

# Cloud and Datacenter Networking

Università degli Studi di Napoli Federico II

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI

Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

**Server virtualization technologies  
and their use in virtualized datacenters**  
**Introduction to VM networking**





- ▶ Hypervisor-based virtualization technologies
- ▶ A Linux-based hypervisor: KVM
- ▶ Hypervisors and VM networking

- ▶ In Computer Engineering, virtualizing a physical resource means:  
“to implement the same functionality provided by a *physical resource* by means of a logical or virtual instance”
  - ▶ Physical resources may be CPU, memory, network connection, etc.
  - ▶ Virtualizing typically involves “multiplexing”, i.e. sharing the same physical resource functionality among several “virtual” resources
  - ▶ Multiplexing, in turn, is typically implemented in software
- ▶ «A framework or methodology of dividing the resources of a computer hardware into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time sharing, partial or complete machine simulation, emulation, quality of service, and many others.»

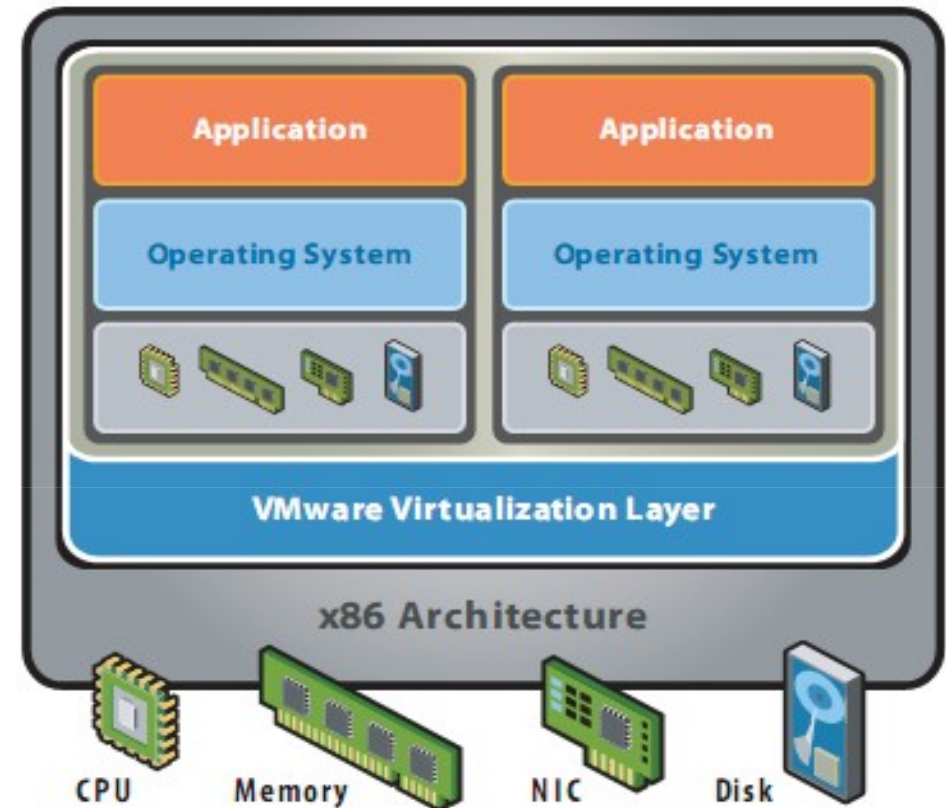
From: <http://www.kernelthread.com/publications/virtualization/>

- ▶ Almost any hardware or software resource may be virtualized:
  - ▶ CPU
  - ▶ RAM memory
  - ▶ Storage capacity of a Hard Disk
  - ▶ Operating Systems
- ▶ A typical virtualization example:
  - ▶ a hard disk capacity split into multiple logical partitions
- ▶ By *hardware virtualization* we mean the set of techniques that allow us to virtualize a whole computer by creating multiple *Virtual Machines* (VM) concurrently running in the same computer
- ▶ These virtualization techniques must be able to virtualize ALL the physical resources of a computer by creating a «virtual» hardware exposed to VMs

# Virtual Machines



- ▶ A single physical server (*host*) ‘hosts’ a number of Virtual Machines (VM) (*guests*)
- ▶ A VM is a software-based system behaving as a single computer
- ▶ A VM behaves exactly as a real computer and is configured with a set of *virtualized resources* (CPU, RAM, NICs, Disks, etc.)
- ▶ Each VM runs its own operating system and, on top of it, a number of applications (*processes*)
- ▶ It is an important feature of the hypervisor to guarantee that each VM is completely isolated from all the other VMs running in the same host



Source:

<http://www.vmware.com/pdf/virtualization.pdf>

# Server Virtualization vs Desktop Virtualization

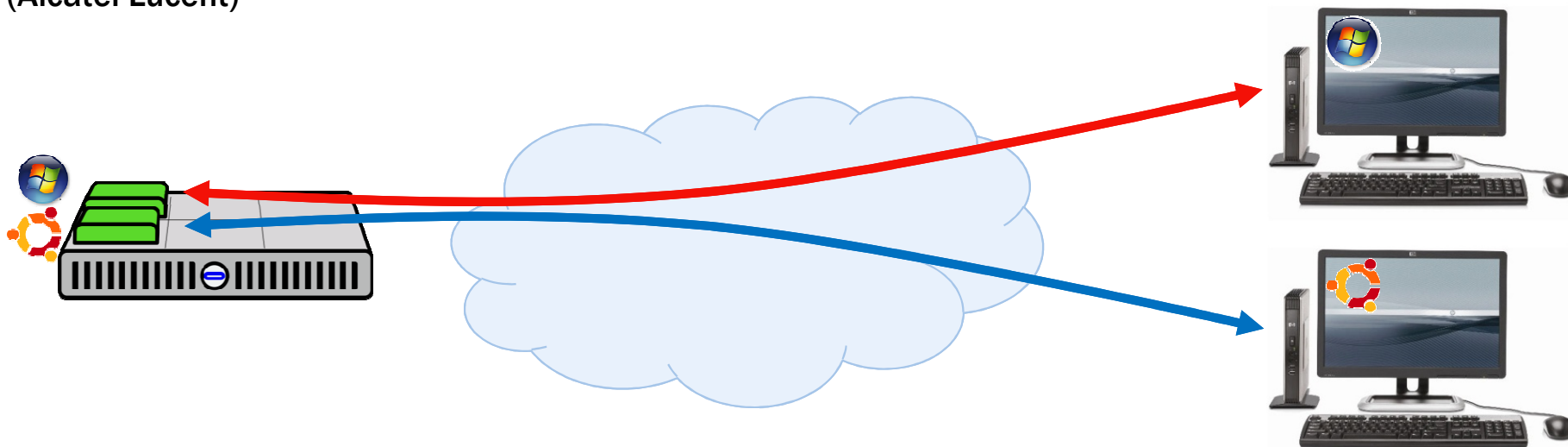


- ▶ Hardware Virtualization techniques may be applied in two different scenarios:
  - ▶ Server Virtualization
  - ▶ Desktop Virtualization
- ▶ In both cases, the objective is to create VMs behaving as a computer
- ▶ In the case of *Server Virtualization*, the VM is instantiated to run server processes, that can be configured and instantiated from a command line interface
  - ▶ Typically, a computer acting as server is not configured with a Graphical User Interface, and the administration of services is performed through a Command Line Interface that can be remotely accessed through SSH (*Secure Shell*)
- ▶ In the case of *Desktop Virtualization*, it is important for the user to remotely interact with the operating system running in the VM through a graphical interface (*Desktop*)

# Virtual desktop infrastructure (VDI)



- ▶ In the VDI (*Virtual Desktop Infrastructure*) use-case, a single server runs many VMs, each configured with a Desktop of its own
- ▶ VMs are not servers but virtualized personal computers
- ▶ Each user interacts with his/her own VM through a minimalistic computer (*thin client*) equipped with a hi-res screen, a mouse and a keyboard
  - ▶ By means of the thin client and of a broadband network, users interact with their remote desktop running in a VM hosted by a VDI server
- ▶ Access to the VM's remote desktop may produce high data rate traffic to thin clients
- ▶ To preserve Quality of Experience (QoE), a reduced latency is also required
  - ▶ “A VDI service that offers an experience that is almost indistinguishable from a desktop experience for office suite applications requires 6 Mb/s or more of bandwidth and less than 20 milliseconds (ms) of round-trip latency” (Alcatel-Lucent)

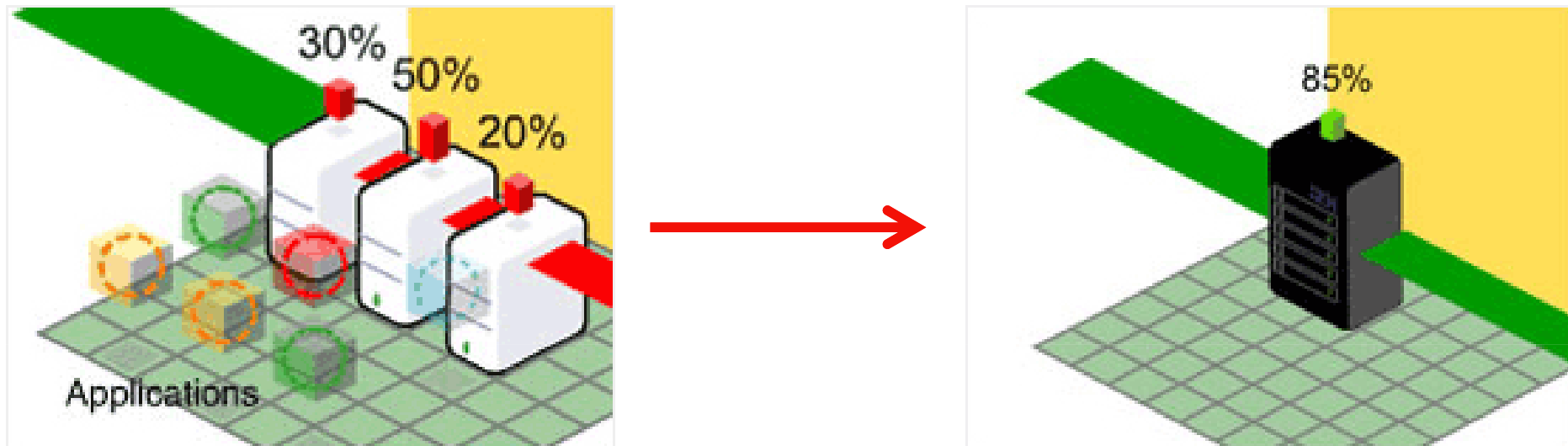


# Server virtualization: pros

- **Costs reduction:**
    - **Purchase costs (CAPEX)**
    - **Consumptions costs (OPEX)**
      - Energy
      - Conditioning
      - Volume – rack space
    - **Maintenance costs (OPEX)**
      - Installation
      - Configuration
      - Replication/cloning
      - Backup
  - **Increase of availability**
    - Service downtime reduction
    - Business Continuity
    - Disaster Recovery
  - **Faster deployment of new services**
- } Faster hence cheaper



# Server virtualization: consolidation



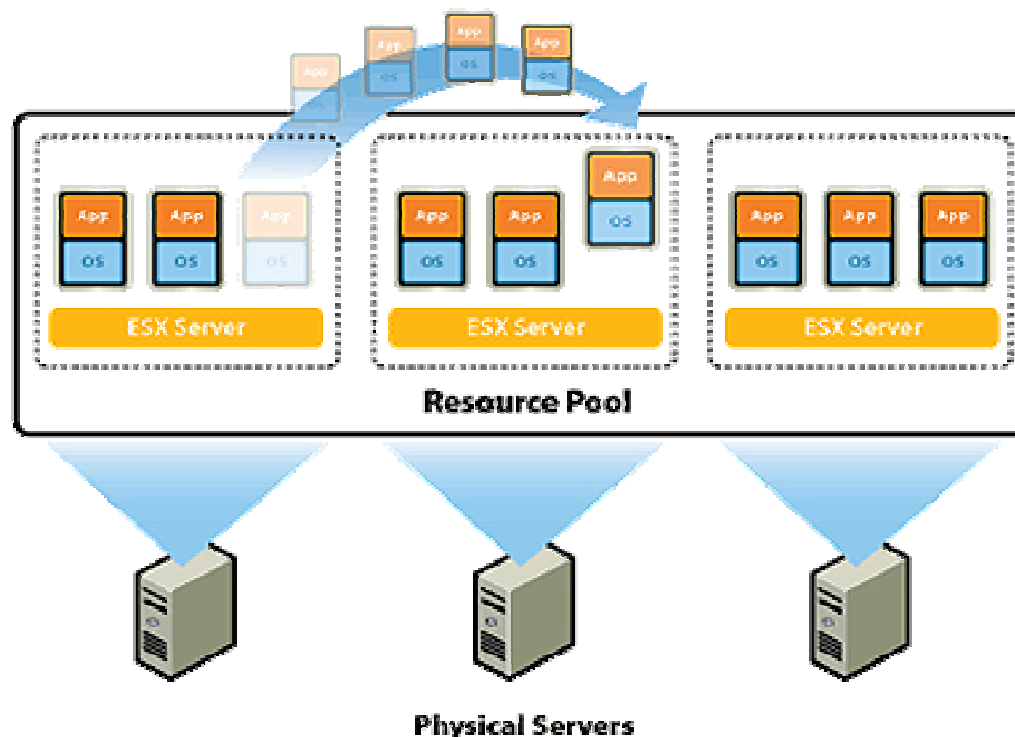
**Server consolidation:** instead of having many physical servers (one per application), each utilized only for a small fraction of its capacity, many VMs running on a smaller number of physical servers

This leads to higher utilization of physical resources, and to a reduction of energy consumption and space

# Server Virtualization: live migration



- ▶ Live Migration: the operation of moving a running VM from a physical server to another one, with no effect on availability of services provided by the VM
  - ▶ Memory, storage, and network connectivity of the virtual machine are transferred from the original hypervisor to the new one
- ▶ This is a key feature of modern hypervisors:
  - ▶ to consolidate VMs in a datacenter
  - ▶ to perform maintenance operations on physical servers' hardware or software (hypervisor)





## ▶ Isolation

- ▶ A VM should never influence other co-located VMs (i.e. running in the same physical host)
- ▶ Isolation may be affected because physical resources are actually shared among VMs
- ▶ The hypervisor should preserve isolation: for instance a VM using 100% of its own virtual CPU should not steal CPU cycles to another co-located VM

## ▶ Management flexibility

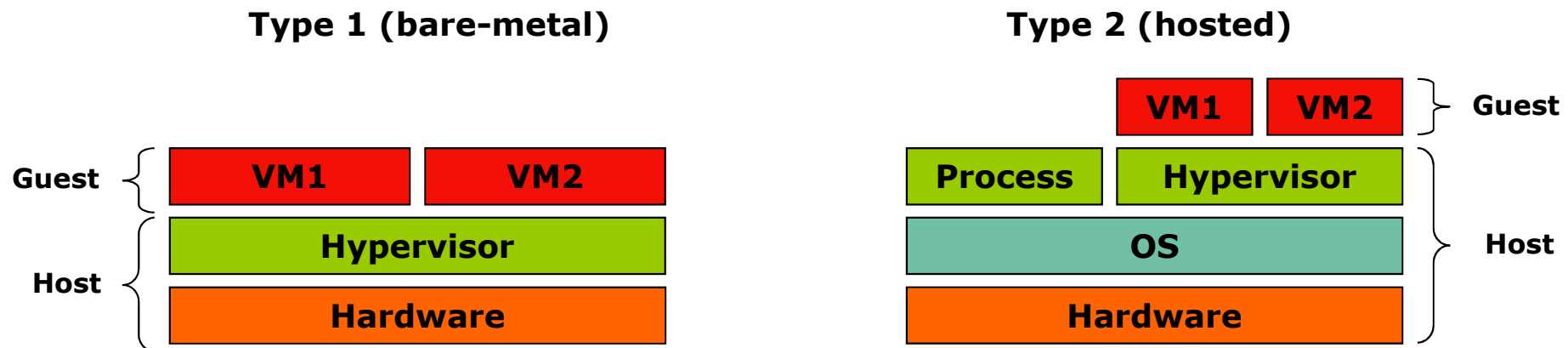
- ▶ The hypervisor should make it possible for the administrator to configure (and possibly change) the amount of virtual resources assigned to a VM
  - ▶ For instance, it should be possible to expand the amount of RAM or change the number of virtual NICs for a VM
- ▶ Possibility of turning on/off VMs at any time
- ▶ Possibility of live migrating a VM from one physical server to another

## ▶ Ability to guarantee Quality of Service

# Two types of hypervisors



- ▶ A Hypervisor (or VMM – Virtual Machine Monitor) is a software layer that allows several virtual machines to run on a physical machine
  - ▶ The physical OS and hardware are called the Host
  - ▶ The virtual machine OS and applications are called the Guest
- ▶ Two types of hypervisors are commonly recognized: Type-1 and Type-2



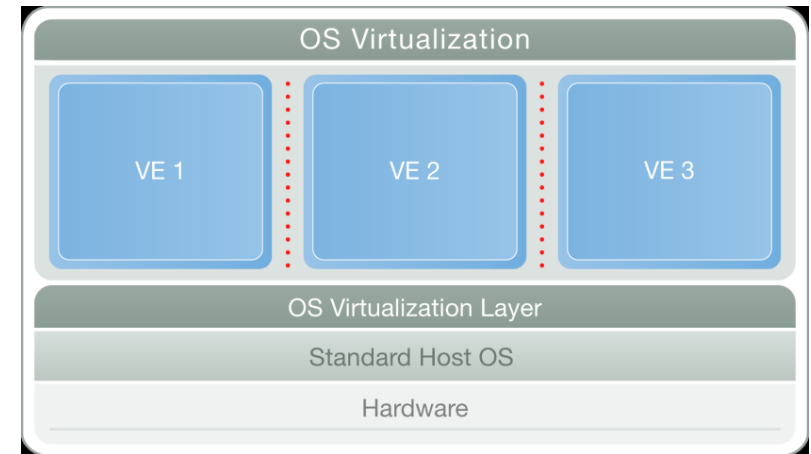
VMware ESX, Microsoft Hyper-V, Xen Server

VMware Workstation, Microsoft Virtual PC, Sun VirtualBox, QEMU, KVM

# A different approach to virtualization: containers



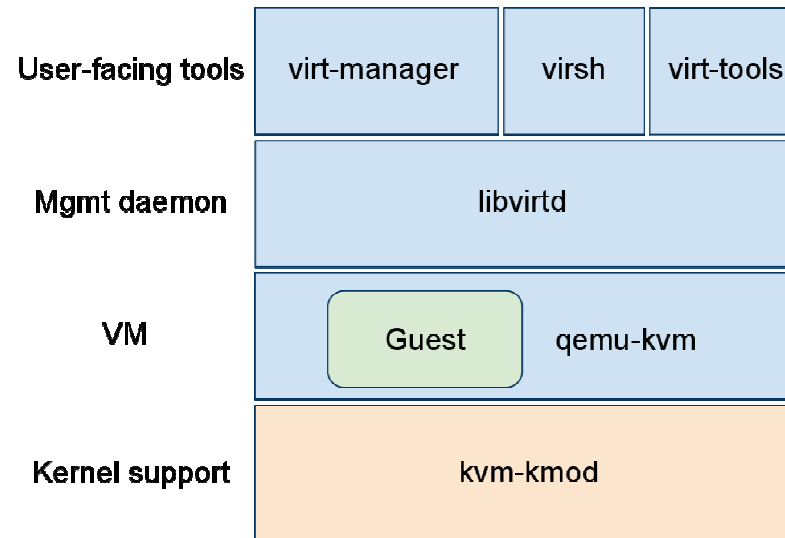
- ▶ Lightweight virtualization provided by the OS
- ▶ One real HW (no virtual HW), one kernel, many user-space instances
- ▶ Less overhead, best performance, more concurrent virtualized execution environments on the same hardware
  
- ▶ An idea that has been implemented in several ways
  - ▶ FreeBSD jails
  - ▶ Linux-Vserver
  - ▶ OpenVZ / Parallels Containers
  - ▶ Solaris Containers/Zones
  - ▶ IBM AIX6 WPARs



# KVM: Kernel-based Virtual Machine for Linux



- ▶ KVM is a full virtualization solution that turns a Linux kernel into a hypervisor using a kernel module (kvm.ko)
- ▶ Open source project: <http://www.linux-kvm.org>
- ▶ Implemented in Linux Kernel since 2.6.20
- ▶ The introduction of the KVM is an interesting evolution of Linux, as it represents the first virtualization technology that is part of the mainline Linux kernel
- ▶ Two components: a kernel module (kvm.ko) and a user process (QEMU)
- ▶ In practice, other user-level tools are needed (libvirtd, virsh, virt-tools, ...)
- ▶ When run on CPUs that supports virtualization, Linux and Windows guests are supported



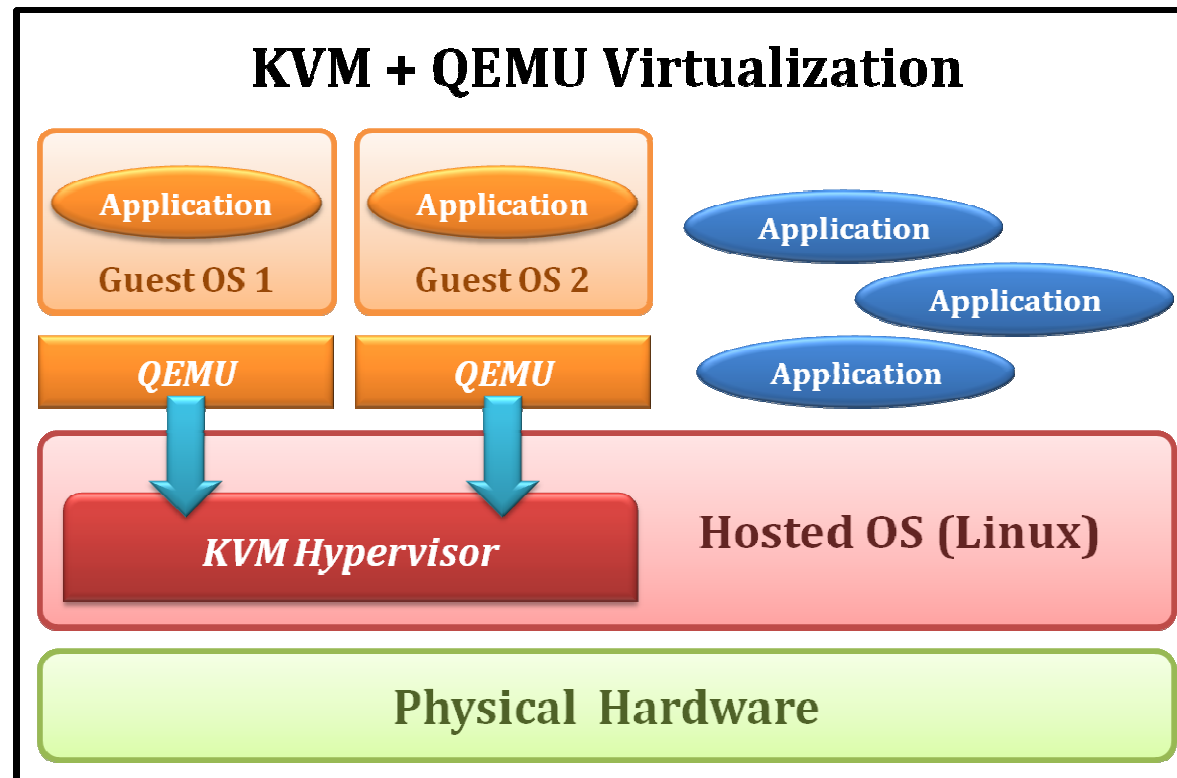
# KVM kernel modules (1)



- ▶ KVM is based on a loadable kernel module (kvm.ko) that allows other guest operating systems to run in user-space
- ▶ kvm.ko exposes virtualized hardware through the /dev/kvm character device
- ▶ The KVM module introduces a third execution mode into the kernel
  - ▶ Where vanilla Linux kernels support *kernel mode* and *user mode*, KVM introduces a *guest mode*
- ▶ The guest mode is used to execute all non-I/O guest code, where normal user mode supports I/O for guests
- ▶ Zero impact on host kernel
- ▶ Guests are scheduled as regular processes
  - ▶ kill(1), top(1) work as expected
- ▶ The guest operating system interfaces to the KVM module using a modified QEMU process for PC hardware emulation

# KVM kernel modules (2)

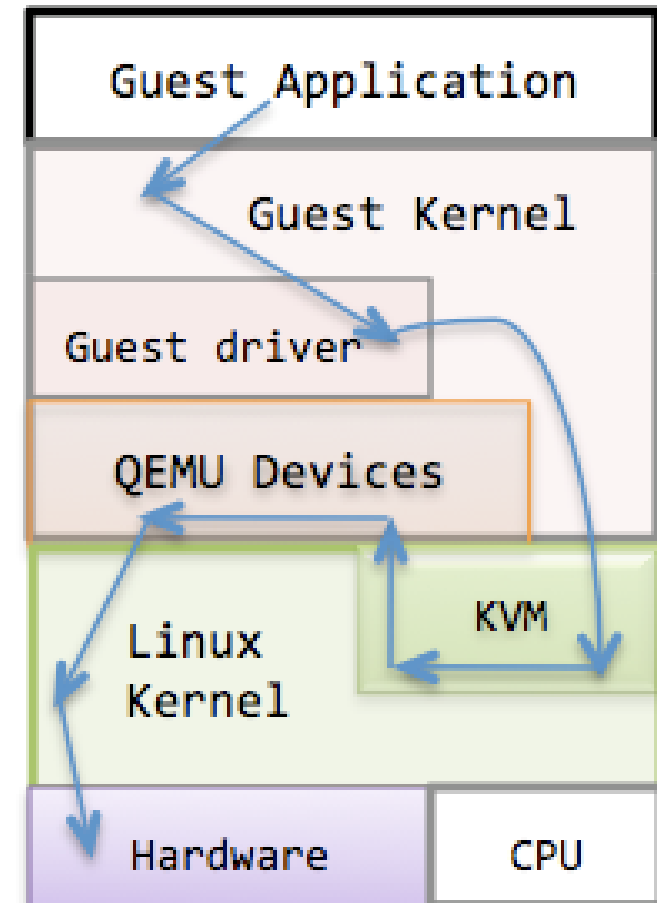
- **kvm.ko**
  - provides the core virtualization infrastructure
- **kvm-intel.ko / kvm-amd.ko**
  - processor specific modules





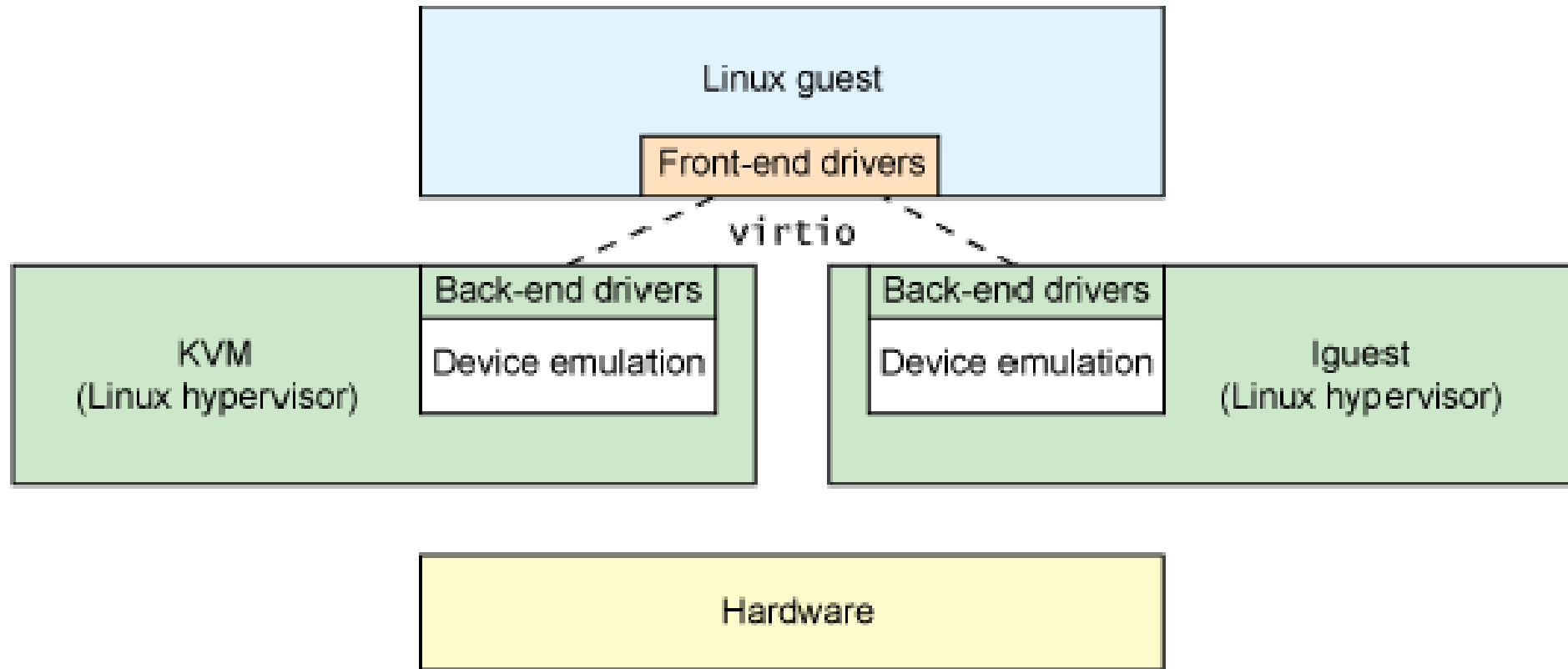
# I/O in KVM (full virtualization)

- Original approach with full-virtualization
  - Guest hardware accesses are intercepted by KVM
  - QEMU emulates hardware behavior of common devices
    - RTL 8139
    - PIIX4 IDE
    - Cirrus Logic VGA



# I/O in KVM (virtio)

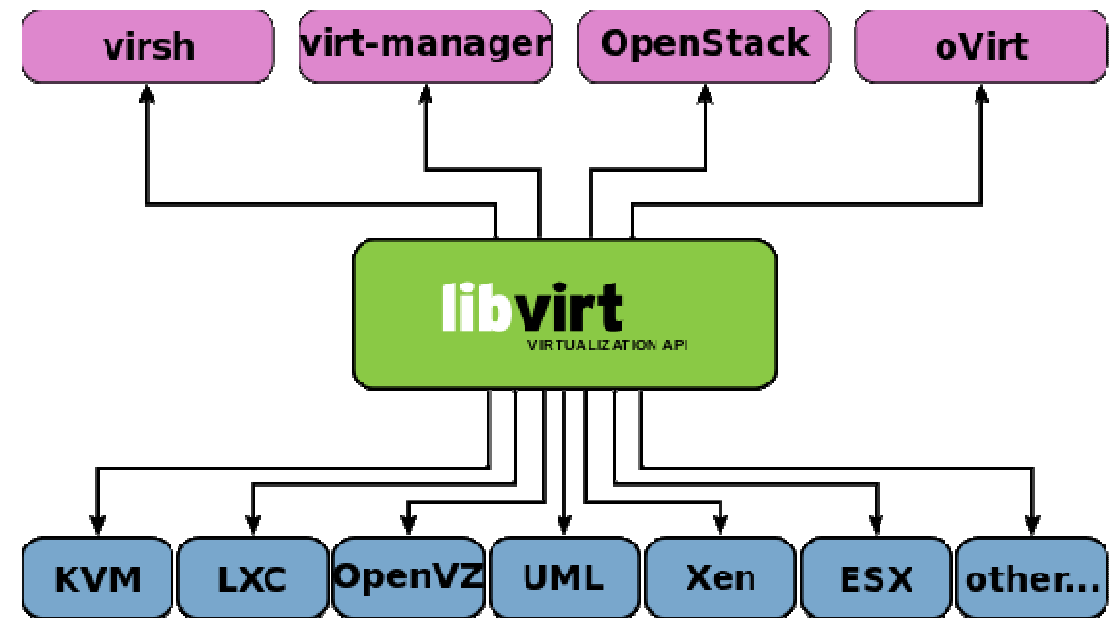
- New approach with para-virtualization



# Libvirt – API for managing VMs



- ▶ Libvirt is an open source API, daemon and management tool for managing platform virtualization
- ▶ Implemented as a C library, can be used by programs written in many languages, such as Python, Perl, OCaml, Ruby, Java, and PHP
- ▶ Widely used in the orchestration layer of hypervisors in the development of a cloud-based solution (e.g. in OpenStack)
- ▶ Can be used to manage KVM, Xen, VMware ESX, QEMU and other virtualization technologies
- ▶ Two User Interfaces:
  - ▶ Graphical Interface: virt-manager
  - ▶ Command line interface: virsh



## ▶ Virtual Machine Manager

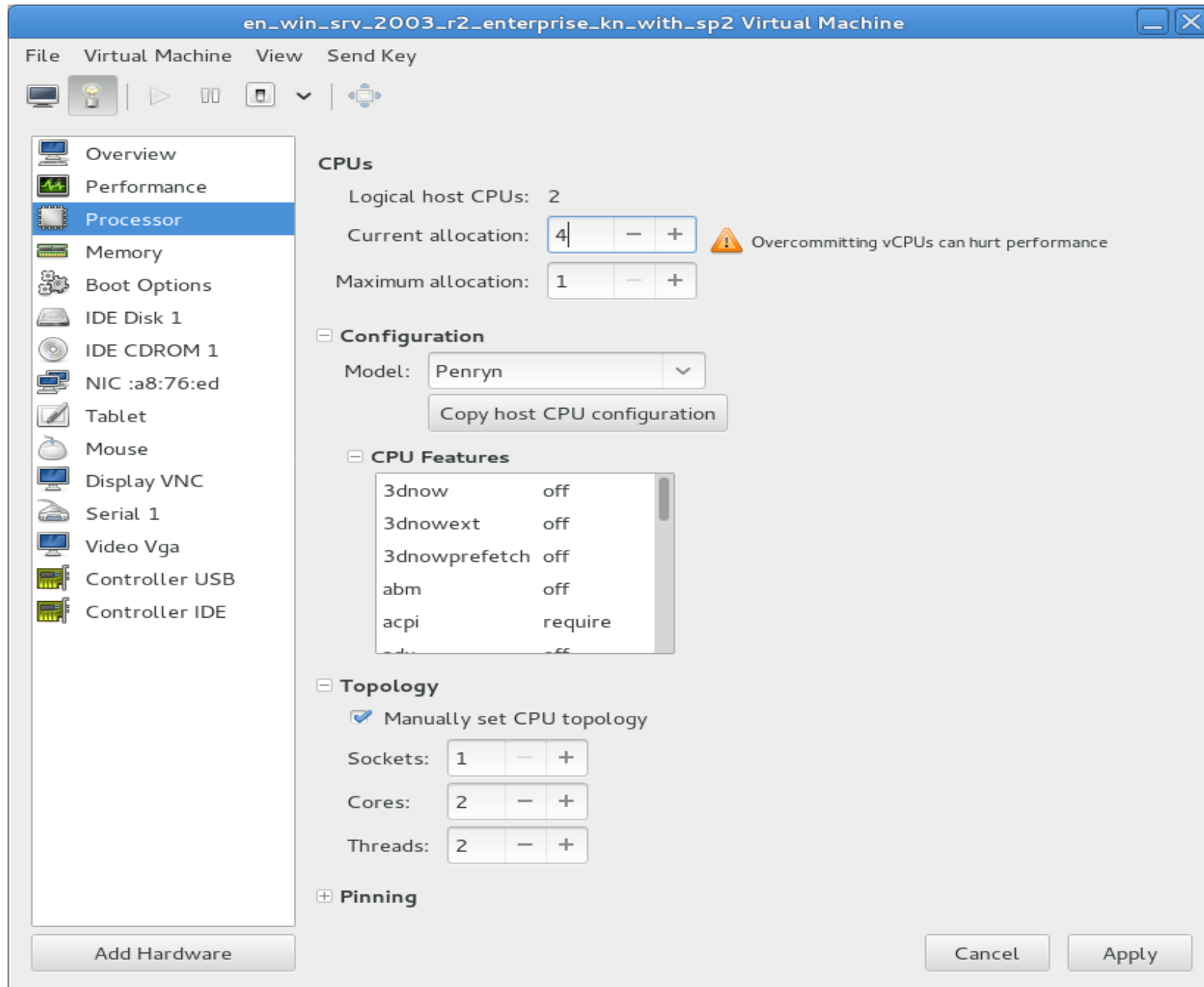
- ▶ is a desktop-driven virtual machine manager with which users can manage virtual machines (VMs)

## ▶ Functions

- ▶ create, edit, start and stop VMs

## ▶ virt-manager's supporting tools

- ▶ virt-install tool
- ▶ virt-clone tool
- ▶ virt-viewer application

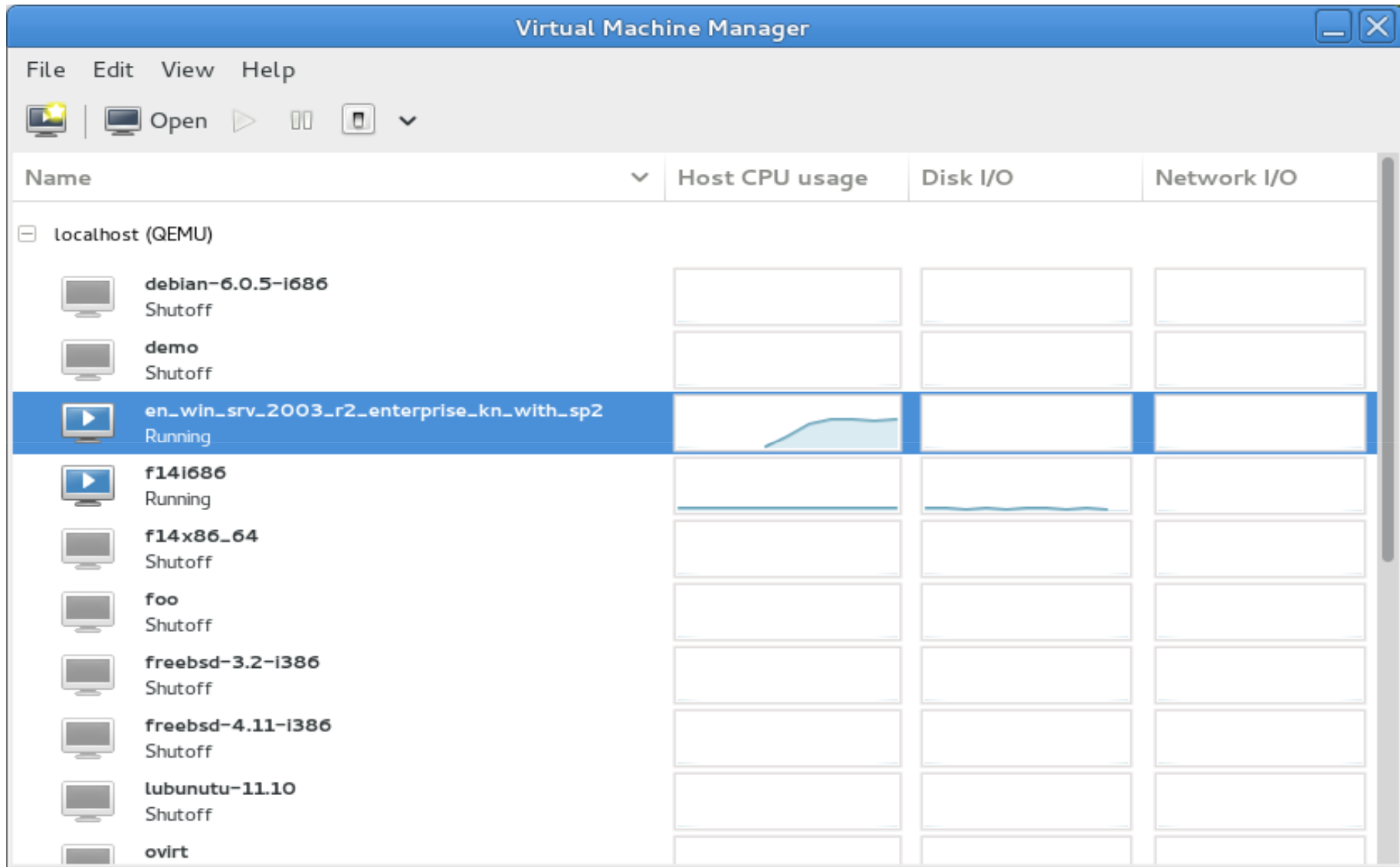


The screenshot shows the 'CPU' configuration tab in the virt-manager interface. The window title is 'en\_win\_srv\_2003\_r2\_enterprise\_kn\_with\_sp2 Virtual Machine'. The left sidebar contains a list of hardware components, with 'Processor' selected. The main area is divided into several sections:







- CPUs:** Logical host CPUs: 2. Current allocation: 4 (with minus and plus buttons). Maximum allocation: 1 (with minus and plus buttons). A warning icon and text state: 'Overcommitting vCPUs can hurt performance'.
- Configuration:** Model: Penryn (dropdown menu). A button labeled 'Copy host CPU configuration' is present.
- CPU Features:** A list of features with their status:

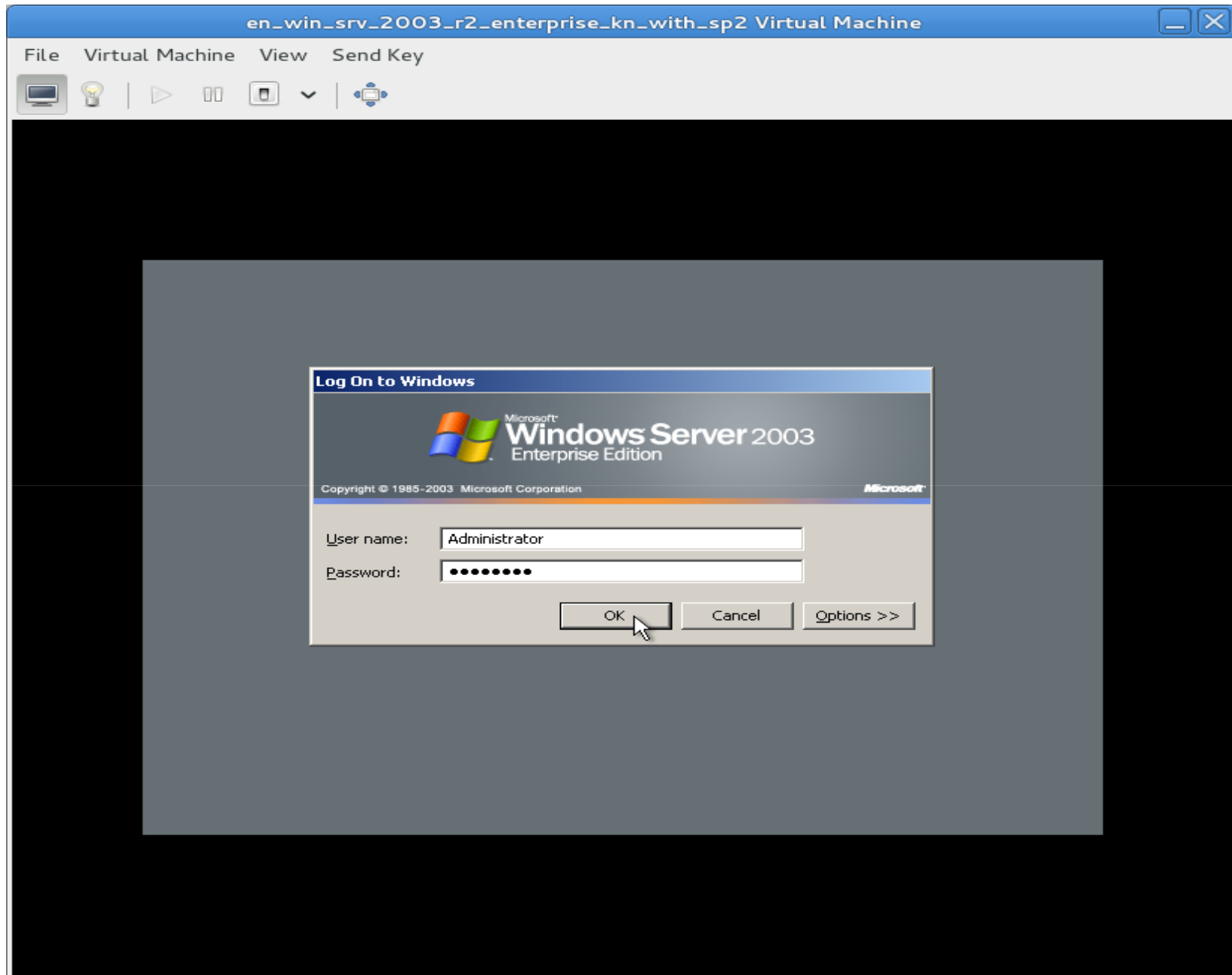
3dnow	off
3dnowext	off
3dnowprefetch	off
abm	off
acpi	require
- Topology:**  Manually set CPU topology. Sockets: 1 (with minus and plus buttons). Cores: 2 (with minus and plus buttons). Threads: 2 (with minus and plus buttons).
- Pinning:** A plus sign icon and the label 'Pinning' are visible, but no specific settings are shown.

At the bottom, there are three buttons: 'Add Hardware', 'Cancel', and 'Apply'.



The screenshot shows the Virtual Machine Manager window. The title bar reads "Virtual Machine Manager". The menu bar includes "File", "Edit", "View", and "Help". The toolbar contains icons for "Open", "Play", "Pause", and "Shutdown".

Name	Host CPU usage	Disk I/O	Network I/O
localhost (QEMU)			
debian-6.0.5-i686 Shutoff			
demo Shutoff			
en_win_srv_2003_r2_enterprise_kn_with_sp2 Running			
f14i686 Running			
f14x86_64 Shutoff			
foo Shutoff			
freebsd-3.2-i386 Shutoff			
freebsd-4.11-i386 Shutoff			
lubunutu-11.10 Shutoff			
ovirt			



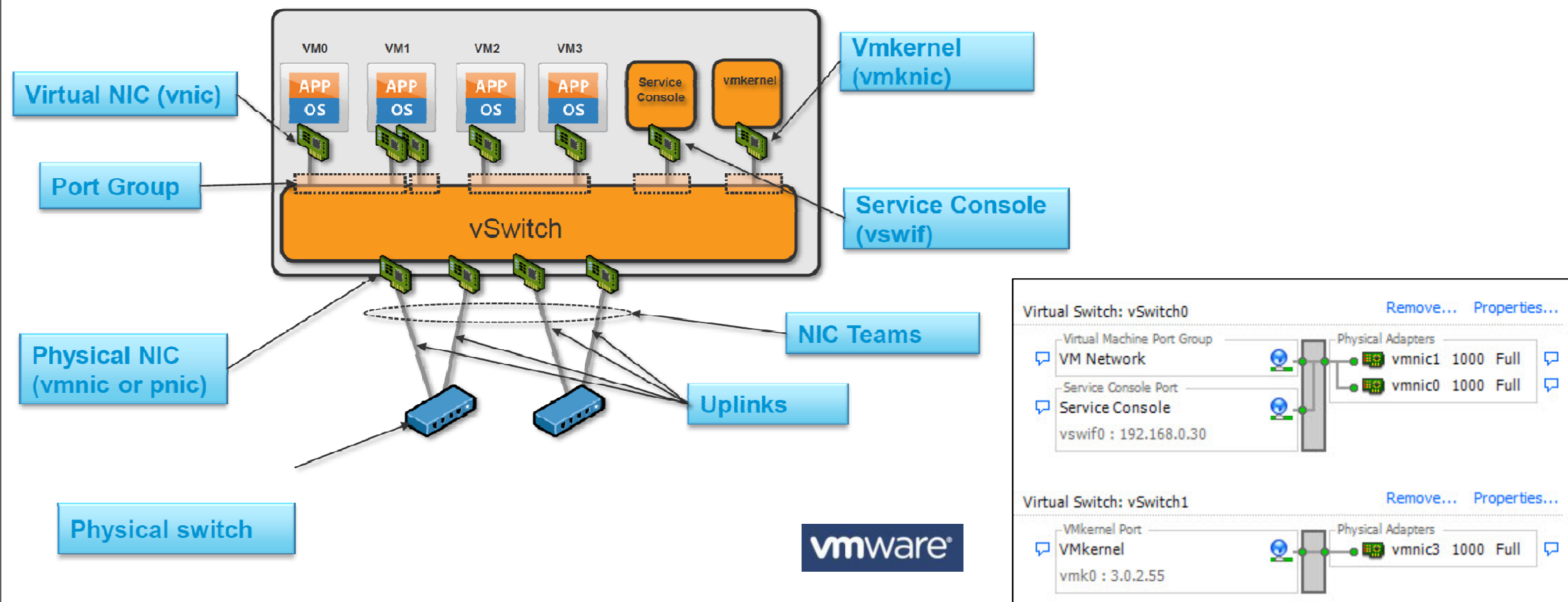
- ▶ **virsh is a command-line tool used to manage domains (i.e. VMs)**
- ▶ **virsh commands need root privileges to be executed**
- ▶ **Common used commands:**
  - ▶ `virsh create`
    - ▶ **start a VM from an XML descriptor file**
  - ▶ `virsh destroy`
  - ▶ `virsh list`
    - ▶ **list all the running VMs**
  - ▶ `virsh console`
    - ▶ **connect to a VM**
  - ▶ `virsh reboot`
  - ▶ `virsh shutdown`
  - ▶ ...



# Hypervisors and VM networking (1)



- ▶ In a physical host with several VMs, each VM has its own virtual NIC(s) (or **vNICs**)
- ▶ Virtual NICs are connected to the host physical NIC(s) by means of a virtual switch (or **vSwitch**) whose job is to dispatch packets from/to VMs according to their virtual MAC addresses
- ▶ A single hypervisor may be configured with several vSwitches



# Hypervisors and VM networking (2)



- ▶ A hypervisor creates virtualized network devices: **vNICs** and **vSwitches**
- ▶ Many VMs connected by virtual network devices create a *virtual network*
- ▶ **Virtual network interface card (vNIC)**
  - ▶ Hypervisor can create one or more vNICs for each VM
  - ▶ The vNIC provides the networking capabilities of the VM
  - ▶ Each vNIC is identical to a physical NIC
- ▶ **Virtual switch(vSwitch)**
  - ▶ Switches also can be virtualized as a virtual switch
  - ▶ Each vNIC is connected to a vSwitch port
  - ▶ A vSwitch may be connected to an external physical network through a physical NIC (pNIC) of the hypervisor

- ▶ **Virtual Network Adapters**
  - ▶ vNic – VM’s interface to the network
  - ▶ vmknics – vSphere hypervisor’s interface to network (nfs, iSCSI, vMotion, FT, Management)
  - ▶ vswif – Interface for Service Console (not present on ESXi)
- ▶ **Physical Network Adapter**
  - ▶ pNic – for communicating with entities outside ESX/ESXi host
- ▶ **Virtual Switch**
  - ▶ vSwitch – forwards packets between vNics, vmknics, and pNics
- ▶ **Port Group**
  - ▶ Group of ports sharing the same configuration (e.g. vlan)
- ▶ **Uplinks: connections to physical switches**
- ▶ **NIC Team: a group of pNics connected to the same physical network**

# VMware: 3 types of Virtual Switches



- ▶ **vNetwork Standard Switch (vSS)**
  - ▶ Created and managed on a per-host basis
  - ▶ Support basic features such as VLAN, NIC teaming, port security
- ▶ **vNetwork Distributed Switch (vDS)**
  - ▶ Created and managed at vSphere vCenter
  - ▶ Supports all vSS features and more (PVLAN, traffic management, etc.)
  - ▶ NOTE: vSS/vDS share same etherswitch module, only control path differ
- ▶ **Cisco Nexus 1000v (N1K)**
  - ▶ Created and managed by VSM (either VM or hardware/Nexus 1010)
  - ▶ Supports features typically available in Cisco hardware switches

- ▶ There are 3 popular methods to connect a VM to the host and to the external networks to which the host is connected
- ▶ **Bridged:** under the Bridged method, the VM will directly contact the DHCP server of the external physical network and apply for a unique local IP address in the external network; the VM will be then able to directly access the external network; this is the preferred connection method, if we run any server in the VM
- ▶ **NAT (Network Address Translation):** under this method, the VM accesses the host's external network with the IP address of the host; within the host, we have a virtual private network involving the host and the VMs running on it. The other hosts in the external network cannot directly access the VM and they have to go through the NAT process at the host; in other words, the host PC acts as the first-stop gateway router for the VM
- ▶ **Host-only:** This method is same as the NAT except the VMs cannot access the external network as the host does not act as a NAT router for the VMs; communication is only allowed among VMs running in the same host

- ▶ **Network address translation (NAT)** configures your VM to share the IP and MAC addresses of the host.
  - ▶ The VM and the host share a single network identity that is not visible **outside the network**.
  - ▶ **NAT can be useful** when you are allowed a **single IP address** or MAC address by your network administrator.
  - ▶ You might also **use NAT to configure separate VMs** for handling http and ftp requests, with both VMs running off the same IP address or domain.
  
- ▶ **Host-only networking** configures your VM to allow network access only to the host.
  - ▶ This can be useful when you want a secure VM that is connected to the host network, but available only through the host machine.
  
- ▶ **Bridged networking** configures your VM as a unique identity on the network, separate and unrelated to its host.

# VM Networking modes



Virtual Network Editor

Name	Type	External Connection	Host Connection	DHCP	Subnet Address
VMnet0	Bridged	Auto-bridging	-	-	-
VMnet1	Host-only	-	Connected	Enabled	192.168.239.0
VMnet8	NAT	NAT	Connected	Enabled	192.168.150.0

Add Network... Remove Network

VMnet Information

Bridged (connect VMs directly to the external network)  
Bridged to: Automatic Automatic Settings...

NAT (shared host's IP address with VMs) NAT Settings...

Host-only (connect VMs internally in a private network)

Connect a host virtual adapter to this network  
Host virtual adapter name: VMware Network Adapter VMnet8

Use local DHCP service to distribute IP address to VMs DHCP Settings...

Subnet IP: 192 . 168 . 150 . 0 Subnet mask: 255 . 255 . 255 . 0

Restore Default OK Cancel Apply Help