Operating system kernel: protection

RECAP FROM COMPUTER ARCHITECTURES AND OPERATING SYSTEMS…

1

Multi-programmed systems

- multi-user system: several programs loaded in memory at the same time
- Resource optimization (processor, memory, devices)

Operating system
Program 1
Program 2
Program 3

2

Challenge: Protection

How do we execute code with restricted privileges?

- Either because the code is buggy or if it might be malicious

Some examples:

- A script running in a web browser
- A program you just downloaded off the Internet
- A program you just wrote that you haven't tested yet

3

Main elements

Process concept

A process is an OS abstraction for executing a program with limited privileges

Dual-mode operation: user vs. kernel

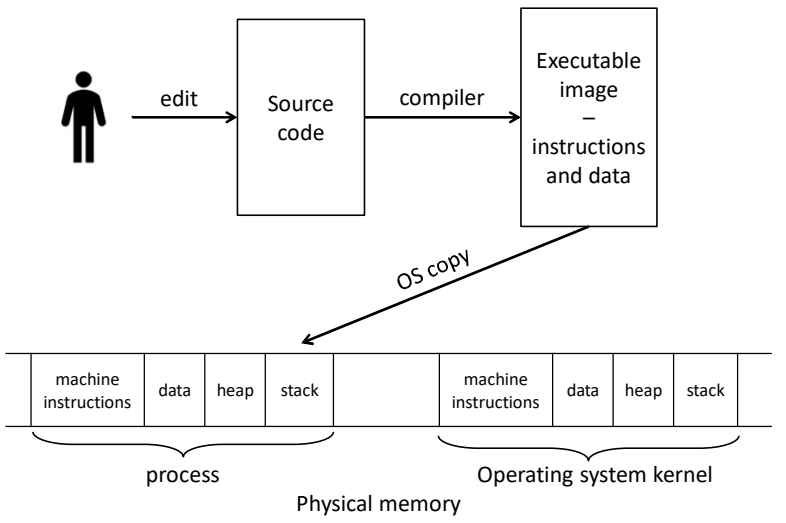
Kernel-mode: execute with complete privileges

User-mode: execute with fewer privileges

Safe control transfer

How do we switch from one mode to the other?

4



Process concept

5

Hardware Support: Dual-Mode Operation

Kernel mode

- Execution with the full privileges of the hardware
- Read/write to any memory, access any I/O device, read/write any disk sector, send/read any packet

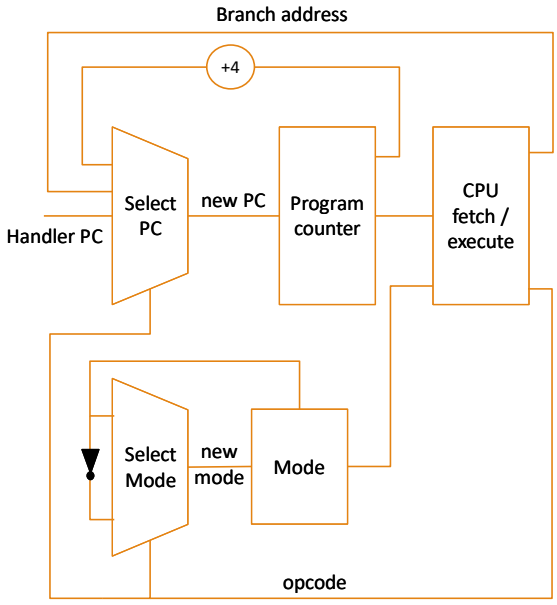
User mode

- Limited privileges
- Only those granted by the operating system kernel

On the x86, mode stored in EFLAGS register

- In general in the Program Status Register

6



A CPU with Dual-Mode Operation

7

Hardware Support: Dual-Mode Operation

Privileged instructions

- Available to kernel
- Not available to user code

Limits on memory accesses

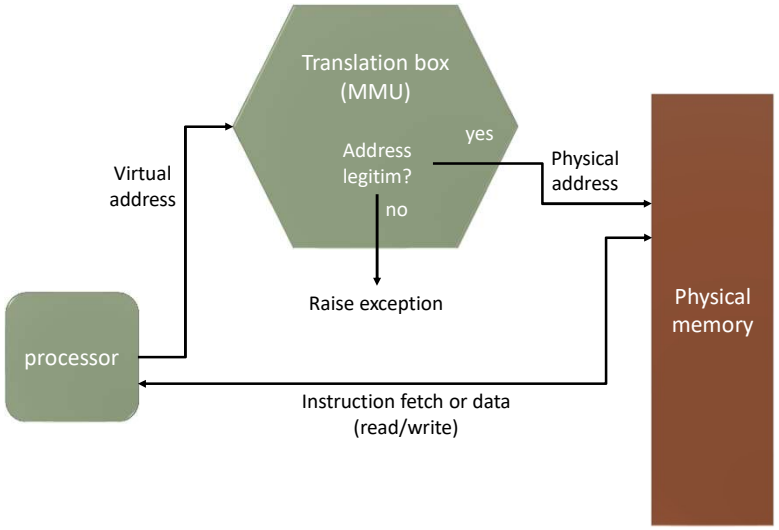
- To prevent user code from overwriting the kernel

Timer

- To regain control from a user program in a loop

Safe way to switch from user mode to kernel mode, and vice versa

8



Memory protection

Virtual memory
- Translation done in hardware, using a table
- Table set up by operating system kernel

9

Mode Switch:

user-mode to kernel mode

- Interrupts
  - Triggered by timer and I/O devices
- Exceptions
  - Triggered by unexpected program behavior
  - Or malicious behavior!
- System calls (aka protected procedure call)
  - Request by program for kernel to do some operation on its behalf
  - Only limited # of very carefully coded entry points

10

Mode Switch:

kernel-mode to user mode

- Return from interrupt, exception, system call
  - Resume suspended execution
- New process/new thread start
  - Jump to first instruction in program/thread
- Process/thread context switch
  - Resume some other process
- User-level upcall
  - Asynchronous notification to user program

11

How to take interrupts safely?

Interrupt vector
- Limited number of entry points into kernel

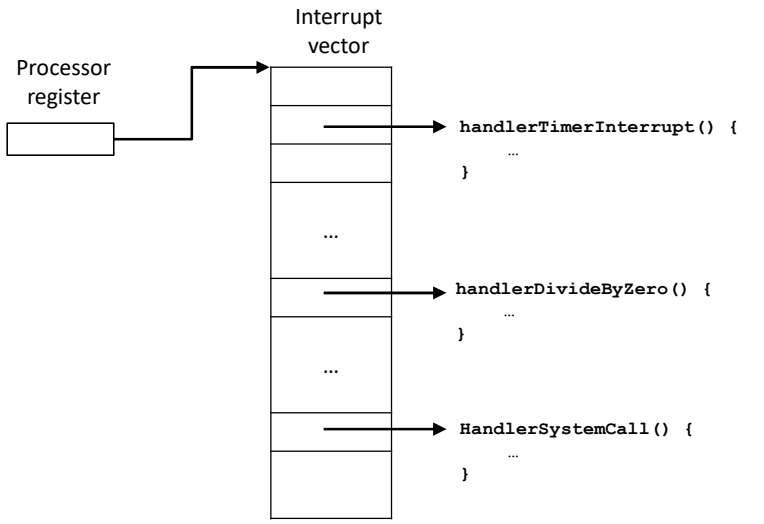
Kernel interrupt stack
- Handler works regardless of state of user code

Interrupt masking
- Handler is non-blocking

Atomic transfer of control
- Single instruction to change:
  - Program counter
  - Stack pointer
  - Memory protection
  - Kernel/user mode

Transparent restartable execution
- User program does not know interrupt occurred

12



Interrupt vector

Table set up by OS kernel; pointers to code to run on different events

13

Atomic Mode Transfer

On interrupt (x86):

In Hardware

- Save current stack pointer
- Save current program counter
- Save current processor status word (condition codes)
- Switch to kernel mode
- Vector through interrupt table
- Switch to handler PC & kernel PSW

- Interrupt handler saves registers it might clobber

14

At end of handler

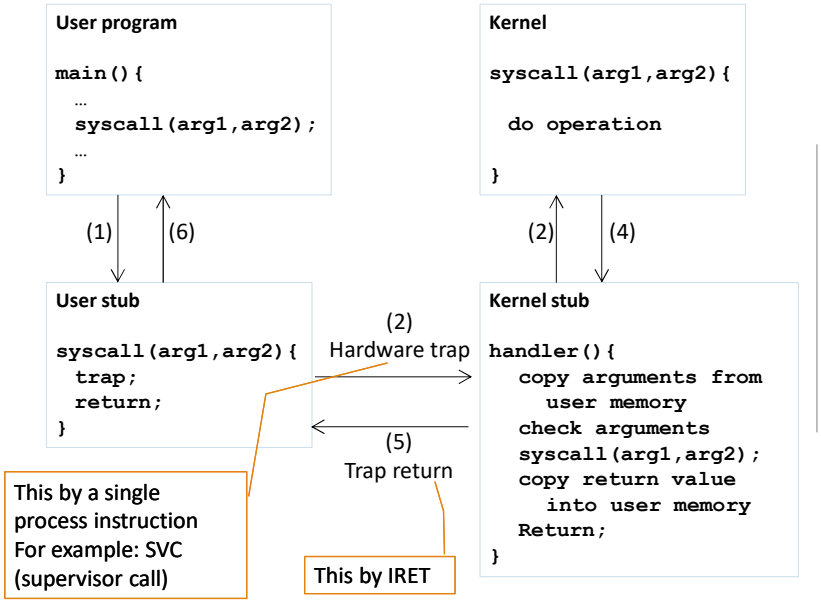
Handler restores saved registers

Atomically return to interrupted process/ thread (IRET instruction)

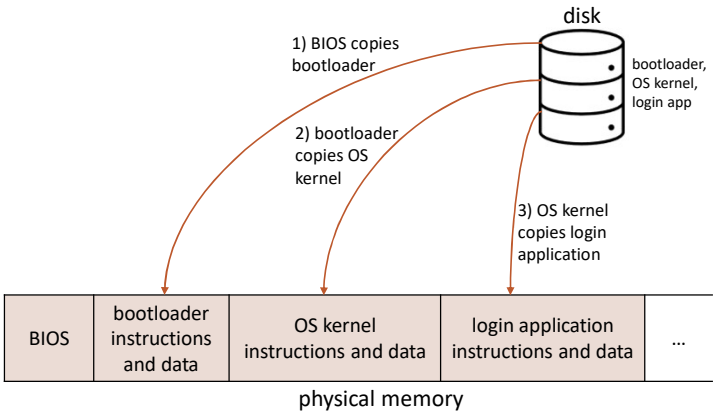
- Restore program counter
- Restore program stack
- Restore processor status word/condition codes
- Switch to user mode
- Enable interrupts

15

System calls



16



Booting

17

Other protection methods

18

System call filtering

Usual protection methods assume that the kernel can be trusted

Protection in fact leverages the dual mode user/kernel and the system calls to switch between the two modes

However, an attack may be directed to system calls unprotected

System-call filtering introduce an additional level of control of system calls

- May limit the kind of system calls a process can invoke
  - Like a firewall…
  - Based on profiles specific for a process
- May also inspect the system call argument
  - May be necessary because also apparently innocuous system calls may reveal problematic
  - The case of fast mutex (futex) in Linux
- An example is the SECCOMP-BPF (Berkeley packet filtering) used in Linux

19

Sandboxes

Normally a process is executed with the credential of its owner

- However, the process does not often need all these privileges

With sandboxing a process is run in a constrained environment:

- Impose set of irremovable restrictions early in startup of process, much before the execution of its `main()`
- Limit system calls
- Process cannot access any resources beyond its allowed set

Examples:

- Java and .net implement sandboxing at the level of the virtual machine
- Android implements sandboxing by combining Linux system calls filtering and a strong policy of mandatory access control (MAC)

20

Code signing

Consists in using crypto hashes to let the developer sign his code

- Ensures that the code is exactly as when compiled by the developer
- Nobody can change the code without being detected
  - Any change alters the signature

The system may decide whether to trust that specific developer and consider that code trusted

Can also be used to disable obsolete programs

- Just by invalidating the signature

21

Compile-based/ Language-based protection

High level programming languages enforce protection mechanisms in the generated codes

- Leverage high-level description of policies for the allocation and use of resources

These mechanisms also work without specific hardware support

Maps language-specific protection mechanisms on the protection mechanisms provided by the underlying OS and hardware

JAVA is a notable example of language enforcing protection

22

Compile-based/ Language-based protection: JAVA

JAVA enforces:

- type safety (by means of load-time and run-time check)
- encapsulation to protect data and methods of other classes

Furthermore, a class in Java is assigned with a protection domain when it is executed by the java virtual machine (JVM)

- the protection domain specifies what the class can do

The protection at run-time is enforced by the JVM:

- if a class invokes a method in a library, and the method need to perform a privileged operation…
- … the JVM inspects the stack to check the arguments of the invocation of the method and approve or block it

23

Virtual machines and virtualization security

IN PART FROM "OPERATING SYSTEMS CONCEPTS", 10TH EDITION

A. SILBERSHATZ, P. B. GALVIN, G. GAGNE

24

Virtualization

- A technology that provides an abstraction of the resources used by some software which runs in a simulated environment called a virtual machine (VM)
- Benefits include:
  - better efficiency in the use of the physical system resources,
  - isolation of different OS/SW systems over the same Hardware (of interest for security)
  - simplified management of OS (restart, migration, checkpointing, cloning etc…)
- Provides support for multiple distinct operating systems and associated applications on one physical system

25

* On this VM you want to run an OS. This OS is using everything of the HW. So the OS expects all the features it needs. If something is missing we'll have a problem.
  - But there are ways to relax this requirements.

Virtualization

- Abstracts the hardware of computer into several "virtual" computers (**virtual machines**):
  - Each virtual computer is (in general) identical to the underlying one *
  - It provides an independent execution environment in which an operating system (but even an application) can run
  - A single physical computer can host multiple operating systems, each in its own virtual machine
- Analogies with the layers of an operating system: ①
  - Each layer of an OS creates an abstraction of the underlying layers
  - The virtual machine creates an abstraction of the HW into several HW
- Typical interplay among three elements: has to deal with 3 elements:
  - The underlying hardware (**Host**)
  - The **Hypervisor**: the software that implements the virtual machines abstraction
  - The **guest** Operating System that runs on the virtual machine
    - running in VM.

and of all physical resources

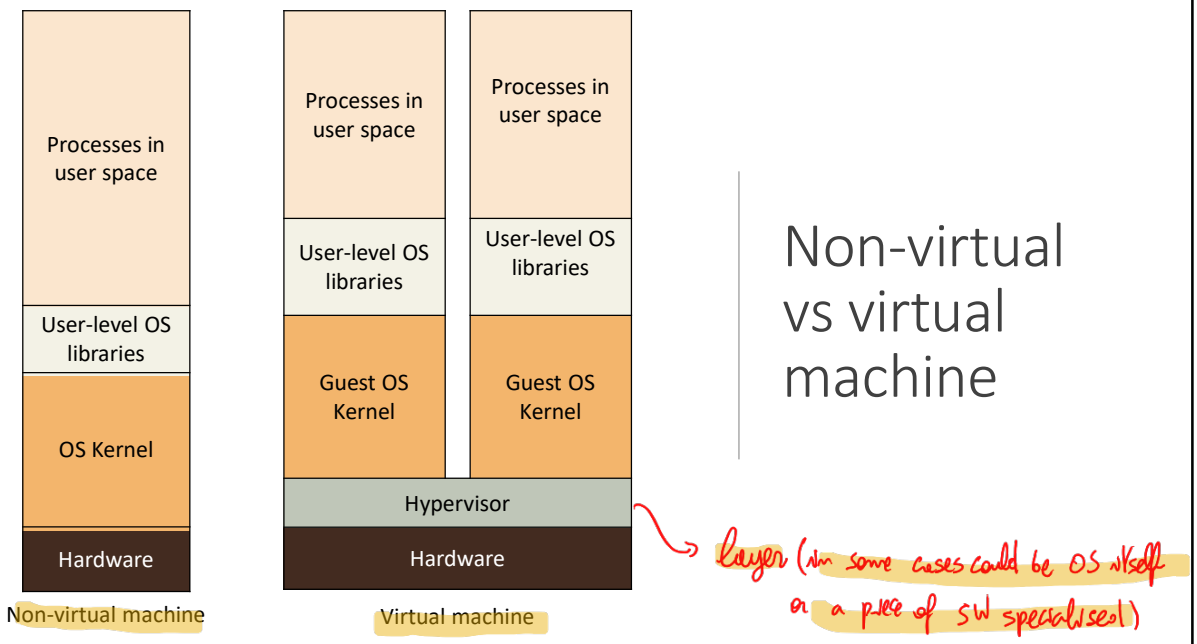
26

① Something that virtualizes the HW in something identical. OS does something similar, but here we do this.

Virtualization

- Introduced by IBM in 1972 with the VM/CMS operating system for the IBM 360 mainframe
- Composed by two OSs:
  - VM: implements virtual machines over a physical machine
  - CMS: a single user batch operating system that runs over a virtual machine
- Main requirements:
  - A virtual machine is identical to the physical machine
  - The overhead should be negligible (limited performance loss due to execution on a virtual machine)
  - Security: each virtual machine is isolated from the others
- The idea proved successful, nowadays virtualization widely used
  - Now widely supported on many CPUs and OSs
  - Adopted in cloud systems and even in personal OSs

27



Non-virtual vs virtual machine

28

HW is an operating system itself, acts as a manager of resources (a broker)
Remember that everything is shared, so Hypervisor should manage this all

The Hypervisor

- Software that sits between the hardware and the VMs
  - We may properly say that an Hypervisor is an OS itself
- Acts as a resource broker
- It allows multiple VMs to safely coexist on a single physical server host and share that host's resources
- Virtualizing software provides abstraction of all physical resources and thus enables multiple computing stacks, called virtual machines, to be run on a single physical host
- Each VM includes an OS, called the guest OS
  - This OS may be the same as the host OS, if present, or a different one

29

Should manage the execution of VM: scheduling for VMs! Manage isolation of VM.
Hypervisor might need to

Hypervisor Functions

The principal functions performed by a hypervisor are:

- Execution management of VMs: scheduling, isolation, …
- Devices emulation and access control
- Execution of privileged operations by hypervisor for guest VMs
- Management of VMs (also called VM lifecycle management) *
- Administration of hypervisor platform and hypervisor software *2

User Mode / Kernel mode: guest OS would be using kernel mode for what they need to do. But Hard W is shared between VMs! If hypervisor restricts kernel mode to itself, this means

30

that OS can't run in Kernel mode! Kernel mode should be reserved by HW. But if OS is running in User mode it will not be able to run properly. The OS will attempt to use HW directly. This is done through interrupts.

* VM could be stopped, moved among etc.

*2: HW is an OS in the end. It is doing a lot of work

Advantages of virtualization

- Protection:
  - Virtual machines independent and protected from each other
  - Host and Hypervisor protected from the virtual machines
  - Virtual machines may share resources only through networking or shared file system
- A virtual machine can be suspended, frozen and run later:
  - The VM can be duplicated,
  - shared with others
  - or restored to a previous state
  - Very useful in a cloud environment and for OS developers (see next slide)

31

Advantages of virtualization

- Advantages in the cloud:
  - Can create template VM (include OS and applications) to be distributed to several customers
  - Can migrate a VM on another host without interrupting services
  - Cloud services usually offer APIs to activate these and other functionalities
- Advantages for OS developers:
  - Can test device drivers or kernel modules without any risk (the VM can be easily restored)

32

Virtual Machines Implementations

A virtual machine has a lifecycle:

- At creation specify (virtual) HW requirements
  - Number of CPUs, memory, disk space, network interface,…
- These resources implemented over HW resources that can be shared or dedicated to the VM
- When the VM is no longer needed, its resources can be freed and reassigned to other VMs

→ and the you assign those resources to another one

Overall setting up and maintaining a VM is much simpler than installing a physical one

- Can also exploit clones of VM already prepared

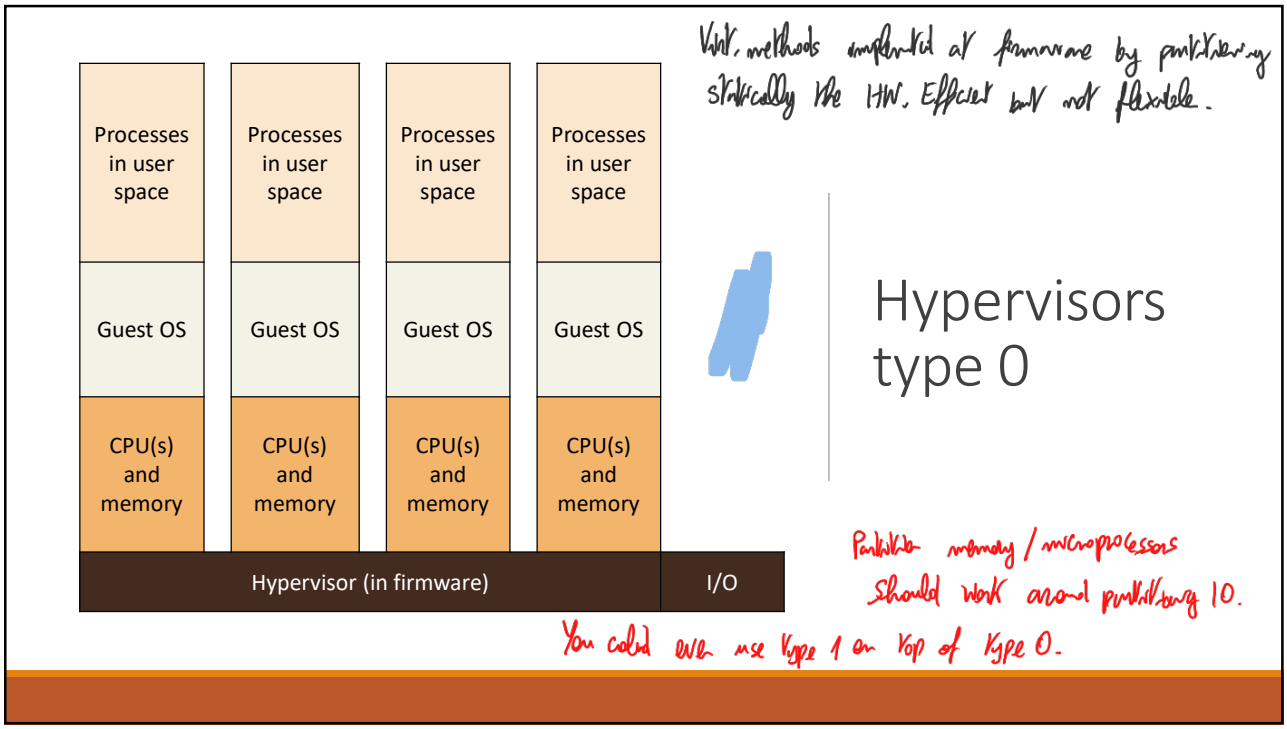
Different ways of implementing virtualization, also depending on the HW support

33

Types of Hypervisors

- **Hypervisors type 0:** if implemented at firmware (part of the HW) not very flexible: will have a physical partition of resources, but efficient.
  - implemented in firmware
  - Examples: Logical partitioning (LPAR) or POWER Hypervisor (PHYP) by IBM
- **Hypervisors type 1:**
  - Software hypervisors, implement virtualization directly on hardware (AKA bare-metal hypervisors)
    - Examples: VMware ESXi, Oracle VM Server, …
    - Operating systems on top of hypervisor (AKA native hypervisors)
  - In some cases are full OS with virtualization services
    - Examples: Microsoft HyperV, RedHat Linux with KVM
- **Hypervisors type 2:**
  - Run over a conventional OS
  - Guest OS runs as a process of the hosting OS
  - Examples: Oracle VirtualBox, Vmware, Parallels Desktop

34

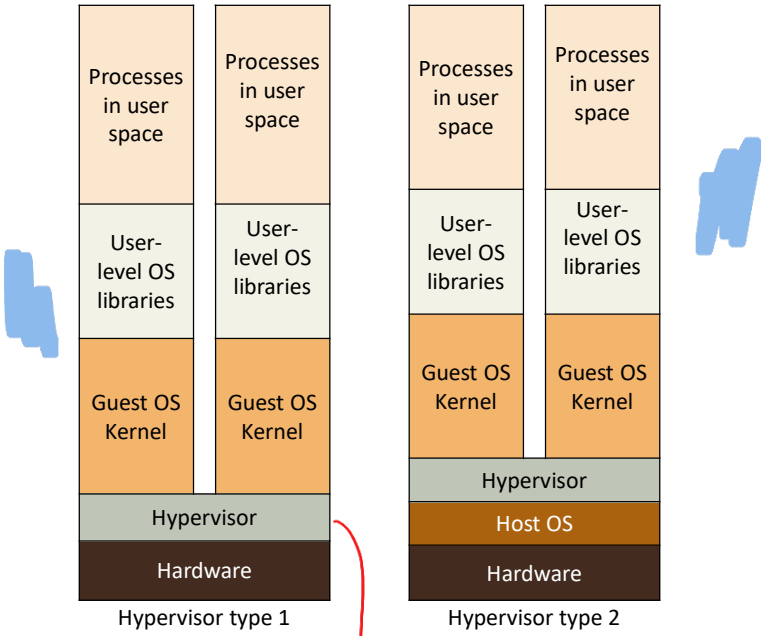


35

Hypervisors type 0

Entirely implemented in the firmware of the CPU
- Software does nothing, OSs just installed as guest hosts
- Offers a smaller set of functionalities as compared to other types of hypervisor
  - Firmware implementations are quite "rigid"
- Often based on physical partitioning of the physical resources
  - Can be difficult for I/O devices if they are not enough, in this case either:
    - Devices must be shared, or
    - Devices are virtualized and their access is through a "remote" partition and virtual devices are implemented by daemons

May also provide virtualization-within-virtualization (a guest can be a virtual machine with other guests)



Type 1 and type 2 Hypervisors

Implemented in other cases.
Hypervisor is an OS itself.
Access to HW is mediated by the HW.

37

which is an OS itself. Very flexible, more powerful, but lose a bit of performance. You will have to pay.
You do not have a rigid partition [1GB to them? Ok, half of one GB]

Hypervisors type 1

Commonly used in datacenters

- Often viewed as the operating systems of the datacenters
- In some cases they are special purpose operating systems that run over bare metal
  - Do not offer system calls to the upper layers…
  - …rather, they offer functionalities to create and manage virtual machines
- In others they are general purpose operating systems also offering virtualization capabilities
  - Typically they are less powerful and offers less features of virtualization than dedicated type 1 Hypervisors
  - Treat guest OSs as user-level processes

38

Hypervisors type 1

- Run in kernel mode
- Implement device drivers for the hosting hardware
- Guests don't know they are running in a virtual machine
- Also provide other traditional OS services like CPU and memory management
- Cannot run over other type 1 Hypervisors
  - But they can run on top of type 0 Hypervisors

39

Hypervisors type 1

In general, Hypervisors type 1 offer several advantages:

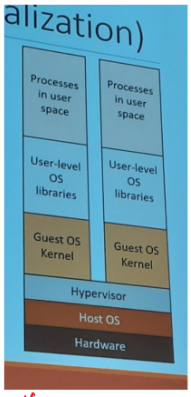
- Datacenter managers can control and manage several guest OSs in a simple and sophisticated way, just by acting on the Hypervisor
- Allows for consolidation of OSs and apps onto less HW
- Move guests between systems to balance performance
- Snapshots and cloning

40

Hypervisors type 2

No host OS involvement in virtualization:

- The virtual machine is just one of the processes managed by the host OS
  - Even the host OS does not need to know of the existence of a virtual machine
- They have severe limitations:
  - The virtual machine is executed without any special privileges
  - In some cased do not even take advantage of specific HW features for virtualization
  - Tend to have very poor performances
- They also have advantages:
  - They do not require any change to the hosting OS
  - Easy to deploy and use



41

Questions

Explain why it is important to secure the boot process in a computer.

Limiting the media from which the system boots can be considered a security requirement?

It is recommended that while using a hypervisor, the access to the hypervisor should be limited to authorized administrators only. Why?

42

Mechanisms for virtualization

43 The point of V. is: taking an hardware (microproc., memories, I/O) and reproduce a virt. copy of this all, identical to original, and give each copy to a VM and install an OS on the VM directly on the HW. But one thing is virtualization of memory, processors and devices. Processor is virtualised by means of time sharing, and memory by means of virtual memory.

For disks you might create partitions. But for the rest is very difficult: network card for ex: traffic directed to VM should be multiplexed to the VM.

Virtualized Systems

- In virtualized systems, the available hardware resources must be appropriately shared among the various guest OS's
- These include CPU, memory, disk, network, and other attached devices
- CPU and memory are generally partitioned between these, and scheduled as required
- Disk storage may be partitioned, with each guest having exclusive use of some disk resources
  - Alternatively, a "virtual disk" may be created for each guest, which appears to it as a physical disk with a full file-system, but is viewed externally as a single "disk image" file on the underlying file-system
  - Attached devices such as optical disks, or USB devices are generally allocated to a single guest OS at a time
- Several ways to share network access (give direct access to a guest, virtualize the network interface etc…)

44

So implementing perfect V. is very difficult. We have seen V. only in recent years. If we talk about recent machines, x86 is the most common microproc. family. For them it was thought V. was impossible. What's the problem? OS has to protect themselves from user processes and prevent u. processes to interfer with each other and OS. This is achieved through double mode, KM, UM.

The OS uses this reserving KM for itself. If you are implementing a V., the OS is reserving KM for itself, and you are running a process that operates as a VM in which you install an OS that wants to work in KM. There are complex solutions, that work under the assumption in which you can recognise when a process needs KM legitimately

Virtualized systems

Perfect virtualization of a host is rather difficult to implement

- For years in x86 CPUs was reputed impossible to achieve
- Is in general difficult for CPUs that provide only dual-mode operations
- Modern CPUs offer a better support to virtualization
- VMMs introduce the abstraction of virtual CPU
  - represents the state of a CPU as it is expected to be by a Guest OS
  - Concept similar to that of the context of the process, but here is the entire CPU state that is stored and restored

Only recent years introduced solutions.

45

Required to let a code that would need KM, can run in US. If something requires KM, HW will raise an exception (Trap). The real OS discovers that something is wrong and Hypervisor will emulate. We will have enough info to know.

1st method for V.

Trap and Emulate Mechanism

- CPU has a double state: user and kernel mode
- The Host OS runs in kernel mode
- Dual mode CPU means guest executes in user mode
  - However, the Guest Kernel believes it runs in kernel mode…
  - … executes privileged instructions…
  - … and expects free access to memory
  - But it is not safe to run the Guest Kernel in the CPU Kernel mode!

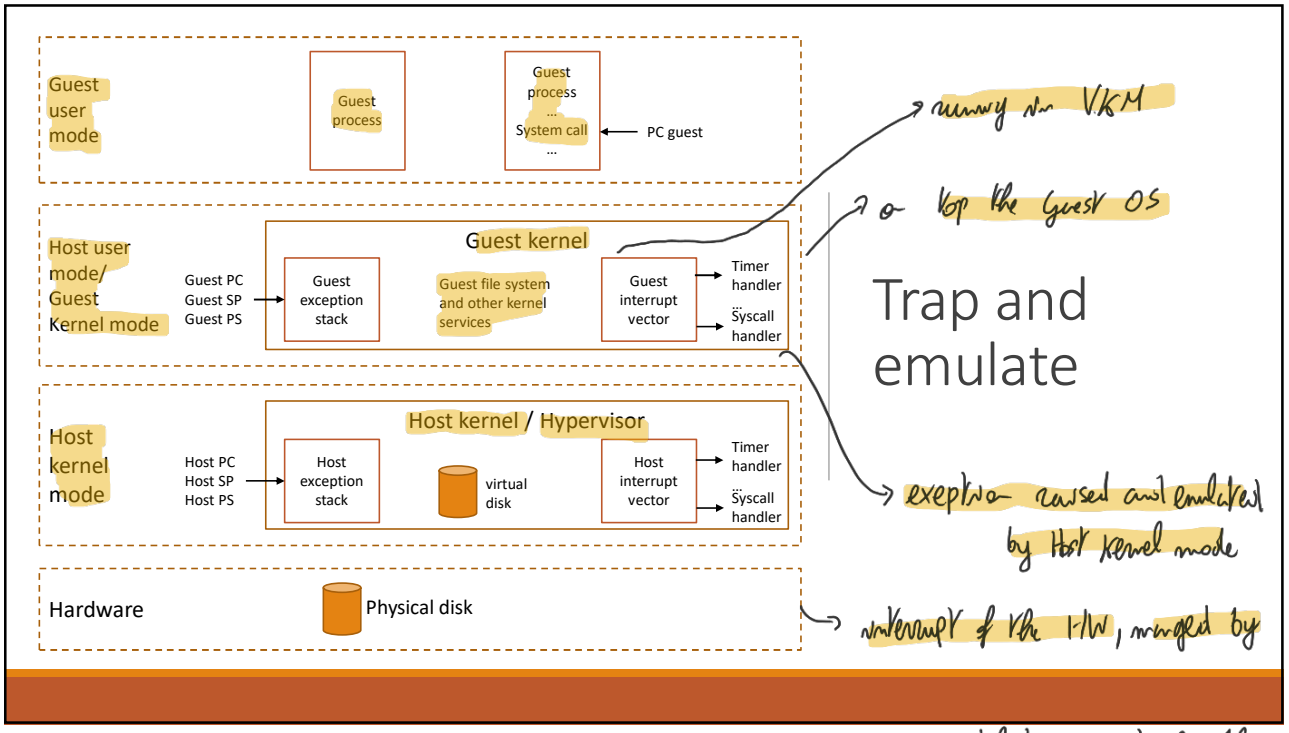
46

VM works like: at the HWor you decide that you want to schedule a VM (like a process), so you put it into execution with usual measures of running a process in UM.
This OS runs as a User level process. When instructions specialised for KM are executed (ex: anything that plays with the flag of mode, Return from interrupts, accessing parts of memory with interrupt vectors etc), an exception is raised. [Type 2] ①

Trap and Emulate Mechanism

- So how does VM Player work?
  - Runs as a user-level application
  - How does it catch privileged instructions, interrupts, device I/O, …
- Installs kernel driver, transparent to host kernel ① Change the ways in which interrupts are managed.
  - Requires administrator privileges! When an interrupt is raised, you have to know whether it is for host or guest OS. ②
  - Modifies interrupt table to redirect to handlers supporting virtualization…
  - If interrupt is for VM, upcall
  - If interrupt is for another process, reinstalls interrupt table and resumes kernel
- The Guest OS thus runs in "virtual" user mode and "virtual" kernel mode
  - Both actually run in the real user mode
  - Operations in the virtual kernel mode are emulated in kernel mode by the host OS by means of a data structure (virtual CPU) = data struct where you record whether VM is running in VUM, VKM, or info of the CPU of the virtual machine. You are emulating only OPS that run in KM.

47 ② The interrupt handler needs to know who should manage the interrupt and do that.



48

In a type 2 you would need a driver for this modification.

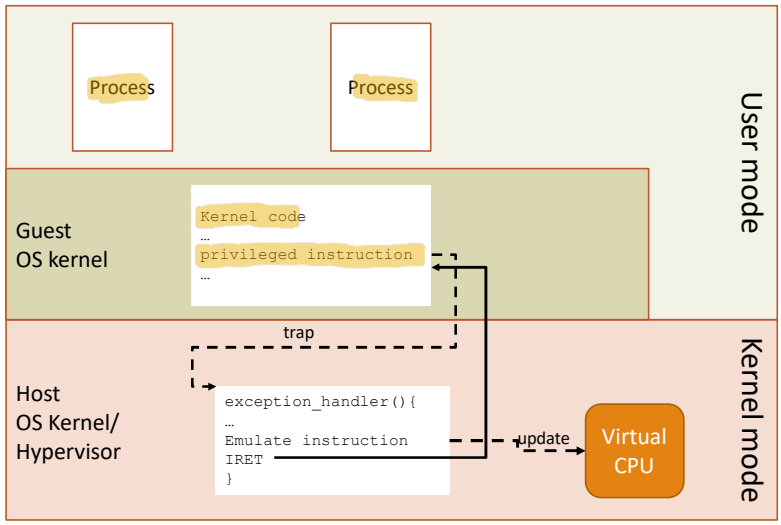
To emulate KM for guest, several things to manage: case of instruction in KM to be emulated, case of emulating a interrupt exc.
If user in VM calls a system call, the sys call will be redirected to the hypervisor, but should be managed by guest OS!

Trap and emulate

- When a process in the guest host runs in user mode no problem
  - Processing occurs at full speed of CPU in user mode
- However, the guest host need to switch to virtual kernel mode:
  - The process attempts to execute a privileged instruction → trap
  - The process violates protection → trap
  - The process invokes a system call to the guest kernel → trap
- In these cases:
  - The host kernel gains control,
  - Analyses the reason for the trap
  - Executes the privileged operation requested by the guest process
  - Returns control to guest in user mode
- This is known as **trap-and-emulate**
  - Used by most virtualization systems

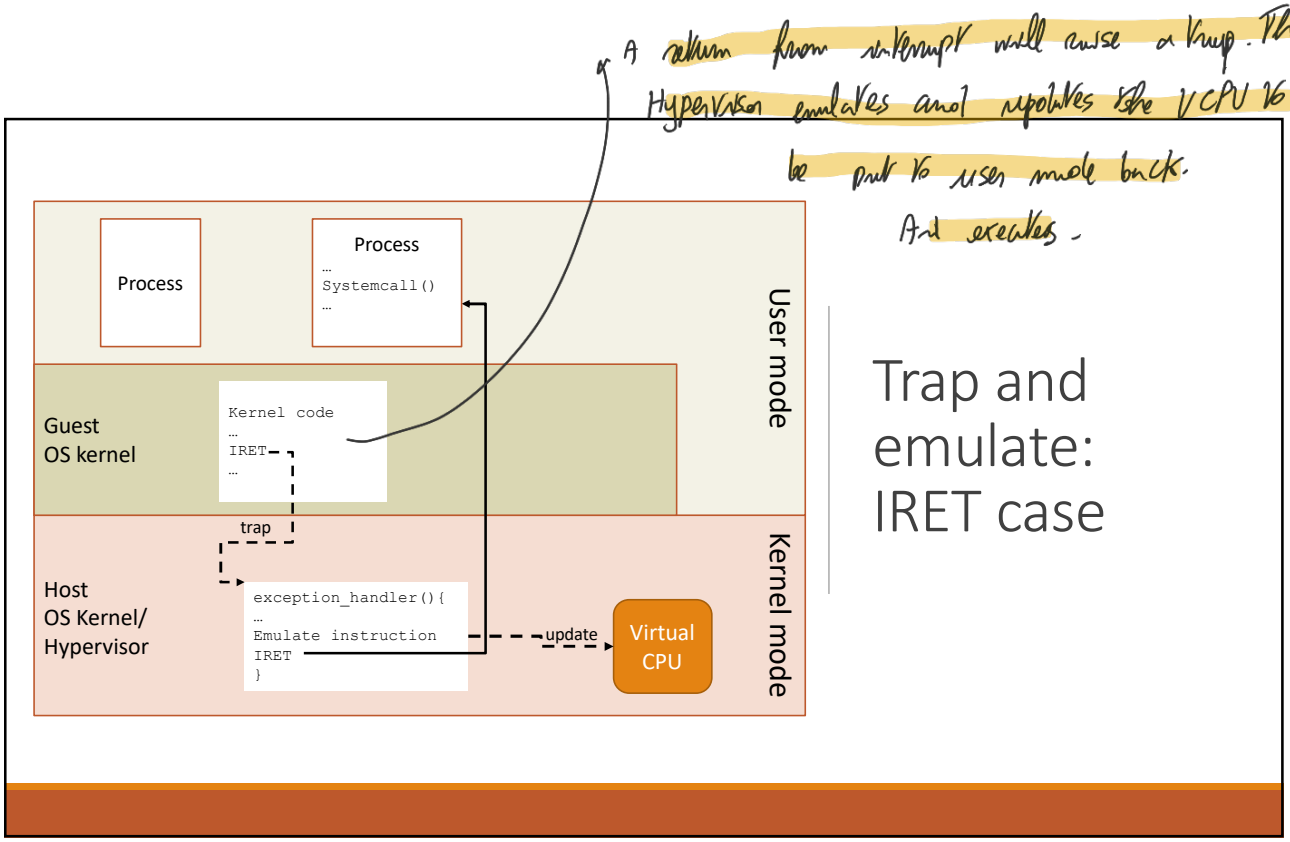
49

If we need a KM instr. the microprocessor will raise an ex. for the host OS. So the hyp ruset for the HWor sees that guest was supposed to run in KM, emulates the execution and returns.

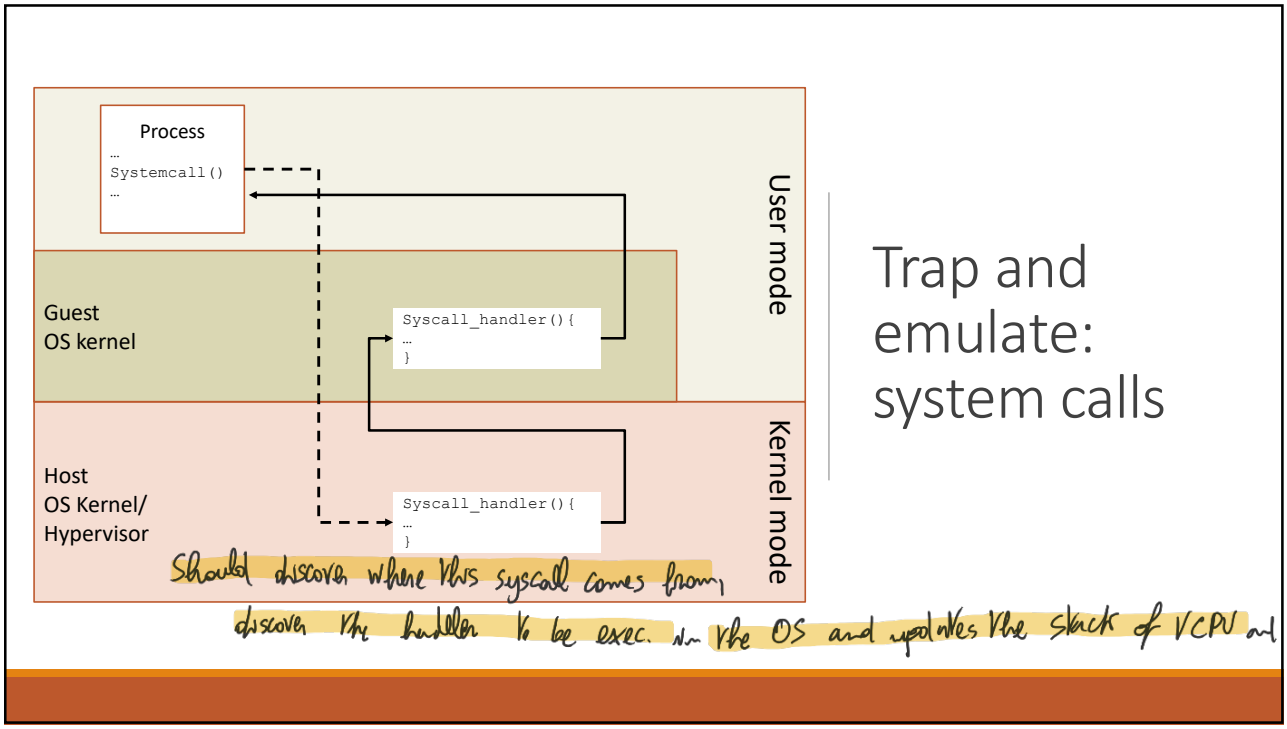


Trap and emulate: privileged instructions

50


A return from interrupt will cause a trap. The Hypervisor emulates and updates the VCPU to be put to user mode back. And executes.
Process
Process
…
Systemcall()
…
User mode
Guest
OS kernel
Kernel code
…
IRET
…
trap
Host
OS Kernel/
Hypervisor
exception_handler(){
…
Emulate instruction
IRET
}
update
Virtual CPU
Kernel mode
Trap and emulate: IRET case

51

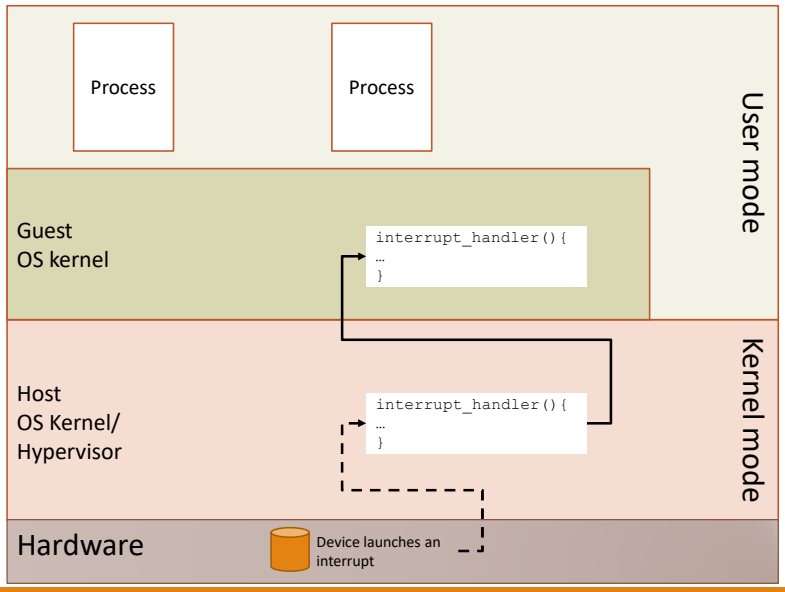

Process
…
Systemcall()
…
User mode
Guest
OS kernel
Syscall_handler(){
…
}
Host
OS Kernel/
Hypervisor
Syscall_handler(){
…
}
Kernel mode
Should discover where this syscall comes from, discover the handler to be exec. in the OS and updates the stack of VCPU and calls the handler of OS, supposed to run in KM. So the host knows that the guest OS should be working in KM.
JMP up
Trap and emulate: system calls

52

Trap and emulate

- However, the guest OS kernel need to interact with devices and receive their interrupts
- The mechanism is always the same…

53



Trap and emulate: managing interrupts

54 Last case: interrupt from HW, and discovers that this was caused by a Guest OS. The Host sets in VCPU that we are supposed to run in KM the VM and make the guest handle the interrupt.

So TAE very complicated. Requires these mechanism.

Trap and emulate

- User mode code in the guest OS runs at full CPU speed
- The slow-down is for the execution of only privileged instructions
  - The trap-and-emulate has an overhead similar to a system call
  - With more support by the hardware this overhead can be reduced

Overhead is limited for KM ops.

55 Prob: with this approach VO V. you can always recognise when a user mode is trying to execute OR that requires KM. But!

Trap and emulate

Unfortunately, some CPUs don't have clean separation between privileged and nonprivileged instructions = instruction that have different effects when executed in UM/KM.

- This is the case of Intel x86 CPUs
- For example the x86 `popf` instruction:
  - Loads CPU flags register from data on top of the stack
  - If the CPU in kernel mode → all flags are replaced
  - If the CPU in user mode → only some of the flags are replaced
  - Legacy with earlier versions of the machine language set
- This means that executing a `popf` does not generate a trap!
  - It is not privileged…

popf in UM not generate any traps! But if you need the full behaviour of popf this won't work.

In these CPUs the trap and emulate mechanism had been considered not applicable for many years… until 1998 considered impossible to virtualise.

56

if switching to VKM in guest OS, before executing code you inspect it and replace it. So you have to inspect Machine language code and slow problem. Very very hard.

Binary Translation

Idea:

- If the guest system runs in user mode → no problem… let it go
- If the guest system has to switch the VCPU in virtual kernel mode:
  - The Hypervisor checks all the next instructions of the guest kernel
  - The "problematic" instructions like the `popf` are translated into others equivalent (or emulated by inserting a trap)
  - The other instructions can just run

Not a nice perspective the performance can be very poor… however…

- Most of the instructions are executed in native mode anyway
  - The "problematic" cases are rare
- Can use several optimizations
  - For example, caching of previously translated code is particularly useful…
- Tested on real OS as guests the overhead was largely acceptable (around 5% of slowdown over native code)

↳ but studies say that overhead is about of 5% of the microproc. speed.

57

Modern microprocessors have evolved to support virtualizations. This became heavy many studies, so real KM, UM, guest KM, UM.

Hardware support to virtualization

Virtualization much more efficient with appropriate hardware support

Modern processors provide specific instruction sets:

- VT-x instructions in Intel processors (since 2005)
- AMD-V instructions in AMD processors (since 2006)

Provide support to:

- Avoid the need for binary translation
- Introduce more CPU modes (guest mode / host mode)
- DMA, memory management, interrupt, …

The hypervisor can:

- Set the features of the virtual machine associated to each guest OS
- Sets the VCPU for each guest → works in real KM.
- Switch to guest mode to run the guest *

In OS in guest mode: believes it is working directly over HW, but is in guest KM.

- thinks it is running natively and see the virtual machine specified by the hypervisor
- If it tries to access to a virtualized device → trap to the hypervisor

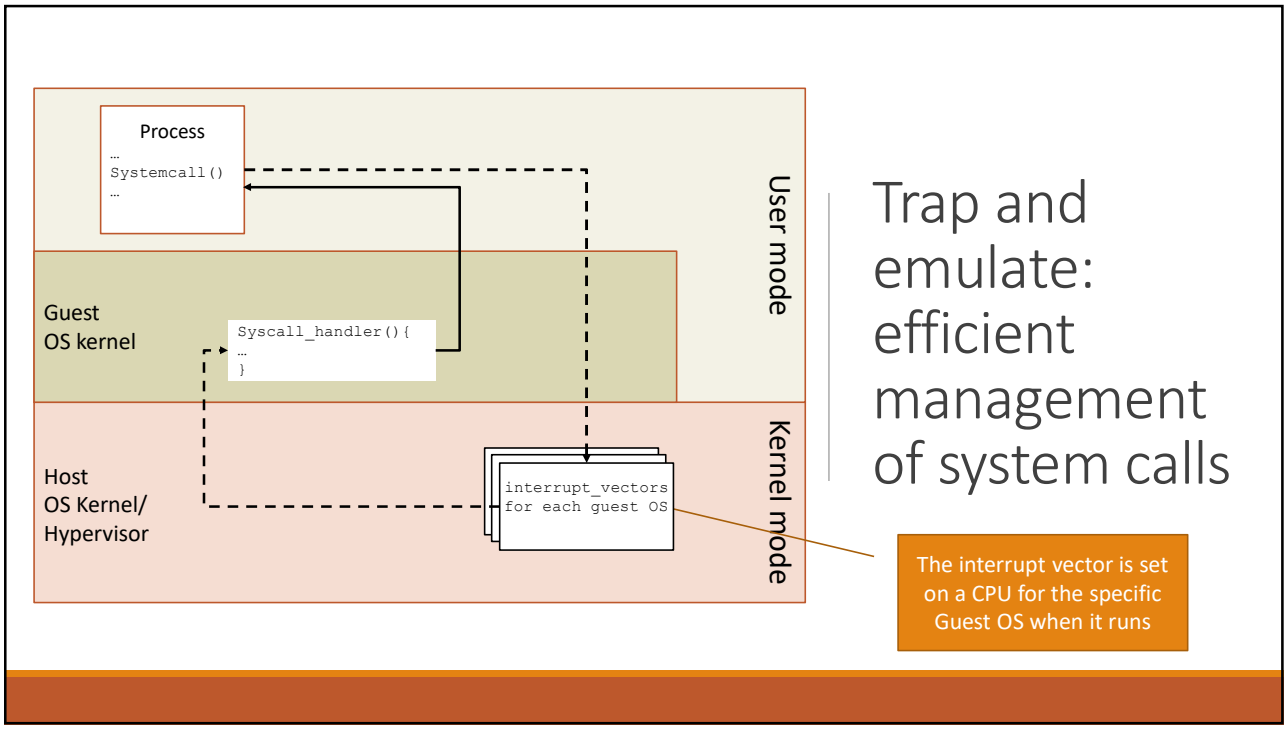
means increasing instruction set. Avoids bynary translation, introduce new modes, and provide new modes for resource management over in guest environment.

58

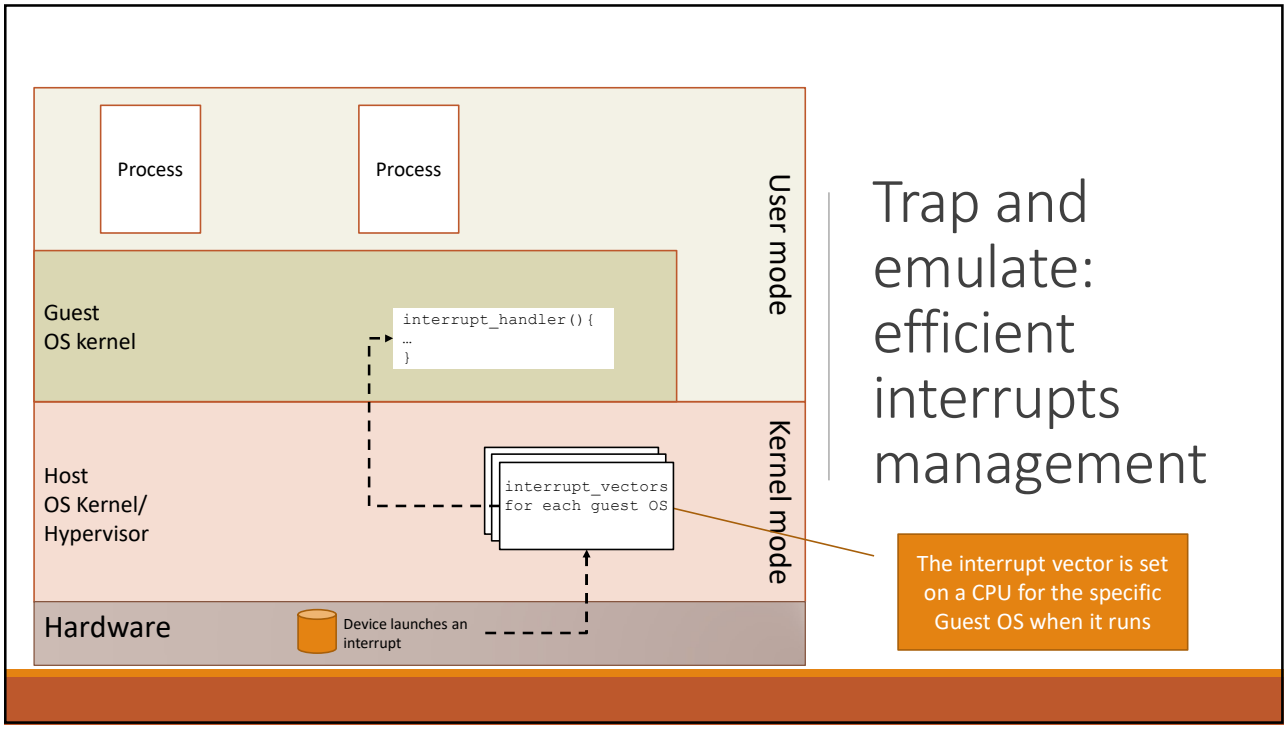
* guest OS run in GUM, GKM. So it's a bit simpler.

To improve efficiency, you can have multiple interrupt vectors associated to the different machines. If you receive an interrupt from HW, for example.

11/17/2020



59



60

In a M you have a # of microprocessors, and over these you will run VM that have a number of Microprocessors, the VM will run on a fraction of time of the Microprocessor. The OS thinks that the MP are all for themselves. Time perceived by the OS could be different than the real time.

Impact of virtualization on underlying OS: scheduling

Operating systems virtualize the CPUs:
- over a single CPU the OS implements several virtual CPUs, in total one virtual CPU for each thread
- virtualization of the physical CPUs by means of scheduling and context switch mechanism

A virtual machine:
- takes several virtual CPUs (implemented by possibly more than one physical CPU)
- the number of virtual CPUs assigned can be reconfigured on the fly
- the guest OS, in turn, further virtualizes the assigned virtual CPUs for its own and application threads

Overcommitment when the virtual machines are allocated a number of virtual CPUs larger than the number of physical CPUs present in the system
- In this condition it's difficult to keep fairness in the scheduling
- Each virtual machine takes a fraction of the physical CPU time

61 GOS schedules for the processes thinking we have all the time possible but not like that. In addition, you have HyV. overhead.

Impact of virtualization on underlying OS: scheduling

In addition, the hypervisor (and the host OS) takes some CPU time for it's own work
- AKA cycle stealing
- Guest OS don't take as many CPU cycles they expect

Cycle stealing may impact on:
- Responsiveness of guest OSs
- Even difficulties in keeping the right time of day

62

Physical memory managed by the Kernel, to each of the process you virtualize the space in a space larger than the actual memory. If this is implemented in a GVM, a Virtual Machine works within a virtual memory, used as a real memory by the GOS. So high overhead for those layers. NOTE: With an OS you assume mem. is fixed, but in a VM is no longer the case. Only part of it is in the physical memory, and the amount given to it might change overtime. May be difficult for the host to manage this effectively.

Virtualization of memory management

Each guest OS keeps its page tables for memory addresses translation
- Of course, each of them has only a partial view of the memory

The Hypervisor is the only one that keeps the real page table, how does it keep this all consistent?

The solution is based on the concept of **nested page tables** (**NPT**)
- Each guest maintains its own page tables
- The Hypervisor maintains NTPs tables for each guest (like what it does for the VCPU)
- When a guest OS is running on a CPU the Hypervisor makes active the corresponding NTPs
- When the guest OS tries to change page table the Hypervisor makes changes the NTPs and its own pages accordingly

It works, although with penalty in the system performances

Modern process provide support to NTP at hardware

63

Proposals: pseudo-always;

Impact of virtualization on underlying OS: memory management

Overcommitment also for memory management
- Virtual machines allocate much more memory than the physical one
- Each virtual machine has its guest OS with its own memory manager
- Need for special measures in the virtualization in order to keep good performances

The hypervisor:
- Determines the “declared” need of memory of each virtual machine
  - The guest OS on the virtual machine do not expect this memory to change dynamically
- Estimates the current need of memory
- Allocates the physical memory dynamically, taking into account the overall requirements of all virtual machines

64

*1 Guest OS believes memory is fixed, but it can be reduced or increased by Host. So GOS does not know this. With balloon, driver coordinated with the Hos, allocates memory for the GOS in a way to match the availability of the physical memory

Impact of virtualization on underlying OS: memory management

Several methods can be used by the host kernel to manage memory:

- Double paging:
  - the host kernel keeps is page table and gives virtual memory to the guest OS.
  - In turn, the guest OS manage paging over this virtual memory
- Pseudo-driver (AKA "balloon"): *1
  - in the guest OS it is possible to install an additional driver to run over the virtual machine.
  - This driver is just a way change the behavior of the guest host without changing its kernel
  - This driver communicates with the host kernel to coordinate the memory management with that of the guest OS
  - May pin pages in memory of the guest host to force it assume it has less amount of free memory, and thus reduce its needs…
  - In practice, pinned pages in the guest OS memory will not be used by the balloon, hence the host kernel can move them to the disk and free up the physical memory
- The host kernel may check the physical memory to look for duplicated pages and make them shared among different virtual machines
  - Since many virtual machine will most likely have the same guest OS, many pages will have the same exact content
  - Fast check to look for duplicates: compare hash of pages and then, if match, compare byte per byte

65

For I/O also other problems: when there's an I/O op. you may require the Hyp. to do it, but inefficient. Most solutions now can make the GOS work with HW itself. So the HW would need to be able to launch interrupt to guests and host.

Impact of virtualization on underliyng OS: I/O management

Can provide specific device drivers for the virtual machine to install in guest OS

- This simplify the management in guest OS

The complexity is in the hypervisor:

- At worst the hypervisor may execute all I/O on behalf of guest OS
- But this is inefficient, better to give direct access to guest OS (DMA, direct interrupt delivery,…)
- Need for extra HW support for this

Also networking is complex as a single network access need to be multiplexed for several virtual machines

- May provide independent IP address to the guest
- Or use a NAT (network address translation, see computer networking class)

66

Impact of virtualization on underliyng OS: I/O management

Each virtual machine need to access its own boot partition

But host kernel cannot implement a standard partition on disk for each virtual machine… two options:

- Allocate a disk image (a standard file in the file system) as boot disk to each virtual machine (Hypervisors Type 2 preferred solution)
- Use the native file system used by hypervisor (Hypervisors type 1 normally use this solution)

In any case:

- Easy to duplicate a guest and to move it on another physical machine
- To create a new guest may copy a physical installation (physical to virtual P-to-V): convert native disk blocks into VMM format
- Also possible Virtual-to-physical (V-to-P) convert from virtual format to native or disk format

67

how can they be managed? A VM will run over a physical machine. Using a VM you can move it w/o port. problems, you can update the HW too, so scaling up.

MV is running and we don't want to freeze it.

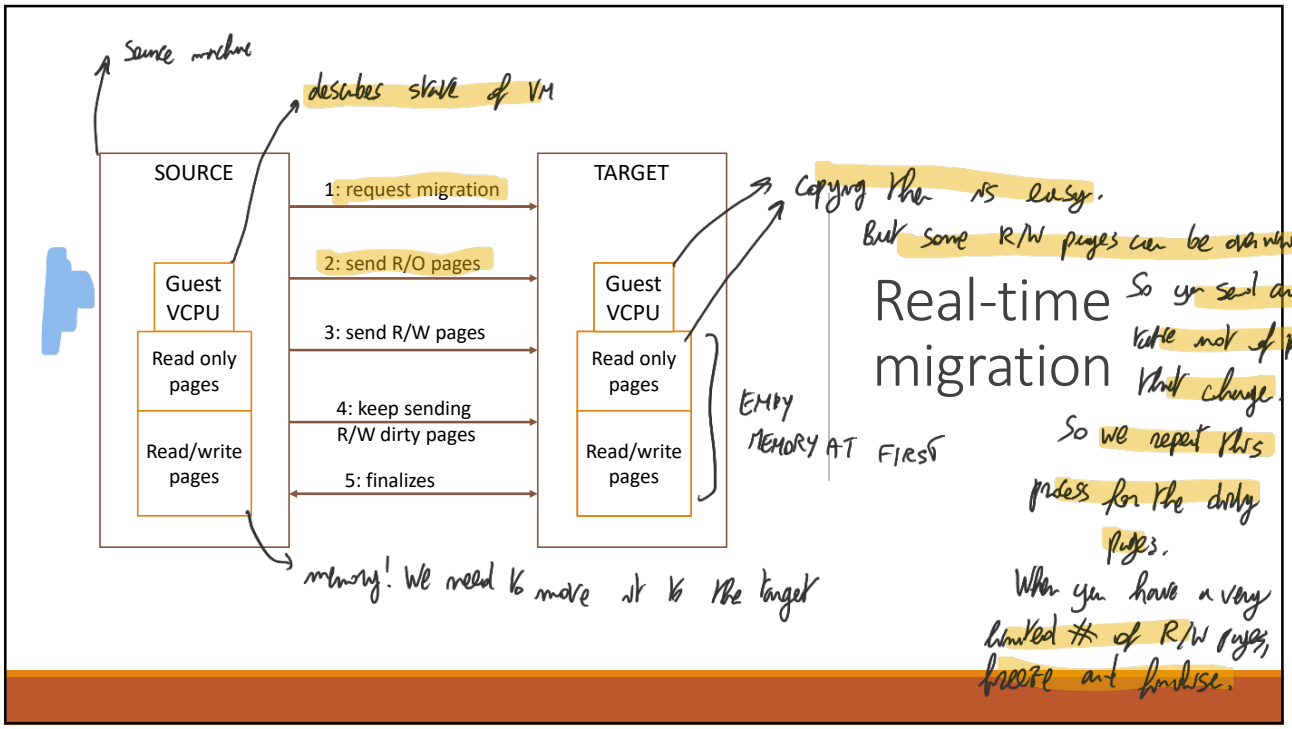
Real time Migration

An advantage of OS installed on virtual machines that cannot be offered by OS on a physical machine

An entire guest can be duplicated on another physical machine or on another virtual machine

- This without interrupting the guest OS and its applications

68



69

Real time Migration

How it works:

1. The source virtual machine (SOURCE) request the migration to a target virtual machine (TARGET)
   - TARGET creates a new guest: allocates the VCPU, memory, disk, I/O etc.
2. SOURCE sends to TARGET all read-only guest memory pages
3. SOURCE sends to TARGET all read-write pages, marking them as clean
4. Step 3 is repeated, because during its execution some pages of the guest may have changed (are now dirty)
   - Afterall, the guest is still running…
5. When only few pages are dirty in steps 4 and 5 (below a threshold):
   - SOURCE freezes guest, sends VCPU's state and any other state details to TARGET
   - SOURCE sends last dirty pages
   - TARGET starts running the guest
   - SOURCE can now kill guest

70

What kind of HW uses the trap and emulate mechanism?

Type 1 is specialised to implement virtualization so it is done to support virtualization. So you have work that is exceptionally optimised and we avoid it. Guest mode/Host mode.

Other virtualization methods

71

(1) Create a OS that can run over VM.

Other virtualization methods

- **Paravirtualization:** not a real virt.
  - The guest OS is modified to be aware of the virtual machine
  - Simplifies the management and may improve performance
  - Not transparent virtualization is a weakness
- **Language-level virtual machines:**
  - The virtual machine implements a virtual environment
  - It does not reproduced a specific hardware
  - Portability is a strength
  - Examples are: Java and .Net

Operated by language: interpreted languages.

- **Emulation:** Piece of SW that emulates the HW. Emulators. Not only for games. Test an embedded sys/for ex
  - The hardware is fully emulated at software
  - Often uses to cross-platform compilation and testing
  - May implement a different CPU
- **Containers:** virtualiz. for application, in the context of apps
  - Segregates applications from the OS
  - Implemented by a software layer over the OS
  - Not a proper virtualization, although has some features and objectives in common
  - Widely used in cloud systems
  - Advantages are security and simplified management of applications. They are also lightweight.
  - Examples: Docker, Oracle Solaris Zones, BSD Jails, IBM AIX WPARs

as lighter of course.

72

Security: VM is under the control of HV (or real time suppos of containers). If you loose control of HV you have complete powers.

Paravirtualization

Not a proper virtualization as the virtual machine implemented is not identical to the physical host

- Needs less (if any) HW support
- Requires the guest OS be aware of the virtual machine
- Generally results in a slight better performances
- Nowadays no longer used

An example is Xen (from Cambridge University)

- Initially motivated by the difficulties of virtualizing the X86 Hardware (see the case of the `popf` instruction)
- Does not use any HW support
- Simple and efficient I/O abstraction
- Windows XP had been ported over Xen, although with strong support of Microsoft

73

Programming Environment Virtualization

Not a proper virtualization

Programming language designed to run within a custom virtual machine

- The most popular example is given by Java by SUN (now Oracle)
- Java code compiled into bytecodes and executed in a Java Virtual Machine (JVM)
  - Performs better than pure interpretation as bytecode is an intermediate language easy to interpret and execute
- Later other languages followed this approach

Virtualization in terms of API to access the OS services

Full portability over different OS and hardware platform

74

Emulation

Emulation provides an interpreter for a brand new CPU
- … while virtualization requires the virtualized CPU be the same as the physical CPU
- Can run over other, non compatible CPUs

Interpretation of instruction of guest OS:
- static – instructions translated before execution
- dynamic - instructions translated one by one

Uses:
- to test OS and software for embedded systems and, in general, in the cases where cross compilation is used
- To run software of old, obsolete HW
  - E.g. console games, but even business applications

It's a very slow method, but new machines are much faster than old ones

75

Containers

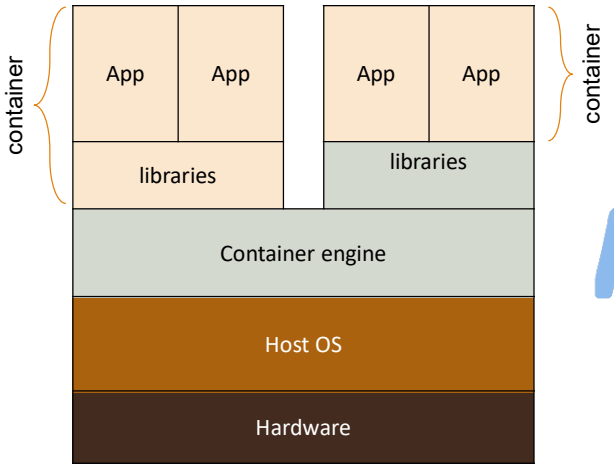
A recent approach to virtualization is known as *container virtualization* or *application virtualization*

Motivated by the need of giving the same advantages of virtualization to a set of applications:
- Segregation of apps for security
- Simplified management of applications (restart, migration, checkpointing, cloning etc…)
- Sharing of resources among different applications

If these applications are indeed written for the same OS: HW virtualization unnecessary, but a higher level of virtualization can be used.

In this approach, software known as a *virtualization container,* runs on top of the host OS kernel and provides an isolated execution environment for applications
- It's the OS that is virtualized this time
- Only the host kernel in the system
- Each container provides a virtualized OS and devices to its applications

76



Container (application virtualization)

Containers

77

Containers

Unlike hypervisor-based VMs, containers do not aim to emulate physical servers

All containerized applications on a host share a common OS kernel
- Built on top of an OS, run as a process, enforces portability

For containers, only a small container engine is required as support for the containers
- Containers are lightweight and efficient

Examples: Docker, Oracle Solaris Zones, BSD Jails, IBM AIX WPARs

Containerization sits in between the OS and applications and incurs lower overhead, but potentially introduces greater security vulnerabilities

78

Software Defined Networks (SDNs)

SDNs enable network segments to logically span multiple servers within and between data centers, while using the same underlying physical network

There are several possible approaches to providing SDNs, including the use of overlay networks

- These abstract all layer 2 and 3 addresses from the underlying physical network into whatever logical network structure is required
- This structure can be easily changed and extended as needed
- The IETF standard DOVE (Distributed Overlay Virtual Network) which uses VXLAN (Virtual Extended Local Area Network) can be used to implement such an overlay network
- With this flexible structure, it is possible to locate virtual servers, virtual IDS, and virtual firewalls anywhere within the network as required

79

VM Security

80

Concerns: ① ensure isolation of GOS. They don't have to mess up with other GOS. Enforced by HV or HW support.
② HV monitors the GOS. So HV needs to be trusted.
③ Concern related to the VE: VM can be stopped, migrated, restarted. We may intercept transfer of VM, or access a frozen VM somewhere on disk ex. For this, we should take into account this. Encryption for freeze and migration.

Virtualization Security Issues

- Guest OS isolation
  - Ensuring that programs executing within a guest OS may only access and use the resources allocated to it
- Guest OS monitoring by the hypervisor
  - Which has privileged access to the programs and data in each guest OS
- Virtualized environment security
  - Particularly image and snapshot management which attackers may attempt to view or modify

81

Same vulnerabilities in an OS one there in a VM. If there is a compromise in the VM, attacker might try to attack other VM too, NOTE: if you have a compromise of a VM, the connection between two machines, we have communication on a Virtual Network in the same HW, NISD don't work. But the HV has control to! You could have Virtual Firewalls to control how they communicate as HIDS too specific.

Virtualization Security Issues

These are similar to those of operating systems and applications:

- If an OS or an app is vulnerable when running directly on hardware in some context, it will most likely also be vulnerable when running in a virtualized environment.
- A compromised system may attack other nearby systems
  - whether they are running on hardware or as guests in a virtualized machine.

Virtualized environments may improve security:

- Isolate network traffic between guests more than would be the case otherwise
- But this traffic is not visible to external IDS or firewall systems, and may require the use of virtual firewalls to manage.

82

Virtualization Security Issues

The hypervisor may act as virtual firewall due to its ability to monitor deeply the guest OSs

However, vulnerabilities in the hypervisor itself may severely reduce security

- attackers may exploit these vulnerabilities to take over a number of guest OSs
- This is known as VM escape

Virtualized systems also often provide support for suspending an executing guest OS in a snapshot, saving that image, and then restarting execution at a later time, possibly even on another system.

- Attacks can be directed to modify this image, to compromise the security of the data and programs contained within it.
- The use of infrastructure with many virtualized systems within and between data centers, linked using software-defined networks, raise further security concerns

83

Virtualization Security Issues

Securing virtualized systems means extending the security process to:

- Secure and harden guest OSs.
- Secure and harden the virtualized environment
- Secure and harden the Hypervisor

There are standards, NIST provides one.

84

Securing Virtualization Systems

Organizations using virtualization should:

- Carefully plan the security of the virtualized system
- Secure all elements of their virtualization solution and maintain their security
- Ensure that the hypervisor is properly secured
- Restrict and protect administrator access to the virtualization solution

85

Hypervisor Security

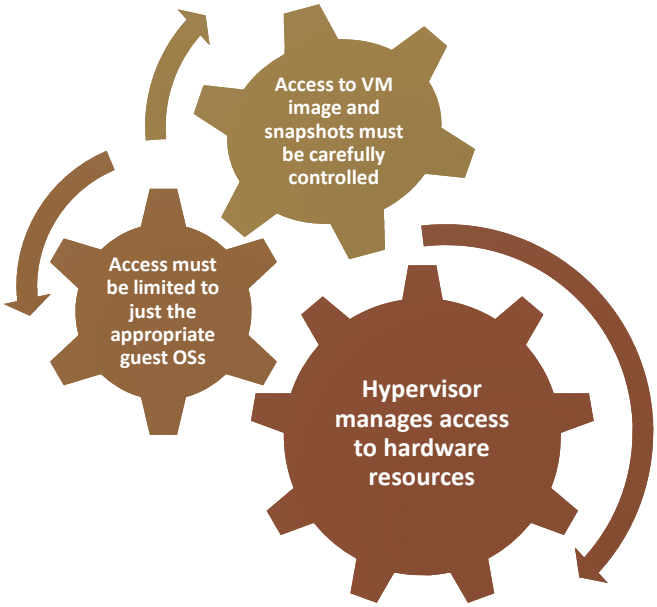
- Should be: considered as an OS so must be hardened. Type 1 usually comes from hardened features, and access control for the HV features.
  - Secured using a process similar to securing an operating system
  - Installed in an isolated environment
  - Configured so that it is updated automatically
  - Monitored for any signs of compromise
  - Accessed only by authorized administration

VM will create a lot of traffic over the network.

- May support both local and remote administration so must be configured appropriately
- Remote administration access should be considered and secured in the design of any network firewall and IDS capability in use
- Ideally administration traffic should use a separate network with very limited access provided from outside the organization

86

Traffic of the management of the HV itself, but also for the management of the VM. Related for ex to migration or save. Often VMs talking to each other or the ext. world. These traffics are to be protected. Need a mngmt to isolate these traffic.



Virtualized Infrastructure Security

87

Virtualized Infrastructure Security

When multiple virtualized systems are used, NIST SP 800-125B (*Secure Virtual Network Configuration for Virtual Machine (VM) Protection*, March 2016) notes three distinct categories of network traffic:

- **Management traffic**: used for hypervisor administration and configuration of the virtualized infrastructure.
- **Infrastructure traffic**: such as migration of VM images, or connections to network storage technologies.
- **Application traffic**: between applications running VMs and to external networks.

This traffic may be further separated into a number of segments, isolating traffic from applications with different sensitivity levels, or from different organizations or departments.

88

Virtual Firewall

Provides firewall capabilities for the network traffic flowing between guest hosts

- this traffic is not routed out to a physically separate network supporting traditional firewall services

Three ways to achieve this:

- VM Bastion Host
  - Where a separate VM is used as a bastion host supporting the same firewall systems and services that could be configured to run on a physically separate bastion, including possibly IDS and IPS services
- VM Host-Based Firewall
  - Where host-based firewall capabilities provided by the Guest OS running on the VM are configured to secure that host in the same manner as used in physically separate systems
- Hypervisor Firewall
  - Where firewall capabilities are provided directly by the hypervisor

Security measures are the same: Sensor on the host, Virtual firewall, one VM could be a bastion for security policies, sensors, analyzers etc.

89

Question

Consider a host on which are running 2 guest OS.

Let's assume an external storage device is connected to the USB port of the host.

1. If the two guest OS need to access the external storage, the access will be granted to both of them at the same time?

external storage must be associated to one or

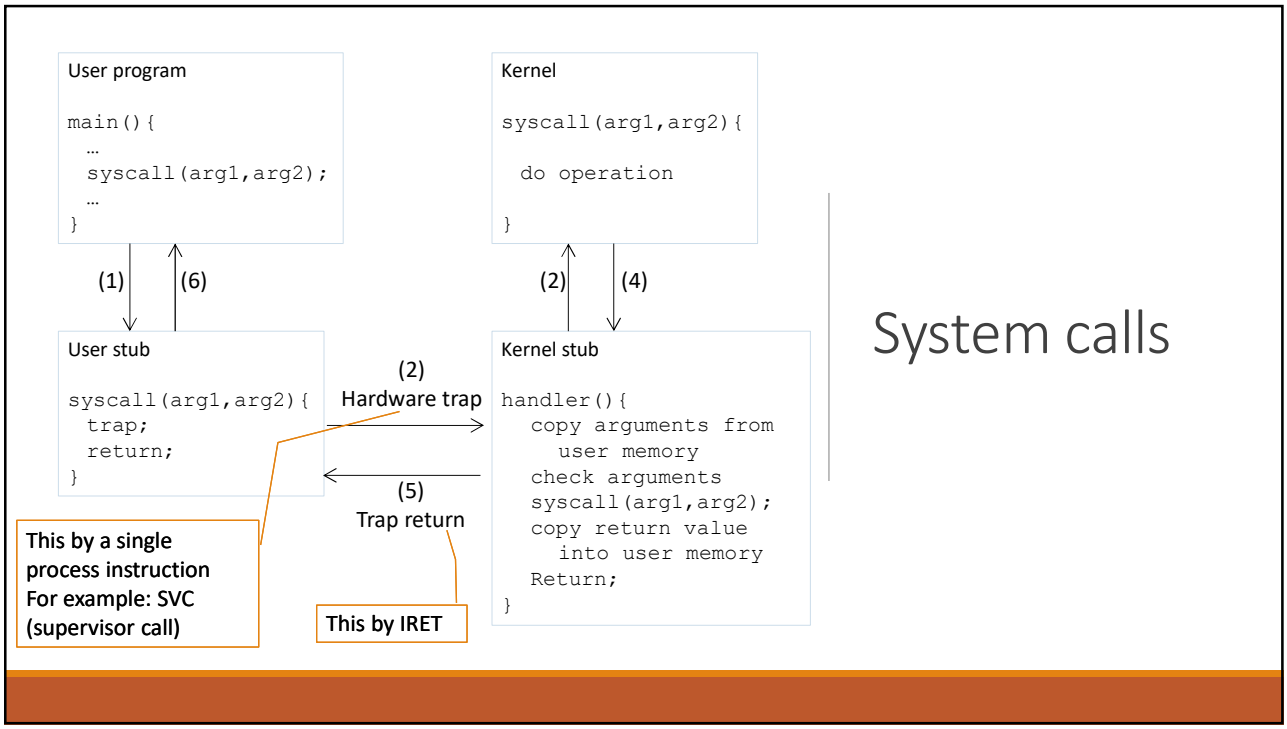
2. What about the network card installed in the host. Can it be allocated at both guest at the same time?

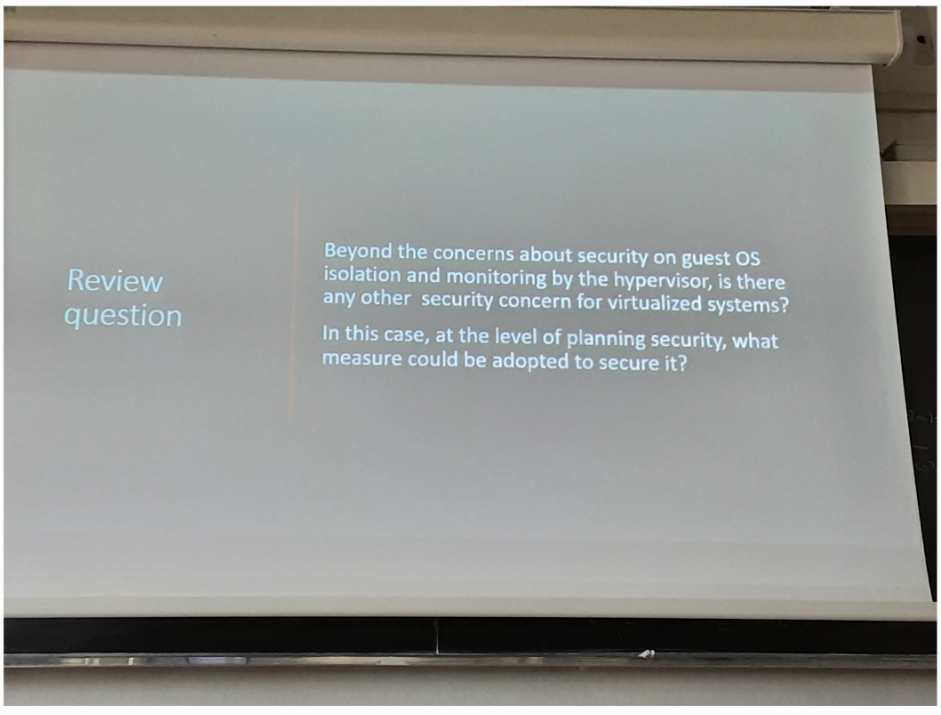
another

↓ methods as for ex. to implement a NAT,

90

System calls





- Secure GOS, Hypervisor, and environment for the management of virtualization. Ex. securing the backup or migration of VM.