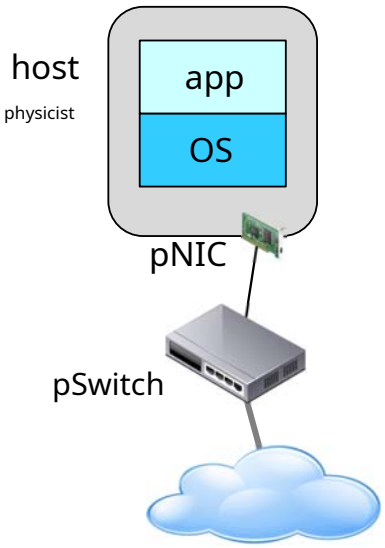
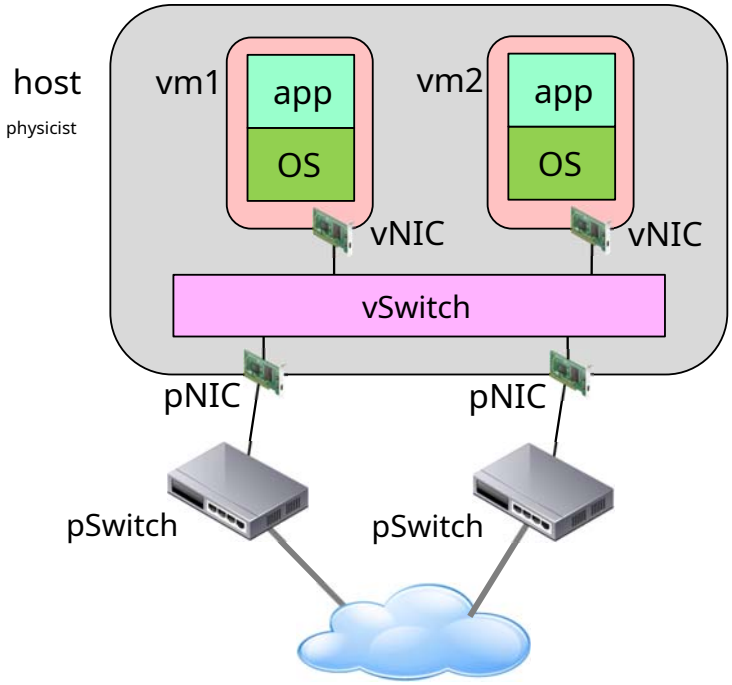
Network virtualization

- A physical network



- A virtual network (with a type 1 hypervisor)



Network virtualization

- Each VM can be equipped with one or more virtual network cards
  - each vNIC can emulate some common real network card - but para-virtualized vNICs (e.g. virtio-net) are often used to aid performance
  - each vNIC can operate in a different virtualization mode (described later)
  - the IP address of a vNIC can be configured statically or dynamically via DHCP (which may be provided by the hypervisor)
  - it is possible to create complex network configurations

Network virtualization mode

- The main modes in which a vNIC can operate - each vNIC can operate in a different mode from the others
  - *NAT* (Network Address Translation)
    - the guest VM sees the external network through the vNIC
  - *Bridged Networking*
    - the vNIC is connected to a pNIC, and exchanges packets with external network (e.g. Internet) directly through it
  - *Internal Networking*
    - to connect a group of guest VMs to each other and create a network of VMs
  - *Host-only Networking*
    - to define a network that contains the host and a set of guest VMs

Network virtualization

- The *port forwarding* matches a port on a guest VM to a port on the host
  - e.g. guest port 80 (HTTP) is connected to host port 8080 - so that guest port 80 can be accessed via host port 8080
  - it is a common way to make services running in a guest VM accessible to the host or external network

- Virtual machine images and instances

- A'*instance* virtual machine (*VM instance*) is an entity dynamic, which has its own state, which can change over time
  - a VM instance may actually be running on a certain host
  - the state of a VM instance includes the state of all its resources, at a certain instant of time - e.g., the state of its disks, memory and registers of its vCPUs
- A'*image* by VM (*VM image*) is instead an entity static
  - a VM image cannot be running
  - a VM image is made up of the VM metadata (e.g., number of vCPUs, amount of memory and MAC address of the network cards), together with the contents of the VM volumes / disks
  - a VM image can be represented by one or more files, in an appropriate format

the instance instead is a running sw?

Virtual machine images and instances

- What is the relationship between VM images and VM instances?
  - a VM instance can easily be created from a VM image
    - e.g., on Amazon EC2, you can create a VM by selecting an instance type (e.g., A1.large) and a VM image (called an AMI, Amazon Machine Images, e.g., the AMI Linux Ubuntu 18.04 for x64)
  - the state of a VM instance can be saved as a VM image - with different purposes
    - eg, to be able to easily create new VMs from that image

- VM cloning

- There *cloning* of a VM is creating a new VM instance from a VM image
  - to avoid installing "from scratch" the OS and the services and applications of interest of a VM
  - it is not a simple copy of a VM image
  - many VM instances can be created from a VM image

Virtual appliance

diff between image(data+metadata) and checkpoint(state of registers)

- A *virtual appliance* is a pre-configured VM image (usually by a third party) from which VM instances with that configuration can be created
  - these images are typically made accessible in a public or private repository, in an appropriate format (e.g., VMware's VMDK or VirtualBox's VDI)
  - the availability of virtual appliances can significantly reduce VM creation times
  - "installing an application, a server or a complex platform is as simple as downloading an app on your smartphone "

- Snapshot / checkpoint

- A VM can be started and stopped - but also paused and restarted

like standby in a real machine

  - the state of a stopped or paused VM can be saved as *snapshot* (or *checkpoint*) for future use
    - this state includes the state of the disk, the state of the memory, and the state of the processor registers
  - it is also possible to start a VM from a snapshot - to reduce the startup time of a VM

- VM migration

Oss: very useful to distribute the systems (failures, speed…)

- There *migration* is intended to move a running VM instance from one physical host to another
  - It is not always acceptable to shut down the VM on the first host and restart it on the second host
  - it may be better to pause the VM, take a snapshot of it, copy it to the second host and restart the VM on the second host starting from the snapshot
    - copying can also be avoided if the snapshot is saved to a SAN / NAS drive shared by the hosts
    - if the VM storage is managed on a SAN / NAS drive shared between the hosts, the snapshot can be limited to just the memory state - and the migration can be extremely fast (*live migration*)

Oss: you have to pay attention to where the disk of the VM is staged: if it's shared you can

(AWS Region shares the drive)

- Interfaces for VM management

- VM management operations - such as creation, configuration, startup and shutdown - can be managed
  - via a GUI or web console - manually
  - through a CLI or REST interface - also through scripts

Cmd line interface: it's a sw program that can be used as a cmd line (in many cases it's simpler to use and:)

    - this supports automated VM management

- Discussion

- Some consequences of system virtualization

now - you can faithfully run an application or service in a VM

oss: - there is a performance overhead - but it can be kept low

adv's - virtualization can sustain some qualities
  - flexibility - for flexible release of distributed software systems in virtual environments
  - security - VMs are isolated from each other and from the host
  - availability - e.g., VM creation and quick startup

something works well in a wide time interval

* Virtualization systems

- We briefly describe some virtualization systems for the x86 platform
  - Xen
  - KVM
  - the VMware family of products
  - Oracle VM VirtualBox

- Xen

the hardware has to be

- Xen is an open source Type 1 hypervisor for x86 systems
  - support for multiple guest OSes, mostly Linux (and other Unix OSes) but also Windows
  - supports both para-virtualization (PV) and hardware-assisted virtualization (HVM)

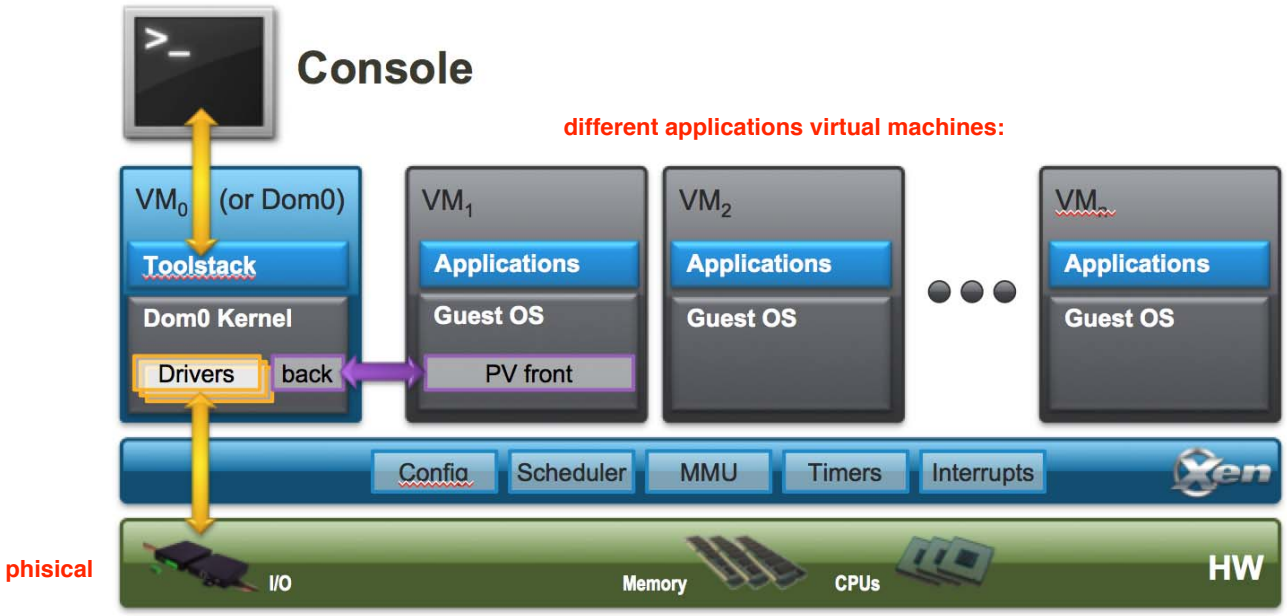
chips on the cpu has to be on some specific hw

  - a research project in the late 1990s, which became an open source project in 2002
  - since 2013, a Linux Foundation collaborative project - members include Amazon, Google, Oracle, Intel and AMD
  - according to Wikipedia, it is used as the primary hypervisor on many systems - including Amazon EC2

Xen

- Xen architecture

VM0 is the the only one that use a kernel (and a toolstack to communicate with it with the console) to communicate to the actual HW

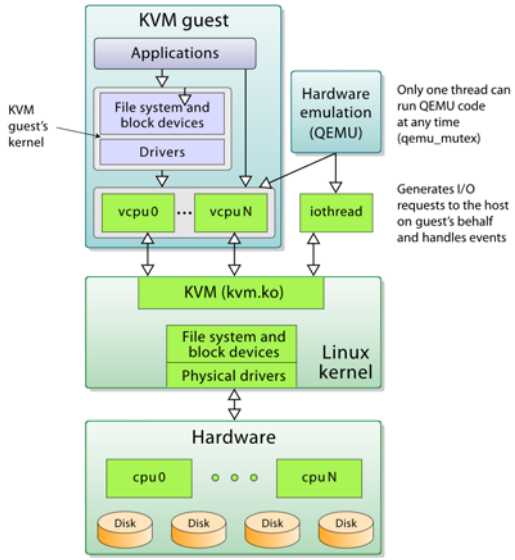


Xen

- Xen architecture
  - based on a thin hypervisor - this supports robustness and security
  - each VM is called a guest or domain
  - domain 0 (or control domain) is a special domain (with special privileges)
    - contains the drivers for the physical hardware, and supports the hypervisor in hardware access
    - contains a control software stack (toolstack) to manage the creation, configuration and destruction of other VMs - which can be accessed from the command line, a graphical interface or other stacks for orchestrating VMs
  - XenServer is a virtualization platform for the cloud based on the Xen hypervisor

- KVM

- KVM (Kernel Virtual Machine) is an open source virtualization solution for x86 systems (with extensions for the virtualization) built into Linux kernels - can be considered a type 1 hypervisor
  - support for unmodified Linux and Windows guest OSes



KVM

- KVM architecture
  - based on a Linux kernel module (kvm.ko) which provides the core of the virtualization infrastructure
  - also QEMU - which is a hosted hypervisor for hardware virtualization (not to be confused with hardware-assisted virtualization) based on binary translation - is used as an environment for running KVM guests
    - where possible, the guest code is executed directly from the host
  - each guest VM vCPU is managed as a host OS thread

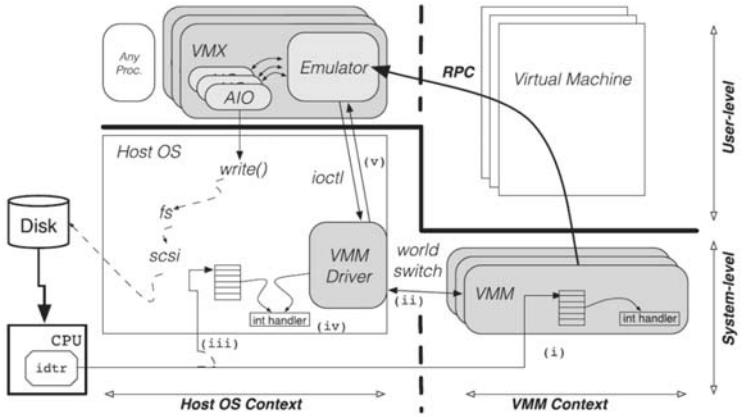
    each VM virtualCPU correspond to a thread on the actual kernel
  - You can interact with the virtualization capabilities of KVM using libvirt - a common API for Linux to manage and control VMs securely and even remotely

- VMware

- VMware is a subsidiary of EMC with a rich offering of virtualization technologies for small, medium and large enterprises, which includes
  - single computer virtualization products
    - like type 2 hypervisors
    VMware Workstation and Fusion (Pro and Player)
  - data center virtualization products
    and cloud management - such as vSphere (a suite of products, with the ESXi type 1 hypervisor and vCenter control plane) and vCloud Suite (includes VM availability, automation and management functions, to provide private cloud)
  - desktop virtualization products - such as Horizon
  - the first version of VMware Workstation was released in 1999, the first version of ESX server in 2001

VMware Workstation

- The VMware Workstation architecture is based on three main components (the figure shows the host OS context on the left and the hypervisor context on the right)
  - VMM (virtual machine monitor) is the hosted hypervisor
  - VMX is the user interface on the host system
  - the VMM driver is installed as a driver in the host OS - but it actually drives the VMM and hides it from the host OS



- VirtualBox

- Oracle VM VirtualBox is a virtualization product for x86 systems for enterprise or personal use (since 2007)
  - a type 2 hypervisor, for Windows, Linux and MacOS host OS, and for Windows and Linux guest OS
  - an open source project controlled by Oracle
  - supports numerous virtualization techniques and options
  - VMs can be created via a GUI or via a command line interface (VBoxManage)

  - VMs can be accessed locally or remotely
  - a common use is that of pre-built VMs for developers
    - you can experiment with complex software stacks by installing only VirtualBox and downloading a single pre-defined virtual appliance

* Applications and benefits of system virtualization

- Let's briefly discuss the applications of system virtualization and its benefits

Virtualization applications

before the use of virtualization sw's the approach to build a distributed system is using different hw's (necessary cause everry service needs a different library/OS to run properly

- Server consolidation

problem: users scalability (differen phisical hw's)

  - consider a distributed software system consisting of multiple services and servers - each server is running on a different (physical) computer
    - there are good reasons to use multiple separate computers
    - it is an expensive and difficult solution to manage - eg it is difficult to size individual computers

then: multiple VM's

  - in *server consolidation* the different servers run in different VMs - on one or more virtualized physical computers
    - system virtualization creates a dynamic infrastructure based on a pool of computational resources
    - the hypervisor guarantees isolation between the different VMs (security risks)
    - this leads to greater use of resources and greater flexibility - but also significant savings
    - hardware reliability deteriorates

oss: more VM's on the same hw's so many other adv's (the hw deteriorates fastly but it's not such a problem)

Virtualization applications

- Other common applications of virtualization

other adv's:

  - provide an execution environment to a legacy application (application consolidation) - e.g., following migration to a new hardware / software platform

still keeping (CONSOLIDATION) some old but well-working applications on new hw's

  - create multiple execution environments, each with their own OS and software stack
    - to support the development of distributed software systems
    - to support testing and quality assurance (QA), in multiple and separate environments

possibility on develop and testing on different but equivalent to final machines

  - run unsafe applications (sandboxing)
  - desktop (client) virtualization
    - allows users to access their virtual desktop from any computer (client)
  - in hosting web services
  - in the context of cloud computing

Benefits of virtualization

- Here are the main benefits of virtualization

other adv's:

  - cost reduction
  - improvement of application quality
    - availability and fault tolerance
    - IT efficiency, agility, productivity and flexibility
    - isolation and safety
    - extend the life of applications
  - simplification of data center management
    - Simplified and speeded provisioning of resources and VMs
    - support for scalability and elasticity
    - centralized management
    - software defined datacenter
  - support for development, testing and QA
  - reduce vendor lock-in and facilitate cloud migration

freedom of changing your hw type

* Virtual machines and software release

for these reason the VM became also the prefered way to release distributed sw's

- VMs are a release option for distributed software systems
  - each VM encapsulates one or more software services, along with the software stack needed for those services

- But how to manage these VMs?
  - "traditional" approach
    - create each VM manually, installing the software and services of interest always manually
    - VMs are simply considered the virtual version of physical computers
  - a modern and better approach
    - automatically build the VM images of interest
    - create VMs from these images - it is also possible to create multiple VMs from each image, to replicate the corresponding services

Benefits

- Benefits of using VMs for software release
  - the release is simple and reliable - the release of a service is managed as the creation of a VM starting from the image related to that service
  - fault isolation and safety - each VM (with related services) runs in isolation
  - VMs can be released both on the cloud and on premises, in a private data center
  - creating and starting a VM (starting from a VM image) typically takes from a few seconds to a few minutes (less than a physical computer)

Drawbacks

it's possible to use the MVs inefficiently (=> contaneraization)

- Drawbacks of using VMs for software release
  - inefficient use of resources is possible - each service or group of services requires an entire VM eg the operation team underestimated the scale
    - this inefficiency increases if each VM is used for a single lightweight service - but releasing multiple services in one VM reduces fault isolation
  - VM system administration overhead - whoever creates the VM (or its image) is also responsible for updating the software installed there
  - creating and starting a VM (starting from a VM image) typically takes from a few seconds to a few minutes (more than a container)

* Discussion

- System virtualization allows a "real" computer to host multiple "virtual" computers (machines) - each VM can be used to run its own OS and its own services and applications
  - System virtualization relies on various virtualization techniques - to virtualize different computational resources
  - system virtualization has numerous applications, and offers several benefits
  - in particular, it favors the definition and management of virtual execution environments - on premises and in the cloud - with the aim of optimizing the use of hardware resources, providing operational flexibility, as well as isolating applications and environments from each other