Cloud Resource Virtualization

Motivation

There are many physical realizations of the fundamental abstractions necessary to describe the operation of a computing systems.

- Interpreters.
- Memory.
- Communications links.

Virtualization is a basic tenet of cloud computing, it simplifies the management of physical resources for the three abstractions.

The state of a virtual machine (VM) running under a virtual machine monitor (VMM) can de saved and migrated to another server to balance the load.

Virtualization allows users to operate in environments they are familiar with, rather than forcing them to idiosyncratic ones.

Motivation (cont'd)

Cloud resource virtualization is important for:

- System security, as it allows isolation of services running on the same hardware.
- Performance and reliability, as it allows applications to migrate from one platform to another.
- The development and management of services offered by a provider.
- Performance isolation.

Virtualization

Simulates the interface to a physical object by:

- Multiplexing: creates multiple virtual objects from one instance of a physical object. Example - a processor is multiplexed among a number of processes or threads.
- Aggregation: creates one virtual object from multiple physical objects. Example - a number of physical disks are aggregated into a RAID disk.
- Emulation: constructs a virtual object from a different type of a physical object. Example - a physical disk emulates a Random Access Memory (RAM).
- Multiplexing and emulation. Examples - virtual memory with paging multiplexes real memory and disk; a virtual address emulates a real address.

Layering

Layering – a common approach to manage system complexity.

- Minimizes the interactions among the subsystems of a complex system.
- Simplifies the description of the subsystems; each subsystem is abstracted through its interfaces with the other subsystems.
- We are able to design, implement, and modify the individual subsystems independently.

Layering in a computer system.

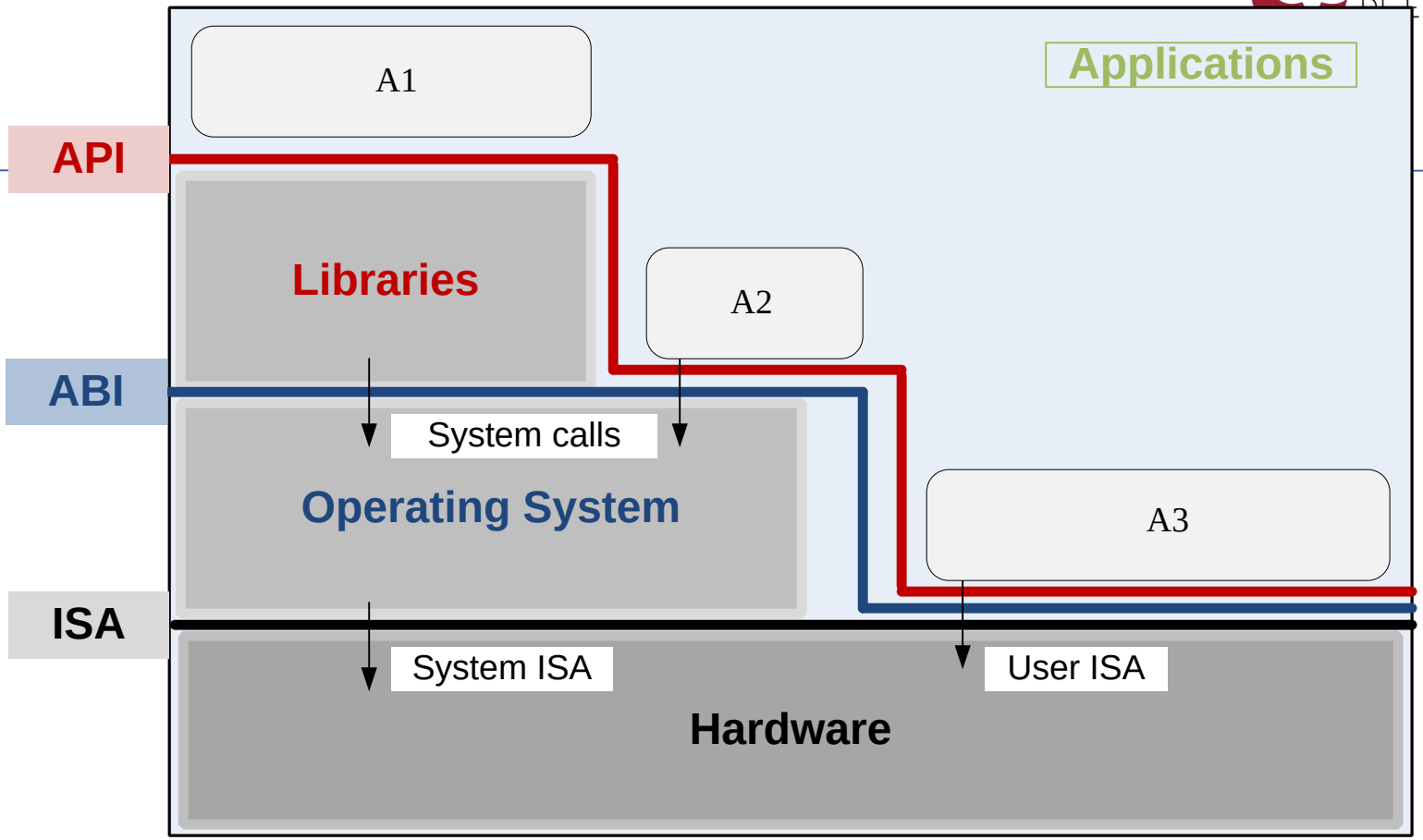
- Hardware.
- Software.
  - Operating system.
  - Libraries.
  - Applications.

Interfaces

Instruction Set Architecture (ISA) – at the boundary between hardware and software.

Application Binary Interface (ABI) – allows the ensemble consisting of the application and the library modules to access the hardware; the ABI does not include privileged system instructions, instead it invokes system calls.

Application Program Interface (API) - defines the set of instructions the hardware was designed to execute and gives the application access to the ISA; it includes HLL library calls which often invoke system calls.



Application Programming Interface, Application Binary Interface, and Instruction Set Architecture . An application uses library functions (A1), makes system calls (A2), and executes machine instructions (A3).

Code portability

Binaries created by a compiler for a specific ISA and a specific operating systems are not portable.

It is possible, though, to compile a HLL program for a virtual machine (VM) environment where portable code is produced and distributed and then converted by binary translators to the ISA of the host system.

A dynamic binary translation converts blocks of guest instructions from the portable code to the host instruction and leads to a significant performance improvement, as such blocks are cached and reused

Virtual machine Monitor (VMM / hypervisor)

Partitions the resources of computer system into one or more virtual machines (VMs). Allows several operating systems to run concurrently on a single hardware platform.

A VMM allows

- Multiple services to share the same platform.
- Live migration - the movement of a server from one platform to another.
- System modification while maintaining backward compatibility with the original system.
- Enforces isolation among the systems, thus security.

Hypervisor

Hypervisors are currently classified in two types:

Type 1 hypervisor (or Type 1 virtual machine monitor) is software that runs directly on a given hardware platform (as an operating system control program). A "guest" operating system thus runs at the second level above the hardware.

- The classic type 1 hypervisor was CP/CMS, developed at IBM in the 1960s, ancestor of IBM's current z/VM. More recent examples are Xen, VMware's ESX Server, and Sun's Hypervisor (released in 2005).

Type 2 hypervisor (or Type 2 virtual machine monitor) is software that runs within an operating system environment. A "guest" operating system thus runs at the third level above the hardware.

- Examples include VMware server and Microsoft Virtual Server.

VMM virtualizes the CPU and the memory

A VMM

- Traps the privileged instructions executed by a guest OS and enforces the correctness and safety of the operation.
- Traps interrupts and dispatches them to the individual guest operating systems.
- Controls the virtual memory management.
- Maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table. This shadow page table points to the actual page frame and it is used by the Memory Management Unit (MMU) for dynamic address translation.
- Monitors the system performance and takes corrective actions to avoid performance degradation. For example, the VMM may swap out a Virtual Machine to avoid thrashing.

Virtual machines (VMs)

VM - isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources.

Process VM - a virtual platform created for an individual process and destroyed once the process terminates.

System VM - supports an operating system together with many user processes.

Traditional VM - supports multiple virtual machines and runs directly on the hardware.

Hybrid VM - shares the hardware with a host operating system and supports multiple virtual machines.

Hosted VM - runs under a host operating system.

Name	Host ISA	Guest ISA	Host OS	guest OS	Company
Integrity VM	*x86*-64	*x86*-64	HP-Unix	Linux,Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Lynx Secure	*x86*	*x86*	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	*x86*-64	*x86*-64	Windows	Windows	Microsoft
Oracle VM	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows	Oracle
RTS Hypervisor	*x86*	*x86*	No host OS	Linux, Windows	Real Time Systems
SUN xVM	*x86*, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware EX Server	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	*x86*, *x86*-64	*x86*, *x86*-64	MAC OS *x86*	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	*x86*, *x86*-64	*x86*, *x86*-64	Linux Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	*x86*	*x86*	Denali	ILVACO, NetBSD	University of Washington
Xen	*x86*, *x86*-64	*x86*, *x86*-64	Linux Solaris	Linux, Solaris NetBSD	University of Cambridge

Performance and security isolation

The run-time behavior of an application is affected by other applications running concurrently on the same platform and competing for CPU cycles, cache, main memory, disk and network access. Thus, it is difficult to predict the completion time!

Performance isolation - a critical condition for QoS guarantees in shared computing environments.

A VMM is a much simpler and better specified system than a traditional operating system. Example - Xen has approximately 60,000 lines of code; Denali has only about half, 30,000.

The security vulnerability of VMMs is considerably reduced as the systems expose a much smaller number of privileged functions.

Computer architecture and virtualization

Conditions for efficient virtualization:

- A program running under the VMM should exhibit a behavior essentially identical to that demonstrated when running on an equivalent machine directly.
- The VMM should be in complete control of the virtualized resources.
- A statistically significant fraction of machine instructions must be executed without the intervention of the VMM.

Two classes of machine instructions:

- Sensitive - require special precautions at execution time:
  - Control sensitive - instructions that attempt to change either the memory allocation or the privileged mode.
  - Mode sensitive - instructions whose behavior is different in the privileged mode.
- Innocuous - not sensitive.

Full virtualization and paravirtualization

Full virtualization – a guest OS can run unchanged under the VMM as if it was running directly on the hardware platform.

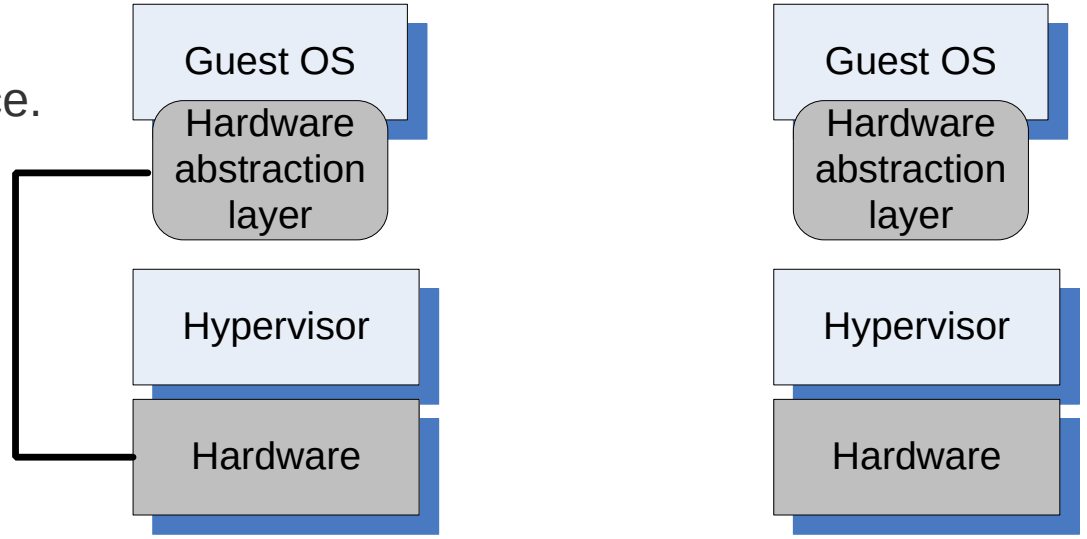
- Requires a virtualizable architecture.

Examples: Vmware.

Paravirtualization - a guest operating system is modified to use only instructions that can be virtualized. Reasons for paravirtualization:

- Some aspects of the hardware cannot be virtualized.
- Improved performance.
- Present a simpler interface.

Examples: Xen, Denaly



(a) Full virtualization (b) Paravirtualization

UNIVERSIDADE DE ÉVORA

Protection Levels

protection levels also known as rings

0 has the highest level privilege and it is in this ring that the operating system kernel normally runs.

- Code executing in ring 0 is said to be running in system space, kernel mode or supervisor mode.
- All other code such as applications running on the operating system operates in less privileged rings, typically ring 3.

Virtualization of x86 architecture

Ring de-privileging - a VMMs forces the operating system and the applications to run at a privilege level greater than 0.

Ring aliasing - a guest OS is forced to run at a privilege level other than that it was originally designed for.

Address space compression - a VMM uses parts of the guest address space to store several system data structures.

Non-faulting access to privileged state - several store instructions can only be executed at privileged level 0 because they operate on data structures that control the CPU operation. They fail silently when executed at a privilege level other than 0.

Guest system calls which cause transitions to/from privilege level 0 must be emulated by the VMM.

Interrupt virtualization - in response to a physical interrupt, the VMM generates a ``virtual interrupt'' and delivers it later to the target guest OS which can mask interrupts.

Virtualization of x86 architecture (cont'd)

Access to hidden state - elements of the system state, e.g., descriptor caches for segment registers, are hidden; there is no mechanism for saving and restoring the hidden components when there is a context switch from one VM to another.

Ring compression - paging and segmentation protect VMM code from being overwritten by guest OS and applications. Systems running in 64-bit mode can only use paging, but paging does not distinguish between privilege levels 0, 1, and 2, thus the guest OS must run at privilege level 3, the so called (0/3/3) mode. Privilege levels 1 and 2 cannot be used thus, the name ring compression.

The task-priority register is frequently used by a guest OS; the VMM must protect the access to this register and trap all attempts to access it. This can cause a significant performance degradation.

Linux KVM (Kernel Virtual Machine)

The most recent news out of Linux is the incorporation of the KVM into the Linux kernel (2.6.20).

KVM is a full virtualization solution that is unique in that it turns a Linux kernel into a hypervisor using a kernel module.

This module allows other guest operating systems to then run in user-space of the host Linux kernel (see Figure in the next slide).

The KVM module in the kernel exposes the virtualized hardware through the /dev/kvm character device.

The guest operating system interfaces to the KVM module using a modified QEMU process for PC hardware emulation.

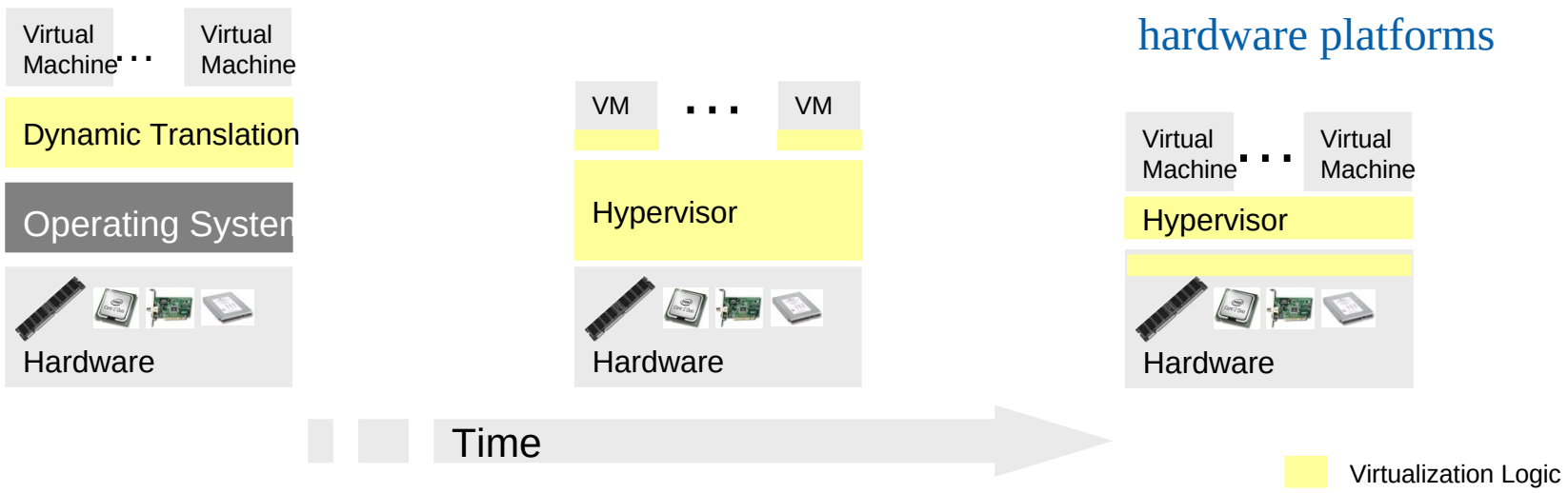
UNIVERSIDADE DE ÉVORA

Virtualization Examples

- Disaster recovery
  - Virtual machines can be used as "hot standby" environments for physical production servers. This changes the classical "backup-and-restore" philosophy, by providing backup images that can "boot" into live virtual machines, capable of taking over workload for a production server experiencing an outage.
- Testing and training
  - Hardware virtualization can give root access to a virtual machine. This can be very useful such as in kernel development and operating system courses.
- Portable workspaces
  - Recent technologies have used virtualization to create portable workspaces on devices like USB memory sticks. Examples:
    - Thinstall
    - MojoPac, Ceedo
    - moka5 and LivePC

Evolution of Software solutions*

- 1st Generation: Full virtualization (Binary rewriting)
  - Software Based
  - VMware and Microsoft
- 2nd Generation: Paravirtualization
  - Cooperative virtualization
  - Modified guest
  - VMware, Xen
- 3rd Generation: Silicon-based (Hardware-assisted) virtualization
  - Unmodified guest
  - VMware and Xen on virtualization-aware hardware platforms



*This slide is from Intel® Corporation

The darker side of virtualization

In a layered structure, a defense mechanism at some layer can be disabled by malware running at a layer below it.

It is feasible to insert a *rogue VMM,* a Virtual-Machine Based Rootkit (VMBR) between the physical hardware and an operating system.

Rootkit - malware with a privileged access to a system.

The VMBR can enable a separate malicious OS to run surreptitiously and make this malicious OS invisible to the guest OS and to the application running under it.

Under the protection of the VMBR, the malicious OS could:

- observe the data, the events, or the state of the target system.
- run services, such as spam relays or distributed denial-of-service attacks.
- interfere with the application.

UNIVERSIDADE DE ÉVORA

Credits, references and reading material

- *Cloud Computing: Theory and Practice*

  Dan C. Marinescu

  Chapter 5

- *Hypervisor, Virtualization Stack, And Device Virtualization Architectures*

  Mike Neil, Microsoft Corporation

- *Distributed and Cloud Computing*

  K. Hwang, G. Fox and J. Dongarra

  Morgan Kaufmann, 2012