Virtualization

Boris Teabe

boris.teabe@inp-toulouse.fr

In this class we will talk about virtualization.

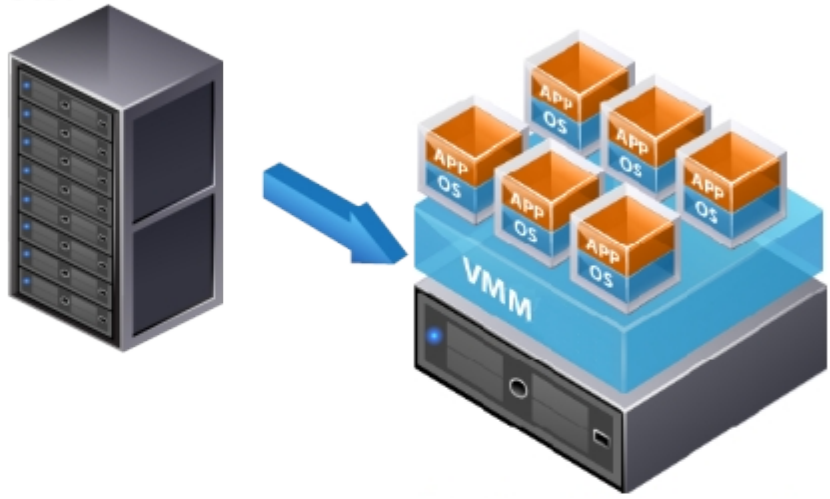
Virtualization: motivations

- Reminder: costs reduction in the cloud
  - For the client
    - Equipments, administration, place, energy, licences ...
  - For the provider (data center)
    - Do more with less (profits)
    - Especially energy
- If we sell physical machines, we mutualize
  - Buildings, equipments (network, UPS, cooling)
  - Remark: licencing is a touchy issue
- We can mutualize machines by giving the illusion of several machines on top of a single machine (we sell machines in a IaaS)

This is a reminder of the last class.

Virtualization: definition

- Set of techniques, hardware and/or software, which allow managing simultaneously on a single machine several operating systems (called virtual machines (VMs)). Examples: Xen, VMware, KVM, HyperV, etc.



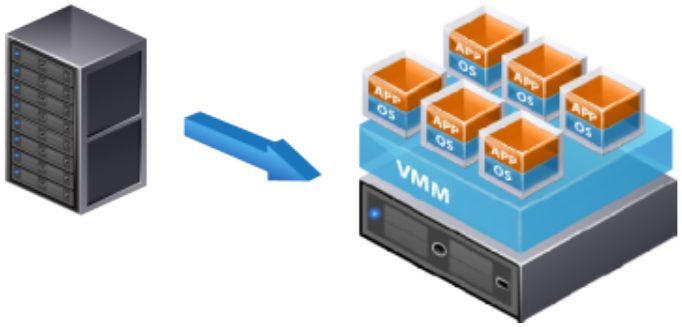
VMM = Virtual Machine monitor

- Concretely
  - I execute at the same time several OS on the same hardware

Virtualization is the process of running a virtual instance of a computer system in a layer abstracted from the actual hardware. Most commonly, it refers to running multiple operating systems on a computer system simultaneously. To the applications running on top of the virtualized machine, it can appear as if they are on their own dedicated machine, where the operating system, libraries, and other programs are unique to the guest virtualized system and unconnected to the host operating system which sits below it.

Virtualization: challenges

- Mainly isolation
  - Security: a VM is protected against potential attacks from other VMs
  - Performance: one VM's performance is not affected by other VMs
  - Failure: one VM's failure should not affect other VMs



Virtualization: implementation

- Different types of virtualization systems (VMM)

- Full virtualization
- OS level virtualization
- Para-virtualization
- Hardware assisted virtualization

Types of VMM

- **Full virtualization**: an OS executes applications at user level. The VMM is one such application. Every instruction from VMs is emulated by the VMM. The OS executed on a VM is not modified and can be of any type (Linux, Windows, etc.). Ex: VirtualBox

The VMM can enforce that every instruction respects isolation

- Concretely:
  - It's hardware simulation (slow)
  - Supports several OS types

Full Virtualization. Virtual machine simulates hardware to allow an unmodified guest OS to be run in isolation. Guest operating system's source information will not be modified. It completely relies on binary translation to trap and virtualize the execution of sensitive, non-virtualizable instructions sets. It emulates the hardware using the software instruction sets. Due to binary translation, it often criticized for performance issue. Here is the list of software which will fall under software assisted (BT).

Types of VMM

- **OS level virtualization**: the host OS includes mechanisms for building isolated containers (VMs). These containers share the same OS (host). The VMM is integrated in the host OS. Ex: openVZ, chroot, Docker, LXC, etc.

The modified host is the VMM and it enforces isolation between instances

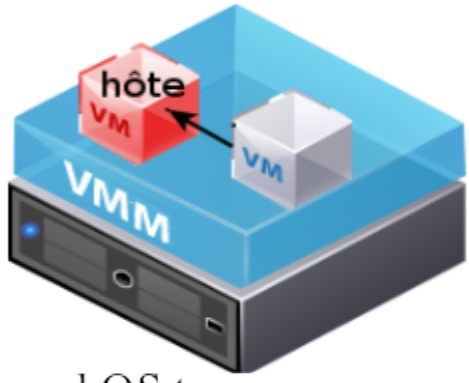
- Concretely:
  - The host OS is modified to managed several instances of itself
  - Native code (efficient)
  - Partial isolation (a single resource management system) and a single OS type

OS-level virtualization is an operating system paradigm in which the kernel allows the existence of multiple isolated user space instance may look like real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can see all resources (connected devices, files and folders, network shares, CPU power, quantifiable hardware capabilities) of that computer. However, programs running inside the isolated user space instance can only see the contents and devices assigned.

Types of VMM

- **Para-virtualization** (PV): The VMM replaces the host OS and behaves as a proxy for accessing the hardware. The host OS is considered as a VM (privileged) and it is used by the VMM to perform particular tasks. Constraint: VMs' OSes have to be modified (to invoke the VMM for privileged operations). Ex: Xen, VMware, etc.

The VMM (small) enforces isolation between VMs



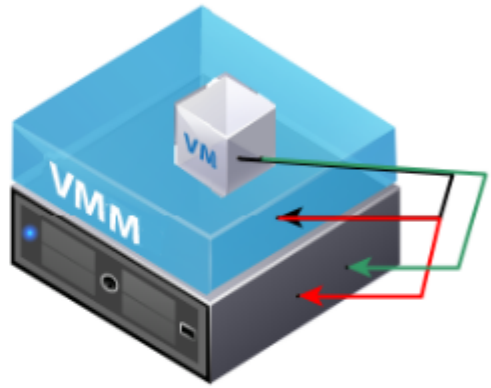
- Concretely:
  - A mix between simulation and native. Several OS types
  - Each VM is allocated hardware resources and the VMM controls access to hardware
  - hypercalls (a modified OS invokes the VMM)

Paravirtualization. It doesn't need to simulate the hardware for the virtual machines. The hypervisor is installed on a physical server (host) and a guest OS is installed into the environment. Virtual guests aware that it has been virtualized, unlike the full virtualization (where the guest doesn't 't know that it has been virtualized) to take advantage of the functions. In this virtualization method, guest source codes will be modified with sensitive information to communicate with the host. Guest Operating systems require extensions to make API calls to the hypervisor.

Types of VMM

- **Hardware assisted virtualization** (HVM): the hardware is aware of virtualization. VMs' OSes don't have to be modified. Ex: Xen, VMware, KVM etc.

The hardware is extended to help virtualization



- Concretely:
  - Native, but without OS modification
  - The hardware is able to manage several VMs (MMU, network device …)
  - Sometimes, para-virtualization is faster or more flexible (hybrid mode PV-HVM)

Hardware-assisted virtualization eliminates the binary translation and it directly interrupts with hardware using the virtualization technology which has been integrated on X86 processors since 2005 (Intel VT-x and AMD-V).

Virtualization: the case of Xen

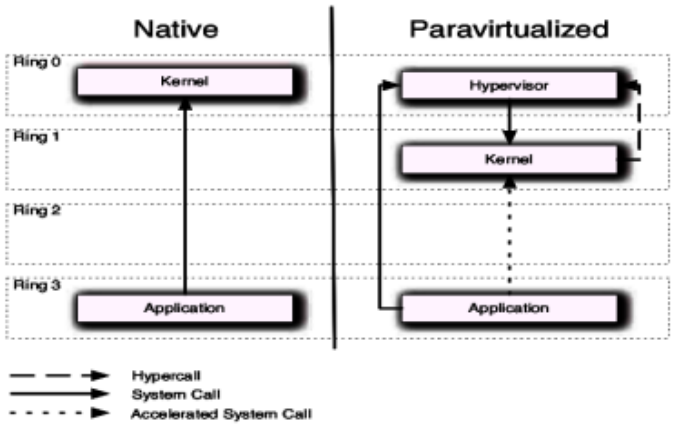
- Open source virtualization system
  - Large community (including AWS), with associated scientific conferences
  - One of the most efficient (~3% overhead)
- Xen is a hypervisor (VMM)
  - Kind of kernel (a small Linux), executed on the bare hardware
  - Replaces the traditional kernel
- Xen provides 3 virtualization modes
  - Para-virtualization (PV) (to be detailed)
  - Hardware assisted virtualization (HVM)
  - A combination of both (PV-HVM)

Xen is a hypervisor that runs directly on the system hardware. Xen inserts a virtualization layer between the system hardware and the virtual machines, turning the system hardware into a pool of logical computing resources that Xen can dynamically allocate to any guest operating system.

Para-virtualization with Xen

- At startup of a physical machine
  - The BIOS provides to the kernel information about the hardware
    - Notably the memory size
    - Memory is seen as a contiguous space
  - The kernel is loaded into memory
  - It initializes its data structures
  - The "init" program is started

But a VM starts as a physical machine. How can it follow the same process ?

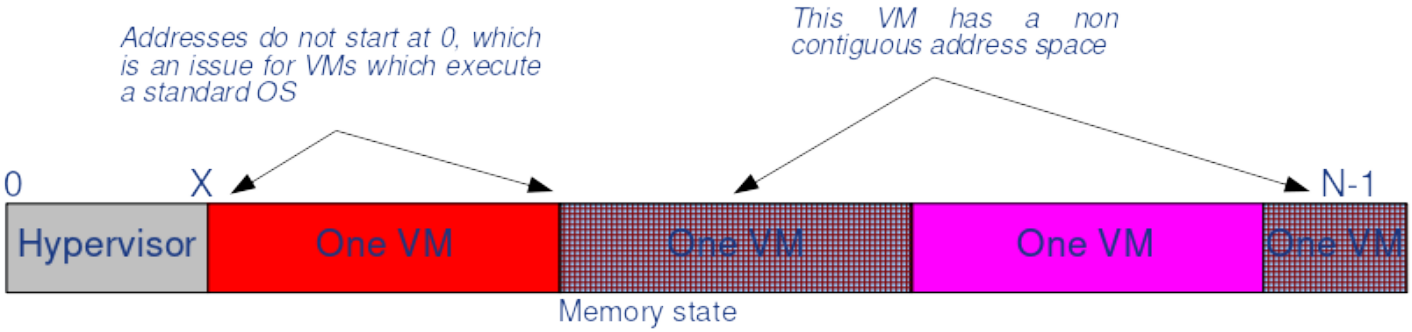


From: The Definitive Guide to the Xen Hypervisor

The BIOS (basic input/output system) is firmware used to perform hardware initialization during the booting process, and to provide runtime services for operating systems and programs.

Para-virtualization with Xen

- Several challenges:
  - How to provide to VMs different BIOS versions ?
  - How to provide to each VM a contiguous memory while memory may be fragmented ? What about paging, TLB, MMU ?
  - How to enforce that a VM cannot access the memory from another VM ?



Para-virtualization with Xen

- Several challenges:
  - How to fairly shared the processor resource?
  - And other devices ? (network, disk, etc)
  - How to, how to, how to????…

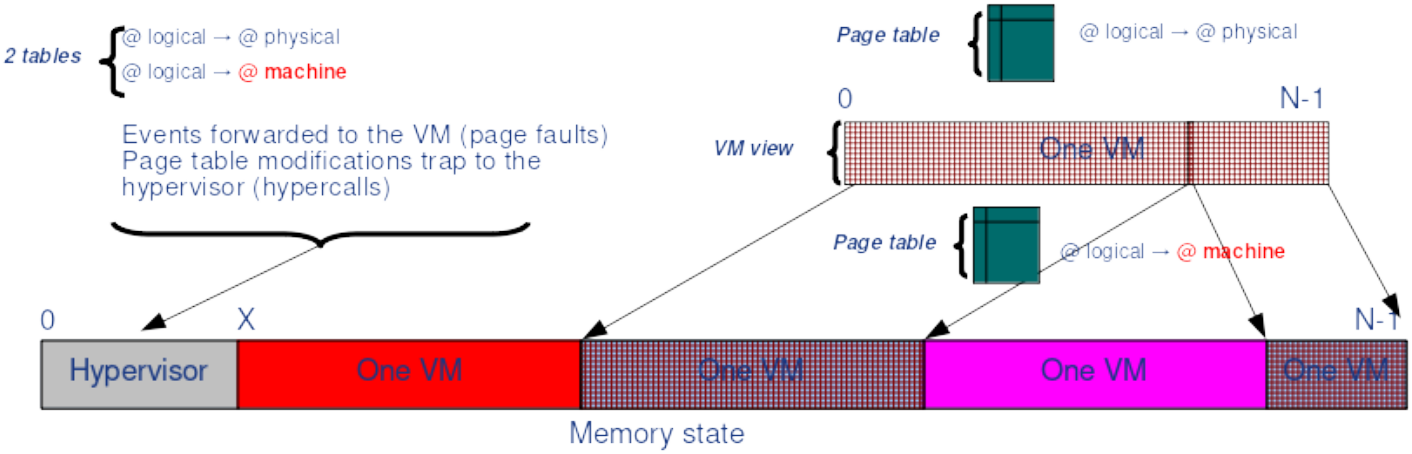
In summary, implementing a virtualization system is much tricky

Para-virtualization with Xen: BIOS and CPU

- For the BIOS
  - The hypervisor provides adapted data to each VM
  - Through a shared data structure
  - The OS executed on the VM is modified to used this data structure
- For the CPU
  - A VM is given a set of vCPUs (virtual CPU)
  - vCPUs are scheduled on pCPUs (physical CPU)
  - Similar to user-level threads

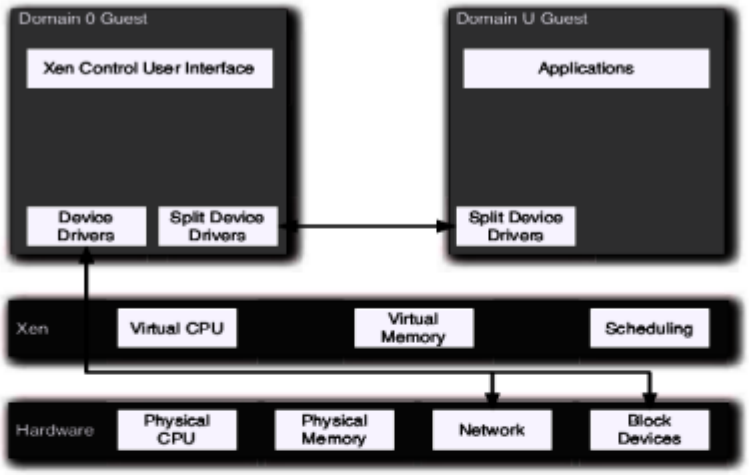
Para-virtualization with Xen: memory

- For memory, the hypervisor
  - Provide to the VM the illusion of a contiguous memory (virtual pages)
  - Implements a kind of virtualized MMU (with address translations)



In order to virtualise the memory subsystem all hypervisors introduce an additional level of abstraction between what the guest sees as physical memory (often called pseudo-physical in Xen) and the underlying memory of the machine (machine addresses in Xen). This is usually done through the introduction of a Physical to Machine (P2M) mapping. Typically this would be maintained within the hypervisor and hidden from the guest Operating System through techniques such as the use of Shadow Page Tables.

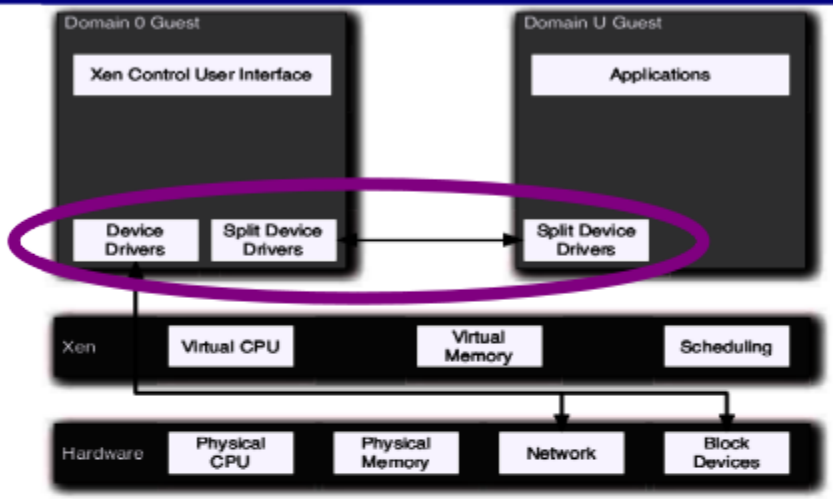
Para-virtualization with Xen: devices



- For other devices, Xen relies on the kernel which was initially on the physical machines (dom0)
  - The dom0 is a privileged VM
  - Other VMs are called domU (executing guest OS)
- The dom0
  - Provides the Xen administration command line
  - Provides drivers for other devices (ex. network, disk)

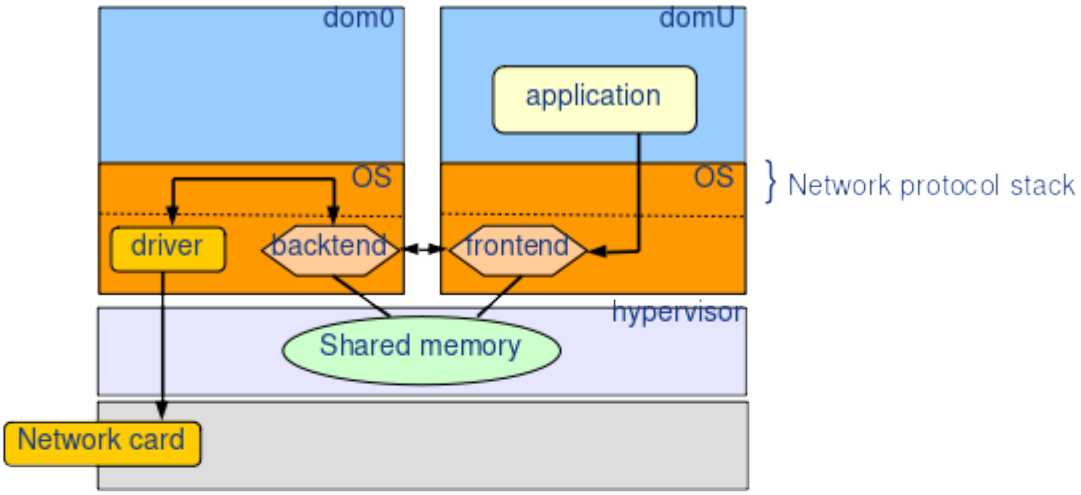
The Split Driver model is one technique for creating efficient virtual hardware. One device driver runs inside the guest Virtual Machine (aka domain U) and communicates with another corresponding device driver inside the control domain Virtual Machine (domain 0). This pair of codesigned device drivers function together, and so can be considered to be a single "split" driver.

Para-virtualization with Xen: devices



- Xen defines a generic mechanism for accessing device drivers
  - Based on "splits drivers"
  - A frontend (in the VM) implements a fake device driver
  - A backend (in the dom0) represents the VM in the dom0
- Frontend and backend behave as a client/server program

Networking in Xen



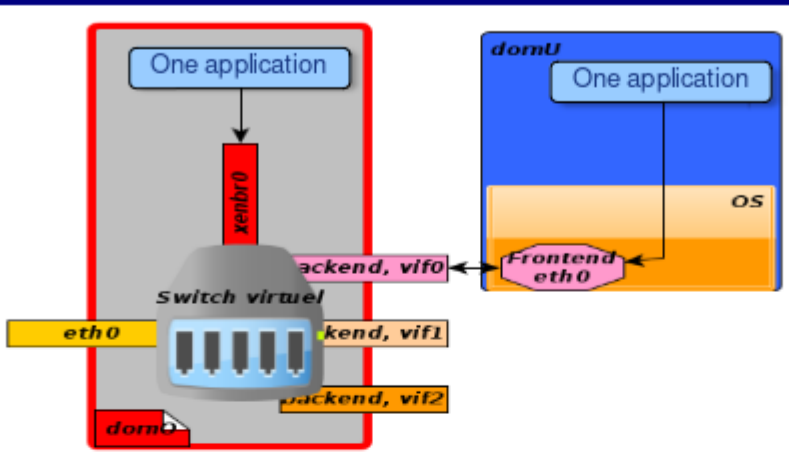
- The real driver is installed in the dom0
- The backend sends/receives packets to/from the driver
- The backend and the frontend interoperate via 2 mechanisms:
  - Shared memory
  - Signals

Making VMs available on the network

A physical machine

- 3 ways to configure networking
  - Route mode: IP routing between VMs
  - NAT mode: address translation as with local IPs
  - Bridge mode (to be detailed) : MAC level management

Xen and bridge mode networking



- All interfaces (MAC) in a VM have a representant in the dom0
  - Managed by an instance of the backend (vif)
- In this mode, a bridge binds all representants
  - Reception: eth0 in promiscuous mode (all MAC @), redirection to the destination backend
  - Emission: redirected to a local MAC @ or eth0
- VMs are therefore accessible from the outside

The default (and most common) Xen configuration uses bridging within the backend domain (typically domain 0) to allow all domains to appear on the network as individual hosts.

In this configuration a software bridge is created in the backend domain (domain 0). The backend virtual network devices are added to this bridge along with an physical Ethernet device to provide connectivity off the host.

Starting a Xen VM

- Have a installation of Xen on the physical machine
- Build the image of the VM
  - The file system which includes the OS to be executed by the VM. Xen provides tools for building this image.
- Edit the VM configuration file
  - # processors (called vCPU)
  - Quantity of RAM
  - # network interface
- Use the Xen command line to start the VM

Exemple of a configuration file:

```
kernel = "/usr/lib/xen-4.0/boot/hvmloader"
type='hvm'
memory = 4096
vcpus=4
name = "myVM"
vif = ['bridge=xenbr0']
disk = ['phy:/dev/vg0/windows,hda,w','file:/root/windows.iso,hdc:cdrom,r']
acpi = 1
device_model_version = 'qemu-xen'
boot="d"
sdl=0
serial='pty'
vnc=1
```

Advantages of virtualization

- Migration
  - Moving VMs between physical machines
  - Suspension and memory copy (copy on write)
  - Without service interruption
- Backup and recovery
  - OS checkpointing
- Virtualisation is now efficient (used by the HPC community)

Migration. The movement of VMs from one resource to another, such as from one physical host to another physical host, or data store to data store, is known as VM migration. There are two types of VM migration: cold and live. Cold migration occurs when the VM is shut down. Live migration occurs while the VM is actually running.

Virtualization: motivations

- Reminder: costs reduction in the cloud
  - For the client
    - Equipments, administration, place, energy, licences ...
  - For the provider (data center)
    - Do more with less (profits)
    - Especially energy
- If we sell physical machines, we mutualize
  - Buildings, equipments (network, UPS, cooling)
  - Remark: licencing is a touchy issue
- We can mutualize machines by giving the illusion of several machines on top of a single machine (we sell machines in a IaaS)