Introduction to virtualisation technology

Predrag Buncic

CERN

CERN School of Computing 2009

CERN School of Computing

History

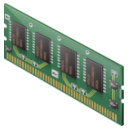
- **Credit for bringing virtualization into computing goes to IBM**
- **IBM VM/370 was a reimplementation of CP/CMS, and was made available in 1972**
  - added virtual memory hardware and operating systems to the System/370 series.

- **Even in the 1970s anyone with any sense could see the advantages virtualization offered**
  - It separates applications and operating systems from the hardware
  - With VM/370 you could even run MVS on top - along with other operating systems such as Unix.
  - In spite of that, VM/370 was not a great commercial success
- **The idea of abstracting computer resources continued to develop**

CERN School of Computing

Resource virtualization

- **Virtualization of specific system computer resources such as**
  - **Memory virtualization**
    - Aggregates RAM resources from networked systems into virtualized memory pool
  - **Network virtualization**
    - Creation of a virtualized network addressing space within or across network subnets
    - Using multiple links combined to work as though they offered a single, higher-bandwidth link
  - **Virtual memory**
    - Allows uniform, contiguous addressing of physically separate and non-contiguous memory and disk areas
  - **Storage virtualization**
    - Abstracting logical storage from physical storage
    - RAID, disk partitioning, logical volume management



Memory

Networking

Storage

CERN School of Computing

Metacomputing

- **A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer.**
  - The components of a cluster are commonly connected to each other through fast local area networks.

- **Grids are usually computer clusters, but more focused on throughput like a computing utility rather than running fewer, tightly-coupled jobs**
  - Often, grids will incorporate heterogeneous collections of computers, possibly distributed geographically, sometimes administered by unrelated organizations
  - Optimized for workloads which consist of many independent jobs or packets of work, which do not have to share data between the jobs during the computation process.

CERN School of Computing

Application Virtualization

- **Technologies that improve portability, manageability and compatibility of applications by encapsulating them from the underlying operating system on which they are executed**
  - A virtualized application is not installed in the traditional sense, although it is still executed as if it is
    - Interaction with OS via special virtualization layer
      - Microsoft Application Virtualization, VMware ThinAPP
    - The application is fooled at runtime into believing that it is directly interfacing with the original operating system and all the resources managed by it, when in reality it is not
  - Alternatively, minimal OS is build around application and deployed as virtual machine
    - Virtual Software Appliances
      - rBuilder from rPath

CERN School of Computing

Platform virtualization

- **This is what most people today identify with term “virtualization”**
  - Also known as server virtualization
  - Hides the physical characteristics of computing platform from the users
  - Host software (hypervisor or VMM) creates a simulated computer environment, a virtual machine, for its guest OS
  - Enables server consolidation
- **Platform virtualization approaches**
  - Operating system-level virtualization
  - Partial virtualization
  - Paravirtualization
  - Full virtualization
  - Hardware-assisted virtualization

CERN School of Computing

Operating System Virtualization

- **Server virtualization method where the kernel of an operating system allows for multiple isolated user-space instances, instead of just one**
  - Such instances (often called containers or jails) may look and feel like a real server, from the point of view of its owner
  - On Unix systems, this technology can be thought of as an advanced implementation of the standard *chroot* mechanism
  - In addition to isolation mechanisms, the kernel often provides resource management features to limit the impact of one container's activities on the other containers.
- **Usually imposes little or no overhead**
  - Because programs in virtual partition use the operating system's normal system call interface and do not need to be subject to emulation or run in an intermediate virtual machine
  - Hypervisor not required
- **It cannot host a guest operating system different from the host one, or a different guest kernel**

CERN School of Computing

Partial Virtualization

- **Virtualization technique used to implement a certain kind of virtual machine environment**
  - Provides only a partial simulation of the underlying hardware
  - Most but not all of the hardware features are simulated
  - Usually, this means that entire operating systems cannot run in the virtual machine but that many applications can run.
- **A key form of partial virtualization is "address space virtualization", in which each virtual machine consists of an independent address space**
- **Partial virtualization was an important historical milestone on the way to full virtualization**
  - It was used in the first-generation time-sharing system CTSS, and in the IBM M44/44X experimental paging system

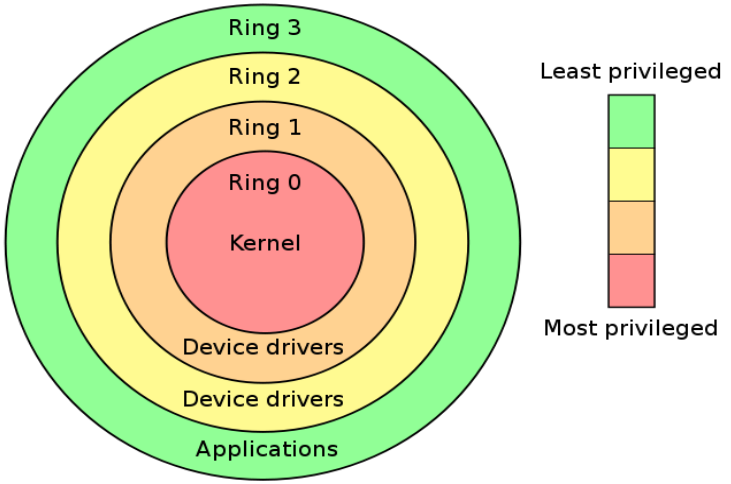
CERN School of Computing

Full virtualization

- **Virtual machine simulates enough hardware to allow an unmodified "guest" OS**
- **A key challenge for full virtualization is the interception and simulation of privileged operations**
  - The effects of every operation performed within a given virtual machine must be kept within that virtual machine
  - The instructions that would "pierce the virtual machine" cannot be allowed to execute directly; they must instead be trapped and simulated.
- **Examples**
  - Parallels Workstation, Parallels Desktop for Mac, VirtualBox, Virtual Iron, Oracle VM, Virtual PC, Virtual Server, Hyper-V, VMware Workstation, VMware Server (formerly GSX Server), QEMU

Hypervisor

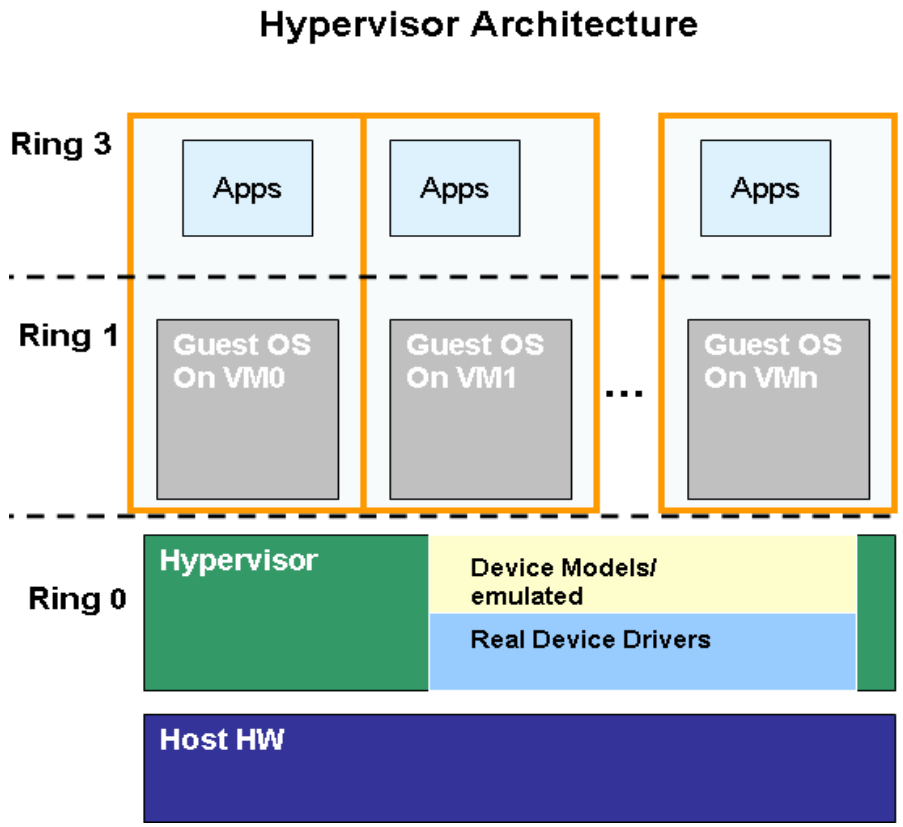
- **To create several virtual servers on one physical machine we need a hypervisor or Virtual Machine Monitor (VMM).**
  - The most important role is to arbitrate the access to the underlying hardware, so that guest OSes can share the machine.
  - VMM manages virtual machines (Guest OS + applications) like an OS manages processes and threads.



- **Most modern operating system work with two modes:**
  - kernel mode
    - allowed to run almost any CPU instructions, including "privileged" instructions that deal with interrupts, memory management…
  - user mode
    - allows only instructions that are necessary to calculate and process data, applications running in this mode can only make use of the hardware by asking the kernel to do some work (a system call).

Hypervisor architecture

- **A technique that all (software based) virtualization solutions use is ring deprivileging:**
  - the operating system that runs originally on ring 0 is moved to another less privileged ring like ring 1.
  - This allows the VMM to control the guest OS access to resources.
  - It avoids one guest OS kicking another out of memory, or a guest OS controlling the hardware directly.



CERN School of Computing

Virtualization and x86

- **IBM S/370, used a robust system to allow the hypervisor to control the virtual machines**
  - Executing the privileged instructions in a virtual machine, while running in a less privileged ring caused a "trap"
  - The VMM intercepts all those traps and emulates the instruction
- **This was not possible on x86 as the 32/64-bit processor does not trap every incident that should lead to VMM intervention.**
  - One example is the POPF instruction that disables and enables interrupts.
    - If this instruction is executed by a guest OS in ring 1, an x86 CPU simply ignores it
  - There are 17 such instructions oon x86 instruction set
- **x86 cannot be virtualized the way that the old mainframes were virtualized.**

CERN School of Computing

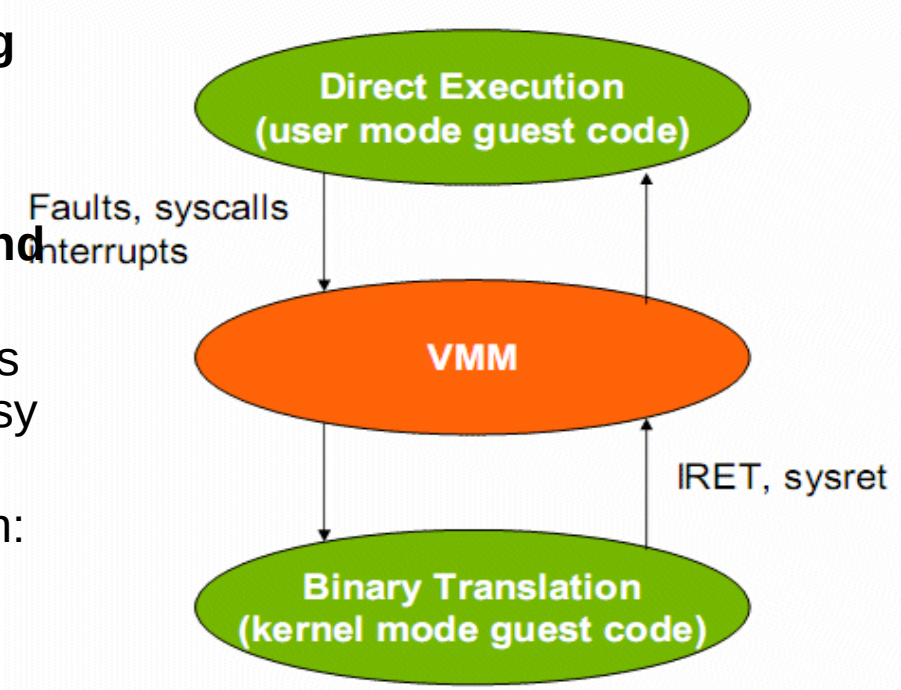
Binary translation

- **To expose improper x86 instructions, VMware introduced Binary Translation**
  - It is based on an x86 to x86 translator.
    - In fact, in some cases it just makes an exact copy of the original instruction.
  - VMware translates the binary code that the kernel of a guest OS wants to execute on the fly and stores the adapted x86 code in a Translator Cache (TC).
    - User applications will not be touched by VMware's Binary Translator as it knows/assumes that user code is safe.
    - User mode applications are executed directly as if they were running natively.

CERN School of Computing

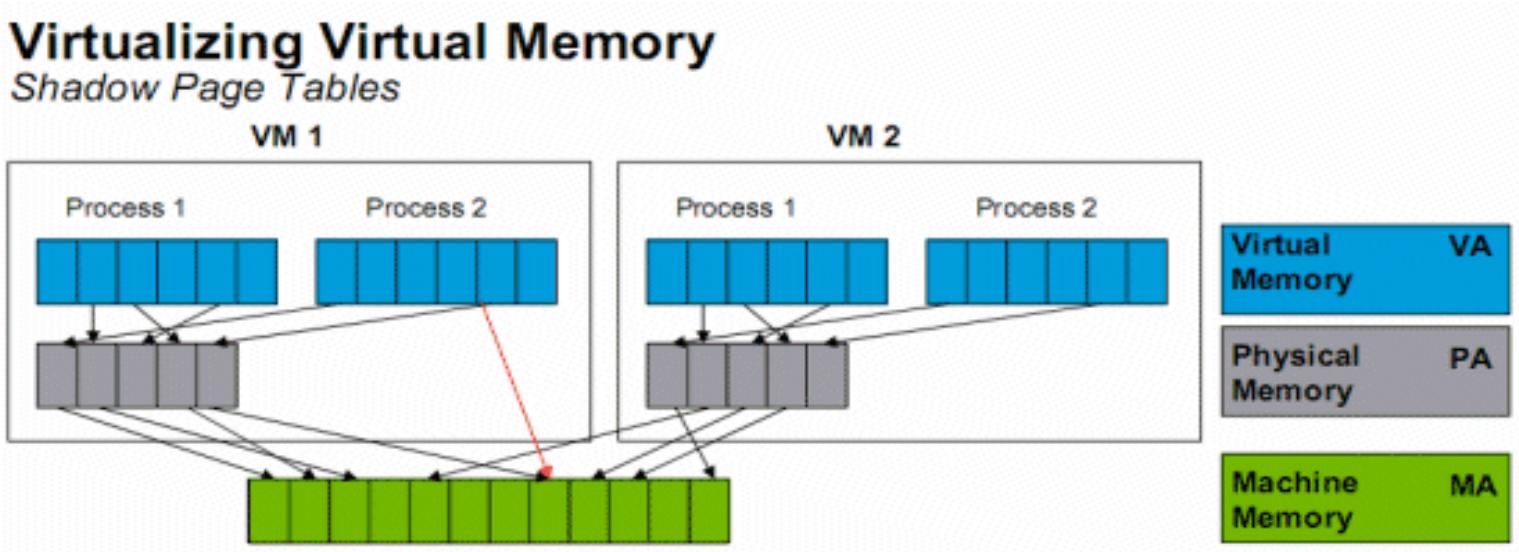
Binary translation issues

- **Binary Translation is lot less costly than letting privileged instructions result in traps and then handling those traps afterwards**
- **That doesn't mean that the overhead of this kind of virtualization is always low.**
  - The "Translator overhead" is rather low, and its impact gets lower and lower over time, courtesy of the Translator cache.
  - Binary Translation cannot be very efficient with:
    - System Calls
    - Accesses to I/O, interrupts and DMA
    - Memory management



- **A system call will give the Virtual Machine Monitor quite a bit of extra work**
  - VMM has to emulate system call, translate the code, and then hand over the control to the translated kernel code which runs in ring 1
    - A system call in VM will cost roughly 10 times more than on a native machine.

Memory management

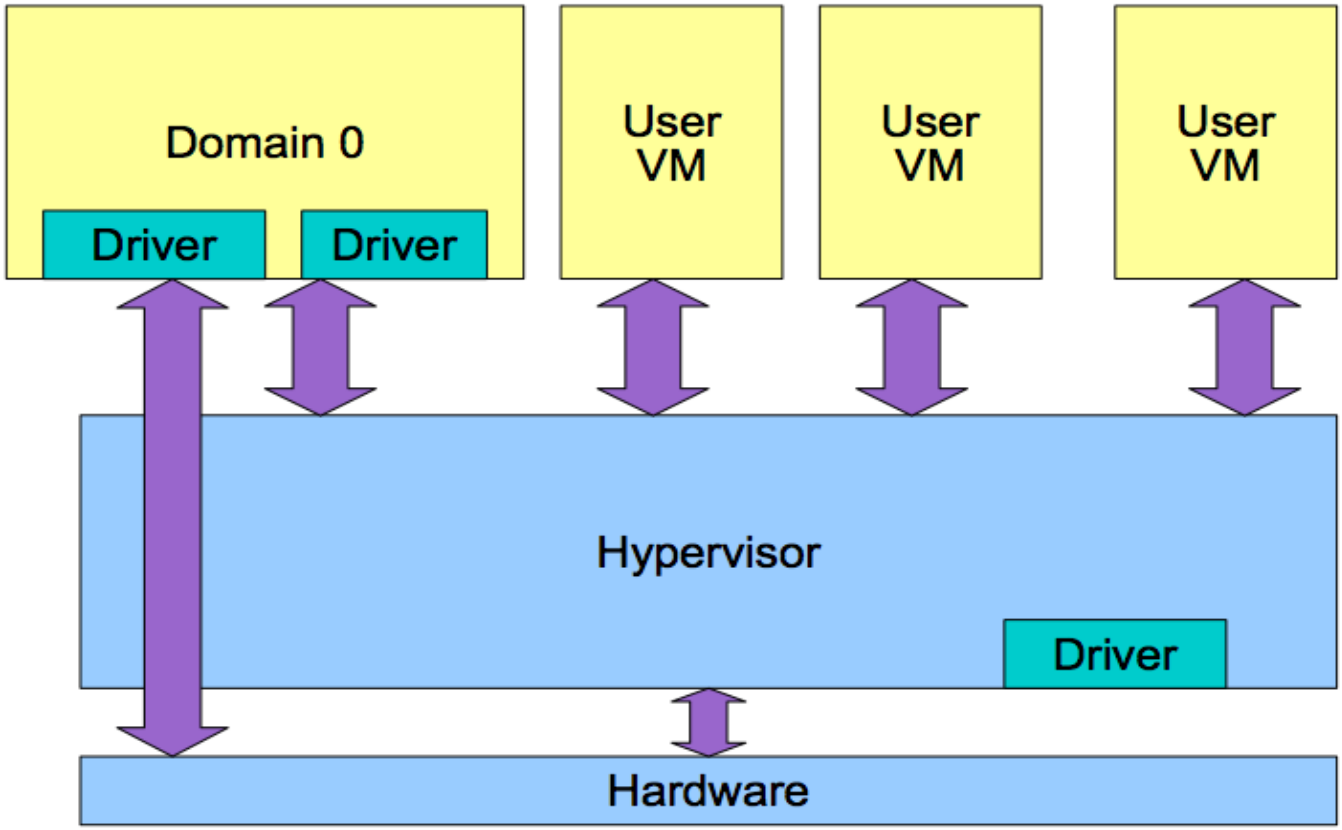


- **An OS maintains page tables to translate the virtual memory pages (blue) into "guest OS physical memory" (gray) and then translate the latter into the real physical memory (green)**
  - The "shadow page tables" make the MMU work with a virtual memory (of the guest OS, blue) to map to real physical memory (green) page table, skipping the intermediate "guest OS physical memory" step.
  - Every time the guest OS modifies its page mapping, the virtual MMU module will capture (trap) the modification and adjust the shadow page tables accordingly.
    - This bookkeeping takes 3 to 400 (!) times more cycles than in the native situation.

Paravirtualization

- **Virtualization technique that presents a software interface to virtual machines that is similar but not identical to that of the underlying hardware.**
  - Guest kernel source code modification instead of binary translation
  - The paravirtualization provides specially defined 'hooks' to allow the guest(s) and host to request and acknowledge these tasks, which would otherwise be executed in the virtual domain (where execution performance is worse)
  - Paravirtualized platform may allow the virtual machine monitor (VMM) to be simpler (by relocating execution of critical tasks from the virtual domain to the host domain) and faster
- **Paravirtualization requires the guest operating system to be explicitly ported for the para-API**
  - a conventional O/S distribution which is not paravirtualization aware cannot be run on top of a paravirtualized VMM.

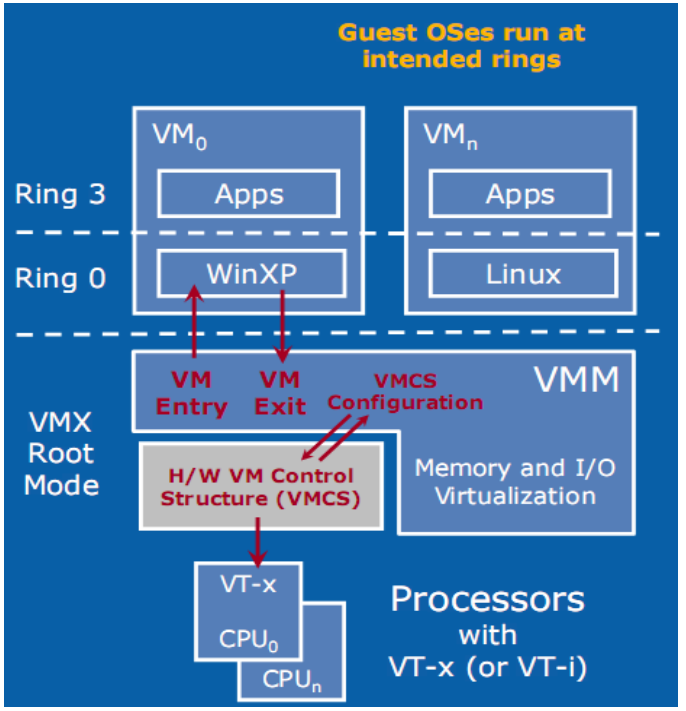
Example: Xen


Domain 0
Driver
Driver
User VM
User VM
User VM
Hypervisor
Driver
Hardware

CERN School of Computing

Hardware assisted virtualization

- **Idea behind hardware virtualization is to fix the problem that the x86 instructions architecture cannot be virtualized.**
  - Trying to trap all exceptions and privileged instructions by forcing a transition from the guest OS to the VMM
  - Big advantage is that the guest OS runs at its intended privilege level (ring 0), and that the VMM is running at a new ring with an even higher privilege level (Ring -1, or "Root mode").



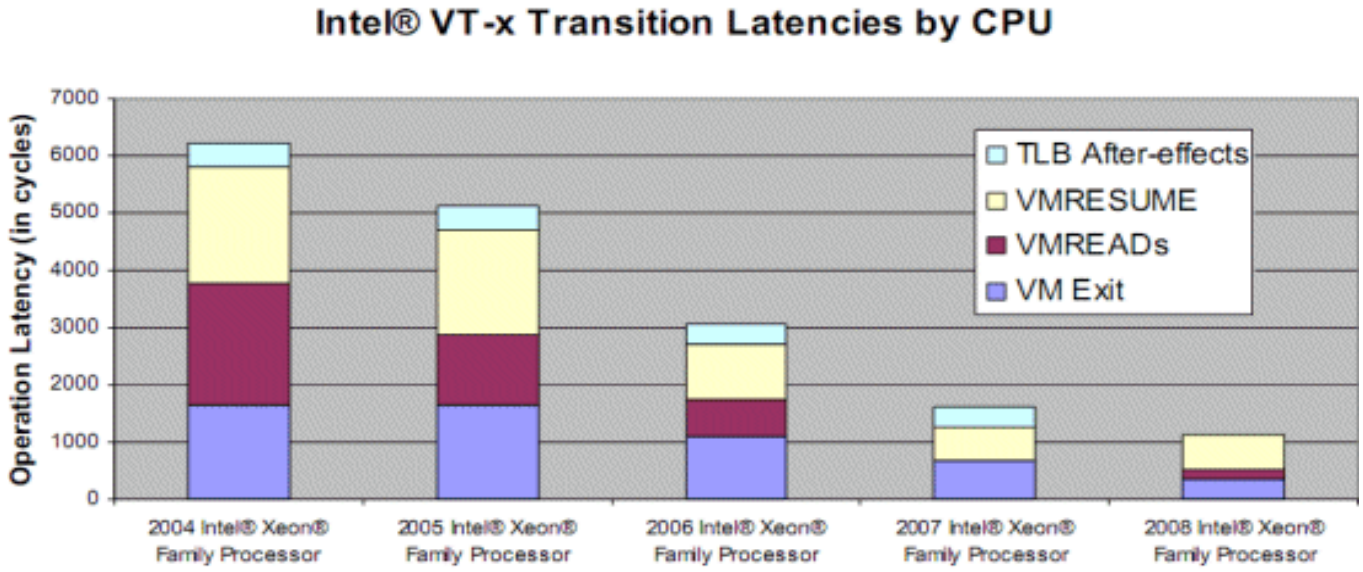
System calls do not automatically result in VMM interventions: as long as system calls do not involve critical instructions, the guest OS can provide kernel services to the user applications.

CERN School of Computing

Hardware assisted virtualization issues

- **Even if it is implemented in hardware - each transition from the VM to the VMM requires a fixed and large number of CPU cycles**
  - The specific number of these "overhead cycles" depends on the internal CPU architecture.
  - Depending on the exact operation these kinds of events can take a few hundred up to a few thousand CPU cycles!
- **The VM/VMM roundtrip of hardware virtualization is a heavy event**
  - When CPU has to handle complex operations such as system calls the VMexit/VMentry switching penalty has little impact.
  - If the actual operation that the VMM has to intercept and emulate is rather simple, the overhead of switching to VMM is huge!
    - Simple operations such as creating processes, context switches, small page table updates
  - With BT, the translator replaces the code with slightly longer code that the VMM handles. The same is true for paravirtualization, which is a lot faster than hardware virtualization at handling these kinds of events

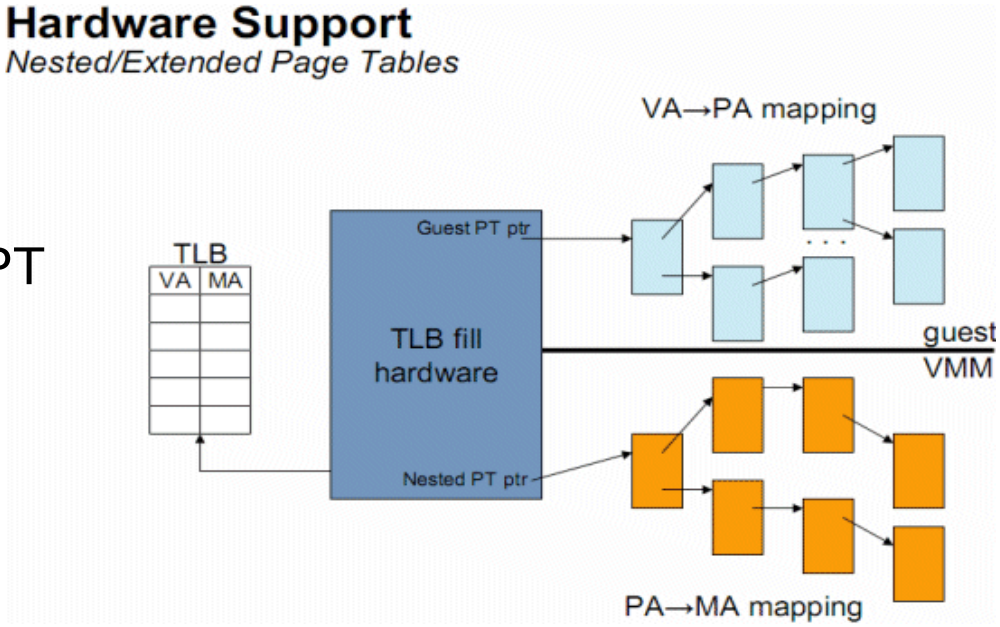
Improvements



- **Two strategies to reduce total overhead**
  - Total Overhead = Sum of (Frequency of "VMM to VM" events * Latency of event)
  - Reducing latency
    - Reducing the number of cycles that the VT-x instructions take.
      - VMentry latency was reduced from 634 (Xeon 70xx) to 352 cycles in the (Xeon 51xx, Xeon 53xx, Xeon 73xx)
    - Reducing frequency of VMM to VM events
      - Virtual Machine Control Block contains the state of the virtual CPU(s) for each guest OS allowing them to run directly without interference from the VMM.

Nested page tables

- **The second generation of hardware virtualization improves memory access from VM**
  - AMD's nested paging and Intel's EPT technology



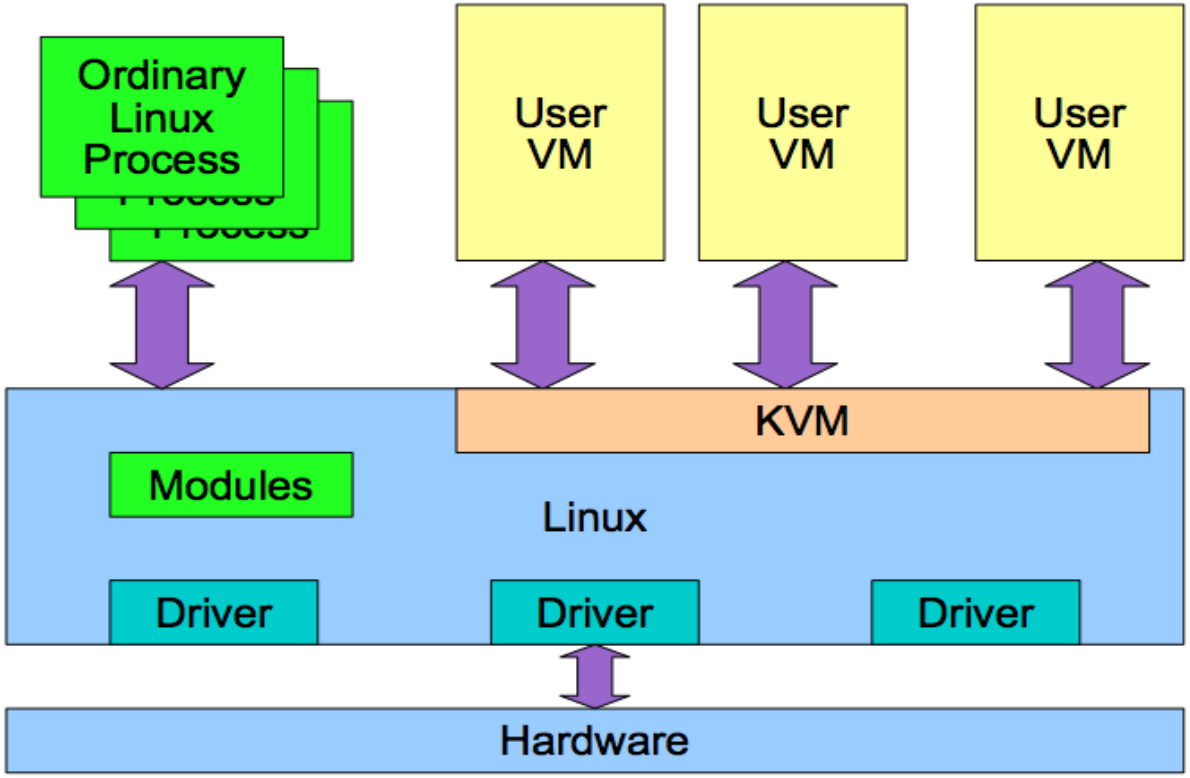
- **The Translation Lookaside Buffer (TLB) has a new VM specific tag, called the Address Space IDentifier (ASID)**
  - This allows the TLB to keep track of which TLB entry belongs to which VM.
    - The result is that a VM switch does not flush the TLB.
    - This makes the VMM a lot simpler and completely annihilates the need to update the shadow page tables constantly.

Hybrid strategy

- **Second generation hardware virtualization (VT-x+EPT and AMD-V+NPT) is more promising**
  - it is not guaranteed that it will improve performance across all applications due to the heavy TLB miss cost.
- **Software virtualization is very mature, but there is very little headroom left to improve**
- **The smartest way is to use a hybrid approach like VMware ESX**
  - paravirtualized drivers for the most critical I/O components
  - emulation for the less important I/O
  - Binary Translation to avoid the high "trap and emulate" performance penalty
  - hardware virtualization for 64-bit guests

CERN School of Computing

Other examples: KVM



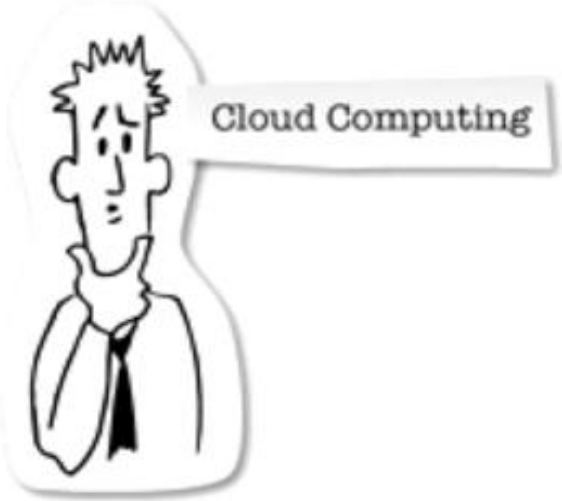
CERN School of Computing

Virtualization and Cloud computing

- **Utility Computing on demand**
  - Amazon Elastic Computing Cloud
  - Blue Cloud (IBM)

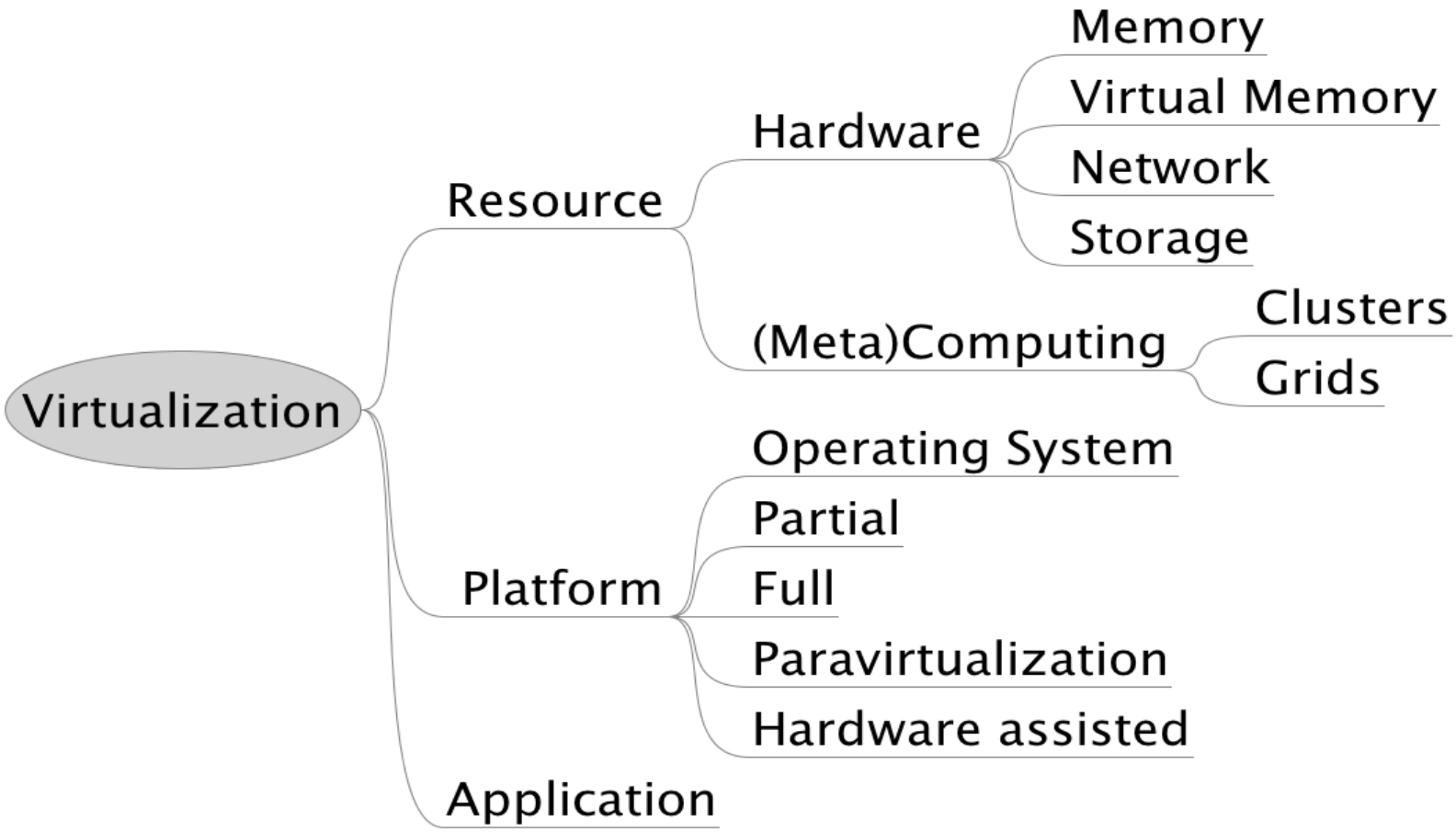
- **Software as a Service (SaaS)**
  - Google App Engine
    - Delivering an application through the browser to thousands of customers
    - Investing ~$2 billion per year in new generation data centres
  - Microsoft Azzuro
    - Hosted office applications & data
    - Plans to match Google in terms of number of data centres, $600 million per site

- **Virtualization as enabling technology**



http://www.youtube.com/watch?v=XdBd14rjcs0

Taxonomy


Virtualization
Resource
Hardware
Memory
Virtual Memory
Network
Storage
(Meta)Computing
Clusters
Grids
Platform
Operating System
Partial
Full
Paravirtualization
Hardware assisted
Application

CERN School of Computing

Conclusions

- **Virtualization is a broad term that refers to the abstraction of computer resources**
  - Old technology making comeback thanks to breakdown in frequency scaling and appearance of multi and many core CPU technology
  - Enabling technology of Cloud computing
  - Virtualization is here to stay for foreseeable future
- **Not limited to platform (server) virtualization**
  - Server consolidation is the most often quoted reason to use virtualization
- **Common virtualization techniques**
  - Resource virtualization
    - storage, network
  - Computer clusters & grids
    - combination of multiple discrete computers into a large metacomputer
  - Platform virtualization
    - separation OS from underlying hardware resource