15.03.2018

dit
UPM

Centro de Datos y Provisión de Servicios
Tema 2:

Virtualización

Alejandro Alonso

Departamento de Ingeniería de Sistemas Telemáticos
http://moodle.dit.upm.es

ETSIT UPM

Objetivos del tema

dit
UPM

- **Describir los esquemas de virtualización en Linux:**
  - Basado en kvm
  - Virtualización ligera: lxc, lxd
- **Operaciones básicas con máquinas virtuales**
- **Representación en Linux de máquinas virtuales**
- **Gestión de máquinas virtuales**

Contenidos

dit UPM

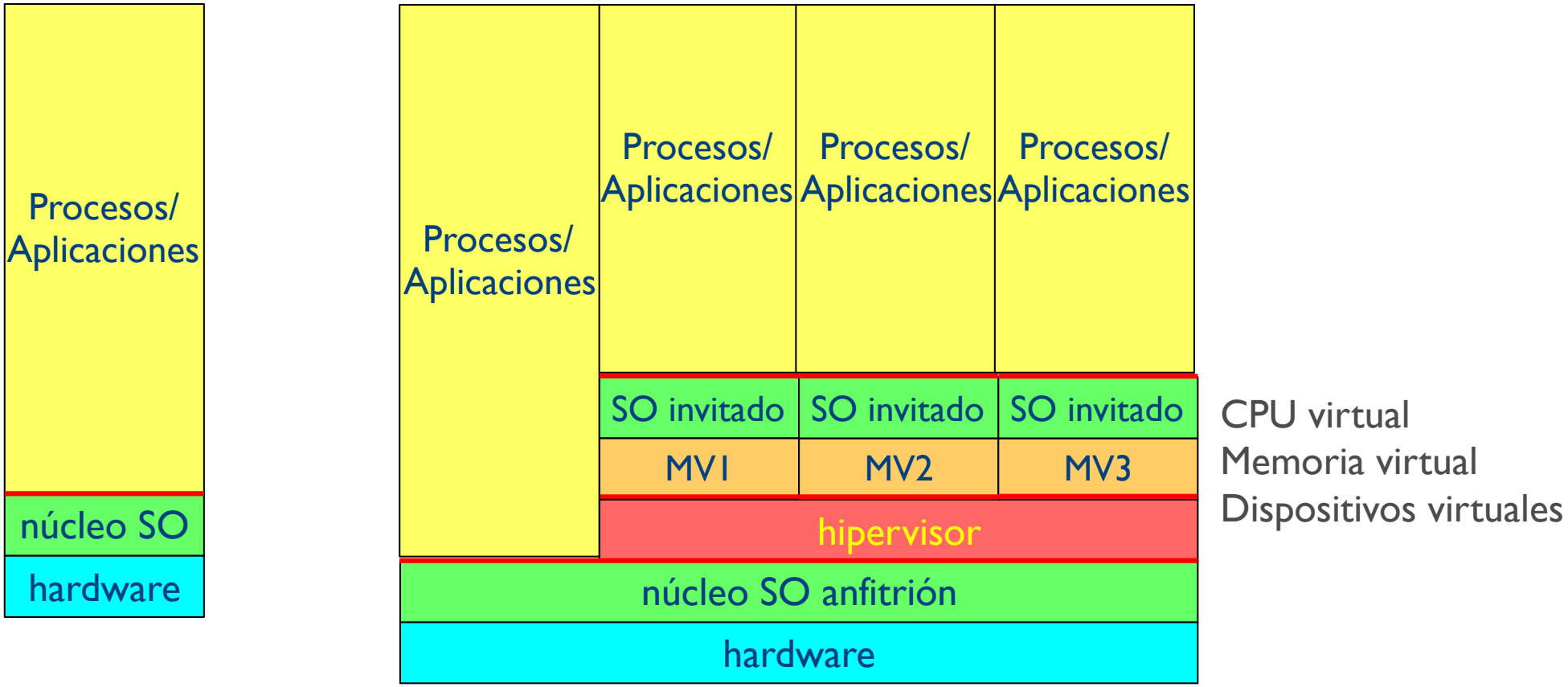
3

1. Introducción a las máquinas virtuales

dit UPM

- **Máquina virtual (*virtual machine*, VM)**
  - Software que ejecuta programas como si fuera la máquina física
- **Se abstrae el hw y se representa mediante una capa de sw**
  - Se proporciona una interfaz idéntica al hardware
- **Apariencia de varios procesadores idénticos**
- **Anfitrión (*host*): el hardware real**
  - posiblemente con un SO nativo
- **Invitado (*guest*): el SO que se instala en la máquina virtual**
  - puede haber varios SO invitados, cada uno en una máquina virtual

Máquinas virtuales



Capa de virtualización o hipervisor

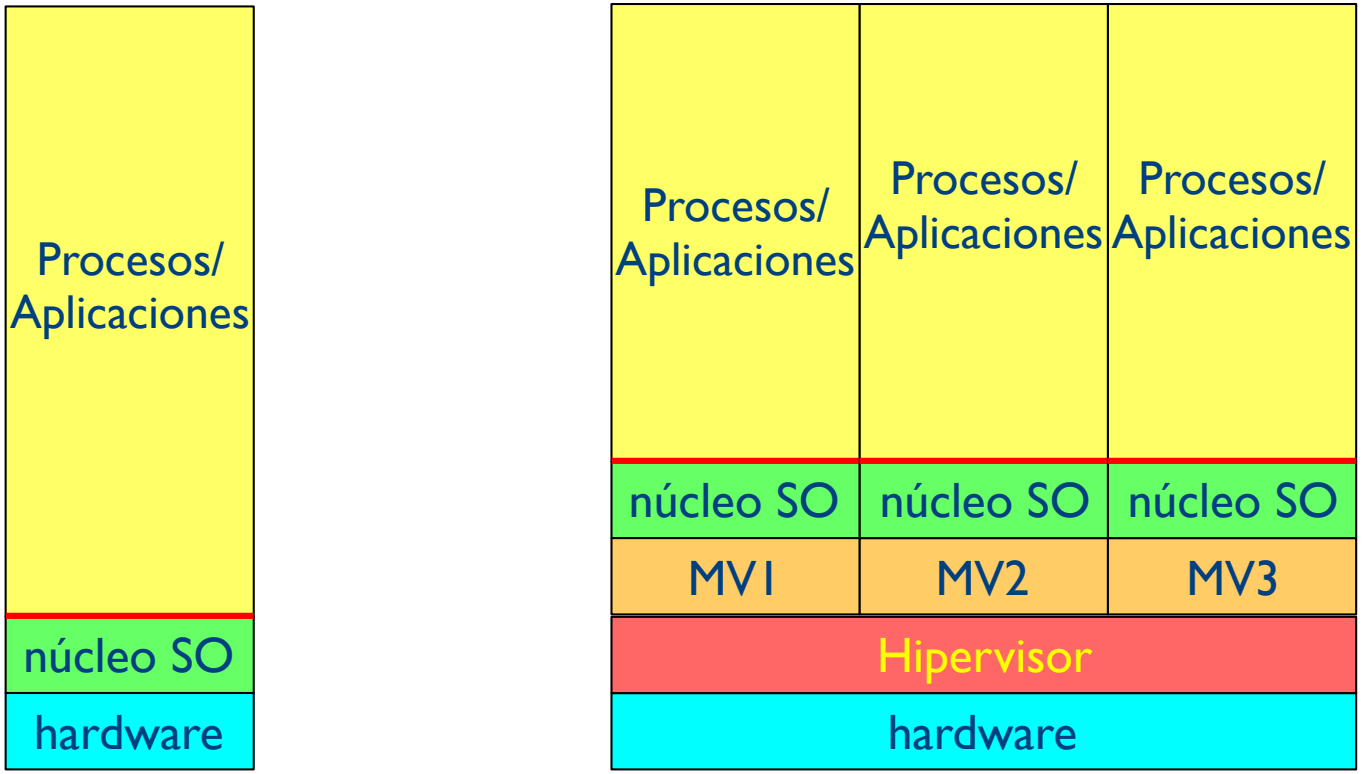
Ventajas

dit
UPM

- **Varios SO en el mismo hardware**
  - posiblemente con distintos SO
- **El sistema anfitrión está protegido contra fallos en las VM**
  - y las VM están protegidas unas frente a otras
  - Aislamiento de los recursos de los invitados
- **Consolidación de servidores**
  - una sola máquina potente contiene varios servidores virtuales
- **Distribución de aplicaciones/servidores junto con SO**
  - Replicar en diferentes computadores

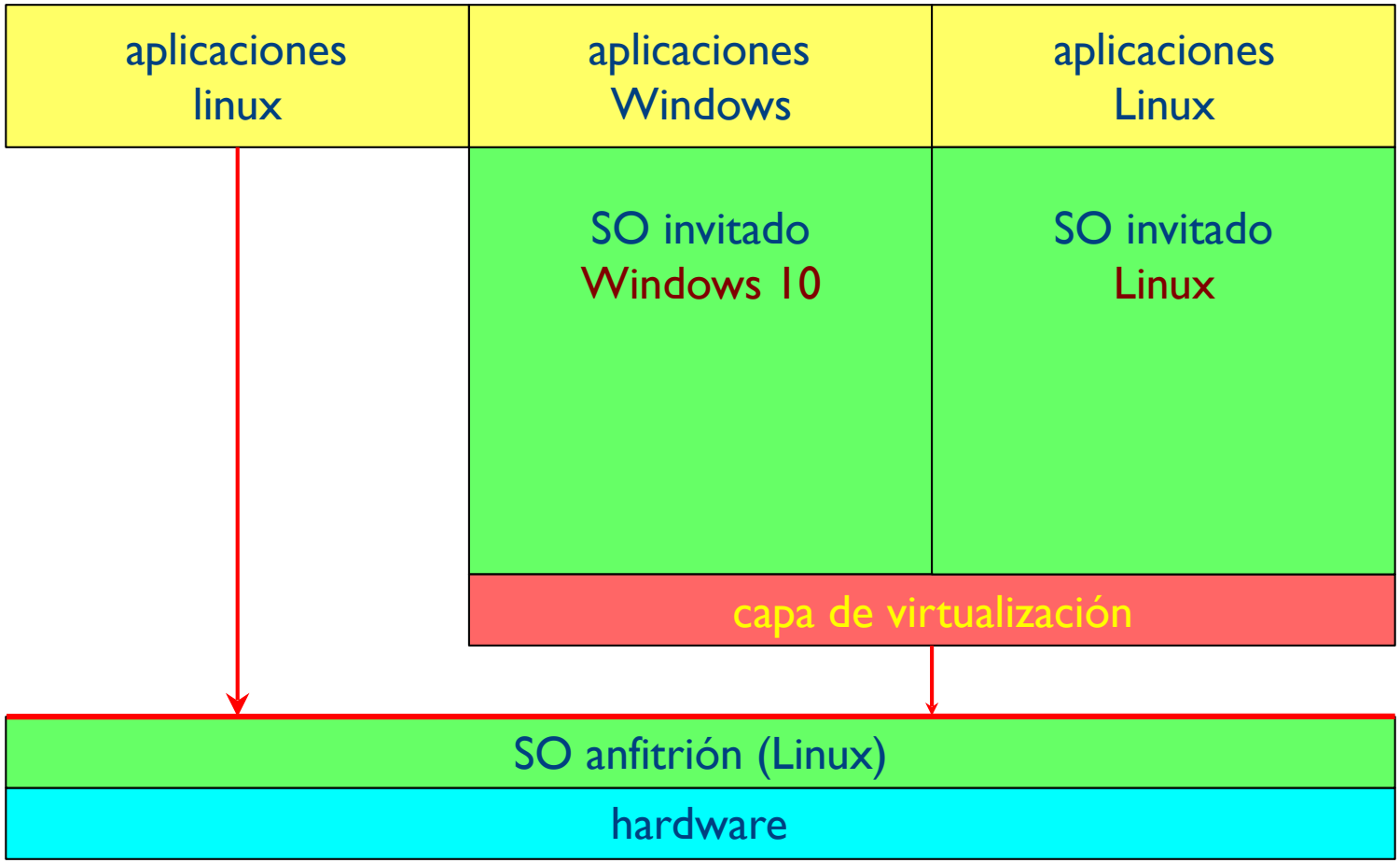
Tipo 1: Hipervisor nativo o *bare-metal*

- **El hipervisor ejecuta sobre el hardware**
  - No se ejecuta un SO clásico sobre el hardware



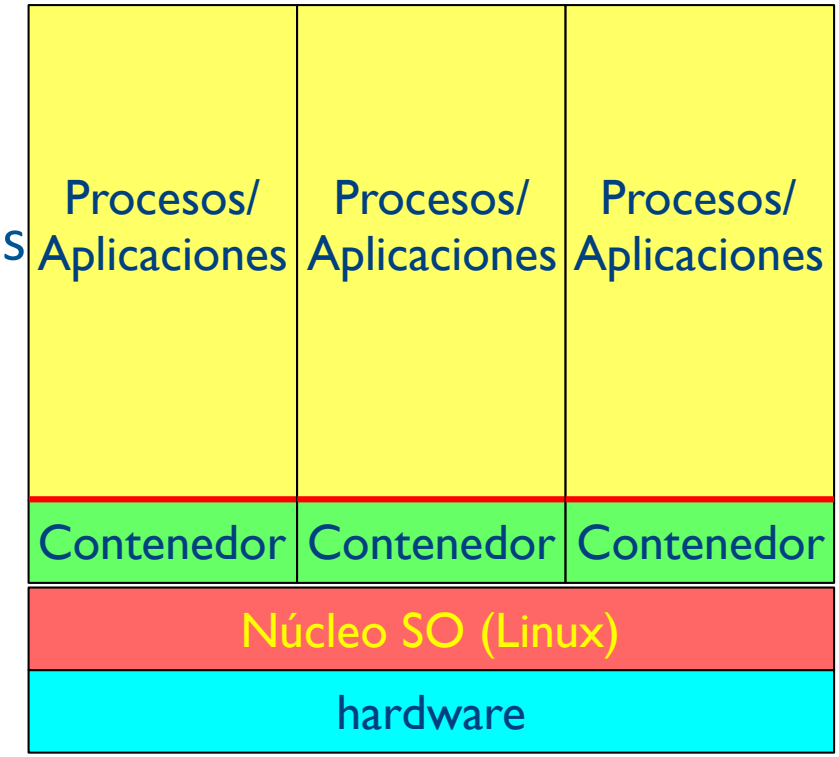
Tipo 2: Hipervisor sobre el anfitrión

- **El hipervisor ejecuta sobre un SO**



Virtualización ligera: Contenedor de Linux

- **No tiene hipervisor**
  - Se apoya sobre el SO
  - Menos sobrecarga y recursos necesarios
  - Mantiene el aislamiento de recursos
  - No soporta otros SO
- **Contenedor:**
  - Encapsula los recursos de los procesos de sus aplicaciones
  - Permite integrar varias aplicaciones



Virtualización ligera: Contenedor de Linux

dit UPM

- **LXC: *Linux containers***
  - Aíslan los procesos y recursos de una MV
  - *Namespaces*: representar los recursos de un proceso(s)
  - *cgroups*: asignar y aislar recursos de un conjunto de procesos
- **LXD: Proporciona un API REST sobre LXC**
- **Características:**
  - Sobrecarga de virtualización muy pequeña
  - No hay emulación de hardware, ni de SO
- ***docker*: mayor nivel de abstracción que lxc**
  - Funciones similares, pero más fácil de utilizar
  - Facilita el despliegue

Técnicas relacionadas

dit
UPM

- **Emulación**
  - se simula un procesador distinto del anfitrión
  - muy costoso en tiempo de ejecución
    - hay que interpretar todas las instrucciones de máquina del procesador invitado
  - útil para ejecutar software antiguo
- **Para-virtualización**
  - la plataforma de máquina virtual presenta una interfaz similar, pero no idéntica, a la arquitectura de máquina original
  - el software invitado debe modificarse para ejecutarse en la máquina virtual

Arrancar un computador

dit UPM

- **Se requiere un disco o partición de arranque, que =ene**
  - El código del SO
  - Ficheros de configuración del SO
  - El sistema de fichero de raíz del SO,
  - incluyendo programas y ficheros del sistema y usuarios
- **Carga del núcleo (boot)**
  - cargador inicial, arranca por hardware: bootstrap loader
    - en ROM/EPROM (firmware): comprobación de dispositivos básicos
    - Cargador general desde un bloque de disco predeterminado (GRUB)
    - Se selecciona una partición de arranque
    - Finalmente, se carga el núcleo del SO en memoria
- **El cargador permite elegir un SO entre los disponibles en los discos del computador**

Imagen de Máquina Virtual

dit UPM

- **Es un disco o partición virtual en un fichero**
- **Es una plantilla que se usa para crear instancias de MV**
  - Las instancias pueden ejecutar (LXD)
  - En algunos hipervisores (KVM), la imagen es ejecutable
- **Se convierte en la raíz de la MV**
- **Proceso e Información asociada a una imagen de MV**
  - Crear el fichero
  - Crear el sistemas de fichero
  - Instalar el sistemas operativo y aplicaciones necesarias
  - Configuración el hardware virtual de la MV: CPU, memoria, red, dispositivos de E/S, a veces en ficheros externos a la imagen

Generación de una MV

dit UPM

- **Crear una máquina virtual:**
  - Generar un contenedor a partir de una imagen (LXD)
  - Clonar una imagen a partir de una imagen ejecutable (KVM)
- **La generación de una imagen a partir de una creada:**
  - Duplicación de la imagen original
  - El fichero de la imagen se duplica virtualmente (copy-on-write)
- **Configurar la instancia: personalizar la imagen de base**
  - Puede ser necesario para diferenciar al clonar
  - Se puede generar una configuración automáticamente
- **Arrancar la MV**
  - Instalar y configurar los programas necesarios
  - Ejecutar aplicaciones

Ejemplo: Creación de Servidores

dit UPM

- **Suponer que hay que crear *n* servidores y ejecutar en diversos anfitriones**
- **Crear una imagen, a partir de un sistema operativo**
  - Definir una configuración de hw inicial
- **Configurar el sw de la imagen:**
  - Arrancar una MV de la imagen inicial
  - Instalar y configuración las aplicaciones necesarias
- **Cada vez que se quiere crear un servidor**
  - Seleccionar el computador donde se quiere instalar y arrancar
  - Generar una imagen a partir de la previa
  - Configuración el hw, de acuerdo del anfitrión
  - Descargar y arrancar la imagen. Si necesario, adaptar el SW

2. Contenedores en Linux: LXD

dit UPM

- **LXD es un demonio en red que proporciona un API REST sobre LXC, que es más sencillo**
- **Los componentes de LXD**
  - Imágenes
  - Contenedores
  - Instantáneas (Snapshot)
    - ✓Similar a un contenedor, pero inmutable. No se puede cambiar
  - Perfiles (Profiles)
  - Remoto
    - Se pueden ejecutar órdenes de demonios remotos
  - Seguridad
    - Mecanismos para disponer contenedores seguros

Imágenes de LXD

dit UPM

- **Imágenes**
  - Es una plantilla para crear contenedores (ejecutarles)
  - Imágenes predefinidas para publicar y descargar:
    - Imágenes únicas identificadas firmadas (sh256)
  - LXD ofrece tres imágenes por defecto
    - **ubuntu:** es la imagen estable y aconsejada en esta asignatura
- **API de LXD**
  - Crear y ejecutar un contenedor de una imagen:
    $ lxc launch ubuntu: <contenedor>
  - Listar información de contenedores: $ lxc list --fast
  - Se pueden almacenar localmente:
    $ lxc image copy ubuntu: local: --alias ubuntu
  - Listar imágenes locales: $ lxc image list

Imágenes de LXD: API

dit UPM

- **Crear y ejecutar un contenedor de una imagen:**
  - $ lxc launch ubuntu: <contenedor>
  - Por defecto, genera un MAC y una dirección de IP
  - Inicialmente, se hereda la configuración de la imagen
- **Crear un contenedor, sin arrancar:**
  - $ lxc init ubuntu: <contenedor>
- **Listar información de contenedores:**
  - $ lxc list --fast
- **Se pueden almacenar localmente:**
  - $ lxc image copy ubuntu: local: --alias ubuntu
- **Listar imágenes locales:** $ lxc image list
- **Los contenedores se almacenan en** /var/lib/lxd/

Contenedores de LXD

dit UPM

- **Contenedor: MV ligera**
  - Es una instancia de una imagen
- **Contenedores: están compuestas por**
  - Un sistema de fichero (rootfs)
  - Una lista de configuraciones: límites de recursos, entorno, opciones de seguridad…
  - Un contenedor heredan configuraciones de perfiles
  - Estado de ejecución
  - Otras propiedades
- **Configuración de los LXD:**
  - Inicialmente, se configura de perfiles
  - Se puede configurar cada contenedor

Contenedor: API de LXD

dit UPM

- **Crear y crear un contenedor de una imagen:**
  - $ lxc launch ubuntu: <contenedor>
- **Listar contenedores:** $ lxc list (opcional --fast)
- **Información de un contenedor** $ lxc info <contenedor>
- **Arrancar un contenedor:** $ lxc start <contenedor>
- **Parar un contenedor:** $ lxc stop <contenedor>
  - Si no para: $ lxc stop <contenedor> --force
- **Pausar un contenedor:** $ lxc pause <contenedor>
- **Arrancar un contenedor en pausa:**
  $ lxc restart <contenedor>

Contenedor: API de LXD

dit UPM

- **Acceder a un contenedor para ejecutar órdenes**
  - Crear y conectar a un contenedor
    Es importante saber si se ejecuta del anfitrión o del invitado
    - $ lxc exec <contenedor> bash
  - Ejecutar una orden de un contenedor
    - $ lxc exec <contenedor> -- ls -lsa /
  - Cargar un fichero en un contenedor(remoto)
    - $ lxc file push <origen> <contenedor>/<camino>
  - Descargar un fichero (remoto)
    - $ lxc file pull <contenedor>/<camino> <destino>
  - Editar un fichero (remoto)
    - $ lxc file edit <contenedor>/<camino>/<fichero>

Perfiles de LXD

dit UPM

- **Definir los parámetros de configuración de contenedor**
  - Es general y se pueden aplicar en varios contenedores
- **Un contenedor puede ejecutar varios perfiles secuencialmente**
- **LXD proporciona dos perfiles configurados:**
  - *default* y *docker*
- **Ejemplos de operaciones generales:**
  - $ lxc profile list
  - $ lxc profile show <perfil>
  - $lxc profile edit <perfil>
- **Aplicar perfiles a un contenedor**
  - $ lxc profile apply <contenedor> <profile1> <profile2>…

Configuración de contenedor

- **Se pueden hacer operaciones de un perfil para un contenedor:**
  - $ lxc config <perfil>
  - $ lxc config set <contenedor> <set> <key>
  - $ lxc config show <contenedor>
  - $ lxc config show --expanded <contenedor>
- **Estas operaciones se aplican inmediatamente a contenedores en ejecución**

Configuración y perfiles: Parámetros

dit UPM

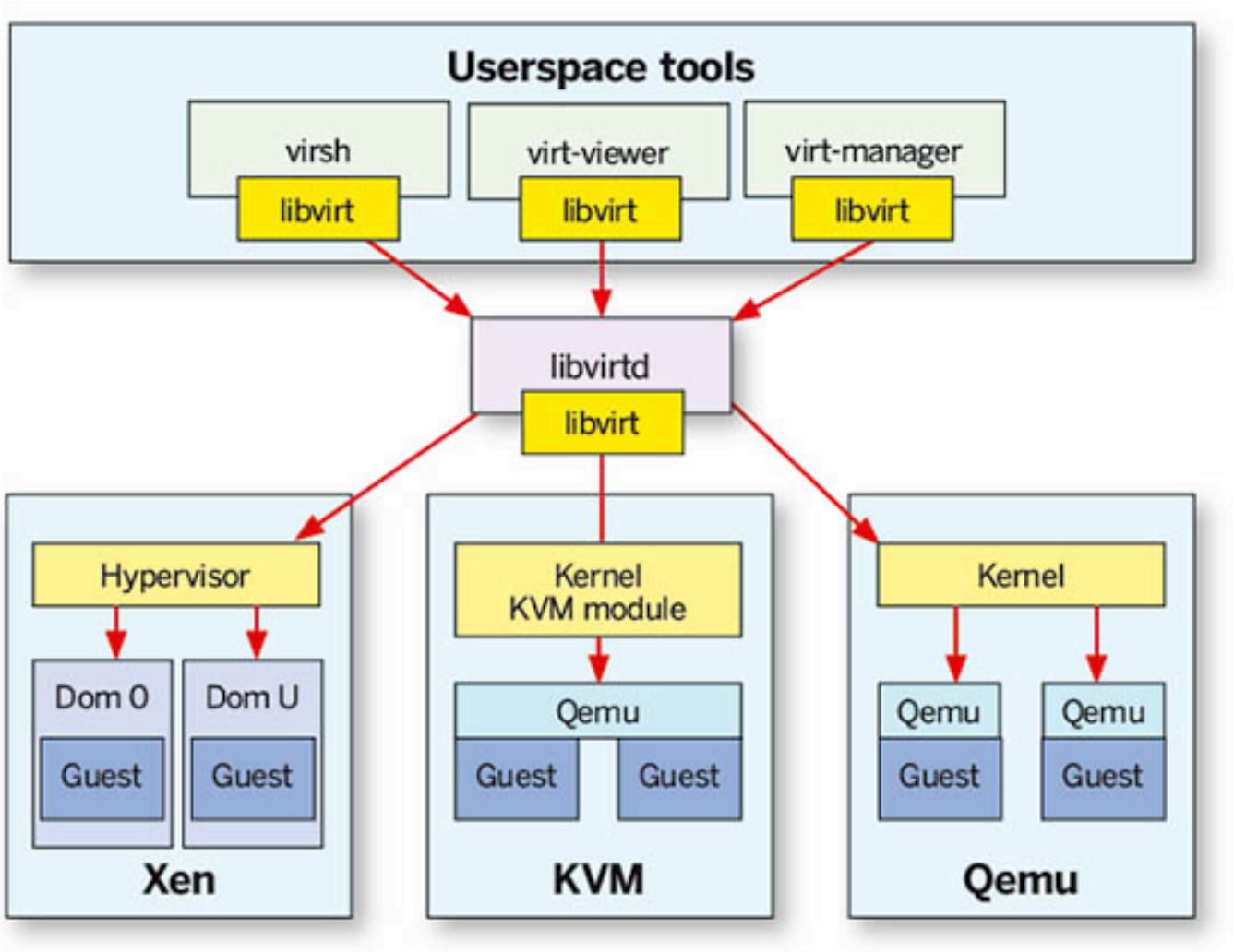
- **Configuración disponible de límites de recursos:**
  - Disco
  - CPU
  - Memoria
  - Red
  - Bloques de disco
- **La lista completa de las configuraciones, tipos de dispositivos y dispositivos se puede acceder en:**
  - https://github.com/lxc/lxd/blob/master/doc/configuration.md

3. Hipervisor sobre el anfitrión en Linux

dit UPM

- **El núcleo de Linux soporta virtualización**
  - Se apoya con instrucciones en hardware para eficiencia
  - **KVM** es un módulo para proporcionar funciones de un hipervisor
- **Hay otras alternativas para Linux:**
  - Xen, VirtualBox, QeMU o VMWare.
- **Para complementar KVM, se usa:**
  - qemu: emulador dispositivos
  - libvirt: biblioteca para la gestión remota de máquinas virtuales
  - Manejadores de dispositivos virtuales

Arquitectura en Linux



Kvm: Kernel-based Virtual Machine

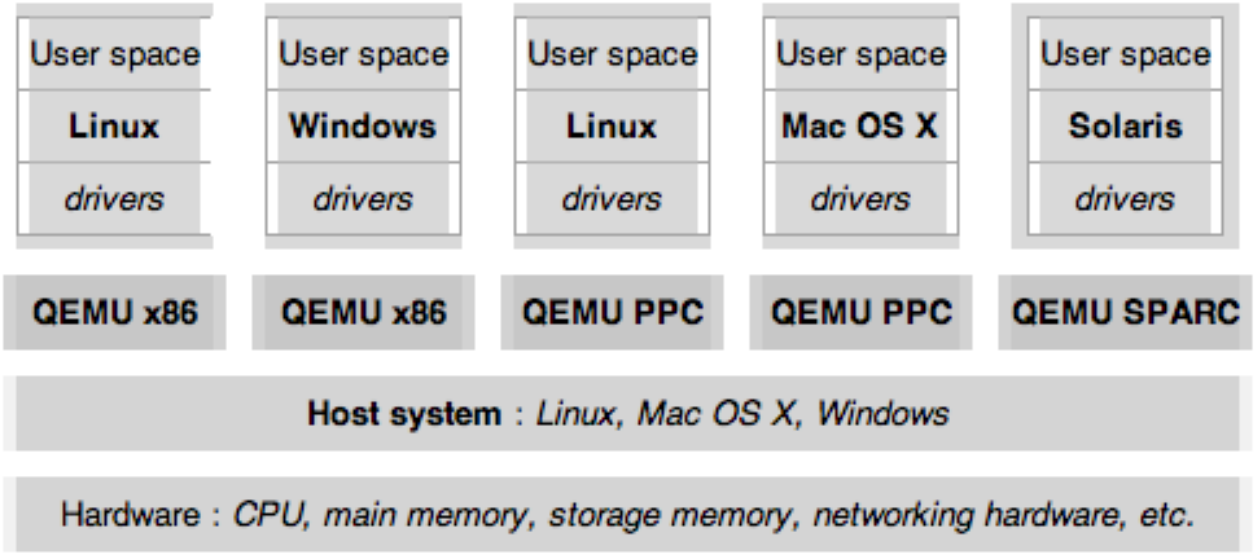
dit UPM

- **Proporciona el soporte básico de virtualización:**
  - Acceso a hardware de virtualización desde el espacio de usuario
  - Requiere extensiones de virtualización en el hardware (Intel o AMD)
- **Es un módulo que se carga dinámicamente**
- **Requiere un emulador en espacio de usuario: qemu**
  - KVM no hace emulación del hardware
- **Proporciona manejadores virtuales para dispositivos**

Qemu: Quick EMUlator

dit UPM

- **Es un hypervisor que emula hw**
  - Ejecuta sobre un sistema anfitrión e interpreta el código binario
  - Ejecuta aplicaciones compiladas en una arquitectura hw en otras.



- **QEMU se apoya en KVM**
  - Si máquina anfitriona y la huésped son la misma, se acelera la ejecución

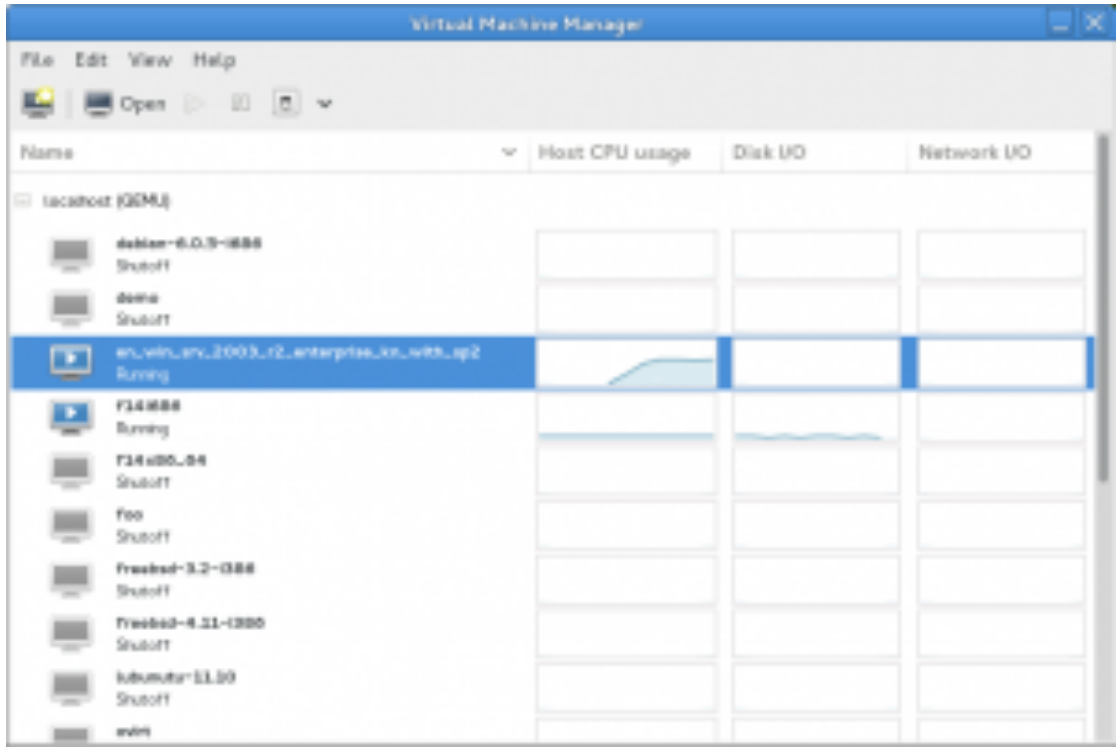
4. Gestión de máquinas virtuales: libvirt

dit
UPM

- **Objetivo: proporcionar una capa estable para gestionar remotamente las máquinas virtuales de un hipervisor**
  - Operaciones: crear, modificar, monitorizar, migrar y parar máquinas virtuales en un computador remoto
  - Se puede hacer de forma remota con seguridad
- **Es una biblioteca con una API para gestionar de forma remota máquinas virtuales**
  - Xen, Qemu, KVM, User Mode Linux y VirtualBox, entre otras
- **El demonio *libvirtd* la usa para comunicar con el sistema de virtualización**

Virt-manager

- **Herramienta gráfica para gestionar máquinas virtuales**
- **Se apoya en las funciones de libvirt**
- **Escrita en python**



Instalación

dit UPM

- **La virtualización tiene que estar habilitada en la BIOS**
- **Asegurar que el hardware tiene extensiones de virtualización:** $ egrep '(vmx|svm)' /proc/cpuinfo
- **Instalar los siguientes paquetes:**
  - qemu-**kvm,** libvirt-bin, ubuntu-vm-builder
  - virt-manager, virtinst
- **Obtener permiso para acceder a funciones de libvirt**
  - $ sudo adduser username kvm
- **Asegurar que el módulo kvm está cargado**
  - $ lsmod
  - Cargarlo, si no: $ sudo modprobe kvm-intel

Creación de máquina virtual: virt-manager

dit UPM

- **Menú para crear una máquina virtual**
- **Inicialmente, hay que proporcionar:**
  - Nombre
  - Método de instalar el sistema operativo
    - A partir de un cdrom o fichero .iso. Hay que indicar el sistema de ficheros raíz (imagen de disco). Se arranca el proceso de instalación.
    - Imagen de disco existente. Se supone que ya está instalado el sistema operativo
  - Número de núcleos y memoria
  - El resto, lo toma por defecto.
- **Se puede arrancar la máquina virtual**

Almacenamiento

dit UPM

- **Diversas opciones:**
  - Imagen de disco en un fichero:
    - Emula un volumen o partición. Emula un sistema de ficheros en el fichero
    - Se puede definir una partición, un disco o un directorio donde están las imágenes
  - Partición de disco
  - Otro dispositivo
  - Sistema de ficheros remoto, montado desde la máquina virtual
- **La imagen se puede crear:**
  - Desde virt-manager o con qemu-image
  - Se puede crear nueva o copiar una existente
  - Opción de copia: copy-on-write.
    - Se crea una referencia al fichero original. Sólo se almacenan los cambios
    - Se puede usar con imágenes qcow2

Red

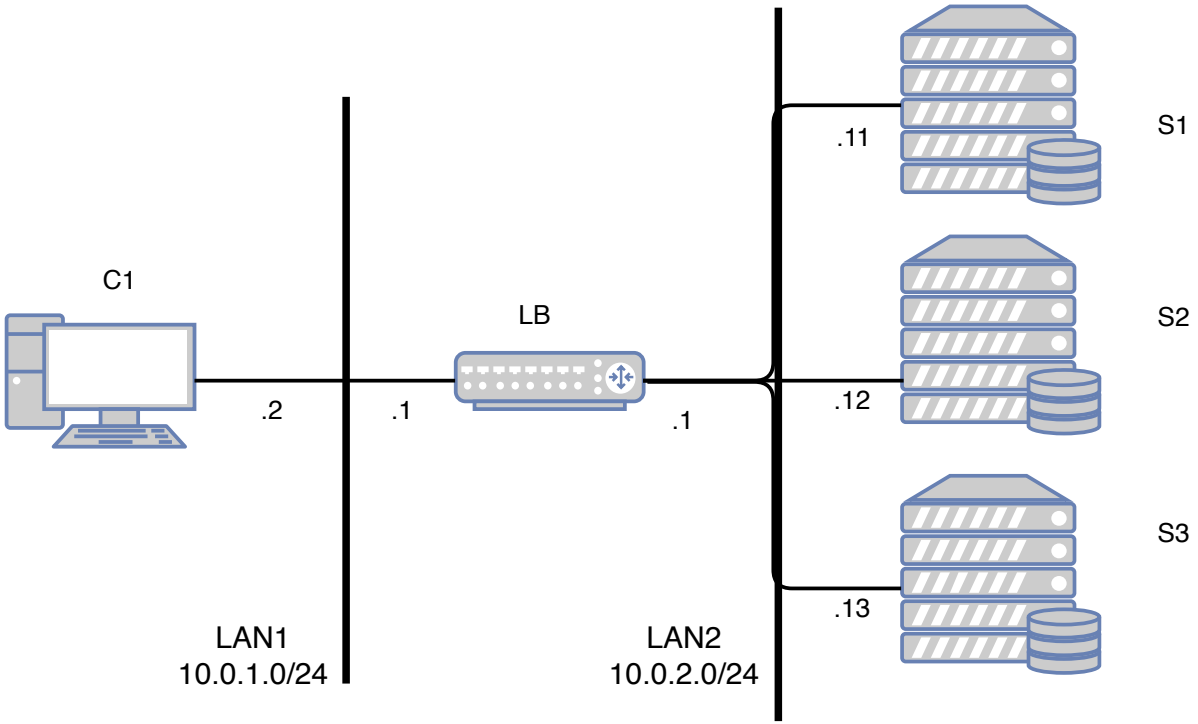
dit
UPM

- **Configuración por defecto: *Usermode networking***
  - Se basa en crear una red virtual
  - Conecta las máquinas virtuales a la red
  - Comparte la conexión con el exterior mediante NAT
  - Las máquinas virtuales no son visibles desde el exterior

- **Alternativa: *Bridge networking***
  - Se conectan dos redes separadas
  - Las máquinas virtuales se conectan a ellas.
  - El *bridge* les comunica
  - Las máquinas virtuales se conectan directamente a la red.

Balanceador de redes

- **Distribuye peticiones de clientes a un conjunto de servidores replicados**
- **Equilibra la carga de los servidores**



Ficheros de un máquina virtual

dit UPM

- **La descripción de la máquina virtual es un fichero xml**
  - Por defecto está en /etc/libvirt/qemu
- **Por defecto, usa un directorio para almacenar imágenes:**
  - /var/lib/libvirt/images
- **Se puede crear una máquina virtual duplicando el sistema de ficheros y la descripción en xml**

Gestión de máquinas virtuales

dit UPM

- **El interfaz gráfico es útil para gestionar pocas máquinas**

- **Métodos alternativos son necesarios:**
  - Creación/arranque/modificación dinámica de máquinas virtuales
  - Gestión automáticas de una gran número de máquinas
  - Monitorización a gran escala

- **Se pueden invocar las funciones de libvirt:**
  - desde la línea de órdenes
  - llamadas desde un programa

Virsh

dit UPM

- **Mostrar la información de MV (guests)**
  - Lista de MV definidas: $ virsh list --all
  - Información de una MV: $ virsh dominfo guest's_name
- **Parar y arrancar MV**:
  - Arrancar una MV: $ virsh start guest's_name
  - Parar una MV: $ virsh shutdown guest's_name
  - Abortar una MV: $ virsh destroy guest's_name
  - Suspender una MV: $ virsh suspend guest's_name
  - Reactivar una MV suspendida: $ virsh resume guest's_name

Virsh

dit UPM

- **Crear y modificar MV**:
  - Crear una MV nueva
  - Crear una MV desde su definición XML: $virsh create xml_file.xml
  - Obtener la definición XML de una MV: $virsh dumpxml guest's_name
  - Modificar la definición de una MV: $virsh edit guest's_name
  - Borrar la definición de una MV (conserva la imagen):
    $ virsh undefine guest's_name
- **Salvar y restaurar MV**:
  - Salvar el estado de una MV en un fichero:
    $ virsh save guest's_name guest's_state_file
  - Restaurar una MV desde un fichero con el estado:
    $ virsh restore guest's_state_file

Virsh: Órdenes de monitorización

dit UPM

- **Obtener el tamaño de un dispositivo de bloques de un dominio:** $ virsh domblkinfo <domain> <device>
- **Obtener estadísticas de dispositivo de bloques en dominio en ejecución:** $virsh domblkstat <domain> <device>
- **Obtener estadísticas del interfaz de red de dominio en ejecución:** $virsh domifstat <domain> <interface>
- **Obtener estadísticas de memoria de un dominio en ejecución:** $virsh dommemstat <domain>
- **Retorna es el estado de un dominio:**
  $virsh domstate <domain> [--reason]

Referencias

dit UPM

- A. Silberschatz, P. Galvin y G. Gagne: ***Operating System Concepts with Java***. 8ª edición. Addison Wesley, 2011.
- http://www.linux-kvm.org: Sitio oficial de kvm
- http://www.qemu.org: Sitio oficial de qemu
- http://www.libvirt.org: Es el sitio oficial de libvirt. Cuenta con extensa documentación sobre esta biblioteca.
- Red Hat Enterprise Linux 5. Virtualization Guide: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/index.html
- Virtualization Getting Started Guide, Fedora: https://docs.fedoraproject.org/en-US/Fedora/19/html/Virtualization_Getting_Started_Guide/index.html

Referencias

- **For CTO's: the no-nonsense way to accelerate your business with containers, February 2017, Canonical**
  - Disponible en moodle
- **LXD 2.0: Blog post series:**
  - https://stgraber.org/2016/03/11/lxd-2-0-blog-post-series-012/