

# Virtualización



# Sistemas Operativos

FAI - UNCOMA - 2023



# Virtualización

**CP/CMS (Control Program/  
Cambridge Monitor System)**

**IBM CP-40/CP-67  
(1966)**

**IBM LPARS  
(2001)**

**Xen  
(2004)**

**FreeBSD Jails  
(2000)**

**Docker  
(2013)**

**Intel VT (Virtualization Technology)  
(2006)**

**AMD SVM (secure virtual machine)  
(2006)**

**Qemu  
(2009)**

**KVM  
(2007)**

**VirtualBox  
(2007)**

**VMWare  
(1999)**

**Hyper-V  
Microsoft  
(2008)**

**Java VM  
(1995)**

**LXC  
(2008)**

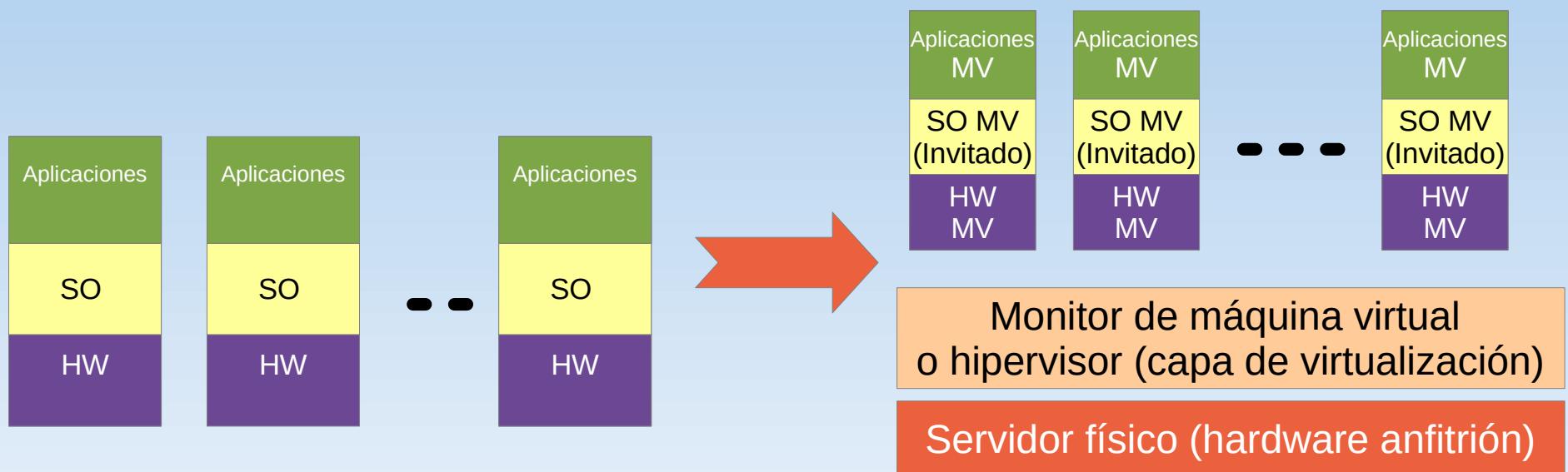


# Concepto de virtualización

- Mecanismos que implementan abstracciones de recursos computacionales físicos o lógicos.
- Un mismo **recurso base** (servidor, sistema operativo, red, etc.) puede actuar como **múltiples recursos lógicos o virtuales**.
- Una misma computadora física pueda actuar como múltiples computadoras simultáneamente → ***Virtualización de plataforma***



# Arquitectura general





# Temas a desarrollar

- ⌚ Terminología
- ⌚ Hipervisores
- ⌚ Virtualización de plataforma
  - Virtualización asistida por hardware → CPU
  - Virtualización completa (mv)
  - Emulación (mv)
  - Paravirtualización (mv)
  - Virtualización de sistema operativo (container)
- ⌚ Aplicaciones
  - ⌚ Consolidación de servidores → GreenIT
  - ⌚ Virtual/software appliance
  - ⌚ Caso de estudio: *docker*
  - ⌚ Caso de estudio: *kvm/qemu/libvirt/virt-manager*



# Terminología

- ◆ Abstracción de todos los recursos de computación de un **guest (huésped)** dentro de un **host (anfitrión)**.
- ◆ El sistema operativo nativo que ejecuta el software de virtualización es llamado el **anfitrión**. Es quién controla el hardware.
- ◆ El sistema operativo virtualizado es llamado **huésped**.
- ◆ Pueden coexistir muchos **huéspedes** en un mismo **anfitrión**; la clave está en que **no interfieren unos con otros**.



# Hipervisor:

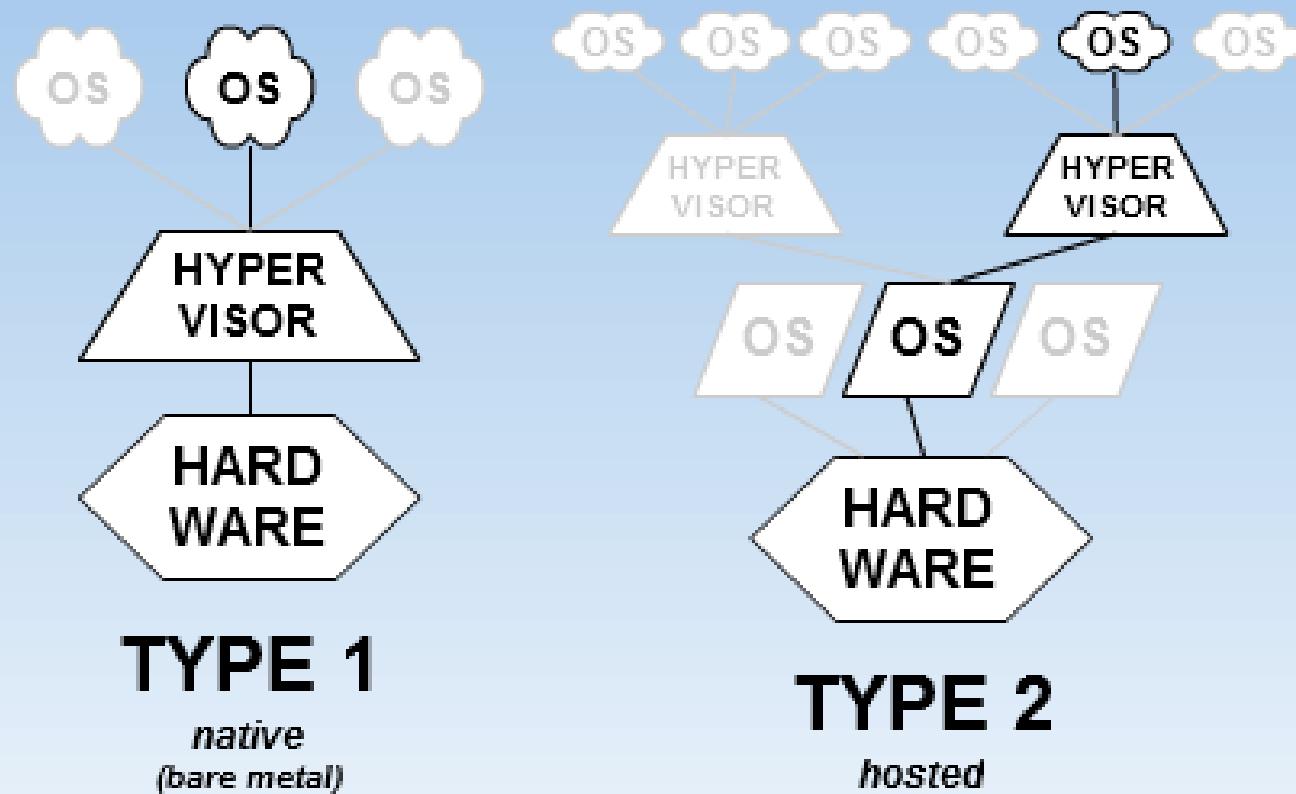
**Hipervisor**: o monitor de máquina virtual. Capa de software que permite utilizar **simultáneamente**, diferentes sistemas operativos en una misma computadora.

**Tipo 1 (nativo o “bare metal”)**: el hipervisor se ejecuta directamente sobre el hardware en modo kernel, es en sí mismo el SO *host* y administra a su vez los *guest*. Ej: Xen, VMware ESX/ESXi.

**Tipo 2 (hosted)**: el hipervisor es software que se ejecuta **sobre el SO** que controla el hardware. Las instrucciones privilegiadas nunca se ejecutan en el hardware real, son emuladas por el hipervisor cada vez que ocurren en el SO huésped. Ej. qemu, VMWare Workstation, VirtualBox, ¿KVM?.



# Tipos de hipervisor:



Robert Popek. y Goldberg (1974)

# Propiedades del hipervisor

- ⌚ **Fidelidad:** el ambiente creado para la MV es en esencia idéntico al de la máquina física, anfitrión. Los programas no deberían ser capaces de distinguir si están ejecutándose en un entorno virtualizado o no. ← *Modificado en los conceptos más actuales.*
- ⌚ **Aislación/seguridad:** el hipervisor debe tener control completo de los recursos del sistema.
- ⌚ **Rendimiento/Eficiencia:** poco o ninguna diferencia en rendimiento entre una MV y su equivalente físico.



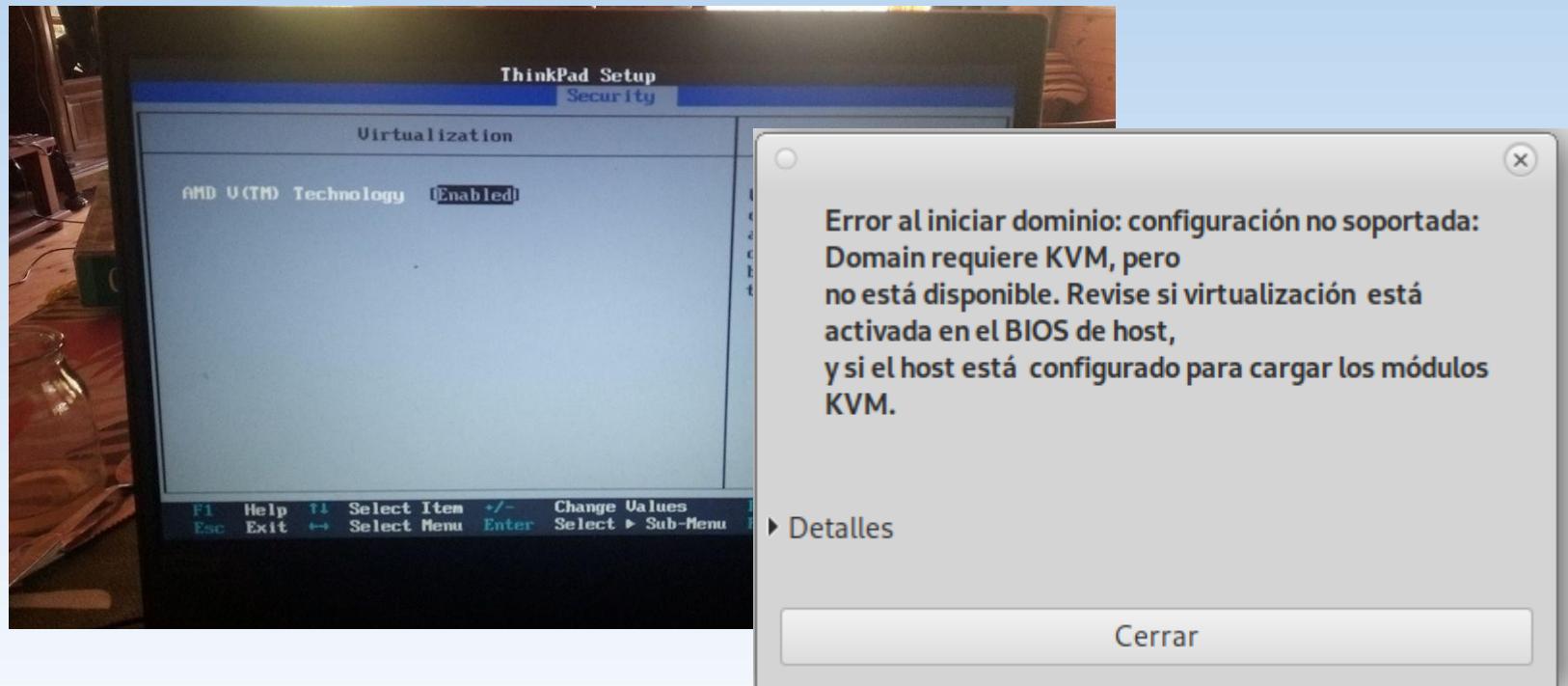
# Virtualización asistida por hardware

- Apoyo provisto por el hardware para virtualización en **arquitecturas que no fueron diseñadas con este propósito** (PC)
- Alrededor del año 2000 re-surge (mainframe ~1970) en servidores, aún muy costoso para el usuario promedio.
- X86 → Un SO por computadora.
- ~ 2006 se introduce en la arquitectura de PC (x86): AMD, VIA, ARM, INTEL (entre otros, pero no todos)
- **El objetivo es la mejora en el desempeño.**



# Virtualización asistida por hardware

- ⌚ ¿Cómo puedo observar esto en el sistema operativo?
- ⌚ ¿El firmware de la placa base (UEFI/BIOS) se ve involucrado?
- ⌚ El hipervisor será quien hace uso de la característica.





# Virtualización asistida por hardware

Dentro del huésped...

```
# cat /proc/cpuinfo
processor : 0
vendor_id : AuthenticAMD
cpu family      : 21
model          : 2
model name     : AMD Opteron 63xx class CPU
[... texto suprimido ...]
wp      : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall
nx pdpe1gb lm art rep_good nopl extd_apicid pni pclmulqdq
ssse3 fma cx16 sse4_1 sse4_2 x2apic popcnt aes xsave avx f16c
hypervisor lahf_lm abm sse4a misalignsse 3dnowprefetch xop
fma4 tbm arat
[... texto suprimido ...]
```

En el anfitrión ...

```
$ cat /proc/cpuinfo
processor : 0
vendor_id : AuthenticAMD
cpu family      : 21
model          : 16
model name     : AMD A8-5600K APU with Radeon(tm) HD Graphics
[... texto suprimido ...]
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht
syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc
rep_good nopl nonstop_tsc cpuid extd_apicid aperfmpref pni
pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 popcnt aes
xsave avx f16c lahf_lm cmp_legacy svm extapic cr8_legacy abm
sse4a misalignsse 3dnowprefetch osvw ibs xop skinit wdt lwp
fma4 tce nodeid_msrs tbm topoext perfctr_core perfctr_nb cpb
hw_pstate vmmcall bmi1 arat npt lbrv svm_lock nrip_save
tsc_scale vmcb_clean flushbyasid decodeassists pausefilter
pfthreshold    UNCOMMA - FAI - 2024
[... texto suprimido ...]
```

PD: virt-what



# Emulación

- ❖ Reproducción, mediante software, de la conducta de **todo el hardware**. Ejemplo : **QEMU**
- ❖ **Ventajas** : la máquina completamente emulada permite ejecutar cualquier sistema operativo sin modificaciones para cualquier arquitectura emulada.
- ❖ **Limitación**: Es necesario escribir emuladores para cada dispositivo.
- ❖ **Mayor fidelidad y transparencia**
- ❖ **Performance limitada** (emula todo por software).
- ❖ ¿Usos?





# Paravirtualización

- ⌚ Se **modifica el SO huésped** que ejecuta “hypervisor calls”.
- ⌚ El hipervisor define la API disponible para un SO huésped.
- ⌚ Las **aplicaciones no sufren modificaciones**.
- ⌚ Usa modelo **split drivers** (drivers divididos).



# Paravirtualización

**Ventaja principal:** gran performance lograda frente a la emulación o virtualización completa.

**Desventaja principal:** modificar o instrumentar el código, tanto del sistema operativo anfitrión como de los huéspedes.

## Limitaciones:

- ❖ SO huésped **no puede** ejecutarse directamente sobre el hardware (bare-metal).
- ❖ La variedad de hipervisores y la falta de una API estándar complejiza el desarrollo de SO huéspedes modificados a tal fin.



# Virtualización de Sistema Operativo

## ▣ Aka Containers

▣ **El Kernel permite múltiples ambientes de usuarios aislados entre sí.**

▣ Ejemplos:

- LXC → Canonical (Ubuntu)
- FreeBSD Jails (FreeBSD)
- Solaris Zones (Solaris → Oracle)
- OpenVZ/Virtuozzo
- Systemd-nspawn (Systemd)
- Docker



# Virtualización de Sistema Operativo

- ❖ Un **container** podríamos hacer ssh, obtener un shell; tiene su propio conjunto de procesos y usuarios; instalar paquetes; ejecutar cosas como root, etc. También podríamos no tener todo eso...
- ❖ **Comparte el kernel con el anfitrión**
- ❖ No requiere de un init con PID 1; no requiere de syslog, cron, etc.
- ❖ En Linux hay soporte en el kernel: [man namespaces](#)
- ❖ Pueden contener **sistemas operativos distintos**
- ❖ **Arquitectura** determinada por el anfitrión.
- ❖ **Consumo de recursos** suele ser muy inferior a una mv completa.
- ❖ **Velocidad de inicio** mayor que la de una mv completa.



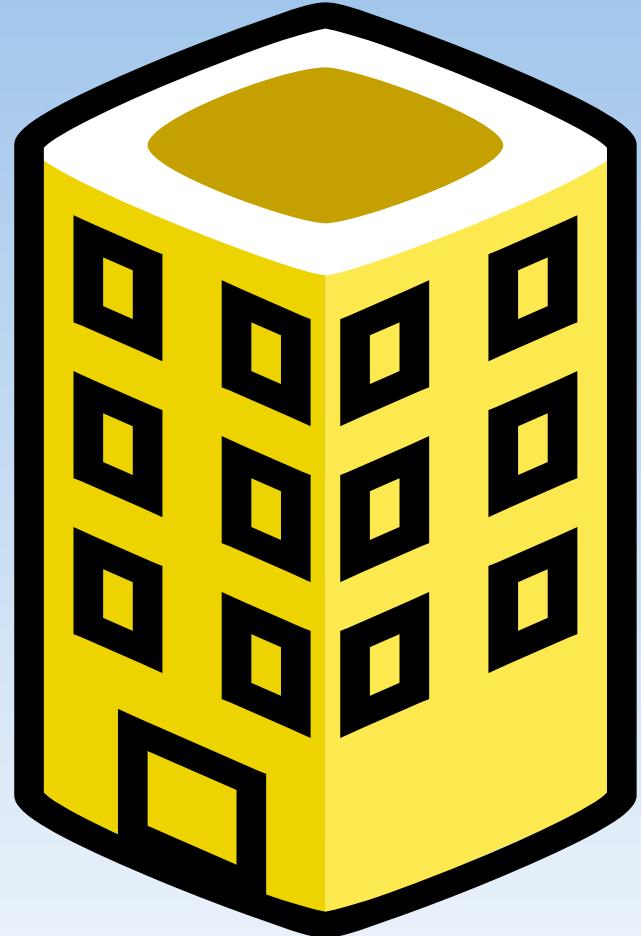
# Virtualización de Sistema Operativo

- ❖ Objetivo es ejecutar **alguna/s aplicación/es con todas sus dependencias** y necesidades PERO NADA MAS...
- ❖ La aplicación se abstrae del host.
- ❖ El container se conecta al host a través de una interfaz estandarizada.
- ❖ La aplicación no debe preocuparse por las particularidades del host, lo necesario para su funcionamiento lo provee el container en sí mismo.
- ❖ Requieren de mayor entrenamiento:
  - Seguridad del container
  - Delegar hardware
  - Dependencias de software

# VMs vs containers



VM



Containers



# Aplicaciones de la virtualización

## Para el usuario final:

- ❖ No requiere mayor capacitación (no es el caso para container).
- ❖ Muy útil para crear entornos de desarrollo (dev,QA,PROD, etc.)
- ❖ Transparencia
- ❖ Crear entornos de prueba para aplicaciones o sistemas operativos.

## Para el administrador de sistemas:

- ❖ Aislación de aplicaciones.
- ❖ Coexistencia de aplicaciones conflictivas entre sí en un mismo hardware.
- ❖ Confiabilidad (la mayoría de las fallas son debidas a errores en el software).
- ❖ Soporte para aplicaciones antiguas (legacy).



# Aplicaciones de virtualización

## Para el administrador de sistemas:

- Muy útil para crear entornos de desarrollo (dev,QA,PROD, etc.).
- Permite aprovechar de manera simple recursos subutilizados.
- Reducción de costos, consumo eléctrico y espacio.
- Seguridad (OJO con el anfitrión).
- Permite balance de carga, disponibilidad y escalabilidad → cloud.
- Migración de máquinas virtuales, es más sencillo que migrar proceso individuales.
- Ciclo de vida de la VM menos trabajoso que un servidor físico.



# Consolidación de Servidores

La consolidación de servidores es un enfoque para el uso eficiente de los recursos de los servidores informáticos con el fin de **reducir el número total de servidores físicos** que una organización requiere.

La práctica se desarrollada en respuesta al problema de la proliferación de servidores, una situación en la que **múltiples servidores subutilizados** ocupan más espacio y consume más recursos del que pueden ser justificados por su carga de trabajo.



# Virtual appliance / Software appliance

**Virtual appliance:** imagen de una máquina virtual diseñada para ser ejecutada por una plataforma de virtualización (ej: VirtualBox, Xen, qemu, etc).

**Software appliance:** es una virtual appliance con una dada aplicación diseñada para correr dentro de dicha máquina virtual. Es un **modo alternativo de proveer software** eliminando la necesidad de instalación. Configuración, resolución de dependencias de bibliotecas, etc. Reduce el costo involucrado en mantener aplicaciones con dependencias complejas.

**Los containers son la tendencia en este sentido.**

Turnkey Linux, Docker Hub...

# Algunas conclusiones

- ★ La virtualización es una solución a muchos problemas de administración.
  - ★ Hay poca estandarización de las implementaciones.
  - ★ Cada solución requiere un análisis antes de proceder a su implementación.
- Algunos puntos a considerar:
- Tipo de virtualización provista (emulación, container, etc.)
  - Dependencia del hardware (arquitectura de CPU).
  - Soporte para diversos sistemas operativos.
  - Estandarización de la solución.
  - Comunidad de soporte y de usuarios.
  - Historia del producto y cuan afianzado está su desarrollo.
  - Tendencia de uso del producto.
  - Licencia
- ★ Se requiere entrenamiento.
  - ★ Es el ladrillo constructivo de “la nube”