Virtualization techniques

Samia Bouzefrane & Ivan Boule
Conservatoire National des Arts et Métiers
http://cedric.cnam.fr/~bouzefra

samia.bouzefrane@lecnam.net

Outline

- Introduction
- History
- Virtualization categories
- Virtualization in more details
  - Full Virtualization
    - Virtualization of the CPU
    - Virtualisation of memory & MMU
  - Para-virtualization
  - Hardware-assisted virtualization

Introduction

Definition

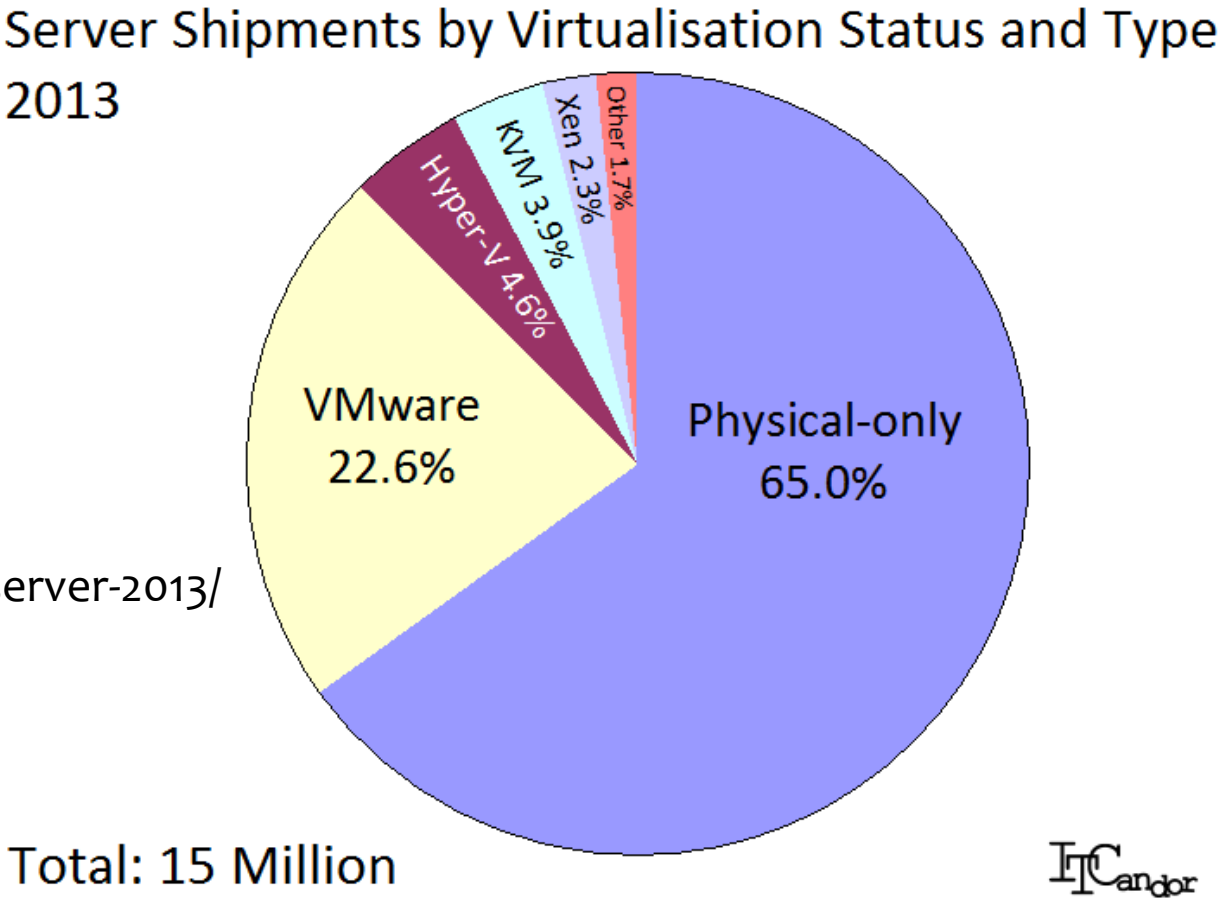
- Virtualization, in computing, refers to the act of creating a **virtual** (rather than actual) version of something, including but not limited to a **virtual computer hardware** platform, **operating system** (OS), **storage device**, or computer **network resources**.

- Source: Wikipedia
  http://en.wikipedia.org/wiki/Virtualization

Advantages

- Sharing physical resources (CPU, RAM, Disk)
- Simplify maintenance
- Reduce the cost in terms of energy and hosting
- Deployment and administration are centralized
- Flexibility to manipulate VMs that are files easy to migrate and to store

Virtualization in 2013



http://www.itcandor.com/server-2013/

History

IBM 370 model

Virtual memory generalized in 1973
(http://fr.wikipedia.org/wiki/IBM_360_et_370)

Some significant dates

- VM introduces in the 60's IBM/370 models
- 1979 : chroot (version 7 Unix)
- 1982 : chroot (BSD)
  - chroot environment allows to create a separate file system
- 1999 : VMware (virtualization for x86)
- 2000 : FreeBSD Jail (*BSD)
- 2003 : Xen (Linux)
- 2005 : Solaris Zones
- 2005/2006 : Intel-VT et AMDV (hardware-assisted virtualization)
- 2006 : OpenVZ (Linux)
- 2007 : KVM (Linux)
- 2008 : LxC (Linux)
- 2008 : HyperV (Microsoft)

Principles of virtualized systems

- Run distinct OSs on the same physical machine
- Share and partition the machine resources between the guest OSs
  - CPU
  - Physical Memory & MMU (Memory Management Unit)
  - Input/output devices
- By design, an OS has a complete control on the physical resources of the machine

Virtualization categories

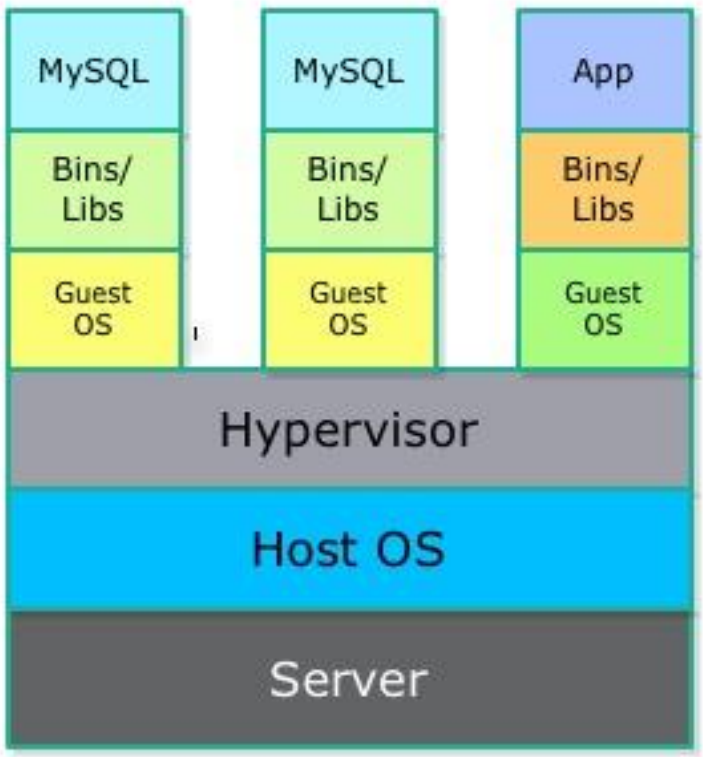
Virtualization categories

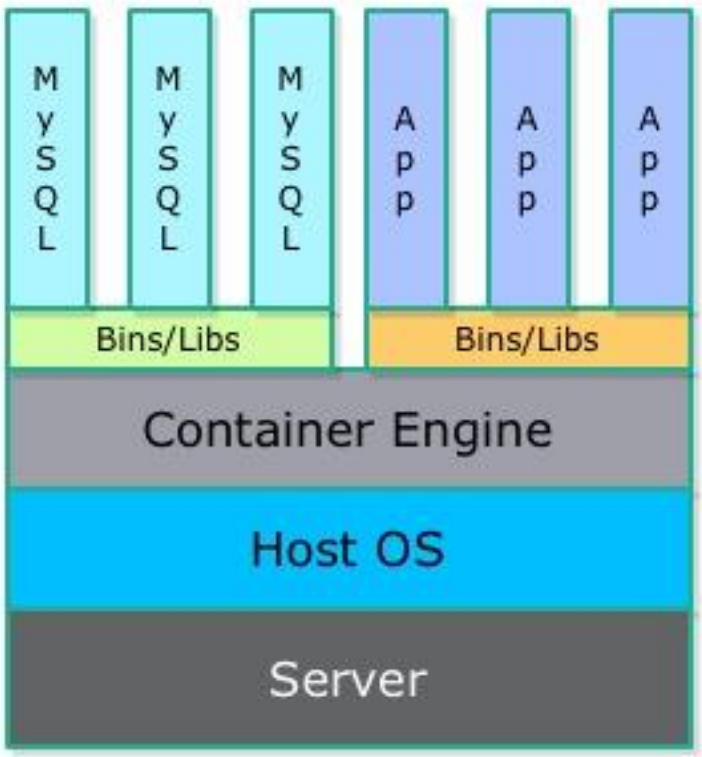
- Virtualization by isolation (container)
- Full virtualization
- Para-virtualization
- Hardware-assisted virtualization

Isolation

- Instanciation of multiple « user spaces » on the same kernel, isolated from each other thanks to name spaces
- Virtualization of the applications (not the systems)

Isolation





http://patg.net/containers,virtualization,docker/2014/06/05/docker-intro/

Example of containers

Jails

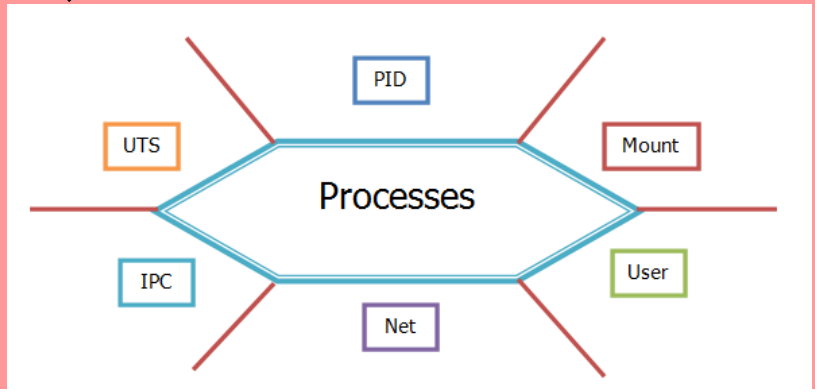
OpenVZ

LxC : Linux Containers/Dockers

https://www.docker.com/whatisdocker/

Docker Namespaces: isolation at each level

- Mount namespaces: mount file systems (chroot)
- Pid namespaces: Pid of processes are attached to Pid namespaces
- IPC namespaces: IPC are defined in each namespace
- Network namespaces: @IP, routing table
- UTS namespaces: domain name, hostname
- User namespaces: not implemented yet

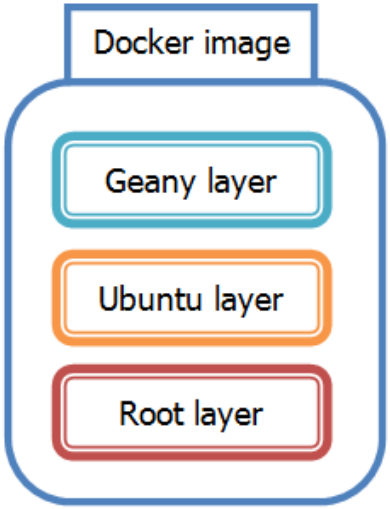


Images in Dockers

- Example of image
  - A minimum kernel (ex. Ubuntu)
  - Application called Geany

- Eache Docker image has a unique ID
  - Binary image contains a basic OS

  (ex. Centos, Ubuntu, etc.)



Docker Container

- Docker Container
  - Is a directory
  - Hosts a binary image

- Docker Daemon: is a daemon that manages containers

See http://cedric.cnam.fr/~bouzefra/cours/Exercice_sur_les_Dockers.pdf

Full Virtualization

- Provides a virtual environment (interfaces and resources) to represent a real architecture
- The guest OS executes within a Virtual Machine
- The guest OS is not aware to be virtualized
- No modification of the source code of the guest OS
- If the virtualized architecture is different from the physical one, we call it ***emulation***.

Hosted/Standalone Virtualization

- Hosted Virtualization
  - Hosted VM Monitor (VMM) runs on top of a native OS
  - VMware WKS, Microsoft VirtualPC, QEMU, UML (User Mode Linux)
- Standalone Virtualization
  - VMM directly runs on bare hardware
  - VMware ESXi, IBM/VM, Xen, VLX, KVM

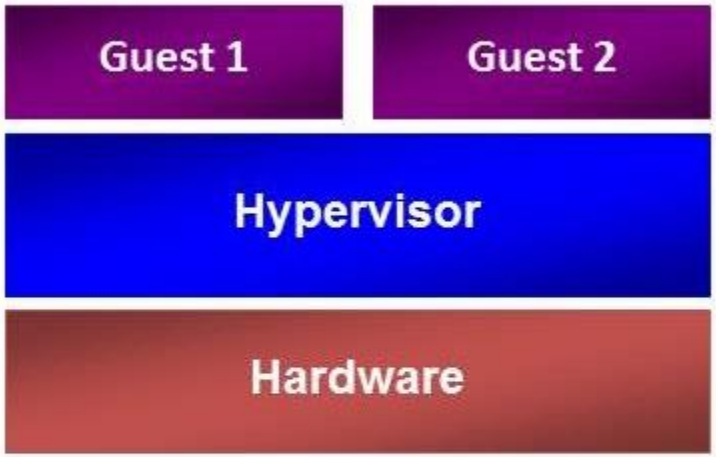
Hypervisor design: two approaches

Type 2 Hypervisor

Examples:

- Virtual PC & Virtual Server
- VMware Workstation

Type 1 Hypervisor



Examples:

- Hyper-V
- Xen
- VMware ESX

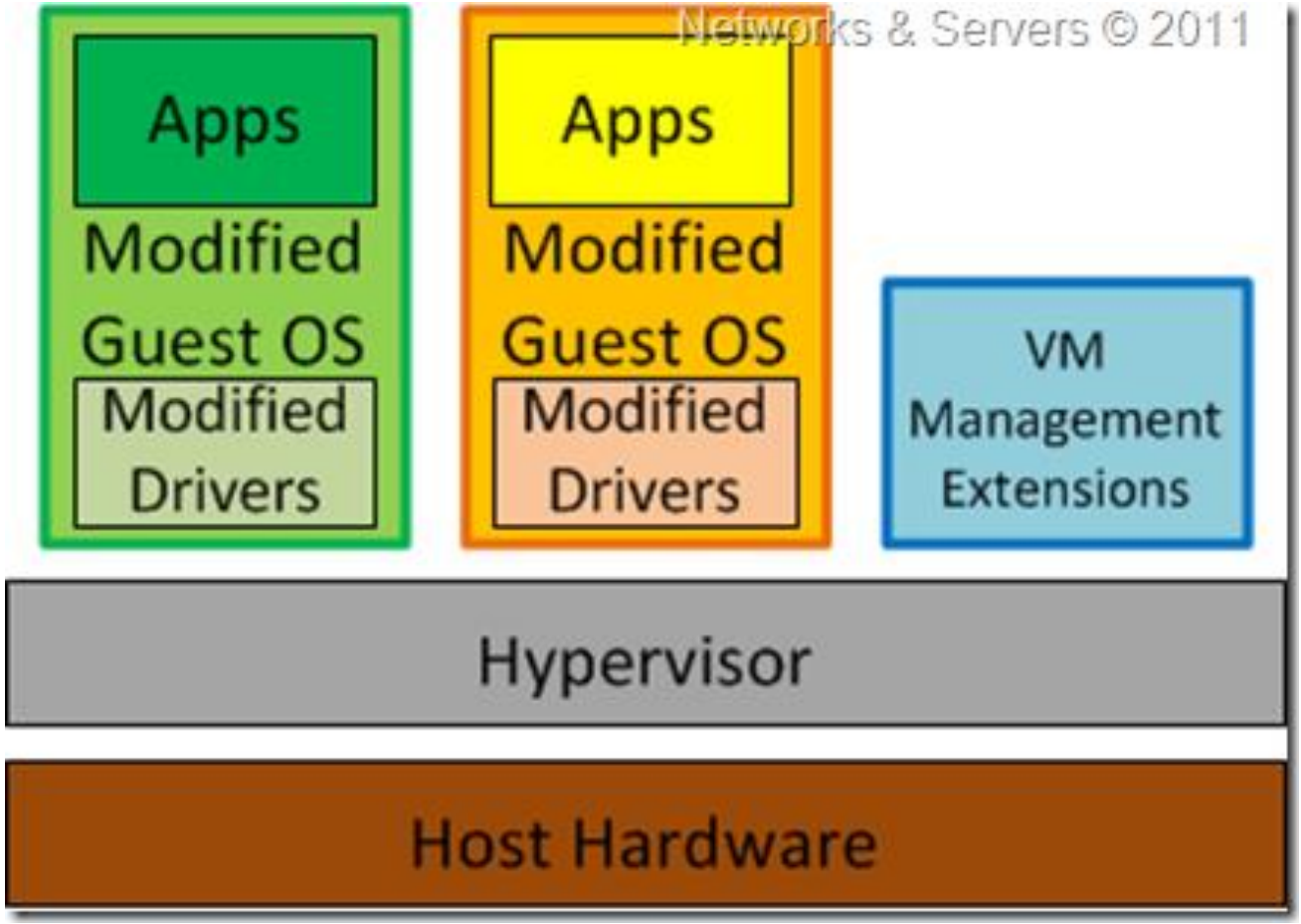
http://blogs.technet.com/b/chenley/archive/2011/02/09/hypervisors.aspx

Para-virtualization

- Achieves a collaboration between the hypervisor and the guest OS

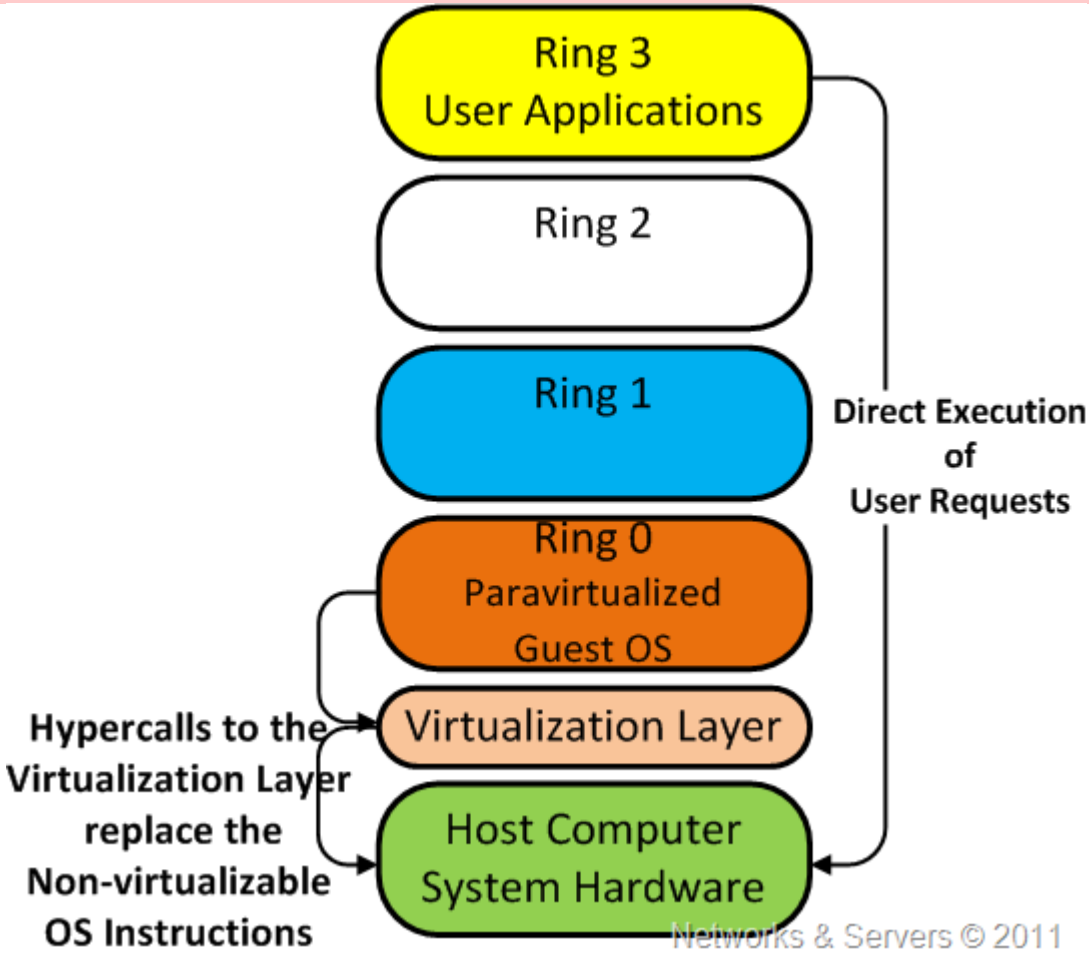
- As a consequence: the guest OS source code is modified to call directly the hypervisor (hypercalls) to execute the privileged instructions

Para-virtualization



http://networksandservers.blogspot.fr/2011/11/para-is-english-affix-of-greek-origin.html

Hypercalls



http://networksandservers.blogspot.fr/2011/11/para-is-english-affix-of-greek-origin.html

Hardware-assisted virtualization

- called also :
  - Accelerated virtualization
- Hardware support: Intel VT-x or AMD-V for x86 processors since 2006
- Virtualization support for x86 used by VMware Workstation, Xen, Linux KVM, Microsoft Hyper-V.

Virtualization in more details

Hosted/Standalone Virtualization

- Hypervisors of type 1 and type 2 rely on
  - Full virtualization
    - Virtualization of the CPU (instructions, interruptions)
    - Virtualization of the memory (memory access, MMU: memory management unit)
    - Virtualization of the devices
- The same physical resources are shared between the guest OSs
- Execute native/unmodified OS binary images

Full CPU virtualization

Full CPU virtualization

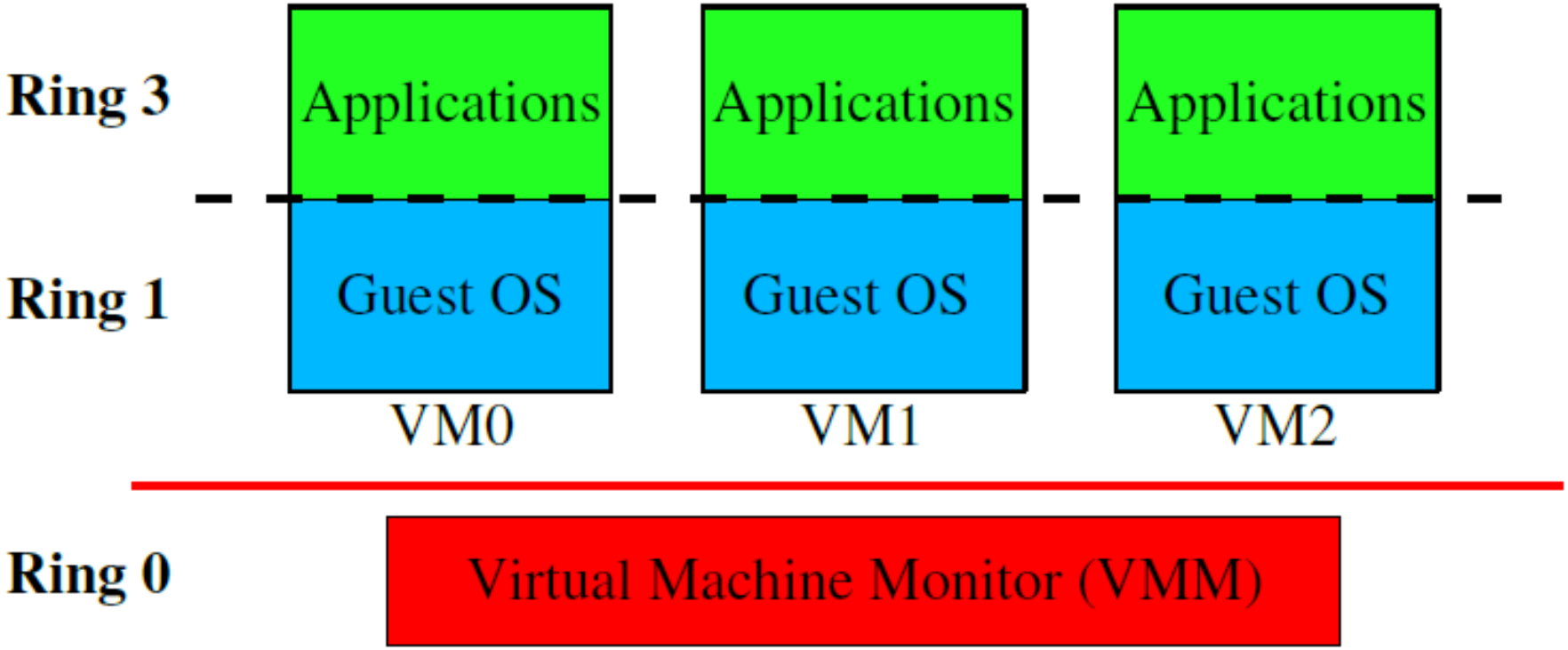
- VMM manages a CPU context for each VM
  - saved copy of CPU registers
  - representation of software emulated CPU context
- VMM includes a VM scheduler
  - Round robin
  - Priority based

Full CPU Virtualization

- Relationships between a VMM and VMs similar to relationships between native OS and applications
- Guarantee mutual isolation between all VMs
- Protect VMM from all VMs
- Directly execute native binary images of Guest OS's in non privileged mode
- VMM emulates access to protected resources performed by Guest OSs

CPU Virtualisation

Run each Guest OS in non privileged mode



“Hardware-Sensitive” instructions

- Interact with protected hardware resources
- Privileged Instructions
- Cannot be directly executed by Guest OS's
- Must be detected and faked by VMM
- Dynamic Binary Translation of kernel code
- Done once, saved in Translation Cache
- Example: Vmware

Privileged Instructions Virtualization

- Only allowed in supervisor mode
  - Ex: cli/sti to mask/unmask interrupts on Intel x86
- When executed in non privileged mode
  - CPU automatically detects a privilege violation
  - Triggers a “privilege violation” exception
- Caught by VMM which fakes the expected effect of the privileged instruction
  - Ex: cli/sti
    - VMM does not mask/unmask CPU interrupts
    - records « interrupt mask status » in context of VM

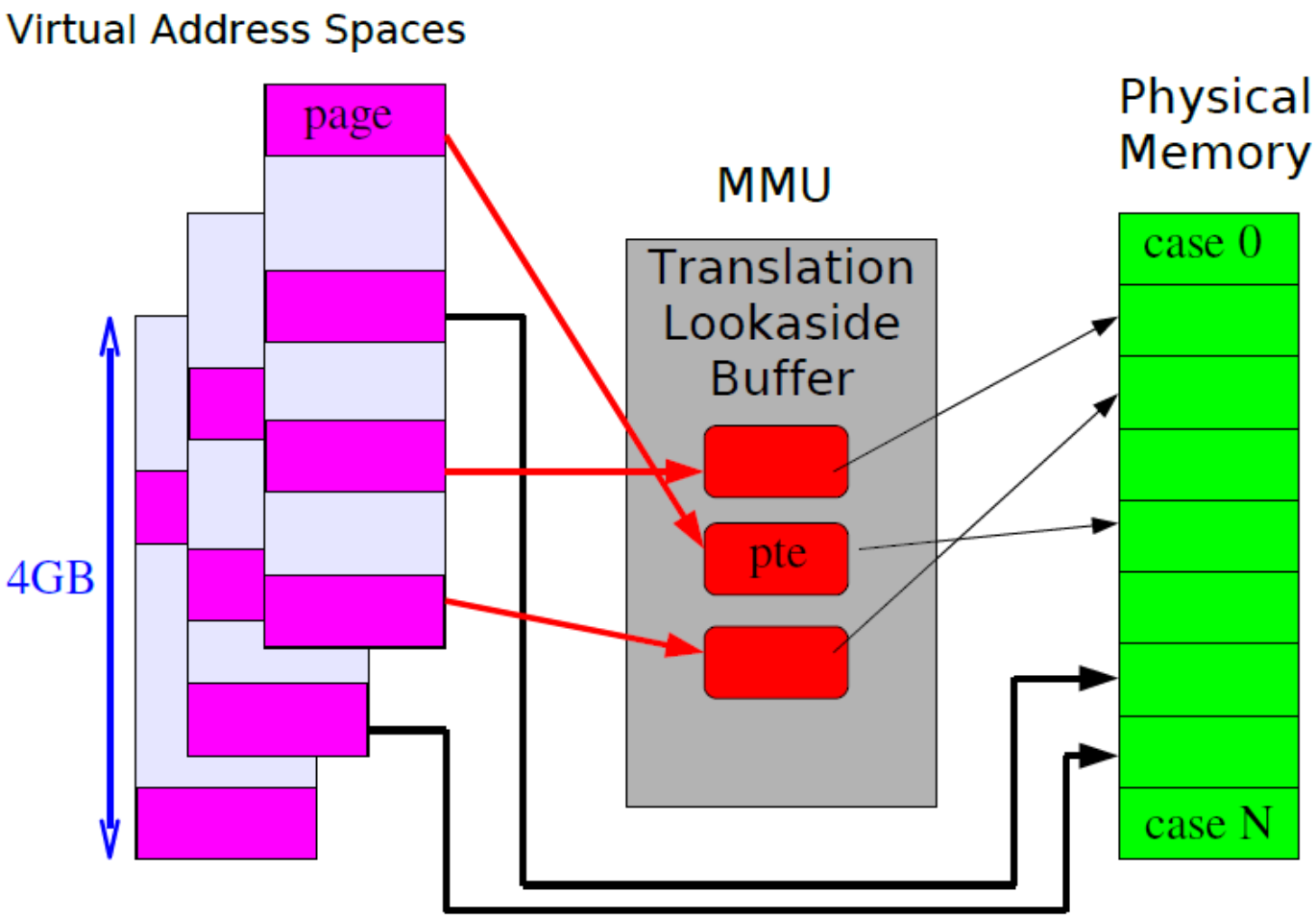
Full memory virtualization

samia.bouzefrane@lecnam.net

MMU

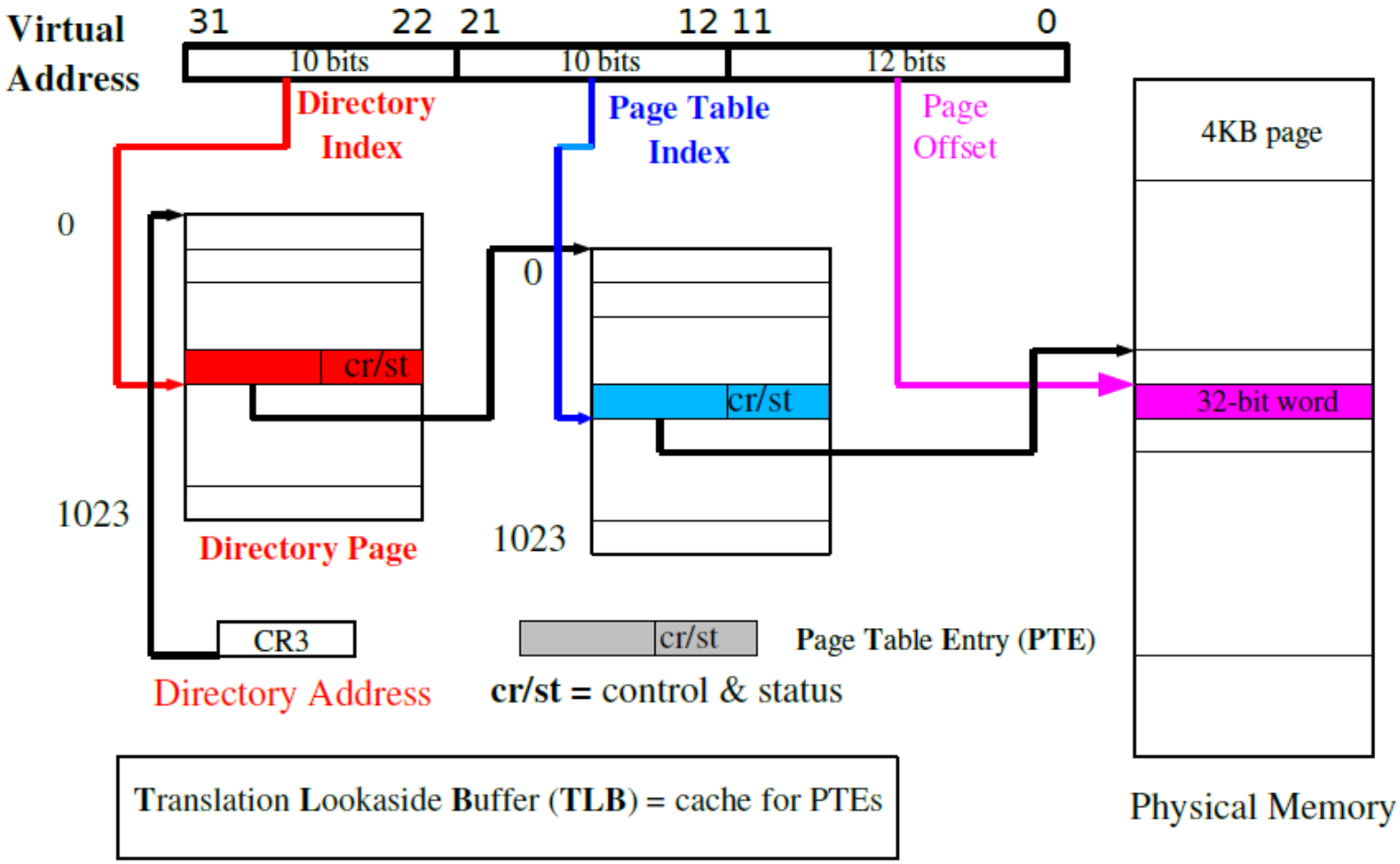
- CPU includes a Memory Management Unit (MMU)
  - Isolated memory addressing spaces
  - Independent of underlying physical memory layout
- Virtual Memory managed by OS kernel
  - Provides a virtual address space to each process
    - 4 GB on most 32bit architectures (Intel x86, PowerPC)
  - Manages virtual page → physical case mappings
  - Manages « swap » space to extend physical memory

MMU & Virtual Address Space



samia.bouzefrane@lecnam.net

Intel x86 MMU



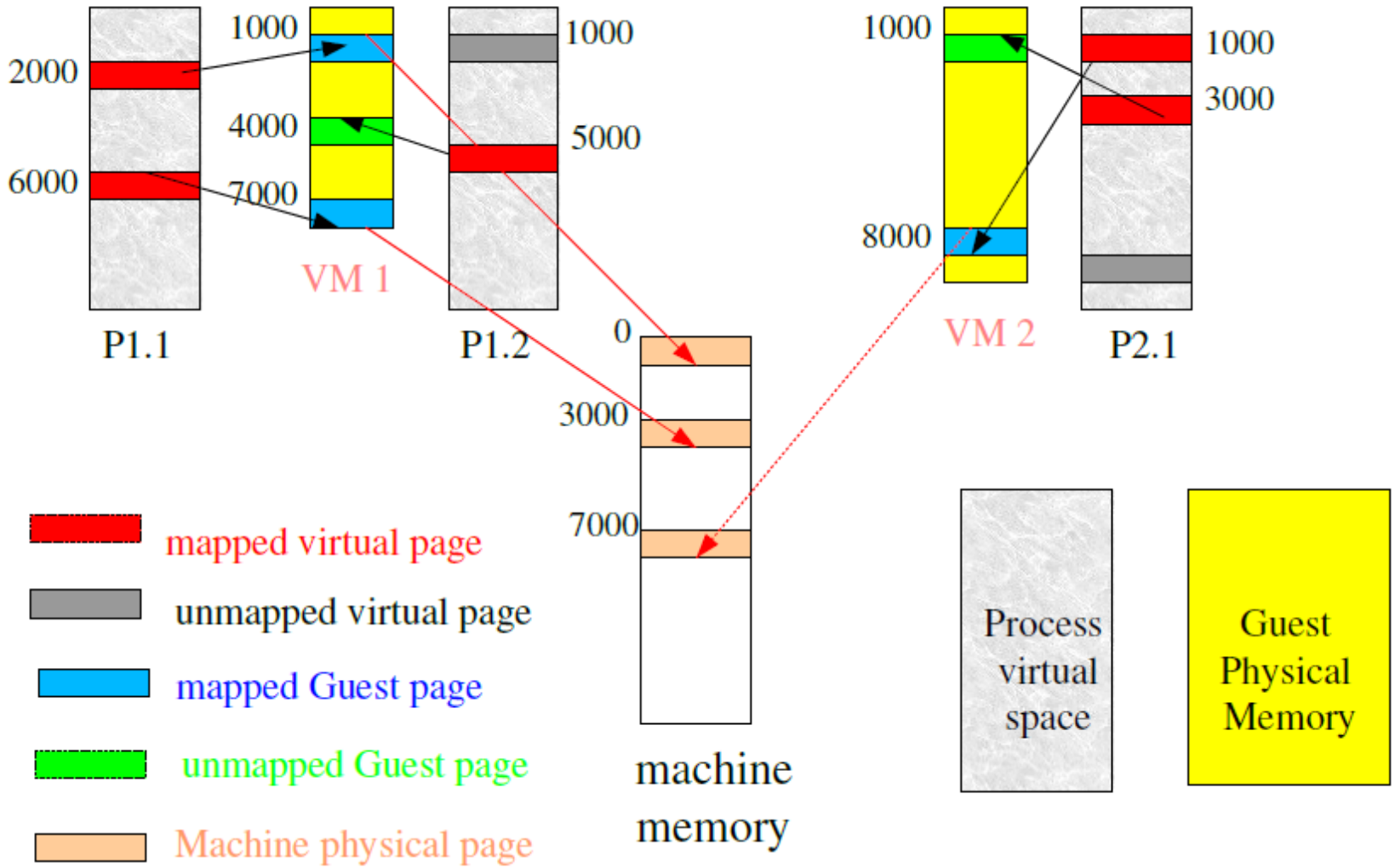
Memory Virtualization/1

- Machine Physical Memory (RAM)
  - Physical memory available on the machine
- Guest OS Physical Memory
  - Part of machine memory assigned to a VM by VMM
  - Σ Guest Physical Memory can be > Machine Memory
    - VMM uses « swap » space
- Guest OS Virtual Memory
  - Guest OS manages virtual address spaces of its processes

Memory Virtualization/2

- Guest OS manages Guest Physical Pages
  - Manages MMU with its own page entries
  - Translates Virtual Addresses into Guest Physical Addresses (GPA)
- VMM transparently manages Machine Physical Pages
  - Guest Physical Address ≠ Machine Physical Address
  - VMM dynamically translates Guest Physical Pages into Machine Physical Pages

Memory Virtualization/3



Memory Virtualization/4

- VMM maintains Shadow Page Tables
  - Copies of Guest OS translation tables
- VMM catches updates operations of translation tables performed by a Guest OS
  - Write protect all guest OS page tables
  - Emulates operation in shadow page table
  - Updates effective MMU page table entry, if needed

Para-virtualization principles

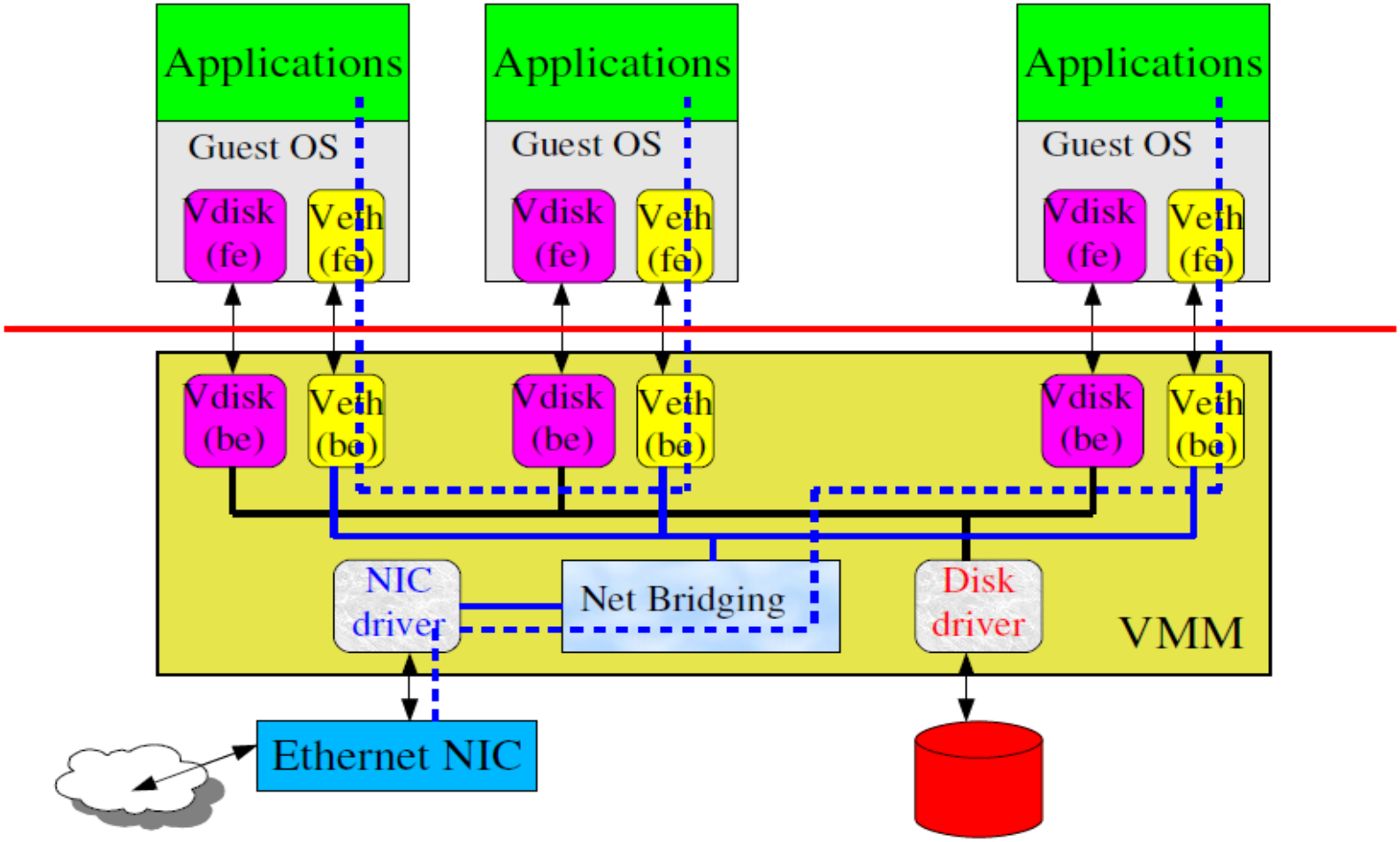
Para-virtualization

- OS adaptation to avoid binary translation overhead
- Requires access to OS source code
- Include drivers of virtual devices
  - Examples
    - Xen
    - **U**ser **M**ode **L**inux (UML)

Para-virtualization

- Guest OS only uses Virtual I/O Devices
  - Frontend driver in Guest OS
  - Backend driver in VMM
- Avoid extra I/O data copies of Full Virtualization
- VMM multiplexes VM Virtual Devices on physical devices
  - Virtual Ethernet
  - Virtual Disks
- Example:
  - Linux guest OS uses **paravirt_ops** to replace the privileged instructions of the guest OS by hypercalls (like in Xen).

Virtual I/O Devices



Paravirtualization Example: Xen

- Objectives
  - Support more than 100 VM
  - Share resources of Server machines
- Intel IA32, x86-64 and ARM architectures
- Special first Guest OS called Domain 0
  - Run in privileged mode
  - Have access (and manages) all physical devices
  - Modified version of Linux, FreeBSD

Hardware-Assisted Virtualization

samia.bouzefrane@lecnam.net

Hardware-Assisted Virtualization

- Support of Virtualization in Hardware
- Run unmodified OS binaries
- With minimal virtualization overhead
- Simplify VMM development
- Examples
  - KVM (Intel-VT, AMD-V)
  - VMware (Intel-VT)

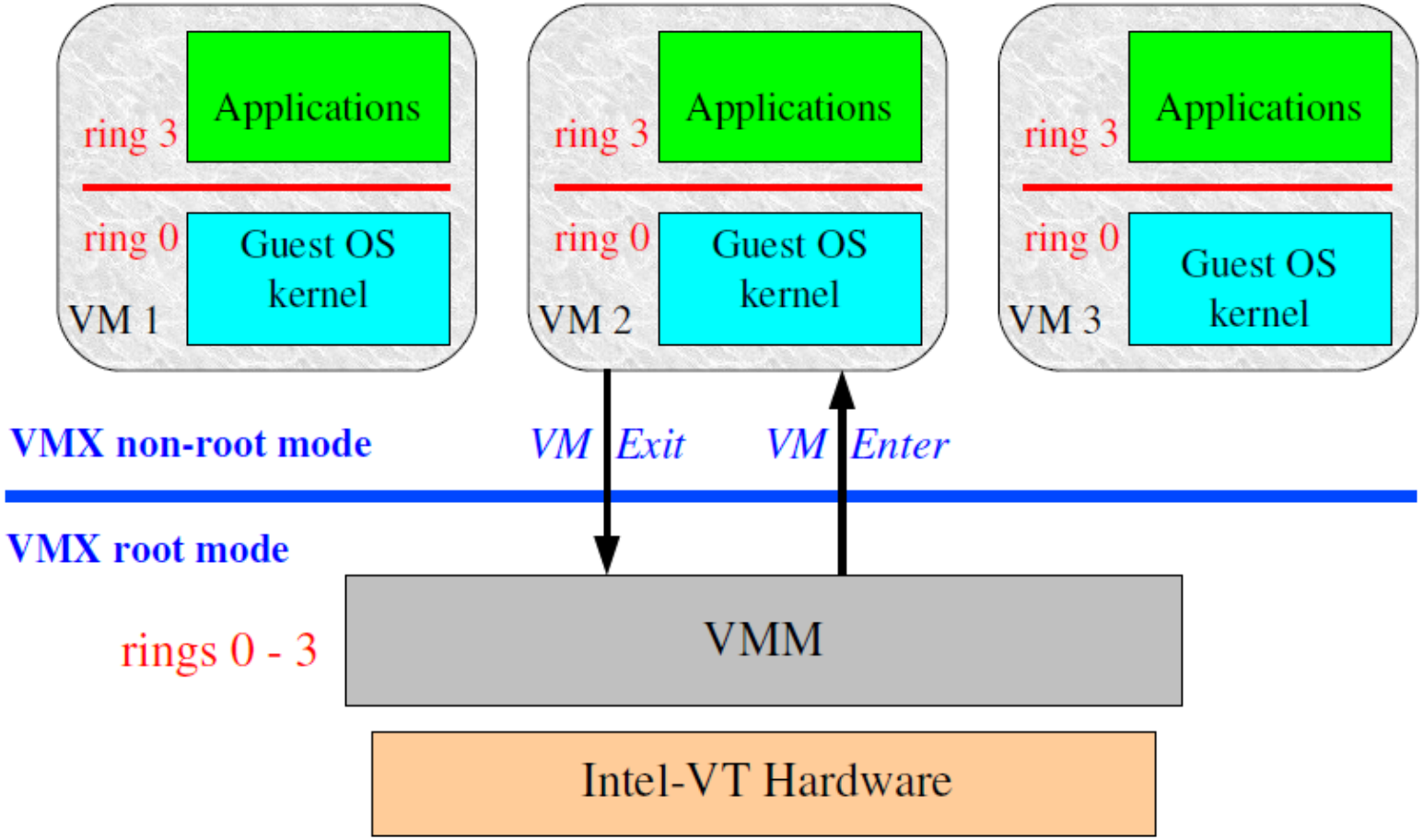
Hardware-Assisted Virtualization

- CPU virtualization
  - AMDV
  - Intel VTx (x86), Intel VTi (Itanium) architectures
  - ARM CortexA15
- MMU virtualization
  - Intel **E**xtended **P**age **T**ables (EPT)
  - AMD **N**ested **P**age **T**ables (NPT)

Intel VT-x Architecture

- Support unmodified Guest OS with no need for paravirtualization and/or binary code translation
- Simplify VMM tasks & improve VMM performances
- Minimize VMM memory footprint
  - Suppress shadowing of Guest OS page tables
- Enable Guest OS to directly manage I/O devices
  - Without performance lost
  - While enforcing VM isolation and mutual protection

Intel VT-x Architecture



Intel VTx CPU Virtualization

- Two additional CPU mode transitions
- From VMX root-mode to VMX non-root mode
  - Named *VM Enter*
- From VMX non-root mode to VMX root mode
  - Named *VM Exit*
- VM entries & VM exits use a new data structure
  - **V**irtual **M**achine **C**ontrol **S**tructure (VMCS) per VM
  - Referenced with a memory physical address
  - Format and layout hidden
  - New VT-x instructions to access a VMCS

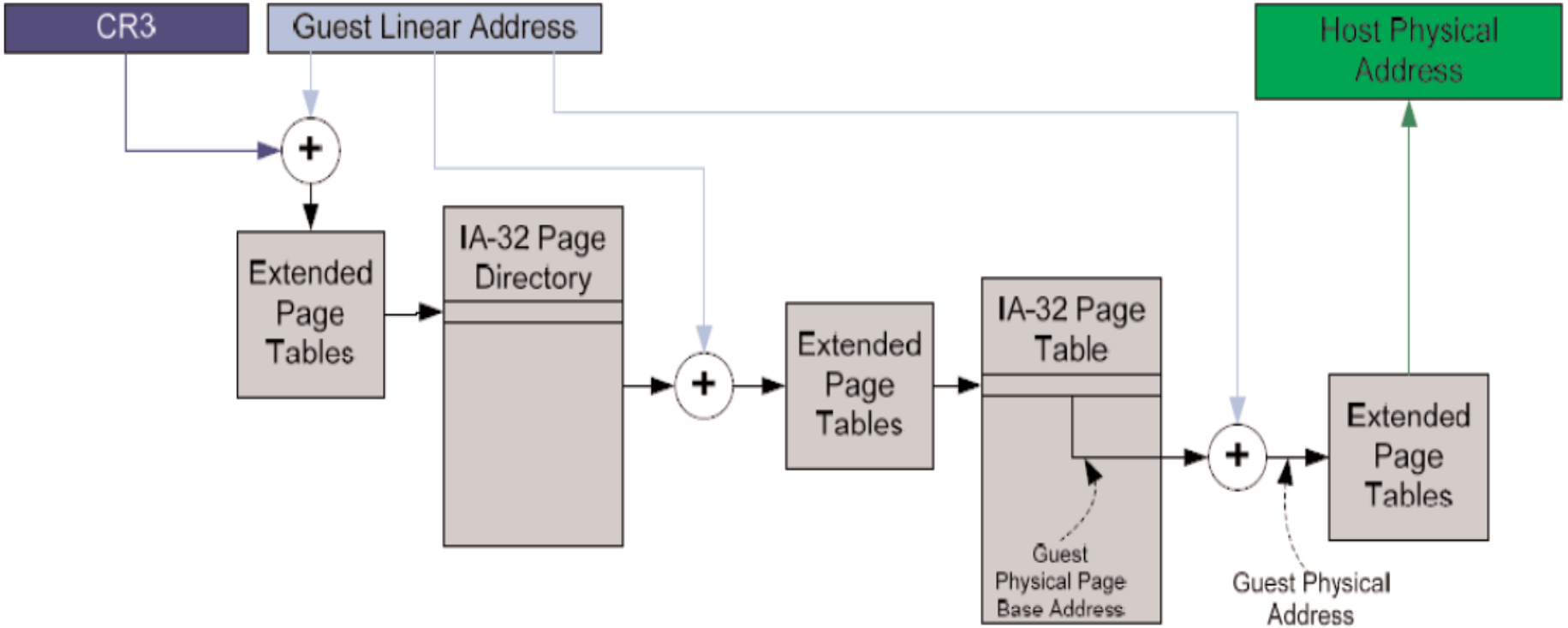
VMX instructions

VMXON/VMXOFF	Enable/Disable VMX operation
VMCLEAR	Initialize VMCS region
VMPTRLD/VMPTRST	Load/Store Current VMCS pointer
VMREAD/VMWRITE	Read or Write VMCS fields
VMLAUNCH/VMRESUME	Launch or resume virtual machine
VMCALL	Issued from virtual machine to call into VMM

Intel VTx Extended Page Tables

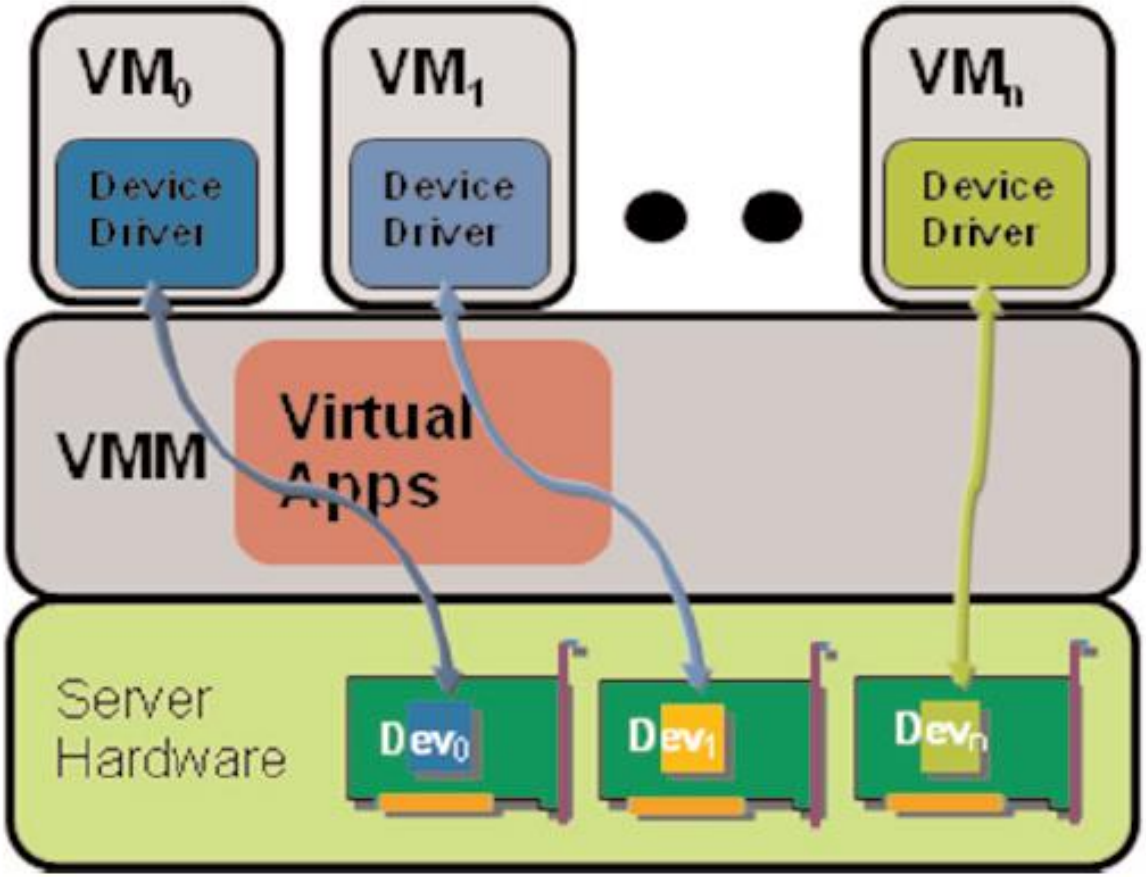
- VMM controls Extended Page Tables
- EPT used in VMX non-root operation
  - Activated on *VM Enter*
  - Desactivated on *VM exit*
- EPTP register points to Extended Page Tables
  - Instanciated by VMM
  - Saved in VMCS
  - Loaded from VMCS on *VM entry*

MMU Virtualization : Intel VT-x



samia.bouzefrane@lecnam.net

DMA virtualization : Intel VT-d



samia.bouzefrane@lecnam.net

References

- Lecture of Dominique Rodriguez, UE SMB204, CNAM.
- http://www-igm.univ-mlv.fr/~dr/XPOSE2008/virtualisation/techniques.html
- http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf
- Intel documentation
- Magazine MISC n°42, « La virtualisation: vecteur de vulnérabilité ou de sécurité? » Mars/avril 2009.
- Dino A. Dai Zovi, “Hardware Virtualization Rootkits”, Matasano, http://dator8.info/2010/16.pdf