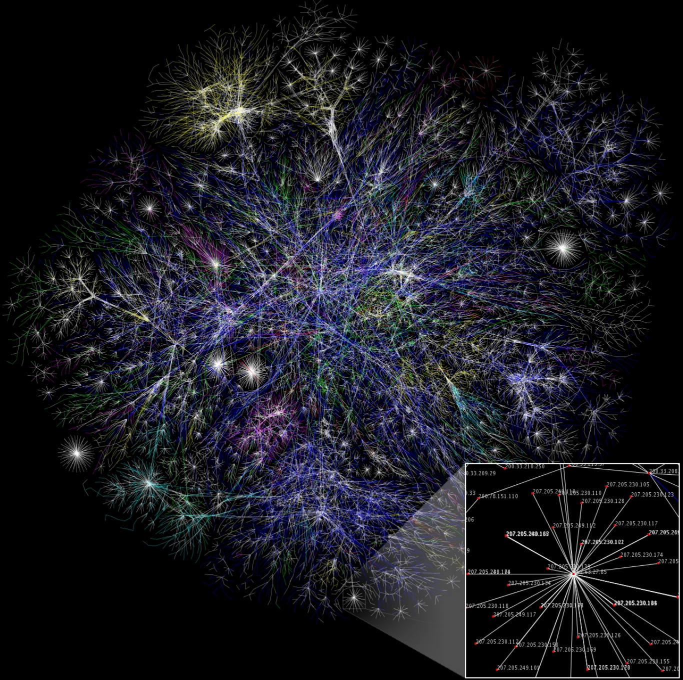
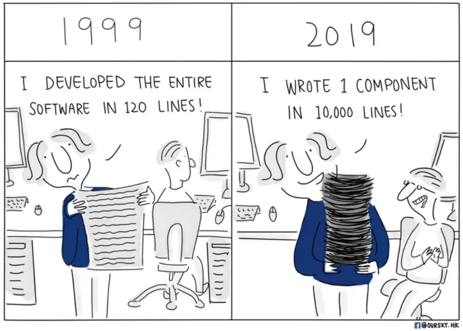
Cloud Operating Systems

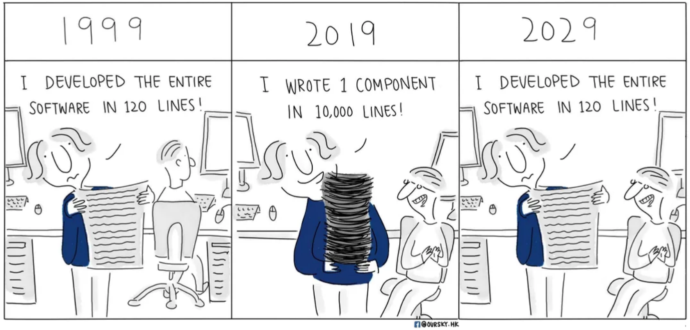
Daniel Gruss

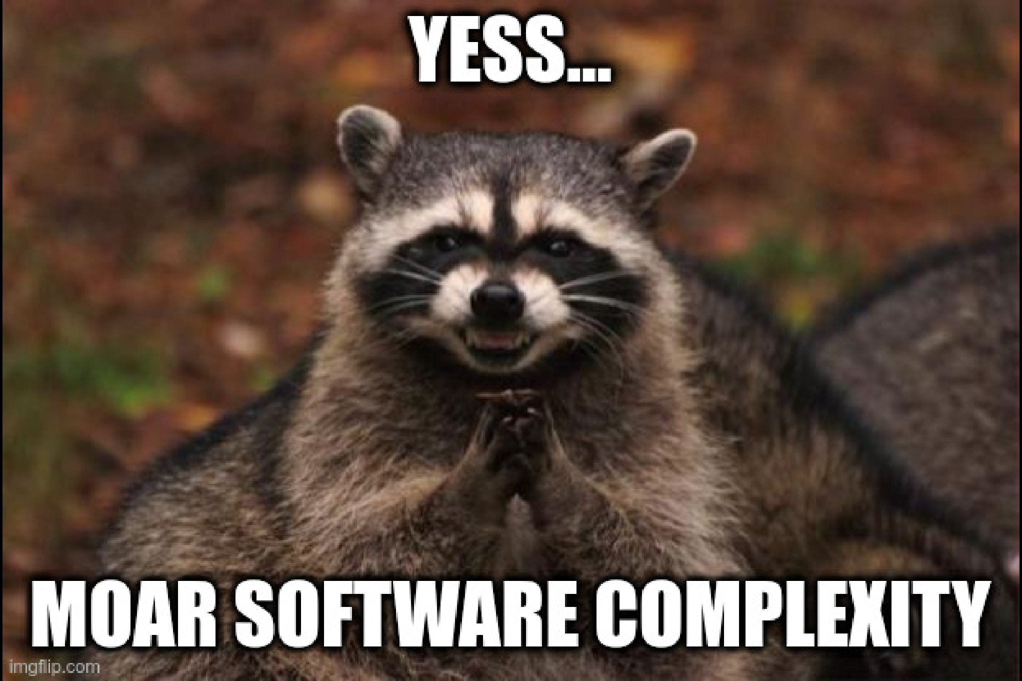
2023-03-01



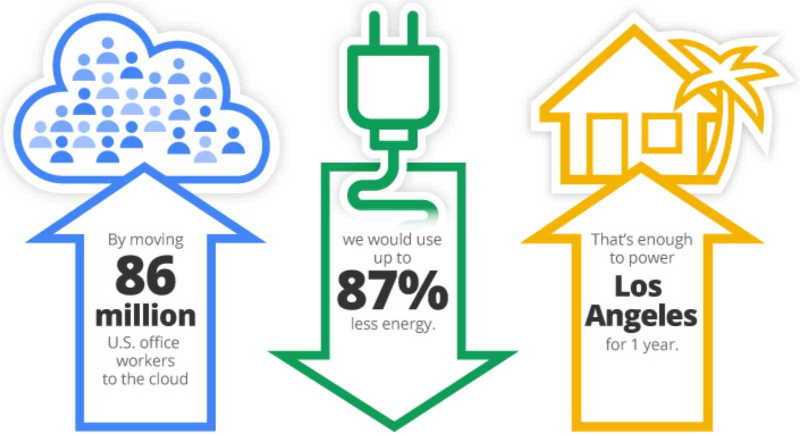
200.33.210.250
200.33.209.29
200.33.208
200.78.151.110
207.205.230.105
207.205.230.110
207.205.230.128
207.205.230.123
207.205.249.112
207.205.230.117
207.205.230.174
63.163.27.85
207.205.230.134
207.205.230.118
207.205.249.117
207.205.230.112
207.205.230.153
207.205.230.169
207.205.230.126
207.205.230.155
207.205.249.101


1999
2019
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
I WROTE 1 COMPONENT IN 10,000 LINES!
@OURSKY.HK


1999
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
2019
I WROTE 1 COMPONENT IN 10,000 LINES!
2029
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
@OURSKY.HK


YESS...
MOAR SOFTWARE COMPLEXITY
imgflip.com

Moving to the cloud can save up to 87% of IT energy



Cloud means Efficiency

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory $\rightarrow$ that's not efficient!

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory $\rightarrow$ that's not efficient!

$\rightarrow$ Virtualize memory $\rightarrow$ processes can share resources of one machine and utilize it better

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory $\rightarrow$ that's not efficient!

$\rightarrow$ Virtualize memory $\rightarrow$ processes can share resources of one machine and utilize it better

- Processes need all the same pages $\rightarrow$ that's not efficient!

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory $\rightarrow$ that's not efficient!

$\rightarrow$ Virtualize memory $\rightarrow$ processes can share resources of one machine and utilize it better

- Processes need all the same pages $\rightarrow$ that's not efficient!

$\rightarrow$ Let them share memory, using COW, page deduplication, etc.

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory → that's not efficient!

→ Virtualize memory → processes can share resources of one machine and utilize it better

- Processes need all the same pages → that's not efficient!

→ Let them share memory, using COW, page deduplication, etc.

- Processes often cannot do anything but wait → that's not efficient!

How does the cloud bring efficiency?

Apply OS techniques - Example?

- Processes used to have access to all physical memory → that's not efficient!

→ Virtualize memory → processes can share resources of one machine and utilize it better

- Processes need all the same pages → that's not efficient!

→ Let them share memory, using COW, page deduplication, etc.

- Processes often cannot do anything but wait → that's not efficient!

→ Let other processes run in between

IaaS
PaaS
SaaS

Different techniques - similar challenges

- Efficiency

Different techniques - similar challenges

- Efficiency
- Isolation of tenants (security, reliability, availability)

Different techniques - similar challenges

- Efficiency
- Isolation of tenants (security, reliability, availability)
- Abstraction of hardware

What is Virtualization?

Virtualization allows to represent resources in a computer in a way they can be used easily and without the need to know details of their properties

Virtual Machines (VM)

- Decouple operating system from hardware

Virtual Machines (VM)

- Decouple operating system from hardware
    - “computer in computer” - implemented in software

Virtual Machines (VM)

- Decouple operating system from hardware
    - “computer in computer” - implemented in software
    - includes devices (network, keyboard, sound...)

Virtual Machines (VM)

- Decouple operating system from hardware
  - “computer in computer” - implemented in software
  - includes devices (network, keyboard, sound...)
- OS in VM “sees” its hardware, irrespective from the actual hardware in use

Virtual Machines (VM)

- Decouple operating system from hardware
  - “computer in computer” - implemented in software
  - includes devices (network, keyboard, sound...)
- OS in VM “sees” its hardware, irrespective from the actual hardware in use
- OS does not know if HW is concurrently used by other VMS

Why virtualization

Why virtualization

Why virtualization

Why virtualization

- Cheaper hardware: one server for one task was common

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance
  - power consumption (operation, cooling)

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance
  - power consumption (operation, cooling)
  - space

Why virtualization

- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance
  - power consumption (operation, cooling)
  - space
- Virtualization allows consolidation

Why virtualization

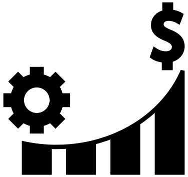
- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance
  - power consumption (operation, cooling)
  - space
- Virtualization allows consolidation
  - multiple servers on one box

Advantages

Advantages

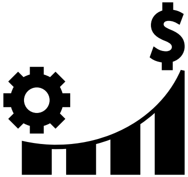
Advantages

Advantages



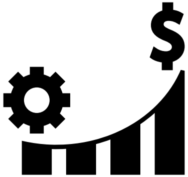
- Better hardware utilization

Advantages



- Better hardware utilization
- Lower administration cost

Advantages



- Better hardware utilization
- Lower administration cost
- long-term operations of older applications

Advantages

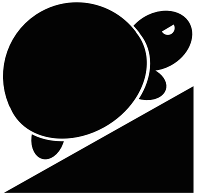
- Better hardware utilization
- Lower administration cost
- long-term operations of older applications
- lower down-times

Advantages

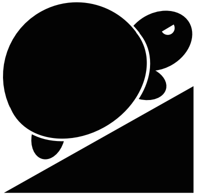
- Better hardware utilization
- Lower administration cost
- long-term operations of older applications
- lower down-times
- simple migration to more powerful hardware

Disadvantages

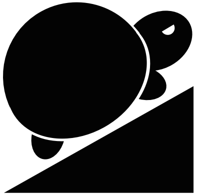
Disadvantages



Disadvantages

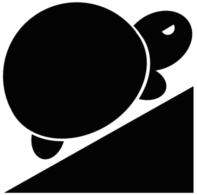


Disadvantages



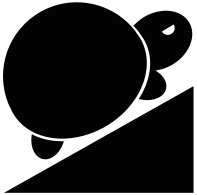
- Performance cost: slower I/O operation

Disadvantages



- Performance cost: slower I/O operation
- single point of failure: requires better hardware reliability

Disadvantages



- Performance cost: slower I/O operation
- single point of failure: requires better hardware reliability
- security gets more complex

Virtualization: Trend 1990s → 2000s

- Virtualization no significant role in internet hosting

Virtualization: Trend 1990s → 2000s

- Virtualization no significant role in internet hosting
- often PaaS

Virtualization: Trend 1990s $\rightarrow$ 2000s

- Virtualization no significant role in internet hosting
- often PaaS
- Web hosts (FTP access, HTTP website)

Virtualization: Trend 1990s $\rightarrow$ 2000s

- Virtualization no significant role in internet hosting
- often PaaS
- Web hosts (FTP access, HTTP website)
- Isolation on the OS level (tenants as users)

Virtualization: Trend 1990s $\rightarrow$ 2000s

- Virtualization no significant role in internet hosting
- often PaaS
- Web hosts (FTP access, HTTP website)
- Isolation on the OS level (tenants as users)
- no hardware support $\rightarrow$ expensive $+$ many problems

Modern Virtualization

- OS-level Virtualization

Modern Virtualization

- OS-level Virtualization
- Para-Virtualization

Modern Virtualization

- OS-level Virtualization
- Para-Virtualization
- Full Virtualization

Modern Virtualization

- OS-level Virtualization
- Para-Virtualization
- Full Virtualization
- Hardware-Assisted Virtualization

OS-level Virtualization

- integrated into kernel

OS-level Virtualization

- integrated into kernel
- all application software intended to run in a virtual environment get strictly separated virtual runtime environments (container, jail)

OS-level Virtualization

- integrated into kernel
- all application software intended to run in a virtual environment get strictly separated virtual runtime environments (container, jail)
- no separate kernels - only process level virtualization

OS-level Virtualization

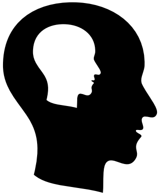
- integrated into kernel
- all application software intended to run in a virtual environment get strictly separated virtual runtime environments (container, jail)
- no separate kernels - only process level virtualization
- can't run other OSes - only for applications

OS-level Virtualization

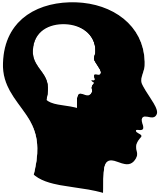
- integrated into kernel
- all application software intended to run in a virtual environment get strictly separated virtual runtime environments (container, jail)
- no separate kernels - only process level virtualization
- can't run other OSes - only for applications
- examples: OpenVZ, Docker, (s)chroot

Para-Virtualization

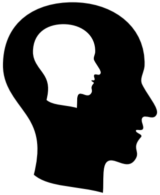
Para-Virtualization



Para-Virtualization

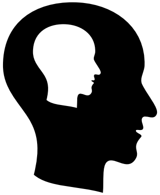


Para-Virtualization



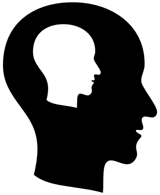
- Cooperation with OS: OS is aware of virtualization

Para-Virtualization



- Cooperation with OS: OS is aware of virtualization
- needs to modify guest

Para-Virtualization



- Cooperation with OS: OS is aware of virtualization
- needs to modify guest
- not usable for closed source OSes

Full Virtualization

- OS not aware of being virtualized

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest
- reduced performance

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest
- reduced performance
    - up to 25%

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest
- reduced performance
  - up to 25%
- full virtualization of HW required (e.g., emulation via qemu)

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest
- reduced performance
  - up to 25%
- full virtualization of HW required (e.g., emulation via qemu)
  - virtual machines not allowed to access physical components

Full Virtualization

- OS not aware of being virtualized
- no need to adapt guest
- reduced performance
  - up to 25%
- full virtualization of HW required (e.g., emulation via qemu)
  - virtual machines not allowed to access physical components
  - every physical component has to be virtualized and requires drivers in OS

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)
    - parts that require kernel privileges won't run

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)
    - parts that require kernel privileges won't run
- hypervisor (VMM) changes binaries of guest-OS on the fly

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)
    - parts that require kernel privileges won't run
- hypervisor (VMM) changes binaries of guest-OS on the fly
- allows supporting any OS

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)
    - parts that require kernel privileges won't run
- hypervisor (VMM) changes binaries of guest-OS on the fly
- allows supporting any OS
    - no need to change source

Some earlier systems: binary translation

- Guest no longer runs in kernel mode (Ring 0)
    - parts that require kernel privileges won't run
- hypervisor (VMM) changes binaries of guest-OS on the fly
- allows supporting any OS
    - no need to change source
- high performance penalty

VMWare

- First full x86 virtualization

VMWare

- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (pre-scan)

VMWare

- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (pre-scan)
- looking for relevant commands

VMWare

- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (pre-scan)
- looking for relevant commands
    - change of system state

VMWare

- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (pre-scan)
- looking for relevant commands
    - change of system state
    - commands depending on CPU state

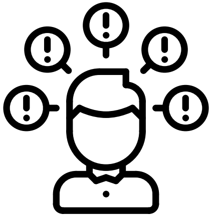
VMWare

- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (pre-scan)
- looking for relevant commands
  - change of system state
  - commands depending on CPU state
- sets breakpoint and lets OS run

Problems virtualizing on IA-32

Problems virtualizing on IA-32

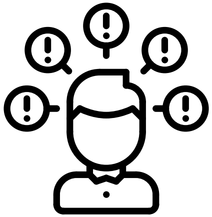
Problems virtualizing on IA-32



Problems virtualizing on IA-32

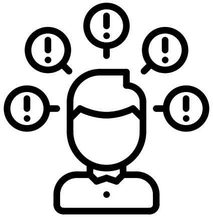
- Diverse problems were to be solved when virtualizing on IA-32:

Problems virtualizing on IA-32



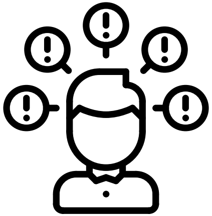
- Diverse problems were to be solved when virtualizing on IA-32:
    - Ring Problems

Problems virtualizing on IA-32



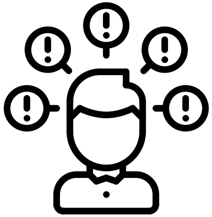
- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems
  - Address Space Compression

Problems virtualizing on IA-32



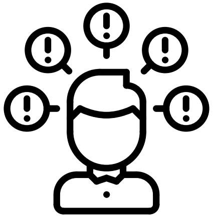
- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems
  - Address Space Compression
  - Non-Faulting Access to Priv. State

Problems virtualizing on IA-32



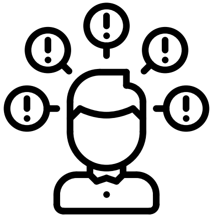
- Diverse problems were to be solved when virtualizing on IA-32:
    - Ring Problems
    - Address Space Compression
    - Non-Faulting Access to Priv. State
    - SYSENTER / SYSEXIT

Problems virtualizing on IA-32



- Diverse problems were to be solved when virtualizing on IA-32:
    - Ring Problems
    - Address Space Compression
    - Non-Faulting Access to Priv. State
    - SYSENTER / SYSEXIT
    - Interrupt Virtualization

Problems virtualizing on IA-32



- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems
  - Address Space Compression
  - Non-Faulting Access to Priv. State
  - SYSENTER / SYSEXIT
  - Interrupt Virtualization
  - Hidden States

Ring Problem 1: which ring

- usually: application run in ring 3, kernel in ring 0

Ring Problem 1: which ring

- usually: application run in ring 3, kernel in ring 0
- guest may not run in ring 0

Ring Problem 1: which ring

- usually: application run in ring 3, kernel in ring 0
- guest may not run in ring 0
- ring de-privileging needed: guest must run in ring > 0

Ring Problem 1: which ring

- usually: application run in ring 3, kernel in ring 0
- guest may not run in ring 0
- ring de-privileging needed: guest must run in ring > 0
    - most often 1 or 3

Ring Problem 2: aliasing

- guest has to run in a ring it has not been developed for

Ring Problem 2: aliasing

- guest has to run in a ring it has not been developed for
- certain instructions contain privilege level in result (e.g. `PUSH CS`)

Ring Problem 2: aliasing

- guest has to run in a ring it has not been developed for
- certain instructions contain privilege level in result (e.g. `PUSH CS`)
- guest OS can find out ring it is running in

Ring Problem 2: aliasing

- guest has to run in a ring it has not been developed for
- certain instructions contain privilege level in result (e.g. `PUSH CS`)
- guest OS can find out ring it is running in
- may result in diverse problems

Address Space Compression

- Guest expects to have full address space available

Address Space Compression

- Guest expects to have full address space available
- hypervisor requires part of address space

Address Space Compression

- Guest expects to have full address space available
- hypervisor requires part of address space
    - control structures for switching between guest and hypervisor

Address Space Compression

- Guest expects to have full address space available
- hypervisor requires part of address space
    - control structures for switching between guest and hypervisor
- Access to these areas not allowed for guest. Invokes switch to hypervisor who has to emulate these accesses

Non-Faulting Access to Priv. State

- unprivileged software may not access certain elements of the CPU state

Non-Faulting Access to Priv. State

- unprivileged software may not access certain elements of the CPU state
- access by guest results in fault: hypervisor can emulate instructions

Non-Faulting Access to Priv. State

- unprivileged software may not access certain elements of the CPU state
- access by guest results in fault: hypervisor can emulate instructions
- IA-32 possesses instructions that do not induce a fault:

Non-Faulting Access to Priv. State

- unprivileged software may not access certain elements of the CPU state
- access by guest results in fault: hypervisor can emulate instructions
- IA-32 possesses instructions that do not induce a fault:
    - Registers GDTR, IDTR, LDTR and TR are only modifiable in ring 0

Non-Faulting Access to Priv. State

- unprivileged software may not access certain elements of the CPU state
- access by guest results in fault: hypervisor can emulate instructions
- IA-32 possesses instructions that do not induce a fault:
    - Registers GDTR, IDTR, LDTR and TR are only modifiable in ring 0
    - can be executed in any ring without fault (without function)

SYSENTER / SYSEXIT

- special commands for fast syscalls

SYSENTER / SYSEXIT

- special commands for fast syscalls
- SYSENTER always switches to ring 0

SYSENTER / SYSEXIT

- special commands for fast syscalls
- SYSENTER always switches to ring 0
- SYSEXIT can only be executed in ring 0

SYSENTER / SYSEXIT

- special commands for fast syscalls
- SYSENTER always switches to ring 0
- SYSEXIT can only be executed in ring 0
- ring 1 thus is problematic

SYSENTER / SYSEXIT

- special commands for fast syscalls
- SYSENTER always switches to ring 0
- SYSEXIT can only be executed in ring 0
- ring 1 thus is problematic
    - SYSENTER switches to hypervisor $\rightarrow$ has to emulate

SYSENTER / SYSEXIT

- special commands for fast syscalls
- SYSENTER always switches to ring 0
- SYSEXIT can only be executed in ring 0
- ring 1 thus is problematic
    - SYSENTER switches to hypervisor → has to emulate
    - SYSEXIT switches to hypervisor → has to emulate

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)
- controlled by IF-flag in EFLAGS-Register

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)
- controlled by IF-flag in EFLAGS-Register
- Interrupts managed by VM though

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)
- controlled by IF-flag in EFLAGS-Register
- Interrupts managed by VM though
- change of IF $\rightarrow$ fault to hypervisor

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)
- controlled by IF-flag in EFLAGS-Register
- Interrupts managed by VM though
- change of IF $\rightarrow$ fault to hypervisor
- OS do this quite often $\rightarrow$ performance problem

Interrupt Virtualization

- interrupts can be masked (so they do not occur if not welcome)
- controlled by IF-flag in EFLAGS-Register
- Interrupts managed by VM though
- change of IF $\rightarrow$ fault to hypervisor
- OS do this quite often $\rightarrow$ performance problem
- forwarding of virtual interrupts must consider IF

hidden state information

hidden state information

hidden state information

hidden state information

- Not all state-information accessible via registers

hidden state information

- Not all state-information accessible via registers
- cannot be saved and restored when switching between VMs

Intel Virtualization Architecture

- Two new operating modes:

Intel Virtualization Architecture

- Two new operating modes:
  - VMX root operation

Intel Virtualization Architecture

- Two new operating modes:
  - VMX root operation
    - for hypervisor

Intel Virtualization Architecture

- Two new operating modes:
  - VMX root operation
    - for hypervisor
  - VMX non-root operation

Intel Virtualization Architecture

- Two new operating modes:
    - VMX root operation
        - for hypervisor
    - VMX non-root operation
        - controlled by hypervisor

Intel Virtualization Architecture

- Two new operating modes:
    - VMX root operation
        - for hypervisor
    - VMX non-root operation
        - controlled by hypervisor
        - supports VMs

Intel Virtualization Architecture

- Two new operating modes:
    - VMX root operation
        - for hypervisor
    - VMX non-root operation
        - controlled by hypervisor
        - supports VMs
- Both modes have ring 0-3

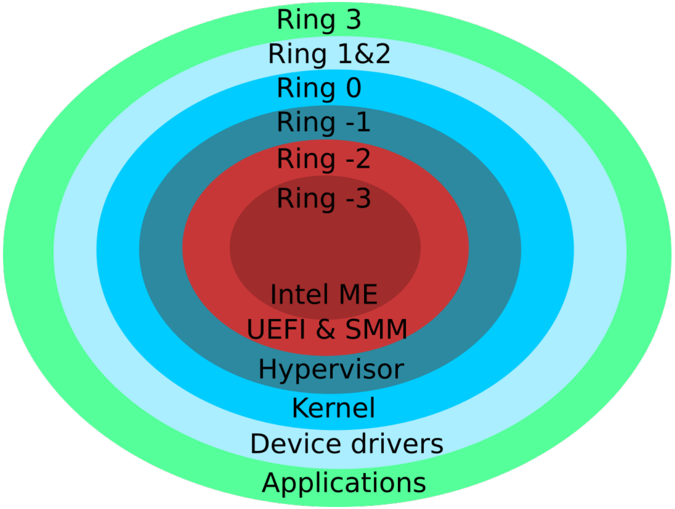
Intel Virtualization Architecture

- Two new operating modes:
  - VMX root operation
    - for hypervisor
  - VMX non-root operation
    - controlled by hypervisor
    - supports VMs
- Both modes have ring 0-3
- guest can run in ring 0

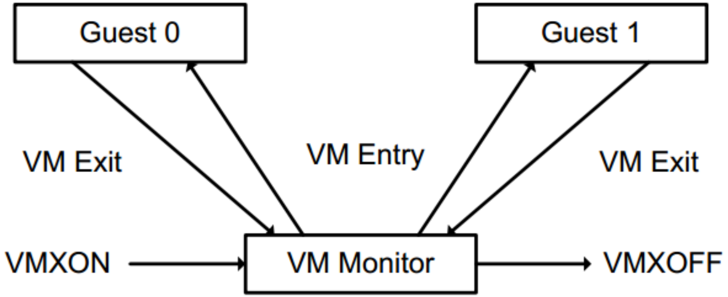
Intel Virtualization Architecture

- Two new operating modes:
  - VMX root operation
    - for hypervisor
  - VMX non-root operation
    - controlled by hypervisor
    - supports VMs
- Both modes have ring 0-3
- guest can run in ring 0
- hypervisor said to be running in “ring -1”

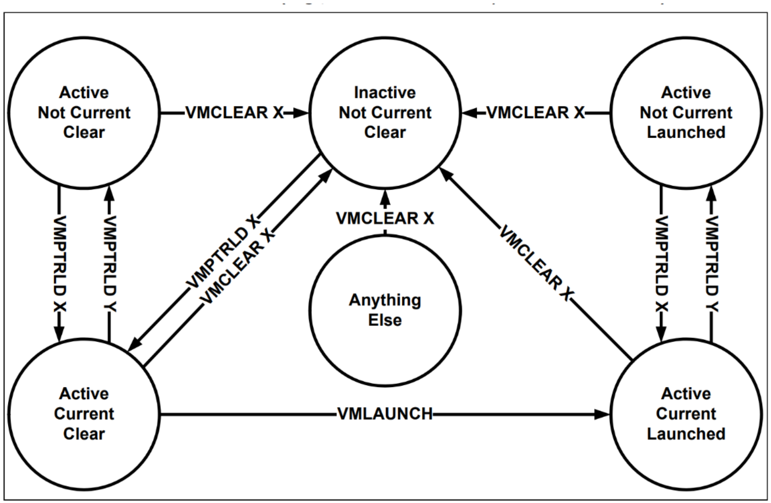
Rings on Intel



VMM Operation



VMM Transitions



VT-x

- VM entry: root operation $\rightarrow$ non-root operation

VT-x

- VM entry: root operation $\rightarrow$ non-root operation
- VM exit: non-root operation $\rightarrow$ root operation

VT-x

- VM entry: root operation $\rightarrow$ non-root operation
- VM exit: non-root operation $\rightarrow$ root operation
- VMCS: Virtual Machine Control Structure

VT-x

- VM entry: root operation $\rightarrow$ non-root operation
- VM exit: non-root operation $\rightarrow$ root operation
- VMCS: Virtual Machine Control Structure
  - Guest-state-area

VT-x

- VM entry: root operation $\rightarrow$ non-root operation
- VM exit: non-root operation $\rightarrow$ root operation
- VMCS: Virtual Machine Control Structure
    - Guest-state-area
    - Host-state-area

VT-x

- VM entry: root operation $\rightarrow$ non-root operation
- VM exit: non-root operation $\rightarrow$ root operation
- VMCS: Virtual Machine Control Structure
    - Guest-state-area
    - Host-state-area
- Entry/Exit loads/safes information using the proper area

VMCS - Guest State Area

- Contains elements comprising the state of the virtual CPU of a VMCS

VMCS - Guest State Area

- Contains elements comprising the state of the virtual CPU of a VMCS
- VM-exit requires loading certain registers (like segment registers, CR3, IRTR...)

VMCS - Guest State Area

- Contains elements comprising the state of the virtual CPU of a VMCS
- VM-exit requires loading certain registers (like segment registers, CR3, IRTR...)
- GSA contains fields for these registers

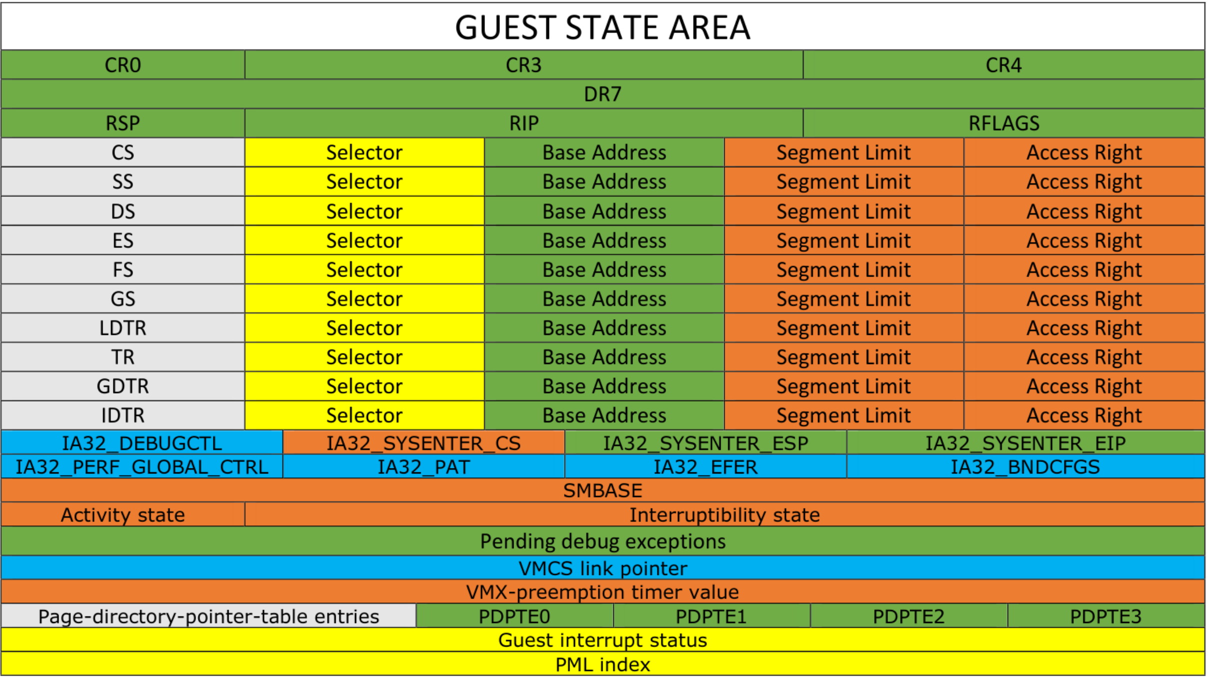
VMCS - Guest State Area

- Contains elements comprising the state of the virtual CPU of a VMCS
- VM-exit requires loading certain registers (like segment registers, CR3, IRTR...)
- GSA contains fields for these registers
- GSA contains fields for other information not readable via registers

VMCS - Guest State Area

- Contains elements comprising the state of the virtual CPU of a VMCS
- VM-exit requires loading certain registers (like segment registers, CR3, IRTR...)
- GSA contains fields for these registers
- GSA contains fields for other information not readable via registers
    - e.g. interruptability state

- 🟩 Natural-Width fields.
- 🟨 16-bits fields.
- 🟧 32-bits fields.
- 🟦 64-bits fields.



GUEST STATE AREA

CR0	CR3		CR4	
DR7				
RSP	RIP		RFLAGS	
CS	Selector	Base Address	Segment Limit	Access Right
SS	Selector	Base Address	Segment Limit	Access Right
DS	Selector	Base Address	Segment Limit	Access Right
ES	Selector	Base Address	Segment Limit	Access Right
FS	Selector	Base Address	Segment Limit	Access Right
GS	Selector	Base Address	Segment Limit	Access Right
LDTR	Selector	Base Address	Segment Limit	Access Right
TR	Selector	Base Address	Segment Limit	Access Right
GDTR	Selector	Base Address	Segment Limit	Access Right
IDTR	Selector	Base Address	Segment Limit	Access Right
IA32_DEBUGCTL	IA32_SYSENTER_CS	IA32_SYSENTER_ESP	IA32_SYSENTER_EIP	
IA32_PERF_GLOBAL_CTRL	IA32_PAT	IA32_EFER	IA32_BNDCFGS	
SMBASE				
Activity state	Interruptibility state			
Pending debug exceptions				
VMCS link pointer				
VMX-preemption timer value				
Page-directory-pointer-table entries	PDPTE0	PDPTE1	PDPTE2	PDPTE3
Guest interrupt status				
PML index				

HOST STATE AREA

CR0	CR3	CR4
RSP	RIP	
CS	Selector	
SS	Selector	
DS	Selector	
ES	Selector	
FS	Selector	Base Address
GS	Selector	Base Address
TR	Selector	Base Address
GDTR	Base Address	
IDTR	Base Address	
IA32_SYSENTER_CS	IA32_SYSENTER_ESP	IA32_SYSENTER_EIP
IA32_PERF_GLOBAL_CTRL	IA32_PAT	IA32_EFER

VMCS

- Addressed using physical addresses

VMCS

- Addressed using physical addresses
- not part of guest address space

VMCS

- Addressed using physical addresses
- not part of guest address space
- hypervisor may run in different address space as guest (CR3 part of state)

VMCS

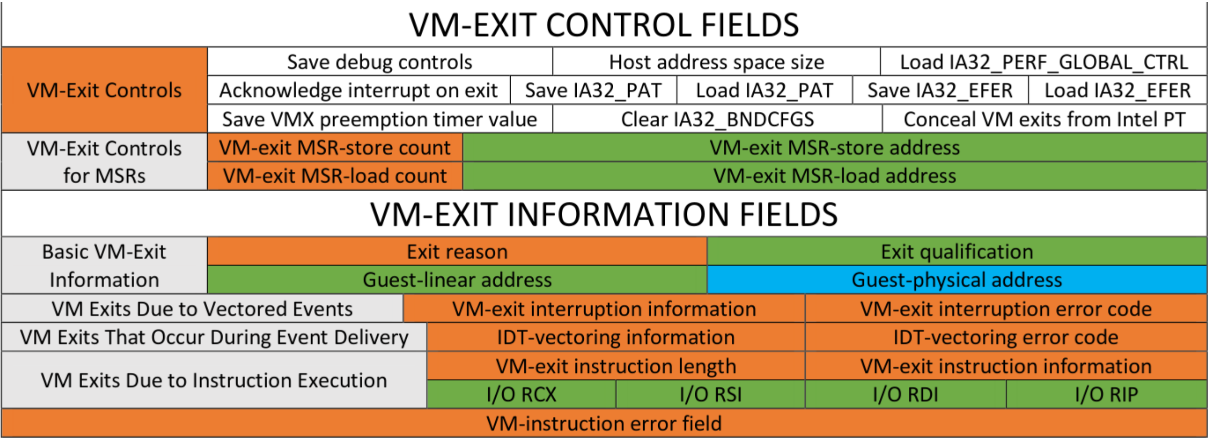
- Addressed using physical addresses
- not part of guest address space
- hypervisor may run in different address space as guest (CR3 part of state)
- VM-exits leave detailed information on reason for exit in VMCS

VMCS

- Addressed using physical addresses
- not part of guest address space
- hypervisor may run in different address space as guest (CR3 part of state)
- VM-exits leave detailed information on reason for exit in VMCS
    - exit reason

VMCS

- Addressed using physical addresses
- not part of guest address space
- hypervisor may run in different address space as guest (CR3 part of state)
- VM-exits leave detailed information on reason for exit in VMCS
  - exit reason
  - exit qualification



VM-EXIT CONTROL FIELDS

VM-Exit Controls	Save debug controls		Host address space size		Load IA32_PERF_GLOBAL_CTRL
	Acknowledge interrupt on exit	Save IA32_PAT	Load IA32_PAT	Save IA32_EFER	Load IA32_EFER
	Save VMX preemption timer value		Clear IA32_BNDCFGS		Conceal VM exits from Intel PT
VM-Exit Controls for MSRs	VM-exit MSR-store count	VM-exit MSR-store address			
	VM-exit MSR-load count	VM-exit MSR-load address			

VM-EXIT INFORMATION FIELDS

Basic VM-Exit Information	Exit reason		Exit qualification	
	Guest-linear address		Guest-physical address	
VM Exits Due to Vectored Events	VM-exit interruption information		VM-exit interruption error code	
VM Exits That Occur During Event Delivery	IDT-vectoring information		IDT-vectoring error code	
VM Exits Due to Instruction Execution	VM-exit instruction length		VM-exit instruction information	
	I/O RCX	I/O RSI	I/O RDI	I/O RIP
VM-instruction error field				

VMCS

- Example: MOV CR

VMCS

- Example: MOV CR
- Exit reason: “control register access”

VMCS

- Example: MOV CR
- Exit reason: “control register access”
- Exit qualification:

VMCS

- Example: MOV CR
- Exit reason: “control register access”
- Exit qualification:
    - which CR

VMCS

- Example: MOV CR
- Exit reason: “control register access”
- Exit qualification:
  - which CR
  - direction (Rx$\rightarrow$CR or CR$\rightarrow$Rx)

VMCS

- Example: MOV CR
- Exit reason: “control register access”
- Exit qualification:
    - which CR
    - direction (Rx→CR or CR→Rx)
    - register used

CONTROL FIELDS

Pin-Based VM-Execution Controls	External-interrupt exiting	NMI exiting	Virtual NMIs	
	Activate VMX-preemption timer	Process posted interrupts		
Primary processor-based VM-execution controls	Interrupt-window exiting	Use TSC offsetting		
	HLT exiting	INVLPG exiting	MWAIT exiting	RDPMC exiting
	RDTSC exiting	CR3-load exiting	CR3-store exiting	CR8-load exiting
	CR8-store exiting	Use TPR shadow	NMI-window exiting	MOV-DR exiting
	Unconditional I/O exiting	Use I/O bitmaps	Monitor trap flag	Use MSR bitmaps
	MONITOR exiting	PAUSE exiting	Activate secondary controls	
Secondary processor-based VM-execution controls	Virtualize APIC accesses	Enable EPT	Descriptor-table exiting	Enable RDTSCP
	Virtualize x2APIC mode	Enable VPID	WBINVD exiting	Unrestricted guest
	APIC-register virtualization	Virtual-interrupt delivery	PAUSE-loop exiting	
	RDRAND exiting	Enable INVPCID	Enable VM functions	VMCS shadowing
	Enable ENCLS exiting	RDSEED exiting	Enable PML	EPT-violation #VE
	Conceal VMX non-root operation from Intel PT	Enable XSAVES/XRSTORS		
	Mode-based execute control for EPT	Use TSC scaling		
Exception Bitmap	I/O-Bitmap Addresses	TSC-offset		
Guest/Host Masks for CR0	Guest/Host Masks for CR4	Read Shadows for CR0	Read Shadows for CR4	
CR3-target value 0	CR3-target value 1	CR3-target value 2	CR3-target value 3	CR3-target count
APIC Virtualization	APIC-access address	Virtual-APIC address	TPR threshold	
	EOI-exit bitmap 0	EOI-exit bitmap 1	EOI-exit bitmap 2	EOI-exit bitmap 3
	Posted-interrupt notification vector	Posted-interrupt descriptor address		
Read bitmap for low MSRs	Read bitmap for high MSRs	Write bitmap for low MSRs	Write bitmap for low MSRs	
Executive-VMCS Pointer	Extended-Page-Table Pointer	Virtual-Processor Identifier		
PLE_Gap	PLE_Window	VM-function controls	VMREAD bitmap	VMWRITE bitmap
ENCLS-exiting bitmap	PML address			
Virtualization-exception information address	EPTP index	XSS-exiting bitmap		

Virtualization: Trend 2000s $\rightarrow$ 2010s

The next step ($\approx$ 2005):

- Virtualization Hardware Extensions for Intel and AMD

Virtualization: Trend 2000s $\rightarrow$ 2010s

The next step ($\approx$ 2005):

- Virtualization Hardware Extensions for Intel and AMD

$\rightarrow$ substantially lower overheads for VMs

Virtualization: Trend 2000s $\rightarrow$ 2010s

The next step ($\approx$ 2005):

- Virtualization Hardware Extensions for Intel and AMD

$\rightarrow$ substantially lower overheads for VMs

$\rightarrow$ better isolation

Virtualization: Trend 2000s $\rightarrow$ 2010s

The next step ($\approx$ 2005):

- Virtualization Hardware Extensions for Intel and AMD

$\rightarrow$ substantially lower overheads for VMs

$\rightarrow$ better isolation

$\rightarrow$ IaaS VMs become widely used

Hardware Exentions: Intel VT

- Support for interrupt-virtualization

Hardware Exentions: Intel VT

- Support for interrupt-virtualization
  - VM-exit with every external interrupt (cannot be masked by guest)

Hardware Exentions: Intel VT

- Support for interrupt-virtualization
  - VM-exit with every external interrupt (cannot be masked by guest)
  - VM-exit when guest-OS ready to accept interrupts (EFLAGS.IF==1)

Hardware Exentions: Intel VT

- Support for interrupt-virtualization
    - VM-exit with every external interrupt (cannot be masked by guest)
    - VM-exit when guest-OS ready to accept interrupts (EFLAGS.IF==1)
- Support for CR0 and CR4-virtualization

Hardware Exentions: Intel VT

- Support for interrupt-virtualization
  - VM-exit with every external interrupt (cannot be masked by guest)
  - VM-exit when guest-OS ready to accept interrupts (EFLAGS.IF==1)
- Support for CR0 and CR4-virtualization
  - VM-exit with any change of these registers

Hardware Exentions: Intel VT

- Support for interrupt-virtualization
    - VM-exit with every external interrupt (cannot be masked by guest)
    - VM-exit when guest-OS ready to accept interrupts (EFLAGS.IF==1)
- Support for CR0 and CR4-virtualization
    - VM-exit with any change of these registers
    - can be set on which bits this shall happen

Solves our problems

- Address Space Compression

Solves our problems

- Address Space Compression
    - change of address space with any switch guest/hypervisor

Solves our problems

- Address Space Compression
    - change of address space with any switch guest/hypervisor
    - guest owns full virtual address space

Solves our problems

- Address Space Compression
    - change of address space with any switch guest/hypervisor
    - guest owns full virtual address space
- Ring Problems, SYSENTER/SYSEXIT

Solves our problems

- Address Space Compression
  - change of address space with any switch guest/hypervisor
  - guest owns full virtual address space
- Ring Problems, SYSENTER/SYSEXIT
  - Guest can now run in ring 0

Solves our problems

- Non-faulting Access to Privileged State

Solves our problems

- Non-faulting Access to Privileged State
    - access raise fault into hypervisor

Solves our problems

- Non-faulting Access to Privileged State
    - access raise fault into hypervisor
- Hidden State

Solves our problems

- Non-faulting Access to Privileged State
  - access raise fault into hypervisor
- Hidden State
  - Saved into VMCS

Hypervisor and Virtual Memory

- Hypervisor uses virtual memory

Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory

Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports page tables

Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports page tables
- how does this work?

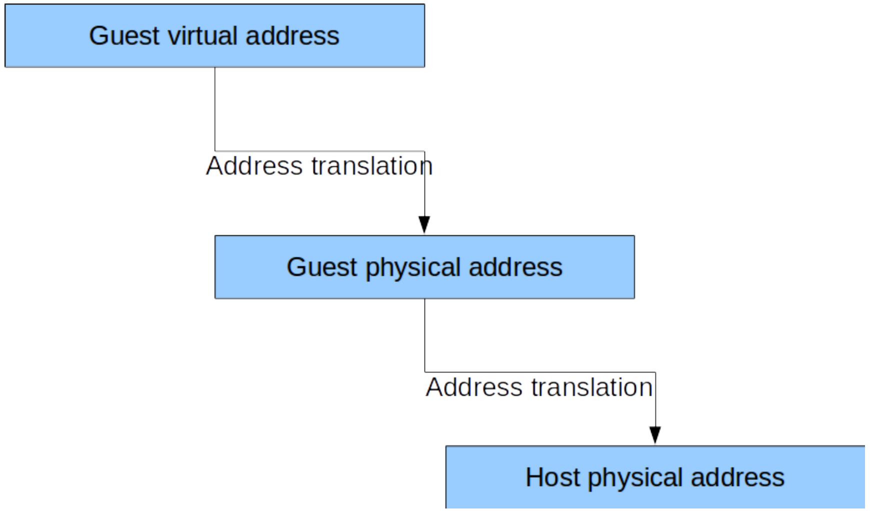
Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports page tables
- how does this work?
    - shadow page tables

Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports page tables
- how does this work?
    - shadow page tables
    - hardware support

Virtual Memory



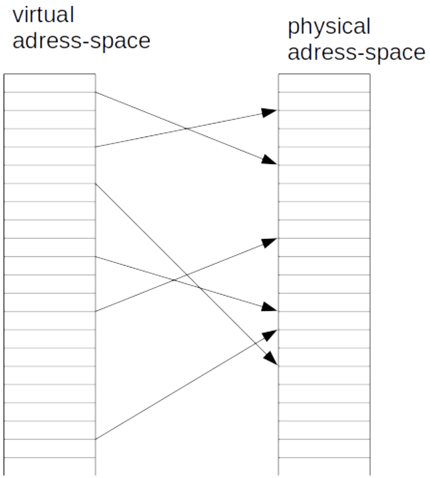
All problems in computer science can be solved by another level of indirection.

All problems in computer science can be solved by another level of indirection.

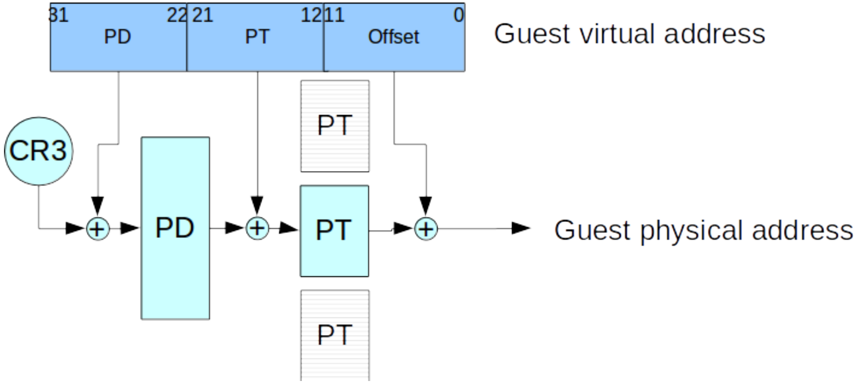
But that usually will create another problem.

David Wheeler

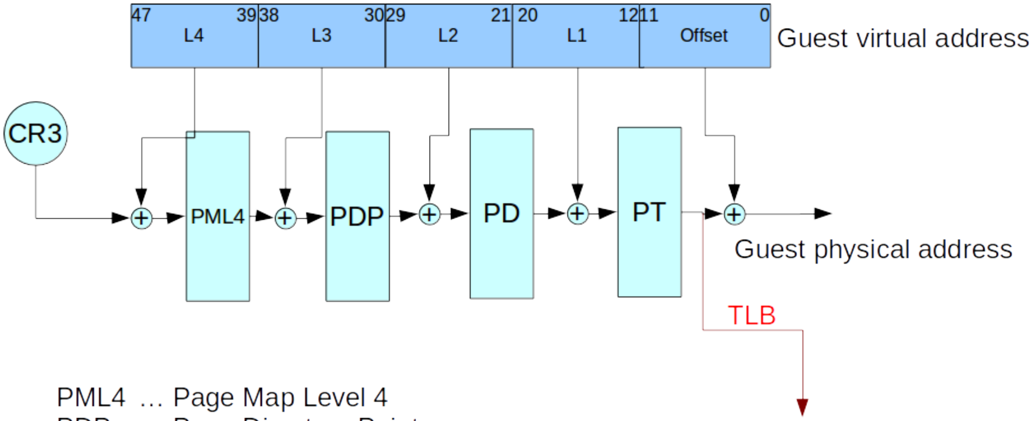
Paging



Known from theory and practice

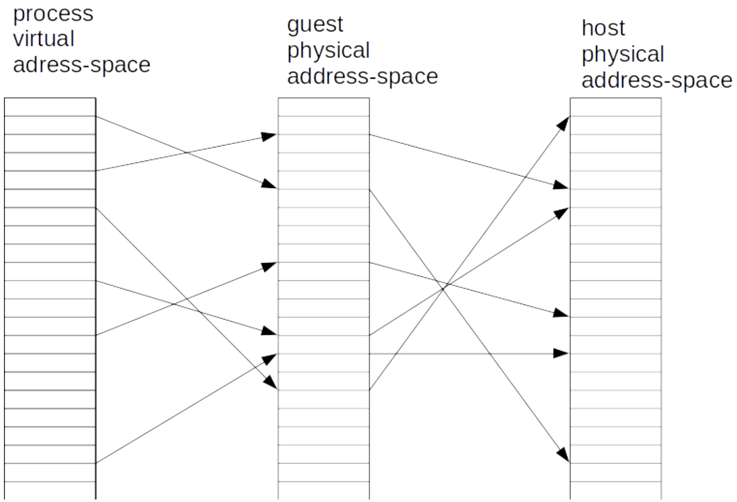


and in 64 bit...

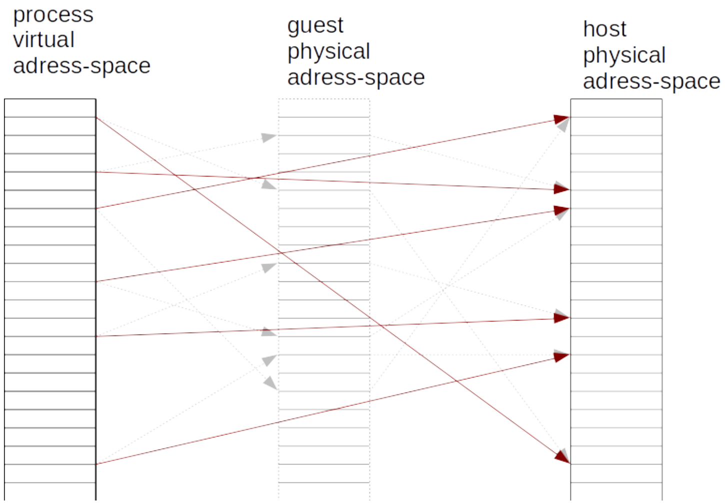


PML4	… Page Map Level 4
PDP	… Page Directory Pointer
PD	… Page Directory
PT	… Page Table

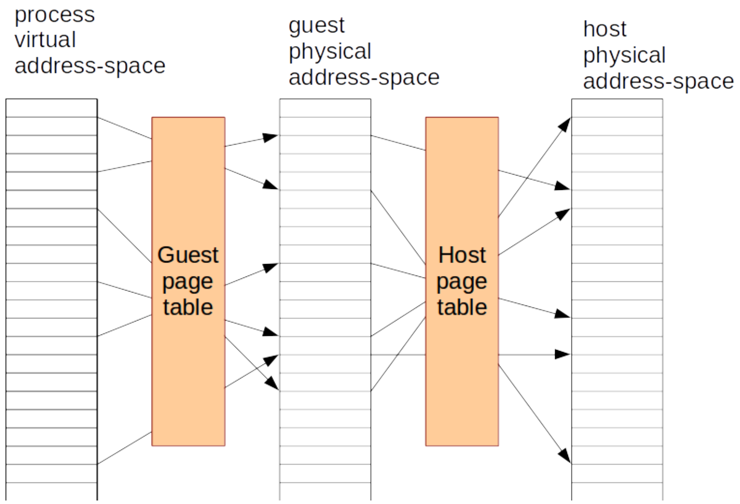
Combined Paging



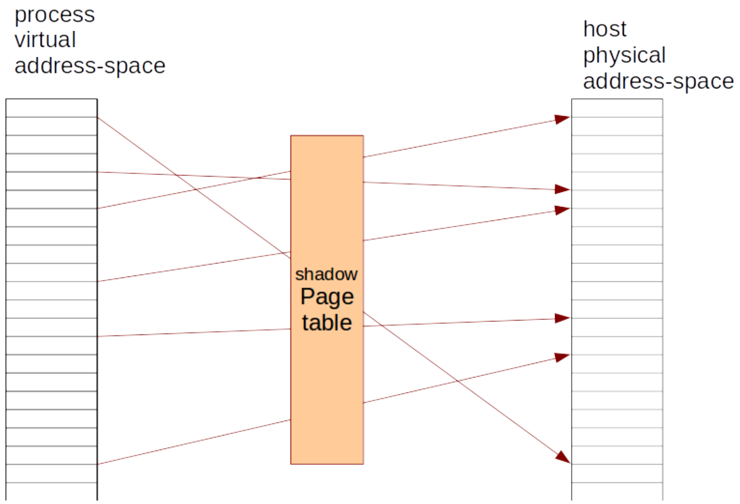
Shadow Page Table



Page Tables


process
virtual
address-space
guest
physical
address-space
host
physical
address-space
Guest
page
table
Host
page
table

Shadow Page Table



Shadow Page Table

- **merges** both page tables into one that the HW uses

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table
  - Hypervisor has to catch access

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table
  - Hypervisor has to catch access
  - update shadow page table

Shadow Page Table

- when HW changes shadow page table

Shadow Page Table

- when HW changes shadow page table
- update guest PT

Shadow Page Table

- when HW changes shadow page table
- update guest PT
    - expensive!

Shadow Page Table

- when HW changes shadow page table
- update guest PT
  - expensive!
  - page faults caught by hypervisor

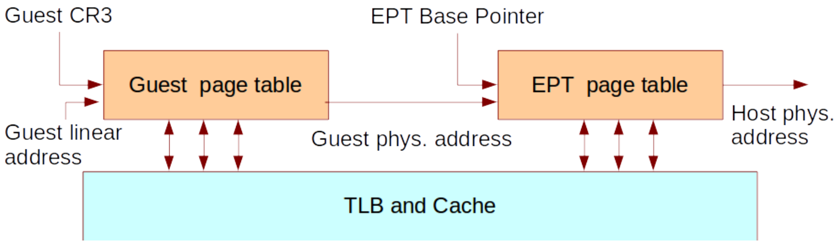
Shadow Page Table

- when HW changes shadow page table
- update guest PT
    - expensive!
    - page faults caught by hypervisor
    - must run through guest PTs

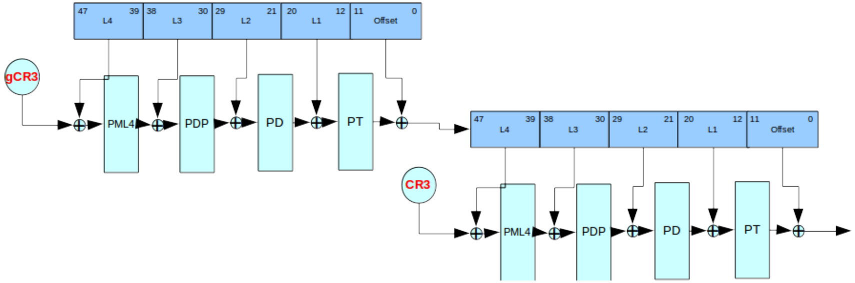
Shadow Page Table

- when HW changes shadow page table
- update guest PT
    - expensive!
    - page faults caught by hypervisor
    - must run through guest PTs
    - must emulate accessed and modified bits for guest

EPT


Guest CR3
EPT Base Pointer
Guest page table
EPT page table
Guest linear address
Guest phys. address
Host phys. address
TLB and Cache

Nested PT (NPT, AMD) / Extended PT (EPT, Intel)



“guest page walk”

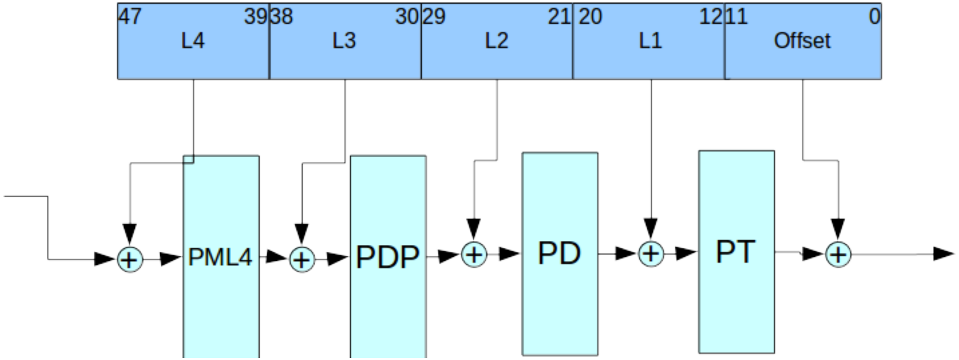
Guest page walk

- lots of memory accesses....

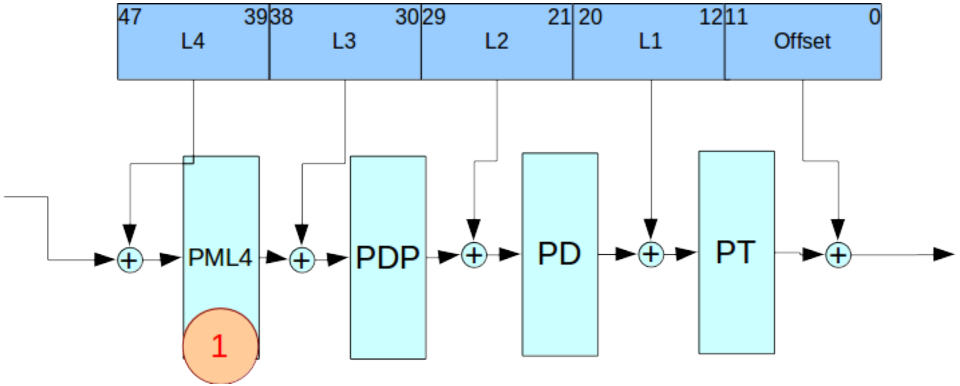
Guest page walk

- lots of memory accesses....
- but how many exactly?

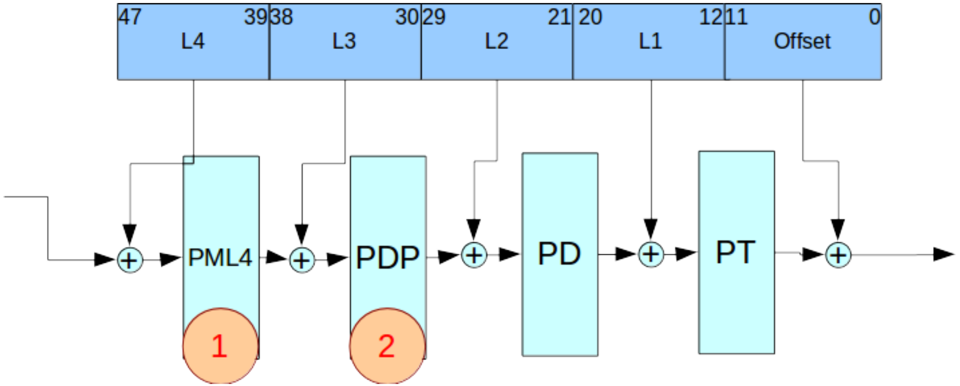
third for the host



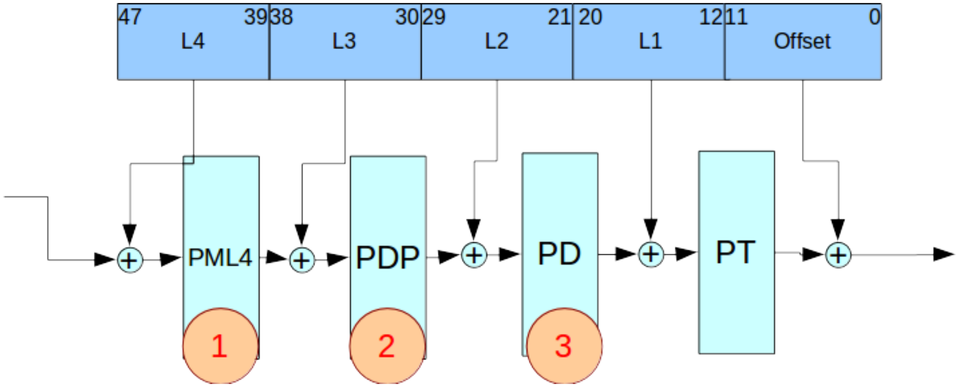
third for the host



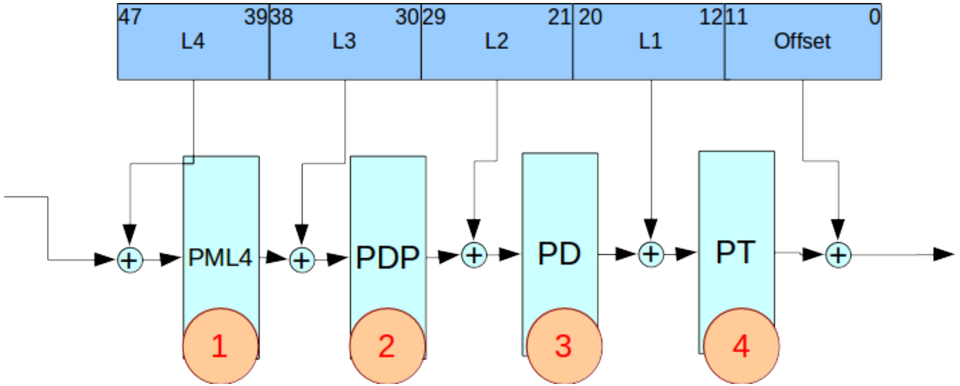
third for the host



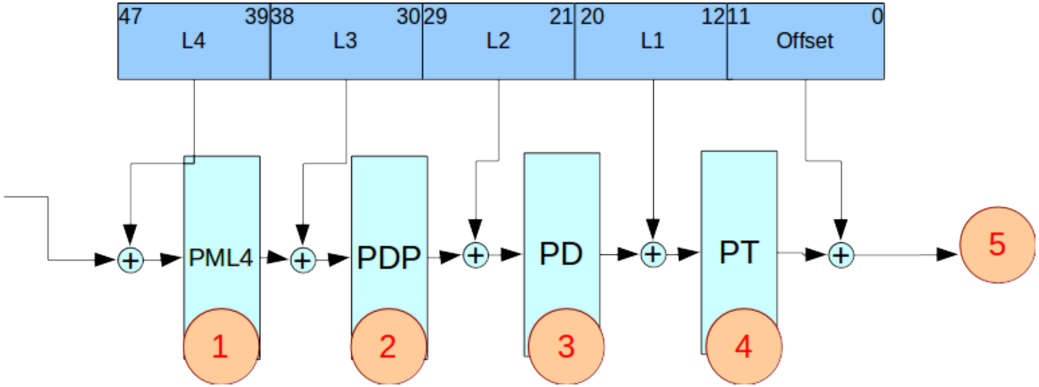
third for the host



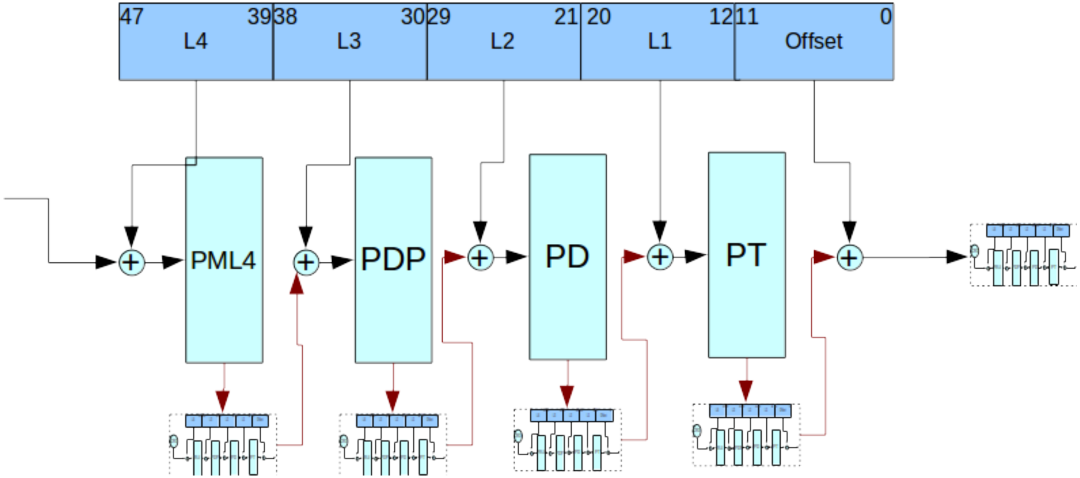
third for the host



third for the host



And Combined



… and combined …

max. number of memory accesses per address translation

- 5 on guest level

… and combined …

max. number of memory accesses per address translation

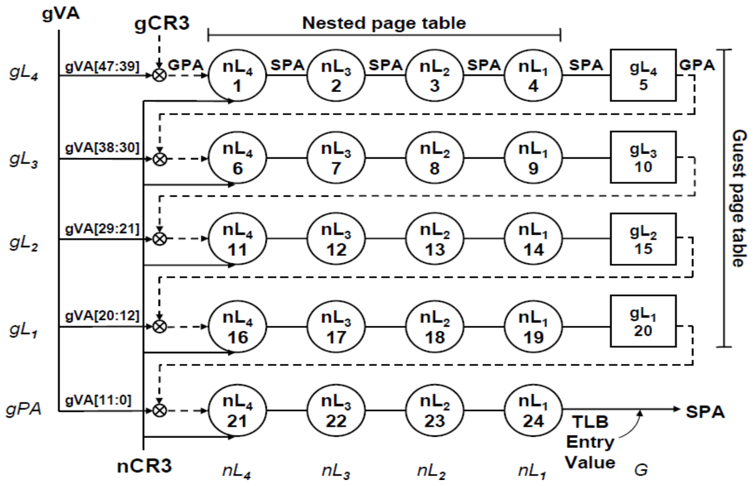
- 5 on guest level
- each induces 5 on host level

… and combined …

max. number of memory accesses per address translation

- 5 on guest level
- each induces 5 on host level
- makes 25!

Guest Page Walk



Performance

- depending on application: 3.9-4.6 times slower

Performance

- depending on application: 3.9-4.6 times slower
- but: TLB

Intel Features

- EPT only used if VM active

Intel Features

- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer

Intel Features

- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer
  - differentiate TLB-entries of different VMs

Intel Features

- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer
  - differentiate TLB-entries of different VMs
  - TLB-flush per guest possible

Intel Features

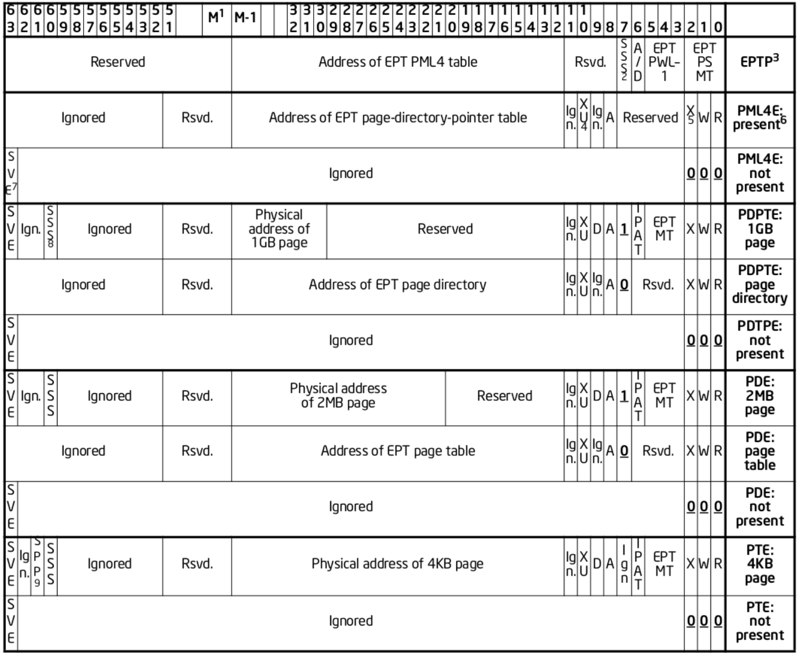
- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer
  - differentiate TLB-entries of different VMs
  - TLB-flush per guest possible
- VPID: virtual processor ID

Intel Features

- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer
    - differentiate TLB-entries of different VMs
    - TLB-flush per guest possible
- VPID: virtual processor ID
    - unique value for each VM

Intel Features

- EPT only used if VM active
- Translations tagged in TLB with EPT base pointer
    - differentiate TLB-entries of different VMs
    - TLB-flush per guest possible
- VPID: virtual processor ID
    - unique value for each VM
    - translations tagged in TLB using VPID



63 62 61 60 59 58 57 56 55 54 53 52 51 M1 M-1 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved | Address of EPT PML4 table | Rsvd. | SSS2 | A/D | EPT PWL-1 | EPT PS MT | EPTP3
Ignored | Rsvd. | Address of EPT page-directory-pointer table | Ign. | XU4 | Ign. | A | Reserved | XS | W | R | PML4E: present6
SVE7 | Ignored | 0 | 0 | 0 | PML4E: not present
SVE | Ign. | SSS8 | Ignored | Rsvd. | Physical address of 1GB page | Reserved | Ign. | XU | D | A | 1 | IPAT | EPT MT | X | W | R | PDPTE: 1GB page
Ignored | Rsvd. | Address of EPT page directory | Ign. | XU | Ign. | A | 0 | Rsvd. | X | W | R | PDPTE: page directory
SVE | Ignored | 0 | 0 | 0 | PDTPE: not present
SVE | Ign. | SSS | Ignored | Rsvd. | Physical address of 2MB page | Reserved | Ign. | XU | D | A | 1 | IPAT | EPT MT | X | W | R | PDE: 2MB page
Ignored | Rsvd. | Address of EPT page table | Ign. | XU | Ign. | A | 0 | Rsvd. | X | W | R | PDE: page table
SVE | Ignored | 0 | 0 | 0 | PDE: not present
SVE | Ign. | SPP9 | SSS | Ignored | Rsvd. | Physical address of 4KB page | Ign. | XU | D | A | Ign | IPAT | EPT MT | X | W | R | PTE: 4KB page
SVE | Ignored | 0 | 0 | 0 | PTE: not present

Figure 28-1. Formats of EPTP and EPT Paging-Structure Entries

Setting up a hypervisor

1. Enable VMX via `CR4`

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction
3. Allocate an MSR Bitmap region (we don't want a trap for all MSRs)

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction
3. Allocate an MSR Bitmap region (we don't want a trap for all MSRs)
4. Use `VMCLEAR` instruction

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction
3. Allocate an MSR Bitmap region (we don't want a trap for all MSRs)
4. Use `VMCLEAR` instruction
5. Execute `VMPTRLD` to make a VMCS the "current VMCS"

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction
3. Allocate an MSR Bitmap region (we don't want a trap for all MSRs)
4. Use `VMCLEAR` instruction
5. Execute `VMPTRLD` to make a VMCS the "current VMCS"
6. Allocate a VMCS region and set up the VMCS (using `VMWRITE`s)

Setting up a hypervisor

1. Enable VMX via `CR4`
2. Allocate a `VMXON` region and use the `VMXON` instruction
3. Allocate an MSR Bitmap region (we don't want a trap for all MSRs)
4. Use `VMCLEAR` instruction
5. Execute `VMPTRLD` to make a VMCS the "current VMCS"
6. Allocate a VMCS region and set up the VMCS (using `VMWRITE`s)
7. Use the `VMLAUNCH`

What if we can’t do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

What if we can’t do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

→ can use a syscall!

What if we can’t do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

→ can use a syscall!

2. What about VMs?

What if we can’t do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

$\rightarrow$ can use a syscall!

2. What about VMs?
3. Same concept different level:

What if we can't do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

$\rightarrow$ can use a syscall!

2. What about VMs?
3. Same concept different level:

$\rightarrow$ Hypercalls!

What if we can't do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

$\rightarrow$ can use a syscall!

2. What about VMs?
3. Same concept different level:

$\rightarrow$ Hypercalls!

What if we can't do something in the Guest?

Similar problem as with Userspace-Kernelspace Isolation:

1. user needs help for some operations (e.g., HW interaction)

→ can use a syscall!

2. What about VMs?
3. Same concept different level:

→ Hypercalls! via the `vmcall` instruction

Virtualization: Trend 2010s → 2022+

Optimization

- Full virtualization often not needed

Virtualization: Trend 2010s → 2022+

Optimization

- Full virtualization often not needed
- Serverless / Edge Computing (it's still a form of cloud computing)

Virtualization: Trend 2010s → 2022+

Optimization

- Full virtualization often not needed
- Serverless / Edge Computing (it's still a form of cloud computing)
- Virtualization is not for free → why not skip it and just use OS level isolation?

Virtualization: Trend 2010s → 2022+

Optimization

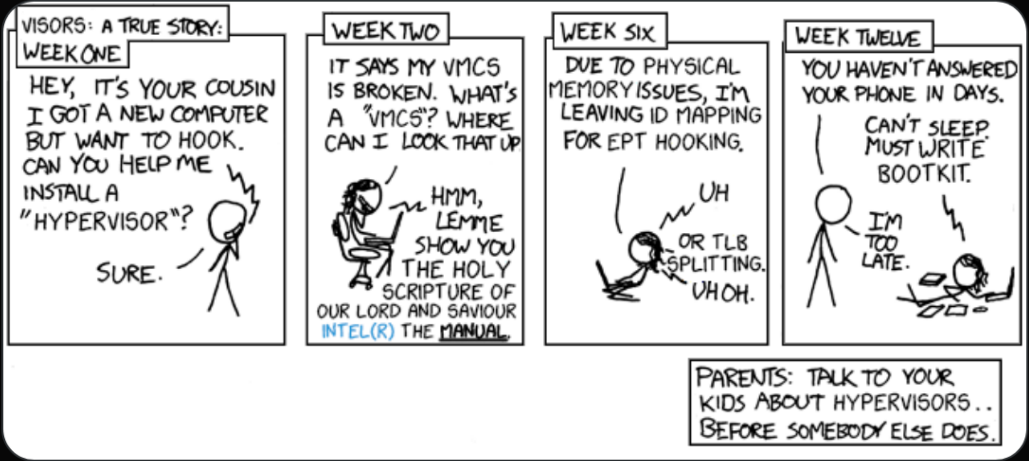
- Full virtualization often not needed
- Serverless / Edge Computing (it's still a form of cloud computing)
- Virtualization is not for free → why not skip it and just use OS level isolation?
- Context switches between processes are expensive → why not skip process isolation and just use language-level isolation?

This course: hardware-assisted virtualization

Cloud Operating Systems → Hardware-assisted virtualization

cts @gf_256 · 5. Apr. 2020

Talk to your kids about hypervisors...before someone else does


VISORS: A TRUE STORY:
WEEK ONE
HEY, IT'S YOUR COUSIN I GOT A NEW COMPUTER BUT WANT TO HOOK. CAN YOU HELP ME INSTALL A "HYPERVISOR"?
SURE.
WEEK TWO
IT SAYS MY VMCS IS BROKEN. WHAT'S A "VMCS"? WHERE CAN I LOOK THAT UP
HMM, LEMME SHOW YOU THE HOLY SCRIPTURE OF OUR LORD AND SAVIOUR INTEL(R) THE MANUAL.
WEEK SIX
DUE TO PHYSICAL MEMORY ISSUES, I'M LEAVING ID MAPPING FOR EPT HOOKING.
UH
OR TLB SPLITTING.
UH OH.
WEEK TWELVE
YOU HAVEN'T ANSWERED YOUR PHONE IN DAYS.
CAN'T SLEEP MUST WRITE BOOTKIT.
I'M TOO LATE.
PARENTS: TALK TO YOUR KIDS ABOUT HYPERVISORS.. BEFORE SOMEBODY ELSE DOES.

19 60

CloudOS: the third time

- Seminar-style

CloudOS: the third time

- Seminar-style
- You code

CloudOS: the third time

- Seminar-style
- You code
- You plan

CloudOS: the third time

- Seminar-style
- You code
- You plan
- You present

Team

Daniel Gruss, Andreas Kogler, Fabian Rauscher, Sudheendra Neela

Grading Scale

- 100 P. $=$ 100%

Grading Scale

- 100 P. $=$ 100%
- 87.5 P. $\rightarrow$ 1

Grading Scale

- 100 P. $=$ 100%
- 87.5 P. $\rightarrow$ 1
- 75 P. $\rightarrow$ 2

Grading Scale

- 100 P. $=$ 100%
- 87.5 P. $\rightarrow$ 1
- 75 P. $\rightarrow$ 2
- 62.5 P. $\rightarrow$ 3

Grading Scale

- 100 P. $=$ 100%
- 87.5 P. $\rightarrow$ 1
- 75 P. $\rightarrow$ 2
- 62.5 P. $\rightarrow$ 3
- 50 P. $\rightarrow$ 4

Teams

- 10 participants $\rightarrow$ 2-3 teams with each 4 participants (default)

Teams

- 10 participants $\rightarrow$ 2-3 teams with each 4 participants (default)
- 5 ECTS $=$ 500h with 125h per team member

Teams

- 10 participants $\rightarrow$ 2-3 teams with each 4 participants (default)
- 5 ECTS = 500h with 125h per team member
- Team of 3? Same effort but +5 points

Teams

- 10 participants $\rightarrow$ 2-3 teams with each 4 participants (default)
- 5 ECTS = 500h with 125h per team member
- Team of 3? Same effort but +5 points
- Team of 2? Same effort but +10 points

Teams

- 10 participants → 2-3 teams with each 4 participants (default)
- 5 ECTS = 500h with 125h per team member
- Team of 3? Same effort but +5 points
- Team of 2? Same effort but +10 points

→ send us your registration until Friday March 10

Teams

- Deadlines: Friday 23:59

Teams

- Deadlines: Friday 23:59
- Grace Period: 48 hours **but no support**

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. $\rightarrow$ AG1

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. → AG1
- 5.5. Interrupt + Emulate PIC + Public Feature Bidding
  Estimated Team Effort: 100h, Points: 5P.

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. $\rightarrow$ AG1
- 5.5. Interrupt + Emulate PIC + Public Feature Bidding
  Estimated Team Effort: 100h, Points: 5P.
- 26.5. Boot Guest SWEB Shell + Virtualize Disk + Private Feature Bidding
  Estimated Team Effort: 75h, Points: 35P. $\rightarrow$ AG2

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. $\rightarrow$ AG1
- 5.5. Interrupt + Emulate PIC + Public Feature Bidding
  Estimated Team Effort: 100h, Points: 5P.
- 26.5. Boot Guest SWEB Shell + Virtualize Disk + Private Feature Bidding
  Estimated Team Effort: 75h, Points: 35P. $\rightarrow$ AG2
- 2.6. Feature PoC in Booted Guest SWEB
  Estimated Team Effort: 75h, Points: 10P.

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. → AG1
- 5.5. Interrupt + Emulate PIC + Public Feature Bidding
  Estimated Team Effort: 100h, Points: 5P.
- 26.5. Boot Guest SWEB Shell + Virtualize Disk + Private Feature Bidding
  Estimated Team Effort: 75h, Points: 35P. → AG2
- 2.6. Feature PoC in Booted Guest SWEB
  Estimated Team Effort: 75h, Points: 10P.
- 16.6. Feature Implementation Done + Final Presentation and Demo in Booted Guest SWEB
  Estimated Team Effort: 75h, Points: 30P. → AG3

Timeline & Deadlines

- 24.3. Structure Setup
  Estimated Team Effort: 125h, Points: 5P.
- 21.4. Executing Guest Code + Video Output
  Estimated Team Effort: 125h, Points: 15P. $\rightarrow$ AG1
- 5.5. Interrupt + Emulate PIC + Public Feature Bidding
  Estimated Team Effort: 100h, Points: 5P.
- 26.5. Boot Guest SWEB Shell + Virtualize Disk + Private Feature Bidding
  Estimated Team Effort: 75h, Points: 35P. $\rightarrow$ AG2
- 2.6. Feature PoC in Booted Guest SWEB
  Estimated Team Effort: 75h, Points: 10P.
- 16.6. Feature Implementation Done + Final Presentation and Demo in Booted Guest SWEB
  Estimated Team Effort: 75h, Points: 30P. $\rightarrow$ AG3
- 16.6. Successful Live Presentation at 21:00, Bonus Points: 5P.