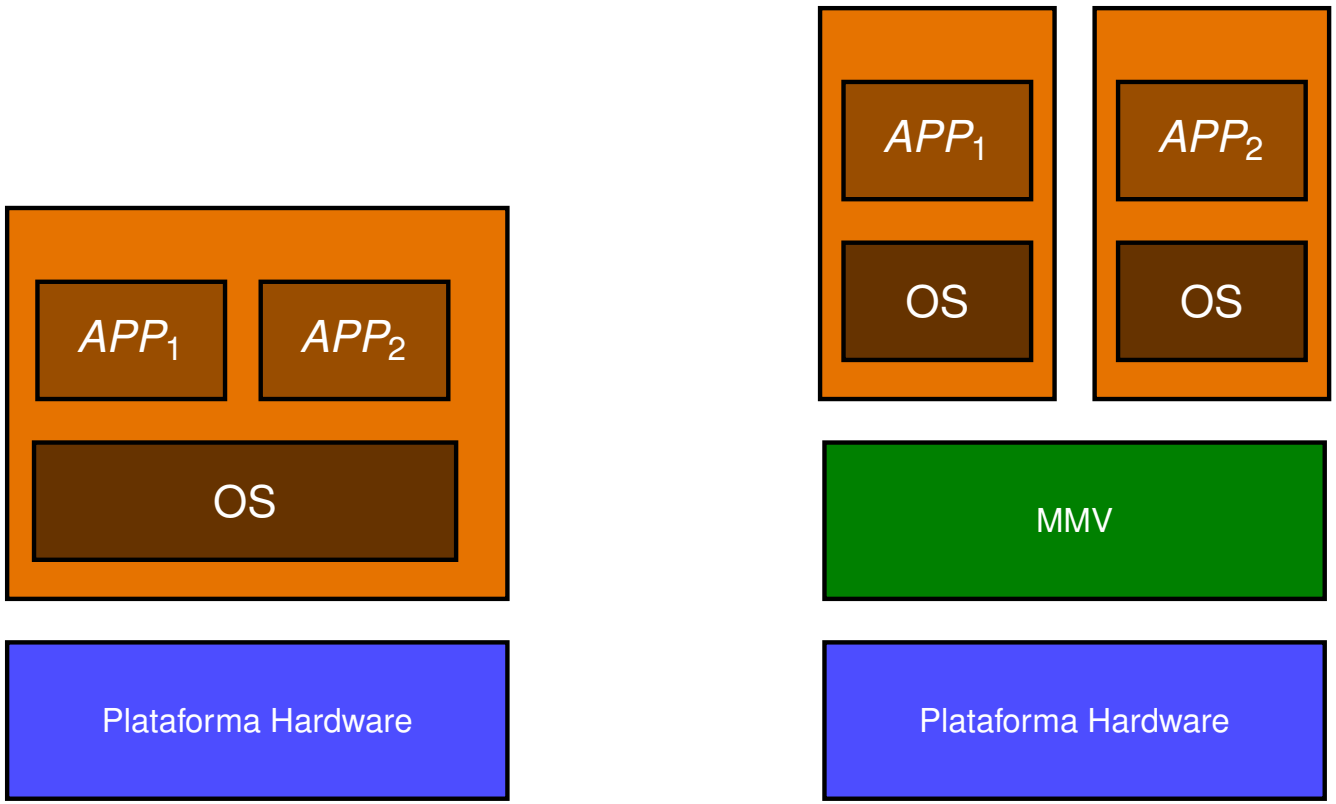
Sobrecarga de un MMV

- Depende de la carga de trabajo.
- Programas **ligados a procesador** a nivel de usuario:
  - **Ejemplo**: SPEC.
  - **Sobrecarga**: 0.
  - Raras invocaciones a SO.
- Programas **intensivos en E/S** $\rightarrow$ **intensivos en SO**.
  - Muchas llamadas al sistema $\rightarrow$ Instrucciones privilegiadas.
  - Puede tener **mucha sobrecarga** de virtualización.
- Programas **intensivos en E/S** y **ligados a E/S**.
  - Baja utilización del procesador.
  - Se puede **ocultar virtualización**.
  - **Baja sobrecarga** de virtualización.

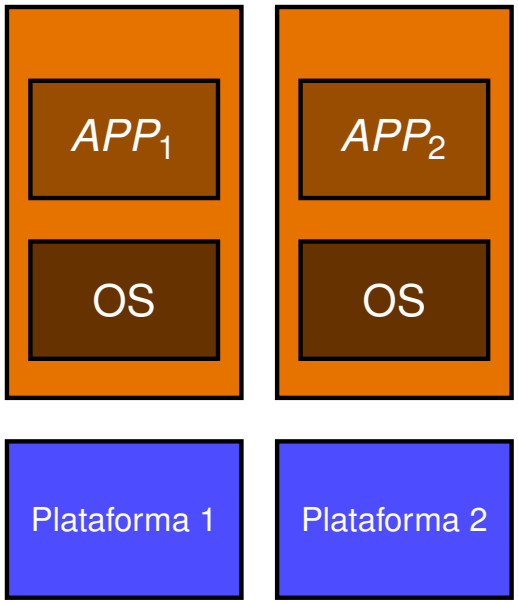
Otros usos (además de protección)

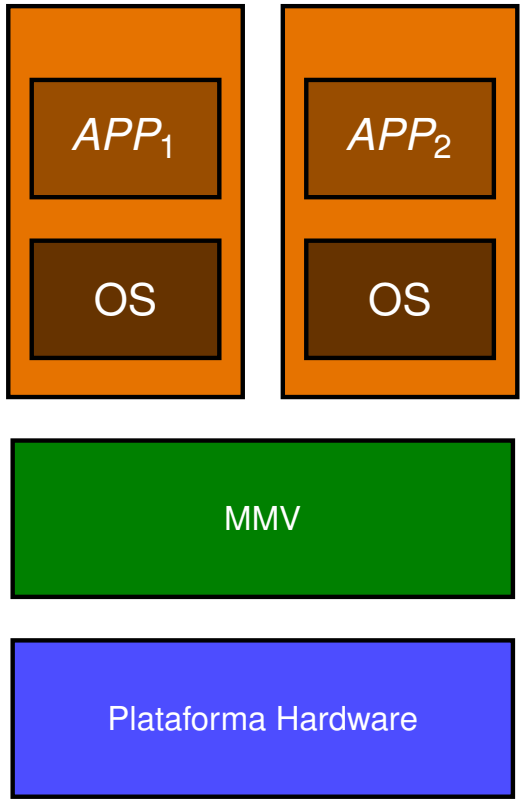
- **Gestión de software**.
  - MV ofrece una **abstracción** que permite ejecutar **pila software completa**.
    - Sistemas operativos antiguos (¿DOS?).
  - Despliegues **combinados** SO estable, SO heredado y siguiente versión de SO.
- **Gestión de hardware**.
  - MV permite ejecutar **pilas de software separadas** pero sobre un mismo hardware.
    - Consolidación de servidores.
    - Independencia $\rightarrow$ Mayor fiabilidad.
  - **Migración** de MV en ejecución.
    - Equilibrio de carga.
    - Evacuación de hardware.

Usos: aislamiento

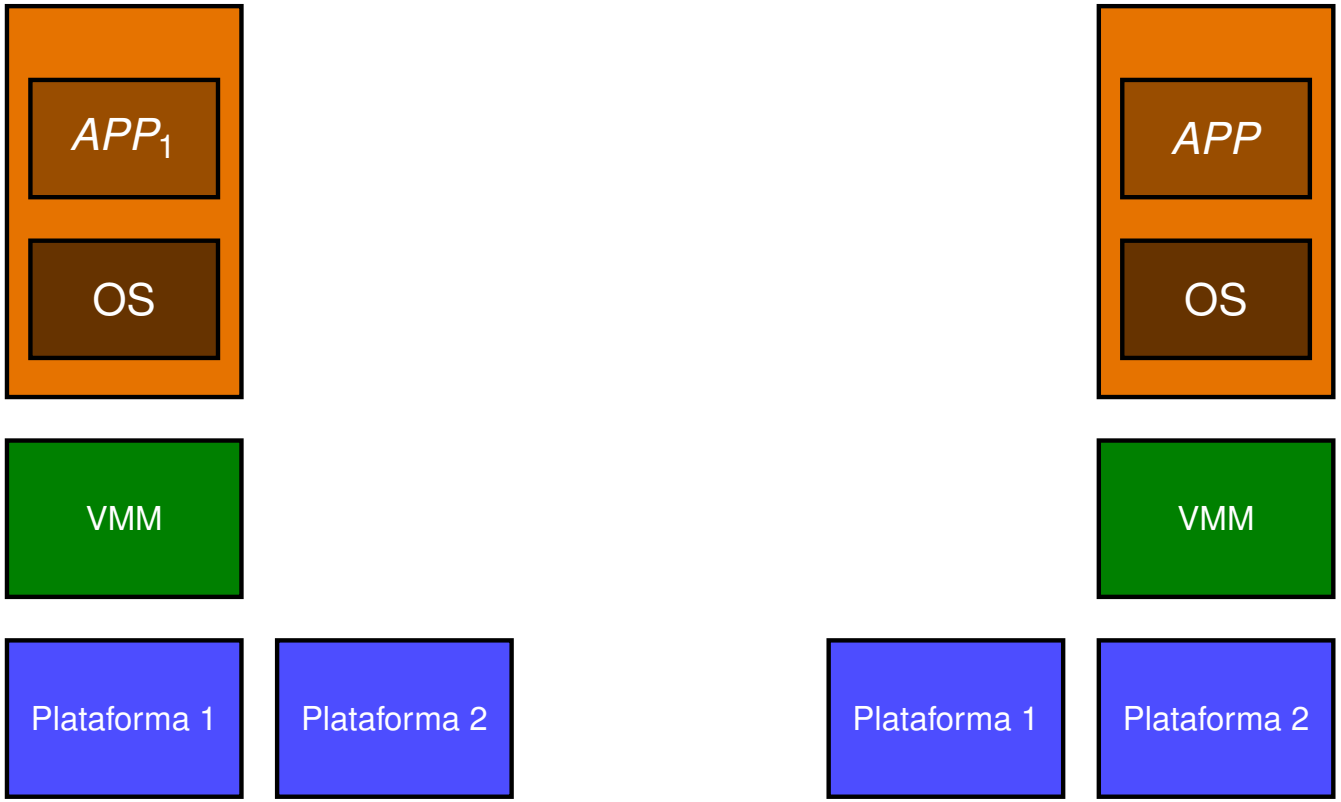


Usos: consolidación





Usos: migración



Requisitos de MMV (I)

- Una **MMV**:
  - Presenta una **interfaz** software a software huésped.
  - **Aísla** el estado de un huésped del resto.
  - Se **protege** a si mismo de los huéspedes.

- **Software huésped** se debería comportar como si no hubiese MMV, excepto por:
  - Comportamiento dependiente del rendimiento.
  - Limitaciones de recursos fijos compartidos por múltiples MMV.

Requisitos de MMV (II)

- El software huésped no debe poder modificar la asignación de recursos reales de forma directa.
- MMV debe controlarlo todo, aunque sea usado por huéspedes.
  - Acceso a estado privilegiado, Traducción de direcciones, E/S, excepciones, interrupciones, ...
- MMV debe ejecutar en un modo más privilegiado que huéspedes.
  - Ejecución de instrucciones privilegiadas por MMV.
- Requisitos de MMV (equiv. a requisitos de memoria virtual).
  - Como mínimo dos modos de procesador.
  - Subconjunto de instrucciones privilegiadas solo en modo privilegiado.
    - Trap si ejecutadas en modo usuario.

Soporte de ISA

- Si MV se tienen en cuenta en el diseño de ISA, es fácil reducir instrucciones que debe ejecutar VMM y cuanto tarda la emulación.
  - **Pero**, la mayoría de ISA para escritorio diseñadas antes de MV.
- MMV debe asegurar que huésped solo interacciona con recursos virtuales.
  - SO huésped ejecutado en modo usuario.
  - Intentos de acceder a HW da lugar a trap.
- Si ISA no es consciente de MV, el MMV debe interceptar instrucciones problemáticas.
  - Introducción de recursos virtuales.

Impacto sobre memoria virtual

- Cada huésped gestiona memoria virtual.
  - ¿Virtualización de memoria virtual?
- VMM distingue entre **memoria real** y **memoria física**.
  - **Memoria real**: Nivel intermedio entre **memoria virtual** y **memoria física**.
  - **Huésped**: Correspondencia entre **memoria virtual** y **real**.
  - **MMV**: Correspondencia entre **memoria real** y **física**.
- Para reducir los niveles de indirección MMV mantiene una **tabla de páginas en la sombra**.
  - Correspondencia entre **memoria virtual** y **física**.
  - MMV tiene que capturar cambios de **tabla de páginas** y **puntero a tabla de páginas**.

Soporte ISA para virtualización de memoria virtual

- IBM 370 incorpora nivel adicional de indirección gestionado por MMV.
  - Elimina la necesidad de tabla de páginas en la sombra.
- Virtualización de TLB.
  - MMV gestiona la TLB y mantiene copias de la TLB de cada huésped.
  - Accesos a TLB generan trap.
  - TLB con identificadores de procesos simplifican la gestión
    - Permite entradas de múltiples MV y del VMM simultáneamente.

Impacto de entrada/salida

- Parte **más compleja** de virtualización.
  - Número creciente de dispositivos de E/S.
  - Diversidad creciente de tipos de dispositivos de E/S.
  - Compartición de dispositivos entre MV.
  - Soporte de gran variedad de drivers.
- Se deja **parte general** del driver en huésped.
  - **Parte específica** en MMV.
- Método **dependiente del dispositivo**.
  - **Discos**: Particionados por MMV para crear discos virtuales.
  - **Interfaces de red**: Multiplexados en el tiempo.
    - MMV gestiona **direcciones de red virtuales**.

Virtualización impura

- Solución para arquitecturas **no virtualizables** y para reducir **problemas de rendimiento**:
- **Enfoques**:
  - **Paravirtualización**: Portar el código de SO huésped a ISA modificado.
    - Esfuerzo de desarrollo.
    - Necesidad de repetir para cada SO.
    - Disponibilidad de código fuente.
  - **Traducción binaria**: Sustituir instrucciones no virtualizables por código de emulación o llamada a MMV.
    - No requiere código fuente.
    - Algunas emulaciones posibles en espacio de usuario.

Ejemplo: XEN

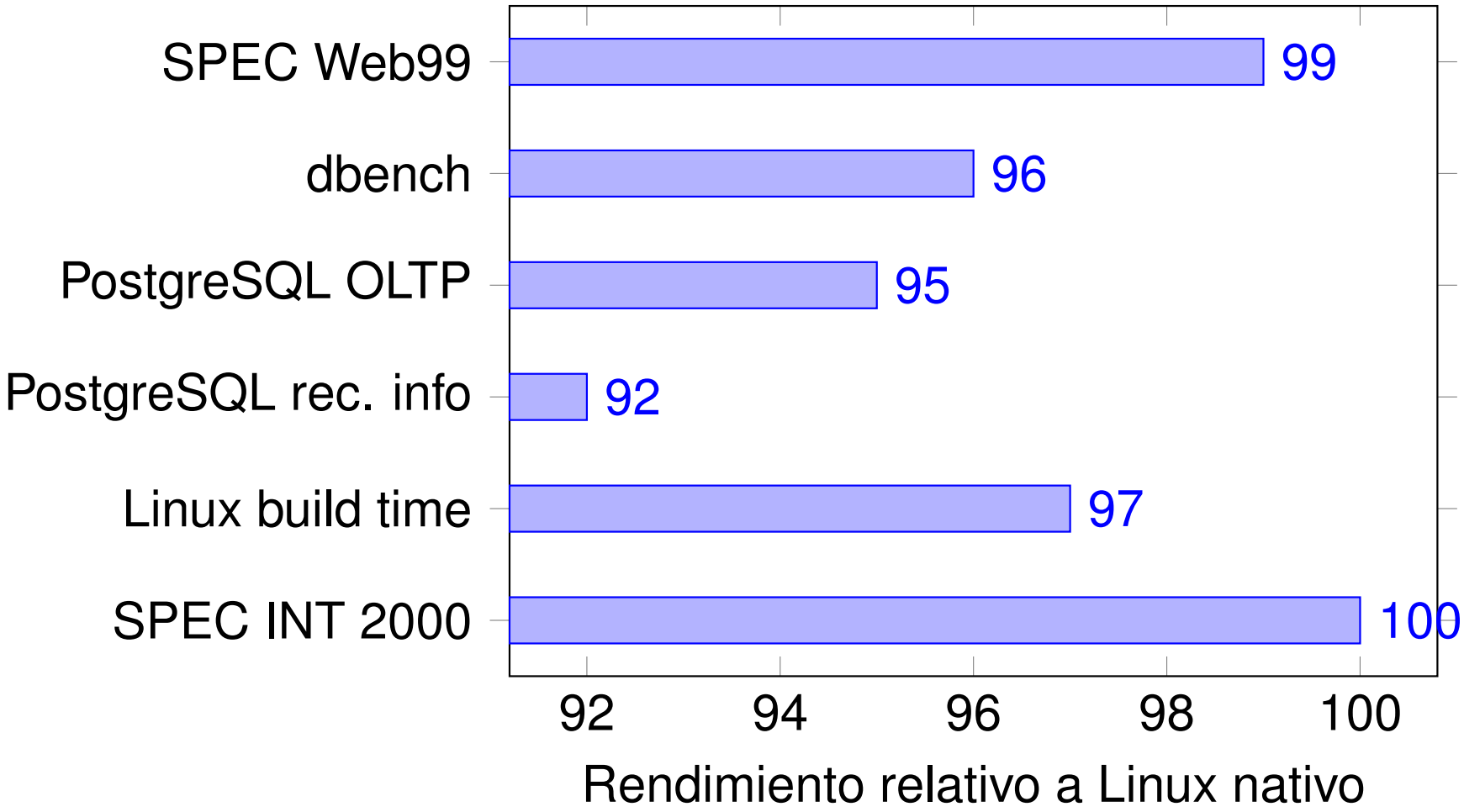
- **Xen**: MMV *open-source* para x-86.
- **Estrategia**: **Paravirtualización**.
  - Pequeñas modificaciones al SO para simplificar virtualización.
- **Ejemplos de paravirtualización**:
  - **Evitar vaciado** de TLB cuando se invoca a MMV.
    - Xen proyectado a 64 MB superiores de cada MV.
  - Se permite que el huésped **pueda asignar** páginas.
    - Se comprueba que no viole restricciones de protección.
  - **Protección** entre programas y huésped → Uso de **niveles** de x86:
    - Xen (0), Huésped (1), Programas (3).

Cambios en Xen

- Cambios necesarios en Linux $\rightarrow$ aproximadamente $3,000$ líneas de código.
  - 1% de código específico x86.

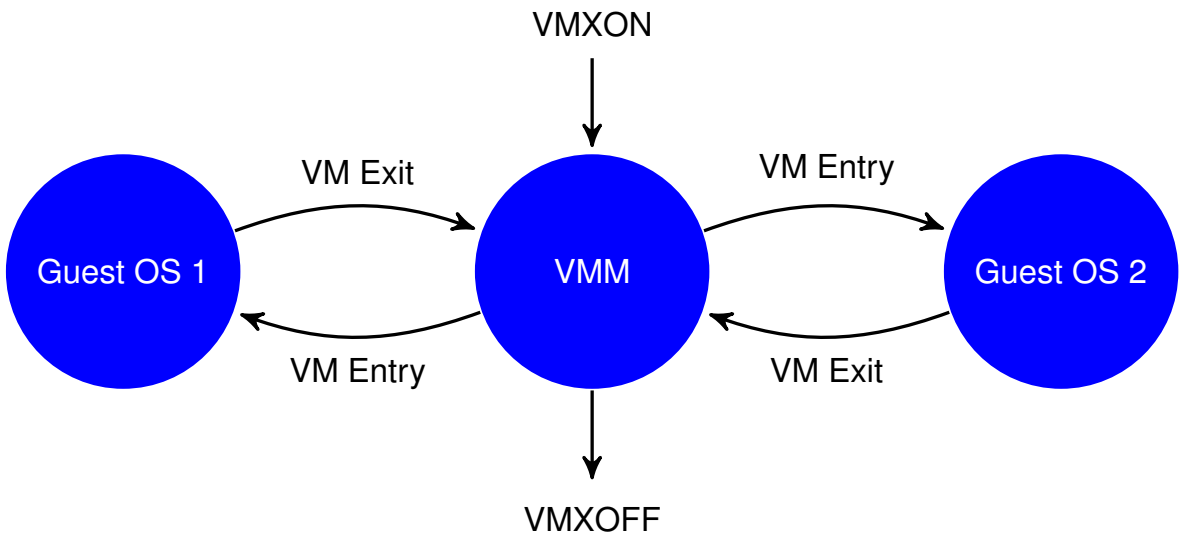
Sistema Operativo	Ejecuta como *host*	Ejecuta como *guest*
Linux 2.4	Si	Si
Linux 2.6	Si	Si
NetBSD 2.0	No	Si
NetBSD 3.0	Si	Si
Plan 9	No	Si
FreeBSD 5	No	Si

Rendimiento de Xen



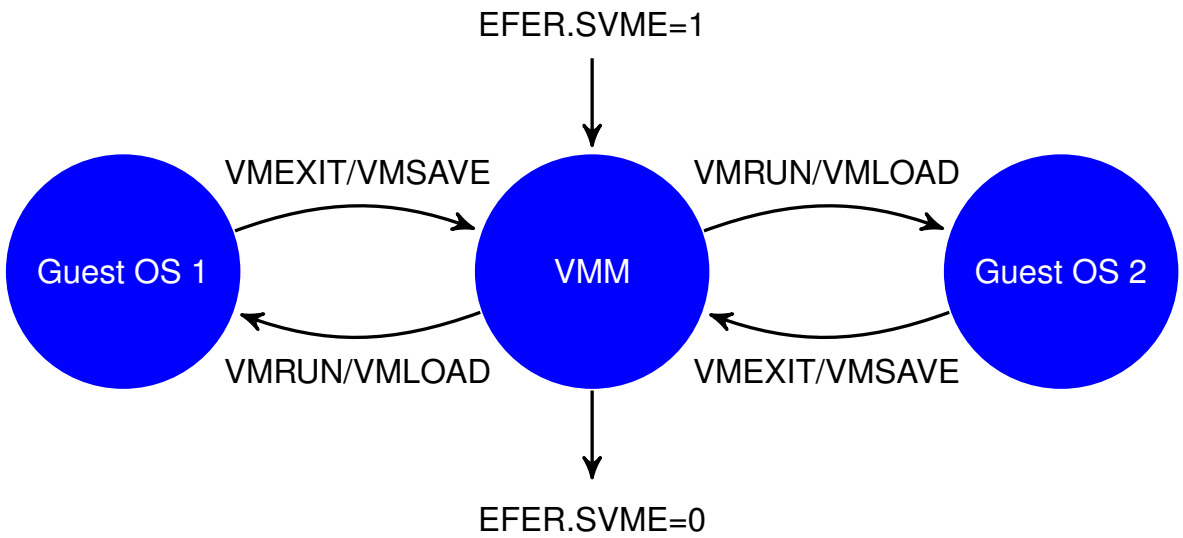
Intel Virtualization Technology

- Añade nuevas instrucciones:
  - VMXON.
  - VMXOFF.
  - VMLAUNCH.
  - VMRESUME.
  - ...



AMD Secure Virtual Machine

- Añade nuevas instrucciones:
  - VMRUN/VMLOAD.
  - VMCALL/VMSAVE.
  - ...



Modos de operación

- **VMX root**:
  - Totalmente privilegiado.
  - Pensado para su uso por MMV.

- **VMX non-root**:
  - No privilegiado.
  - Pensado para SW huésped.

Entrada y salida de máquinas virtuales

- **VM Entry**:
  - Transición de MMV a Huésped.
  - Entra en modo non-root.
  - Carga el estado del huésped.
  - **VMLAUNCH** instrucción usada en entrada inicial.
  - **VMRESUME** instrucción usada en llamadas siguientes.

- **VM Exit**:
  - **VMEXIT** instrucción usada para pasar a MMV.
  - Entra en modo root.
  - Salva el estado del huésped.
  - Carga el estado de MMV.
  - Hay instrucciones y eventos que provocan **VMEXIT**.

Beneficios de tecnología VT

- Reduce dependencia del SO.
  - Elimina necesidad de traducción binaria.
  - Facilita el soporte para SO antiguos
- Mejora de robustez
  - Elimina la necesidad de técnicas complejas
  - MMV más pequeño y simple
- Mejora de rendimiento
  - Menos transiciones a MMV

Resumen

- La memoria virtual ofrece un mecanismos de **traducción** que facilita la **protección** y la **compartición**.
- Políticas de memoria virtual:
  - Ubicación: Totalmente asociativa.
  - Identificación: Tabla de páginas.
  - Remplazo: Típicamente LRU con soporte de TLB.
  - Escritura: Siempre post-escritura.
- Máquinas virtuales: aislamiento, seguridad, fiabilidad, compartición.
- Usos de MMV: protección, gestión sw/hw (aislamiento, consolidación, migración).
- Tecnologías: Virtualización impura y soluciones en la ISA.

Referencias

- **Computer Architecture. A Quantitative Approach**
  5th Ed.
  Hennessy and Patterson.
  Secciones: B.4, 2.4.

- **Ejercicios recomendados**:
  - B.12, B.13, B.14, 2.20, 2.21, 2.22, 2.23

Virtualización y jerarquía de memoria

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid