

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**



- ▶ Hypervisor-based virtualization technologies
- ▶ A Linux-based hypervisor: KVM

- ▶ 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/>

The resource virtualization concept (1)



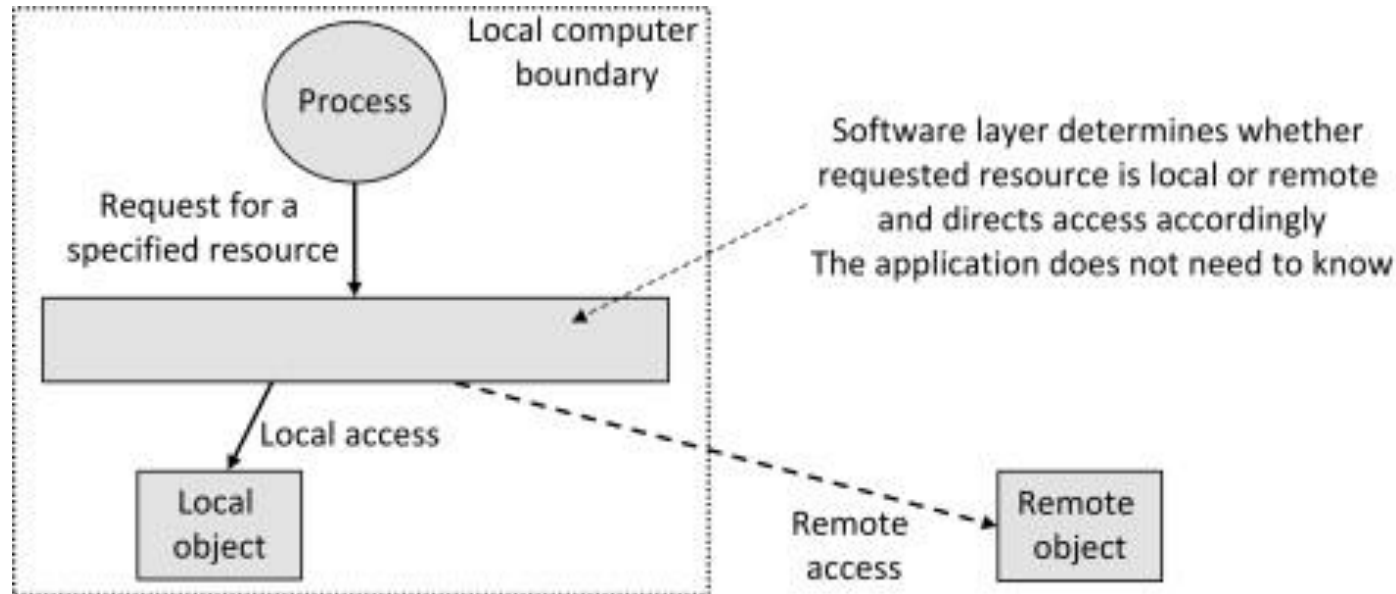
- ▶ In a distributed system, it is likely that the users will routinely use a mix of *resources* and *services*, some of which are locally held at their own computer and some of which are held on or provided by other computers within the system
- ▶ **Access transparency** requires that *objects* (this includes resources and services) are accessed with the same operations regardless of whether they are local or remote
- ▶ In other words, the user's interface to access a particular object should be consistent for that object no matter where it is actually stored in the system
- ▶ A popular way to achieve access transparency is to install a software layer between applications and the operating system that can deal with access resolutions, sending local requests to the local resource provision service and remote requests to the corresponding layer at some other computers where the required object is located
- ▶ This is described as **resource virtualization** because the software layer makes all resources appear to be local to the user

Source: Richard John Anthony, *Systems Programming*, 2016

The resource virtualization concept (2)



- ▶ Resource virtualization: the process makes a request to a service instead of the resource directly
- ▶ The service is responsible for locating the actual resource and passing the access request to it and subsequently returning the result back to the process
- ▶ Using this approach, the process accesses all supported objects via the same call interface, whether they are local or remote, regardless of any underlying implementation differences at the lower level where the resources are stored

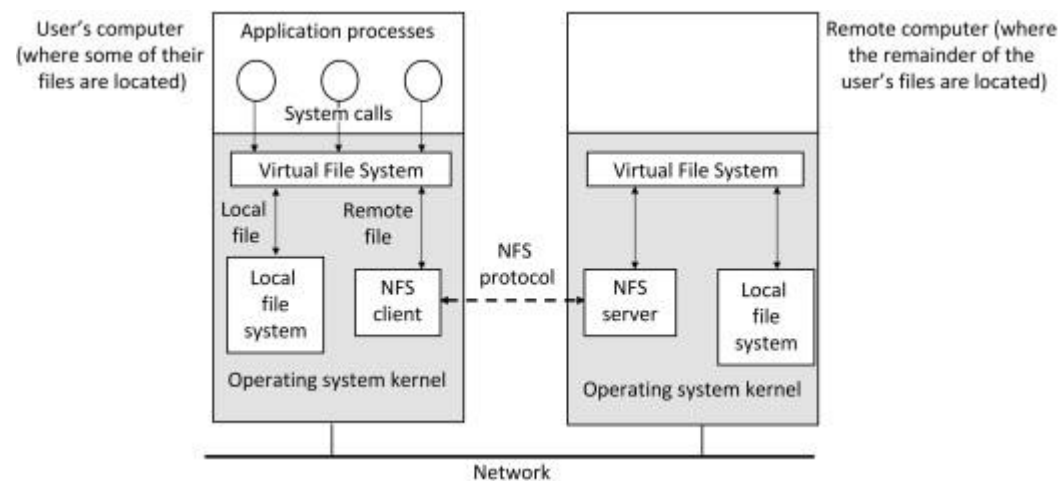


Source: Richard John Anthony, *Systems Programming*, 2016

An example of resource virtualization: VFS



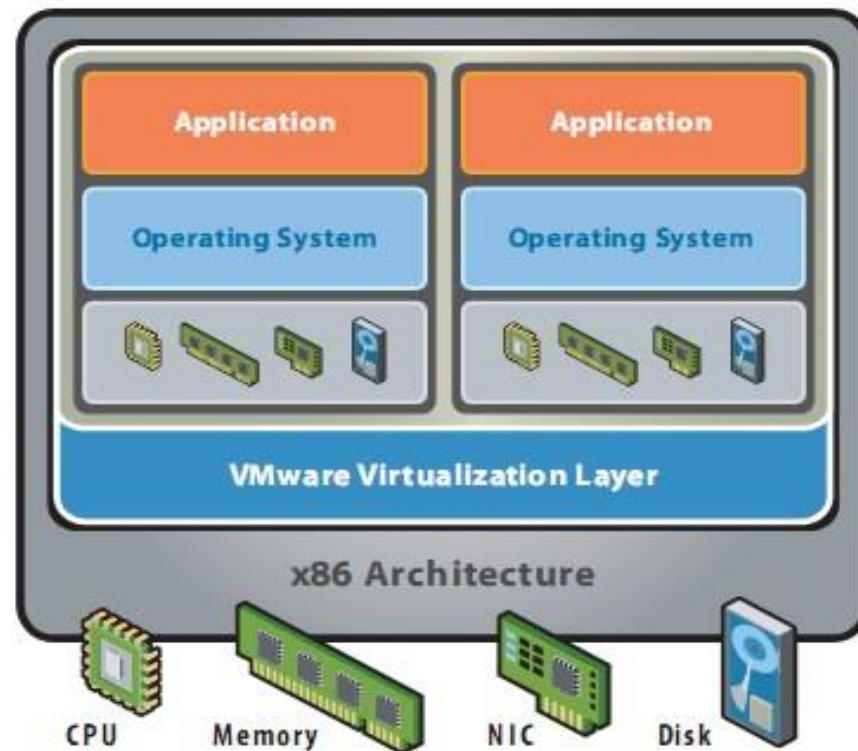
- ▶ An example of resource virtualization is Unix's virtual file system (VFS) that transparently differentiates between accesses to local files that are handled by the Unix file system and accesses to remote files that are handled by the Network File System (NFS)
- ▶ When a request for a file is received, the VFS layer finds where the file is and:
 - ▶ if the file is held locally, it directs the request to the local file storage system
 - ▶ if the file is held on another computer, it directs the request to the NFS client program that makes a request for the file to the NFS server on the appropriate remote computer



Source: Richard John Anthony, Systems Programming, 2016

- ▶ Almost any hardware or software resource may be virtualized:
 - ▶ CPU
 - ▶ RAM memory
 - ▶ Storage capacity of a Hard Disk
 - ▶ Operating Systems
- ▶ The term *hardware virtualization* refers to 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

- ▶ 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



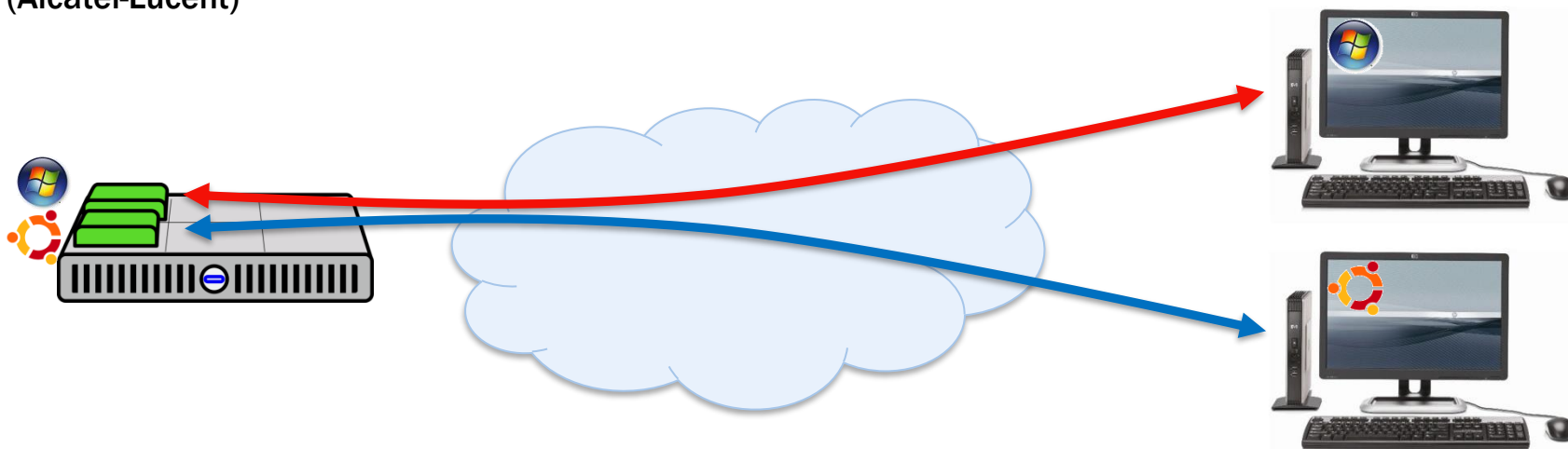
Source:
<http://www.vmware.com/pdf/virtualization.pdf>

- ▶ 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)



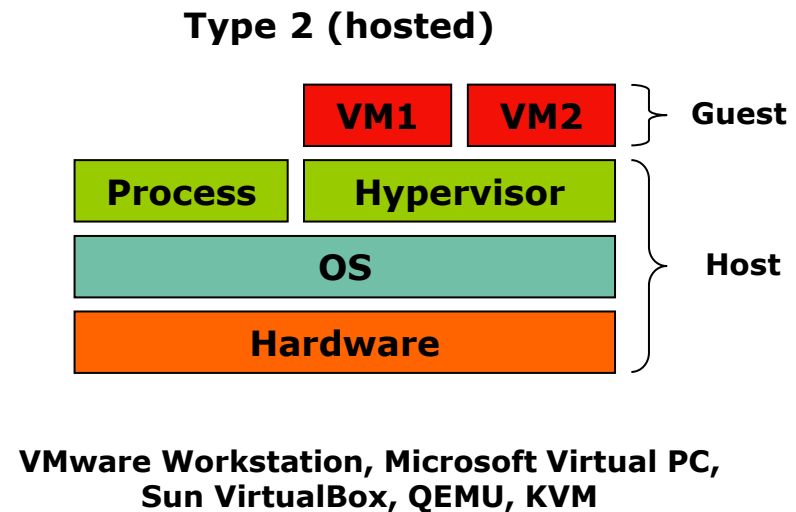
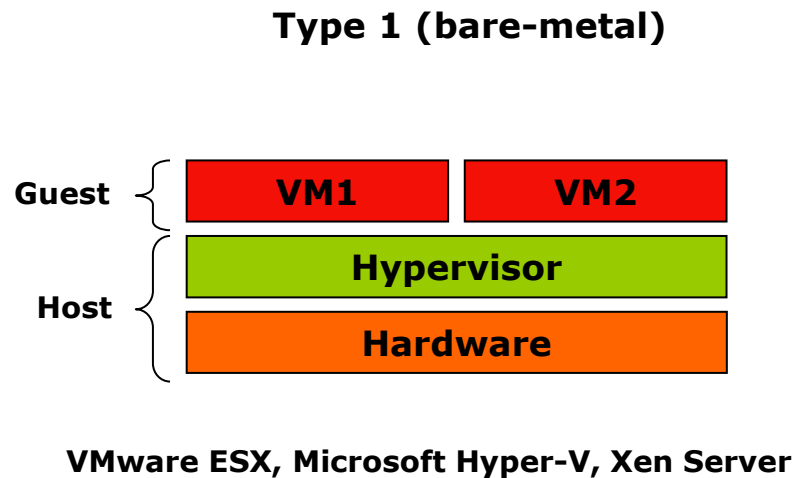
Server virtualization: pros

- 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

Two types of hypervisors



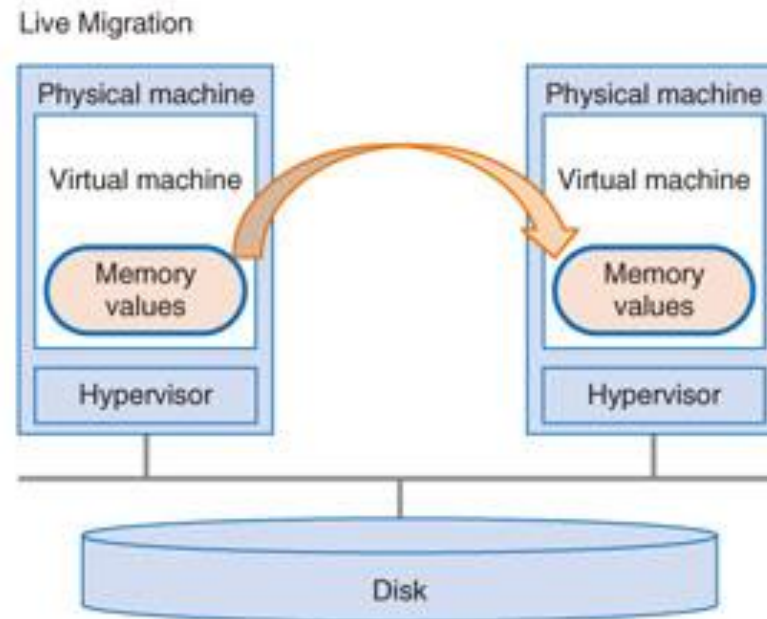
- ▶ A **Hypervisor** (or **VMM** – Virtual Machine Monitor) is a software layer that allows several **virtual machines** to run on a **physical machine**
 - ▶ The physical OS and hardware are called the **Host**
 - ▶ The virtual machine OS and applications are called the **Guest**
- ▶ Two types of hypervisors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



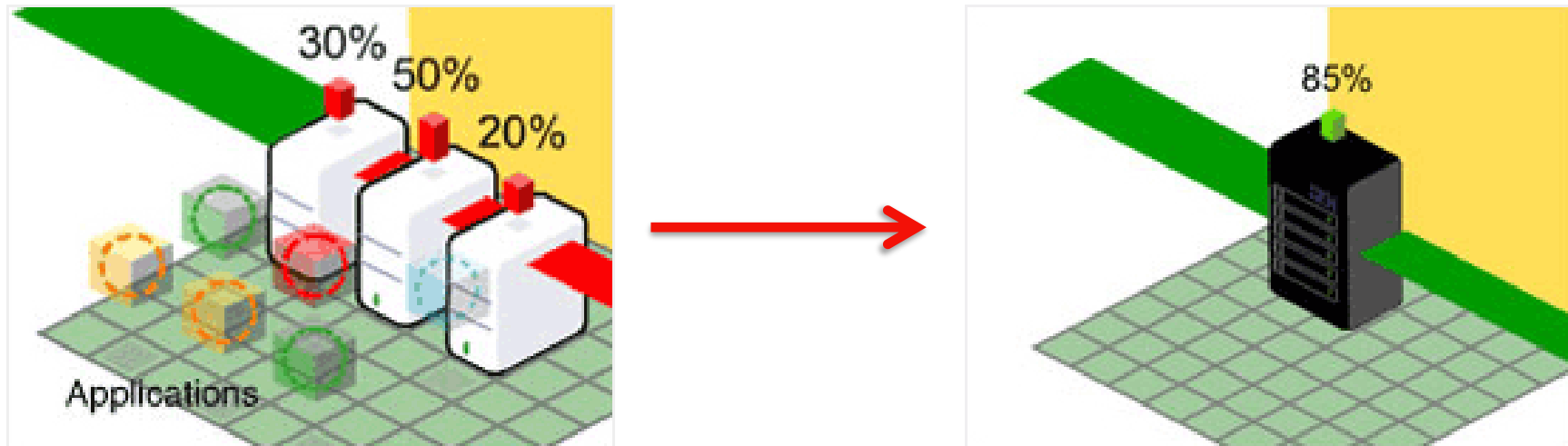
Server Virtualization: live migration



- ▶ Advanced hypervisors allow **Live Migration** of Virtual Machines
- ▶ Live Migration is the operation of moving a running VM from one hypervisor to another one with no effect on availability of services provided by the VM
 - ▶ To perform live migration, the VM's memory needs to be transferred from the original hypervisor to the new one
 - ▶ To avoid transfer of secondary storage, this is usually shared between the two hypervisors



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

▶ 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

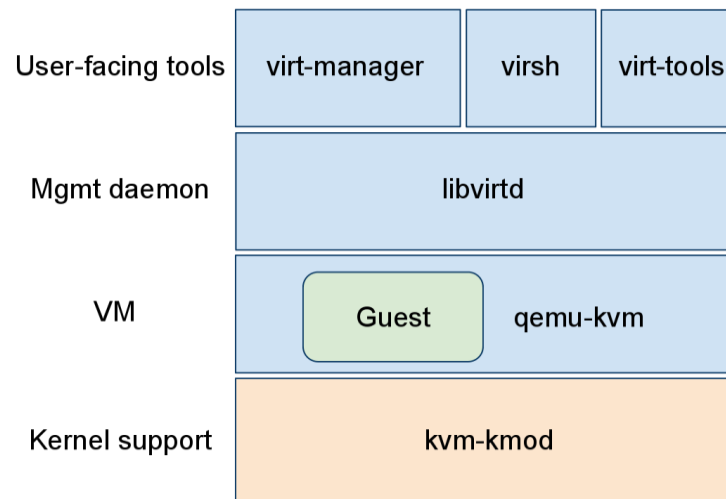
- ▶ The hypervisor should make it possible for the administrator to configure (and possibly change) the amount of virtual resources assigned to a VM
 - ▶ For instance, it should be possible to expand the amount of RAM or change the number of virtual NICs for a VM
- ▶ Possibility of turning on/off VMs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

▶ Ability to guarantee Quality of Service

KVM: Kernel-based Virtual Machine for Linux



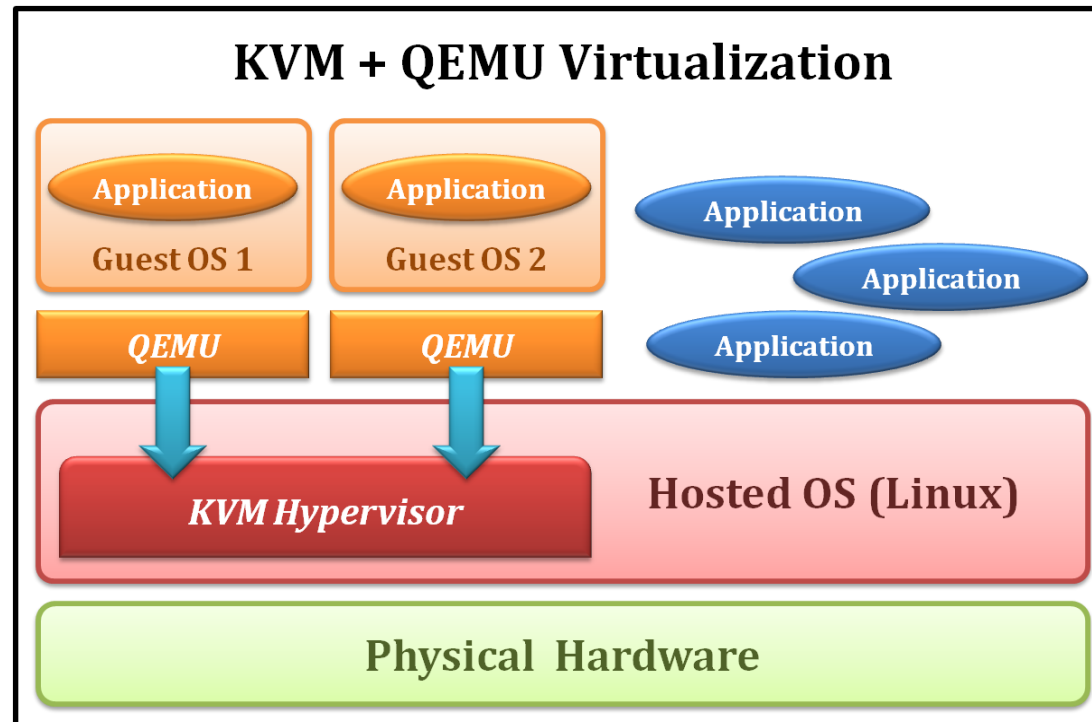
- ▶ KVM is a full virtualization solution that turns a Linux kernel into a hypervisor using a kernel module (kvm.ko)
- ▶ Open source project: <http://www.linux-kvm.org>
- ▶ Implemented in Linux Kernel since 2.6.20
- ▶ The introduction of the KVM is an interesting evolution of Linux, as it represents 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 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

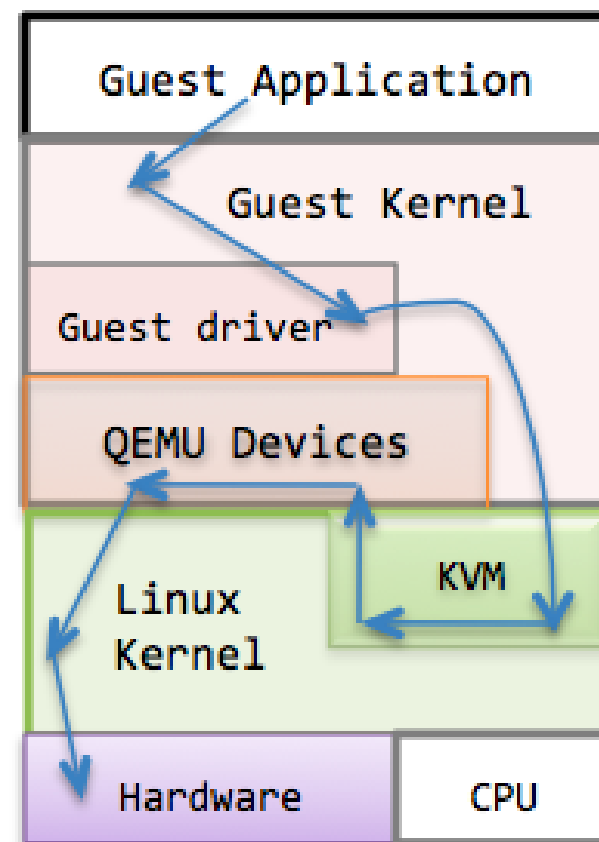
KVM kernel modules (2)

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



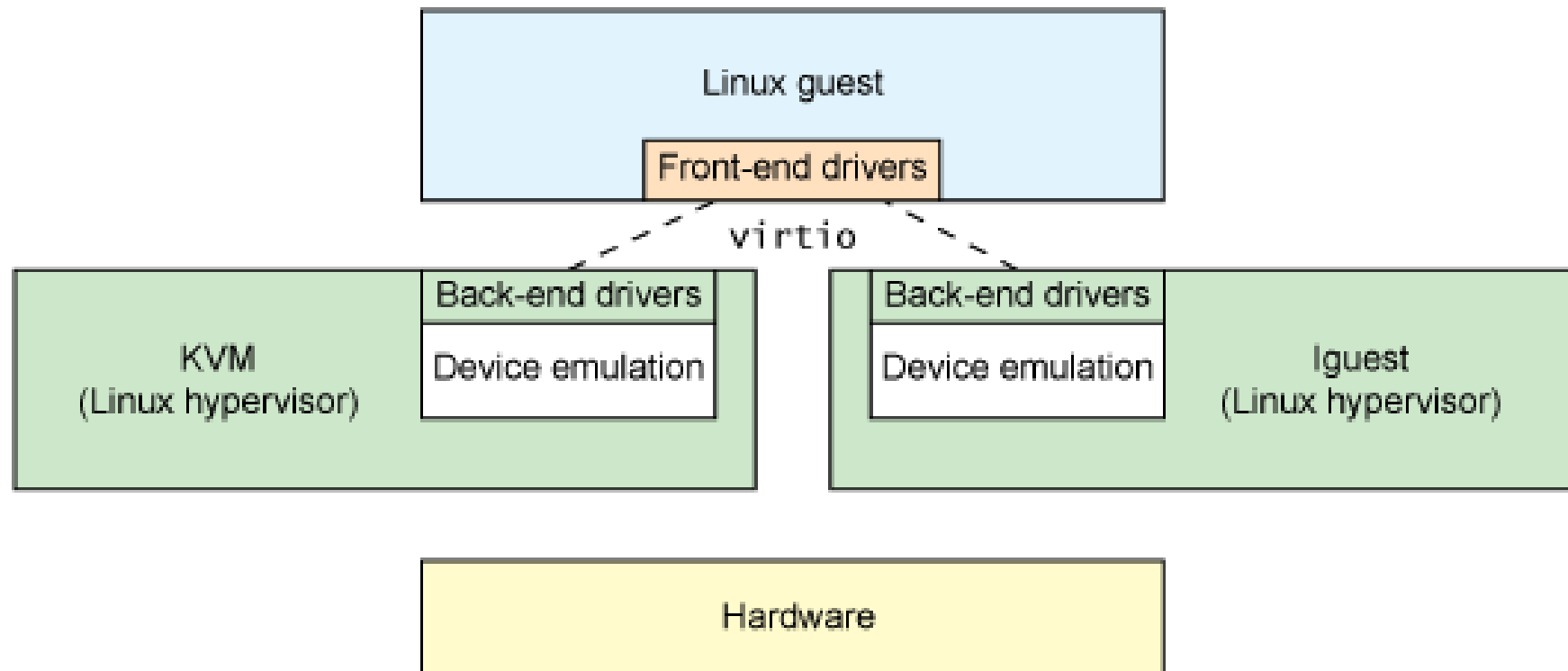
I/O in KVM (full virtualization)

- 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)

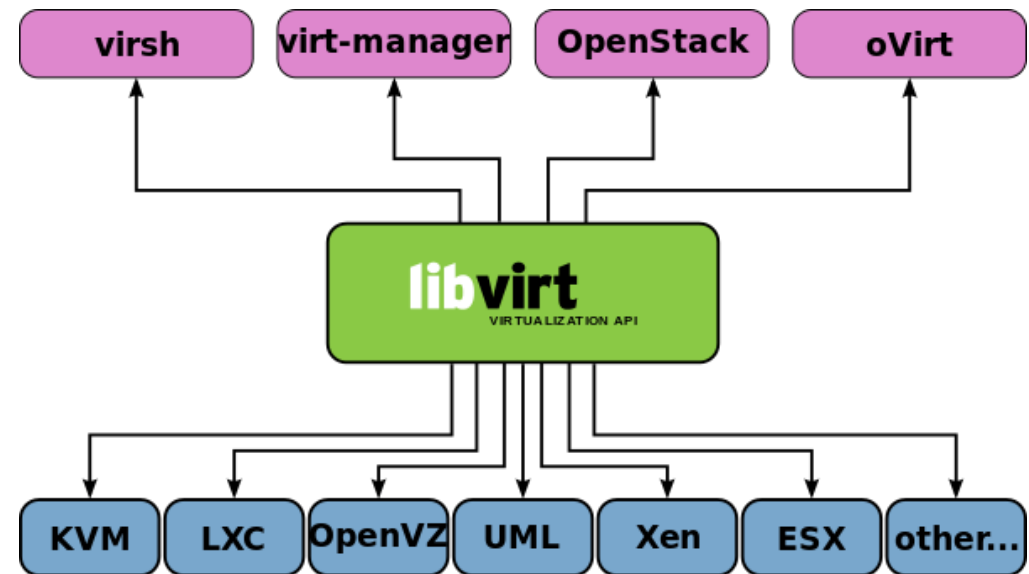
- New approach with para-virtualization



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**

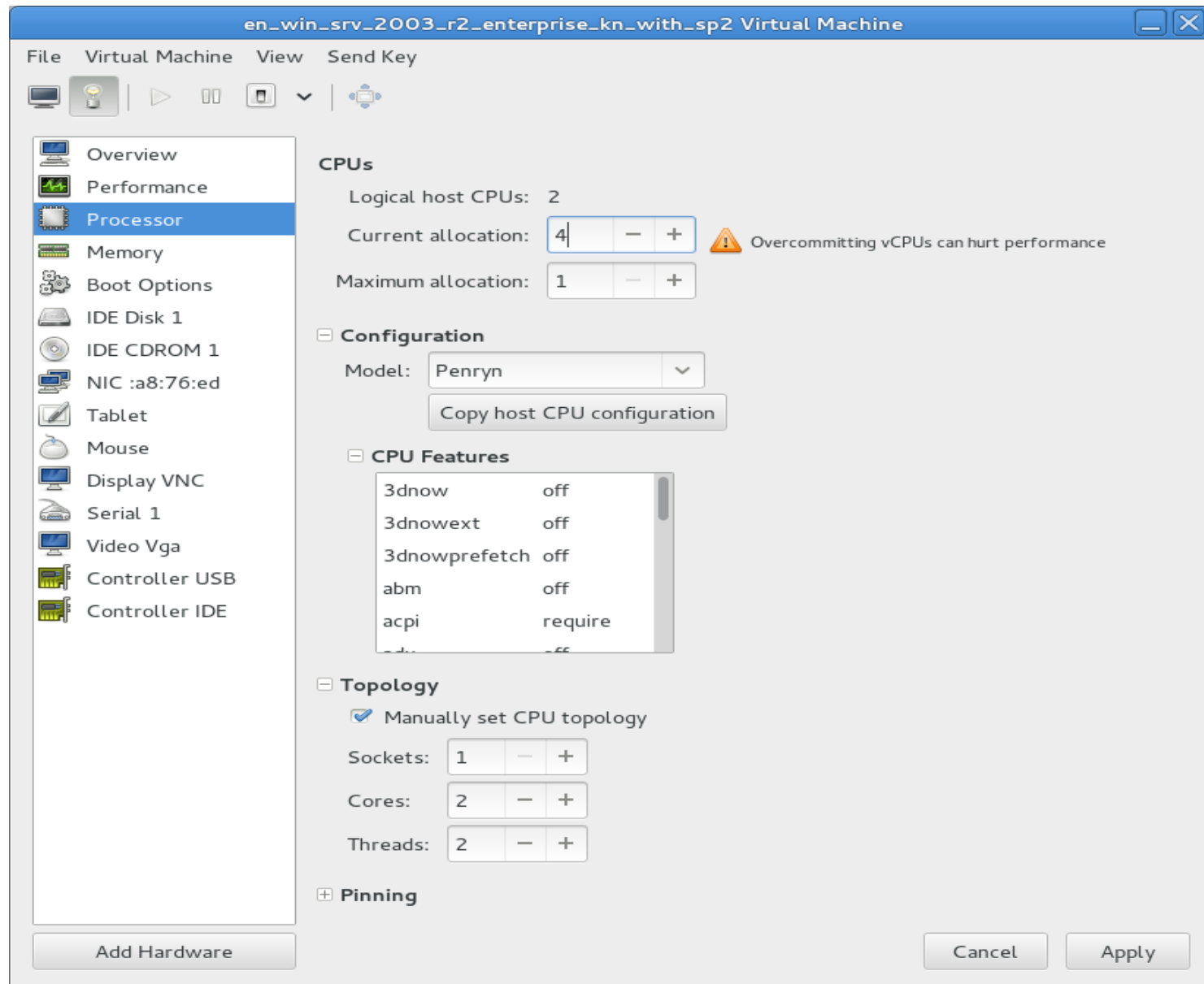
- ▶ is a desktop-driven virtual machine manager with which users can manage virtual machines (VMs)

- ▶ **Functions**

- ▶ create, edit, start and stop VMs







- ▶ **virt-manager's supporting tools**









































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

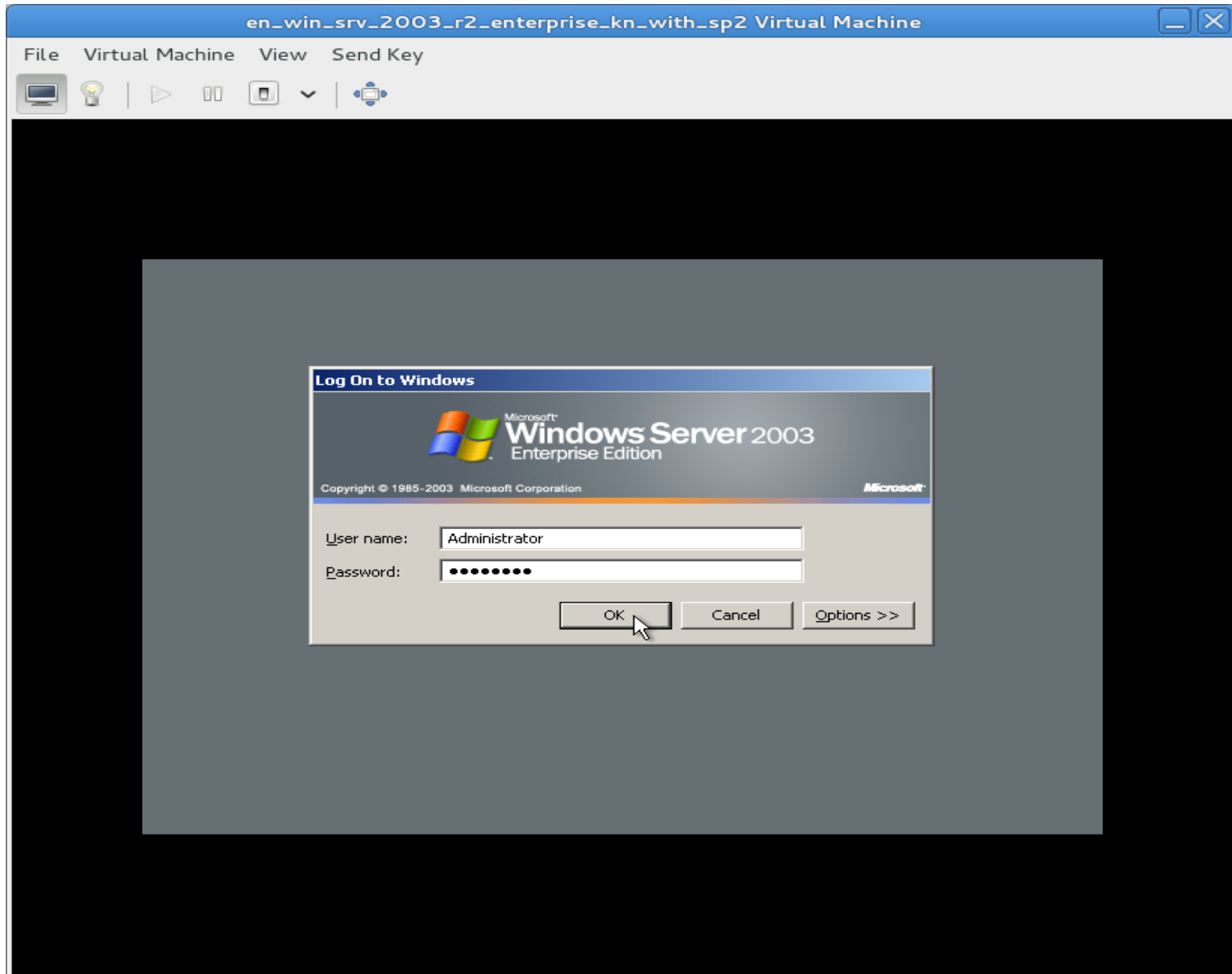


Virtual Machine Manager

File Edit View Help

  Open    

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			

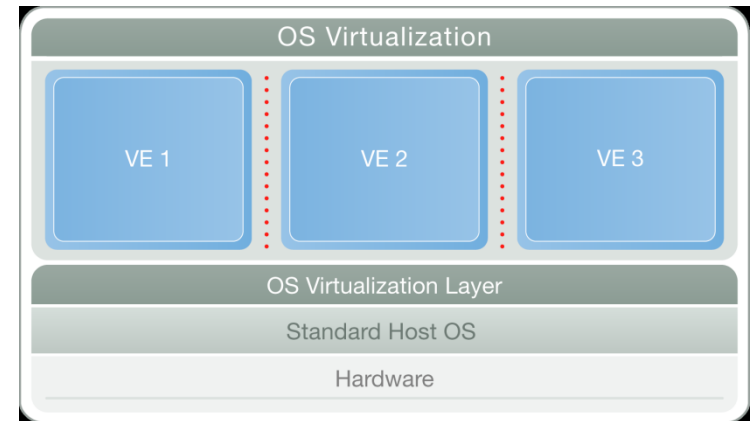


- ▶ **virsh is a command-line tool used to manage domains (i.e. VMs)**
- ▶ **virsh commands need root privileges to be executed**
- ▶ **Common used commands:**
 - ▶ `virsh create`
 - ▶ **start a VM from an XML descriptor file**
 - ▶ `virsh destroy`
 - ▶ `virsh list`
 - ▶ **list all the running VMs**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

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